UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELÉTRICA
DEPARTAMENTO DE ENGENHARIA DE SISTEMAS

PROGRAMAÇÃO DA PRODUÇÃO EM UMA MÁQUINA COM CUSTOS DE AVANÇO E ATRASO EM RELAÇÃO A DATAS DE ENTREGA

RENATA MAZZINI

Orientador:
Prof. Dr. Vinícius Amaral Armentano

Tese apresentada à Faculdade de Engenharia Elétrica da Universidade Estadual de Campinas - UNICAMP como parte dos requisitos exigidos para a obtenção do título de MESTRE EM ENGENHARIA ELÉTRICA.

- MARÇO 1995 -

À minha Família

AGRADECIMENTOS

A todos os que colaboraram para a realização deste trabalho. Em especial:

- ao Professor Vinícius A. Armentano pela orientação dedicada e ajuda fundamental em todas as etapas desses trabalho;
- à minha família pelo interesse constante;
- aos professores e funcionários do DENSIS, pelo apoio;
- à Vitória e à Rossana, por saberem ouvir;
- à Regina e à Fran, pelas dicas em C;
- à Débora e à Cíntia, pelo carinho;
- ao Saulo e ao Eduardo, pela ajuda com o OSL;
- ao Walcir, por toda sua paciência;
- à Marcinha, por aturar os meus "cadê o Vinícius?";
- à CAPES pelo apoio financeiro;
- aos Amigos Daniele, Ronald, Alexandre e Guilherme, simplesmente por serem fantásticos;
- ao meu marido Ricardo, meu administrador de sistemas particular, por suportar minhas ausências e minha presença (houve horas que eu não sabia dizer o que era pior);
- à minha Mãe, por sua ternura e paciência, por estar sempre perto, mesmo quando distante.

RESUMO

Com a popularização do conceito de produção "Just in Time", surgiu um interesse generalizado pelo controle de geração de estoques, por parte dos pesquisadores da área de programação da produção, pois foi introduzida a noção de que esse controle é tão importante quanto cumprir compromissos relativos a datas de entrega contratadas. Entretanto, desde o início do século, altos níveis de estoques de produto acabado, ou em processamento, já representavam uma situação de risco financeiro. Neste trabalho estuda-se o problema de programação da produção que considera custos internos (estoques) e externos (clientes) envolvidos no processamento de tarefas por uma máquina. Esse problema consiste na minimização do avanço e atraso ponderados, com relação a datas de entrega. A resolução desse problema envolve não somente a determinação de uma seqüência de processamento, mas também dos instantes de início de todas as tarefas, inserindo-se, eventualmente, intervalos ociosos entre elas. Propõe-se um procedimento heurístico para a determinação do programa de produção desejado. São apresentados e analisados resultados computacionais de testes em instâncias com até 80 tarefas.

Palavras chave: Programação da produção em uma máquina; Avanço; Atraso; Heurística.

ABSTRACT

With the increasing popularity of the Just in Time production concept, researchers in the scheduling area became more interested in stock control, due to the notion that this control is as important as meeting the agreed delivery dates. However, since the beginning of this century, high stock levels of finished goods or work in process, have represented great financial risks. The scheduling problem, which considers internal (stocks) and external (costumers) costs involved in the processing of jobs in one machine, is studied in this work. This problem consists of minimizing weighted earliness and tardiness with respect to due dates. Solving this problem involves not only the search for a sequence of jobs to be processed, but the determination of all starting times and the insertion of idle times, when necessary. A heuristic procedure is proposed to give the aimed schedule. Computational results are presented and analyzed. The tests were made in instances with up to 80 jobs.

Key words: Single-machine scheduling; Earliness; Tardiness; Heuristics.

ÍNDICE

LISTA DE FIGURAS

LISTA DE DIAGRAMAS

LISTA DE TABELAS

INTRODUÇÃO

O setor de manufatura, atualmente, encontra-se em competição contínua a nível mundial. O mercado consumidor apresenta a tendência crescente de exigir melhor qualidade, maior diversidade de produtos, menores prazos de entrega e preços razoáveis. Para sobreviver dentro desse cenário, um determinado sistema de manufatura deve passar por uma reformulação dos conceitos envolvidos, não apenas com a produção em si, mas com seus aspectos gerenciais e organizacionais.

Um dos pontos mais destacados na literatura diz respeito à integração dos processos que caracterizam o sistema produtivo. O sistema deve funcionar como um todo harmônico e não simplesmente como um conjunto de partes independentes. Segundo os pesquisadores da área, essa integração facilitaria a resolução de um problema chave enfrentado pelos sistemas de manufatura nos dias de hoje: o planejamento da produção.

O problema de planejamento da produção abrange diferentes setores do sistema de manufatura e envolve um número muito grande de informações. Por esse motivo, o problema tem sido abordado de uma forma hierarquizada. As vantagens principais desse tipo de abordagem podem ser resumidas como:

- divisão do problema geral de planejamento em subproblemas menores e
- adequação dos níveis de tomada de decisão à própria hierarquia existente em uma empresa.

Mas, existem alguns riscos que aparecem, principalmente, quando a comunicação entre os diferentes níveis hierárquicos não é eficiente. É preciso que haja grande cuidado na definição exata dos níveis de decisão, assim como no estabelecimento do fluxo de informações, das responsabilidades quanto às decisões tomadas e das medidas de desempenho que avaliarão a qualidade dessas decisões.

A definição de níveis de hierarquia mais utilizada, atualmente, define três níveis principais. Eles são diferenciados entre si pelo nível de detalhe das informações usadas e pela abrangência temporal de suas decisões. O nível mais alto é responsável pela definição dos objetivos principais do sistema, que devem ser atingidos a longo prazo. Nesse nível são

feitas estimativas de demanda e de capacidade de forma agregada e também são definidas as políticas de aquisição e disposição de recursos para atender à demanda estimada. O segundo nível encarrega-se de planejar a utilização eficiente e eficaz dos recursos disponíveis, a médio prazo. As decisões tipicamente tomadas nesse nível dizem respeito a: utilização de mão-de-obra, alocação de capacidade agregada de recursos a famílias de produtos, acumulação de estoques, definição de canais de distribuição e seleção de alternativas de transporte. Esse planejamento é, então, passado para o último nível da hierarquia, onde são tomadas as decisões necessárias à programação detalhada do fluxo de materiais na planta do sistema, a curto prazo. Essa programação envolve a alocação de operações de produção a recursos disponíveis e o roteamento de cada produto através da planta. As atividades correspondentes a essa programação são conhecidas como "scheduling" da produção[1]. O conjunto das decisões tomadas nesse nível da hierarquia, ao longo de um intervalo de tempo, pode ser vital para que se cumpram satisfatoriamente os objetivos principais estabelecidos para o sistema.

O estudo da teoria de programação de produção começou na década de 1950 mas, apesar do grande desenvolvimento ocorrido até os dias atuais, observam-se poucas aplicações práticas dessa teoria nos sistemas de manufatura. Devido à estrutura combinatorial complexa apresentada pela maioria dos problemas, os pesquisadores, com frequência, introduzem simplificações nos modelos a fim de encontrar uma versão mais tratável dos mesmos. Uma conseqüência dessas simplificações é a não correspondência entre as situações estudadas e as situações reais encontradas na maioria dos sistemas.

Um exemplo de simplificação, freqüentemente adotada, diz respeito aos critérios de desempenho estudados. Em geral, usa-se apenas um critério de cada vez na avaliação das decisões e esse critério, quase sempre, corresponde a uma medida regular de desemprenho. Medidas regulares são aquelas em que o custo é uma função não decrescente dos instantes de término de processamento dos produtos dentro do sistema. Um exemplo desse tipo de medida é o atraso no término dos produtos em relação a suas datas de entrega ("tardiness"). A quantidade e a diversidade dos custos envolvidos na produção exige o estudo de problemas com múltiplos objetivos. Além disso, muitos desses custos correspondem a medidas não regulares de desempenho, como o avanço no término dos produtos ("earliness"), que implica em custos de estocagem. Nesse caso, o custo de estocagem é uma função decrescente dos instantes de término.

---

[1] Para evitar o uso da palavra "scheduling" , ela será substituída por programação da produção ou, simplesmente, programação, sempre quando não se comprometa o entendimento do texto.

Na última década, surgiram esforços no sentido de preencher essa lacuna existente entre a teoria e a prática de programação da produção. Esses esforços incluem a adição de novas restrições tecnológicas nos modelos matemáticos, otimização conjunta de vários critérios de desempenho e o abandono de algumas hipóteses presentes apenas em sistemas ideais. Muitas vezes, os modelos resultantes são de grande porte, e sua resolução por métodos exatos exige grande esforço computacional. Quanto mais próximo da realidade se torna o modelo, maior é a sua complexidade e mais tempo é requerido para encontrar sua solução exata. Por outro lado, a grande maioria das decisões de programação deve ser tomada num curto espaço de tempo, diariamente ou a cada turno de trabalho. O que se observa, então, é que a pesquisa tem se concentrado no desenvolvimento e aperfeiçoamento de métodos heurísticos de resolução para esses problemas. Em geral, esses métodos não garantem a obtenção de uma solução ótima. Mas, usando informações específicas sobre a estrutura do sistema de manufatura e as características matemáticas do modelo do problema, pode-se aproveitar os resultados teóricos existentes para criar heurísticas capazes de gerar soluções muito boas, tendo a vantagem da rapidez.

Um dos desafios impostos pelo mercado aos sistemas de manufatura é a confiabilidade em relação às datas de entrega pré-determinadas. Em geral, o atraso da produção é responsável pelo surgimento de custos adicionais, de natureza externa ao sistema, como multas contratuais e até a perda de clientes. Além disso, o atraso pode ser visto como um sintoma indicador da utilização errônea da capacidade disponível, causada pela falta de um controle preciso do fluxo de materiais ao longo das etapas de processamento. Grande parte da pesquisa realizada sobre esse assunto se volta para a minimização de medidas padrão, como o atraso total, o atraso médio e o atraso ponderado. Entretanto, esse tipo de abordagem pode levar a uma situação em que os níveis de estoque de produto acabado, ou em processamento, aumentem temerariamente.

Um interesse generalizado pelo controle e gerenciamento de estoques começou a surgir em meados da década de 1980. Isso se deveu à absorção de grande parte do mercado consumidor mundial por algumas indústrias japonesas, que aplicavam o conceito de produção "Just in Time". Contudo, desde o início deste século, já se desenhava um cenário industrial onde a manutenção de altos níveis de estoque representava mais um risco que um sinal de prosperidade. Na verdade, os custos de estocagem implicam na inatividade de parte do capital disponível, enquanto poderia estar sendo empregado de forma a tornar a produção mais eficiente, ou flexível.

Para englobar os aspectos internos (estoques) e externos (atrasos) dos custos de produção, pode-se modelar o problema de programação de produção de forma que se minimize tanto o avanço ("earliness") quanto o atraso ("tardiness") no término do

processamento de tarefas com relação às suas datas de entrega previstas. A solução desse modelo deve levar em consideração não apenas o sequenciamento das tarefas a serem processadas mas, também, a eventual inserção de intervalos ociosos entre as tarefas. Ignorar essa inserção é inconsistente com o uso de uma medida não regular de desempenho.

O objetivo desse trabalho é contribuir para o estudo de sistemas de manufatura onde existe um centro de trabalho crítico (ou máquina crítica) que represente um gargalo para o fluxo de produção. Apenas uma restrição é imposta a essa máquina: a de que somente uma tarefa pode ser processada de cada vez. Algumas hipóteses tradicionais são abandonadas na formulação do problema. Uma delas é a de que todas as tarefas estejam disponíveis para processamento no mesmo instante. Pensando na máquina gargalo como parte de um sistema mais complexo, o estabelecimento de instantes de liberação para processamento independentes para cada tarefa (ou operação) parece mais consistente. Além disso, cada tarefa possui uma data de término prevista, ou data de entrega, independente das demais tarefas. Tenta-se, dessa forma, abranger o problema de minimização do atraso e avanço ponderados, problema "Earliness/Tardiness" (E/T), na sua forma mais geral.

Após a determinação do modelo matemático do problema e o estudo dos trabalhos já realizados sobre esse mesmo tema, espera-se chegar a um procedimento heurístico que construa uma seqüência de processamento e determine os instantes de início de cada tarefa.

A seguir é feita uma apresentação sucinta da organização dos capítulos que se seguem.

O Capítulo 1 consiste na apresentação formal do problema E/T. Descreve-se a notação utilizada durante esse trabalho e a formulação de programação inteira mista correspondente ao modelo do problema.

No Capítulo 2, faz-se uma revisão bibliográfica relativa ao problema E/T.

No Capítulo 3, apresenta-se uma descrição detalhada da heurística proposta, acompanhada da resolução de um exemplo que tem por finalidade ilustrar os procedimentos envolvidos.

O Capítulo 4 contém a descrição da estratégia de geração dos problemas de teste e os resultados computacionais obtidos. Para problemas pequenos, com 8 tarefas, realiza-se uma comparação entre a heurística e um algoritmo de "Branch & Bound". Para problemas maiores, comparam-se os resultados heurísticos em relação à melhora obtida pela aplicação de uma busca na vizinhança das soluções encontradas. Faz-se, então, uma análise desses resultados e do desempenho da heurística.

CAPÍTULO 1
APRESENTAÇÃO DO PROBLEMA

Neste capítulo, apresenta-se uma descrição formal do problema E/T e a terminologia usada nesse trabalho.

Seja $\mathbf{N} = \{1, 2, \ldots, n\}$ o conjunto de índices das tarefas que devem ser processadas na única máquina disponível. A cada tarefa J_i, $i \in \mathbf{N}$, são associados os seguintes parâmetros:

- p_i - tempo de processamento;
- r_i - instante de liberação para processamento;
- d_i - data de entrega;
- s_i - instante de início de processamento da tarefa na máquina;
- c_i - instante de término de processamento da tarefa;
- h_i - custo (penalidade) por unidade de tempo de avanço no processamento;
- t_i - custo (penalidade) por unidade de tempo de atraso no processamento.

O objetivo da programação é minimizar o custo total representado pela soma ponderada dos avanços e atrasos ocorridos no processamento das tarefas de **N**. As medidas de desempenho utilizadas são definidas matematicamente, em função de cada tarefa, como:

- $E_i = \max\{0, d_i - c_i\}$ - avanço no término de J_i, em relação a sua data de entrega;
- $T_i = \max\{0, c_i - d_i\}$ - atraso no término de J_i, em relação a sua data de entrega;

A função a ser minimizada pela programação, então:

$$Z = \sum_{i=1}^{n} (h_i . E_i + t_i . T_i) \qquad (1.1)$$

As restrições tecnológicas impostas para a máquina e as tarefas podem ser formuladas da seguinte forma:

$$s_i - s_j \geq p_j \text{ ou } s_j - s_i \geq p_i \quad \forall i, j \in \mathbf{N},\ i \neq j \qquad (1.2)$$

$$s_i \geq r_i \quad \forall i \in \mathbf{N} \qquad (1.3)$$

O problema de minimização de Z pode ser modelado como:

$$\text{Minimizar } Z = \sum_{i=1}^{n} (h_i . E_i + t_i . T_i) \qquad (1.4a)$$

sujeito a

$$s_i - s_j \geq p_j \text{ ou } s_j - s_i \geq p_i \quad \forall i, j \in \mathbf{N},\ i \neq j \qquad (1.4b)$$

$$s_i \geq r_i \quad \forall i \in \mathbf{N} \qquad (1.4c)$$

Dada qualquer das $n!$ seqüências de processamento possíveis, pode-se encontrar um programa de produção factível, determinando os instantes de início de cada tarefa de acordo com as expressões (1.4a) e (1.4b). Para uma determinada seqüência de processamento, a minimização de Z pode implicar na inserção de intervalos ociosos no programa de produção.

Para apresentar a formulação de Programação Linear Inteira Mista correspondente ao modelo acima é preciso criar a seguinte variável auxiliar:

x_{ij} - variável que indica precedência imediata na seqüência de processamento.

Se a tarefa J_i precede imediatamente a tarefa J_j na solução, $x_{ij} = 1$. Caso contrário, $x_{ij} = 0$.

Seja M um número suficientemente grande.

O modelo (1.4) pode, agora, ser formulado como:

$$\text{Minimizar } \sum_{i=1}^{n} (h_i . E_i + t_i . T_i) \qquad (1.5a)$$

sujeito a

$$T_i - (s_i + p_i - d_i) \geq 0,\ i = 1, .., n \qquad (1.5b)$$

$$E_i - (d_i - s_i - p_i) \geq 0,\ i = 1, .., n \qquad (1.5c)$$

$$s_i \geq s_j + p_j - M.(1 - x_{ji}),\ i = 1, .. n;\ j = 1, .., n;\ i \neq j \qquad (1.5d)$$

$$\sum_{j=1}^{n+1} x_{ij} = 1,\ i = 0, .., n;\ i \neq j \qquad (1.5e)$$

$$\sum_{i=0}^{n} x_{ij} = 1,\ j = 1, .., n+1;\ j \neq i \qquad (1.5f)$$

$$\left.\begin{array}{l} T_i \geq 0,\ i = 1,..,n \\ E_i \geq 0,\ i = 1,..,n \\ s_i \geq r_i,\ i = 1,..,n \\ x_{ij} \in \{0,1\},\ i = 1,..,n+1; \\ \quad j = 0,..,n;\ i \neq j \end{array}\right\} \qquad (1.5g)$$

As restrições (1.5a), (1.5b) e (1.5c) correspondem à linearização da função objetivo Z. A restrição (1.5d) traduz matematicamente a restrição (1.4b). Basicamente, quando a tarefa J_j precede imediatamente a tarefa J_i, $x_{ji} = 1$ e $x_{ij} = 0$. Portanto:

$$s_i \geq s_j + p_j \text{ e } s_j \geq s_i + p_i - M. \qquad (1.6)$$

Qualquer outro posicionamento relativo, faz com que $x_{ji} = 0$ e $x_{ij} = 0$ e, portanto:

$$s_i \geq s_j + p_j - M \text{ e } s_j \geq s_i + p_i - M. \qquad (1.7)$$

As desigualdades (1.6) e (1.7), sozinhas, não garantem a não sobreposição de processamento de duas tarefas. Essa garantia é obtida através das restrições (1.5e) e (1.5f), que obrigam cada tarefa a ter um predecessor e um sucessor. As tarefas fictícias J_0 e $J_{(n+1)}$ foram criadas para prover um predecessor para a primeira tarefa do programa e um sucessor para a última tarefa da seqüência.

O conjunto de restrições (1.5g) estabelece a região de factibilidade do problema em relação às variáveis E_i, T_i, s_i e x_{ij}, para $\forall i, j \in \mathbf{N}$.

Tendo apresentado o modelo do problema e a terminologia a ser utilizada, passa-se agora para a revisão bibliográfica referente ao problema E/T.

CAPÍTULO 2
REVISÃO BIBLIOGRÁFICA

Neste capítulo, procura-se apresentar como o problema E/T vem sendo tratado pelos pesquisadores da área de Programação da Produção. A pesquisa bibliográfica abrange o período de 1987 a 1994 e teve as seguintes fontes: "Engineering Index", "Computer & Control Abstracts" e "International Abstracts in Operations Research". A escolha do período de pesquisa se justifica pela grande concentração de trabalhos relacionados ao problema apresentado a partir do surgimento do interesse pelo estudos de ambientes "Just in Time".

Uma revisão bastante ampla dos trabalhos relacionados ao problema E/T é dada em BAKER e SCUDDER (1990), onde os autores procuraram reunir os principais resultados disponíveis até aquela data. Também são mostradas as várias formas em que o problema geral pode ser abordado, dependendo das hipóteses feitas pelo pesquisador. A grande maioria dos artigos citados corresponde ao estudo de programas de produção estáticos, ou seja, todo o conjunto de tarefas é conhecido "a priori" e está simultaneamente disponível para processamento. Além disso, são poucos os trabalhos que consideram intervalos ociosos incluídos na seqüência de processamento, quando as datas de entrega de cada tarefa são distintas.

Ao longo da análise dos artigos consultados, observou-se que existem duas linhas de estudo bem definidas. Uma delas engloba problemas de minimização de avanço e atraso quando existe uma data de entrega comum para todas as tarefas. A outra, considera que cada tarefa possui uma data de entrega distinta das demais. Torna-se, então, conveniente dividir a apresentação dos trabalhos em duas categorias conforme seja a linha adotada pelo autor.

2.1 - PROBLEMAS COM DATA DE ENTREGA COMUM

Nesse tipo de problema, não é interessante considerar intervalos ociosos inseridos na seqüência, pois prova-se que a solução ótima, nesse caso, corresponde a um programa em que as tarefas são processadas continuamente (BAKER e SCUDDER, 1990; SUNG e JOO, 1992; DE, GOSH e WELLS, 1993; KAHLBACHER, 1993). Deve-se salientar que, em todos os trabalhos consultados, consideram-se somente problemas em que todas as tarefas estão disponíveis para processamento no mesmo instante, ou seja $r_i = 0, \forall i \in \mathbf{N}$.

Dado um valor para a data de entrega e a seqüência ótima correspondente, o problema de minimização com data de entrega comum pode assumir duas formas:

- problema irrestrito: aumentando-se o valor da data de entrega, a seqüência ótima se mantém e é possível adicionar um intervalo ocioso no início dessa seqüência de forma que o custo ótimo continue constante;
- problema restrito: um aumento no valor da data de entrega pode provocar uma diminuição do custo de processamento e/ou uma modificação na seqüência ótima.

O menor valor da data de entrega a partir do qual se considera o problema como irrestrito, é chamado de data ótima.

BAGCHI, CHANG e SULLIVAN (1987) estudaram o problema E/T com dois objetivos diferentes: minimização dos custos de avanço e atraso absolutos e minimização dos custos quadráticos. Para cada um dos objetivos, o problema foi estudado na sua forma restrita e irrestrita. Foram apresentados, ainda, procedimentos exatos de solução baseados em técnicas de "Branch & Bound" e em propriedades de dominância da seqüência ótima. Essas propriedades permitem reduzir o conjunto de soluções sobre o qual se realiza a busca pela solução ótima, tornando essa busca mais rápida. Os procedimentos foram testados para instâncias com, no máximo, 12 tarefas e apresentaram um custo computacional alto no que diz respeito ao número de nós explorados.

Um estudo detalhado sobre o problema E/T com data de entrega comum, encontra-se em HALL e POSNER (1991) e HALL, KUBIAK e SETHI (1991), onde são consideradas as formas restrita e irrestrita do problema, respectivamente. Nesses dois trabalhos são analisadas condições para que uma seqüência seja ótima. Também é feita uma prova de que o problema é NP-difícil, tanto para o caso restrito como para o caso irrestrito.

Em HALL e POSNER (1991), foi apresentada uma formulação de programação dinâmica para o caso irrestrito. O algoritmo apresentou-se bastante eficiente para problemas de teste com até 1000 tarefas. Em HALL, KUBIAK e SETHI (1991), também foi usada uma formulação de programação dinâmica para construção de um algoritmo de

solução. Entretanto, os experimentos computacionais demonstraram que, para o problema restrito, a eficiência da programação dinâmica diminui.

Ao contrário de propor um procedimento de solução para problemas específicos, DE, GHOSH e WELLS (1991) demonstraram uma propriedade segundo a qual é possível determinar a data de entrega ótima quando se conhece todos os demais parâmetros do problema. Quando a data de entrega de um conjunto de tarefas é maior que ou igual à data ótima, o problema se torna irrestrito. Isso serviu como base para um algoritmo de programação dinâmica que contém subrotinas específicas conforme se verifique que o problema é restrito ou irrestrito. Um experimento computacional mostrou que, quando o problema é restrito, o algoritmo perde eficiência, do ponto de vista de tempo computacional. Diante disso, os autores consideraram um procedimento aproximado, garantindo encontrar uma solução dentro de uma fração ε da solução ótima desconhecida.

DE, GHOSH e WELLS (1993) apresentaram uma extensão da análise do problema E/T com data de entrega comum, usando como função objetivo $Z_r = \sum_{i=1}^{n} |c_i - d|^r$, onde r é um inteiro positivo. Foram demonstradas propriedades e conceitos relacionados à solução ótima. Foi proposta uma formulação de programação dinâmica para a resolução do problema geral. Os autores não apresentaram experimentos computacionais com o algoritmo proposto mas discutiram as limitações da programação dinâmica nos casos em que $r \geq 2$.

Em KAHLBACHER (1993), foi proposto um algoritmo de programação dinâmica para a resolução do problema E/T com funções monotônicas de avanço e atraso. O desenvolvimento do algoritmo apresentado se baseou fortemente em uma propriedade da seqüência ótima. Essa propriedade estabelece que a tarefa com o maior tempo de processamento deve ser colocada na primeira ou na última posição da seqüência. Os autores também apresentam uma equivalência entre o problema E/T com data de entrega comum e o problema E/T com datas de entrega distintas mas com $(d_i - p_i)$ constante, $\forall i \in \mathbf{N}$. O autor não apresenta experimentos computacionais que comprovem a eficiência do algoritmo proposto.

HOOGEVEEN, OOSTERHOUT e VAN DE VELDE (1994) estudaram o problema E/T com data comum na sua forma restrita. Os autores apresentaram um algoritmo de "Branch & Bound" onde os limitantes inferiores e superiores eram calculados a partir da relaxação lagrangeana do modelo matemático do problema. Os testes computacionais realizados mostram que o número de nós explorados é pequeno (no máximo 320) e o tempo computacional fica em torno de 1 segundo, mesmo para instâncias com 1000 tarefas.

Em todos os trabalhos apresentados, até esse momento, procurou-se resolver o problema E/T de forma ótima ou com aproximações que garantissem a ε-otimalidade da solução. No caso de uma data de entrega comum para todas as tarefas, são poucos os procedimentos heurísticos encontrados na literatura. Um desses procedimentos é proposto por SUNG e JOO (1992) e também se baseia em propriedades de otimalidade exploradas pelos demais autores. Os experimentos computacionais mostraram que a heurística proposta teve seu comportamento favorecido por valores restritivos da data de entrega. Os tempos computacionais obtidos foram baixos, comparando-se com os tempos do procedimento ótimo, e ficam em torno de 1 minuto, para instâncias com 90 tarefas.

2-2 - PROBLEMAS COM DATAS DE ENTREGAS DISTINTAS

Assim como no caso das datas de entrega comum, na maioria dos trabalhos encontrados, os autores consideraram que as tarefas estivessem disponíveis simultaneamente para processamento.

Segundo BAKER e SCUDDER (1990), o problema com datas de entregas distintas é mais difícil de ser resolvido que o problema com data de entrega comum, a não ser no caso em que as datas de entrega são consideradas variáveis do problema. Verifica-se que a maioria dos trabalhos consiste no desenvolvimento de algoritmos heurísticos e comparação dos resultados com valores ótimos conhecidos para os problemas de teste.

Um exemplo desse procedimento é dado no trabalho de FRY, ARMSTRONG e BLACKSTONE (1987), onde se constrói uma heurística baseada em propriedades de adjacência ótima e na inserção de intervalos ociosos através de um Programa Linear. Os testes computacionais realizados para problemas com 15 tarefas mostraram que os resultados heurísticos ficam a menos de 2% dos valores ótimos, na média. O tempo de execução do algoritmo não foi maior que 1 segundo.

GAREY, TARJAN e WILFONG (1988) provaram que o problema E/T é NP-difícil, mesmo para o caso em que $h_i = t_i = 1$, $\forall i \in \mathbf{N}$ e, para esse caso, propuseram um procedimento de inserção ótima de intervalos ociosos, dada uma seqüência fixa de processamento. Os autores deixaram indicada a forma de estender esse procedimento para o caso de penalidades de atraso e avanço quaisquer. Não foram apresentados testes computacionais.

Com o objetivo de evitar os problemas relativos ao esforço computacional excessivo gasto pelos algoritmos de "Branch & Bound", OW e MORTON (1989) propuseram uma heurística de busca que limita o número de nós visitados. O procedimento consiste,

basicamente, em pesquisar um número limitado de caminhos de solução em paralelo. A essa heurística foi dado o nome de "Filtered Beam Search". Os testes computacionais mostraram que ela é superior às heurísticas de sequenciamento por regras de prioridade de tarefas e à busca em vizinhança. Nas instâncias estudadas, não foi permitida a inserção de intervalos ociosos e N continha, no máximo, 25 tarefas. Nesse mesmo trabalho, os autores apresentam um Teorema de Adjacência Ótima. Esse teorema torna possível avaliar o efeito da troca de posição entre duas tarefas adjacentes sem que seja necessário construir uma nova seqüência.

YANO e KIM (1991), estudaram um caso particular do problema E/T, onde as penalidades de avanço e atraso são proporcionais aos tempos de processamento das tarefas. Foi feito um extenso experimento computacional para a comparação de desempenho entre cinco heurísticas e um algoritmo "Branch & Bound", em instâncias com até 40 tarefas. Quatro dessas heurísticas possuíam fases distintas de ordenação de tarefas e inserção ótima de intervalos ociosos. A quinta heurística partia da melhor solução encontrada pelas outras quatro e realizava uma busca em vizinhança. Essa heurística se comportou extremamente bem nos testes computacionais. Em apenas um dos 100 problemas testados, o algoritmo ótimo (gastando um tempo computacional proibitivo em aplicações práticas) encontrou uma solução melhor que essa heurística.

Dos trabalhos pesquisados, o único a considerar o problema E/T sujeito a instantes de liberação diferentes de zero foi MANNUR e ADDAGATLA (1993). Os autores também consideraram intervalos ociosos fixos ao longo de horizonte de planejamento, correspondendo a intervalos necessários a manutenção de máquina, trocas de turno e intervalos de refeição de operários. Foram propostas duas heurísticas de solução para o problema, uma delas baseada nas características de distribuição dos valores dos instantes de liberação e outra baseada nas datas de entrega. Os testes computacionais consistiram na comparação dos resultados gerados por essas duas heurísticas, em cenários de produção distintos. Infelizmente, não foi fornecida nenhuma comparação entre os valores heurísticos e valores ótimos globais ou locais. Os experimentos mostraram que ambas as heurísticas são rápidas, demorando não mais de 0,9 segundos para resolver instâncias com 150 tarefas.

Um trabalho recente que, ao contrário dos anteriormente apresentados, não sugere um procedimento heurístico de solução é o de DAVIS e KANET (1993). Os autores estudaram o problema com penalidades quaisquer para avanço e atraso. Inicialmente, provou-se que a solução ótima para o problema E/T considerado encontra-se dentro do conjunto de seqüências semiativas. Uma seqüência semiativa é aquela em que nenhum deslocamento de tarefas pode diminuir o custo total da seqüência. Obviamente, numa seqüência semiativa, há intervalos ociosos inseridos entre as tarefas. O procedimento de

construção da seqüência semiativa correspondente a uma determinada ordenação das tarefas foi chamado de TIMETABLER. Os autores propuseram, então, um algoritmo "Branch & Bound", onde TIMETABLER foi usado em cada nó da árvore de busca. Foram feitos experimentos computacionais para comparação do algoritmo proposto e heurísticas já existentes para problemas com 8 tarefas. Destaca-se a comparação feita com o trabalho de OW e MORTON (1989). Como estes não consideraram intervalos ociosos inseridos na seqüência, os autores concluíram que os custos obtidos pela Busca em Raio Filtrada ficariam bem acima daqueles obtidos em seu trabalho (no pior caso, chegariam a ficar a mais de 300% acima do ótimo, para os problemas testados). Isso veio a confirmar a hipótese feita por BAKER e SCUDDER (1990) de que, quando se usa uma medida não regular, como o avanço no problema E/T com datas distintas, a inserção de intervalos ociosos entre as tarefas diminui o custo total de processamento.

Outro trabalho onde se desenvolve um algoritmo ótimo de solução para o problema é KIM e YANO (1994). Nesse artigo, os autores desenvolvem um limitante inferior bastante forte, capaz de reduzir consideravelmente o número de nós explorados por um algoritmo "Branch & Bound". Nos testes compuacionais, primeiro comparou-se o comportamento de 10 heurísticas, quanto à qualidade da solução e o tempo computacional dispendido. A heurística que apresentou o melhor resultado foi a de Busca Tabu, e sua solução final foi usada como limitante superior para o algoritmo ótimo. Os resultados computacionais gerados pela aplicação desse algoritmo a instâncias com até 28 tarefas mostrou que o tempo computacional cresce rapidamente conforme cresce o número de tarefas, mas o aumento no tempo é muito menor que o aumento do número de soluções possíveis, o que mostra a força do limitante inferior desenvolvido.

Nenhuma das heurísticas propostas nos trabalhos consultados considerou a inserção de intervalos ociosos de forma simultânea à construção de uma seqüência de processamento. O procedimento ótimo de DAVIS e KANET (1993) também insere tais intervalos após ser determinada a seqüência parcial correspondente a cada nó da árvore de busca. Além disso, nenhum dos autores estudou o caso geral em que as tarefas são liberadas para processamento em instantes distintos. Num ambiente de produção real, a máquina crítica realiza apenas uma parte do processamento de tarefas provenientes de vários centros de trabalho distintos. Não é uma hipótese muito realista considerar que todos os centros terminem seu trabalho ao mesmo tempo ou que se deva deixar a máquina crítica ociosa até que todas as tarefas estejam disponíveis.

Conclui-se que, alcançados os objetivos deste trabalho, terá sido feita uma contribuição valiosa para o estudo do problema E/T.

CAPÍTULO 3
HEURÍSTICA PROPOSTA

Para a solução do problema apresentado, propõe-se, neste capítulo, uma heurística que se divide em duas fases : uma fase construtiva e uma fase de busca em vizinhança.

Na fase construtiva, procura-se obter um programa **S**, factível, das tarefas de **N**. O processo de construção possui três procedimentos e procura obter um programa com um custo total próximo à otimalidade. A fase construtiva é apresentada em detalhes na seção 3.1.

Na fase de busca em vizinhança, procede-se à exploração da vizinhança da solução **S**, construída de acordo com a estratégia "Troca de Tarefas Adjacentes" (TTA). Na seção 3.2, apresenta-se um resultado teórico que possibilita a diminuição do esforço envolvido na avaliação das soluções vizinhas de **S**.

Em seguida, na seção 3.3, é apresentada a estratégia de exploração da vizinhança TTA. Finalmente, em 3.4, encontra-se um diagrama de blocos que resume esquematicamente o processo de solução proposto.

Ao longo das seções 3.1 e 3.3 é resolvido, passo a passo, um exemplo com 8 tarefas. O objetivo é mostrar de forma clara como funcionam os procedimentos da fase construtiva e também a aplicação da busca em vizinhança.

3.1 - FASE CONSTRUTIVA DA HEURÍSTICA

A fase construtiva da heurística proposta consiste de três procedimentos básicos:

I - Procedimento de Ordenação: pré-ordenação do conjunto **N**. A solução obtida ao final desse procedimento representa um programa que é, em geral, infactível. A factibilização desse programa é feita pelos próximos procedimentos.

II - Procedimento de Factibilização: construção, tarefa a tarefa, de um programa factível **S**. Nessa fase, a heurística já introduz intervalos ociosos no programa, embora não de forma definitiva.

III - Procedimento de Atualização de Intervalos Ociosos: verifica se o custo de processamento do programa parcial pode ser diminuído pelo aumento ou diminuição dos intervalos ociosos inseridos entre as tarefas.

O Procedimento I é utilizado somente uma vez, enquanto que os demais procedimentos são utilizados iterativamente durante a construção do programa **S**. O Diagrama 3.1 mostra como os procedimentos se interrelacionam.

Os procedimentos II e III são utilizados uma vez a cada iteração, quando se insere a tarefa J_j no programa de produção.

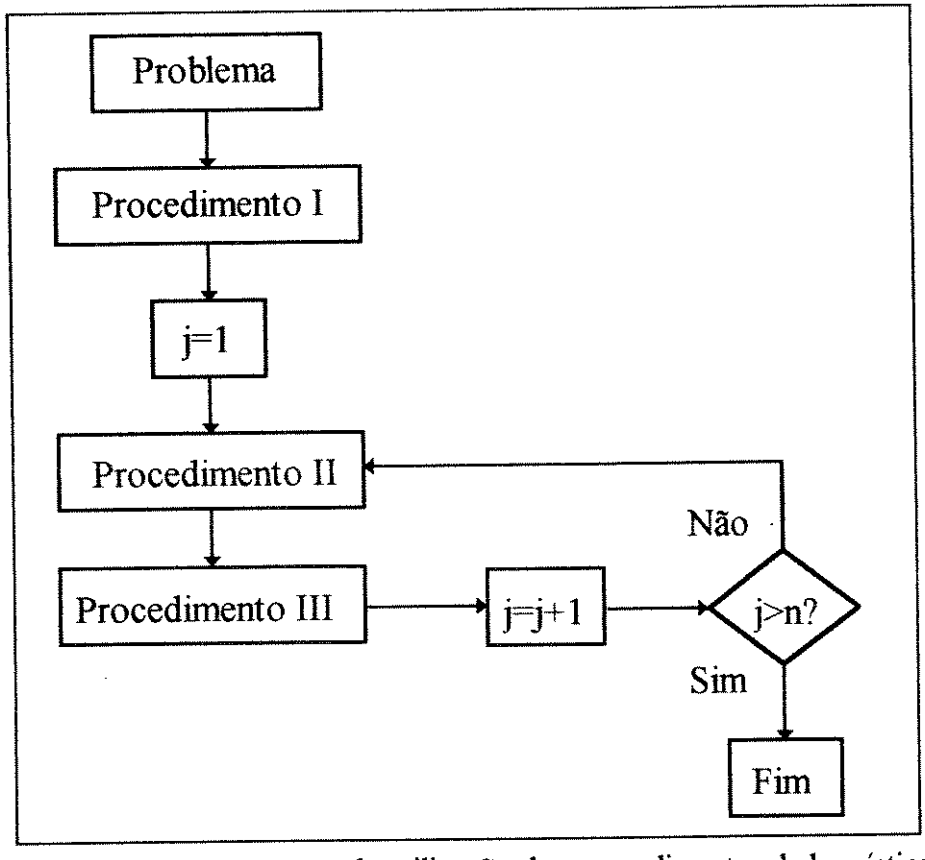

Diagrama 3.1 - Esquema de utilização dos procedimentos da heurística

A descrição dos procedimentos é feita de acordo com sua ordem de utilização na fase construtiva.

EXEMPLO ILUSTRATIVO

O exemplo utilizado nesta seção e em 3.3 possui os seguintes parâmetros:

Tarefa (J_i)	r_i	p_i	d_i	h_i	t_i
J_1	69	34	159	25	31
J_2	166	97	205	18	84
J_3	14	22	264	34	1
J_4	87	30	121	6	30
J_5	62	24	221	61	97
J_6	270	47	237	86	46
J_7	310	91	181	84	98
J_8	255	40	136	73	69

Tabela 3.1 - Dados do exemplo utilizado nesta seção

PROCEDIMENTO I - ORDENAÇÃO

Dado o conjunto $N = \{1, 2, \ldots, n\}$ de tarefas que requerem processamento na única máquina disponível, é necessário que se escolha um critério de ordenação inicial, a partir da qual se possa construir um programa factível. A seguir são definidos alguns parâmetros envolvidos no desenvolvimento desse critério.

Supõe-se, inicialmente, que $|N| = 1$, ou seja, há apenas uma tarefa no programa e sua posição deve ser escolhida de forma que o processamento termine no instante mais próximo possível de sua data de entrega. Associados à tarefa J_1, são dados os parâmetros p_1, r_1, d_1, h_1 e t_1.

Para essa única tarefa, há dois casos a considerar:

CASO 1: $r_1 + p_1 > d_1 \rightarrow$ Como mostra a Figura 3.1, a tarefa J_1 está sempre atrasada e, portanto, deve-se fazer $s_1 = r_1$ e $c_1 = r_1 + p_1$.

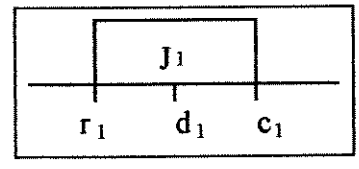

Figura 3.1 - Posicionamento ótimo de J_1 quando $r_1 + p_1 > d_1$.

CASO 2: $r_i + p_i \leq d_i \rightarrow$ a tarefa J_i deve ser iniciada em $s_i = d_i - p_i$ para que seu custo seja igual a zero.

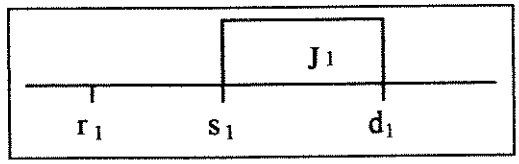

Figura 3.2 - Posicionamento ótimo de J_i quando $r_i + p_i \leq d_i$.

A partir desse dois casos é possível definir alguns conceitos que formam a base para o procedimento proposto.

Definição 1: "Chama-se Intervalo Preferencial de Posicionamento, $[u_i, v_i]$, de uma tarefa J_i, $i \in \mathbf{N}$, àquele construído através da aplicação do caso 1 ou 2, descritos anteriormente, a essa tarefa."

Portanto, para cada tarefa J_i, $i \in \mathbf{N}$, existe um intervalo $[u_i, v_i]$, onde:

$$u_i = \begin{cases} r_i, \text{ se } r_i + p_i > d_i \\ d_i - p_i, \text{ se } r_i + p_i \leq d_i \end{cases}$$
$$v_i = \begin{cases} r_i + p_i, \text{ se } r_i + p_i > d_i \\ d_i, \text{ se } r_i + p_i \leq d_i \end{cases} \qquad (3.1)$$

Definição 2: "Chama-se de Intervalo de Folga de uma tarefa, w_i, $i \in \mathbf{N}$ ao mínimo entre $s_i - r_i$ e $s_i - I$, onde I representa o instante de início do intervalo ocioso mais próximo de s_i, à esquerda do intervalo de processamento de J_i."

Durante o Procedimento I, considera-se $I = 0$. Portanto, o valor inicial do intervalo w_i é calculado como $w_i = u_i - r_i$, $\forall i \in \mathbf{N}$.

Além desses parâmetros, derivados das definições, é preciso sinalizar quando uma tarefa estará sempre atrasada ou quando ela tem possibilidade de se deslocar à esquerda de seu Intervalo Preferencial de Posicionamento. Esse deslocamento provoca o aparecimento de um custo de avanço e só será realizado em determinadas circunstâncias, discutidas posteriormente. A sinalização da possibilidade de um deslocamento à esquerda é feita através do parâmetro a_i, onde:

$$a_i = \begin{cases} 1, & \text{se } r_i + p_i > d_i \\ 0, & \text{se } r_i + p_i \leq d_i \end{cases} \qquad (3.2)$$

O Procedimento de Ordenação advém da generalização da análise de um exemplo particular do problema E/T, onde todos os intervalos $[u_i, v_i]$ são disjuntos. Para este caso, o programa de menor custo é aquele obtida pela ordenação dos índices de **N** de acordo com o critério de valores não decrescentes de u_i. Nesse caso, observa-se que o posicionamento das tarefas, dado por $[u_i, v_i]$ também é ótimo com relação aos intervalos ociosos. O conjunto de índices ordenados por esse critério foi chamado de **A**.

O Procedimento de Ordenação pode ser, então, resumido como:

Procedimento I:

Passo 1: Calcule os valores u_i, v_i, w_i e a_i, $\forall i \in \mathbf{N}$.

Passo 2: Faça $s_i = u_i$ e $c_i = v_i$, $\forall i \in \mathbf{N}$.

Passo 3: Construa um conjunto de índices **A**, ordenado de acordo com valores não decrescentes de u_i.

APLICAÇÃO DO PROCEDIMENTO I AO EXEMPLO

Pode-se agrupar os valores calculados nos passos 1 e 2 na seguinte tabela:

Tarefa (J_i)	u_i	v_i	w_i	a_i	s_i	c_i
J_1	125	159	56	0	125	159
J_2	166	263	0	1	166	263
J_3	242	264	228	0	242	264
J_4	91	121	4	0	91	121
J_5	197	221	135	0	197	221
J_6	270	317	0	1	270	317
J_7	310	401	0	1	310	401
J_8	255	295	0	1	255	295

Tabela 3.2 - Aplicação dos passos 1 e 2 do Procedimento I aos dados do exemplo.

O conjunto **A**, para o exemplo, é igual a :

$$\mathbf{A} = \{J_4, J_1, J_2, J_5, J_3, J_8, J_6, J_7\}$$

PROCEDIMENTO II - FACTIBILIZAÇÃO

Se for válida a hipótese de intervalos $[u_i, v_i]$ disjuntos, o que se obtém ao término do Procedimento I são os intervalos ótimos de processamento das tarefas de **N**. Entretanto, essa hipótese só é válida para algumas instâncias do problema E/T. Em geral, o programa representado pelo conjunto **A** é uma solução infactível para o problema pois os valores de s_i violam a hipótese de que a máquina deve processar uma tarefa por vez. Portanto, o conjunto **A** será usado apenas como ponto de partida para a fase construtiva.

A heurística proposta procura resolver as infactibilidades pela construção, tarefa a tarefa, de um programa, **S**, factível. Esse programa conterá, eventualmente, intervalos ociosos. Para trabalhar melhor com esses intervalos, é conveniente dividir o programa em blocos de tarefas, usando a seguinte definição:

Definição 3: "Um bloco B do programa **S** é definido como um conjunto de tarefas processadas continuamente, isto é, sem intervalos ociosos entre elas. Portanto, em $B = \{J_r, J_{(r+1)}, \ldots, J_{(r+t)}\}$, deve-se ter $s_{(r+k)} = c_{(r+k-1)}$, para $k = 1, 2, \ldots, t$."

Na descrição do Procedimento II, as convenções usadas para a indexação das tarefas são as seguintes:

- $i(j)$ é o índice da tarefa que está sendo inserida no programa parcial **S**, na j-ésima chamada ao Procedimento II, pela heurística;
- $[k]$ é o índice da tarefa que ocupa a k-ésima posição no programa **S**.

O passo inicial da construção do programa factível é a tomada da primeira tarefa de **A**, ou seja, $J_{i(1)}$, e sua inserção em **S** como pertencente ao bloco B_1, sem que sejam alterados os valores de $s_{i(1)}$ e $c_{i(1)}$.

A seguir, toma-se a segunda tarefa de **A**, a tarefa $J_{i(2)}$, e verifica-se se há sobreposição no processamento de $J_{[1]}$ e $J_{i(2)}$. Se não houver sobreposição, existem duas possibilidades:

- $s_{i(2)} = c_{[1]} \Rightarrow B_1 = B_1 \cup \{J_{i(2)}\}$; (3.3a)
- $s_{i(2)} > c_{[1]} \Rightarrow B_1 = \{J_{[1]}\}$ e $B_2 = \{J_{i(2)}\}$. (3.3b)

Se houver sobreposição, o Procedimento II deve encontrar o melhor posicionamento relativo entre as duas tarefas para que o bloco B_1 seja factível e apresente o menor custo possível. Então, passa-se à análise da tarefa $J_{i(3)}$, repetindo-se o processo, para todas as tarefas, até que $J_{i(n)}$ esteja incluída no programa **S**.

Para generalizar o processo, assume-se que a iteração atual seja a iteração j ($j > 1$). Portanto, há ($j - 1$) tarefas no programa parcial corrente, distribuídas em t blocos. Deseja-se analisar a inserção de $J_{i(j)}$ nesse programa. Para tanto, a primeira coisa a se fazer é verificar se $s_{i(j)} \geq c_{[j-1]}$, ou seja, se $J_{i(j)}$ pode ser inserida diretamente no programa, sem se sobrepor a nenhuma tarefa já sequenciada (Figura 3.3a). Caso isso não ocorra, procura-se, no programa parcial, qual é a primeira tarefa em que há sobreposição no processamento (Figura 3.3b). Seja k a posição dessa tarefa.

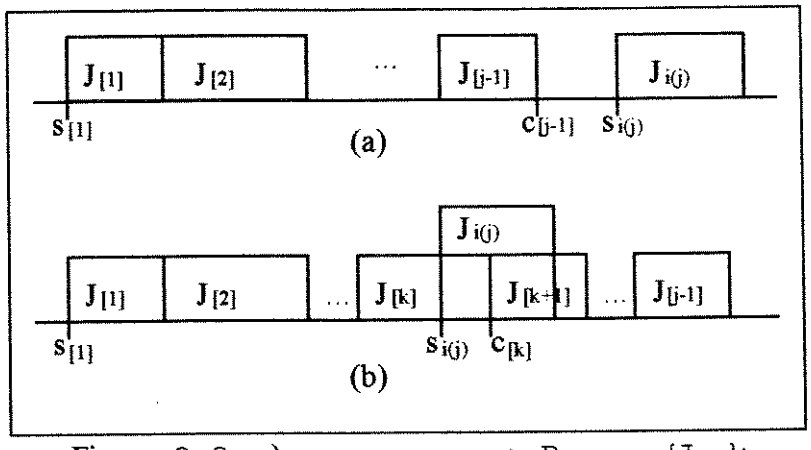

Figura 3.3 - a) $s_{i(j)} > c_{[j-1]} \rightarrow B_{(t+1)} = \{J_{i(j)}\}$;
b) $s_{i(j)} < c_{[k]}$.

O próximo passo deve ser, então, achar a melhor ordem entre $J_{[k]}$ e $J_{i(j)}$ de forma que o aumento no custo do processamento das duas tarefas seja mínimo. Há quatro tipos de alterações possíveis nos valores de $s_{i(j)}$, $c_{i(j)}$, $s_{[k]}$ e $c_{[k]}$. Os dois primeiros tipos de alteração são mais simples, pois, mantendo-se a posição de uma das tarefas, desloca-se a outra para a direita, observando-se apenas a variação no custo de atraso. Essas alterações de posicionamento são sempre possíveis de serem realizadas pois, a partir do instante de liberação para processamento, qualquer tarefa pode se deslocar à direita. Os dois últimos tipos de alteração são mais complexos pois envolvem deslocamentos à direita e à esquerda da posição atual das duas tarefas, observando-se a variação dos custos de avanço e atraso.

Alteração 1: A tarefa $J_{i(j)}$ fica fixa e a tarefa $J_{[k]}$ se desloca à direita de uma distância igual a $(c_{i(j)} - s_{[k]})$ como mostra a Figura 3.4. A variação no custo de processamento da tarefa $J_{[k]}$ é igual a:

$$\Delta_1 = t_{[k]} \cdot (c_{i(j)} - s_{[k]}). \qquad (3.4)$$

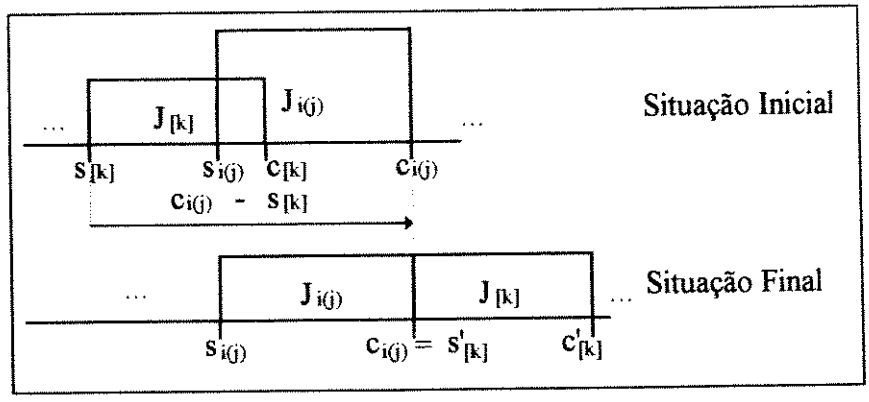

Figura 3.4 - Diagrama da Alteração 1.

Alteração 2: A tarefa $J_{[k]}$ fica fixa e a tarefa $J_{i(j)}$ se desloca à direita de uma distância igual a $(c_{[k]} - s_{i(j)})$ como mostrado na Figura 3.5. A variação no custo de processamento da tarefa $J_{i(j)}$ é igual a:

$$\Delta_2 = t_{i(j)} \cdot (c_{[k]} - s_{i(j)}). \qquad (3.5)$$

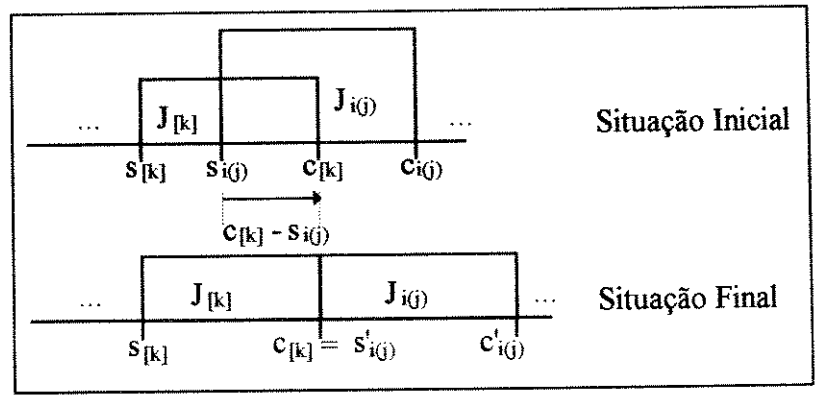

Figura 3.5 - Diagrama da Alteração 2.

Alteração 3: Essa alteração só é possível quando $a_{i(j)} = 0$, ou seja, quando $r_{i(j)} \leq u_{i(j)}$. Nesse caso, é preciso calcular o instante mais cedo em que $J_{i(j)}$ pode começar a ser processada, sem que se desloque para a esquerda mais do que o suficiente para desfazer a sobreposição. Isso porque não se deseja aumentar o custo de avanço de $J_{i(j)}$ desnecessariamente. O instante em questão será chamado de s_{aux} e é calculado como:

$$s_{aux} = \max\{c_{[k-1]}, r_{i(j)}, s_{[k]} - p_{i(j)}\}, \qquad (3.6)$$

onde:

$c_{[k-1]}$ = instante em que a máquina termina de processar a tarefa $J_{[k-1]}$ e fica liberada para o processamento das demais tarefas;

$r_{i(j)}$ = instante que garante que $J_{i(j)}$ não começará a ser processada antes de estar disponível.

$s_{[k]} - p_{i(j)}$ = instante que garante o deslocamento mínimo necessário para desfazer a sobreposição entre $J_{[k]}$ e $J_{i(j)}$.

Ilustra-se o caso em que $s_{aux} = r_{i(j)}$ na Figura 3.6. A tarefa $J_{[k]}$ se desloca de uma distância igual a $(s_{aux} + p_{i(j)} - s_{[k]})$ à direita e a tarefa $J_{i(j)}$ se desloca de $(s_{i(j)} - s_{aux})$ à esquerda. A variação no custo das duas tarefas é igual a:

$$\Delta_3 = t_{[k]} \cdot (s_{aux} + p_{i(j)} - s_{[k]}) + h_{i(j)} \cdot (s_{i(j)} - s_{aux}). \qquad (3.7)$$

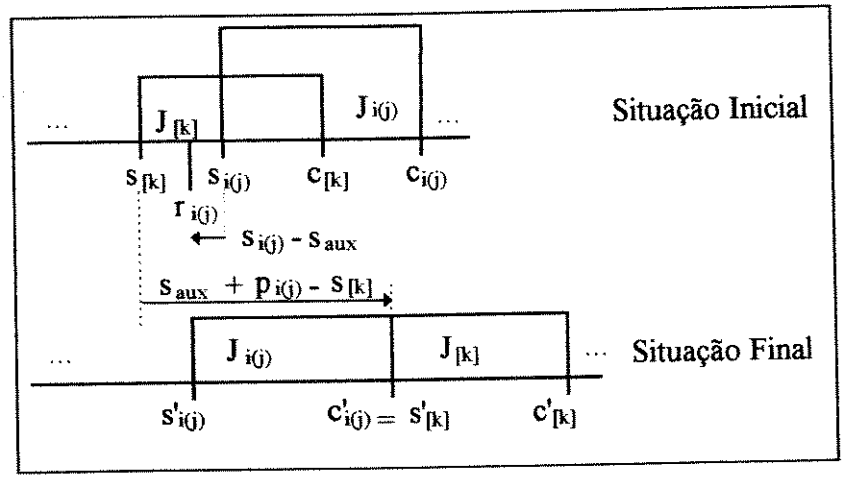

Figura 3.6 - Diagrama da Alteração 3.

Quando $s_{aux} = s_{[k]} - p_{i(j)}$, $J_{[k]}$ não sofre deslocamento para a direita e, quando $s_{aux} = c_{[k-1]}$, $J_{i(j)}$ se desloca à esquerda e $J_{[k]}$ se desloca para a direita.

Quando $a_{i(j)} = 1$, o deslocamento da tarefa $J_{i(j)}$ à esquerda é impossível. Para evitar que esse movimento seja realizado, faz-se $\Delta_3 = M$, onde M é um número suficientemente grande.

Alteração 4: Analogamente ao caso anterior, essa alteração só é possível quando $a_{[k]} = 0$, ou seja, quando $r_{[k]} \leq s_{[k]}$. Calcula-se s_{aux} para $J_{[k]}$ da mesma forma:

$$s_{aux} = \max\{c_{[k-1]}, r_{[k]}, s_{i(j)} - p_{[k]}\} \qquad (3.8)$$

Novamente, ilustra-se o caso em que $s_{aux} = r_{[k]}$ na Figura 3.7. A tarefa $J_{[k]}$ se desloca de $(s_{[k]} - s_{aux})$ à esquerda e a tarefa $J_{i(j)}$ se desloca de $(s_{aux} + p_{[k]} - s_{i(j)})$ à direita. A variação no custo das duas tarefas é igual a:

$$\Delta_4 = t_{i(j)} \cdot (s_{aux} + p_{[k]} - s_{i(j)}) + h_{[k]} \cdot (s_{[k]} - s_{aux}). \qquad (3.9)$$

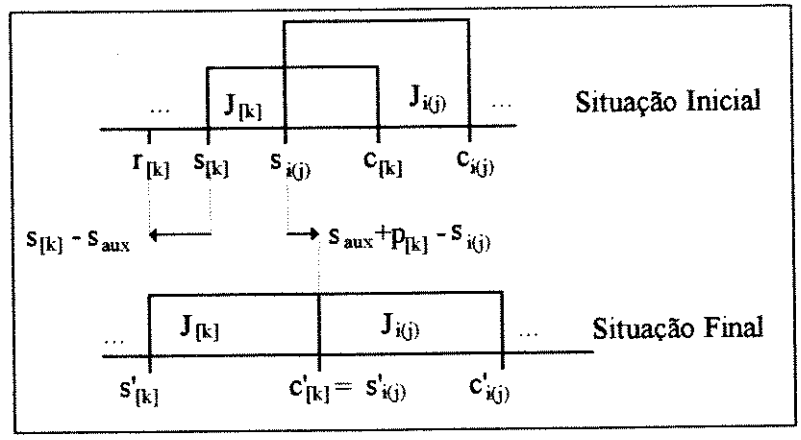

Figura 3.7 - Diagrama da Alteração 4.

Quando $s_{aux} = s_{i(j)} - p_{[k]}$, $J_{i(j)}$ não se desloca e, quando $s_{aux} = c_{[k-1]}$, $J_{i(j)}$ se desloca à direita e $J_{[k]}$ se desloca à esquerda.

De forma análoga ao que ocorre na Alteração 3, se $a_{[k]} = 1$, o deslocamento de $J_{[k]}$ à esquerda é impossível e faz-se $\Delta_4 = M$, onde M é um número suficientemente grande.

Após serem calculados esses quatro valores de variação de custo, é possível escolher $\Delta = \min\{\Delta_1, \Delta_2, \Delta_3, \Delta_4\}$. De acordo com a alteração que gera o valor mínimo para Δ, são atualizados os valores de $s_{i(j)}$, $c_{i(j)}$, $s_{[k]}$ e $c_{[k]}$. Uma das tarefas é mantida na k-ésima posição de **S** e a outra é deslocada para a direita, sobrepondo-se com $J_{[k+1]}$. O Procedimento II continua analisando sobreposições e deslocando tarefas à direita até que seja resolvido o posicionamento relativo entre $J_{i(j)}$ e $J_{[j-1]}$ de modo que o programa parcial **S** não contenha nenhuma infactibilidade. Todas as análises de sobreposição são feitas de forma similar. Portanto, a descrição das atitudes tomadas após a determinação do valor de Δ são descritas utilizando-se apenas os índices [k] e i(j).

Se $\Delta = \Delta_2$ ou $\Delta = \Delta_4$, $J_{[k]}$ é mantida na k-ésima posição de **S** e $J_{i(j)}$ se desloca à direita. Nesse caso, a próxima sobreposição a ser analisada é entre $J_{i(j)}$ e $J_{[k+1]}$.

Quando $\Delta = \Delta_1$ ou $\Delta = \Delta_3$, conclui-se que o custo de deslocamento à direita de $J_{i(j)}$ é maior que o custo correspondente para $J_{[k]}$. A melhor ordem de processamento encontrada para essas tarefas é $J_{i(j)}$ - $J_{[k]}$, como pode ser visto na Figura 3.8a e 3.8b, para o caso em que $\Delta = \Delta_1$. Sendo assim, fixa-se $J_{i(j)}$ na k-ésima posição de **S** e passa-se a avaliar a sobreposição entre $J_{[k]}$ e $J_{[k+1]}$, a menos que $J_{[k]} = J_{[j-1]}$ e, portanto, o programa **S** já tenha se tornado factível. A partir desse instante, tudo se passa como se a tarefa a ser inserida no programa parcial fosse $J_{[k]}$. Portanto, para manter a coerência com a convenção de índices adotada, trocam-se os índices i(j) e [k], gerando uma nova situação inicial para a próxima iteração do Procedimento II (Figura 3.8c).

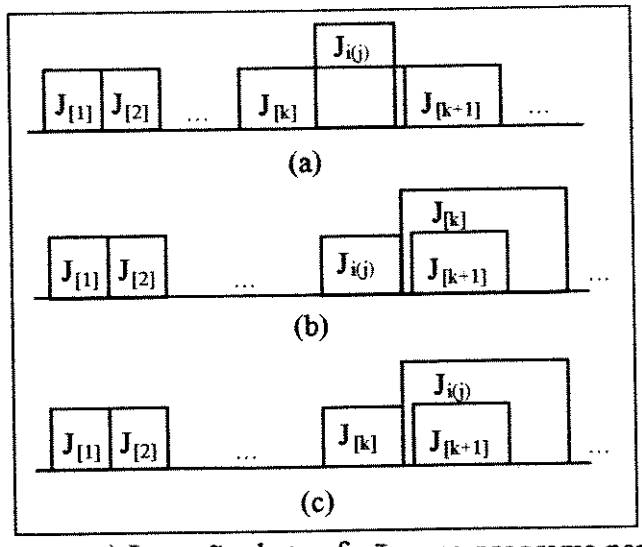

Figura 3.8: a) Inserção da tarefa $J_{i(j)}$ no programa parcial **S**.
Verifica-se sobreposição de processamento com $J_{[k]}$.
b) Posicionamento resultante quando $\Delta = \Delta_1$.
c) Nova situação inicial, após a troca de índices.

As modificações realizadas, então, após a escolha de Δ, são as seguintes:

- Se $\Delta = \Delta_1 \Rightarrow J_{i(j)}$ precede $J_{[k]}$;
  $c_{[k]} = c_{i(j)} + p_{[k]}$;
  $s_{[k]} = c_{i(j)}$;
  $i(j) \leftarrow [k]$ e $[k] \leftarrow i(j)$.
- Se $\Delta = \Delta_2 \Rightarrow J_{[k]}$ precede $J_{i(j)}$;
  $c_{i(j)} = c_{[k]} + p_{i(j)}$;
  $s_{i(j)} = c_{[k]}$.
- Se $\Delta = \Delta_3 \Rightarrow J_{i(j)}$ precede $J_{[k]}$;
  $c_{i(j)} = s_{aux} + p_{i(j)}$;
  $s_{i(j)} = s_{aux}$;
  $s_{[k]} = c_{i(j)}$;
  $c_{[k]} = s_{[k]} + p_{[k]}$;
  $i(j) \leftarrow [k]$ e $[k] \leftarrow i(j)$.
- Se $\Delta = \Delta_4 \Rightarrow J_{[k]}$ precede $J_{i(j)}$;
  $c_{[k]} = s_{aux} + p_{[k]}$;
  $s_{[k]} = s_{aux}$;
  $s_{i(j)} = c_{[k]}$;
  $c_{i(j)} = s_{i(j)} + p_{i(j)}$.

Essas modificações garantem que a sobreposição entre as tarefas $J_{i(j)}$ e $J_{[k]}$ será desfeita. Mas, para garantir que o programa parcial **S**, obtido ao final do Procedimento II seja factível, é preciso observar mais detalhadamente o que ocorre a cada modificação no programa como um todo.

Primeiramente, note-se que, quando uma das tarefas, $J_{i(j)}$ ou $J_{[k]}$, é deslocada à esquerda, evita-se a sobreposição com a tarefa $J_{[k-1]}$ através da inclusão de $c_{[k-1]}$ no cálculo de s_{aux}. Portanto, se houver alguma infactiblidade provocada pelas modificações descritas, ela afetará o processamento das tarefas $J_{[k+1]}, J_{[k+2]}, \ldots, J_{[j-1]}$.

Quando se obtém $\Delta = \Delta_2$ ou $\Delta = \Delta_4$, a tarefa $J_{[k]}$ continua ocupando a posição k no programa parcial. Ao se analisar a sobreposição entre $J_{i(j)}$ e $J_{[k+1]}$, é como se a tentativa de inclusão de $J_{i(j)}$ se desse em outra parte do programa, não alterando a factibilidade do programa parcial $\mathbf{S}-\{J_{i(j)}\}$. As Figuras 3.9 e 3.10 ilustram esses casos.

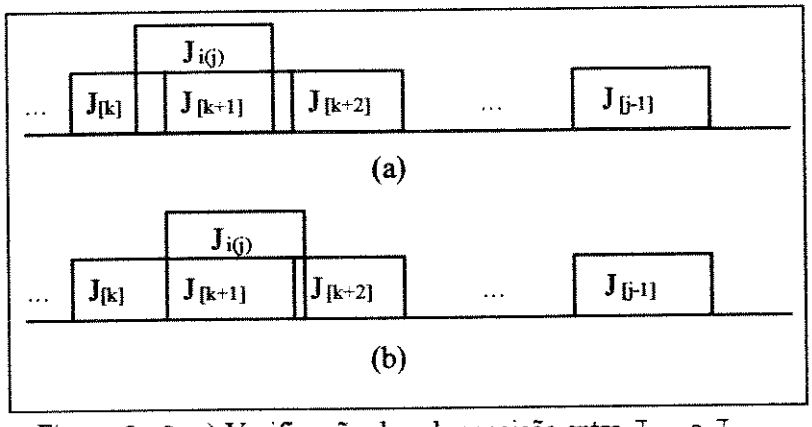

Figura 3.9: a) Verificação de sobreposição entre $J_{i(j)}$ e $J_{[k+1]}$

b) Calcula-se $\Delta = \Delta_2$.

O programa parcial $\mathbf{S}-\{J_{i(j)}\}$ continua factível.

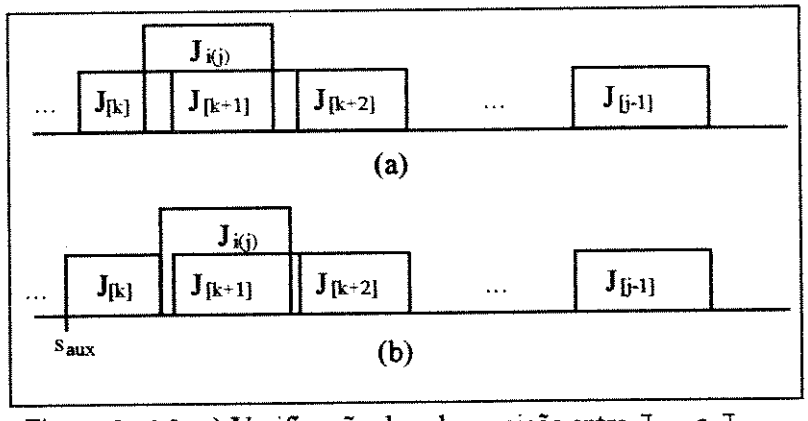

Figura 3.10: a) Verificação de sobreposição entre $J_{i(j)}$ e $J_{[k+1]}$

b) Calcula-se $\Delta = \Delta_4$.

O programa parcial $\mathbf{S}-\{J_{i(j)}\}$ continua factível.

Ao se obter $\Delta = \Delta_1$ ou $\Delta = \Delta_2$, já não se pode garantir que o programa $\mathbf{S} - \{J_{i(j)}\}$ continue factível. Quando a sobreposição entre $J_{i(j)}$ e $J_{[k]}$ for tal que $J_{i(j)}$ também se sobreponha a $J_{[k+1]}$ e, eventualmente, a outras tarefas posteriores a $J_{[k]}$ em **S**, a realização das modificações descritas, para o caso em que $\Delta = \Delta_1$, podem levar à situação mostrada na Figura 3.11b.

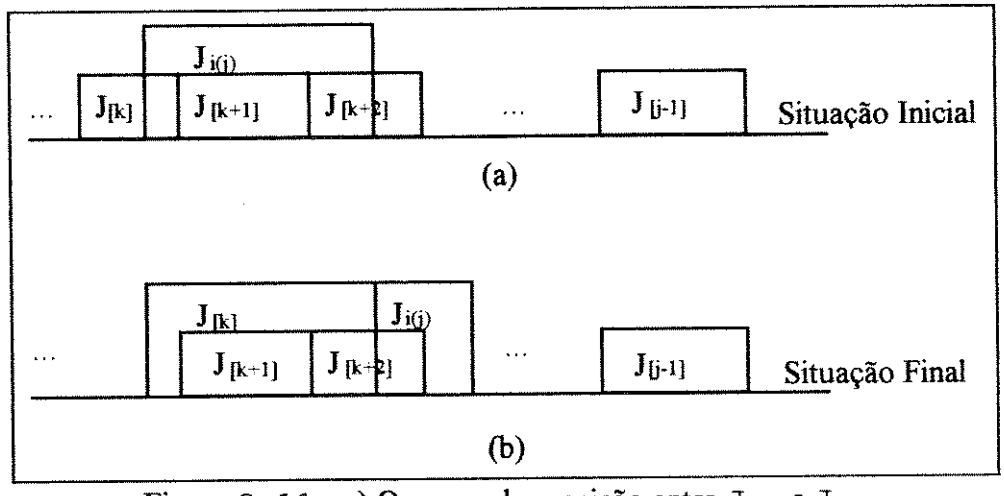

Figura 3.11 - a) Ocorre sobreposição entre $J_{i(j)}$ e $J_{[k]}$.

b) Programa parcial infactível gerado pelo Procedimento II

Como o Procedimento II procura resolver as infactibilidades provocadas pelo posicionamento de $J_{i(j)}$ apenas, a tarefa $J_{[k]}$ pode continuar sobreposta a outras tarefas em **S** se esse caso não for analisado com cuidado. Para detectar esse problema, incluiu-se um teste bastante simples quando $\Delta = \Delta_1$ ou $\Delta = \Delta_2$: após a realização das modificações indicadas, se $c_{[k]} = s_{i(j)} > s_{[k+1]}$, é sinal que a situação indicada na Figura 3.11b ocorreu. Caso contrário, o Procedimento II pode prosseguir normalmente, pois somente $J_{i(j)}$ se sobrepõe a outras tarefas.

Na Figura 3.11b, observa-se que há duas tarefas provocando situações de infactibilidade no programa parcial, $J_{i(j)}$ e $J_{[k]}$. Como o Procedimento II não é capaz de analisar as infactibilidades provocadas por duas tarefas simultaneamente, é preciso devolver uma dessas tarefas ao conjunto **A**, para que se possa dar prosseguimento ao Procedimento II.

A seguir, discute-se como é feita a escolha de qual tarefa deve ser devolvida ao conjunto **A**. As tarefas desse conjunto **A** estão ordenadas de acordo com o instante de início de seu Intervalo Preferencial de Posicionamento. Para manter essa ordem, a tarefa devolvida deverá ocupar a primeira posição de **A**, sendo reinserida em **S** na próxima

iteração do Procedimento II. Deseja-se que, nessa reinserção, a situação ocorrida não se repita.

Na situação inicial (Figura 3.11a), observa-se que: ou a tarefa $J_{[k+1]}$ não pode se deslocar à esquerda, por iniciar em seu instante de liberação para processamento, ou, em alguma iteração anterior do Procedimento II, foi analisada a sobreposição entre $J_{[k]}$ e $J_{[k+1]}$, encontrando-se o posicionamento relativo atual como sendo o de menor custo. No segundo caso, o fato de $J_{[k+1]}$ ser processada após $J_{[k]}$ indica que seu custo de deslocamento à direita é menor que o custo correspondente para $J_{[k]}$. Sendo assim, na hipótese de se retirar de **S** a tarefa $J_{[k]}$, na Figura 3.11b, quando essa tarefa for reinserida no programa parcial, e o Procedimento II analisar sua sobreposição com $J_{[k+1]}$, pode haver a repetição da situação em que duas tarefas provoquem infactibilidades em **S**.

Se, por outro lado, $J_{i(j)}$ for devolvida ao conjunto **A**, passa-se diretamente à análise de sobreposição entre $J_{[k]}$ e $J_{[k+1]}$. Dessa forma, todos os problemas causados pela inserção da tarefa com o índice i(j), na Figura 3.11a, são resolvidos de uma só vez, sendo que mais de uma tarefa pode ser devolvida a **A**, durante esse processo.

Portanto, na Figura 3.11b, a tarefa escolhida para ser retirada de **S** e devolvida à posição j de **A** é $J_{i(j)}$. Todos os parâmetros dessa tarefa são reinicializados, ou seja, faz-se

$$s_{i(j)} = u_{i(j)}, c_{i(j)} = v_{i(j)} \text{ e } w_{i(j)} = u_{i(j)} - r_{i(j)}. \qquad (3.10)$$

O programa parcial **S** contém, agora, uma tarefa a menos e é preciso reorganizar os índices das tarefas para que o Procedimento II possa resolver a sobreposição entre $J_{[k]}$ e $J_{[k+1]}$. A Figura 3.12 mostra a configuração desejada para o programa parcial, após essa reorganização de índices.

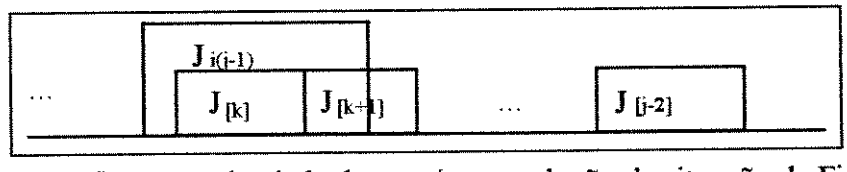

Figura 3.12 - Configuração desejada de **S** após a resolução da situação da Figura 3.11b.

Para transformar o programa parcial de forma que ele possua a configuração apresentada, é preciso indicar que $J_{[k]}$ é a tarefa que provoca infactibilidade no programa parcial, fazendo:

$$J_{i(j)} = J_{[k]}; \qquad (3.11a)$$

Após essa modificação, deve-se atualizar o posicionamento da demais tarefas de **S**. Essa atualização é feita da seguinte forma:

$$J_{[l]} = J_{[l+1]}, \text{ para } l = k, k+1, \ldots, j-1. \qquad (3.11b)$$

Finalmente, faz-se $j = j - 1$, e retoma-se o Procedimento II para resolver a sobreposição entre $J_{i(j-1)}$ e $J_{[k]}$.

Ainda assim, o Procedimento II pode não ser capaz de realizar a inserção de todas as tarefas de **A** em **S**, impossibilitando que a heurística determine um programa completo em um número de passos finito. Por exemplo, sejam três tarefas: J_a, J_b e J_c, com J_b e J_c já inseridos de forma factível em **S**, e não necessariamente adjacentes. Uma ciclagem no Procedimento II pode ser caracterizada pela repetição, por um número indefinido de iterações, da seguinte seqüência de eventos:

J_a é inserida em **S** $\rightarrow$ J_b retorna a **A**;

J_b é inserida em **S** $\rightarrow$ J_c retorna a **A**;

J_c é inserida em **S** $\rightarrow$ J_a retorna a **A**.

Infelizmente, uma prova formal de que essa ciclagem não ocorre é muito difícil de ser obtida, visto que, no cálculo da variação de custo em cada alteração de posicionamento relativo são levados em consideração não só valores fixos como tempos de processamento e custos de avanço e atraso, mas também o valor da sobreposição entre as tarefas. Também não é de fácil obtenção uma instância que prove a ocorrência de ciclagem no Procedimento II.

Portanto, para a iteração j da heurística (inclusão da j-ésima tarefa no programa) o Procedimento de Factibilização toma a seguinte forma:

Procedimento II:

Passo 0: Se $j = 1$, faça $t = 1$.
Insira $J_{i(1)}$ em B_t sem alterar os valores de $s_{i(1)}$, $c_{i(1)}$ e $w_{i(1)}$.
Pare.

Passo 1: Se $s_{i(j)} > c_{[j-1]}$ faça $t = t + 1$ e insira $J_{i(j)}$ em B_t sem alterar seus parâmetros.
Pare.
Se $s_{i(j)} = c_{[j-1]}$ insira $J_{i(j)}$ em B_t sem alterar seus parâmetros.
Pare.

Passo 2: Encontre o menor k tal que $s_{i(j)} < c_{[k]}$.

Passo 3: Calcule $\Delta = \min\{\Delta_1, \Delta_2, \Delta_3, \Delta_4\}$ de acordo com as equações dadas;

Passo 4: Atualize os valores de $s_{i(j)}$, $c_{i(j)}$, $s_{[k]}$, $c_{[k]}$.

Se $\Delta = \Delta_2$ ou $\Delta = \Delta_4$ e $k < j$, faça $k = k + 1$

Se $k = j$, Pare.

Caso contrário, vá para o Passo 3.

Se $\Delta = \Delta_1$ ou $\Delta = \Delta_3$, vá para o Passo 5.

Passo 5: Troque os índices $i(j)$ e $[k]$.

Se $k = j - 1$, Pare.

Se $s_{i(j)} \leq s_{[k+1]}$, faça $k = k + 1$.

Se $k = j$, Pare.

Caso contrário, vá para o Passo 3.

Se $s_{i(j)} > s_{[k+1]}$, devolva $J_{i(j)}$ ao conjunto **A** na posição j e reinicialize os valores de $s_{i(j)}$, $c_{i(j)}$ e $w_{i(j)}$.

Vá para o Passo 6.

Passo 6: Faça $J_{i(j)} = J_{[k]}$.

Faça $J_{[l]} = J_{[l+1]}$ para $l = k, k+1, \ldots, j-1$.

Se $j > 1$, faça $j = j - 1$ e volte para o Passo 3.

Pare.

PROCEDIMENTO III - ATUALIZAÇÃO DOS INTERVALOS OCIOSOS

Quando o Procedimento II termina a factibilização do programa parcial **S**, na iteração j, não se pode garantir que exista o mesmo número de blocos que existia ao final da iteração $(j - 1)$, pois intervalos ociosos podem ter sido inseridos ou removidos ao longo do processo de factibilização. Também é preciso verificar se é possível alterar os intervalos ociosos inseridos de forma a diminuir o custo de **S**.

Este procedimento realiza ,basicamente, a contagem dos blocos de tarefas, após a inserção da j-ésima tarefa no programa parcial **S** e a atualização dos intervalos ociosos existentes entre esses blocos.

De acordo com a convenção adotada, seja $[k]$ o índice da tarefa que ocupa a k-ésima posição no programa parcial, $k = 1, 2, \ldots, j$.

O Procedimento III inicia fazendo uma contagem dos blocos de tarefas existentes de acordo com a Definição 3 (página 19). Simplesmente, deseja-se saber em quanto blocos se

divide **S** e quais as tarefas pertencentes a cada bloco. A rotina que realiza essa contagem também é responsável pela atualização dos valores de w_i para todas as tarefas J_i de **S**, de acordo com a Definição 2 (página 17).

Quando a contagem de blocos termina, seja t o índice do último bloco. A atualização de intervalos ociosos é feita em cada um dos blocos B_b, $b = t, t-1, \ldots, 1$. O Procedimento III faz a verificação em cada bloco para descobrir se o custo de processamento de suas tarefas pode ser diminuído através de um deslocamento à esquerda de sua posição atual. Nos cálculos de Δ_1 e Δ_2, deslocamentos à esquerda não são considerados e, no caso de Δ_3 e Δ_4, limita-se esse deslocamento ao mínimo necessário através de s_{aux}. Como não há um limitante explícito para os deslocamentos à direita das tarefas, entende-se que eles são favorecidos pelo Procedimento II e a atualização de intervalos ociosos pode levar em consideração apenas a diminuição de custo associada a deslocamentos das tarefas de B_b à esquerda.

Essa afirmação é testada no Capítulo 4, onde são apresentados os resultados computacionais.

Durante a atualização de intervalos ociosos, pode haver a divisão de um bloco em dois ou mais, ou a concatenação entre dois blocos. Na Figura 3.13, pode-se visualizar esses dois eventos quando é atualizado um bloco B_b do programa de produção.

Pode-se observar que, quando há uma divisão de um bloco, ou uma concatenação de dois blocos, o número total de blocos em **S** é alterado. Entretanto, esse novo valor não precisa ser levado em consideração pelo Procedimento III. Quando há a criação de um novo bloco, as tarefas pertencentes a ele não podem mais se deslocar à esquerda e, portanto, não é necessário considerá-las novamente. E, quando se concatenam dois blocos de tarefas, a atualização do bloco atual pára, e passa-se a atualizar o bloco anterior, que teve seu número de tarefas aumentado. Dessa forma, garante-se que nenhuma tarefa é desprezada na atualização descrita e, também, evita-se a avaliação desnecessária dos blocos criados com tarefas que não podem mais se deslocar.

Portanto, o número total de blocos no programa **S** só é importante na inicialização do índice b. A partir da primeira atualização de intervalos ociosos, o Procedimento III só precisa considerar qual o índice do bloco sendo analisado na iteração atual.

O mecanismo de atualização de intervalos ociosos pode ser descrito da seguinte maneira: tendo sido identificadas as tarefas que pertencem ao bloco B_b de **S**, procura-se, entre essas tarefas, aquela que apresenta o menor valor de w. Ou seja, procura-se a tarefa que pode se deslocar o mínimo possível à esquerda, calculando-se $w_{min} = \min_{i \in B_b}\{w_i\}$. Seja m a posição da tarefa J_i tal que $w_{min} = w_i$, em **S**.

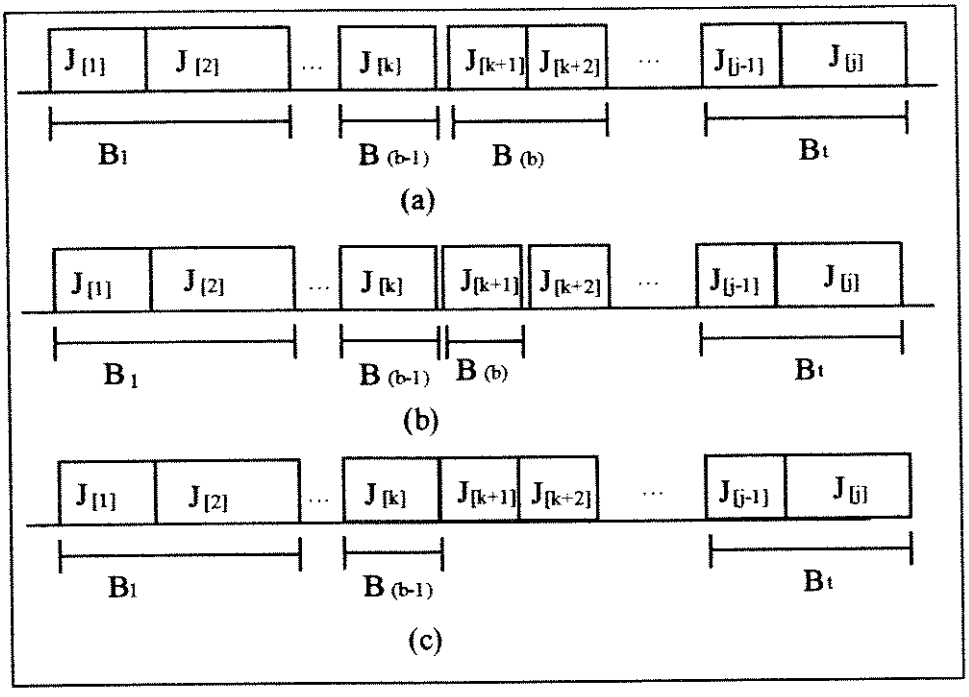

Figura 3.13 - a) Divisão de S em blocos após o Procedimento II
b) A tarefa $J_{[k+2]}$ não se desloca à esquerda, mas há uma diminuição do custo pelo deslocamento de $J_{[k+1]}$.
c) O deslocamento de B_b à esquerda provoca uma concatenação da tarefas $J_{[k]}$, $J_{[k+1]}$ e $J_{[k+2]}$ ao bloco $B_{(b-1)}$.

Calcula-se, então, a variação do custo do bloco B_b, se for feito um deslocamento desse bloco de tarefas, à esquerda, igual a w_{min}. Essa variação será igual a:

$$V = \sum_{i \in B_b} \left(\max\{E_i, T_i\}_{|c_i - w_{min}} - \max\{E_i, T_i\}_{|c_i} \right) \qquad (3.13)$$

$$V = \sum_{i \in B_b} \max\{t_i.(c_i - w_{min} - d_i), h_i.(d_i - c_i + w_{min})\} - \\ - \sum_{i \in B_b} \max\{t_i.(c_i - d_i), h_i.(d_i - c_i)\} \qquad (3.14)$$

Se $V < 0$, conclui-se que o deslocamento à esquerda trará uma diminuição do custo total de processamento do bloco de tarefas. Se $V \geq 0$, o deslocamento não reduz o custo e, portanto, não deve ser feito.

Apenas essa verificação inicial não é suficiente para garantir a melhor política de inserção de intervalos ociosos. Se a tarefa na qual se verifica o valor de w_{min} for a última do bloco, por exemplo, é preciso averiguar se, excluindo-se essa tarefa do bloco, as demais podem se deslocar à esquerda, diminuindo mais ainda o custo de processamento. Portanto, após se fazer a verificação do bloco todo, se a posição armazenada em m não corresponder

à da primeira tarefa do bloco, as tarefas posicionadas entre m e a última tarefa do bloco, no programa, são excluídas de B_b e novamente realizam-se os cálculos indicados acima para as tarefas restantes. O processo continua até que a posição marcada pela variável m seja igual à posição da primeira tarefa em B_b.

A cada vez que termina a análise de um bloco ou há uma concatenação entre dois blocos de tarefas, é feita uma recontagem dos blocos no programa, apenas para atualizar os valores de w_i para todas as tarefas J_i de **S**. Novamente, salienta-se que o novo número total de blocos não é considerado pelo Procedimento III.

Dentro do processo de construção, sempre que se insere uma tarefa no programa parcial, encerrando uma chamada ao Procedimento II, chama-se o Procedimento III para realizar a divisão de **S** em blocos e atualizar os intervalos ociosos inseridos.

Supondo-se que há j tarefas em **S**, a rotina de contagem de blocos pode ser formalizada da seguinte maneira:

Rotina Contagem

Passo 1: $B_1 = \{J_{[1]}\}, k = 1, t = 1;$
$w_{[1]} = s_{[1]} - r_{[1]}, I = 0;$

Passo 2: Se $k = j$, Pare.
Se $s_{[k+1]} = c_{[k]}$, $B_t = B_t \cup \{J_{[k+1]}\}$;
Se $s_{[k+1]} > c_{[k]}$, $B_{(t+1)} = \{J_{[k+1]}\}, t = t + 1$ e $I = c_{[k]}$;
$w_{[k+1]} = \min\{s_{[k+1]} - I, s_{[k+1]} - r_{[k+1]}\}$;
$k = k + 1$;
Repita o Passo 2.

Para cada bloco B_b de **S**, aplica-se a rotina de atualização de intervalos ociosos, formalizada como:

Rotina Atualiza (b)

Passo 1: Seja l a posição em **S** da primeira tarefa de B_b;

Passo 2: Encontre $w_{min} = \min_{J_i \in B_b}\{w_i\}$ e seja m a posição em **S** da tarefa J_i mais à esquerda, tal que $w_{min} = w_i$;

Passo 3: Calcule

$$V = \sum_{i \in B_b} \max\{t_i.(c_i - w_{min} - d_i), h_i.(d_i - c_i + w_{min})\} -$$

$$- \sum_{i \in B_b} \max\{t_i.(c_i - d_i), h_i.(d_i - c_i)\}$$

Passo 4: Se $V < 0$, faça
$s_i = s_i - w_{min}$, $c_i = c_i - w_{min}$ e
$w_i = w_i - w_{min}$ para $\forall J_i \in B_b$;

Passo 5: Se $m = 1$, Pare.
Caso contrário, reduza B_b, fazendo
$B_b = \{J_{[1]}, J_{[1+1]}, \ldots, J_{[m-1]}\}$ e volte para o Passo 2.

O Procedimento III pode ser, então, colocado na forma:

Procedimento III:

Passo 1: Contagem;
b = t;

Passo 2: Se b = 0, Pare.
Atualiza(b);
Contagem;
b = b - 1;
Repita o Passo 2;

APLICAÇÃO DOS PROCEDIMENTOS II E III AO EXEMPLO

Os dados do exemplo serão repetidos na tabela a seguir, juntamente com os valores calculados pelo Procedimento I.

Tarefa (J_i)	r_i	p_i	d_i	h_i	t_i	w_i	a_i	s_i	c_i
J_1	69	34	159	25	31	56	0	125	159
J_2	166	97	205	18	84	0	1	166	263
J_3	14	22	264	34	1	228	0	242	264
J_4	87	30	121	6	30	4	0	91	121
J_5	62	24	221	61	97	135	0	197	221
J_6	270	47	237	86	46	0	1	270	317
J_7	310	91	181	84	98	0	1	310	401
J_8	255	40	136	73	69	0	1	255	295

Tabela 3.3 - Dados referentes ao exemplo ilustrativo.

Do resultado da aplicação do Procedimento I tem-se que:

$\mathbf{A} = \{J_4, J_1, J_2, J_5, J_3, J_8, J_6, J_7\}$

$j = 1 \rightarrow J_{i(1)} = A_{[1]} = J_4$.

Procedimento II:

Passo 0: $B_1 = \{J_4\}$, $s_4 = 91$; $c_4 = 121$ e $w_4 = 4$. Pare.

Para a primeira tarefa inserida em **S**, não há necessidade de se fazer uma chamada ao Procedimento III.

$j = 2 \rightarrow J_{i(2)} = A_{[2]} = J_1$; $s_1 = 125$, $c_1 = 159$ e $w_1 = 56$.

Procedimento II:

Passo 0: $j > 1$;

Passo 1: $s_{i(2)} = s_1 > c_{[1]} = c_4 \Rightarrow B_2 = \{J_1\}$. Pare.

Procedimento III:

Passo 1: contagem

Passo 1: $B_1 = \{J_4\}$, $k = 1, t = 1$.

$w_4 = s_4 - r_4 = 4$, $I = 0$.

Passo 2: $k < j$;

$s_{[2]} = s_1 > c_{[1]} = c_4 \Rightarrow B_2 = \{J_1\}$ e $t = 2$;

$I = c_4 = 121$;

$w_1 = \min\{s_1 - I, s_1 - r_1\} = \min\{4,56\} = 4$.

$k = 2$.

Passo 2: $k = j$. Pare.

$b = 2$.

Passo 2: $b > 0$

Atualiza(2)

Passo 1: $l = 2$;

Passo 2: $w_{min} = 4$; $m = 2$;

Passo 3:

$V = \max\{t_1.(c_1 - d_1 - w_{min}), h_1.(d_1 - c_1 + w_{min}) -$

$- \max\{t_1.(c_1 - d_1), h_1.(d_1 - c_1)\}$

$V = \max\{31.(159 - 159 - 4), 25.(159 - 159 + 4)\} = 100$;

Passo 4: $V > 0$;

Passo 5: $m = 1$; Pare.

Como não houve deslocamento desse bloco, não é necessário exemplificar a recontagem de blocos.

$b = b - 1 = 1$.

Passo 2: $b > 0$

Atualiza(1)

Passo 1: $l = 1$;

Passo 2: $w_{min} = 4$; $m = 1$;

Passo 3:

$V = \max\{t_4.(c_4 - d_4 - w_{min}), h_4.(d_4 - c_4 + w_{min}) -$

$- \max\{t_1.(c_4 - d_4), h_4.(d_4 - c_4)\}$

$V = \max\{30.(121 - 121 - 4), 6.(121 - 121 + 4)\} = 24;$

Passo 4: $V > 0$;

Passo 5: $m = 1$; Pare.

Novamente, não houve deslocamento do bloco e não se mostrará o processo de recontagem.

$b = b - 1 = 0.$

Passo 2: $b = 0$, Pare.

O programa parcial, após a iteração 2 é:

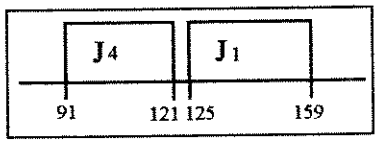

Figura 3.14 - Programa parcial quando $j = 2$.

$j = 3 \rightarrow J_{i(3)} = A_{[3]} = J_2; s_2 = 166, c_2 = 263$ e $w_2 = 0.$

Procedimento II:

Passo 0: $j > 1$;

Passo 1: $s_{i(3)} = s_2 > c_{[2]} = c_1 \Rightarrow B_3 = \{J_2\}$. Pare.

Procedimento III:

Passo 1: contagem

Passo 1: $B_1 = \{J_4\}$, $k = 1, t = 1.$

$w_4 = s_4 - r_4 = 4, I = 0.$

Passo 2: $k < j$;

$s_{[2]} = s_1 > c_{[1]} = c_4 \Rightarrow B_2 = \{J_1\}$ e $t = 2$;

$I = c_4 = 121$;

$w_1 = \min\{s_1 - I, s_1 - r_1\} = \min\{4,56\} = 4.$

$k = 2.$

Passo 2: $k < j$;

$s_{[3]} = s_2 > c_{[2]} = c_1 \Rightarrow B_3 = \{J_2\}$ e $t = 3$;

$I = c_1 = 159$;

$w_2 = \min\{s_2 - I, s_2 - r_2\} = \min\{7,0\} = 0.$

$k = 3.$

Passo 2: k = j.Pare.

Passo 2: b = 3;

Atualiza(3)

Passo 1: l = 3;

Passo 2: $w_{min} = 0$; m = 3;

Passo 3:

$V = \max\{t_2.(c_2 - d_2 - w_{min}), h_2.(d_2 - c_2 + w_{min}) -$

$- \max\{t_2.(c_2 - d_2), h_2.(d_2 - c_2)\}$

Como $w_{min} = 0$, V = 0;

Passo 4: V = 0;

Passo 5: m = 1; Pare.

Como não houve deslocamento no bloco B_3, não há necessidade de uma nova contagem de blocos do programa parcial. Além disso, a atualização dos blocos B_2 e B_1 é idêntica à realizada na iteração 2.

Ao final da iteração 3, o programa parcial **S** fica:

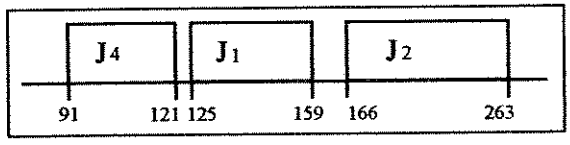

Figura 3.15 - Programa parcial quando j = 3.

$j = 4 \rightarrow J_{i(4)} = A_{[4]} = J_5$; $s_5 = 197$, $c_5 = 221$ e $w_5 = 135$.

Procedimento II:

Passo 0: j > 1;

Passo 1: $s_{i(4)} = s_5 < c_{[3]} = c_2$;

Passo 2: $s_5 > c_4 = c_{[1]}$; $s_5 > c_1 = c_{[2]}$;

$s_5 < c_2 = c_{[3]} \Rightarrow k = 3$;

Passo 3: $\Delta_1 = t_{[3]}.(c_{i(4)} - s_{[3]}) = t_2.(c_5 - s_2) =$

$= 84.(221 - 166) = 4620$;

$\Delta_2 = t_{i(4)}.(c_{[3]} - s_{i(4)}) = t_5.(c_2 - s_5) =$

$= 97.(263 - 197) = 6402$;

$a_{i(4)} = a_5 = 0 \Rightarrow s_{aux} = \max\{c_{[2]}, r_{i(4)}, s_{[3]} - p_{i(4)}\} =$

$= \max\{c_2, r_5, s_2 - p_5\} = \max\{159,62,142\} = 159$;

$\Delta_3 = t_{[3]}.(s_{aux} + p_{i(4)} - s_{[3]}) + h_{i(4)}.(s_{i(4)} - s_{aux})$

$= t_2.(159 + p_5 - s_2) + h_5.(s_5 - 159)$

$= 84.(159 - 142) + 61.(197 - 159) = 3746.$

Como $a_{[3]} = a_2 = 1 \Rightarrow \Delta_4 = M$.

$\Delta = \min\{\Delta_1, \Delta_2, \Delta_3, \Delta_4\} = \Delta_3$;

Passo 4: $s_{i(4)} = s_{aux} \Rightarrow s_5 = 159$;

$c_{i(4)} = s_{aux} + p_{i(4)} \Rightarrow c_5 = 159 + p_5 = 183$;

$s_{[3]} = c_{i(4)} \Rightarrow s_2 = c_5 = 183$;

$c_{[3]} = s_{[3]} + p_{[3]} \Rightarrow c_2 = s_2 + p_2 = 280$;

Passo 5: $J_{i(4)} = J_2$ e $J_{[3]} = J_5$.

$k = 3 = j - 1$. Pare.

Procedimento III:

Passo 1: contagem

Passo 1: $B_1 = \{J_4\}$, $k = 1, t = 1$.

$w_4 = s_4 - r_4 = 4, I = 0$.

Passo 2: $k < j$;

$s_{[2]} = s_1 > c_{[1]} = c_4 \Rightarrow B_2 = \{J_1\}$ e $t = 2$;

$I = c_4 = 121$;

$w_1 = \min\{s_1 - I, s_1 - r_1\} = \min\{4,56\} = 4$.

$k = 2$.

Passo 2: $k < j$;

$s_{[3]} = s_5 = c_{[2]} = c_1 \Rightarrow B_2 = \{J_1, J_5\}$;

$w_5 = \min\{s_5 - I, s_5 - r_5\} = \min\{38,97\} = 38$.

$k = 3$.

Passo 2: $k < j$;

$s_{[4]} = s_2 = c_{[3]} = c_5 \Rightarrow B_2 = \{J_1, J_5, J_2\}$;

$w_2 = \min\{s_2 - I, s_2 - r_2\} = \min\{62,17\} = 17$.

$k = 4$.

Passo 2: $k = j$. Pare.

Passo 2: $b = 2$;

Atualiza(2)

Passo 1: $l = 2$;

Passo 2: $w_{min} = 4$; $m = 2$;

Passo 3:

$V = \max\{31.(159 - 159 - 4), 25.(159 - 159 + 4)\} +$

$\max\{97.(183 - 221 - 4), 61.(221 - 183 + 4)\} -$

$\max\{97.(183 - 221), 61.(221 - 183)\} +$

$\max\{84.(280 - 205 - 4), 18.(205 - 280 + 4)\} -$

$\max\{84.(280 - 205), 18.(205 - 280)\} = 8$

Passo 4: $V > 0$;

Passo 5: $m = 1$; Pare.

Como não houve deslocamento no bloco B_2, não se realiza nova contagem de blocos. Além disso, a aplicação do Procedimento III ao bloco B_1 é idêntica à realizada na iteração 2.

O programa parcial **S** fica, então:

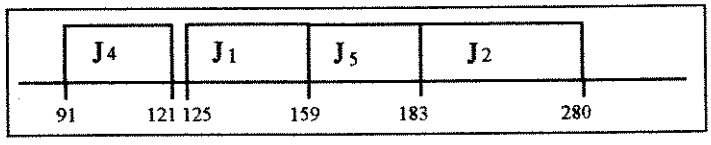

Figura 3.16 - Programa parcial quando $j = 4$.

$j = 5 \rightarrow J_{i(5)} = A_{[5]} = J_3$; $s_3 = 242$, $c_3 = 264$ e $w_3 = 228$.

Procedimento II:

Passo 0: $j > 1$;

Passo 1: $s_{i(5)} = s_3 < c_{[4]} = c_2$;

Passo 2: $s_3 > c_4 = c_{[1]}$; $s_3 > c_1 = c_{[2]}$;

$s_3 > c_5 = c_{[3]}$; $s_3 < c_2 = c_{[4]} \Rightarrow k = 4$;

Passo 3: $\Delta_1 = t_{[4]}.(c_{i(5)} - s_{[4]}) = t_2.(c_3 - s_2) =$

$= 84.(264 - 183) = 6804$;

$\Delta_2 = t_{i(5)}.(c_{[4]} - s_{i(5)}) = t_3.(c_2 - s_3) =$

$= 1.(280 - 242) = 38$;

$a_{i(5)} = a_3 = 0 \Rightarrow s_{aux} = \max\{c_{[3]}, r_{i(5)}, s_{[4]} - p_{i(5)}\}$

$= \max\{c_5, r_3, s_2 - p_3\} = \max\{183,14,161\} = 183$;

$\Delta_3 = t_{[4]}.(s_{aux} + p_{i(5)} - s_{[4]}) + h_{i(5)}.(s_{i(5)} - s_{aux})$

$= t_2.(183 + p_3 - s_2) + h_3.(s_3 - 183)$

$= 84.(183 - 161) + 34.(242 - 183) = 3854$.

Como $a_{[4]} = a_2 = 1 \Rightarrow \Delta_4 = M$.

$\Delta = \min\{\Delta_1, \Delta_2, \Delta_3, \Delta_4\} = \Delta_2$;

Passo 4: $s_{i(5)} = c_{[4]} \Rightarrow s_3 = c_2 = 280$;

$c_{i(5)} = s_{i(5)} + p_{i(5)} \Rightarrow c_3 = s_3 + p_3 = 302$;

$k = 5 = j$; Pare.

Procedimento III:

Passo 1: contagem

Passo 1: $B_1 = \{J_4\}$, $k = 1, t = 1$.

$w_4 = s_4 - r_4 = 4, I = 0$.

Passo 2: $k < j$;

$s_{[2]} = s_1 > c_{[1]} = c_4 \Rightarrow B_2 = \{J_1\}$ e $t = 2$;

$I = c_4 = 121$;

$w_1 = \min\{s_1 - I, s_1 - r_1\} = \min\{4,56\} = 4$.

$k = 2$.

Passo 2: $k < j$;

$s_{[3]} = s_5 = c_{[2]} = c_1 \Rightarrow B_2 = \{J_1, J_5\}$;

$w_5 = \min\{s_5 - I, s_5 - r_5\} = \min\{38,97\} = 38$.

$k = 3$.

Passo 2: $k < j$;

$s_{[4]} = s_2 = c_{[3]} = c_5 \Rightarrow B_2 = \{J_1, J_5, J_2\}$;

$w_2 = \min\{s_2 - I, s_2 - r_2\} = \min\{62,17\} = 17$.

$k = 4$.

Passo 2: $k < j$;

$s_{[5]} = s_3 = c_{[4]} = c_2 \Rightarrow B_2 = \{J_1, J_5, J_2, J_3\}$;

$w_3 = \min\{s_3 - I, s_3 - r_3\} = \min\{159,266\} = 159$.

$k = 5$.

Passo 2: $k = j$. Pare.

Passo 2: $b = 2$;

Atualiza(2)

Passo 1: $l = 2$;

Passo 2: $w_{min} = 4$; $m = 2$;

Passo 3: As tarefas J_1, J_5 e J_2 não tiveram suas posições alteradas. Portanto, basta somar o valor da contribuição de J_3 ao valor da variação de custo do bloco B_2:

$V = 8 + \max\{(1.(302 - 264 - 4), 34.(264 - 302 + 4)\} -$

$\max\{(1.(302 - 264), 34.(264 - 302)\} = 4$

Passo 4: $V > 0$;

Passo 5: $m = 1$; Pare.

Novamente, não houve deslocamento do bloco B_2, não se faz a recontagem do blocos e a aplicação do Procedimento III ao bloco B_1 fica como nas iterações anteriores.

O programa **S** torna-se, então:

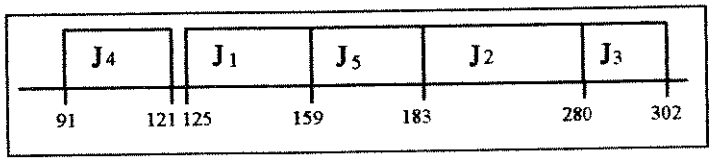

Figura 3.17 - Programa parcial quando $j = 5$.

$j = 6 \rightarrow J_{i(6)} = A_{[6]} = J_8$; $s_8 = 255$, $c_8 = 295$ e $w_8 = 0$.

Procedimento II:

Passo 0: $j > 1$;

Passo 1: $s_{i(6)} = s_8 < c_{[5]} = c_3$;

Passo 2: $s_8 > c_4 = c_{[1]}$; $s_8 > c_1 = c_{[2]}$;

$s_8 > c_5 = c_{[3]}$; $s_8 < c_2 = c_{[4]} \Rightarrow k = 4$;

Passo 3: $\Delta_1 = t_{[4]}.(c_{i(6)} - s_{[4]}) = t_2.(c_8 - s_2) =$

$= 84.(295 - 183) = 9184$;

$\Delta_2 = t_{i(6)}.(c_{[4]} - s_{i(6)}) = t_8.(c_2 - s_8) =$

$= 69.(280 - 255) = 1725$;

$a_{i(6)} = a_8 = 1 \Rightarrow \Delta_3 = M$;

$a_{[4]} = a_2 = 1 \Rightarrow \Delta_4 = M$;

$\Delta = \min\{\Delta_1, \Delta_2, \Delta_3, \Delta_4\} = \Delta_2$;

Passo 4: $s_{i(6)} = c_{[4]} \Rightarrow s_8 = c_2 = 280$;

$c_{i(6)} = s_{i(6)} + p_{i(6)} \Rightarrow c_8 = s_8 + p_8 = 320$;

$k = 5 < j$;

Passo 3: $\Delta_1 = t_{[5]}.(c_{i(6)} - s_{[5]}) = t_3.(c_8 - s_3) = 1.(320 - 280) = 40$;

$\Delta_2 = t_{i(6)}.(c_{[5]} - s_{i(6)}) = t_8.(c_3 - s_8) =$

$= 69.(302 - 280) = 1518$;

$a_{i(6)} = a_8 = 1 \Rightarrow \Delta_3 = M$;

$a_{[5]} = a_3 = 0 \Rightarrow$

$s_{aux} = \max\{c_{[4]}, r_{[5]}, s_{i(6)} - p_{[5]}\} = \max\{c_2, r_3, s_8 - p_3\}$

$= \max\{280,276,258\} = 280$;

$\Delta_4 = t_{i(6)}.(s_{aux} + p_{[5]} - s_{i(6)}) + h_{[5]}.(s_{[5]} - s_{aux})$

$= t_8.(280 + p_3 - s_8) + h_3.(s_3 - 280)$

$= 69.(280 - 258) + 34.(280 - 280) = 1518$;

$\Delta = \min\{\Delta_1, \Delta_2, \Delta_3, \Delta_4\} = \Delta_1$;

Passo 4: $s_{[5]} = c_{i(6)} \Rightarrow s_3 = c_8 = 320$;

$c_{[5]} = s_{[5]} + p_{[5]} \Rightarrow c_3 = s_3 + p_3 = 342$;

Passo 5: $J_{i(6)} = J_3$ e $J_{[5]} = J_8$;

$k = 5 = j - 1$; Pare.

Procedimento III:

Passo 1: contagem

Passo 1: $B_1 = \{J_4\}$, $k = 1, t = 1$.

$w_4 = s_4 - r_4 = 4, I = 0$.

Passo 2: $k < j$;

$s_{[2]} = s_1 > c_{[1]} = c_4 \Rightarrow B_2 = \{J_1\}$ e $t = 2$;

$I = c_4 = 121$;

$w_1 = \min\{s_1 - I, s_1 - r_1\} = \min\{4,56\} = 4$.

$k = 2$.

Passo 2: $k < j$;

$s_{[3]} = s_5 = c_{[2]} = c_1 \Rightarrow B_2 = \{J_1, J_5\}$;

$w_5 = \min\{s_5 - I, s_5 - r_5\} = \min\{38,97\} = 38$.

$k = 3$.

Passo 2: $k < j$;

$s_{[4]} = s_2 = c_{[3]} = c_5 \Rightarrow B_2 = \{J_1, J_5, J_2\}$;

$w_2 = \min\{s_2 - I, s_2 - r_2\} = \min\{62,17\} = 17$.

$k = 4$.

Passo 2: $k < j$;

$s_{[5]} = s_8 = c_{[4]} = c_2 \Rightarrow B_2 = \{J_1, J_5, J_2, J_8\}$;

$w_8 = \min\{s_8 - I, s_8 - r_8\} = \min\{159,25\} = 25$.

$k = 5$.

Passo 2: $k < j$;

$s_{[6]} = s_3 = c_{[5]} = c_8 \Rightarrow B_2 = \{J_1, J_5, J_2, J_8, J_3\}$;

$w_3 = \min\{s_3 - I, s_3 - r_3\} = \min\{199,306\} = 199$.

$k = 6$.

Passo 2: $k = j$. Pare.

Passo 2: $b = 2$;

Atualiza(2)

Passo 1: $l = 2$;

Passo 2: $w_{min} = 4$; $m = 2$;

Passo 3: Novamente, a contribuição das tarefas J_1, J_5 e J_2 para o valor de V não muda, pois essas tarefas não tiveram suas posições alteradas. É preciso, então, somar apenas as contribuições de J_8 e J_3 ao valor calculado na iteração $j = 4$:

$V = 8 + \max\{69.(320 - 136 - 4), 73.(136 - 320 + 4)\} -$
$\max\{69.(320 - 136), 73.(136 - 320)\} +$
$\max\{1.(342 - 264 - 4), 34.(264 - 342 + 4)\} -$
$\max\{1.(342 - 264), 34.(264 - 342)\} = -272$

Passo 4: $V < 0$;

$w_1 = 4 - w_{min} = 0$;
$s_1 = s_1 - w_{min} = 121$;
$c_1 = c_1 - w_{min} = 155$;

$w_5 = 38 - w_{min} = 34$;
$s_5 = s_5 - w_{min} = 155$;
$c_5 = c_5 - w_{min} = 179$;

$w_2 = 17 - w_{min} = 13$;
$s_2 = s_2 - w_{min} = 179$;
$c_2 = c_2 - w_{min} = 276$;

$w_8 = 25 - w_{min} = 21$;
$s_8 = s_8 - w_{min} = 276$;
$c_8 = c_8 - w_{min} = 316$;

$w_3 = 199 - w_{min} = 195$;
$s_3 = s_3 - w_{min} = 316$;
$c_3 = c_3 - w_{min} = 338$;

Passo 5: $m = 1$; Pare.

contagem

Passo 1: $B_1 = \{J_4\}$, $k = 1, t = 1$.
$w_4 = s_4 - r_4 = 4, I = 0$.

Passo 2: $k < j$;
$s_{[2]} = s_1 = c_{[1]} = c_4 \Rightarrow B_1 = \{J_4, J_1\}$;
$w_1 = \min\{s_1 - I, s_1 - r_1\} = \min\{121,52\} = 52$.
$k = 2$.

Passo 2: $k < j$;
$s_{[3]} = s_5 = c_{[2]} = c_1 \Rightarrow B_1 = \{J_4, J_1, J_5\}$;
$w_5 = \min\{s_5 - I, s_5 - r_5\} = \min\{155,93\} = 93$.
$k = 3$.

Passo 2: $k < j$;
$s_{[4]} = s_2 = c_{[3]} = c_5 \Rightarrow B_1 = \{J_4, J_1, J_5, J_2\}$;
$w_2 = \min\{s_2 - I, s_2 - r_2\} = \min\{179,13\} = 13$.
$k = 4$.

Passo 2: $k < j$;
$s_{[5]} = s_8 = c_{[4]} = c_2 \Rightarrow B_1 = \{J_4, J_1, J_5, J_2, J_8\}$;

$w_8 = \min\{s_8 - I, s_8 - r_8\} = \min\{276,21\} = 21.$

$k = 5.$

Passo 2: $k < j$;

$s_{[6]} = s_3 = c_{[5]} = c_8 \Rightarrow B_1 = \{J_4, J_1, J_5, J_2, J_8, J_3\};$

$w_3 = \min\{s_3 - I, s_3 - r_3\} = \min\{316,302\} = 302.$

$k = 6.$

Passo 2: $k = j$. Pare.

$b = b - 1 = 1;$

Passo 2: $b > 0$

Atualiza(1)

Passo 1: $l = 1;$

Passo 2: $w_{min} = 4;\ m = 1;$

Passo 3:

$V = \max\{30.(121 - 121 - 4), 6.(121 - 121 + 4)\}$
$+ \max\{31.(155 - 159 - 4), 25.(159 - 155 + 4)\}$
$- \max\{31.(155 - 159), 25.(159 - 155)\}$
$+ \max\{97.(179 - 221 - 4), 61.(221 - 179 + 4)\}$
$- \max\{97.(179 - 221), 61.(221 - 179)\}$
$+ \max\{84.(276 - 205 - 4), 18.(205 - 276 + 4)\}$
$- \max\{84.(276 - 205), 18.(205 - 276)\}$
$+ \max\{69.(316 - 136 - 4), 73.(136 - 316 + 4)\}$
$- \max\{69.(316 - 136), 73.(136 - 316)\}$
$+ \max\{1.(338 - 264 - 4), 34.(264 - 338 + 4)\}$
$- \max\{1.(338 - 264), 34.(264 - 338)\} = -248$

Passo 4: $V < 0;$

$w_4 = 4 - w_{min} = 0;$
$s_4 = s_4 - w_{min} = 87;$
$c_4 = c_4 - w_{min} = 117;$

$w_1 = 52 - w_{min} = 48;$
$s_1 = s_1 - w_{min} = 117;$
$c_1 = c_1 - w_{min} = 151;$

$w_5 = 93 - w_{min} = 89;$
$s_5 = s_5 - w_{min} = 151;$
$c_5 = c_5 - w_{min} = 175;$

$w_2 = 13 - w_{min} = 9;$
$s_2 = s_2 - w_{min} = 175;$
$c_2 = c_2 - w_{min} = 272;$

$w_8 = 21 - w_{min} = 17;$

$s_8 = s_8 - w_{min} = 272;$

$c_8 = c_8 - w_{min} = 312;$

$w_3 = 302 - w_{min} = 298;$

$s_3 = s_3 - w_{min} = 312;$

$c_3 = c_3 - w_{min} = 334;$

Passo 5: m = 1; Pare.

Nesse ponto, apesar de ter havido um deslocamento no bloco B_1, não há necessidade de se mostrar a recontagem de blocos pois não há nenhuma tarefa que não tenha sido considerada.

b = b - 1 = 0;

Passo 2: b = 0, Pare.

O programa **S** torna-se, então:

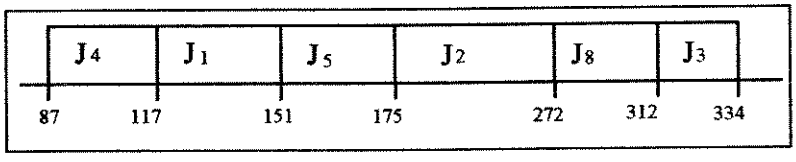

Figura 3.18 - Programa parcial quando j = 6.

A partir dessa iteração, a não ser que a tarefa J_4 deixe de ocupar a primeira posição em **S**, serão omitidas as chamadas ao Procedimento III.

$j = 7 \rightarrow J_{i(7)} = A_{[7]} = J_6; s_6 = 270, c_6 = 317$ e $w_6 = 0.$

Procedimento II:

Passo 0: j > 1;

Passo 1: $s_{i(7)} = s_6 < c_{[6]} = c_3;$

Passo 2: $s_6 > c_4 = c_{[1]};$ $s_6 > c_1 = c_{[2]};$

$s_6 > c_5 = c_{[3]};$ $s_6 < c_2 = c_{[4]} \Rightarrow k = 4;$

Passo 3: $\Delta_1 = t_{[4]}.(c_{i(7)} - s_{[4]}) = t_2.(c_6 - s_2) =$

$= 84.(317 - 175) = 11928;$

$\Delta_2 = t_{i(7)}.(c_{[4]} - s_{i(7)}) = t_6.(c_2 - s_6) =$

$= 46.(272 - 270) = 92;$

$a_{i(7)} = a_6 = 1 \Rightarrow \Delta_3 = M;$

$a_{[4]} = a_2 = 1 \Rightarrow \Delta_4 = M;$

$\Delta = \min\{\Delta_1, \Delta_2, \Delta_3, \Delta_4\} = \Delta_2;$

Passo 4: $s_{i(7)} = c_{[4]} \Rightarrow s_6 = c_2 = 272;$

$c_{i(7)} = s_{i(7)} + p_{i(7)} \Rightarrow c_6 = s_6 + p_6 = 319;$

$k = 5 < j;$

Passo 3: $\Delta_1 = t_{[5]}.(c_{i(7)} - s_{[5]}) = t_8.(c_6 - s_8) =$

$= 69.(319 - 272) = 3243;$

$\Delta_2 = t_{i(7)}.(c_{[5]} - s_{i(7)}) = t_6.(c_8 - s_6) =$

$= 46.(312 - 272) = 1840;$

$a_{i(7)} = a_6 = 1 \Rightarrow \Delta_3 = M;$

$a_{[5]} = a_8 = 1 \Rightarrow \Delta_4 = M;$

$\Delta = \min\{\Delta_1, \Delta_2, \Delta_3, \Delta_4\} = \Delta_2;$

Passo 4: $s_{i(7)} = c_{[5]} \Rightarrow s_6 = c_8 = 312;$

$c_{i(7)} = s_{i(7)} + p_{i(7)} \Rightarrow c_6 = s_6 + p_6 = 359;$

$k = 6 < j;$

Passo 3: $\Delta_1 = t_{[6]}.(c_{i(7)} - s_{[6]}) = t_3.(c_6 - s_3) =$

$= 1.(359 - 312) = 47;$

$\Delta_2 = t_{i(7)}.(c_{[6]} - s_{i(7)}) = t_6.(c_3 - s_6) =$

$= 46.(334 - 312) = 1012;$

$a_{i(7)} = a_6 = 1 \Rightarrow \Delta_3 = M;$

$a_{[6]} = a_3 = 0 \Rightarrow$

$s_{aux} = \max\{c_{[5]}, r_{[6]}, s_{i(7)} - p_{[6]}\}$

$= \max\{c_8, r_3, s_6 - p_3\} = \max\{312,14,290\} = 312;$

$\Delta_4 = t_{i(7)}.(s_{aux} + p_{[6]} - s_{i(7)}) + h_{[6]}.(s_{[6]} - s_{aux})$

$= t_6.(312 + p_3 - s_6) + h_3.(s_3 - 312)$

$= 46.(312 - 290) + 0 = 1012;$

$\Delta = \min\{\Delta_1, \Delta_2, \Delta_3, \Delta_4\} = \Delta_1;$

Passo 4: $s_{[6]} = c_{i(7)} \Rightarrow s_3 = c_6 = 359;$

$c_{[6]} = s_{[6]} + p_{[6]} \Rightarrow c_3 = s_3 + p_3 = 381;$

Passo 5: $J_{i(7)} = J_3$ e $J_{[6]} = J_6;$

$k = 6 = j - 1;$ Pare.

Após a execução do Procedimento III, o programa parcial toma a seguinte forma:

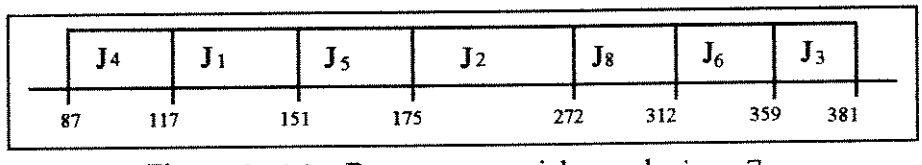

Figura 3.19 - Programa parcial quando $j = 7$.

$j = 8 \rightarrow J_{i(8)} = A_{[5]} = J_7$; $s_7 = 310$, $c_7 = 401$ e $w_7 = 0$.

Procedimento II:

Passo 0: $j > 1$;

Passo 1: $s_{i(8)} = s_7 < c_{[7]} = c_3$;

Passo 2: $s_7 > c_4 = c_{[1]}$; $s_7 > c_1 = c_{[2]}$;

$s_7 > c_5 = c_{[3]}$; $s_7 > c_2 = c_{[4]}$;

$s_7 > c_8 = c_{[5]} \Rightarrow k = 5$;

Passo 3: $\Delta_1 = t_{[5]} \cdot (c_{i(8)} - s_{[5]}) = t_8 \cdot (c_7 - s_8) =$

$= 69 \cdot (401 - 272) = 8901$;

$\Delta_2 = t_{i(8)} \cdot (c_{[5]} - s_{i(8)}) = t_7 \cdot (c_8 - s_7) =$

$= 98 \cdot (312 - 310) = 196$;

$a_{i(8)} = a_7 = 1 \Rightarrow \Delta_3 = M$;

$a_{[5]} = a_8 = 1 \Rightarrow \Delta_4 = M$;

$\Delta = \min\{\Delta_1, \Delta_2, \Delta_3, \Delta_4\} = \Delta_2$;

Passo 4: $s_{i(8)} = c_{[5]} \Rightarrow s_7 = c_8 = 312$;

$c_{i(8)} = s_{i(8)} + p_{i(8)} \Rightarrow c_7 = s_7 + p_7 = 403$;

$k = 6 < j$;

Passo 3: $\Delta_1 = t_{[6]} \cdot (c_{i(8)} - s_{[6]}) = t_6 \cdot (c_7 - s_6) =$

$= 46 \cdot (403 - 312) = 4186$;

$\Delta_2 = t_{i(8)} \cdot (c_{[6]} - s_{i(8)}) = t_7 \cdot (c_6 - s_7) =$

$= 98 \cdot (359 - 312) = 4606$;

$a_{i(8)} = a_7 = 1 \Rightarrow \Delta_3 = M$;

$a_{[6]} = a_6 = 1 \Rightarrow \Delta_4 = M$;

$\Delta = \min\{\Delta_1, \Delta_2, \Delta_3, \Delta_4\} = \Delta_1$;

Passo 4: $s_{[6]} = c_{i(8)} \Rightarrow s_6 = c_7 = 403$;

$c_{[6]} = s_{[6]} + p_{[6]} \Rightarrow c_6 = s_6 + p_6 = 450$;

Após esse passo, o programa parcial apresenta uma infactibilidade como a que foi prevista para quando $\Delta = \Delta_1$ (Ver Figura 3.11). O programa **S** fica:

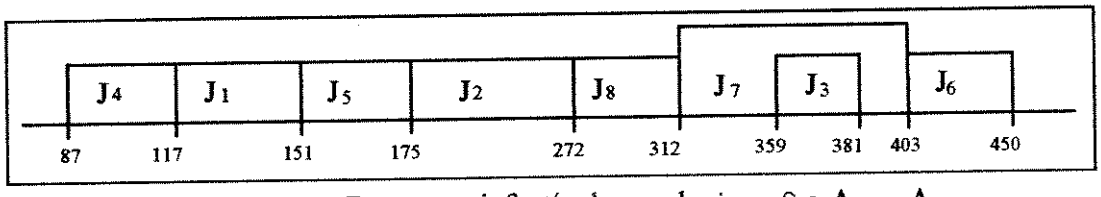

Figura 3.20 - Programa infactível quando $j = 8$ e $\Delta = \Delta_1$.

Passo 5: $J_{i(8)} = J_6$ e $J_{[6]} = J_7$;

$k = 6 < j - 1$;

$s_{i(8)} = s_6 > s_{[7]} = s_3$;

$A_{[8]} = J_6$, $s_6 = 270$; $c_6 = 317$; $w_6 = 0$;

Passo 6: $J_{i(8)} = J_7$;

$l = 6 \Rightarrow J_{[6]} = J_{[7]} = J_3$;

$l = 7 = j - 1 \Rightarrow J_{[7]} = J_{[8]} = J_7$;

$j = 7$.

Passo 3: $\Delta_1 = t_{[6]}.(c_{i(7)} - s_{[6]}) = t_3.(c_7 - s_3) =$

$= 1.(403 - 359) = 44$;

$\Delta_2 = t_{i(7)}.(c_{[6]} - s_{i(7)}) = t_7.(c_3 - s_7) =$

$= 98.(381 - 312) = 6762$;

$a_{i(7)} = a_7 = 1 \Rightarrow \Delta_3 = M$;

$a_{[6]} = a_3 = 0 \Rightarrow$

$s_{aux} = \max\{c_{[5]}, r_{[6]}, s_{i(7)} - p_{[6]}\}$

$= \max\{c_6, r_3, s_7 - p_3\} = \max\{312,14,298\} = 312$;

$\Delta_4 = t_{i(7)}.(s_{aux} + p_{[6]} - s_{i(7)}) + h_{[6]}.(s_{[6]} - s_{aux})$

$= t_7.(312 + p_3 - s_7) + h_3.(s_3 - 312)$

$= 98.(312 - 298) + 0 = 1372$;

$\Delta = \min\{\Delta_1, \Delta_2, \Delta_3, \Delta_4\} = \Delta_1$;

Passo 4: $s_{[6]} = c_{i(7)} \Rightarrow s_3 = c_7 = 403$;

$c_{[6]} = s_{[6]} + p_{[6]} \Rightarrow c_3 = s_3 + p_3 = 425$;

Passo 5: $J_{i(7)} = J_3$ e $J_{[6]} = J_7$;

$k = 6 = j - 1$; Pare.

Após a execução do Procedimento III, o programa parcial toma a seguinte forma:

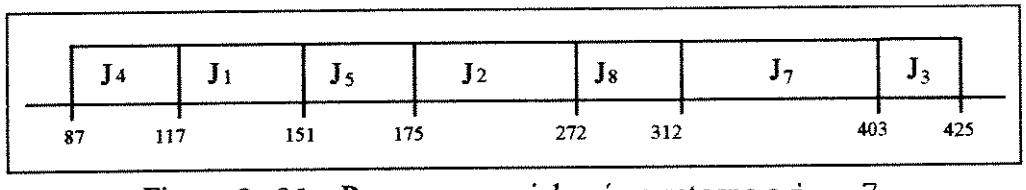

Figura 3.21 - Programa parcial após o retorno a $j = 7$.

$j = 8 \rightarrow J_{i(8)} = A_{[8]} = J_6$; $s_6 = 270$, $c_6 = 317$ e $w_6 = 0$.

Procedimento II:

Passo 0: $j > 1$;

Passo 1: $s_{i(8)} = s_6 < c_{[7]} = c_3$;

Passo 2: $s_6 > c_4 = c_{[1]}$; $s_6 > c_1 = c_{[2]}$;
$s_6 > c_5 = c_{[3]}$; $s_6 < c_2 = c_{[4]} \Rightarrow k = 4$;

Passo 3: $\Delta_1 = t_{[4]}.(c_{i(8)} - s_{[4]}) = t_2.(c_6 - s_2) =$
$= 84.(317 - 175) = 11928$;
$\Delta_2 = t_{i(8)}.(c_{[4]} - s_{i(8)}) = t_6.(c_2 - s_6) =$
$= 46.(272 - 270) = 92$;
$a_{i(8)} = a_6 = 1 \Rightarrow \Delta_3 = M$;
$a_{[4]} = a_2 = 1 \Rightarrow \Delta_4 = M$;
$\Delta = \min\{\Delta_1, \Delta_2, \Delta_3, \Delta_4\} = \Delta_2$;

Passo 4: $s_{i(8)} = c_{[4]} \Rightarrow s_6 = c_2 = 272$;
$c_{i(8)} = s_{i(8)} + p_{i(8)} \Rightarrow c_6 = s_6 + p_6 = 319$;
$k = 5 < j$;

Passo 3: $\Delta_1 = t_{[5]}.(c_{i(8)} - s_{[5]}) = t_8.(c_6 - s_8) =$
$= 69.(319 - 272) = 3243$;
$\Delta_2 = t_{i(8)}.(c_{[5]} - s_{i(8)}) = t_6.(c_8 - s_6) =$
$= 46.(312 - 272) = 1840$;
$a_{i(8)} = a_6 = 1 \Rightarrow \Delta_3 = M$;
$a_{[5]} = a_8 = 1 \Rightarrow \Delta_4 = M$;
$\Delta = \min\{\Delta_1, \Delta_2, \Delta_3, \Delta_4\} = \Delta_2$;

Passo 4: $s_{i(8)} = c_{[5]} \Rightarrow s_6 = c_8 = 312$;
$c_{i(8)} = s_{i(8)} + p_{i(8)} \Rightarrow c_6 = s_6 + p_6 = 359$;
$k = 6 < j$;

Passo 3: $\Delta_1 = t_{[6]}.(c_{i(8)} - s_{[6]}) = t_7.(c_6 - s_7) =$
$= 98.(359 - 312) = 4606$;
$\Delta_2 = t_{i(8)}.(c_{[6]} - s_{i(8)}) = t_6.(c_7 - s_6) =$
$= 46.(403 - 312) = 4186$;
$a_{i(8)} = a_6 = 1 \Rightarrow \Delta_3 = M$;
$a_{[6]} = a_7 = 1 \Rightarrow \Delta_4 = M$;
$\Delta = \min\{\Delta_1, \Delta_2, \Delta_3, \Delta_4\} = \Delta_2$;

Passo 4: $s_{i(8)} = c_{[6]} \Rightarrow s_6 = c_7 = 403$;
$c_{i(8)} = s_{i(8)} + p_{i(8)} \Rightarrow c_6 = s_6 + p_6 = 450$;
$k = 7 < j$;

Passo 3: $\Delta_1 = t_{[7]}.(c_{i(8)} - s_{[7]}) = t_3.(c_6 - s_3) =$
$= 1.(450 - 403) = 47$;
$\Delta_2 = t_{i(8)}.(c_{[7]} - s_{i(8)}) = t_6.(c_3 - s_6) =$
$= 46.(425 - 403) = 1012$;

$a_{i(8)} = a_6 = 1 \Rightarrow \Delta_3 = M;$

$a_{[7]} = a_3 = 0 \Rightarrow$

$s_{aux} = \max\{c_{[6]}, r_{[7]}, s_{i(8)} - p_{[7]}\}$

$= \max\{c_7, r_3, s_6 - p_3\} = \max\{403,14,381\} = 403;$

$\Delta_4 = t_{i(8)}.(s_{aux} + p_{[7]} - s_{i(8)}) + h_{[7]}.(s_{[7]} - s_{aux})$

$= t_6.(403 + p_3 - s_6) + h_3.(s_3 - 403)$

$= 46.(403 - 381) + 0 = 1012;$

$\Delta = \min\{\Delta_1, \Delta_2, \Delta_3, \Delta_4\} = \Delta_1;$

Passo 4: $s_{[7]} = c_{i(8)} \Rightarrow s_3 = c_6 = 450;$

$c_{[7]} = s_{[7]} + p_{[7]} \Rightarrow c_3 = s_3 + p_3 = 472;$

Passo 5: $J_{i(8)} = J_3;\ J_{[7]} = J_6;$

$k = 7 = j - 1.$ Pare.

Após a realização do Procedimento III, obtém-se o programa final **S**:

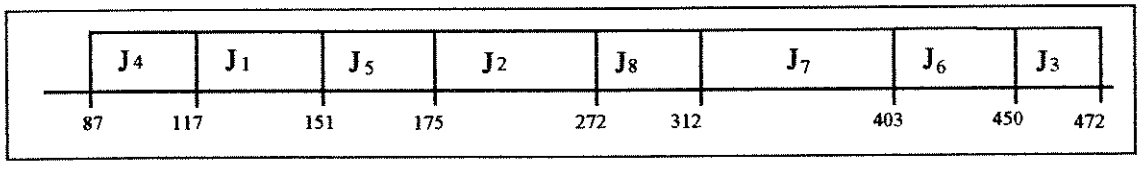

Figura 3.22 - Programa obtido ao final da fase construtiva.

A Tabela 3.4, a seguir, resume os resultados obtidos nessa fase da heurística, mostrando os custos individuais decorrentes do posicionamento encontrado para as tarefas.

Tarefa (J_i)	r_i	d_i	s_i	c_i	$custo_i$
J_4	87	121	87	117	24
J_1	69	159	117	151	200
J_5	62	221	151	175	2806
J_2	166	205	175	272	5628
J_8	255	136	272	312	12144
J_7	310	181	312	403	21756
J_6	270	237	403	450	9706
J_3	14	264	450	472	208

Custo total de **S** = 52472.

Tabela 3.4 - Resultados obtidos na fase construtiva da heurística.

Tendo sido completada a fase construtiva da heurística, passa-se à fase de busca em vizinhança. Na seção 3.3, volta-se ao estudo desse exemplo, para verificação dos pares adjacentes de tarefas em **S**.

3.2 - ANÁLISE DE PARES ADJACENTES

A inclusão de uma fase de busca em vizinhança na heurística tem como objetivo averiguar se houve algum erro de posicionamento durante a fase de construção. Para realizar essa verificação, escolheu-se uma estrutura de vizinhança do tipo Troca de Pares Adjacentes (TTA).

A vizinhança TTA de **S** é definida como o conjunto de seqüências **S'** que diferem de **S** pela troca de posição entre duas tarefas adjacentes, pertencentes ao mesmo bloco. Se **S** possui n tarefas, então o número de vizinhos é igual ao número de pares de tarefas adjacentes na seqüência. Ou seja, **S** possui, no máximo, $n - 1$ soluções vizinhas.

A busca nessa vizinhança visa encontrar uma solução **S'** que possua um custo total menor que o de **S**. Se essa solução **S'** existir, ela toma o lugar de **S**, e constrói-se uma nova vizinhança. O processo continua até que se chegue a um ótimo local.

O tempo gasto na exploração de uma vizinhança é proporcional ao tamanho desta e o número de vizinhanças exploradas depende da qualidade da solução inicial **S**. Para que a busca se torne mais rápida, é desejável que se encontre uma forma de, analisando-se apenas a seqüência **S**, descobrir quais pares de tarefas contribuiriam para a diminuição do custo total, se suas posições fossem trocadas.

Em OW e MORTON (1989) é apresentado um Teorema de Adjacência Ótima (TAO), que estabelece uma condição necessária e suficiente para que duas tarefas adjacentes não precisem ter suas posições trocadas no programa. O teorema apresentado é valido para o caso em que se tem $r_i = 0, \forall i \in \mathbf{N}$. Para que se pudesse aplicar esse resultado ao problema com $r_i \geq 0, \forall i \in \mathbf{N}$, foi necessário realizar uma generalização do TAO.

Nesta seção, apresenta-se a forma como foi feita a generalização citada, assim como uma explicação detalhada de todos os conceitos envolvidos na formulação do Teorema de Adjacência Ótima Generalizado.

Sejam duas tarefas adjacentes J_i e J_j, como as da Figura 3.23.

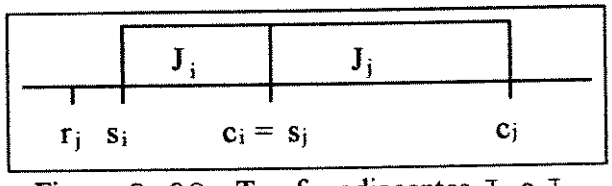

Figura 3.23 - Tarefas adjacentes J_i e J_j.

A Figura 3.23 mostra o caso em que J_j pode ter seu processamento iniciado num instante anterior a s_i, pois $r_j < s_i$. Mas, para que a troca de posição entre J_i e J_j não

afete o processamento das tarefas que terminam antes de J_i, assume-se que o máximo deslocamento de J_j, à esquerda, será aquele suficiente para que essa tarefa inicie no instante s_i.

Sendo assim, a variação no custo total de processamento, devida à troca, pode ser calculada como a variação nos custos de J_i e J_j, apenas.

Sejam então:

- custo(ij) = custo do processamento de J_i e J_j antes da troca;
- custo(ji) = custo do processamento de J_i e J_j após a troca;
- Δcusto(J_i) = variação no custo de J_i, devida à troca de posições, e
- Δcusto(J_j) = variação no custo de J_j, devida à troca de posições.

A variação no custo total será dada por:

$$\text{custo(ji)} - \text{custo(ij)} = \Delta\text{custo}(J_i) + \Delta\text{custo}(J_j). \quad (3.14)$$

Na Figura 3.23, não estão especificados os instantes r_i, d_i e d_j. Isso se deve ao fato de que existem muitas combinações entre os posicionamentos relativos desses valores. Essas combinações devem ser agrupadas para que se possa reduzir o trabalho de busca. Ou seja, deseja-se que a busca na vizinhança TTA de **S** se reduza à conferência da validade de uma expressão única.

A primeira análise a ser feita para alcançar esse objetivo diz respeito às medidas de deslocamento das tarefas, considerando a posição de r_j em relação a s_i.

Quando $r_j \leq s_i$, J_j se desloca de $(s_j - s_i) = p_i$, à esquerda, e J_i se desloca de uma distância igual a p_j, à direita (ver Figura 3.24).

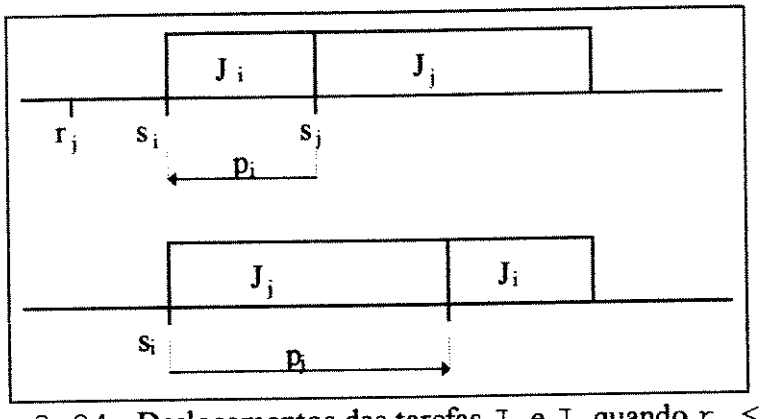

Figura 3.24 - Deslocamentos das tarefas J_i e J_j quando $r_j \leq s_i$.

Quando $r_j > s_i$, o deslocamento de J_j, à esquerda, é de $(s_j - r_j)$ e o deslocamento de J_i, à direita, é de $(p_j + r_j - s_i)$. Esses deslocamentos são mostrados na Figura 3.25.

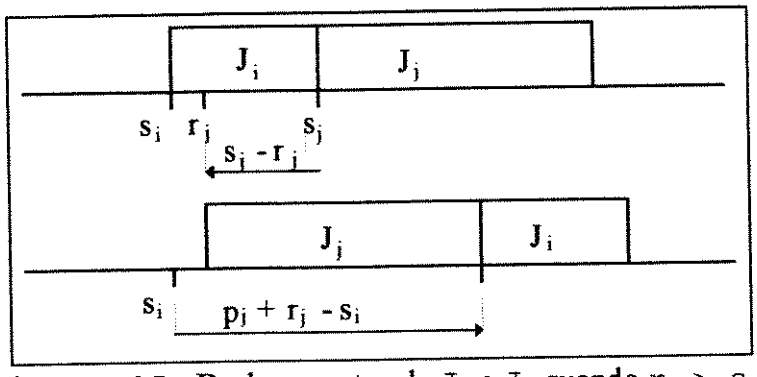

Figura 3.25 - Deslocamentos de J_i e J_j quando $r_j > s_i$.

Pode-se colocar então, os deslocamentos das tarefas numa forma fechada, da seguinte maneira:

$$\begin{cases} \text{Deslocamento de } J_i\text{: } p_j + \max\{0, r_j - s_i\}, \text{ e} \\ \text{Deslocamento de } J_j\text{: } \min\{p_i, s_j - r_j\} \end{cases} \qquad (3.15)$$

Resolvido o problema de como trabalhar com a posição relativa de r_j, resta encontrar uma forma de medir o avanço e o atraso de cada tarefa em relação a sua data de entrega, antes e depois de realizada a troca.

A forma encontrada por OW e MORTON(1989) para trabalhar com as diferentes configurações possíveis de avanço e atraso foi a criação de uma medida de folga para as tarefas envolvidas na análise de adjacência. Será apresentada uma interpretação dessa folga, para um melhor entendimento da generalização realizada.

A folga da tarefa J_i é calculada antes da troca e mede o quanto essa tarefa pode se deslocar à direita sem que haja atraso. Define-se a folga de J_i como :

$$\Delta_i = d_i - p_i - s_i \qquad (3.16)$$

e esse valor pode ser positivo, negativo ou nulo. São considerados três intervalos relevantes em que Δ_i pode estar.

- $\Delta_i < 0 \rightarrow$ isso significa que J_i está atrasada antes e depois da troca. Portanto, o deslocamento da tarefa J_i à direita só aumentará seu custo de atraso. A variação de custo será, então,

$$\Delta custo(J_i) = t_i.(p_j + \max\{0, r_j - s_i\}). \qquad (3.17)$$

A Figura 3.26 ilustra essa variação quando $r_j \leq s_i$.

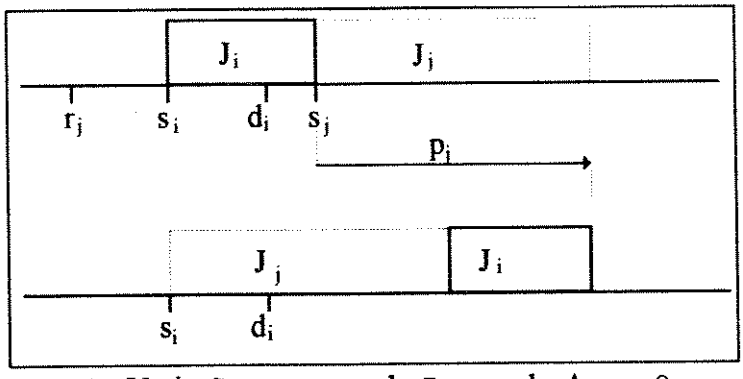

Figura 3.26 - Variação no atraso de J_i quando $\Delta_i < 0$ e $r_j \leq s_i$.

• $0 \leq \Delta_i < (p_j + \max\{0, r_j - s_i\}) \rightarrow$ ocorre o caso em que J_i está adiantada antes da troca. Mas, o deslocamento realizado é maior que o avanço de J_i. Portanto, parte desse deslocamento gera um custo de atraso. A variação de custo será de

$$\Delta custo(J_i) = t_i.(p_j + \max\{r_j - s_i\} - \Delta_i) - h_i.\Delta_i. \quad (3.18)$$

Na Figura 3.27, pode-se visualizar essa variação quando $r_j > s_i$.

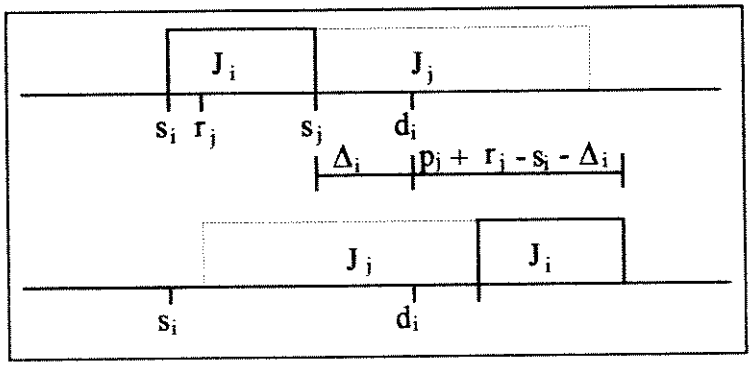

Figura 3.27 - Variação no avanço e no atraso de J_i quando $0 \leq \Delta_i < (p_j + \max\{0, r_j - s_i\})$ e $r_j > s_i$.

• $\Delta_i \geq (p_j + \max\{0, r_j - s_i\}) \rightarrow J_i$ está adiantada antes e depois da troca, pois o deslocamento à direita não é suficiente para anular todo o avanço dessa tarefa. Com a troca, ocorrerá somente uma diminuição do avanço, equivalente ao deslocamento realizado. A variação no custo será

$$\Delta custo(J_i) = -h_i.(p_j + \max\{0, r_j - s_i\}). \quad (3.19)$$

Novamente para o caso em que $r_j > s_i$, essa variação de custo é mostrada na Figura 3.28.

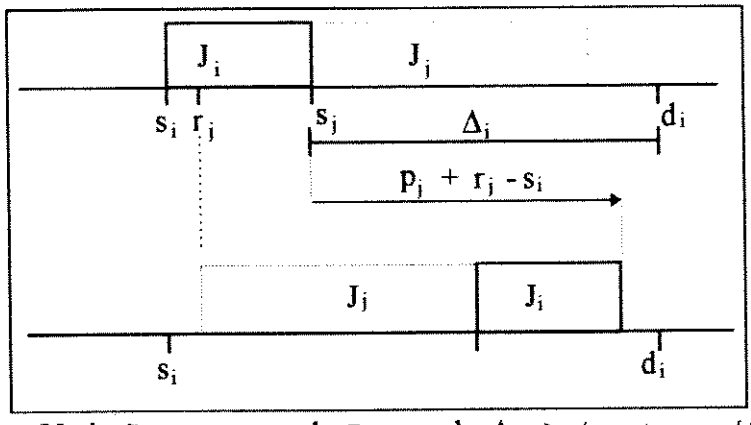

Figura 3.28 - Variação no avanço de J_i quando $\Delta_i \geq (p_j + \max\{0, r_j - s_i\})$.

Para se colocar o cálculo de $\Delta\text{custo}(J_i)$ numa fórmula fechada, faz-se:

$$\Omega_{ij} = \begin{cases} 0, & \text{se } \Delta_i < 0; \\ \Delta_i, & \text{se } 0 \leq \Delta_i < p_j + \max\{0, r_j - s_i\} \\ p_j + \max\{0, r_j - s_i\}, & \text{caso contrário} \end{cases} \qquad (3.20)$$

Das expressões (3.17) a (3.20), formula-se a variação de custo no movimento de J_i como:

$$\Delta\text{custo}(J_i) = t_i.(p_j + \max\{0, r_j - s_i\}) - \Omega_{ij}.(h_i + t_i) \qquad (3.21)$$

A folga da tarefa J_j é calculada depois da troca e mede o efeito do deslocamento realizado sobre o atraso e/ou avanço dessa tarefa. Define-se a folga de J_j como

$$\Delta_j = d_j - p_j - \max\{r_j, s_i\}. \qquad (3.22)$$

O valor de Δ_j pode estar em três intervalos análogos aos estudados para Δ_i.

- $\Delta_j < 0 \rightarrow$ isso significa que J_j está atrasada antes e depois da troca. O deslocamento dessa tarefa, à esquerda, terá o efeito de diminuir o custo de atraso envolvido no seu processamento. A variação nesse custo será de:

$$\Delta\text{custo}(J_j) = -t_j.\min\{p_i, s_j - r_j\}. \qquad (3.23)$$

A Figura 3.29, mostra essa diminuição de custo quando $r_j > s_i$.

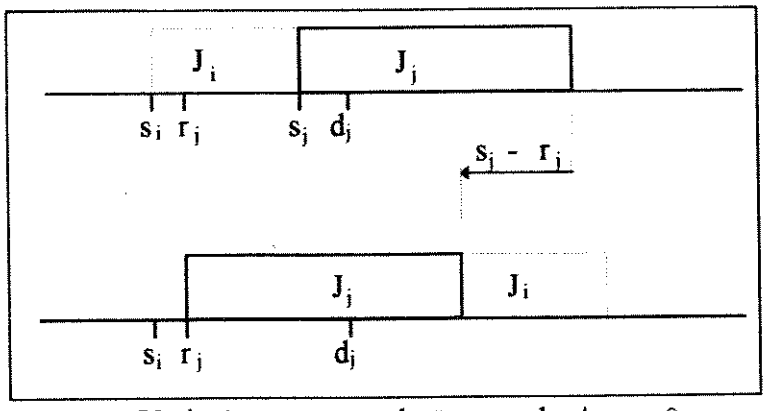

Figura 3.29 - Variação no atraso de J_j quando $\Delta_j < 0$ e $r_j > s_i$.

• $0 \leq \Delta_j < \min\{p_i, s_j - r_j\} \rightarrow$ é o caso em que J_j está atrasada antes da troca e o deslocamento realizado é maior que seu atraso. Portanto, parte desse deslocamento gera um custo de avanço. A variação no custo será de:

$$\Delta custo(J_j) = -t_j.(\min\{p_i, s_j - r_j\} - \Delta_j) + h_j.\Delta_j. \qquad (3.24)$$

Na Figura 3.30, mostram-se as variações de avanço e atraso de J_j, quando $r_j \leq s_i$.

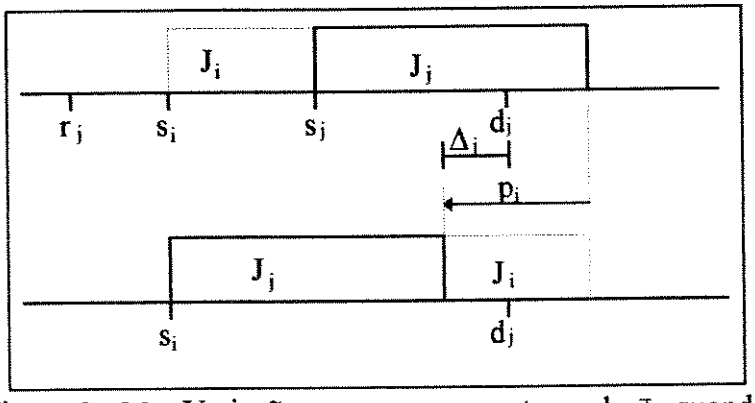

Figura 3.30 - Variação no avanço e no atraso de J_j quando $0 \leq \Delta_j < \min\{p_i, s_j - r_j\}$ e $r_j \leq s_i$.

• $\Delta_j \geq \min\{p_i, s_j - r_j\} \rightarrow J_j$ está adiantada antes e depois da troca. O deslocamento realizado acarretará apenas um aumento no custo de avanço. A variação de custo é, nesse caso:

$$\Delta custo(J_j) = h_j.\min\{p_i, s_j - r_j\}. \qquad (3.25)$$

A Figura 3.31 ilustra esse caso, quando $r_j \leq s_i$.

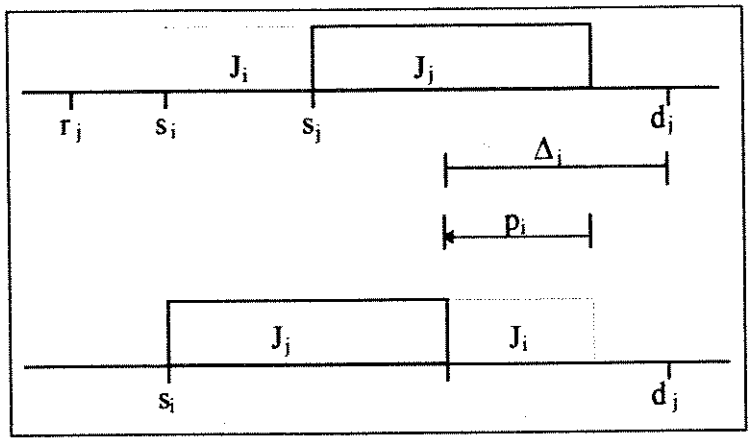

Figura 3.31 - Variação no avanço de J_j quando $\Delta_j \geq \min\{p_i, s_j - r_j\}$ e $r_j \leq s_i$.

Analogamente ao caso de J_i, faz-se:

$$\Omega_{ji} = \begin{cases} 0, & \text{se } \Delta_j < 0; \\ \Delta_j, & \text{se } 0 \leq \Delta_j < \min\{p_i, s_j - r_j\} \\ \min\{p_i, s_j - r_j\}, & \text{caso contrário} \end{cases} \qquad (3.26)$$

De (3.23) a (3.26):

$$\Delta\text{custo}(J_j) = -t_j . \min\{p_i, s_j - r_j\} + \Omega_{ji} . (h_j + t_j). \qquad (3.27)$$

A variação no custo do programa, no caso de se realizar a troca de posições entre J_i e J_j, pode ser calculada como:

$$\begin{aligned} \text{custo}(ij) - \text{custo}(ji) = \Delta\text{custo}(J_i) + \Delta\text{custo}(J_j) = \\ t_i . (p_j + \max\{0, r_j - s_i\}) - \Omega_{ij} . (h_i + t_i) \\ -t_j . \min\{p_i, s_j - r_j\} + \Omega_{ji} . (h_j + t_j) \end{aligned} \qquad (3.28)$$

Quando $\text{custo}(ij) - \text{custo}(ji) \geq 0$, isso significa que a troca não deve ser realizada. Caso contrário, a troca deve ser feita.

Tendo-se uma expressão para a variação no custo incorrida com a troca, pode-se enunciar o Teorema de Adjacência Ótima Generalizado. A verificação da validade do teorema pode ser dividida em quatro casos, dependendo dos valores de a_i e a_j. Esses índices estão definidos na página 18 e a verificação de cada caso é apresentada no Apêndice A.

TEOREMA DE ADJACÊNCIA ÓTIMA GENERALIZADO:

"A tarefa J_i precede imediatamente a tarefa J_j se a seguinte condição for verdadeira:

$$t_i.(p_j + \max\{0, r_j - s_i\}) - \Omega_{ij}.(h_i + t_i) - t_j.\min\{p_i, s_j - r_j\} + \Omega_{ji}.(h_j + t_j) \geq 0 \quad (3.29)$$

onde:

$$\Omega_{ij} = \begin{cases} 0, & \text{se } \Delta_i < 0; \\ \Delta_i, & \text{se } 0 \leq \Delta_i < p_j + \max\{0, r_j - s_i\} \\ p_j + \max\{0, r_j - s_i\}, & \text{caso contrário} \end{cases};$$

$$\Omega_{ji} = \begin{cases} 0, & \text{se } \Delta_j < 0; \\ \Delta_j, & \text{se } 0 \leq \Delta_j < \min\{p_i, s_j - r_j\} \\ \min\{p_i, s_j - r_j\}, & \text{caso contrário} \end{cases};$$

$\Delta_i = d_i - p_i - r_i$ e

$\Delta_j = d_j - p_j - \max\{r_j, s_i\}$."

Na seção seguinte, apresenta-se a forma como esse teorema será usado para a exploração da vizinhança do programa S, obtido ao final da fase construtiva da heurística.

3.3 - FASE DE BUSCA EM VIZINHANÇA

A partir do resultado teórico apresentado na seção anterior, desenvolveu-se um procedimento iterativo de busca na vizinhança TTA da solução **S**. Esse procedimento é iniciado com o primeiro par de tarefas adjacentes do programa inicial e, ao chegar ao último par, o procedimento pára. O programa obtido no final será um ótimo local para o problema em estudo, pois não será mais possível melhorar o valor do custo pela troca de posição entre duas tarefas adjacentes.

O passo inicial do Procedimento de Busca Local é, então, identificar o primeiro par de tarefas adjacentes no programa **S**. Marcam-se as duas tarefas do par como $J_{[i]}$ e $J_{[j]}$. O passo seguinte consiste em verificar se a troca entre essas tarefas não irá alterar o processamento das tarefas que as seguem, em **S**.

Estuda-se, primeiramente, o caso em que $r_{[j]} \leq s_{[i]}$, como as tarefas da Figura 3.32.

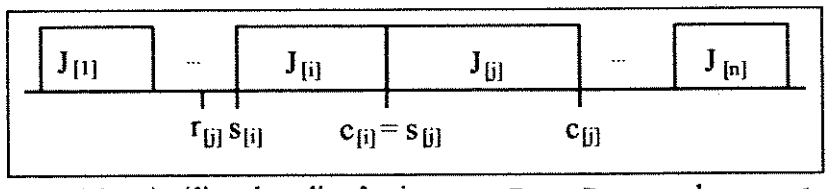

Figura 3.32 - Análise da adjacência entre $J_{[i]}$ e $J_{[j]}$ quando $r_{[j]} \leq s_{[i]}$.

Quando isso ocorre, basta conferir o Teorema de Adjacência Ótima Generalizado para essas tarefas e realizar a troca, caso a condição (3.29) não seja satisfeita.

Se $r_{[j]} > s_{[i]}$, podem ocorrer as situações mostradas na Figura 3.33.

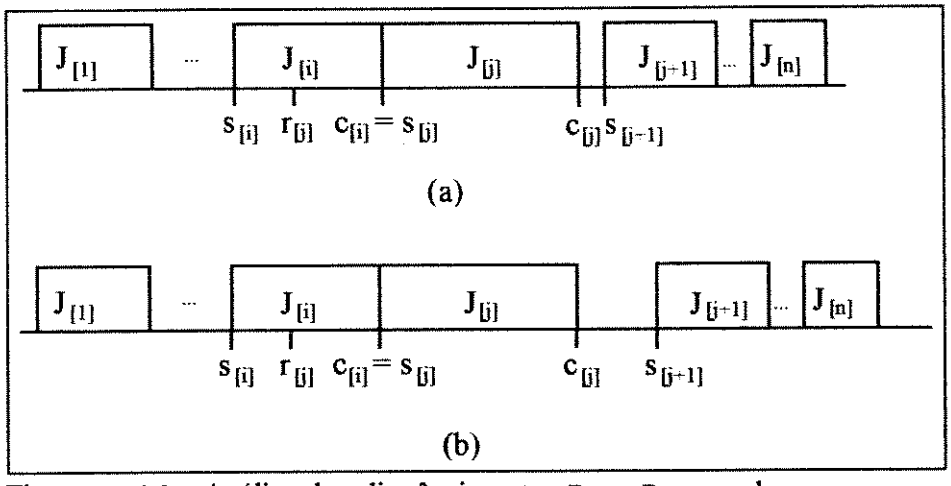

Figura 3.33 - Análise da adjacência entre $J_{[i]}$ e $J_{[j]}$ quando $r_{[j]} > s_{[i]}$.

No caso representado na Figura 3.33b, a aplicação do Teorema de Adjacência é imediata como quando se tinha $r_{[j]} \leq s_{[i]}$.

Entretanto, no caso da Figura 3.33a, a troca entre as tarefas $J_{[i]}$ e $J_{[j]}$, interferirá no processamento de $J_{[j+1]}$, pois $s_{[j+1]} - c_{[j]} < r_{[j]} - s_{[i]}$. A atitude tomada nessa situação consiste em simplesmente verificar se a expressão (3.29) do Teorema de Adjacência é válida para $J_{[i]}$ e $J_{[j]}$. Caso a troca deva ser feita, é preciso analisar seu efeito nos custos das tarefas que sucedem $J_{[j]}$. Essa análise é feita como uma busca em vizinhança convencional, ou seja, constrói-se a solução alternativa e calcula-se seu custo para comparação com o programa atual.

É preciso garantir que, ao chegar ao último par de tarefas adjacentes, o procedimento tenha chegado a um ótimo local. Para isso, a cada par analisado, se houver uma troca, volta-se à análise do par anterior.

A Figura 3.34a mostra um pequeno exemplo com 4 tarefas onde $J_{[1]} = J_1$, $J_{[2]} = J_2$, $J_{[3]} = J_3$ e $J_{[4]} = J_4$, onde já foi analisada a adjacência entre $J_{[1]}$ e $J_{[2]}$. Passando-se à análise do segundo par de tarefas, $J_{[2]}$ e $J_{[3]}$, supõe-se que a troca entre essas

duas tarefas deva ser feita. O programa resultante após essa troca é mostrado na Figura 3.34b. Se o Procedimento de Busca Local passasse diretamente à análise de adjacência do terceiro par de tarefas, ao término do procedimento o programa obtido seria considerado um ótimo local. Mas, essa conclusão seria errônea, visto que o par de tarefas formado por J_1 e J_3 não teria sido analisado. Por isso, quando há uma troca de posição entre duas tarefas, é preciso voltar ao par de tarefas anterior, garantindo assim que, ao final do Procedimento de Busca Local, não exista nenhum par de tarefas que não tenha sido analisado.

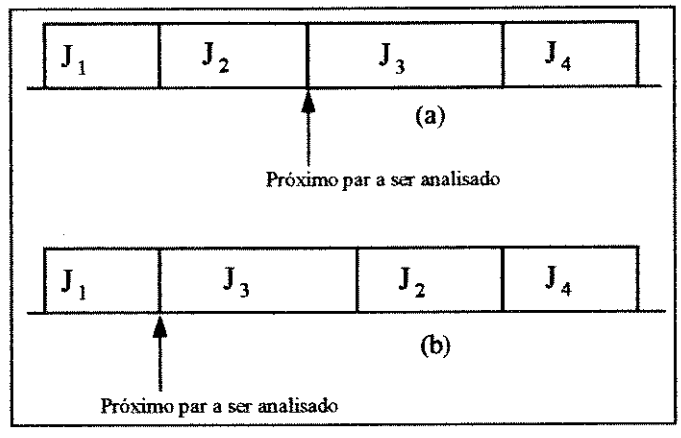

Figura 3.34 - Exemplo de retorno à análise do par anterior, quando ocorre uma troca de posição entre tarefas

O Procedimento de Busca Local pode ser resumido como:

Procedimento de Busca Local:

Passo 0: $i = 1, j = 2$.

Passo 1: Se as tarefas $J_{[i]}$ e $J_{[j]}$ forem adjacentes, vá para o Passo 2.
Caso contrário, vá para o Passo 7.

Passo 2: Se a condição (3.29) for verdadeira, vá para o Passo 7.
Caso contrário, vá para o Passo 3.

Passo 3: Se $r_{[j]} \leq s_{[i]}$, faça a troca das tarefas.

Passo 4: Se $r_{[j]} > s_{[i]}$ e $s_{[j+1]} - c_{[j]} \geq r_{[j]} - s_{[i]}$, faça a troca das tarefas.

Passo 5: Se $r_{[j]} > s_{[i]}$ e $s_{[j+1]} - c_{[j]} < r_{[j]} - s_{[i]}$, construa um novo programa factível

com $J_{[i]}$ e $J_{[j]}$ trocadas e avalie a variação de custo obtida.
Caso haja diminuição no custo, mantenha a troca.
Se o custo aumentar, desfaça a troca.

Passo 6: Se alguma troca foi feita nesta iteração e $i > 1$, faça $i = i - 2$.

Passo 7: Faça $i = i + 1$ e $j = i + 1$.
Se $i = n$, Pare.
Caso contrário, vá para o Passo 1.

APLICAÇÃO DO PROCEDIMENTO DE BUSCA LOCAL AO EXEMPLO

Passo 0: $i = 1, j = 2$;

Passo 1: $J_{[1]} = J_4$; $J_{[2]} = J_1$;
$c_4 = s_1 \Rightarrow J_4$ e J_1 são adjacentes.

Passo 2: $\Delta_4 = d_4 - p_4 - s_4 = 121 - 30 - 87 = 4$;
$p_1 + \max\{0, r_1 - s_4\} = 34 + \max\{0,69 - 87\} = 34$;
$0 < \Delta_4 < 34 \Rightarrow \Omega_{41} = 4$;
$\Delta_1 = d_1 - p_1 - \max\{r_1, s_4\} = 159 - 34 - \max\{69,87\} = 38$;
$\min\{p_4, s_1 - r_1\} = \min\{30,48\} = 30$;
$\Delta_1 > 30 \Rightarrow \Omega_{14} = 30$;
De 3.29:
$t_4.(p_1 + \max\{0, r_1 - s_4\}) - \Omega_{41}.(t_4 + h_4) -$
$-t_1.\min\{p_4, s_1 - r_1\} + \Omega_{14}.(t_1 + h_1) =$
$30.34 - 4.36 - 31.30 + 30.56 = 1626 > 0$;

Passo 7: $i = 2 < 8, j = 3$;

Passo 1: $J_{[2]} = J_1$; $J_{[3]} = J_5$;
$c_1 = s_5 \Rightarrow J_1$ e J_5 são adjacentes.

Passo 2: $\Delta_1 = d_1 - p_1 - s_1 = 159 - 34 - 117 = 8$;
$p_5 + \max\{0, r_5 - s_1\} = 24 + \max\{0,62 - 117\} = 24$;
$0 < \Delta_1 < 24 \Rightarrow \Omega_{15} = 8$;
$\Delta_5 = d_5 - p_5 - \max\{r_5, s_1\} = 221 - 24 - \max\{62,117\} = 80$;

$\min\{p_1, s_6 - r_5\} = \min\{34,89\} = 34;$

$\Delta_5 > 34 \Rightarrow \Omega_{51} = 34;$

De 3.29:

$t_1.(p_6 + \max\{0, r_5 - s_1\}) - \Omega_{16}.(t_1 + h_1) -$

$-t_5.\min\{p_1, s_5 - r_5\} + \Omega_{51}.(t_5 + h_5) =$

$31.24 - 8.56 - 97.34 + 34.158 = 2370 > 0;$

Passo 7: $i = 3 < 8, j = 4;$

Passo 1: $J_{[3]} = J_5; J_{[4]} = J_2;$

$c_5 = s_2 \Rightarrow J_5$ e J_2 são adjacentes.

Passo 2: $\Delta_5 = d_5 - p_5 - s_5 = 221 - 24 - 151 = 46;$

$p_2 + \max\{0, r_2 - s_5\} = 97 + \max\{0,166 - 151\} = 112;$

$0 < \Delta_5 < 112 \Rightarrow \Omega_{52} = 46;$

$\Delta_2 = d_2 - p_2 - \max\{r_2, s_5\} =$

$= 205 - 97 - \max\{166,151\} = -58;$

$\Delta_2 < 0 \Rightarrow \Omega_{25} = 0;$

$\min\{p_5, s_2 - r_2\} = \min\{24,9\} = 9;$

De 3.29:

$t_5.(p_2 + \max\{0, r_2 - s_5\}) - \Omega_{52}.(t_5 + h_5) -$

$-t_2.\min\{p_5, s_2 - r_2\} + \Omega_{25}.(t_2 + h_2) =$

$97.112 - 46.158 - 84.9 + 0.102 = 2840 > 0;$

Passo 7: $i = 4 < 8, j = 5;$

Passo 1: $J_{[4]} = J_2; J_{[5]} = J_8;$

$c_2 = s_8 \Rightarrow J_2$ e J_8 são adjacentes.

Passo 2: $\Delta_2 = d_2 - p_2 - s_2 = 205 - 97 - 175 = -67;$

$\Delta_2 < 0 \Rightarrow \Omega_{28} = 0;$

$\Delta_8 = d_8 - p_8 - \max\{r_8, s_2\} =$

$= 136 - 40 - \max\{255,175\} = -159;$

$\Delta_8 < 0 \Rightarrow \Omega_{82} = 0;$

$p_8 + \max\{0, r_8 - s_2\} = 40 + \max\{0,255 - 175\} = 120;$

$\min\{p_2, s_8 - r_8\} = \min\{97,17\} = 17;$

De 3.29:

$t_2.(p_8 + \max\{0, r_8 - s_2\}) - \Omega_{28}.(t_2 + h_2) -$

$-t_8.\min\{p_2, s_8 - r_8\} + \Omega_{82}.(t_8 + h_8) =$

$84.120 - 0.102 - 69.17 + 0.142 = 8907 > 0;$

Passo 7: $i = 5 < 8, j = 6;$

Passo 1: $J_{[5]} = J_8; J_{[6]} = J_7;$

$c_8 = s_7 \Rightarrow J_8$ e J_7 são adjacentes.

Passo 2: $\Delta_8 = d_8 - p_8 - s_8 = 136 - 40 - 272 = -176$;

$\Delta_8 < 0 \Rightarrow \Omega_{87} = 0$;

$\Delta_7 = d_7 - p_7 - \max\{r_7, s_8\} =$

$= 181 - 91 - \max\{310,272\} = -220$;

$\Delta_7 < 0 \Rightarrow \Omega_{78} = 0$;

$p_7 + \max\{0, r_7 - s_8\} = 91 + \max\{0,310 - 272\} = 129$;

$\min\{p_8, s_7 - r_7\} = \min\{40,2\} = 2$;

De 3.29:

$t_8.(p_7 + \max\{0, r_7 - s_8\}) - \Omega_{87}.(t_8 + h_8) -$

$-t_7.\min\{p_8, s_7 - r_7\} + \Omega_{78}.(t_7 + h_7) =$

$69.129 - 0.142 - 98.2 + 0.182 = 8705 > 0$;

Passo 7: $i = 6 < 8, j = 7$;

Passo 1: $J_{[6]} = J_7$; $J_{[7]} = J_6$;

$c_7 = s_6 \Rightarrow J_7$ e J_6 são adjacentes.

Passo 2: $\Delta_7 = d_7 - p_7 - s_7 = 181 - 91 - 312 = -222$;

$\Delta_7 < 0 \Rightarrow \Omega_{76} = 0$;

$\Delta_6 = d_6 - p_6 - \max\{r_6, s_7\} =$

$= 237 - 47 - \max\{270,312\} = -122$;

$\Delta_6 < 0 \Rightarrow \Omega_{67} = 0$;

$p_6 + \max\{0, r_6 - s_7\} = 47 + \max\{0,270 - 312\} = 47$;

$\min\{p_7, s_6 - r_6\} = \min\{91,133\} = 91$;

De 3.29:

$t_7.(p_6 + \max\{0, r_6 - s_7\}) - \Omega_{76}.(t_7 + h_7) -$

$-t_6.\min\{p_7, s_6 - r_6\} + \Omega_{67}.(t_6 + h_6) =$

$98.47 - 0.182 - 46.91 + 0.132 = 420 > 0$;

Passo 7: $i = 7 < 8, j = 8$;

Passo 1: $J_{[7]} = J_6$; $J_{[8]} = J_3$;

$c_6 = s_3 \Rightarrow J_6$ e J_3 são adjacentes.

Passo 2: $\Delta_6 = d_6 - p_6 - s_6 = 237 - 47 - 403 = -213$;

$\Delta_6 < 0 \Rightarrow \Omega_{63} = 0$;

$\Delta_3 = d_3 - p_3 - \max\{r_3, s_6\} =$

$= 264 - 22 - \max\{14,403\} = -161$;

$\Delta_3 < 0 \Rightarrow \Omega_{36} = 0$;

$p_3 + \max\{0, r_3 - s_6\} = 22 + \max\{0,14 - 403\} = 22$;

$\min\{p_6, s_3 - r_3\} = \min\{47,436\} = 47$;

De 3.29:

$$t_6.(p_3 + \max\{0, r_3 - s_6\}) - \Omega_{63}.(t_6 + h_6) -$$
$$-t_3.\min\{p_6, s_3 - r_3\} + \Omega_{36}.(t_3 + h_3) =$$
$$46.22 - 0.132 - 1.47 + 0.35 = 965 > 0;$$

Passo 7: i = 8, Pare.

A aplicação do Procedimento de Busca em Vizinhança ao resultado da fase construtiva mostra que não foi possível melhorar o custo de **S** através da troca de posição entre duas tarefas adjacentes. Portanto, a saída de heurística para o exemplo é aquela mostrada na Tabela 3.3, no final da seção 3.2.

Esse mesmo exemplo foi resolvido através do algoritmo ótimo descrito no Capítulo 4. O valor de custo obtido foi 52387. A solução heurística fica a 0,2% da solução ótima.

3.4 - DIAGRAMA EM BLOCOS

No início da seção 3.1 foi apresentado um diagrama resumido indicando qual a relação entre os procedimentos envolvidos na fase construtiva da heurística. Tendo sido detalhados esses procedimentos, assim como a estratégia de Busca em Vizinhança, é possível apresentar um diagrama mais completo da heurística como um todo. A análise de complexidade da heurística proposta é mostrada no Apêndice B

Com esse diagrama, encerra-se o Capítulo 3. No capítulo seguinte, apresentam-se os resultados computacionais obtidos pela aplicação da heurística ao problema E/T. Foram comparados os valores de custo e as seqüências obtidas para problemas com n = 8, com os custos e as sequências ótimos. Além disso, para problemas com n = 20, 40, 60 e 80 tarefas, avaliam-se os resultados obtidos em relação à vizinhança construída pela estratégia de Troca de Todos os Pares de Tarefas.

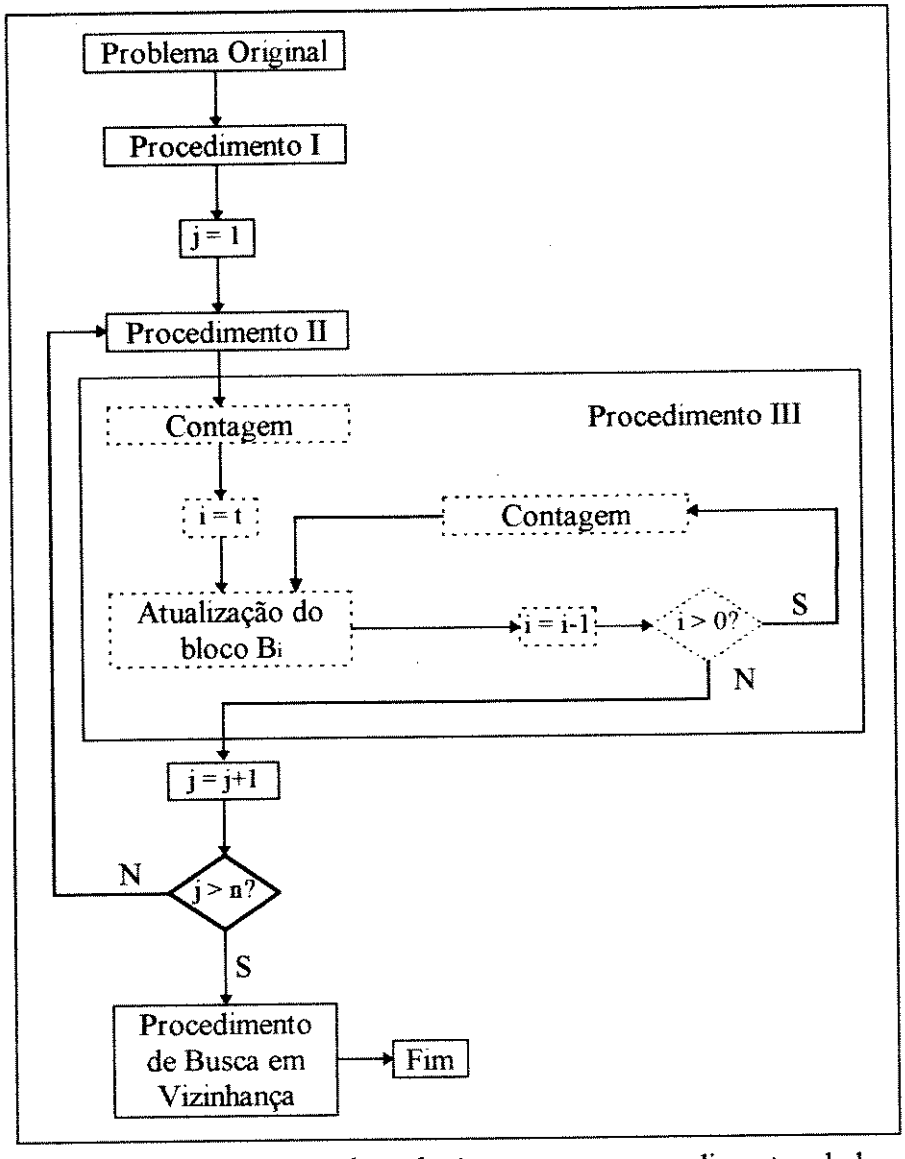

Diagrama 3.2 - Detalhamento das relações entre os procedimentos da heurística.

CAPÍTULO 4
RESULTADOS COMPUTACIONAIS

Neste capítulo são apresentados os resultados de toda experiência computacional realizada com a finalidade de avaliar a heurística proposta.

Na seção 4.1 apresenta-se a estratégia de geração de problemas de teste. Às instâncias geradas segundo essa estratégia, foi aplicado o procedimento heurístico descrito no Capítulo 3.

As tentativas de obtenção de resultados ótimos são descritas na seção 4.2. Infelizmente, os algoritmos exatos são capazes de resolver apenas problemas com poucas tarefas, dentro de um tempo considerado razoável. O experimento descrito nessa seção corresponde a um conjunto de instâncias com 8 tarefas. Esse conjunto adicional de instâncias de teste foi gerado de forma idêntica aos que se pretendia testar inicialmente com n = 20, 40, 60 e 80. No final da seção 4.2, são mostrados os resultados da comparação entre os valores ótimos e os heurísticos, bem como uma análise das seqüências obtidas.

Visto que, para as instâncias geradas, não foi possível encontrar os valores ótimos de custo, foi feita uma comparação dos valores heurísticos com ótimos locais. Esses valores foram encontrados através da aplicação de uma Busca Local à solução heurística, utilizando-se uma estratégia de Troca de Todos os Pares de Tarefas para a construção das soluções vizinhas. Essa comparação é apresentada na seção 4.3.

Todos os programas computacionais foram codificados na linguagem C e executados em estações de trabalho SUN Sparc Classic e Sparcstation IPX.

Na seção 4.4 apresentam-se as conclusões desse trabalho.

4.1 - GERAÇÃO DAS INSTÂNCIAS DE TESTE

Para se gerar uma instância do Problema E/T é preciso determinar os valores numéricos dos parâmetros constantes das tarefas. São eles:

- p_i - tempo de processamento;
- r_i - instante de liberação para processamento;
- d_i - data de entrega;
- h_i - custo (penalidade) por unidade de tempo de avanço; e
- t_i - custo (penalidade) por unidade de tempo de atraso.

Esses parâmetros foram gerados aleatoriamente de acordo com as regras descritas nesta seção.

A primeira constante a ser gerada foi p_i, $\forall i \in \mathbf{N}$. Usou-se uma distribuição uniforme no intervalo $[1, 100]$. Em uma distribuição uniforme, a variância é proporcional ao quadrado da largura do intervalo. Portanto foi escolhido um limitante superior que permitisse uma dispersão razoável dos valores dos tempos de processamento dentro do intervalo.

Para cada instância, quando os valores de p_i, $\forall i \in \mathbf{N}$ foram gerados, calculou-se sua soma.

Seja, então, $P = \sum_{i \in \mathbf{N}} p_i$.

No contexto prático, pode-se pensar no intervalo $[0, P]$ como um horizonte de planejamento. Os valores de r_i e d_i, $\forall i \in \mathbf{N}$ foram gerados dentro de intervalos criados com base no valor de P, com o objetivo de obter valores com algum significado prático.

Os valores de r_i, $\forall i \in \mathbf{N}$ foram gerados a partir de uma distribuição uniforme no intervalo $[0, P]$. Esse intervalo foi escolhido por se entender que não faz sentido considerar tarefas que estejam liberadas para processamento após o final do horizonte de planejamento.

A geração dos valores das datas de entrega seguiu o método sugerido por Potts e Van Wassenhove (1982) e utilizado em YANO e KIM (1991) e KIM e YANO (1994) onda cada d_i, $i \in \mathbf{N}$ é gerada de acordo com uma distribuição uniforme sobre o intervalo:

$$I = [P.(1 - FA - FD / 2), P.(1 - FA + FD / 2)],$$

onde:

$FA \rightarrow$ fator de atraso. Esse fator é sugerido na literatura e representa a porcentagem de tarefas atrasadas numa seqüência ótima, quando $r_i = 0$, $\forall i \in \mathbf{N}$.

Quanto maior o valor de FA, maior o número de tarefas atrasadas na seqüência ótima. Entretanto, para o problema E/T, na forma estudada neste trabalho, tem-se $r_i \geq 0, \forall i \in N$. Portanto, esse fator perde um pouco de seu sentido, mas é utilizado para variar a média do intervalo de geração das datas de entrega.

FD → é definido como a faixa de datas de entrega. É usado para variar a dispersão dos valores gerados em torno da média P.(1 - FA). Para um valor fixo de FA, quanto maior o valor de FD, maior é a dispersão dos valores gerados.

Para cada um desses fatores, foram atribuídos três valores. São eles:

FA = 0,2;0,5 e 0,8;

FD = 0,4;0,7 e 1,0.

Pode-se, então, definir uma classificação inicial das instâncias de teste. Essa classificação divide o conjunto de instâncias em nove casos, listados na tabela abaixo:

Caso	FA	FD	P.(1 - FA)	I
1	0,2	0,4	0,8P	[0,6P, P]
2	0,2	0,7	0,8P	[0,45P, 1,15P]
3	0,2	1,0	0,8P	[0,3P, 1,3P]
4	0,5	0,4	0,5P	[0,3P, 0,7P]
5	0,5	0,7	0,5P	[0,15P, 0,85P]
6	0,5	1,0	0,5P	[0, P]
7	0,8	0,4	0,2P	[0, 0,4P]
8	0,8	0,7	0,2P	[0, 0,55P]
9	0,8	1,0	0,2P	[0P, 0,7P]

Tabela 4.1 - Classificação inicial de tipos de instâncias de teste.

Pelo mesmo motivo explicado no caso dos tempos de processamento, as penalidades de avanço e atraso foram geradas de acordo com uma distribuição uniforme no intervalo [0,100] pois, quaisquer que sejam os custos reais, sempre pode-se normalizá-los para que se situem numa faixa entre 0 a 100%. Para avaliar a influência da relação entre as penalidades h_i e t_i, para cada caso mostrado na Tabela 4.1, foram geradas instâncias com:

- $h_i = t_i / 2$
- $h_i = t_i$
- $t_i = h_i / 2$
- h_i e t_i gerados independentemente.

Dessa forma, expande-se a classificação dada anteriormente. Como, para cada caso na Tabela 4.1, há quatro combinações dos valores de h_i e t_i, chega-se a 36 tipos diferentes.

Para testar o comportamento da heurística para cada um dos tipos de instância definidos pelos parâmetros constantes da tarefas, foram geradas 5 amostras. Ou seja, para cada um dos 36 tipos foram geradas cinco instâncias diferentes, por se entender que podem ser tiradas conclusões errôneas baseando-se em um único experimento para cada tipo.

Foram geradas, então, 180 instâncias para cada valor de n (n=20, 40, 60, 80), num total de 720 instâncias. A elas foi aplicado o procedimento heurístico proposto e obtiveram-se os valores heurísticos de custo e o programa de produção composto da seqüência de processamento e dos instantes de início de cada tarefa. Os tempos computacionais obtidos foram extremamente satisfatórios, visto que a heurística não demorou mais que 5 segundos para resolver cada instância com o maior número de tarefas. Para avaliar a qualidade da solução heurística procurou-se, então, encontrar resultados ótimos para comparação. O próximo passo, portanto, foi procurar algum pacote computacional, ou algoritmo exato que pudesse resolver essas instâncias otimamente ou, pelo menos, fornecer um bom valor de limitante inferior para o valor do custo de cada instância. Na seção seguinte, descreve-se essa etapa do trabalho.

4.2 - RESULTADOS ÓTIMOS

A primeira tentativa no sentido de se obter os valores ótimos de custo para as instâncias de teste foi feita usando-se o pacote de subrotinas de otimização OSL ("Optimization Subroutines Library") da IBM. Esse pacote contém rotinas de resolução de problemas de Programação Linear Inteira Mista através de "Branch & Bound". A cada nó da árvore de busca um limitante inferior é obtido pela relaxação linear do subproblema correspondente.

Foram gastos alguns meses de trabalho para que se pudesse construir um programa de chamada às subrotinas do OSL e testar seu funcionamento. O modelo (1.5), apresentado no Capítulo 1, foi utilizado como entrada para o programa, na forma exigida pelo pacote para entrada de dados. Infelizmente, os resultados computacionais foram decepcionantes pois, devido ao grande número de variáveis binárias presentes no modelo, o programa criado não conseguiu resolver nem mesmo as instâncias com 20 tarefas. Mesmo os limitantes inferiores encontrados não apresentaram valores significativamente fortes para serem considerados de utilidade na comparação desejada.

Como exemplo, para uma instância com $n = 20$, o número de variáveis binárias é 420 e, como se deseja determinar o valor de s_i, T_i e E_i, $\forall i \in \mathbf{N}$, existem mais 60 variáveis a serem determinadas pelo programa. Essas últimas 60 variáveis foram declaradas como contínuas pois, como todos os parâmetros das tarefas são inteiros, existe uma solução ótima inteira. Para uma instância com esse número de variáveis foi feita uma experiência para determinar qual o número de nós necessário para que o OSL produzisse um bom limitante inferior. Infelizmente, após 30 horas, aproximadamente, não havia sido obtido um limitante inferior de boa qualidade. Levando em conta a possibilidade de que a instância utilizada pudesse ser um caso patológico, realizaram-se outros testes com instâncias de tipos diferentes. Novamente, para diversas dessas instâncias, não foram obtidos resultados satisfatórios em 24 horas de execução do programa.

Uma explicação para a dificuldade encontrada pelo pacote da IBM em trabalhar com o Problema E/T pode ser exatamente o modelo matemático utilizado. Para esse problema, a resolução do modelo parece não ser a maneira mais adequada de se obter o programa ótimo, devido ao grande número de soluções factíveis e pelo fato do limitante inferior fornecido pela relaxação linear ser muito pobre.

Existem, entretanto, maneiras de melhorar o desempenho do programa de chamada ao OSL através de rotinas especializadas em que se pode aperfeiçoar a estratégia de busca. Essas rotinas são chamadas de "USER EXITS". Entretanto, a documentação do pacote não fornece informações precisas sobre os mecanismos internos dessas rotinas, nem mesmo sobre como construí-las de forma a aproveitar melhor as potencialidades do OSL. O tempo necessário para que essas rotinas fossem programadas e testadas poderia ser impraticável e, portanto, decidiu-se abandonar essa linha de trabalho e usar outro algoritmo que aproveitasse melhor a estrutura do problema para obter os valores ótimos de custo.

O trabalho de DAVIS e KANET(1993), descrito sucintamente no Capítulo 2, contém um algoritmo "Branch & Bound" desenvolvido especialmente para o problema E/T. Cada nível da árvore de busca corresponde a seqüências parciais com um determinado número de tarefas. No primeiro nível, são geradas, a partir do nó raiz, seqüências parciais com uma tarefa. No segundo nível, são geradas as seqüências parciais de duas tarefas, e assim por diante, até o último nível, que corresponde a todos os programas de produção com n tarefas. O limitante superior deve ser o custo de alguma seqüência factível conhecida. Um novo limitante superior somente é encontrado quando se atinge o último nível da árvore de busca. A cada nó gerado, o limitante inferior é calculado como o custo do programa semiativo correspondente à seqüência parcial desse nó. O cálculo do limitante inferior é feito pelo procedimento TIMETABLER, que insere otimamente os intervalos ociosos necessários para a obtenção de um programa semiativo, dada uma seqüência de

processamento. Se esse limitante inferior for maior que o limitante superior fornecido, o nó é eliminado da árvore.

A implementação desse algoritmo não teve como finalidade encontrar os valores ótimos para todas as instâncias de teste, devido ao grande número de nós que deveriam ser explorados para alcançar esse objetivo. Portanto, foi escolhida uma estratégia de busca que favorecesse o crescimento rápido do limitante inferior do problema, que é o menor limitante inferior de todos os nós ativos da árvore de "Branch & Bound" no instante de parada do programa. Essa estratégia consiste em escolher para ramificação sempre o nó ativo com o menor limitante inferior.

A seguir é apresentado o procedimento TIMETABLER, apropriadamente modificado para considerar instantes de liberação para processamento não nulos. A seguinte notação é adotada na descrição do procedimento:

t	número do estágio;
S(t,P)	um programa parcial com (t - 1) tarefas, onde P é uma permutação dessas tarefas. Trata-se P como um conjunto ordenado e seja iP a permutação na qual P é imediatamente precedida pela tarefa J_i;
P'	um conjunto ordenado de tarefas que ainda não foram programadas. As tarefas que seguem aquelas em P' estão em P, isto é, P' P é a permutação completa de tarefas em qualquer estágio.

O procedimento TIMETABLER (modificado) pode ser resumido da seguinte forma:

Procedimento TIMETABLER (modificado):

Passo 1: Faça t = 1 com S(t,P) nulo e P = ∅;

Passo 2: Selecione a última tarefa J_i de P'
e elimine-a de P';

Passo 3: Passe para o próximo estágio, fazendo:

3.1: Adicione J_i a S(t, P) para criar S(t + 1, P) da seguinte forma:

3.1.1: Inicialmente, a tarefa J_i é iniciada no instante $\max\{0, r_i, \sum_{j \in P'} p_j\}$, movendo as tarefas de S(t, P) para a direita, se necessário;

3.1.2: Desloque a tarefa J_i para a direita até que o custo marginal desse deslocamento deixe de ser negativo;

3.2: Faça t = t + 1;

3.3: Faça P = iP;

Passo 4: Se P' ≠ ∅, vá para o Passo 2.
Caso contrário, Pare.

No Apêndice C é apresentado um exemplo do funcionamento do Procedimento TIMETABLER (modificado). Para uma demonstração de sua otimalidade, ver DAVIS e KANET (1993).

Para determinar o número máximo de nós a serem explorados, a fim de se obterem bons limitantes inferiores, foram escolhidos três casos diferentes, com intervalos de geração de datas de entrega distintos, e com h_i e t_i gerados independentemente. Foi usada uma instância de cada caso para n = 20, 40 e 60. Os valores de teste para o número máximo de nós explorados foram fixados em 50.000, 100.000 e 150.000. Os resultados desse teste podem ser resumidos nas tabelas 4.2 a 4.4.

Nessas tabelas, a segunda coluna apresenta o valor heurístico usado como limitante superior. A terceira coluna apresenta o valor do limitante inferior do nó raiz e a quarta coluna mostra o menor limitante inferior da árvore, após ser explorado o número de nós determinado. A penúltima coluna das tabelas 4.2 a 4.4 mostra a porcentagem de nós criados e ainda ativos quando se atingiu o máximo número de nós permitido. A última coluna de cada tabela mostra o tempo necessário para explorar esses nós.

		Menor LI		Nós		
Caso	LS	Inicial	Final	Explorados	Ativos(%)	Tempo(s)
3	511156	0	13395	50.000	0,81	43,22
			15578	100.000	0,80	107,15
			16958	150.000	0,79	189,07
6	560984	394	26605	50.000	0,90	40,97
			31867	100.000	0,89	116,43
			34628	150.000	0,89	197,55
9	561302	1533	36557	50.000	0,85	38,75
			41004	100.000	0,84	96,57
			43209	150.000	0,83	169,68

Tabela 4.2 - Resultados dos testes para determinação do número máximo de nós a serem explorados pelo algoritmo de DAVIS e KANET (1993), com n = 20.

		Menor LI		Nós		
Caso	LS	Inicial	Final	Explorados	Ativos(%)	Tempo(s)
3	604850	0	130	50.000	0,69	41,08
			130	100.000	0,67	83,45
			330	150.000	0,67	135,13
6	1558394	342	9396	50.000	0,87	36,33
			12024	100.000	0,87	81,22
			13800	150.000	0,87	137,90
9	2043900	1366	29930	50.000	0,90	44,23
			33330	100.000	0,90	94,33
			34994	150.000	0,90	189,27

Tabela 4.3 - Resultados dos testes para determinação do número máximo de nós a serem explorados pelo algoritmo de DAVIS e KANET (1993), com n = 40.

		Menor LI		Nós		
Caso	LS	Inicial	Final	Explorados	Ativos(%)	Tempo(s)
3	1692429	0	0	50.000	0,69	48,47
			0	100.000	0,68	96,43
			0	150.000	0,68	152,42
6	4446913	390	4229	50.000	0,90	59,75
			5484	100.000	0,90	120,37
			6738	150.000	0,90	194,95
9	5733858	4800	66646	50.000	0,90	50,53
			83585	100.000	0,90	103,22
			90210	150.000	0,90	169,50

Tabela 4.4 - Resultados dos testes para determinação do número máximo de nós a serem explorados pelo algoritmo de DAVIS e KANET (1993), com n = 60.

Pode-se observar, pelo apresentado nas tabelas 4.2 a 4.4 que, mesmo aproveitando a estrutura do problema, o limitante inferior final ainda está muito distante do limitante superior fornecido pela heurística. Esse distanciamento pode ser devido a uma de duas razões. Ou o limitante superior fornecido pela heurística é ruim, indicando um desempenho fraco da heurística para as instâncias testadas, ou o limitante inferior é de má qualidade. Mas, visto que a porcentagem de nós ativos restantes na árvore de busca é bastante alta em relação aos nós explorados (em torno de 90%), seria um tanto precipitado concluir alguma coisa em relação aos valores dos limitantes inferiores. Observa-se, também, que o valor do limitante inferior para cada instância testada não aumenta significativamente conforme é aumentado o número de nós explorados. Isso fez com que não fossem conduzidos testes adicionais com um número máximo de nós maior que 150.000 pois a melhora obtida no valor do limitante inferior não compensou o tempo adicional despendido e o gasto de memória observado.

Para que, mesmo assim, pudesse ser avaliada a qualidade da heurística em relação a

valores ótimos, foi gerado um conjunto adicional de instâncias de teste, com n = 8. As regras de geração de parâmetros utilizada foram as mesmas das instâncias anteriores. Portanto, esse conjunto adicional continha 180 instâncias, sendo 5 amostras para cada um dos 36 tipos definidos. Essas instâncias foram resolvidas pela heurística. O valor heurístico de custo foi usado como limitante superior para o algoritmo de DAVIS e KANET (1993) e, então, puderam ser obtidos os valores ótimos.

A comparação entre o procedimento heurístico e o algoritmo ótimo pode ser feita através da Tabela 4.5. Os valores apresentados referem-se à média calculada para as 5 amostras de cada tipo de instância. Acreditou-se ser interessante apresentar os resultados de cada fase da heurística para avaliar o efeito das trocas adjacentes no valor do custo. A segunda coluna da Tabela 4.5 apresenta o tempo médio gasto pela fase construtiva e a terceira coluna mostra o valor de tempo correspondente para a fase de busca em vizinhança. A seguir, é apresentado o número médio de trocas realizadas para as 5 amostras e a melhora obtida com essas trocas. A sexta e sétima colunas referem-se ao tempo gasto pelo algoritmo de "Branch & Bound" e ao erro médio calculado entre o custo heurístico e ótimo. A última coluna da Tabela 4.5 mostra o maior erro encontrado, dentre as 5 amostras, isto é, a maior diferença percentual entre o valor ótimo e o valor heurístico de custo dentre as 5 instâncias de cada tipo.

Sejam:

custo1	custo obtido ao final da fase construtiva da heurística;
custo2	custo obtido ao final da fase de busca em vizinhança: custo final da heurística;
custo3	custo ótimo.

As diferenças percentuais tabeladas foram calculadas da seguinte forma:

$\left(\frac{custo1}{custo2} - 1\right).100\%$	diferença percentual entre o custo final da heurística e o custo obtido pela fase construtiva;
$\left(\frac{custo2}{custo3} - 1\right).100\%$	diferença percentual entre o custo ótimo e o custo final da heurística.

O valor chamado de Pior Caso se refere ao maior valor encontrado para $\left(\frac{custo2}{custo3} - 1\right).100\%$ durante o cálculo da média das 5 amostras.

Para uma melhor visualização dos resultados, as entradas nulas na tabela foram substituídas por hífens.

Os resultados mostrados na Tabela 4.5 foram analisados da seguinte forma:

•Tempo computacional: a heurística possui um tempo computacional claramente menor que o algoritmo ótimo, como era esperado. Usando os resultados da terceira e da quarta colunas da tabela, calcula-se que a heurística demora, em média, 0,073 segundos para resolver uma instância com 8 tarefas. Esse tempo é bem menor que a média dos tempos da sétima coluna, igual a 18,446 segundos.

•Influência da Fase de Busca em Vizinhança: Em nenhum caso, a média da melhoria obtida nas cinco amostras ultrapassou os 10%. Isso mostra que a Fase Construtiva já produz uma solução razoável para o problema E/T, embora essa solução não seja um ótimo local em relação a sua vizinhança TTA. Observa-se, também, que o número de trocas não tem uma influência direta na melhora obtida.

•Comparação dos custos heurísticos com os custos ótimos: O desempenho da heurística, para problemas pequenos, é satisfatório para a maioria das instâncias testadas. Os resultados da nona coluna da Tabela 4.5 mostram claramente que a heurística proposta resolve muito bem instâncias com parâmetros que as identifique com o décimo primeiro tipo em diante, onde o erro percentual médio em relação aos custos ótimos fica abaixo dos 10%. Entretanto, para os primeiros dez tipos de instâncias, o erro médio é bastante alto, chegando a passar de 25%. Percebe-se, através desses resultados que os intervalos de geração das datas de entrega são fatores determinantes do desempenho da heurística. Da Tabela 4.1, observa-se que as tarefas dos primeiros 10 tipos de instância possuem datas de entrega distribuídas em intervalos definidos por FA = 0,2, ou seja, com média igual a 0,8P. Essa média é a maior dentre todas aquelas definidas por FA, gerando instâncias com datas de entrega pouco restritivas. Portanto, o comportamento da heurística fica favorecido quando as datas de entrega são mais restritivas. A relação h_i / t_i parece ter pouca influência no desempenho da heurística, visto que não é possível associar a ela nenhuma tendência de aumento ou diminuição dos valores de erro. A coluna de Pior Caso mostra ainda que, mesmo para os tipos de instâncias em que o erro médio é baixo, o pior erro encontrado pode ser bastante alto. Tomando como exemplo o décimo quarto tipo de instância, observa-se que o erro médio é praticamente igual ao erro de pior caso dividido por cinco, o que indica que em apenas uma das amostras ocorreu um erro muito alto, enquanto nas outras amostras desse tipo, o erro em relação ao ótimo ficou próximo de zero. Esse comportamento é observado para vários tipos a partir do décimo primeiro, mas não pode ser explicado pois ocorre em instâncias isoladas, não dependendo da estrutura de geração de parâmetros.

Instâncias			Fase construtiva	Fase de Busca em Vizinhança			Algoritmo Ótimo		
Tipo	Caso	h_2/t_2	Tempo(s)	Tempo(s)	Trocas	Melhora	Tempo(s)	Erro Médio	Pior caso
1	1	0.5	-	-	-	-	19,422	5,6	15,2
2		1,0	0,003	0,200	1,2	7,8	10,084	12,1	23,8
3		2,0	0,003	-	-	-	13,240	5,7	16,8
4		indep	-	-	0,6	4,7	20,266	13,4	61,1
5	2	0.5	-	0,200	-	-	11,934	8,5	30,0
6		1,0	-	0,400	1,4	9,9	10,842	17,2	57,8
7		2,0	-	-	1,8	6,2	19,356	26,2	62,4
8		indep	-	-	0,2	0,5	11,314	16,9	40,1
9	3	0.5	0,003	-	-	-	12,652	2,1	6,1
10		1,0	-	-	1,2	4,9	10,232	25,4	48,8
11		2,0	-	0,200	-	-	27,096	8,2	18,4
12		indep	-	-	0,4	2,4	8,214	7,2	19,3
13	4	0.5	0,003	-	-	-	14,100	0,2	0,5
14		1,0	-	-	0,2	1,7	10,042	7,2	16,4
15		2,0	-	-	-	-	14,802	4,0	9,9
16		indep	0,003	-	-	-	14,660	8,5	15,9
17	5	0.5	-	-	0,6	0,4	13,152	1,3	2,3
18		1,0	-	0,200	0,4	0,2	18,156	4,2	14,9
19		2,0	-	0,200	-	-	16,896	2,9	10,2
20		indep	0,003	0,400	0,4	0,5	16,962	1,5	4,1
21	6	0.5	-	-	0,4	0,1	24,086	3,4	9,9
22		1,0	0,003	-	0,6	3,1	13,514	7,1	22,2
23		2,0	-	0,200	0,4	0,9	21,292	4,3	11,6
24		indep	0,003	0,200	0,2	0,4	23,790	0,4	2,0
25	7	0.5	-	-	0,2	-	26,436	0,6	1,4
26		1,0	0,003	-	-	-	35,568	-	0,2
27		2,0	-	-	-	-	24,026	-	-
28		indep	-	-	-	-	17,176	-	-
29	8	0.5	-	0,200	-	-	21,840	0,1	0,6
30		1,0	-	-	-	-	24,632	0,3	1,5
31		2,0	-	-	0,2	0,1	35,074	0,4	2,1
32		indep	0,003	-	0,2	0,6	18,630	0,6	1,5
33	9	0.5	-	-	-	-	23,364	0,6	3,0
34		1,0	-	-	-	-	11,932	0,8	4,2
35		2,0	-	-	0,2	0,1	43,798	0,6	3,0
36		indep	-	0,200	-	-	31,912	1,8	8,8

Tabela 4.5 - Comparação entre os resultados heurísticos e os valores ótimos de custo para $n = 8$.

Outra comparação que pode ser feita entre o procedimento heurístico e o algoritmo ótimo diz respeito ao sequenciamento de tarefas. Na Tabela 4.6 apresentam-se as diferenças encontradas entre as seqüências finais ao término dos dois métodos. A última coluna dessa tabela é uma repetição da coluna de Erro Médio do Algoritmo Ótimo da Tabela 4.5. Esses resultados foram repetidos para que se pudesse analisar o erro médio de sequenciamento da heurística com relação à diferença percentual média do custo ótimo. As colunas correspondentes às Instâncias apresentam o número de tarefas que não se encontravam na posição ótima dada pelo algoritmo de "Branch & Bound". Salienta-se, entretanto, que cada instância pode ter mais de uma solução ótima. Nos casos em que a heurística obteve o valor de custo ótimo, a entrada correspondente, na tabela, está representada por um hífen. Em algumas das instâncias, a seqüência heurística corresponde à seqüência ótima, ou seja, não há nenhum erro de sequenciamento, mas o valor de custo é superior ao ótimo. Isso mostra que ocorreram erros na inserção de intervalos ociosos, pois o Procedimento III da Fase Construtiva só considera deslocamentos de tarefas à esquerda.

Observa-se que existe uma influência do número de tarefas fora de posição na solução heurística no erro médio calculado em relação ao custo ótimo, embora não se possa dizer que a relação entre esses dois fatores seja muito forte. Há casos em que ocorre um número alto de erros de sequenciamento e o erro é pequeno, e vice-versa. Apesar disso, a tendência de diminuição dos erros segue o que foi discutido com relação aos custos. Conforme as datas de entrega vão se tornando mais restritivas, a diminuição de erros de sequenciamento confirma o melhor desempenho da heurística nesses casos.

Para os tipos de instância com datas de entrega pouco restritivas (tipos 1 a 12) foi feita outra comparação entre o algoritmo ótimo e o procedimento heurístico, com respeito ao número de tarefas adiantadas nos programas de produção. Essa comparação teve como objetivo avaliar se, no Procedimento III (Atualização de Intervalos Ociosos), a não consideração de deslocamentos à direita prejudica o desempenho da heurística. Somente foram considerados os tipos de instância em que a média de erros de sequenciamento não foi maior que 2,0 pois, se a média ultrapassar esse valor, a comparação poderia perder seu sentido.

Observou-se que, nos programas heurísticos, há sempre uma ou duas tarefas adiantadas a mais. Em apenas três das instâncias, o programa ótimo possui uma tarefa adiantada a mais que o programa heurístico. Isso indica que a consideração de deslocamentos à direita no Procedimento III da fase construtiva da heurística pode melhorar seu desempenho ao resolver instâncias com datas de entrega pouco restritivas.

			Instâncias						
Tipo	Caso	t_i	1	2	3	4	5	Média	Erro Médio
1	1	0,5	2	5	0*	2	-	1,8	5,6
2		1,0	0*	5	0*	4	8	3,4	12,1
3		2,0	0*	4	0*	2	3	1,8	5,7
4		indep	2	-	2	4	-	1,6	13,4
5	2	0,5	5	8	3	5	3	4,8	8,5
6		1,0	0*	7	6	0*	8	4,2	17,2
7		2,0	-	6	4	6	5	4,2	26,2
8		indep	2	2	2	-	8	2,8	16,9
9	3	0,5	5	7	8	3	0*	4,6	2,1
10		1,0	5	2	7	5	2	4,2	25,4
11		2,0	6	-	8	0*	4	3,6	8,2
12		indep	0*	3	-	-	3	1,2	7,2
13	4	0,5	-	-	0*	0*	0*	-	0,2
14		1,0	5	0*	0*	2	3	2,0	7,2
15		2,0	-	0*	-	2	3	1,0	4,0
16		indep	0*	0*	2	3	0*	1,0	8,5
17	5	0,5	7	4	2	4	5	4,4	1,3
18		1,0	8	0*	-	0*	0*	1,6	4,2
19		2,0	7	3	-	3	0*	2,6	2,9
20		indep	5	7	0*	3	-	3,4	1,5
21	6	0,5	3	2	0*	3	2	2,0	3,4
22		1,0	0*	-	2	0*	0*	0,4	7,1
23		2,0	-	0*	2	2	-	0,8	4,3
24		indep	4	-	4	-	-	1,6	0,4
25	7	0,5	2	8	0*	-	5	2,0	0,6
26		1,0	-	-	-	-	4	0,8	-
27		2,0	-	-	-	-	-	-	-
28		indep	-	-	-	-	-	-	-
29	8	0,5	-	0*	-	-	-	-	0,1
30		1,0	2	-	-	3	-	1,0	0,3
31		2,0	-	-	-	2	-	-	0,4
32		indep	-	3	3	-	-	1,2	0,6
33	9	0,5	-	0*	-	-	-	-	0,6
34		1,0	0*	-	-	-	-	-	0,8
35		2,0	-	2	-	0*	-	0,4	0,6
36		indep	-	-	-	-	6	1,2	1,8

*A seqüência obtida pela heurística é ótima mas, o custo heurístico é maior que o custo ótimo.

Tabela 4.6 - Análise das seqüências de tarefas obtidas ao final de heurística e do algoritmo ótimo para n = 8.

Conclui-se portanto que, para as instâncias de teste com n = 8, a heurística apresentou um desempenho satisfatório. Na seção seguinte, será apresentada a análise dos conjuntos de instâncias com n = 20, 40, 60 e 80.

4.3 - ÓTIMOS LOCAIS

Devido à impossibilidade de se obter os valores ótimos de custo para as 720 instâncias geradas com 20, 40, 60 e 80 tarefas, os testes realizados com essas instâncias consistiram do seguinte. A cada instância, foi aplicado o procedimento heurístico, medindo-se o tempo necessário para gerar uma solução, o número de trocas efetuadas na Fase de Busca em Vizinhança e a diferença percentual do custo heurístico final e o custo após a Fase Construtiva.

Foram feitas duas avaliações das soluções heurísticas. A primeira teve por finalidade observar os erros de inserção de intervalos ociosos ocorridos durante o Procedimento III da Fase Construtiva. Para isso, foi aplicado, a cada seqüência final, o procedimento TIMETABLER (modificado), discutido na seção 4.2 obtendo-se, assim um novo valor de custo correspondente à seqüência final. A segunda avaliação correspondeu à realização de uma busca na vizinhança da solução heurística segundo a estratégia de Troca de Todos os Pares de Tarefas (TTP). Essa estratégia consiste em trocar de posição, entre si, cada par de tarefas da seqüência e avaliar o impacto dessa troca no custo da nova seqüência obtida. A cada troca, foi aplicado o procedimento TIMETABLER (modificado) para inserir otimamente os intervalos ociosos. A busca pára quando se encontra um ótimo local, ou seja, quando o custo marginal de qualquer troca de pares de tarefas se torna positivo ou nulo. Para cada instância de teste, ao realizar a busca em vizinhança, mediu-se o tempo necessário para alcançar um ótimo local, o número de vizinhanças exploradas e o erro percentual entre o valor de custo do ótimo local e o valor obtido após a aplicação de TIMETABLER (modificado) à seqüência final heurística. Além disso, foi monitorado o erro percentual entre o custo do ótimo local e o custo final heurístico para que se pudesse fazer um análise de pior caso. Nas tabelas 4.7 a 4.10, são apresentados os resultados obtidos para as instâncias com n = 20, 40, 60 e 80. Novamente, cada valor apresentado corresponde à média das 5 amostras geradas para cada tipo de instância.

Sejam:

custo1	custo obtido ao final da fase construtiva da heurística;
custo2	custo obtido ao final da fase de busca em vizinhança: custo final da heurística;
custo3	custo obtido pela aplicação do procedimento TIMETABLER à seqüência final heurística: custo com inserção ótima

custo4	custo obtido ao final da aplicação da busca TTP: custo do ótimo local.

As diferenças percentuais tabeladas foram calculadas da seguinte forma:

$\left(\frac{\text{custo1}}{\text{custo2}} - 1\right).100\%$	diferença percentual entre o custo final da heurística e o custo obtido pela fase construtiva;
$\left(\frac{\text{custo2}}{\text{custo3}} - 1\right).100\%$	diferença percentual entre o custo com inserção ótima e o custo final da heurística;
$\left(\frac{\text{custo3}}{\text{custo4}} - 1\right).100\%$	diferença percentual entre o custo com inserção ótima e o custo do ótimo local.
$\left(\frac{\text{custo2}}{\text{custo4}} - 1\right).100\%$	diferença percentual entre o custo final da heurística e o custo do ótimo local.

O valor chamado de Pior Caso se refere ao maior valor encontrado para $\left(\frac{\text{custo2}}{\text{custo4}} - 1\right).100\%$ durante o cálculo da média das 5 amostras.

A análise das tabelas 4.7 a 4.10 pode ser feita de forma análoga à da Tabela 4.5.

• Tempo Computacional: O tempo médio requerido pela heurística é bastante satisfatório, confirmando o que foi observado para o caso em que n = 8. Em nenhum dos casos estudados, esse tempo ultrapassou 2,0 segundos, com exceção do tipo 3 para n = 80. Em compensação, o tempo médio gasto pela busca TTP chegou a passar de 1000 segundos para o tipo 2 com n = 80. Para verificar se a velocidade da heurística não prejudica seu desempenho, foram feitas considerações sobre os custos.

• Influência da Fase de Busca na Vizinhança TTA: A influência da trocas de pares adjacentes de tarefas continua pequena. A melhora média obtida não ultrapassa os 5% para nenhum tipo de instância com n = 20 e decresce conforme aumenta o valor de n. Mesmo com esse resultado, conclui-se que não vale a pena abandonar essa fase pois o tempo computacional consumido não é crítico.

• Inserção de Intervalos Ociosos: A sexta coluna das tabelas 4.7 a 4.10 mostra o erro médio encontrado entre o custo final heurístico e o custo com inserção ótima de intervalos ociosos na seqüência heurística. Apenas para o tipo 11 com n = 40, tipo 3 para n = 60 e tipo 7 para n = 80 houve um erro médio acima dos 10%.

			Heurística			IO	Troca de Todos os Pares			
Tipo	Caso	h_i / t_i	Tempo(s)	Trocas	Melhora	Erro	Tempo(s)	Viz.	Erro Médio	Pior Caso
1	1	0,5	0,617	2,8	1,2	2,9	2,867	4,4	3,8	12,3
2		1,0	0,633	2,0	3,6	6,4	6,877	9,6	6,6	21,6
3		2,0	0,627	1,8	4,2	5,4	3,230	4,2	1,9	19,2
4		indep	0,430	0,4	0,2	6,3	5,483	7,4	8,3	33,6
5	2	0,5	0,633	1,4	0,2	5,5	6,117	8,6	10,0	45,9
6		1,0	0,427	0,6	1,3	8,3	4,977	6,6	8,7	43,2
7		2,0	0,817	3,6	2,6	7,1	2,787	3,8	6,4	32,3
8		indep	0,430	0,4	0,3	1,6	2,320	3,2	1,1	5,4
9	3	0,5	0,623	1,0	0,1	3,6	3,440	5,0	4,1	16,8
10		1,0	0,417	1,4	0,2	2,5	2,750	4,0	2,4	10,7
11		2,0	0,227	0,8	3,0	4,2	1,570	2,4	1,5	12,9
12		indep	0,223	0,6	0,3	4,7	2,403	3,6	1,4	9,9
13	4	0,5	0,423	0,4	0,1	0,3	2,087	3,2	1,2	4,7
14		1,0	0,417	1,2	1,2	0,7	2,993	4,4	2,8	7,3
15		2,0	0,820	1,4	1,0	3,7	1,694	2,6	0,3	10,9
16		indep	0,617	1,0	0,8	0,8	1,690	2,6	0,5	2,9
17	5	0,5	0,220	0,4	0,1	0,9	1,097	1,8	0,3	2,2
18		1,0	0,227	0,6	0,1	3,3	2,393	3,6	1,4	15,7
19		2,0	0,220	0,8	0,4	2,9	1,447	2,2	0,3	12,2
20		indep	0,423	0,2	0,3	0,7	0,907	1,4	-	1,4
21	6	0,5	0,230	0,2	-	1,1	1,497	2,4	0,5	5,1
22		1,0	0,623	0,4	0,1	0,3	1,867	2,8	0,7	2,8
23		2,0	0,413	2,4	2,7	2,2	1,340	2,0	0,4	6,3
24		indep	0,217	0,4	0,2	1,4	1,287	2,0	0,2	6,5
25	7	0,5	0,417	-	-	0,3	0,980	1,6	0,2	1,9
26		1,0	0,420	0,2	-	0,1	0,993	1,6	0,1	0,5
27		2,0	0,023	0,2	0,1	1,0	1,217	2,0	0,1	2,7
28		indep	0,417	0,4	0,1	0,2	1,754	2,8	0,4	1,0
29	8	0,5	0,423	-	-	0,1	0,837	1,4	0,1	0,7
30		1,0	0,420	-	-	0,9	1,873	3,0	1,4	5,2
31		2,0	0,227	-	-	0,6	0,743	1,2	-	3,0
32		indep	0,420	0,4	0,1	0,1	1,117	1,8	0,2	0,5
33	9	0,5	0,414	0,2	-	0,1	1,270	2,0	0,2	0,5
34		1,0	0,223	0,6	-	0,2	1,033	1,6	0,1	1,2
35		2,0	0,230	0,2	-	-	1,347	2,2	0,4	1,9
36		indep	0,420	-	-	0,4	1,967	3,0	0,5	2,0

Tabela 4.7 - Resultados Obtidos para instâncias com $n = 20$.

			Heurística			IO	Troca de Todos os Pares			
Tipo	Caso	h_j / t_j	Tempo(s)	Trocas	Melhora	Erro	Tempo(s)	Viz.	Erro Médio	Pior Caso
1	1	0,5	1,837	4,8	0,7	1,0	141,490	23,2	6,9	16,5
2		1,0	0,723	2,6	0,7	5,0	70,613	18,0	5,5	16,7
3		2,0	0,930	4,8	4,2	7,8	49,493	13,6	6,0	35,5
4		indep	0,697	2,4	0,8	4,6	57,180	16,8	3,3	19,7
5	2	0,5	0,696	4,8	1,0	2,3	58,160	18,4	10,1	26,0
6		1,0	0,484	3,8	0,2	1,3	27,083	8,4	2,1	9,0
7		2,0	1,327	5,2	3,7	6,6	30,180	8,6	2,4	15,1
8		indep	0,477	2,6	0,7	5,6	43,710	14,4	4,8	20,8
9	3	0,5	0,883	3,4	0,2	2,2	42,797	12,0	3,7	16,2
10		1,0	0,690	3,2	0,5	3,5	31,367	9,4	3,6	11,0
11		2,0	0,533	6,6	2,9	12,9	28,977	8,2	2,8	30,5
12		indep	0,690	2,0	0,4	2,4	21,587	7,6	2,2	8,2
13	4	0,5	0,677	1,6	0,4	1,6	44,023	15,0	2,6	5,3
14		1,0	0,070	1,6	0,2	1,3	11,083	4,2	0,5	4,3
15		2,0	0,480	5,0	1,0	4,8	11,027	4,0	0,6	11,1
16		indep	0,463	1,2	0,4	2,1	26,690	10,4	1,8	6,5
17	5	0,5	0,870	0,6	-	0,8	14,683	5,2	0,7	2,4
18		1,0	0,470	1,6	0,7	3,7	34,333	13,0	2,1	14,1
19		2,0	0,503	1,2	0,3	0,5	13,603	5,0	0,3	1,5
20		indep	0,670	0,4	0,1	1,3	18,160	7,0	0,9	3,5
21	6	0,5	0,497	1,4	0,2	0,3	17,830	6,6	0,7	1,6
22		1,0	0,680	1,2	-	1,0	14,757	5,8	1,1	4,1
23		2,0	0,287	2,2	0,4	2,5	14,010	4,8	0,9	8,5
24		indep	0,883	-	-	1,0	15,650	6,0	0,6	4,1
25	7	0,5	0,483	0,6	-	0,2	5,140	2,2	-	0,6
26		1,0	0,270	0,2	-	0,1	4,583	2,0	-	0,3
27		2,0	0,267	0,2	-	0,7	9,257	3,6	0,1	1,7
28		indep	0,657	-	-	0,3	5,177	2,4	0,1	1,2
29	8	0,5	0,277	-	-	0,2	2,294	1,0	-	0,1
30		1,0	0,477	0,8	0,2	0,5	22,100	8,6	0,7	2,5
31		2,0	0,093	0,6	-	0,4	3,333	1,4	-	1,4
32		indep	0,270	0,4	-	0,3	8,543	3,6	0,3	2,2
33	9	0,5	0,467	0,4	-	0,1	17,503	6,6	0,9	2,8
34		1,0	0,273	1,0	-	0,2	9,827	4,0	0,2	1,1
35		2,0	0,697	2,0	0,3	1,1	5,717	2,4	0,2	2,5
36		indep	0,660	0,2	-	0,4	8,950	4,0	0,3	1,2

Tabela 4.8 - Resultados Obtidos para instâncias com $n = 40$.

			Heurística			IO	Troca de Todos os Pares			
Tipo	Caso	h_i/t_i	Tempo(s)	Trocas	Melhora	Erro	Tempo(s)	Viz.	Erro Médio	Pior Caso
1	1	0,5	1,067	12,2	0,7	1,6	326,687	30,0	6,3	13,3
2		1,0	1,097	5,6	0,5	5,8	381,467	32,8	5,4	20,7
3		2,0	1,440	11,6	2,8	11,4	153,237	13,0	2,7	49,6
4		indep	1,150	11,8	1,4	3,3	284,027	28,2	4,9	17,2
5	2	0,5	1,243	5,2	0,4	1,9	320,247	29,8	3,1	8,0
6		1,0	0,907	1,6	0,1	2,1	202,206	20,0	4,5	12,3
7		2,0	1,513	9,6	1,8	6,3	182,353	16,8	3,3	17,6
8		indep	0,587	6,6	0,5	3,4	191,653	21,6	2,9	11,8
9	3	0,5	1,213	3,0	0,1	1,3	167,027	18,0	4,0	9,2
10		1,0	1,333	5,2	0,5	4,0	132,123	13,8	3,1	12,7
11		2,0	1,000	3,6	0,7	2,7	214,007	20,2	3,8	10,6
12		indep	0,987	6,0	0,9	2,6	117,563	13,2	2,1	7,0
13	4	0,5	0,540	1,2	-	1,2	155,627	17,0	1,3	4,6
14		1,0	0,990	3,2	0,1	1,0	111,410	12,4	0,8	2,5
15		2,0	1,140	5,4	0,8	3,3	100,210	12,0	0,7	10,0
16		indep	0,943	5,0	0,4	1,6	55,817	7,2	0,5	5,8
17	5	0,5	0,960	2,2	-	0,4	172,027	22,6	2,0	4,8
18		1,0	1,147	3,4	0,2	0,8	75,840	8,8	0,8	4,0
19		2,0	1,190	4,6	0,3	2,0	76,616	9,4	0,7	4,9
20		indep	1,000	1,0	0,1	0,8	74,190	10,0	0,9	3,5
21	6	0,5	0,843	1,0	0,1	0,2	48,910	6,8	0,7	2,7
22		1,0	0,800	2,2	0,1	0,9	97,920	12,2	0,8	2,9
23		2,0	0,773	1,0	0,1	1,5	29,603	3,8	0,2	3,3
24		indep	0,603	1,0	0,1	0,6	34,797	5,0	0,2	1,0
25	7	0,5	0,370	0,6	-	0,1	33,100	5,2	0,1	0,3
26		1,0	0,747	0,8	0,1	0,1	28,927	4,6	0,1	0,7
27		2,0	0,797	0,8	0,1	0,3	21,187	3,2	0,1	0,7
28		indep	0,547	0,4	-	0,1	22,274	3,6	0,1	0,4
29	8	0,5	0,970	0,2	-	-	41,933	6,6	0,1	0,2
30		1,0	0,580	0,2	-	0,2	39,567	5,6	0,4	1,8
31		2,0	0,417	0,2	-	0,2	23,690	3,6	0,1	0,6
32		indep	1,190	0,6	-	0,1	25,263	4,0	0,1	0,5
33	9	0,5	0,780	0,8	-	0,1	83,417	11,4	0,5	0,9
34		1,0	0,563	0,2	-	0,1	26,760	4,0	0,1	0,8
35		2,0	0,957	1,8	0,1	0,2	34,856	5,2	0,1	0,7
36		indep	0,767	0,6	-	0,1	31,913	5,0	0,1	0,4

Tabela 4.9 - Resultados Obtidos para instâncias com n = 60.

			Heurística			IO	Troca de Todos os Pares			
Tipo	Caso	h_1/t_1	Tempo(s)	Trocas	Melhora	Erro	Tempo(s)	Viz.	Erro Médio	Pior Caso
1	1	0,5	1,913	14,2	0,4	1,4	884,247	40,6	4,1	7,9
2		1,0	1,947	8,4	0,4	0,8	1399,640	59,6	7,7	13,8
3		2,0	2,023	4,6	1,0	5,1	781,777	36,8	4,2	18,0
4		indep	1,657	11,0	0,8	3,3	464,480	24,8	1,9	7,7
5	2	0,5	1,620	4,0	0,2	1,6	478,623	29,4	2,8	9,1
6		1,0	1,333	8,4	0,5	2,9	686,263	36,2	3,0	7,2
7		2,0	1,280	7,6	0,5	11,1	434,747	24,2	2,1	29,9
8		indep	1,337	4,0	0,3	3,3	446,520	24,4	3,0	10,9
9	3	0,5	1,160	5,0	0,1	1,9	324,047	20,6	1,7	8,1
10		1,0	1,367	5,0	0,2	1,4	367,163	24,0	1,8	4,8
11		2,0	1,993	9,4	1,5	3,9	349,837	22,2	1,9	12,8
12		indep	1,160	3,0	0,2	2,2	495,963	29,0	2,8	7,8
13	4	0,5	1,383	3,4	0,1	0,8	425,493	28,0	1,7	4,9
14		1,0	1,553	4,4	0,4	1,1	419,206	26,6	1,6	4,8
15		2,0	1,283	6,6	1,0	1,2	208,390	12,6	0,3	3,2
16		indep	1,087	5,2	0,3	0,9	372,783	27,0	1,1	3,8
17	5	0,5	1,523	3,4	0,1	0,5	236,904	17,4	0,8	2,1
18		1,0	1,080	1,8	-	0,7	217,553	15,0	0,6	2,8
19		2,0	1,633	3,4	0,2	0,9	133,007	9,8	0,6	2,9
20		indep	0,813	4,6	0,1	1,0	163,693	11,8	0,6	2,4
21	6	0,5	1,423	2,4	-	0,2	300,660	20,4	1,2	3,3
22		1,0	1,103	2,4	0,1	0,3	117,143	8,8	0,2	1,0
23		2,0	1,187	1,8	0,1	0,8	102,377	7,2	0,3	1,7
24		indep	1,330	0,8	-	0,7	125,107	9,4	0,2	1,4
25	7	0,5	1,083	0,8	-	0,1	77,333	6,4	0,2	0,7
26		1,0	1,137	1,0	-	0,1	84,406	7,2	0,1	0,3
27		2,0	0,890	3,0	0,1	0,4	64,117	5,4	0,2	1,3
28		indep	1,260	0,8	-	0,1	83,087	7,2	0,2	0,4
29	8	0,5	1,687	0,8	-	-	65,804	5,6	0,2	0,8
30		1,0	1,130	0,4	-	-	104,873	8,4	0,2	0,6
31		2,0	1,240	1,8	-	0,7	66,420	5,6	0,1	1,4
32		indep	0,667	1,2	-	0,1	185,663	15,8	0,4	1,2
33	9	0,5	1,287	0,8	-	0,1	62,867	5,2	0,1	0,8
34		1,0	1,253	1,2	-	0,1	45,196	3,8	0,1	0,3
35		2,0	0,890	0,8	0,1	0,1	112,027	9,2	0,2	0,7
36		indep	1,077	0,6	-	-	61,773	5,6	0,1	0,5

Tabela 4.10 - Resultados Obtidos para instâncias com n = 80.

• Vizinhanças exploradas: O número de vizinhanças exploradas pela busca TTP foi maior que o esperado, mostrando que, conforme aumenta o número de tarefas em uma instância, o número de erros de posicionamento de tarefas também aumenta. Observa-se que, para os 10 primeiros tipos de instância, o número de vizinhanças é maior que para os demais tipos, embora essa diferença não seja muito acentuada para as instâncias com n = 20.

• Comparação de custos: Com exceção do quinto tipo para n = 40, nenhuma dos erros médios calculados ultrapassou 10%. Esse erro foi calculado comparando-se o custo do ótimo local da busca TTP com o custo da sequência de produção determinada pela heurística após a inserção ótima de intervalos ociosos feita pelo procedimento TIMETABLER (modificado). Isso sugere que a solução final gerada pela heurística, apesar de conter erros de posicionamento de tarefas está bastante próxima, em termos de custo, do ótimo local de sua vizinhança TTP.

• Pior Caso: A última coluna das tabelas 4.7 a 4.10 mostra os valores de melhora ocorrida na realização da busca TTP em relação à pior solução heurística encontrada. alguns dos valores de pior caso são bastante altos, principalmente para os dez primeiros tipos de instâncias.

Observa-se, pela evolução dos resultados ao longo de cada uma das tabelas 4.7 a 4.10 que as piores soluções heurísticas são produzidas entre o primeiro e o décimo primeiro tipo de instâncias. Também pode ser verificado que a relação entre os custos de avanço e atraso, h_i / t_i , não influi no desempenho da heurística. Novamente, verifica-se a influência dos intervalos de geração das datas de entrega. Tanto os valores de tempo computacional, como as porcentagens de erro e melhora decrescem sensivelmente a partir do décimo segundo tipo, confirmando o que foi discutido com relação aos resultados das instâncias com 8 tarefas. Observa-se, também que, conforme aumenta o valor de n, o erro médio diminui para as instâncias do oitavo tipo em diante.

Considerando-se as tabelas 4.7 a 4.10 em conjunto, pode-se observar que a melhora obtida com a aplicação da Busca em Vizinhança com Troca de Todos os Pares de Tarefas diminui conforme aumenta o número de tarefas nas instâncias.

4.4 - CONCLUSÕES

A partir dos testes computacionais realizados, conclui-se que, o procedimento heurístico ainda comete erros no posicionamento de tarefas e na inserção de intervalos ociosos. A consideração de deslocamentos à direita no Procedimento III da fase construtiva pode melhorar o desempenho da heurística quanto a esse último aspecto. O desempenho fica prejudicado para instâncias em que o intervalo de geração das datas de entrega possui uma média alta (datas de entrega menos restritivas). Isso significa que a heurística é favorecida pela dispersão das datas de entrega em torno de uma média relativamente restritiva.

O desempenho da heurística para problemas com 8 tarefas é razoável para a maioria dos problemas. Os erros médios relativos entre os custos heurísticos e os ótimos ficam abaixo de 10%, exceto para alguns tipos de instâncias. A análise do erro de pior caso mostra que algumas das instâncias apresentam erros altos mas, dentro de um mesmo tipo de instância, em geral uma delas apresenta um erro muito grande e as outras apresentam erros próximos de zero. Observou-se que o estágio de busca local na vizinhança na vizinhança TTA tem pouca influência no valor final da solução heurística. A heurística é claramente mais rápida que o algoritmo ótimo, com um tempo médio para os 36 tipos de problemas igual a 0,073 segundos. O algoritmo ótimo possui um tempo médio igual a 19,180 segundos. A análise dos resultados computacionais mostra que a heurística tem um desempenho melhor quando as datas de entrega se tornam mais restritivas. A relação entre as penalidades de avanço e atraso parecem não influenciar o desempenho da heurística.

Para os problemas com 20 a 80 tarefas, a heurística tem um desempenho razoável com respeito à melhor solução encontrada através da busca local na vizinhança TTP. O erro médio relativo é menor que 10%, exceto para um tipo de problema com n = 40. Em geral, o pior erro encontrado entre os cinco problemas de cada tipo é alto para os primeiro onze tipos de instâncias e diminuindo conforme as datas de entrega se tornam mais restritivas. A análise de resultados também mostra que os valores de erro encontrados decrescem conforme o número de tarefas cresce. Mesmo para os problemas maiores, a busca local na vizinhança TTA apresentou uma melhoria média em relação à fase construtiva menor que 5% e o número de trocas de tarefas adjacentes mantém-se muito pequeno. O tempo médio da heurística é muito bom. Esse tempo é menor que 2 segundos, mesmo para problemas com 80 tarefas. Apenas para efeito de comparação, a busca local na vizinhança TTP alcança um tempo igual a 1399,64 segundos para um dos tipos de problema com 80 tarefas. Assim como nos problemas com 8 tarefas, o desempenho da

heurística melhora quando as datas de entrega são mais restritivas e a relação entre as penalidades de avanço e atraso não parece influenciar a solução final.

Com o objetivo de melhorar os valores de custo obtidos pela heurística, sugere-se como pesquisa futura a consideração de deslocamentos de tarefas à direita na atualização de intervalos ociosos. Esses deslocamentos podem influir inclusive nos erros de posicionamento de tarefas, visto que a heurística proposta insere intervalos ociosos durante o processo de construção. Inserindo esses intervalos de forma mais precisa no programa parcial pode fazer com que os erros de sequenciamento observados diminuam.

O cálculo da complexidade da heurística é mostrado no Apêndice B. Calcula-se a complexidade de pior caso para cada um dos procedimentos da heurística e obtém-se como complexidade da heurística aquela que for dominante entre todos os procedimentos. O Procedimento II possui essa complexidade dominante, que é igual a $O(n^{n+1})$. Essa complexidade é causada pelo processo de retirada e reinserção de tarefas no programa parcial durante o processo construção de um programa factível.

Uma possível mudança no sentido de melhorar a complexidade do Procedimento de Factibilização é a alteração no critério de ordenação utilizado no Procedimento I. Se, ao ocorrer empate nos instantes iniciais dos Intervalos Preferenciais de Posicionamento das tarefas, o desempate for feito pela regra LPT (do inglês "Longest Processing Time"), a complexidade resultante para o Procedimento II é $O(n^2)$. Mas, a melhoria na complexidade do procedimento é apenas aparente pois é possível encontrar uma instância de pior caso em que todas as tarefas possuam tempos de processamento iguais, obtendo-se uma complexidade igual a $O(n^{n+1})$.

Analisando o Procedimento II com o objetivo de encontrar pontos de possíveis mudanças sugere-se que, quando a situação em que mais de uma tarefa cause infactibilidades no programa parcial ocorrer, não se retire nenhuma tarefa de **S**. Uma decisão alternativa pode ser tomada e é ilustrada na Figura 4.1.

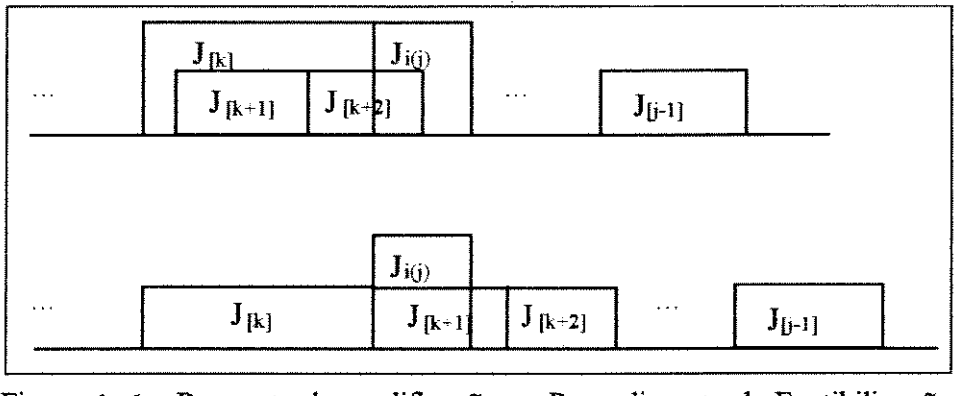

Figura 4.1 - Proposta de modificação no Procedimento de Factibilização

A modificação proposta consiste em deslocar à direita as tarefas em posições posteriores a $J_{[k]}$ de forma que, novamente, apenas a tarefa $J_{1(j)}$ seja responsável por infactibilidades no Programa parcial **S**.

Dessa forma, a complexidade do Procedimento II passa a ser $O(n^2)$ e a complexidade dominante entre todos os procedimentos torna-se a do Procedimento de Inserção de Intervalos Ociosos e a complexidade da heurística torna-se $O(n^3)$.

APÊNDICE A

VERIFICAÇÃO DA VALIDADE DO TEOREMA DE ADJACÊNCIA ÓTIMA GENERALIZADO

Este apêndice apresenta a verificação da validade do Teorema de Adjacência Ótima Generalizado para os casos:

- $a_i = a_j = 1$;
- $a_i = 0$ e $a_j = 1$;
- $a_i = 1$ e $a_j = 0$; e
- $a_i = a_j = 0$.

Para um melhor acompanhamento das verificações, é repetido o enunciado do teorema:

TEOREMA DE ADJACÊNCIA ÓTIMA GENERALIZADO:

"A tarefa J_i precede imediatamente a tarefa J_j se a seguinte condição for verdadeira:

$$t_i.(p_j + \max\{0, r_j - s_i\}) - \Omega_{ij}.(h_i + t_i) - t_j.\min\{p_i, s_j - r_j\} + \Omega_{ji}.(h_j + t_j) \geq 0 \quad (3.29)$$

onde:

$$\Omega_{ij} = \begin{cases} 0, & \text{se } \Delta_i < 0; \\ \Delta_i, & \text{se } 0 \leq \Delta_i < p_j + \max\{0, r_j - s_i\} \\ p_j + \max\{0, r_j - s_i\}, & \text{caso contrário} \end{cases};$$

$$\Omega_{ji} = \begin{cases} 0, & \text{se } \Delta_j < 0; \\ \Delta_j, & \text{se } 0 \leq \Delta_j < \min\{p_i, s_j - r_j\} \\ \min\{p_i, s_j - r_j\}, & \text{caso contrário} \end{cases};$$

$\Delta_i = d_i - p_i - r_i$ e

$\Delta_j = d_j - p_j - \max\{r_j, s_i\}$."

CASO 1:

$a_i = a_j = 1 \rightarrow$

As duas tarefas estão sempre atrasadas, pois:

$a_i = 1 \rightarrow d_i < r_i + p_i$ e

$a_j = 1 \rightarrow d_j < r_j + p_j$.

Sendo assim, a localização exata de d_i e d_j não é importante. Esses valores são omitidos nas Figuras A.1 e A.2.

$\Delta_j = d_j - p_j - s_i \leq d_j - p_i - r_i < 0 \Rightarrow \Omega_{ij} = 0;$

$\Delta_j = d_j - p_j - s_j \leq d_j - p_j - r_j < 0 \Rightarrow \Omega_{ji} = 0.$

Subcaso A: $r_j \leq s_i$ - Ver Figura A.1.

$p_j + \max\{0, r_j - s_i\} = p_j;$

$\min\{p_i, s_j - r_j\} = p_i;$

De 3.29:

$$t_i . p_j - t_j . p_i \geq 0$$

$custo(ij) = t_i.(s_i + p_i - d_i) + t_j.(s_i + p_i + p_j - d_j)$

$custo(ji) = t_i.(s_i + p_i + p_j - d_i) + t_j.(s_i + p_j - d_j)$

$custo(ji) - custo(ij) = t_i . p_j - t_j . p_i \geq 0 \Rightarrow$

$custo(ji) \geq custo(ij)$

Portanto J_i precede J_j.

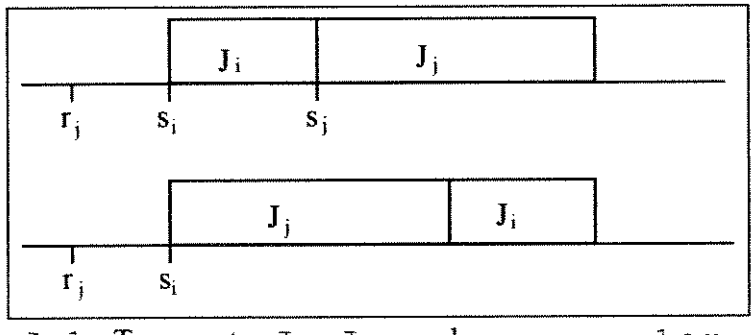

Figura A.1 - Troca entre J_i e J_j quando $a_i = a_j = 1$ e $r_j \leq s_i$.

Subcaso B: $r_j > s_i$ - Ver Figura A.2.

$p_j + \max\{0, r_j - s_i\} = p_j + r_j - s_i;$

$\min\{p_i, s_j - r_j\} = s_j - r_j;$

De 3.29:

$$t_i . (p_j + r_j - s_i) - t_j . (s_j - r_j) \geq 0$$

$custo(ij) = t_i.(s_i + p_i - d_i) + t_j.(s_j + p_j - d_j)$

$custo(ji) = t_i.(r_j + p_j + p_i - d_i) + t_j.(r_j + p_j - d_j)$

$custo(ji) - custo(ij) = t_i.(p_j + r_j - s_i) - t_j.(s_j - r_j) \geq 0 \Rightarrow$

$custo(ji) \geq custo(ij)$

Portanto, J_i precede J_j.

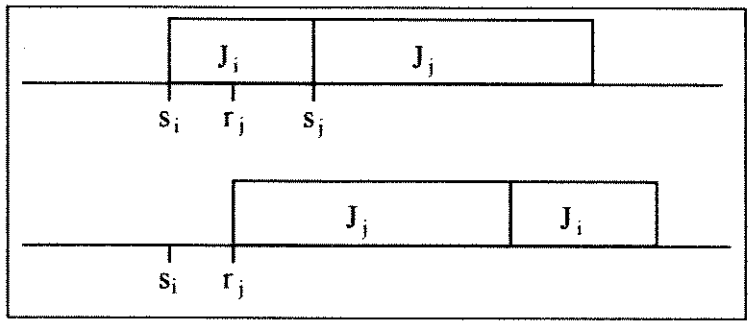

Figura A.2 - Troca entre J_i e J_j quando $a_i = a_j = 1$ e $r_j > s_i$.

CASO 2: $a_i = 0$ e $a_j = 1 \rightarrow$

A tarefa J_j está sempre atrasada, pois

$$a_j = 1 \rightarrow d_j < r_j + p_j.$$

Portanto: $\Delta_j = d_j - p_j - s_j \leq d_j - p_j - r_j \leq 0 \Rightarrow \Omega_{ji} = 0$

Sendo assim, a localização exata de d_j não é importante e esse valor é omitido nas figuras que ilustram este caso.

A tarefa J_i pode estar adiantada ou atrasada, antes ou depois da troca conforme seja a localização de d_i.

Antes da troca:

$d_i \geq s_i + p_i \rightarrow J_i$ adiantada;

$d_i < s_i + p_i \rightarrow J_i$ atrasada.

Depois da troca:

$d_i \geq s_i + \max\{0, r_j - s_i\} + p_j + p_i$

$= \max\{r_j, s_i\} + p_j + p_i \rightarrow J_i$ adiantada;

$d_i < \max\{r_j, s_i\} + p_j + p_i \rightarrow J_i$ atrasada.

Existem, então, quatro combinações de atraso e avanço para J_i:

1) J_i atrasada antes e depois da troca:

$$d_i < s_i + p_i$$

2) J_i adiantada antes e atrasada depois da troca:

$$s_i + p_i \leq d_i < \max\{r_j, s_i\} + p_j + p_i$$

3) J_i adiantada antes e depois da troca:

$$d_i \geq \max\{r_j, s_i\} + p_j + p_i$$

4) J_i atrasada antes e adiantada depois da troca: essa combinação é fisicamente impossível, pois, se J_i se desloca à direita, pode ocorrer somente um aumento no atraso. Portanto, a demonstração se restringe às combinações 1, 2 e 3.

Combinação 1:

$$d_i < s_i + p_i \Rightarrow$$

$$\Delta_i = d_i - s_i - p_i < 0 \Rightarrow \Omega_{ij} = 0.$$

Subcaso A: $r_j \leq s_i$ - Ver Figura A.3.

$p_j + \max\{0, r_j - s_i\} = p_j$;

$\min\{p_i, s_j - r_j\} = p_i$;

De (3.29):

$$t_i . p_j - t_j . p_i \geq 0.$$

$$custo(ij) = t_i.(s_i + p_i - d_i) + t_j.(s_i + p_i + p_j - d_j)$$

$$custo(ji) = t_i.(s_i + p_j + p_i - d_i) + t_j.(s_i + p_j - d_j)$$

$$custo(ji) - custo(ij) = t_i . p_j - t_j . p_i \geq 0 \Rightarrow$$

$custo(ji) \geq custo(ij)$.

Portanto, J_i precede J_j.

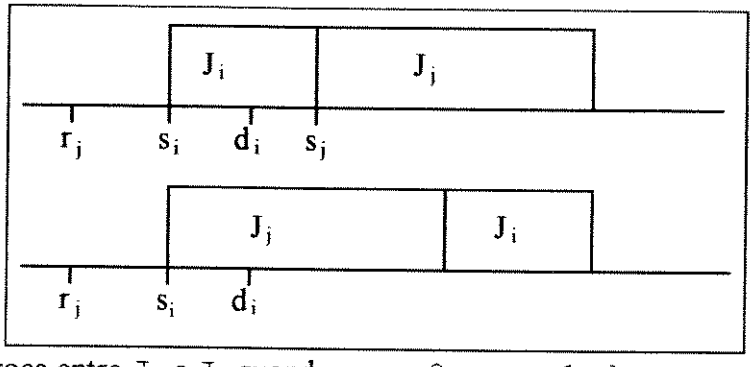

Figura A.3 - Troca entre J_i e J_j quando $a_i = 0$ e $a_j = 1$, $d_i < s_i + p_i$ e $r_j \leq s_i$.

Subcaso B: $r_j > s_i$ - Ver Figura A.4.

$p_j + \max\{0, r_j - s_i\} = p_j + r_j - s_i$;

$\min\{p_i, s_j - r_j\} = s_j - r_j$;

De (3.29):

$$t_i.(p_j + r_j - s_i) - t_j.(s_j - r_j) \geq 0.$$

$$custo(ij) = t_i.(s_i + p_i - d_i) + t_j.(s_i + p_i + p_j - d_j)$$

$$custo(ji) = t_i.(r_j + p_j + p_i - d_i) + t_j.(r_j + p_j - d_j)$$

$$custo(ji) - custo(ij) = t_i.(p_j + r_j - s_i) - t_j.(s_j - r_j) \geq 0 \Rightarrow$$

$custo(ji) \geq custo(ij)$

Portanto, J_i precede J_j.

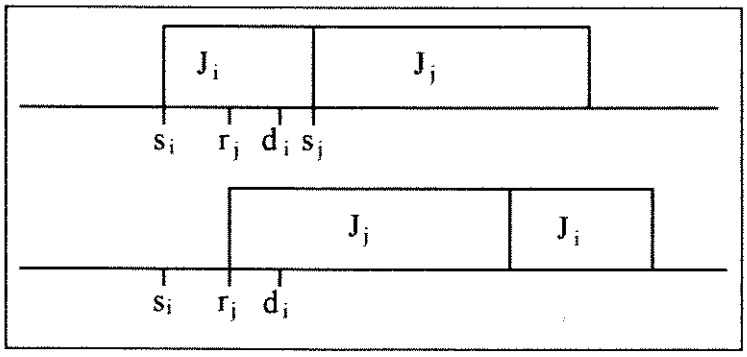

Figura A.4 - Troca entre J_i e J_j quando $a_i = 0$ e $a_j = 1$, $d_i < s_i + p_i$ e $r_j > s_i$.

Combinação 2:

$$s_i + p_i \leq d_i < \max\{r_j, s_i\} + p_j + p_i \Rightarrow$$
$$0 \leq d_i - s_i - p_i < p_j + \max\{0, r_j - s_i\} \Rightarrow$$
$$0 \leq \Delta_{ij} < p_j + \max\{0, r_j - s_i\} \Rightarrow$$
$$\Omega_{ij} = d_i - s_i - p_i.$$

Subcaso A: $r_j \leq s_i$ - Ver Figura A.5.

$p_j + \max\{0, r_j - s_i\} = p_j$;

$\min\{p_i, s_j - r_j\} = p_i$;

De (3.29):

$t_i.p_j - (d_i - p_i - s_i).(t_i + h_i) - t_j.p_i \geq 0 \Rightarrow$

$t_i.(s_i + p_j + p_i - d_i) - h_i.(d_i - p_i - s_i) - t_j.p_i \geq 0.$

$custo(ij) = h_j.(d_i - s_i - p_i) + t_j.(s_i + p_i + p_j - d_j)$

$custo(ji) = t_i.(s_i + p_j + p_i - d_i) + t_j.(s_i + p_j - d_j)$

$custo(ji) - custo(ij) =$

$t_i.(s_i + p_j + p_i - d_i) - h_i.(d_i - s_i - p_i) - t_j.p_i \geq 0 \Rightarrow$

$custo(ji) \geq custo(ij)$.

Portanto, J_i precede J_j.

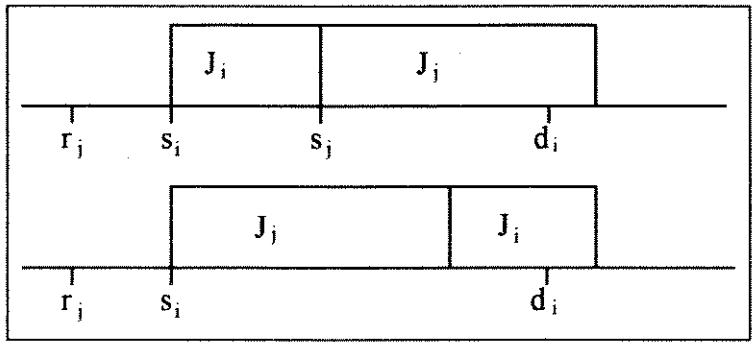

Figura A.5 - Troca entre J_i e J_j quando $a_i = 0$ e $a_j = 1$, $s_i + p_i \leq d_i < \max\{r_j, s_i\} + p_j + p_i$ e $r_j \leq s_i$.

Subcaso B: $r_j > s_i$ - Ver Figura A.6.

$p_j + \max\{0, r_j - s_i\} = p_j + r_j - s_i$;

$\min\{p_i, s_j - r_j\} = s_j - r_j$;

De (3.29):

$$t_i.(p_j + r_j - s_i) - (d_i - p_i - s_i).(h_i + t_i) - t_j.(s_j - r_j) \geq 0 \Rightarrow$$

$$t_i.(r_j + p_j + p_i - d_i) - h_i.(d_i - p_i - s_i) - t_j.(s_j - r_j) \geq 0$$

$custo(ij) = h_i.(d_i - s_i - p_i) + t_j.(s_i + p_i + p_j - d_j)$

$custo(ji) = t_i.(r_j + p_j + p_i - d_i) + t_j.(r_j + p_j - d_j)$

$custo(ji) - custo(ij) =$

$$t_i.(r_j + p_j + p_i - d_i) - h_i.(d_i - s_i - p_i) - t_j.(s_j - r_j) \geq 0 \Rightarrow$$

$custo(ji) \geq custo(ij)$

Portanto, J_i precede J_j.

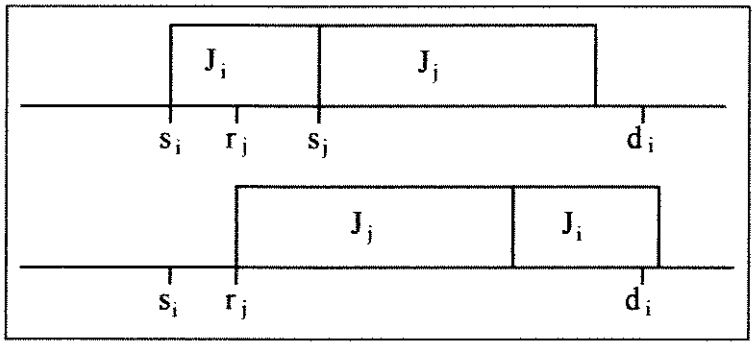

Figura A.6 - Troca entre J_i e J_j quando $a_i = 0$ e $a_j = 1$, $s_i + p_i \leq d_i < \max\{r_j, s_i\} + p_j + p_i$ e $r_j > s_i$.

Combinação 3:

$$d_i \geq \max\{r_j, s_i\} + p_j + p_i \Rightarrow$$

$$\Delta_i = d_i - s_i - p_i \geq p_j + \max\{0, r_j - s_i\} \Rightarrow$$

$$\Omega_{ij} = p_j + \max\{0, r_j - s_i\}.$$

Como a tarefa J_i está adiantada e a tarefa J_j está atrasada, seu posicionamento não é ótimo e a troca deve ser feita. A troca causará uma diminuição do custo de avanço de J_i e do custo de atraso de J_j. Portanto, a expressão (3.29) não é verdadeira nesse caso.

Subcaso A: $r_j \leq s_i$ - Ver Figura A.7.

$p_j + \max\{0, r_j - s_i\} = p_j;$

$\min\{p_i, s_j - r_j\} = p_i;$

De (3.29):

$$t_i . p_j - p_j . (t_i + h_i) - t_j . p_i \geq 0$$

$$\Rightarrow -h_i . p_j - t_j . p_i \geq 0.$$

A desigualdade obtida nunca é verdadeira pois todos os parâmetros são positivos. Portanto, custo(ji) < custo(ij) e J_j precede J_i.

$$\text{custo(ij)} = h_i . (d_i - s_i - p_i) + t_j . (s_i + p_i + p_j - d_j)$$

$$\text{custo(ji)} = h_i . (d_i - s_i - p_j - p_i) + t_j . (s_i + p_j - d_j)$$

$$\text{custo(ji)} - \text{custo(ij)} = -h_i . p_j - t_j . p_i \leq 0$$

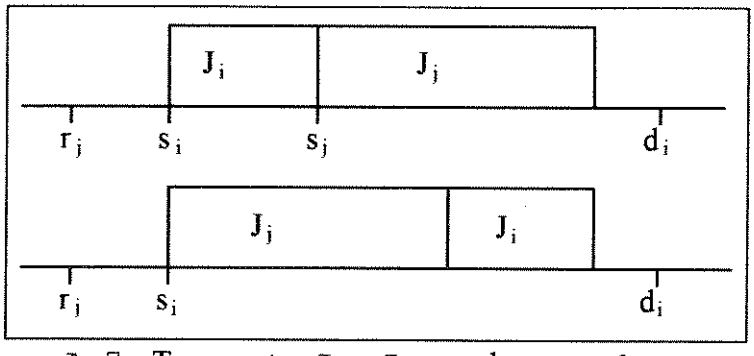

Figura A.7 - Troca entre J_i e J_j quando $a_i = 0$ e $a_j = 1$, $d_i \geq \max\{r_j, s_i\} + p_j + p_i$ e $r_j \leq s_i$.

Subcaso B: $r_j > s_i$ - Ver Figura A.8.

$p_j + \max\{0, r_j - s_i\} = p_j + r_j - s_i;$

$\min\{p_i, s_j - r_i\} = s_j - r_j;$

De (3.29):

$$t_i.(p_j + r_j - s_i) - (p_j + r_j - s_i).(h_i + t_i) - t_j.(s_j - r_i) \geq 0$$

$$\Rightarrow -h_i.(p_j + r_j - s_i) - t_j.(s_j - r_i) \geq 0$$

Novamente, chega-se a uma desigualdade falsa, confirmando que, para esta combinação de atraso e avanço, J_j precede J_i.

$$custo(ij) = h_i.(d_i - s_i - p_i) + t_j.(s_i + p_i + p_j - d_j)$$

$$custo(ji) = h_i.(d_i - r_j - p_j - p_i) + t_i.(r_j + p_j - d_j)$$

$$custo(ji) - custo(ij) = - h_i.(p_j + r_j - s_i) - t_j.(s_i + p_i - r_j) = - h_i.(p_j + r_j - s_i) - t_i.(s_i - r_j) \leq 0$$

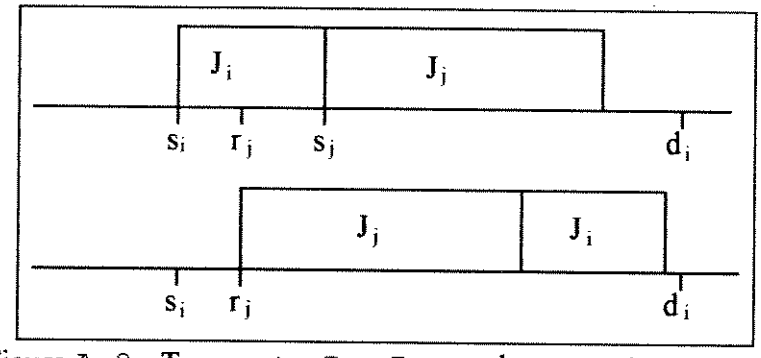

Figura A.8 - Troca entre J_i e J_j quando $a_i = 0$ e $a_j = 1$, $d_i \geq \max\{r_j, s_i\} + p_j + p_i$ e $r_j > s_i$.

CASO 3: $a_i = 1$ e $a_j = 0 \rightarrow$

A tarefa J_i está sempre atrasada, pois

$$a_i = 1 \rightarrow d_i < r_i + p_i.$$

Portanto: $\Delta_i = d_i - p_i - s_i \leq d_i - p_i - r_i \leq 0 \Rightarrow \Omega_{ij} = 0$

Sendo assim, a localização exata de d_i não é importante e esse valor é omitido nas figuras que ilustram as demonstrações para este caso.

A tarefa J_j pode estar adiantada ou atrasada, antes ou depois da troca dependendo da posição de d_j.

Antes da troca:

$d_j \geq s_i + p_i + p_j \rightarrow J_j$ adiantada;

$d_j < s_i + p_i + p_j \rightarrow J_j$ atrasada.

Depois da troca:

$d_j \geq \max\{r_j, s_i\} + p_j \rightarrow J_j$ adiantada;

$d_j < \max\{r_j, s_i\} + p_j \rightarrow J_j$ atrasada.

Tem-se, então, quatro combinações de atraso e avanço para J_j:

1) J_j atrasada antes e depois da troca:

$d_j < \max\{r_j, s_i\} + p_j$

2) J_j atrasada antes e adiantada depois da troca:

$\max\{r_j, s_i\} + p_j \leq d_j < s_i + p_i + p_j$

3) J_j adiantada antes e depois da troca:

$d_j \geq s_i + p_j + p_i$

4) J_j adiantada antes e atrasada depois da troca: como observado na Combinação 4 do Caso 2, essa combinação é fisicamente impossível, pois, se J_j se desloca à esquerda, pode ocorrer somente um aumento no avanço. Dessa forma, serão somente demostradas as combinações 1, 2 e 3.

Combinação 1:

$d_j < \max\{r_j, s_i\} + p_j \Rightarrow$

$\Delta_j < 0 \Rightarrow \Omega_{ji} = 0.$

Subcaso A: $r_j \leq s_i$ - Ver Figura A.9.

$p_j + \max\{0, r_j - s_i\} = p_i;$

$\min\{p_i, s_j - r_j\} = p_i;$

De (3.29):

$t_i . p_j - t_j . p_i \geq 0$

$custo(ij) = t_i.(s_i + p_i - d_i) + t_j.(s_i + p_i + p_j - d_j)$

$custo(ji) = t_i.(s_i + p_j + p_i - d_i) + t_j.(s_i + p_j - d_j)$

$custo(ji) - custo(ij) = t_i . p_j - t_j . p_i \geq 0 \Rightarrow$

$custo(ji) \geq custo(ij)$.

Portanto, J_i precede J_j.

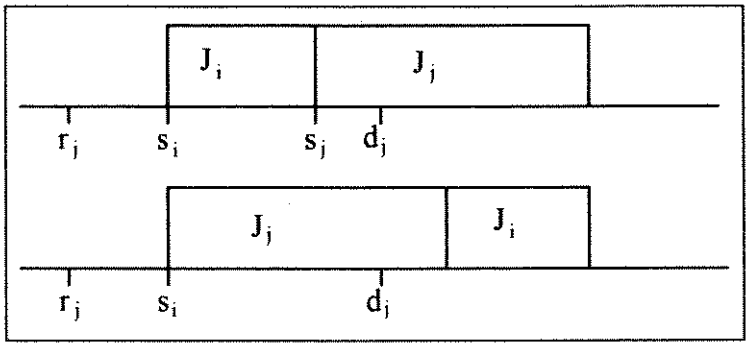

Figura A.9 - Troca entre J_i e J_j quando $a_j = 1$ e $a_i = 0$, $d_j < \max\{r_j, s_i\} + p_j$ e $r_j \leq s_i$.

Subcaso B: $r_j > s_i$.

Esse caso nunca ocorre na prática pois

$$a_j = 0 \rightarrow d_j \geq r_j + p_j.$$

Portanto, quando $r_j > s_i$, J_j nunca poderá estar atrasada após da troca.

Combinação 2:

$$\max\{r_j, s_i\} + p_j \leq d_j < s_i + p_i + p_j \Rightarrow$$

$$0 \leq d_j - p_j - \max\{r_j, s_i\} < p_i + s_i - \max\{r_j, s_i\} \Rightarrow$$

$$0 \leq \Delta_j < p_i - \max\{0, r_j - s_i\} = \min\{p_i, s_j - r_j\}$$

$$\Rightarrow \Omega_{ji} = d_j - p_j - \max\{r_j, s_i\}.$$

Subcaso A: $r_j \leq s_i$ - Ver Figura A.10.

$p_j + \max\{0, r_j - s_i\} = p_j$;

$\min\{p_i, s_j - r_j\} = p_i$;

De (3.29):

$$t_i.p_j - t_j.p_i + (d_j - p_j - s_i).(t_j + h_j) \geq 0 \Rightarrow$$

$$t_i.p_j - t_j.(p_i + p_j + s_i - d_j) + h_j.(d_j - p_j - s_i) \geq 0.$$

$$\text{custo}(ij) = t_i.(s_i + p_i - d_i) + t_j.(s_i + p_i + p_j - d_j)$$

$$\text{custo}(ji) = t_i.(s_i + p_j + p_i - d_i) + h_j.(d_j - s_i - p_j)$$

$$\text{custo}(ji) - \text{custo}(ij) =$$

$$t_i.p_j - t_j.(s_i + p_i + p_j - d_j) + h_j.(d_j - s_i - p_j) \geq 0 \Rightarrow$$

$\text{custo}(ji) \geq \text{custo}(ij)$.

Portanto, J_i precede J_j.

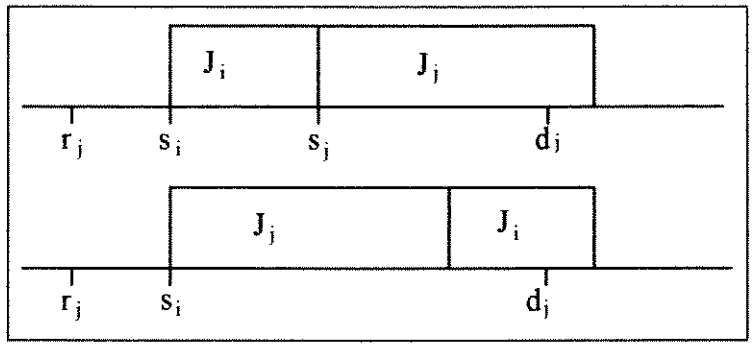

Figura A.10 - Troca entre J_i e J_j quando $a_i = 1$ e $a_j = 0$, $\max\{r_j, s_i\} + p_j \leq d_j < s_i + p_i + p_j$ e $r_j \leq s_i$.

Subcaso B: $r_j > s_i$ - Ver Figura A.11.

$p_j + \max\{0, r_j - s_i\} = p_j + r_j - s_i$;

$\min\{p_i, s_j - r_j\} = s_j - r_j$;

De (3.29):

$$t_i.(p_j + r_j - s_i) - t_j.(s_j - r_j) + (d_j - p_j - r_j).(h_j + t_j) =$$

$$t_i.(p_j + r_j - s_i) - t_j.(s_j - d_j + p_j) + h_j.(d_j - p_j - r_j) \geq 0$$

$$custo(ij) = t_i.(s_i + p_i - d_i) + t_j.(s_i + p_i + p_j - d_j)$$

$$custo(ji) = t_i.(r_j + p_j + p_i - d_i) + h_j.(d_j - r_j - p_j)$$

$$custo(ji) - custo(ij) =$$

$$t_i.(r_j + p_j - s_i) - t_j.(s_j - d_j + p_j) + h_j.(d_j - p_j - r_j) \geq 0 \Rightarrow$$

$$custo(ji) \geq custo(ij)$$

Portanto, J_i precede J_j.

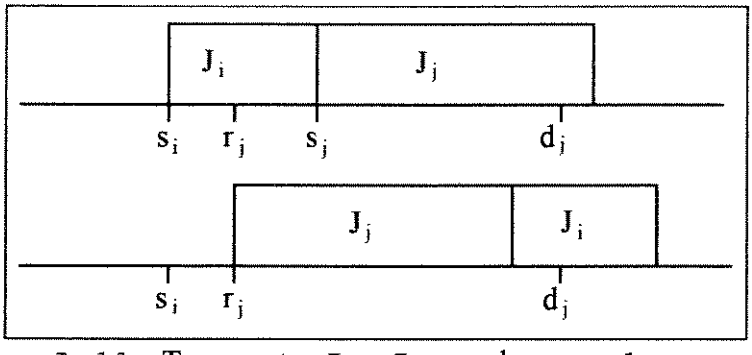

Figura A.11 - Troca entre J_i e J_j quando $a_i = 1$ e $a_j = 0$, $\max\{r_j, s_i\} + p_j \leq d_j < s_i + p_i + p_j$ e $r_j > s_i$.

Combinação 3:

$$d_j \geq s_i + p_j + p_i \Rightarrow$$

$$\Delta_j = d_j - p_j - \max\{r_j, s_i\} \geq s_i + p_i - \max\{r_j, s_i\}$$

$$\Rightarrow \Delta_j \geq p_i - \max\{0, r_j - s_i\} = \min\{p_i, s_j - r_j\}$$

$$\Rightarrow \Omega_{ji} = \min\{p_i, s_j - r_j\}.$$

Como a tarefa J_i está atrasada e a tarefa J_j está adiantada, a troca aumentaria o custo de atraso de J_i e o custo de avanço de J_j. Portanto, a expressão (3.29) é verdadeira.

Subcaso A: $r_j \leq s_i$ - Ver Figura A.12.

$p_j + \max\{0, r_j - s_i\} = p_j;$

$\min\{p_i, s_j - r_j\} = p_i;$

De (3.29):

$$t_i . p_j - t_j . p_i + p_i . (t_j + h_j) \geq 0 \Rightarrow$$

$$h_j . p_i + t_i . p_j \geq 0.$$

A desigualdade obtida é sempre verdadeira pois todos os parâmetros são positivos. Como verificação da variação de custo:

$custo(ij) = t_i . (s_i + p_i - d_i) + h_j . (d_j - s_i - p_i - p_j)$

$custo(ji) = t_i . (s_i + p_j + p_i - d_i) + h_j . (d_j - s_i - p_j)$

$custo(ij) - custo(ji) = t_i . p_j + h_j . p_i \geq 0$

Portanto, $custo(ji) \geq custo(ij)$ e J_i precede J_j.

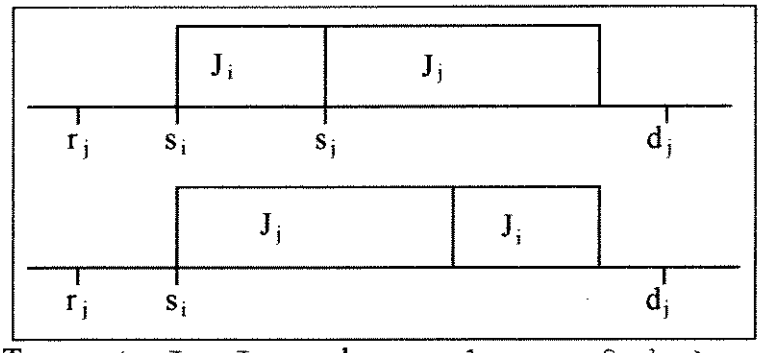

Figura A.12 - Troca entre J_i e J_j quando $a_i = 1$ e $a_j = 0$, $d_j \geq s_i + p_j + p_i$ e $r_j \leq s_i$.

Subcaso B: $r_j > s_i$ - Ver Figura A.13.

$p_j + \max\{0, r_j - s_i\} = p_j + r_j - s_i;$

$\min\{p_i, s_j - r_j\} = s_j - r_j;$

De (3.29):

$t_i.(p_j + r_j - s_i) - t_j.(s_i - r_j) + (s_j - r_j).(h_j + t_j) \geq 0 \Rightarrow$

$t_i.(p_j + r_j - s_i) + h_j.(s_j - r_j) \geq 0$

Novamente, chega-se a uma desigualdade que é sempre verdadeira.

Verificando-se a variação do custo:

$custo(ij) = t_i.(s_i + p_i - d_i) + h_j.(d_j - s_i - p_i - p_j)$

$custo(ji) = t_i.(r_j + p_j + p_i - d_i) + h_j.(d_j - r_j - p_j)$

$custo(ji) - custo(ij) = t_i.(p_j + r_j - s_i) + h_j.(s_i + p_i - r_j)$

$= t_i.(p_j + r_j - s_i) + h_j.(s_j - r_j) \geq 0 \Rightarrow$

$custo(ji) \geq custo(ij)$

Portanto, J_i precede J_j.

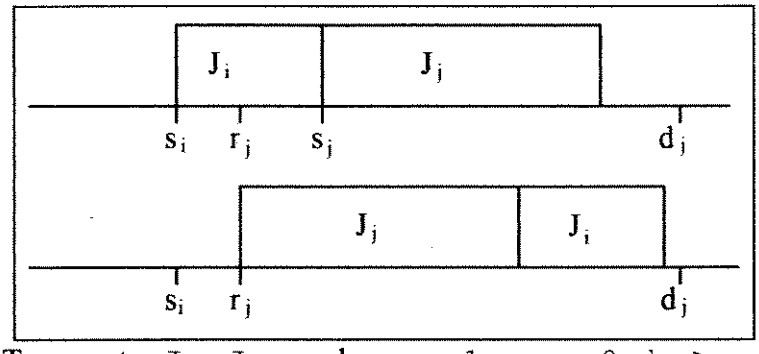

Figura A.13 - Troca entre J_i e J_j quando $a_i = 1$ e $a_j = 0$, $d_j \geq s_i + p_j + p_i$ e $r_j > s_i$.

CASO 4: $a_i = a_j = 0 \rightarrow$

As duas tarefas podem estar adiantadas ou atrasadas, antes ou depois da troca. De acordo com o que foi discutido nos casos 1 e 2, há dezesseis combinações de avanço e atraso para J_i e J_j. Entretanto, certas transições no estado das tarefas são impossíveis. Por exemplo, quando J_i está atrasada antes da troca, nunca haverá um avanço dessa tarefa, após a troca. Da mesma forma, se J_j está adiantada antes da troca, um deslocamento à esquerda somente aumentará seu avanço. Portanto, pode-se reduzir o conjunto de combinações àquelas possíveis de serem realizadas fisicamente:

1) J_i e J_j atrasadas antes e depois da troca:

$d_i < s_i + p_i$ e

$d_j < \max\{r_j, s_i\} + p_j$.

2) J_i atrasada antes e depois da troca, J_j atrasada antes e adiantada depois:

$d_i < s_i + p_i$ e

$\max\{r_j, s_i\} + p_j \leq d_j < s_i + p_i + p_j$.

3) J_i adiantada antes e atrasada depois da troca, J_j atrasada antes e depois:

$s_i + p_i \leq d_i < \max\{r_j, s_i\} + p_j + p_i$ e

$d_j < \max\{r_j, s_i\} + p_j$.

4) J_i adiantada antes e depois da troca, J_j atrasada antes e depois:

$d_i \geq \max\{r_j, s_i\} + p_j + p_i$ e

$d_j < \max\{r_j, s_i\} + p_j$.

5) J_i adiantada antes e atrasada depois da troca, J_j atrasada antes e adiantada depois:

$s_i + p_i \leq d_i < \max\{r_j, s_i\} + p_j + p_i$ e

$\max\{r_j, s_i\} + p_j \leq d_j < s_i + p_i + p_j$.

6) J_i adiantada antes e depois da troca, J_j atrasada antes e adiantada depois:

$d_i \geq \max\{r_j, s_i\} + p_j + p_i$ e

$\max\{r_j, s_i\} + p_j \leq d_j < s_i + p_i + p_j$.

7) J_i atrasada antes e depois da troca, J_j adiantada antes e depois:

$d_i < s_i + p_i$ e

$d_j \geq s_i + p_i + p_j$.

8) J_i adiantada antes e atrasada depois da troca, J_j adiantada antes e depois:

$s_i + p_i \leq d_i < \max\{r_j, s_i\} + p_j + p_i$ e

$d_j \geq s_i + p_i + p_j$.

9) J_i e J_j adiantadas antes e depois da troca:

$d_i \geq \max\{r_j, s_i\} + p_j + p_i$ e

$d_j \geq s_i + p_i + p_j$.

Combinação 1:

$d_i < s_i + p_i \Rightarrow \Delta_i < 0 \Rightarrow \Omega_{ij} = 0$;

$d_j < \max\{r_j, s_i\} + p_j \Rightarrow \Delta_j < 0 \Rightarrow \Omega_{ji} = 0$.

Subcaso A: $r_j \leq s_i$ - Ver Figura A.14.

$p_j + \max\{0, r_j - s_i\} = p_j$;

$\min\{p_i, s_j - r_j\} = p_i$;

De (3.29):

$t_i \cdot p_j - t_j \cdot p_i \geq 0$

$custo(ij) = t_i.(s_i + p_i - d_j) + t_j.(s_i + p_i + p_j - d_j)$

$custo(ji) = t_i.(s_i + p_j + p_i - d_i) + t_j.(s_i + p_j - d_j)$

$custo(ji) - custo(ij) = t_i.p_j - t_j.p_i \geq 0 \Rightarrow$

$custo(ji) \geq custo(ij)$

Portanto, J_i precede J_j.

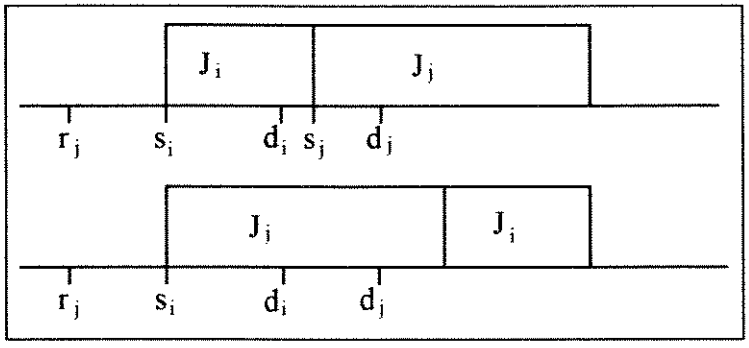

Figura A.14 - Troca entre J_i e J_j quando $a_i = 0$ e $a_j = 0$, $d_i < s_i + p_i$, $d_j < \max\{r_j, s_i\} + p_j$ e $r_j \leq s_i$.

Subcaso B: $r_j > s_i$.

Esse caso nunca ocorre na prática pois

$a_j = 0 \rightarrow d_j \geq r_j + p_j$.

Portanto, quando $r_j > s_i$, J_j nunca poderá estar atrasada após a troca.

Combinação 2:

$d_i < s_i + p_i \Rightarrow \Delta_i < 0 \Rightarrow \Omega_{ji} = 0;$

$\max\{r_j, s_i\} + p_j \leq d_j < s_i + p_i + p_j \Rightarrow$

$0 \leq d_j - p_j - \max\{r_j, s_i\} < s_i + p_i - \max\{r_j, s_i\} \Rightarrow$

$0 \leq \Delta_j < p_i - \max\{0, r_j - s_i\} = \min\{p_i, s_j - r_j\} \Rightarrow$

$\Omega_{ji} = d_j - p_j - \max\{r_j, s_i\}$.

Subcaso A: $r_j \leq s_i$ - **Ver Figura A.15.**

$p_j + \max\{0, r_j - s_i\} = p_j;$

$\min\{p_i, s_j - r_j\} = p_i;$

De (3.29):

$t_i.p_j - t_j.p_i + (d_j - p_j - s_i).(h_j + t_j) \geq 0 \Rightarrow$

$t_i.p_j + h_j.(d_j - p_j - s_i) - t_j.(s_i + p_i + p_j - d_j) \geq 0$

$custo(ij) = t_i.(s_i + p_i - d_i) + t_j.(s_i + p_i + p_j - d_j)$

$custo(ji) = t_j.(s_j + p_j + p_i - d_i) + h_j.(d_j - s_i - p_j)$

$custo(ji) - custo(ij) =$

$t_i.p_j + h_j.(d_j - p_j - s_i) - t_j.(s_i + p_i + p_j - d_j) =$

$t_i.p_j + h_j.(d_j - p_j - s_i) - t_j.(s_j + p_j - d_j) \geq 0 \Rightarrow$

$custo(ji) \geq custo(ij)$

Portanto, J_i precede J_j.

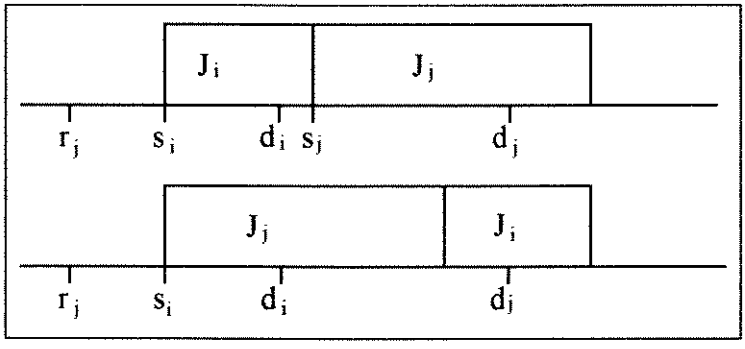

Figura A.15 - Troca entre J_i e J_j quando $a_i = 0$ e $a_j = 0$, $d_i < s_i + p_i$, $\max\{r_j, s_i\} + p_j \leq d_j < s_i + p_i + p_j$ e $r_j \leq s_i$.

Subcaso B: $r_j > s_i$ Ver Figura A.16.

$p_j + \max\{0, r_j - s_i\} = p_j + r_j - s_i;$

$\min\{p_i, s_j - r_j\} = s_j - r_j;$

De (3.29):

$t_i.(p_j + r_j - s_i) - t_j.(s_j - r_j) + (d_j - p_j - r_j).(h_j + t_j) \geq 0 \Rightarrow$

$t_i.(p_j + r_j - s_i) + h_j.(d_j - p_j - s_i) - t_j.(s_i + p_i + p_j - d_j) \geq 0$

$custo(ij) = t_i.(s_i + p_i - d_i) + t_j.(s_i + p_i + p_j - d_j)$

$custo(ji) = t_j.(r_j + p_j + p_i - d_i) + h_j.(d_j - r_j - p_j)$

$custo(ji) - custo(ij) =$

$t_i.(p_j + r_j - s_i) + h_j.(d_j - p_j - r_j) - t_j.(s_i + p_i + p_j - d_j) \geq 0 \Rightarrow$

$custo(ji) \geq custo(ij)$

Portanto, J_i precede J_j.

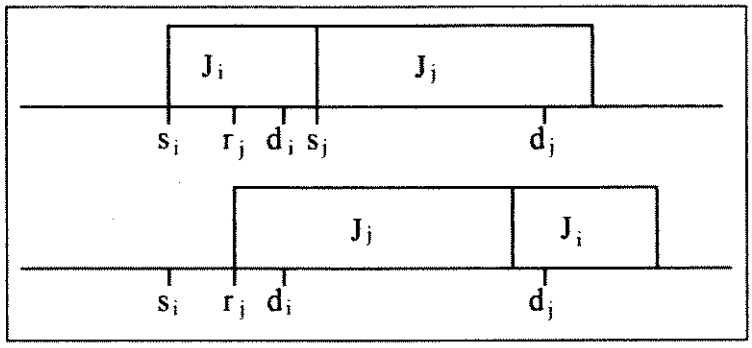

Figura A.16 - Troca entre J_i e J_j quando $a_i = 0$ e $a_j = 0$, $d_i < s_i + p_i$, $\max\{r_j, s_i\} + p_j \leq d_j < s_i + p_i + p_j$ e $r_j > s_i$.

Combinação 3:

$$s_i + p_i \leq d_i < \max\{r_j, s_i\} + p_j + p_i \Rightarrow$$
$$0 \leq d_i - s_i - p_i < p_j + \max\{0, r_j - s_i\} \Rightarrow$$
$$0 \leq \Delta_i < p_j + \max\{0, r_j - s_i\} \Rightarrow$$
$$\Omega_{ij} = d_i - s_i - p_i;$$

$$d_j < \max\{r_j, s_i\} + p_j \Rightarrow \Delta_j < 0 \Rightarrow \Omega_{ij} = 0.$$

Subcaso A: $r_j \leq s_i$ - Ver Figura A.17.

$p_j + \max\{0, r_j - s_i\} = p_j;$

$\min\{p_i, s_j - r_j\} = p_i;$

De (3.29):

$$t_i.p_j - (d_i - p_i - s_i).(h_i + t_i) - t_j.p_i \geq 0$$
$$\Rightarrow t_i.(s_i + p_i + p_j - d_i) - h_i.(d_i - p_i - s_i) - t_j.p_i \geq 0$$

$custo(ij) = h_i.(d_i - s_i - p_i) + t_j.(s_i + p_i + p_j - d_j)$

$custo(ji) = t_i.(s_i + p_j + p_i - d_i) + t_j.(s_i + p_j - d_j)$

$custo(ji) - custo(ij) =$

$t_i.(s_i + p_i + p_j - d_j) - h_i.(d_i - p_i - s_i) - t_j.p_i \geq 0 \Rightarrow$

$custo(ji) \geq custo(ij)$

Portanto, J_i precede J_j.

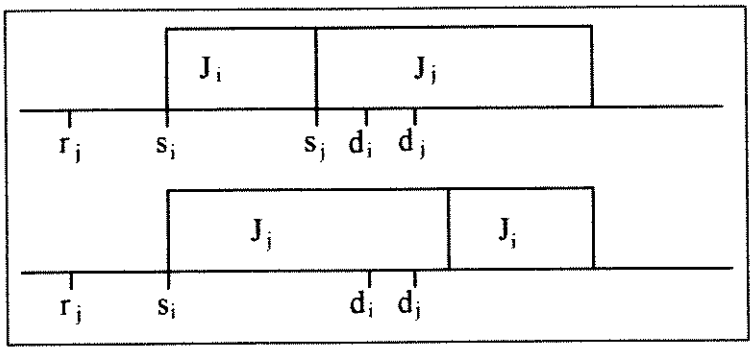

Figura A.17 - Troca entre J_i e J_j quando $a_i = 0$ e $a_j = 0$, $s_i + p_i \leq d_i < \max\{r_j, s_i\} + p_j + p_i, d_j < \max\{r_j, s_i\} + p_j$ e $r_j \leq s_i$.

Subcaso B: $r_j > s_i$ Ver Figura A.18.

$p_j + \max\{0, r_j - s_i\} = p_j + r_j - s_i$;

$\min\{p_i, s_j - r_j\} = s_j - r_j$;

De (3.29):

$$t_i.(p_j + r_j - s_i) - (d_i - p_i - s_i).(h_i + t_i) - t_j.(s_j - r_j) \geq 0 \Rightarrow$$

$$t_i.(p_j + p_i + r_j - d_i) - h_i.(d_i - p_i - s_i) - t_j.(s_j - r_j) \geq 0$$

$custo(ij) = h_i.(d_i - s_i - p_i) + t_j.(s_i + p_i + p_j - d_j)$

$custo(ji) = t_i.(r_j + p_j + p_i - d_i) + t_j.(r_j + p_j - d_j)$

$custo(ji) - custo(ij) =$

$$t_i.(r_j + p_j + p_i - d_i) - h_i.(d_i - p_j - s_i) - t_j.(s_i + p_i - r_j) =$$

$$t_i.(r_j + p_j + p_i - d_i) - h_i.(d_i - p_i - s_i) - t_j.(s_i - r_j) \geq 0 \Rightarrow$$

$custo(ji) \geq custo(ij)$

Portanto, J_i precede J_j.

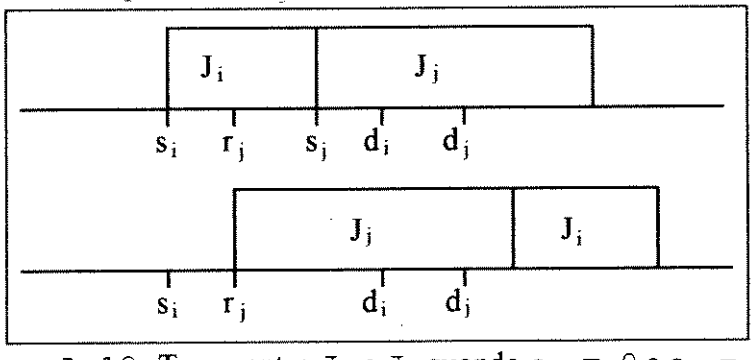

Figura A.18 -Troca entre J_i e J_j quando $a_i = 0$ e $a_j = 0$, $s_i + p_i \leq d_i < \max\{r_j, s_i\} + p_j + p_i, d_j < \max\{r_j, s_i\} + p_j$ e $r_j > s_i$.

Combinação 4:

$$d_i \geq \max\{r_j, s_i\} + p_j + p_i \Rightarrow$$
$$d_i - s_i - p_i \geq p_j + \max\{0, r_j - s_i\} \Rightarrow$$
$$\Omega_{ij} = p_j + \max\{0, r_j - s_i\};$$

$$d_j < \max\{r_j, s_i\} + p_j \Rightarrow \Delta_j < 0 \Rightarrow \Omega_{ji} = 0.$$

Como a tarefa J_i está adiantada e a tarefa J_j está atrasada, seu posicionamento não é ótimo e a troca deve ser feita. A troca causará uma diminuição do custo de avanço de J_i e do custo de atraso de J_j. Portanto, a expressão (3.29) não é verdadeira nesse caso.

Subcaso A: $r_j \leq s_i$ - Ver Figura A.19.

$p_j + \max\{0, r_j - s_i\} = p_j;$

$\min\{p_i, s_j - r_j\} = p_i;$

De (3.29):

$$t_i.p_j - p_j.(h_i + t_i) - t_j.p_i \geq 0 \Rightarrow$$
$$- h_i.p_j - t_j.p_i \geq 0$$

A desigualdade obtida nunca é verdadeira pois todos os parâmetros são positivos. Verificando-se a variação de custo:

$custo(ij) = h_i.(d_i - s_i - p_i) + t_j.(s_i + p_i + p_j - d_j)$

$custo(ji) = h_j.(d_i - s_i - p_j - p_i) + t_j.(s_i + p_j - d_j)$

$custo(ji) - custo(ij) = -h_i.p_j - t_j.p_i \leq 0$

Portanto, J_j precede J_i.

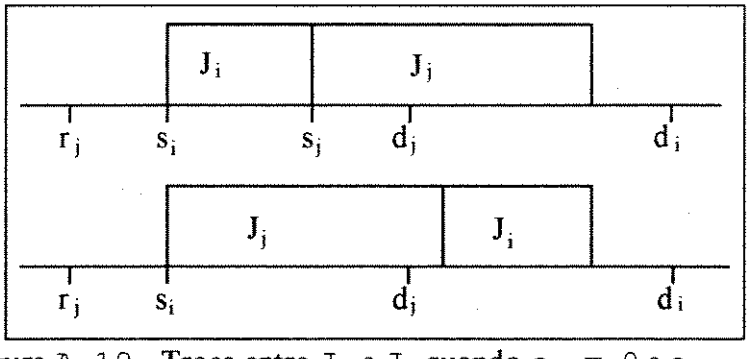

Figura A.19 - Troca entre J_i e J_j quando $a_i = 0$ e $a_j = 0$, $d_i \geq \max\{r_j, s_i\} + p_j + p_i$, $d_j < \max\{r_j, s_i\} + p_j$ e $r_j \leq s_i$.

Subcaso B: $r_j > s_i$ Ver Figura A.20.

$p_j + \max\{0, r_j - s_i\} = p_j + r_j - s_i;$

$\min\{p_i, s_j - r_j\} = s_j - r_j$;

De (3.29):

$$t_i.(p_j + r_j - s_i) - (p_i + r_j - s_i).(h_i + t_i) - t_j.(s_j - r_j) \geq 0 \Rightarrow$$

$$-h_i.(p_j + r_j - s_i) - t_j.(s_j - r_j) \geq 0$$

Novamente, chega-se a uma desigualdade falsa, confirmando que, para esta combinação de atraso e avanço, J_j precede J_i. A variação no custo é dada por:

$$custo(ij) = h_i.(d_i - s_i - p_i) + t_j.(s_i + p_i + p_j - d_j)$$

$$custo(ji) = h_i.(d_i - r_j - p_j - p_i) + t_j.(r_j + p_j - d_j)$$

$$custo(ji) - custo(ij) = -h_i.(p_j + r_j - s_i) - t_j.(s_i + p_i - r_j)$$

$$= -h_i.(p_j + r_j - s_i) - t_j.(s_j - r_j) \leq 0 \Rightarrow$$

$$custo(ji) \leq custo(ij)$$

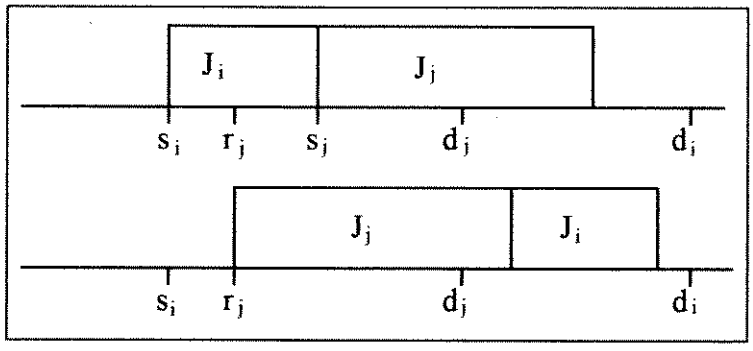

Figura A.20 -Troca entre J_i e J_j quando $a_i = 0$ e $a_j = 0$, $d_i \geq \max\{r_j, s_i\} + p_j + p_i$, $d_j < \max\{r_j, s_i\} + p_j$ e $r_j > s_i$.

Combinação 5:

$$s_i + p_i \leq d_i < \max\{r_j, s_i\} + p_j + p_i \Rightarrow$$

$$0 \leq \Delta_i < p_j + \max\{0, r_j - s_i\} \Rightarrow$$

$$\Omega_{ij} = d_i - s_i - p_i;$$

$$\max\{r_j, s_i\} + p_j \leq d_j < s_i + p_i + p_j \Rightarrow$$

$$0 \leq \Delta_j < s_i + p_i - \max\{r_j, s_i\} \Rightarrow$$

$$0 \leq \Delta_j < p_i - \max\{0, r_j - s_i\} = \min\{p_i, s_j - r_j\}$$

$$\Rightarrow \Omega_{ji} = d_j - p_j - \max\{r_j, s_i\}.$$

Subcaso A: $r_j \leq s_i$ - Ver Figura A.21.

$p_j + \max\{0, r_j - s_i\} = p_j$;

$\min\{p_i, s_j - r_j\} = p_i$;

De (3.29):

$$t_i.p_j - (d_i - p_i - s_i).(h_i + t_i) - t_j.p_i + (d_j - p_j - s_i).(h_j + t_j) \geq 0 \Rightarrow$$

$$t_i.(s_i + p_i + p_j - d_i) - h_i.(d_i - p_i - s_i) - t_j.(p_i + p_j + s_i - d_j) + h_j.(d_j - p_j - s_i) \geq 0$$

$custo(ij) = h_i.(d_i - s_i - p_i) + t_j.(s_i + p_i + p_j - d_j)$

$custo(ji) = t_i.(s_i + p_j + p_i - d_i) + h_j.(d_j - s_i - p_j)$

$custo(ji) - custo(ij) =$

$$t_i.(s_i + p_i + p_j - d_j) - h_i.(d_i - p_i - s_i) - t_j.(p_i + p_j + s_i - d_j) + h_j.(d_j - s_i - p_j) \geq 0 \Rightarrow$$

$custo(ji) \geq custo(ij)$

Portanto, J_i precede J_j.

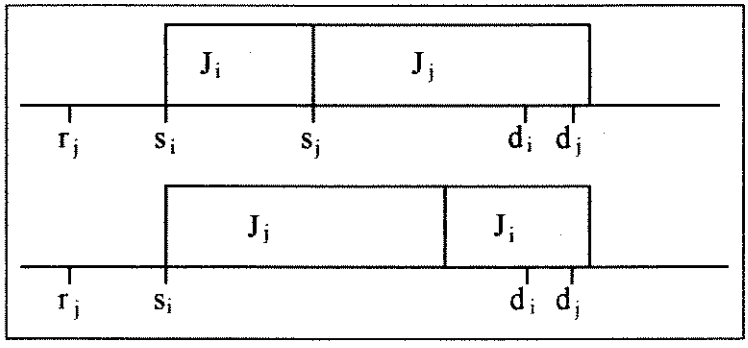

Figura A.21 - Troca entre J_i e J_j quando $a_i = 0$ e $a_j = 0$, $s_i + p_i \leq d_i < \max\{r_j, s_i\} + p_j + p_i, d_j < \max\{r_j, s_i\} + p_j$ e $r_j \leq s_i$.

Subcaso B: $r_j > s_i$ Ver Figura A.22.

$p_j + \max\{0, r_j - s_i\} = p_j + r_j - s_i$;

$\min\{p_i, s_j - r_j\} = s_j - r_j$;

De (3.29):

$$t_i.(p_j + r_j - s_i) - (d_i - p_i - s_i).(h_i + t_i) - t_j.(s_j - r_j) + (d_j - p_j - r_j).(h_j + t_j) \geq 0 \Rightarrow$$

$$t_i.(p_j + p_i + r_j - d_i) - h_i.(d_i - p_i - s_i) - t_j.(s_j + p_j - d_j) + h_j.(d_j - p_j - r_j) \geq 0$$

$custo(ij) = h_i.(d_i - s_i - p_i) + t_j.(s_i + p_i + p_j - d_j)$

$custo(ji) = t_i.(r_j + p_j + p_i - d_i) + h_j.(d_j - r_j - p_j)$

$$custo(ji) - custo(ij) =$$
$$t_i.(r_j + p_j + p_i - d_i) - h_i.(d_i - p_i - s_i)$$
$$- t_j.(s_i + p_i + p_j - d_j) + h_j.(d_j - r_j - p_j) =$$
$$t_i.(r_j + p_j + p_i - d_i) - h_i.(d_i - p_i - s_i)$$
$$- t_j.(s_j + p_j - d_j) + h_j.(d_j - r_j - p_j) \geq 0 \Rightarrow$$
$$custo(ji) \geq custo(ij)$$

Portanto, J_i **precede** J_j.

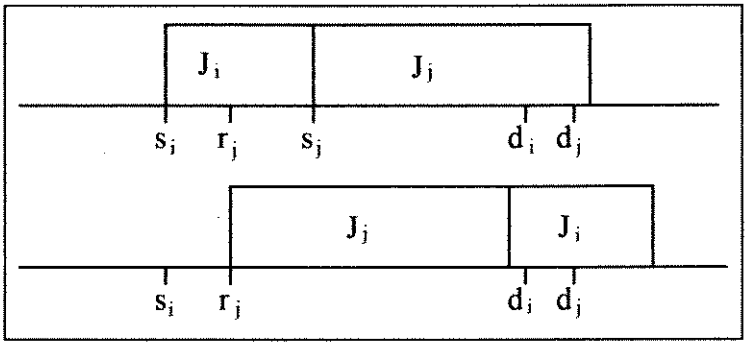

Figura A.22 -Troca entre J_i e J_j quando $a_i = 0$ e $a_j = 0$, $s_i + p_i \leq d_i < \max\{r_j, s_i\} + p_j + p_i, d_j < \max\{r_j, s_i\} + p_j$ e $r_j > s_i$.

Combinação 6:

$$d_i \geq \max\{r_j, s_i\} + p_j + p_i + \Rightarrow$$
$$\Delta_i = d_i - p_i - s_i \geq p_j + \max\{0, r_j - s_i\} \Rightarrow$$
$$\Omega_{ij} = p_j + \max\{0, r_j - s_i\};$$

$$\max\{r_j, s_i\} + p_j \leq d_j < s_i + p_i + p_j \Rightarrow$$
$$0 \leq \Delta_j < s_i + p_i - \max\{r_j, s_i\} \Rightarrow$$
$$0 \leq \Delta_j < p_i - \max\{0, r_j - s_i\} = \min\{p_i, s_j - r_j\}$$
$$\Rightarrow \Omega_{ji} = d_j - p_j - \max\{r_j, s_i\}.$$

Subcaso A: $r_j \leq s_i$ - Ver Figura A.23.

$$p_j + \max\{0, r_j - s_i\} = p_j;$$
$$\min\{p_i, s_j - r_j\} = p_i;$$

De (3.29):

$$t_i . p_j - p_j . (h_i + t_i) - t_j . p_i + (d_j - p_j - s_i) . (h_i + t_j) \geq 0 \Rightarrow$$

$$-h_i . p_j - t_j . (s_i + p_i + p_j - d_j) + h_j . (d_j - p_j - s_i) \geq 0$$

$$custo(ij) = h_i . (d_i - s_i - p_i) + t_j . (s_i + p_i + p_j - d_j)$$

$$custo(ji) = h_i . (d_i - s_i - p_j - p_i) + h_j . (d_j - s_i - p_j)$$

$$custo(ji) - custo(ij) =$$

$$-h_i . p_j - t_j . (s_i + p_i + p_j - d_j) + h_j . (d_j - s_i - p_j) \geq 0 \Rightarrow$$

$$custo(ji) \geq custo(ij)$$

Portanto, J_i precede J_j.

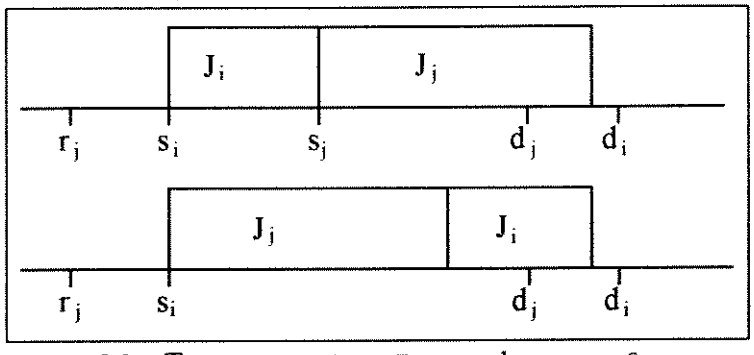

Figura A.23 - Troca entre J_i e J_j quando $a_i = 0$ e $a_j = 0$, $d_i \geq \max\{r_j, s_i\} + p_j + p_i$, $\max\{r_j, s_i\} + p_j \leq d_j < s_i + p_i + p_j$ e $r_j \leq s_i$.

<u>Subcaso B</u>: $r_j > s_i$ Ver Figura A.24.

$p_j + \max\{0, r_j - s_i\} = p_j + r_j - s_i$;

$\min\{p_i, s_j - r_j\} = s_j - r_j$;

De (3.29):

$$t_i . (p_j + r_j - s_i) - (p_j + r_j - s_i) . (h_i + t_i) - t_j . (s_j - r_j) + (d_j - p_j - r_j) . (h_j + t_j) \geq 0 \Rightarrow$$

$$-h_i . (p_j + r_j - s_i) - t_j . (s_j + p_j - d_j) + h_j . (d_j - p_j - r_j) \geq 0$$

$$custo(ij) = h_i . (d_i - s_i - p_i) + t_j . (s_i + p_i + p_j - d_j)$$

$$custo(ji) = h_i . (d_i - r_j - p_j - p_i) + h_j . (d_j - r_j - p_j)$$

$$custo(ji) - custo(ij) =$$
$$-h_i.(p_j + r_j - s_i) - t_j.(s_i + p_i + p_j - d_j) + h_j.(d_j - r_j - p_j) =$$
$$-h_i.(p_j + r_j - s_i) - t_j.(s_i + p_j - d_j) + h_j.(d_i - r_j - p_j) \geq 0 \Rightarrow$$
$$custo(ji) \geq custo(ij)$$

Portanto, J_i **precede** J_j.

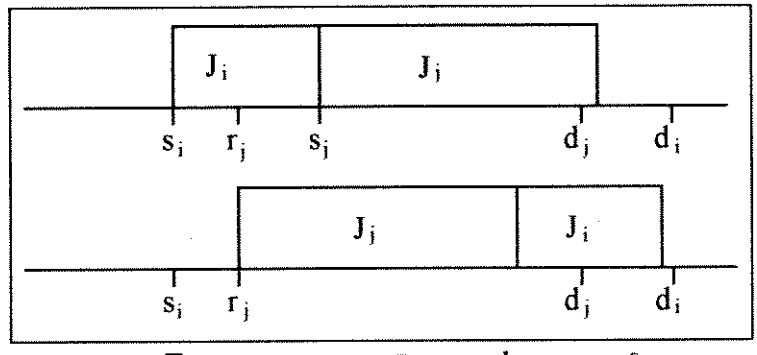

Figura A.24 -Troca entre J_i **e** J_j **quando** $a_i = 0$ **e** $a_j = 0$, $d_i \geq \max\{r_j, s_i\} + p_j + p_i, \max\{r_j, s_i\} + p_j \leq d_j < s_i + p_i + p_j$ **e** $r_j > s_i$.

Combinação 7:

$$d_i < s_i + p_i \Rightarrow \Delta_i < 0 \Rightarrow \Omega_{ij} = 0;$$

$$d_j \geq s_i + p_i + p_j \Rightarrow \Delta_j \geq s_i + p_i - \max\{r_j, s_i\}$$
$$\Rightarrow \Delta_j \geq p_i - \max\{0, r_j - s_i\} = \min\{p_i, s_j - r_j\}$$
$$\Rightarrow \Omega_{ji} = \min\{p_i, s_j - r_j\}.$$

Subcaso A: $r_j \leq s_i$ - **Ver Figura** A.25.

$p_j + \max\{0, r_j - s_i\} = p_j$;

$\min\{p_i, s_j - r_j\} = p_i$;

De (3.29):

$$t_i.p_j - t_j.p_i + p_i.(h_j + t_j) \geq 0 \Rightarrow$$
$$t_i.p_j + h_j.p_i \geq 0$$

$$custo(ij) = t_i.(s_i + p_i - d_i) + h_j.(d_j - s_i - p_i - p_j)$$

$$custo(ji) = t_j.(s_i + p_j + p_i - d_i) + h_j.(d_j - s_i - p_j)$$

$custo(ji) - custo(ij) = t_i . p_j + h_j . p_i \geq 0 \Rightarrow$

$custo(ji) \geq custo(ij)$

Portanto, J_i precede J_j.

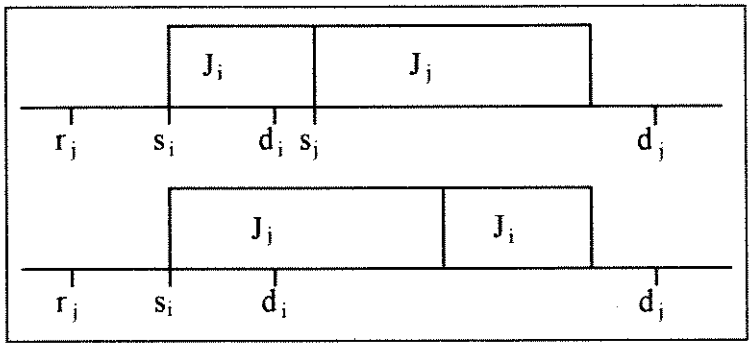

Figura A.25 - Troca entre J_i e J_j quando $a_i = 0$ e $a_j = 0$, $d_i < s_i + p_i$, $d_j \geq s_i + p_i + p_j$ e $r_j \leq s_i$.

Subcaso B: $r_j > s_i$ - Ver Figura A.26.

$p_j + \max\{0, r_j - s_i\} = p_j + r_j - s_i$;

$\min\{p_i, s_j - r_j\} = s_j - r_j$;

De (3.29):

$$t_i . (p_j + r_j - s_i) - t_j . (s_j - r_j) + (s_j - r_j) . (h_j + t_i) \geq 0 \Rightarrow$$

$$t_i . (p_j + r_j - s_i) + h_j . (s_j - r_j) \geq 0$$

$custo(ij) = t_i . (s_j + p_j - d_i) + h_j . (d_j - s_j - p_i - p_j)$

$custo(ji) = t_j . (r_j + p_j + p_i - d_j) + h_j . (d_j - r_j - p_j)$

$custo(ji) - custo(ij) = t_i . (p_j + r_j - s_i) + h_j . (s_i + p_i - r_j)$

$= t_i . (p_j + r_j - s_i) + h_i . (s_i - r_j) \geq 0 \Rightarrow$

$custo(ji) \geq custo(ij)$

Portanto, J_i precede J_j.

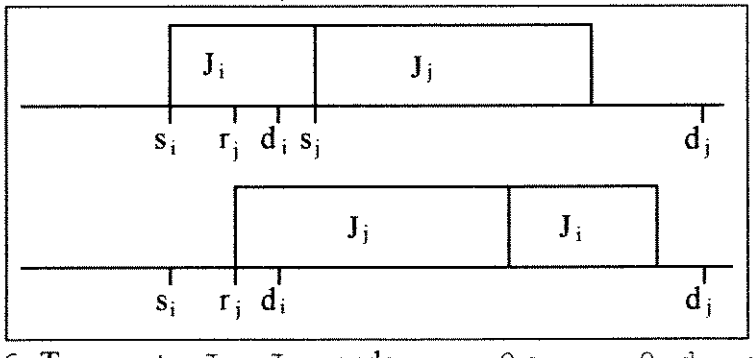

Figura A.26 -Troca entre J_i e J_j quando $a_i = 0$ e $a_j = 0$, $d_i < s_i + p_i$, $d_j \geq s_i + p_i + p_j$ e $r_j > s_i$.

Combinação 8:

$$s_i + p_i \leq d_i < \max\{r_j, s_i\} + p_j + p_i \Rightarrow$$
$$0 \leq \Delta_i < p_j + \max\{0, r_j - s_i\} \Rightarrow$$
$$\Omega_{ij} = d_i - s_i - p_i;$$

$$d_j \geq s_i + p_i + p_j \Rightarrow$$
$$\Delta_j \geq s_i + p_i - \max\{r_j, s_i\} \Rightarrow$$
$$\Delta_j \geq p_i - \max\{0, r_j - s_i\} = \min\{p_i, s_j - r_j\}$$
$$\Rightarrow \Omega_{ji} = \min\{p_i, s_j - r_j\}.$$

Subcaso A: $r_j \leq s_i$ - Ver Figura A.27.

$p_j + \max\{0, r_j - s_i\} = p_j;$

$\min\{p_i, s_j - r_j\} = p_i;$

De (3.29):

$$t_i.p_j - (d_i - s_i - p_i).(h_i + t_i) - t_j.p_i + p_i.(h_j + t_j) \geq 0 \Rightarrow$$

$$t_i.(s_i + p_j + p_i - d_i) - h_i.(d_i - s_i - p_i) + h_j.p_i \geq 0$$

$custo(ij) = h_i.(d_i - s_i - p_i) + h_j.(d_j - s_i - p_i - p_j)$

$custo(ji) = t_i.(s_i + p_j + p_i - d_i) + h_j.(d_j - s_i - p_j)$

$custo(ji) - custo(ij) =$

$$t_i.(s_i + p_i + p_j - d_i) - h_i.(d_i - s_i - p_i) + h_i.p_i \Rightarrow$$

$custo(ji) \geq custo(ij)$

Portanto, J_i precede J_j.

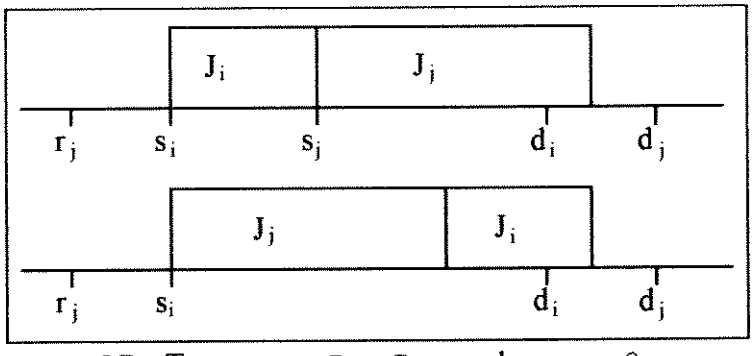

Figura A.27 - Troca entre J_i e J_j quando $a_i = 0$ e $a_j = 0$, $s_i + p_i \leq d_i < \max\{r_j, s_i\} + p_j + p_i, d_j \geq s_i + p_i + p_j$ e $r_j \leq s_i$.

Subcaso B: $r_j > s_i$ Ver Figura A.28.

$p_j + \max\{0, r_j - s_i\} = p_j + r_j - s_i;$

$\min\{p_i, s_j - r_j\} = s_j - r_j;$

De (3.29):

$$t_i.(p_j + r_j - s_i) - (d_i - s_i - p_i).(h_i + t_i) - t_j.(s_j - r_j) + (s_j - r_j).(h_j + t_j) \geq 0 \Rightarrow$$

$$t_i.(r_j + p_j + p_i - d_i) - h_i.(d_i - s_i - p_i) + h_j.(s_j - r_j) \geq 0$$

$$custo(ij) = h_i.(d_i - s_i - p_i) + h_j.(d_j - s_i - p_i - p_j)$$

$$custo(ji) = t_i.(r_j + p_j + p_i - d_i) + h_j.(d_j - r_j - p_j)$$

$$custo(ji) - custo(ij) =$$

$$t_i.(r_j + p_j + p_i - d_i) - h_i.(d_i - s_i - p_i) + h_j.(s_i + p_i - r_j) =$$

$$t_i.(r_j + p_j + p_i - d_i) - h_i.(d_i - s_i - p_i) + h_j.(s_j - r_j) \geq 0 \Rightarrow$$

$$custo(ji) \geq custo(ij)$$

Portanto, J_i precede J_j.

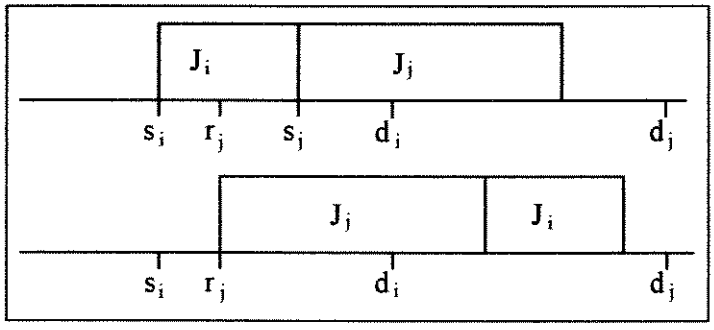

Figura A.28 -Troca entre J_i e J_j quando $a_i = 0$ e $a_j = 0$, $s_i + p_i \leq d_i < \max\{r_j, s_i\} + p_j + p_i, d_j \geq s_i + p_i + p_j$ e $r_j > s_i$.

Combinação 9:

$$d_i \geq \max\{r_j, s_i\} + p_j + p_j \Rightarrow$$

$$\Delta_i \geq p_j + \max\{0, r_j - s_i\} \Rightarrow$$

$$\Omega_{ij} = p_j + \max\{0, r_j - s_i\};$$

$$d_j \geq s_i + p_i + p_j \Rightarrow$$
$$\Delta_j \geq s_i + p_i - \max\{r_j, s_i\} \Rightarrow$$
$$\Delta_j \geq p_i - \max\{0, r_j - s_i\} = \min\{p_i, s_j - r_j\}$$
$$\Rightarrow \Omega_{ji} = \min\{p_i, s_j - r_j\}.$$

Subcaso A: $r_j \leq s_i$ - Ver Figura A.29.

$p_j + \max\{0, r_j - s_i\} = p_j$;

$\min\{p_i, s_j - r_j\} = p_i$;

De (3.29):

$$t_i.p_j - p_j.(h_i + t_i) - t_j.p_i + p_i.(h_j + t_j) \geq 0 \Rightarrow$$
$$- h_i.p_j + h_j.p_i \geq 0$$

$custo(ij) = h_i.(d_i - s_i - p_i) + h_j.(d_j - s_i - p_i - p_j)$

$custo(ji) = h_i.(d_i - s_i - p_j - p_i) + h_j.(d_j - s_i - p_j)$

$custo(ji) - custo(ij) = -h_i.p_j + h_j.p_i \geq 0 \Rightarrow$

$custo(ji) \geq custo(ij)$

Portanto, J_i precede J_j.

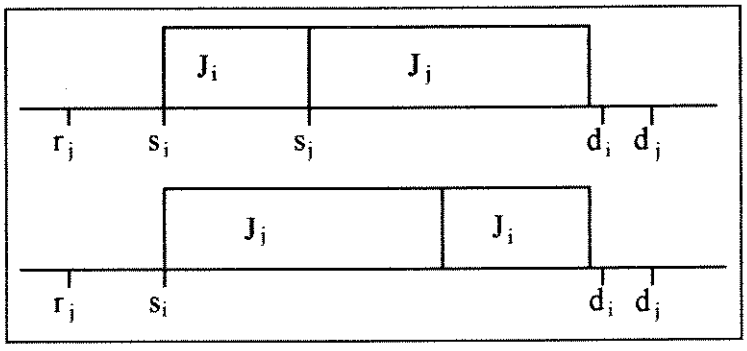

Figura A.29 - Troca entre J_i e J_j quando $a_i = 0$ e $a_j = 0$, $d_i \geq \max\{r_j, s_i\} + p_j + p_i$, $d_j \geq s_i + p_i + p_j$ e $r_j \leq s_i$.

Subcaso B: $r_j > s_i$ Ver Figura A.30.

$p_j + \max\{0, r_j - s_i\} = p_j + r_j - s_i$;

$\min\{p_i, s_j - r_j\} = s_j - r_j$;

De (3.29):

$$t_i.(p_j + r_j - s_i) - (p_j + r_j - s_i).(h_i + t_i)$$
$$- t_j.(s_j - r_j) + (s_j - r_j).(h_j + t_j) \geq 0 \Rightarrow$$
$$- h_i.(p_j + r_j - s_i) + h_j.(s_j - r_j) \geq 0$$

$custo(ij) = h_i.(d_i - s_i - p_i) + h_j.(d_j - s_i - p_i - p_j)$

$custo(ji) = h_i.(d_i - r_j - p_j - p_i) + h_j.(d_j - r_j - p_j)$

$$custo(ji) - custo(ij) =$$
$$- h_i.(p_j + r_j - s_i) + h_j.(s_i + p_i - r_j) =$$
$$- h_i.(p_j + r_j - s_i) + h_j.(s_j - r_j) \geq 0 \Rightarrow$$
$$custo(ji) \geq custo(ij)$$

Portanto, J_i precede J_j.

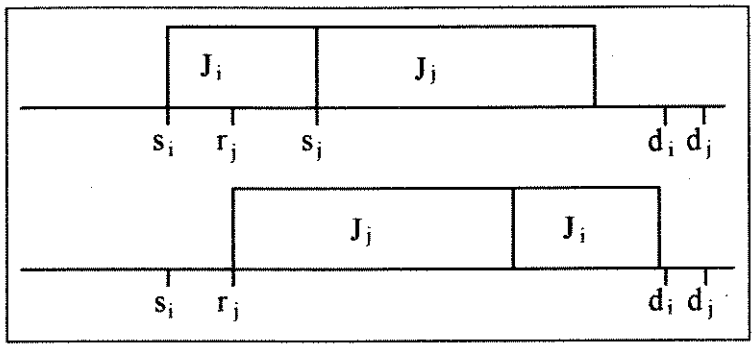

Figura A.30 -Troca entre J_i e J_j quando $a_i = 0$ e $a_j = 0$, $d_i \geq \max\{r_j, s_i\} + p_j + p_i$, $d_j \geq s_i + p_i + p_j$ e $r_j > s_i$.

APÊNDICE B

ANÁLISE DE COMPLEXIDADE

Neste apêndice é apresentada a análise de complexidade de pior caso para a heurística proposta.

A análise de pior caso é feita para cada um dos procedimentos da fase construtiva e para a Busca em Vizinhança. Em seguida, calcula-se a complexidade total da heurística.

B.1 - PROCEDIMENTO DE ORDENAÇÃO

Nesse procedimento são calculados os valores de u_i, v_i, w_i e a_i, para $\forall J_i$, $i \in \mathbf{N}$. A complexidade desses cálculos é $O(n)$.

A ordenação realizada é do tipo "quick sort" e possui uma complexidade igual a $O(n.\log(n))$.

Portanto, a complexidade total do Procedimento de Ordenação é $O(n.\log(n))$.

B.2 - PROCEDIMENTO DE FACTIBILIZAÇÃO

Para calcular a complexidade do Procedimento de Factibilização, assume-se que não haja ciclagem durante o processo de devolução de tarefas ao conjunto **A** para posterior reinserção em **S**. O pior caso para o cálculo da complexidade do Procedimento de Factibilização ocorre quando há o maior número de devoluções de tarefas ao conjunto **A**, causando a maior quantidade de análises de sobreposição possível. Quando não há nenhuma devolução, a inserção da j-ésima tarefa em **S** provoca, no máximo, $(j-1)$ análises de sobreposição. Portanto, uma instância de pior caso para esse procedimento pode ser caracterizada da seguinte forma:

1. Tempos de processamento:

$$p_j \geq \sum_{i=1}^{j-1} p_i, \ \forall j \in \mathbf{N}, j > 1 \qquad (B.1)$$

2. Instantes de liberação para processamento:

$$r_j = r, \ \forall j \in \mathbf{N} \qquad (B.2)$$

3. Datas de entrega:

$$d_j \leq r_j + p_j, \ \forall j \in \mathbf{N} \qquad (B.3)$$

Devido às características 2 e 3, todos os intervalos preferenciais de posicionamento das tarefas começam no mesmo instante. Portanto, a ordenação feita pelo Procedimento I pode ter como saída qualquer permutação das tarefas. A pior permutação será aquela em que o

conjunto **A** esteja ordenado na ordem dos índices das tarefas.

4. Penalidades:

Todas as penalidades são tais que, ao se realizar uma análise de sobreposição entre duas tarefas J_i e J_j, a menor alteração de custo será dada quando J_i se desloca à direita, se $i < j$.

Uma estrutura de custos que garante o cumprimento desse objetivo, é:

$$t_j . p_i - t_i . p_j > 0, \forall i, j \in \mathbf{N}, i < j$$
$$t_j, h_j > 0, \forall j \in \mathbf{N} \qquad (B.4)$$

Antes de mostrar como cada uma dessas características é necessária para que a instância represente o pior caso, deve-se visualizar o que ocorre nas primeiras iterações da heurística. As figuras B.1 a B.4 mostram os eventos ocorridos até a iteração 4. A estrutura de custos sugerida em (B.4) é utilizada nos cálculos de sobreposição das figuras B.2 e B.3 para mostrar que essa expressão é suficiente para caracterizar as penalidades de uma instância de pior caso, como descrito em 4.

Nas figuras, cada seta indica uma análise de sobreposição realizada. Na iteração j, seja S_j o número total dessas análises. A complexidade do Procedimento de Factibilização será calculada em função de S_j.

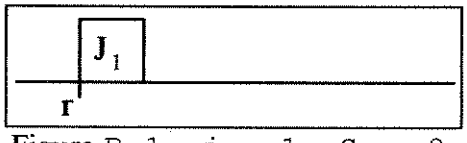

Figura B.1- $j = 1$, $S_1 = 0$.

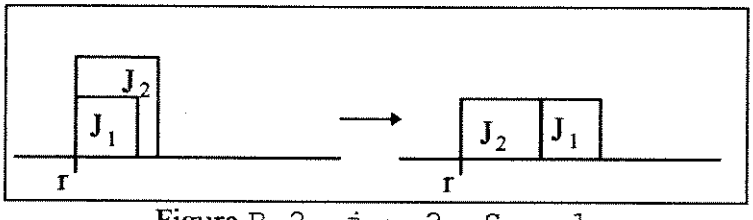

Figura B.2- $j = 2$, $S_2 = 1$.

Sobreposição entre J_2 e J_1:

$\Delta_1 = t_1 . p_2$

$\Delta_2 = t_2 . p_1$

$\Delta_3 = \Delta_1$

$\Delta_4 = \Delta_2$

$\Delta_2 - \Delta_1 = t_2 . p_1 - t_1 . p_2$

De (B.4):

$t_2.p_1 - t_1.p_2 > 0 \rightarrow \Delta_2 > \Delta_1 \rightarrow J_2$ se desloca à direita.

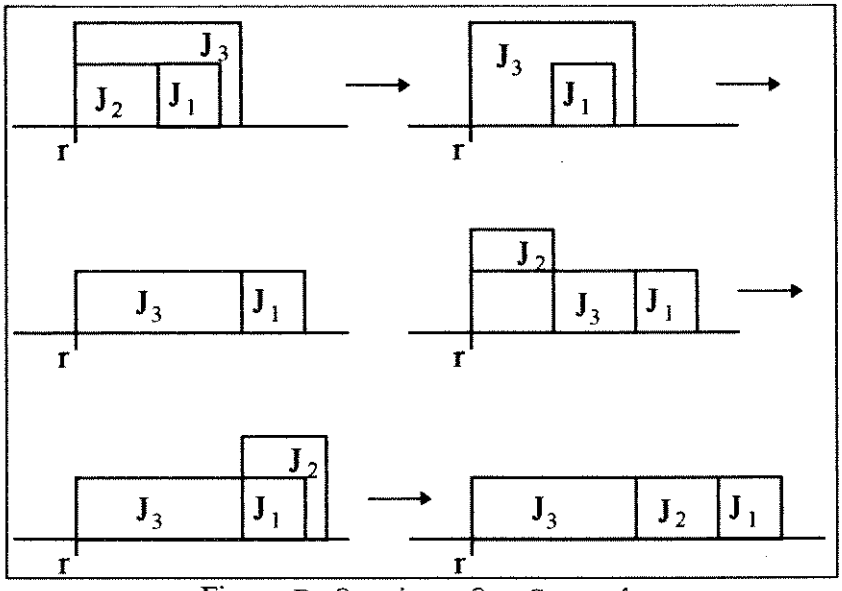

Figura B.3- j = 3, S_3 = 4.

Sobreposição entre J_3 e J_2:

$\Delta_1 = t_2.p_3$

$\Delta_2 = t_3.p_2$

$\Delta_3 = \Delta_1$

$\Delta_4 = \Delta_2$

$\Delta_2 - \Delta_1 = t_3.p_2 - t_2.p_3$

De (B.4):

$t_3.p_2 - t_2.p_3 > 0 \rightarrow \Delta_2 > \Delta_1 \rightarrow J_2$ se desloca à direita.

Sobreposição entre J_3 e J_1:

$\Delta_1 = t_1.(p_3 - p_2)$

$\Delta_2 = t_3.(p_2 + p_1)$

$\Delta_3 = \Delta_1$

$\Delta_4 = t_3.p_1 + h_1.p_2$

$\Delta_2 - \Delta_1 = t_3.p_2 + t_3.p_1 - t_1.p_3 + t_1.p_2$

$= p_2.(t_1 + t_3) + t_3.p_1 - t_1.p_3$

$\Delta_3 - \Delta_1 = t_3.p_1 + h_1.p_2 - t_1.p_3 + t_1.p_2 =$

$= p_2.(h_1 + t_1) + t_3.p_1 - t_1.p_3$

De (B.4):

$t_3.p_1 - t_1.p_3 > 0 \rightarrow \Delta_2 > \Delta_1$ e $\Delta_4 > \Delta_1 \rightarrow$ J_1 se desloca à direita.

A análise das sobreposições de J_2 com J_3 e J_1, na sua reinserção em **S**, ocorrem de maneira idêntica às análises já realizadas.

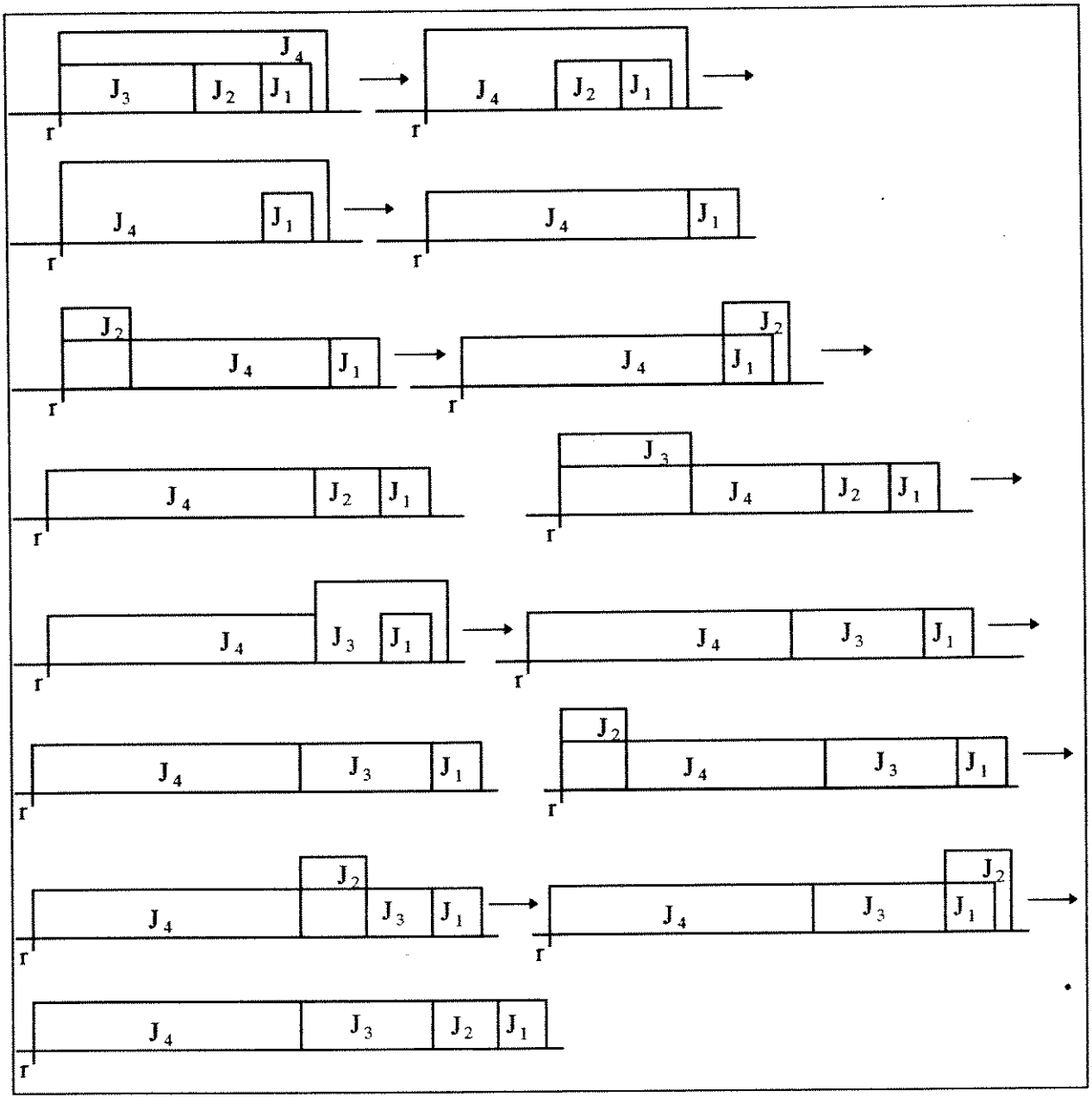

Figura B.4- j = 4, S_4 = 11.

A partir da ilustração das primeiras iterações da heurística, pode-se mostrar que qualquer instância que não tenha uma das características enumeradas no início dessa seção causará um número menor de análises de sobreposição no Procedimento II.

1. Tempos de processamento:

Se todas as tarefas da instância possuem a característica dada em (B.1), sempre que uma tarefa for inserida no programa parcial **S**, ocorrerá a situação de duas tarefas causando infactibilidades e uma delas terá que ser devolvida ao conjunto **A**. Se

(B.1) não for observada em alguma das tarefas dessa instância, pode ocorrer que sua inserção em **S** seja feita sem que se devolva nenhuma tarefa a **A**, evitando-se todas as análises de sobreposição adicionais decorrentes de reinserções de tarefas em **S**.Usando como exemplo a Figura B.4, observa-se que: se o tempo de processamento de J_4 fosse menor que o tempo de processamento de J_3, quando a análise de sobreposição indicasse que J_3 se desloca à direita, somente essa última tarefa causaria infactibilidade em **S**, e não teria que ser devolvida a **A**. O valor de S_4, portanto, seria menor que o indicado na Figura B.4.

Portanto, se (B.1) não for observada para todas as tarefas de uma instância, ela não representará o pior caso para o Procedimento de Factibilização.

2. Instantes de liberação para processamento

No caso de alguma das tarefas possuir um instante de liberação diferente de r, e a característica (B.3), relativa às datas de entrega, for mantida o intervalo preferencial de posicionamento dessa tarefa terá um instante de início diferente do mesmo intervalo para as demais tarefas da instância. Nesse caso, ao se inserir essa tarefa em **S**, a análise de sobreposição não envolverá todas as tarefas previamente programadas. Portanto, o número de tais análises será menor. Como exemplo, se a instância fosse tal que:

$$r_4 > r + p_1 + p_2 + p_3$$

não haveria nenhuma análise de sobreposição na inserção de J_4 em **S**.

Portanto, a característica (B.2) deve ser observada na instância de pior caso para o Procedimento de Factibilização.

3. Datas de entrega

O efeito da não observância da característica (B.3) é análogo ao discutido com relação aos instantes de liberação para processamento, pois as datas de entrega também são utilizadas no cálculo dos intervalos preferenciais de posicionamento de cada tarefa.

4. Penalidades

Para analisar o efeito das penalidades sobre a instância de pior caso, supõe-se que todas as características anteriores são observadas e as penalidades são tais que haja a possibilidade de ocorrência de $\Delta = \Delta_2$ (ou $\Delta = \Delta_4$). Esse valor de Δ, provoca o deslocamento de $J_{i(j)}$ à direita, e não provoca a situação de duas tarefas causando infactibilidades em **S**. Sendo assim, não haverá devolução de tarefas ao conjunto **A** e, portanto, ocorrerá um número menor de análises de sobreposição.

O último fator a ser analisado é a ordenação do conjunto **A** pelo Procedimento I. Considera-se que a pior ordenação possível é a que segue a ordem dos índices das tarefas. Para mostrar que o número de análises de sobreposição diminui com qualquer outra ordenação, supõe-se que, na Figura B.3, a tarefa a ser inserida é J_4 e não J_3. O número de sobreposições, nessa iteração é o mesmo. Mas, supondo que J_3 seja inserida em **S** na próxima iteração, a análise de sobreposição indicará que J_3 se desloca à direita (pois todas as características da instância de pior caso se mantém, em particular a que diz respeito às penalidades). Sendo assim, só restam ser analisadas as sobreposições envolvendo J_3, J_2 e J_1. O resultado seria, então: $S_4 = 6$, um número menor que o encontrado supondo a ordenação pelos índices das tarefas.

Tendo sido mostrado que essa é realmente a instância de pior caso para o Procedimento de Factibilização, resta estabelecer uma relação entre o número de tarefas em **S** ao final da iteração j e o número de análises de sobreposição realizadas nessa iteração. As figuras B.1 a B.4 mostram as iterações de 1 a 4. O efeito da devolução de tarefas ao conjunto **A** começa a ser sentido com mais clareza quando se passa de $j = 3$ para $j = 4$. Analisa-se, a seguir, os eventos da iteração 4.

Ocorrem três análises de sobreposição iniciais entre J_4, J_3, J_2 e J_1. As tarefas J_3 e J_2 são devolvidas a **A**, nessa ordem. Portanto, serão reinseridas em **S** na ordem inversa (pois sempre se reinsere a última tarefa devolvida).

Quando J_2 é reinserida, ela encontra duas tarefas em **S**, ou seja, serão feitas duas análises de sobreposição na reinserção de J_2.

Quando J_3 é reinserida, ela encontra três tarefas em **S** e, portanto, serão feitas três análises de sobreposição. Quando se analisa J_3 com J_2, J_2 retorna a **A** e será a próxima tarefa a ser reinserida nessa iteração.

Na reinserção de J_2, há três tarefas em **S** e são feitas as três últimas análises de sobreposição dessa iteração.

A Tabela B.1 mostra um resumo desses eventos

tarefa reinserida em **S**	número de análises de sobreposição
J_2	2
J_3	3
J_2	3

Tabela B.1 - Número de análises de sobreposição para $j = 4$.

Observa-se que : A inserção de J_4 causou 3 análises de sobreposição,
2 retornos provocaram 3 análises de sobreposição, e
1 retorno provocou 2 análises de sobreposição.

Para poder estender a análise é preciso observar o que ocorre na iteração 5. Isso pode ser feito através das figuras B.5 a B.8, a seguir.

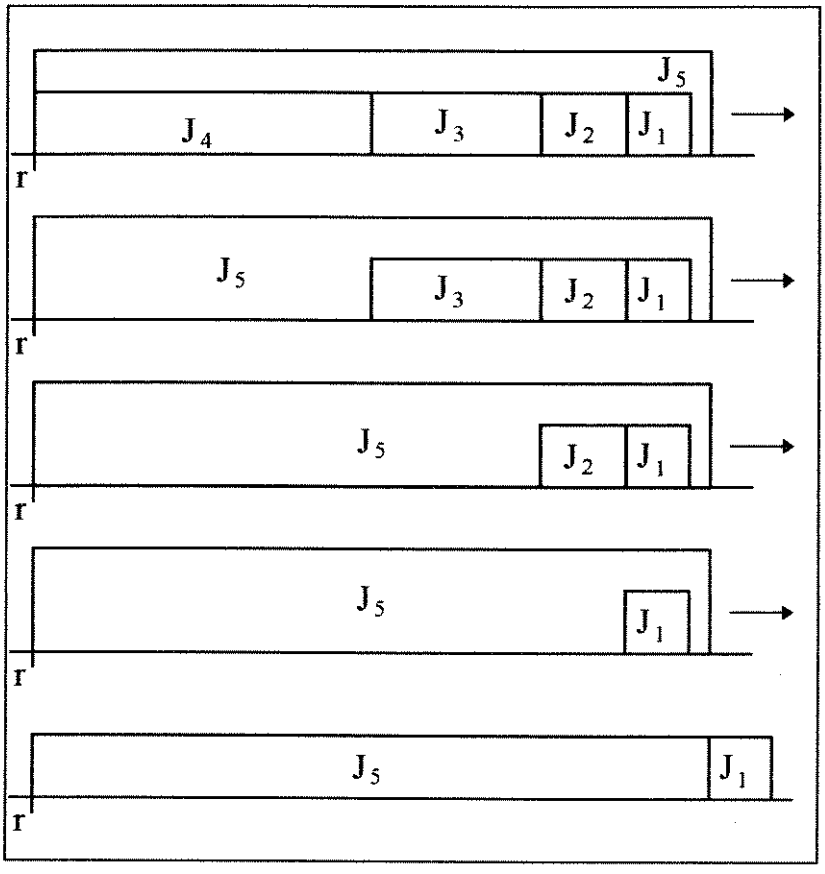

Figura B.5 - j = 5, S_5 = 27 (continua).

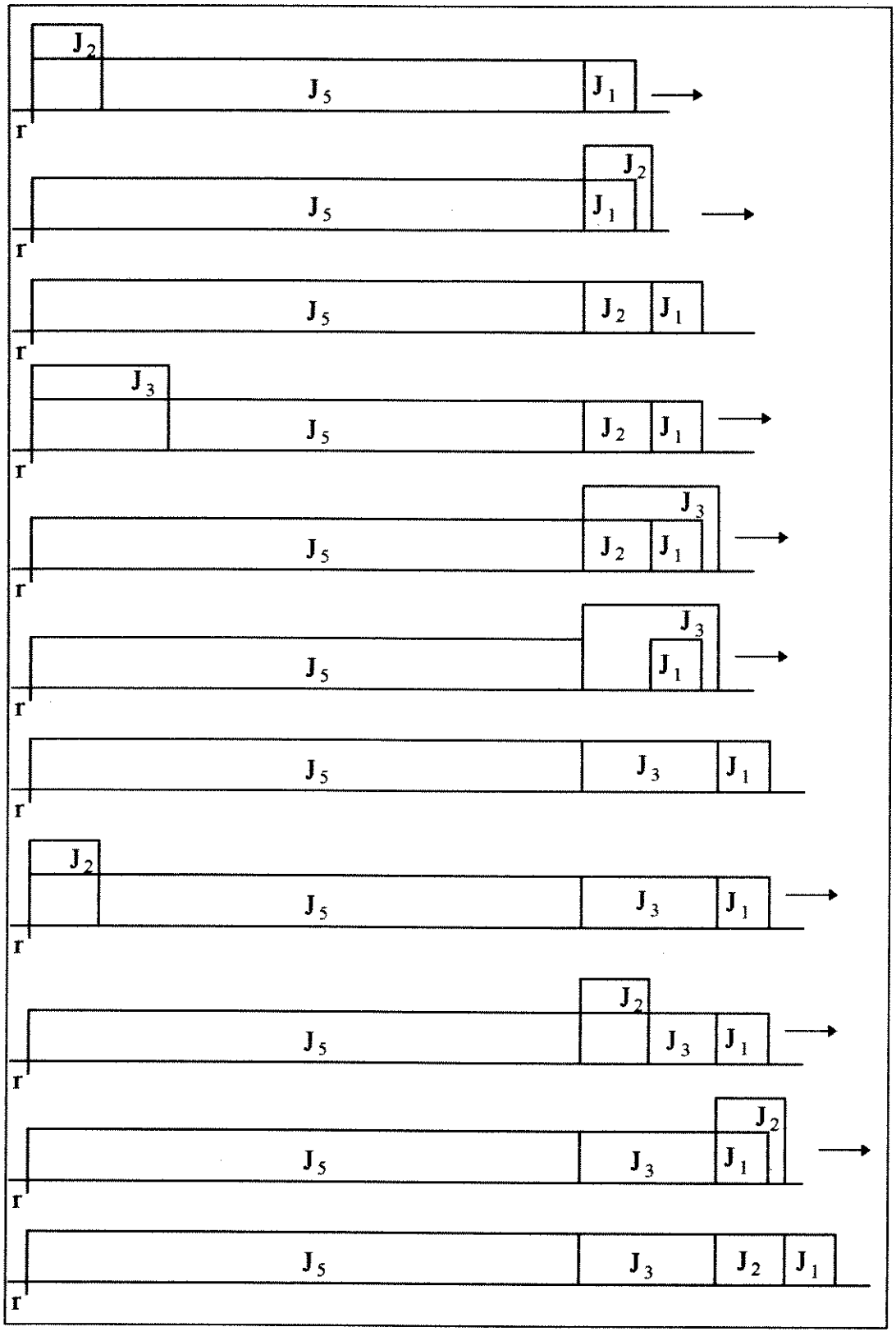

Figura B.6 (continuação da Figura B.5) - $j = 5$, $S_5 = 27$ (continua).

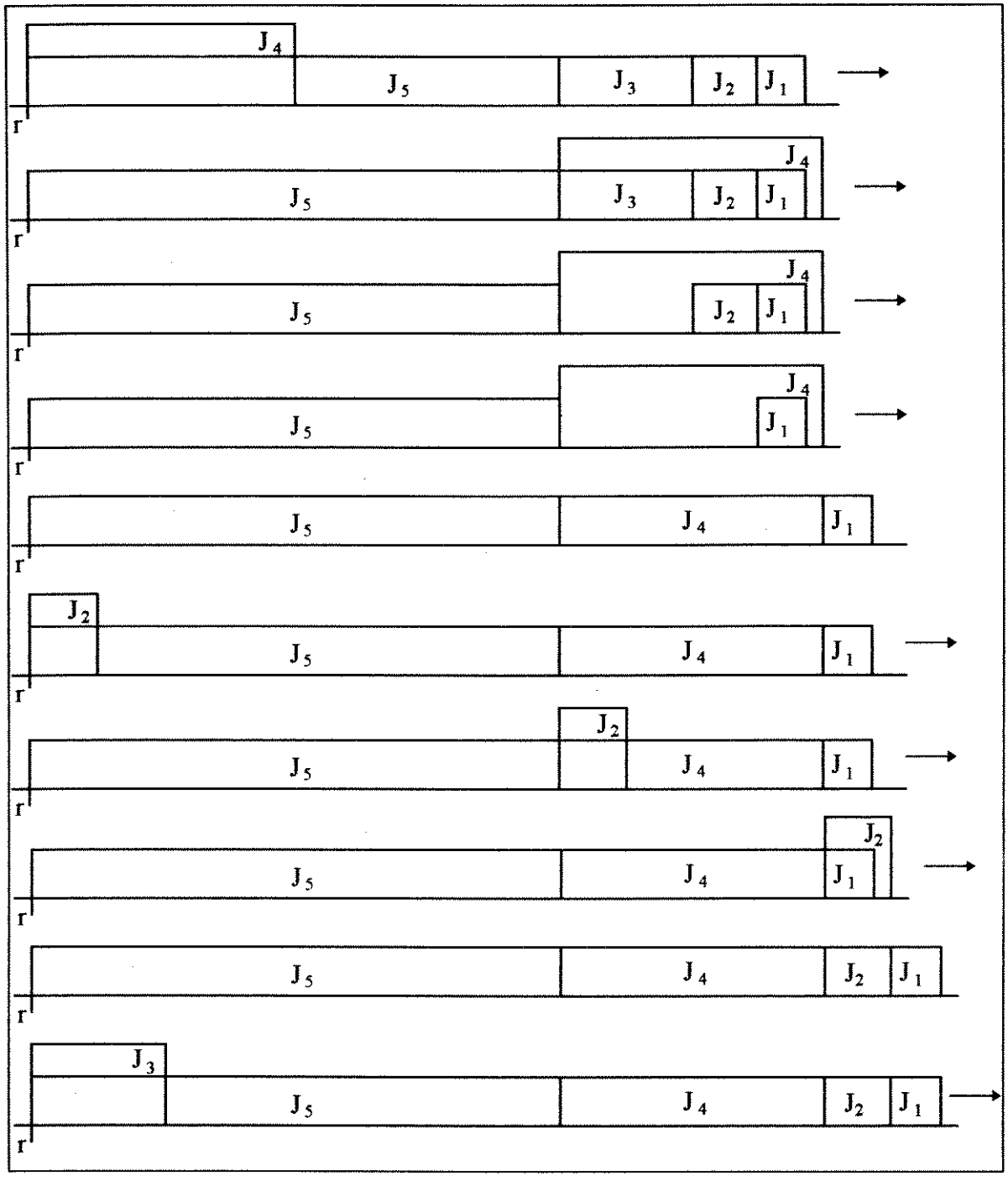

Figura B.7 (continuação da Figura B.6) - j = 5, S_5 = 27 (continua).

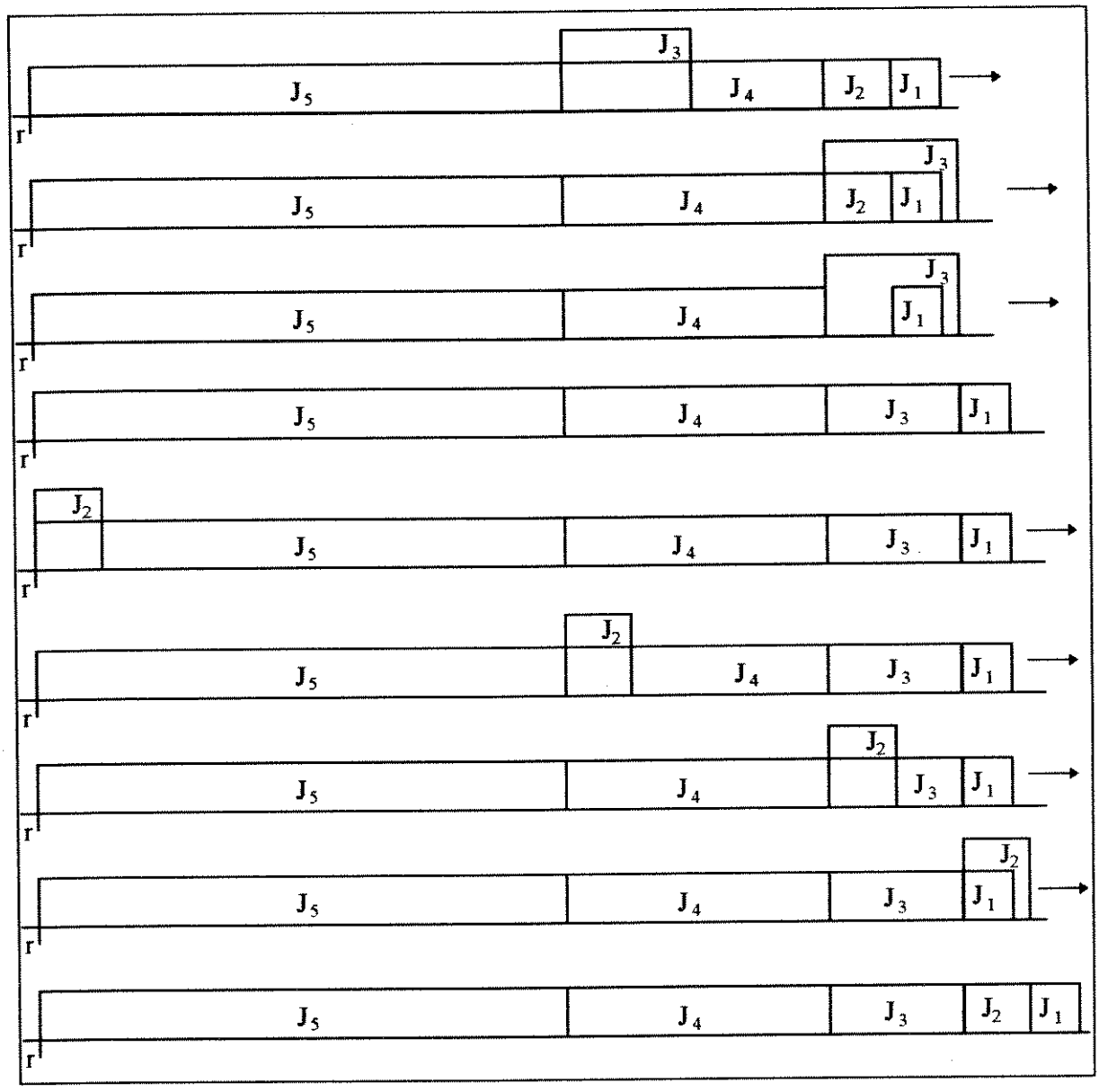

Figura B.8 (continuação da Figura B.7) - j = 5, S_5 = 27.

Pode-se fazer um resumo análogo ao que foi feito para j = 4:

Ao se inserir J_5 no programa **S**, ocorrem quatro análises de sobreposição iniciais entre J_5 e J_4,J_3,J_2 e J_1. As tarefas J_4, J_3 e J_2 são devolvidas ao conjunto **A**. A Tabela B.2 mostra o número de análises de sobreposição ocorridas em função da reinserção de tarefas, na ordem de ocorrência na Figura B.5.

tarefa reinserida em **S**	número de análises de sobreposição
J_2	2
J_3	3
J_2	3
J_4	4
J_2	3
J_3	4
J_2	4

Tabela B.2 - Número de análises de sobreposição para $j = 5$.

Observa-se que: A inserção de J_5 provocou 4 análises de sobreposição,
3 reinserções provocaram 4 análises de sobreposição,
3 reinserções provocaram 3 análises de sobreposição, e
1 reinserção provocou 2 análises de sobreposição.

Pode-se visualizar de uma forma concisa as observações feitas para $j = 4$ e $j = 5$ na Tabela B.3. Nessa tabela, cada elemento representa o número de reinserções que provocam o mesmo número de análises de sobreposição k.

j	k = (j-1)	k = (j-1)	k = (j-2)	k = (j-3)
4	1	2	1	
5	1	3	3	1

Tabela B.3 - Resumo dos resultados das tabelas B.1 e B.2.

Conclui-se que, a partir de $j = 4$, o número de reinserções de tarefas que provocam o mesmo número de sobreposições a serem analisadas segue uma lei de formação dada pelos coeficientes binomiais calculados como:

$$\binom{j-2}{i}, \forall i = 0,1,\ldots,j-2 \qquad (B.5)$$

Portanto, pode-se calcular S_j, para $\forall j \geq 4$ como:

$$S_j = (j-1) + (j-1)\cdot\binom{j-2}{1} + \sum_{i=2}^{j-2}\binom{j-2}{i}\cdot(j-i)$$

$$= (j-1)^2 + \sum_{i=2}^{j-2}\binom{j-2}{i}\cdot(j-i) \leq$$

$$\leq (j-1)^2 + \binom{j-2}{\left\lceil \frac{j-2}{2} \right\rceil} \cdot \sum_{i=2}^{j-2} (j-i) \qquad (B.6)$$

onde $\lceil x \rceil$ é o menor inteiro maior ou igual a x.

Fazendo $t = \left\lceil \frac{j-2}{2} \right\rceil$:

$$S_j \leq (j-1)^2 + \binom{j-2}{t} \cdot \frac{(j-3) \cdot j}{2} =$$

$$= (j-1)^2 + \frac{(j-2)!}{t!\,(j-2-t)!} \cdot \frac{j.(j-3)}{2} \qquad (B.7)$$

Primeiro caso: j par $\rightarrow t = \frac{j-2}{2} \rightarrow (j-2) = 2.t$:

$$\frac{(j-2)!}{t!\,(j-2-t)!} = \frac{(2.t)!}{t!\,(2.t-t)!} = \frac{(2.t)!}{t!\,t!} = \frac{\prod_{i=1}^{2.t} (2.t-i)}{\left(\prod_{i=1}^{t} (t-i)\right)^2} =$$

$$= \frac{1.2.\ldots.(t).(t+1).(t+2).\ldots.(2.t)}{(1.2.\ldots.t).(1.2.\ldots.t)} =$$

$$= \frac{(t+1).(t+2).\ldots.(2.t)}{1.2.\ldots.t} <$$

$$< \frac{(2.t)^t}{t!} < \frac{(2.t)^t}{t} = 2^t.t^{t-1} \qquad (B.8)$$

Segundo caso: j ímpar $\rightarrow t = \frac{j-2}{2} + \frac{1}{2} = \frac{j-1}{2} \rightarrow (j-2) = 2.t-1$:

$$\frac{(j-2)!}{t!\,(j-2-t)!} = \frac{(2.t-1)!}{(t)!\,(2.t-1-t)!} = \frac{(2.t-1)!}{t!\,(t-1)!} =$$

$$= \frac{\prod_{i=1}^{2.t} (2.t-i)}{t.\left(\prod_{i=1}^{t} (t-1-i)\right)^2} =$$

$$= \frac{1.2.\ldots.(t).(t+1).(t+2).\ldots.(2.t-1)}{(1.2.\ldots.t).(1.2.\ldots.(t-1))} =$$

$$= \frac{(t+1).(t+2).....(2.t-1)}{1.2.....(t-1)} <$$

$$< \frac{(2.t-1)^{t-1}}{(t-1)!} < \frac{(2.t)^t}{t-1} \qquad (B.9)$$

De (B.7), (B.8) e (B.9):

$$S_j < (j-1)^2 + \frac{j.(j-3)}{2} . \begin{cases} 2.(j-2)^{\left(\frac{j-4}{2}\right)}, & \text{se } j \text{ é par} \\ \frac{2.(j-1)^{\left(\frac{j-1}{2}\right)}}{j-3}, & \text{se } j \text{ é ímpar} \end{cases} \qquad (B.10)$$

Como

$$2.(j-2)^{\left(\frac{j-4}{2}\right)} < 2.j^{(j-2)} \rightarrow \frac{j.(j-3)}{2}.2.(j-2)^{\left(\frac{j-4}{2}\right)} < j^j \text{ e}$$

$$\frac{2.(j-1)^{\left(\frac{j-1}{2}\right)}}{j-3} < \frac{2.j^{(j-1)}}{j-3} \rightarrow \frac{j.(j-3)}{2} . \frac{2.(j-1)^{\left(\frac{j-1}{2}\right)}}{j-3} < j^j$$

Pode-se calcular um limitante para S_j como:

$$S_j < (j-1)^2 + j^j \qquad (B.11)$$

Calculando o número total de análises de sobreposição realizadas:

$$T = \sum_{j=1}^{n} S_j = S_1 + S_2 + S_3 + \sum_{j=4}^{n} (j-1)^2 + \sum_{j=4}^{n} j^j <$$

$$< 5 + \left[\frac{n.(n-1).(2.n-1)}{6} - 13\right] + \sum_{j=4}^{n} n^j =$$

$$= \frac{2.n^3 - 3.n^2 + n - 48}{6} + \left[\frac{n^{n+1} - 1}{n-1} - 1 - n - n^2 - n^3\right] <$$

$$= \frac{2.n^3 - 3.n^2 + n - 48 + 6.n^{n+1}}{6} \qquad (B.12)$$

De (B.12) conclui-se que a complexidade de pior caso para o Procedimento de Factibilização é: $O(n^{n+1})$.

B.3 - PROCEDIMENTO DE ATUALIZAÇÃO DE INTERVALOS OCIOSOS

A instância de pior caso para esse procedimento pode ser descrita da seguinte forma:

Após a factibilização do programa **S**, na iteração j (inserção da j-ésima tarefa), há j blocos formados de apenas uma tarefa, separados entre si por intervalos ociosos. O procedimento analisa os blocos começando pelo último deles. Por hipótese, todos os blocos analisados se concatenam com o anterior. Portanto, ao analisar o k-ésimo bloco ($k = j, j-1, \ldots, 1$), haverá ($j - k + 1$) tarefas nesse bloco e, portanto, ocorrerá um número de avaliações de possibilidade de deslocamento e cálculos do custo marginal, V, proporcionais a ($j - k + 1$).

Para analisar todos os blocos de **S**, na iteração j, o número de operações básicas realizadas é proporcional a

$$\sum_{k=1}^{j} (j - k + 1) = j + (j - 1) + (j - 2) + \ldots + 1 = \frac{j.(j + 1)}{2} \qquad (B.13)$$

Somando para todas a iterações ($j=1, \ldots, n$):

$$\sum_{j=1}^{n} \frac{j.(j + 1)}{2} = \sum_{j=1}^{n} \left(\frac{j^2}{2} + \frac{j}{2} \right) = \frac{1}{2} \cdot \left(\frac{n.(n + 1).(2.n + 1)}{6} + \frac{n.(n + 1)}{2} \right) =$$

$$= \frac{n^3 + 3.n^2 + 2.n}{6} \qquad (B.14)$$

De (B.14) conclui-se que o tempo computacional requerido pelo Procedimento de Atualização de Intervalos Ociosos é $O(n^3)$.

B.4 - PROCEDIMENTO DE BUSCA EM VIZINHANÇA

A instância de pior caso para o Procedimento de Busca em Vizinhança é aquela em que a Fase Construtiva da Heurística gera um programa **S** com uma sequência de produção invertida com relação ao ótimo local da busca na sua vizinhança TTA. Por exemplo, supõe-se **S** = $\{J_1, J_2, \ldots, J_n\}$ e o ótimo local obtido ao final da busca em vizinhança igual a $\{J_n, J_{(n-1)}, \ldots, J_1\}$. Além disso, deve-se supor que todos os pares de tarefas consecutivas sejam adjacentes, ou seja, há ($n - 1$) pares de tarefas adjacentes em **S**, sem intervalo ocioso entre elas.

Usando o exemplo dado como referência e seguindo o algoritmo dado para o Procedimento de Busca em Vizinhança, pode-se construir a Tabela B.5 onde se mostra o número de análises de adjacência e o número de trocas realizadas para cada par de tarefas $J_{[i]}$ e $J_{[i+1]}$ no programa de produção. Lembrando a notação utilizada, [i] representa o índice da tarefa na i-ésima posição do programa.

par($J_{[i]}$-$J_{[i+1]}$)	número de análises de adjacência	número de trocas
i=1	1	1
i=2	3	2
i=3	5	3
i=4	7	4
i=5	9	5

Tabela B.5 - Número de análises de adjacência e trocas no pior caso para o Procedimento de Busca em Vizinhança.

Para cada valor de i, realizam-se i + (i - 1) análises de adjacência. Calculando a soma para todos os valores de i, obtém-se o número total dessas análises :

$$T = \sum_{i=1}^{n-1} (2.i - 1) = 2.\sum_{i=1}^{n-1} i - (n-1) = n.(n-1) - (n-1) =$$

$$T = (n-1)^2 \qquad (B.15)$$

Portanto, a complexidade de pior caso para o Procedimento de Busca em Vizinhança é $O(n^2)$.

B.5 - COMPLEXIDADE DA HEURÍSTICA

A complexidade dominante entre todos os procedimentos da heurística proposta é a do Procedimento de Factibilização. Portanto, a complexidade total da heurística no pior caso é $O(n^{n+1})$.

APÊNDICE C

INSERÇÃO ÓTIMA DE INTERVALOS OCIOSOS

Na seção 4.2 foi apresentado o procedimento TIMETABLER (modificado) para inserção ótima de intervalos ociosos em uma dada seqüência de processamento de tarefas em **N**. Neste apêndice são apresentadas algumas definições necessárias ao entendimento do mecanismo de ação do procedimento e um exemplo numérico.

DEFINIÇÃO 1: "Dado um programa de produção, um *movimento* de qualquer tarefa J_i é definido como a mudança em seu instante de término c_i que não altere a permutação original das tarefas. Mais precisamente, assuma-se que as tarefas em um dado programa de produção $(c_1, c_2, \ldots, c_n)$ estão sequenciadas de acordo com seus índices. Um *movimento* à direita da tarefa J_i faz com que seu instante de término se torne $c'_i > c_i$ e produz um novo programa $(c'_1, c'_2, \ldots, c'_n)$ de acordo com a seguinte sequência de passos:

$c'_j = c_j$, $j < i$;

$c'_i > c_i$ de acordo com o movimento da tarefa J_i;

$c'_{i+1} = \max\{c'_i + p_{i+1}, c_{i+1}\}$;

.

.

.

$c'_n = \max\{c'_{n-1} + p_n, c_n\}$.

De forma similar, um movimento de J_i à esquerda modifica seu instante de término para $c'_i < c_i$ e produz um novo programa de produção $(c'_1, c'_2, \ldots, c'_n)$ de acordo com os seguintes passos:

$c'_j = c_j$, $j > i$;

$c'_i < c_i$ de acordo com o movimento da tarefa J_i;

$c'_{i-1} = \min\{c'_i - p_{i-1}, c_{i-1}\}$;

.

.

.

$c'_1 = \min\{c'_2 - p_2, c_1\}$."

DEFINIÇÃO 2: "Um programa de produção semiativo é aquele em que nenhum movimento de tarefas diminuirá o custo de processamento."

O exemplo simples, na Figura C.1 é uma instância do problema E/T com $h_1 > t_2 > 0$. O exemplo ilustra a conversão de um programa não semiativo em um programa semiativo. Note-se que o movimento de uma tarefa pode modificar o instante de término de mais de uma das demais tarefas.

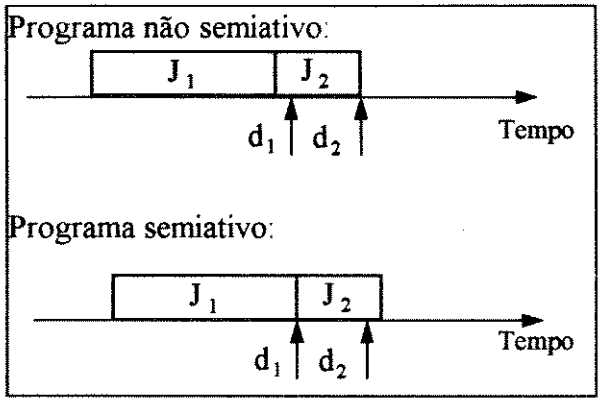

Figura C.1 - Exemplo de programa de produção semiativo com $h_1 > t_2 > 0$.

A seguir, será dado um exemplo do funcionamento do procedimento TIMETABLER (modificado). Antes disso, porém, deve-se repetí-lo para que se possa acompanhar melhor o desenvolvimento do exemplo.

Procedimento TIMETABLER (modificado):

Passo 1: Faça t = 1 com S(t,P) nulo e P = ∅;

Passo 2: Selecione a última tarefa J_i de P' e elimine-a de P';

Passo 3: Passe para o próximo estágio, fazendo:

3.1: Adicione J_i a S(t, P) para criar S(t + 1, P) da seguinte forma:

3.1.1: Inicialmente, a tarefa J_i é iniciada no instante $\max\{0, r_i, \sum_{j \in P'} p_j\}$, movendo as tarefas de S(t, P) para a direita, se necessário;

3.1.2: Movimente a tarefa J_i para a direita até que o custo marginal desse movimento se torne positivo;

3.2: Faça t = t + 1;

3.3: Faça P = iP;

Passo 4: Se P' ≠ ∅, vá para o Passo 2.
Caso contrário, Pare.

Em DAVIS e KANET (1993), prova-se que o procedimento acima insere intervalos ociosos de forma ótima, dada uma sequência de processamento para as tarefas.

A tabela a seguir apresenta os dados utilizados:

Tarefa (J_i)	r_i	p_i	d_i	h_i	t_i
J_1	69	34	159	25	31
J_2	166	97	295	108	84
J_3	14	22	264	34	1
J_4	87	30	121	6	30
J_5	62	24	221	61	17
J_6	270	47	237	86	46
J_7	310	91	181	84	98
J_8	55	40	136	73	69

Tabela C.1 - Dados do exemplo utilizado neste apêndice.

Seja **S** = $\{J_4, J_8, J_1, J_7, J_2, J_5, J_6, J_3\}$ a sequência de processamento em que se deseja inserir os intervalos ociosos de maneira ótima. As Figuras C.2 a C.9 ilustram as fases de construção do programa de produção correspondente a **S**.

A execução do procedimento TIMETABLER fica da seguinte forma:

Passo 1: $t = 1$; $S(1,P) = \emptyset$; $P = \emptyset$;
e $P' = \{J_4, J_8, J_1, J_7, J_2, J_5, J_6, J_3\}$

Passo 2: $J_i = J_3$; $P' = \{J_4, J_8, J_1, J_7, J_2, J_5, J_6\}$;

Passo 3:

3.1: $S(2, P) = \{J_3\}$;

3.1.1: $\max\{0, r_3, \sum_{j \in P'} p_j\} = \max\{0, 14, 363\} = 363$

3.1.2: J_3 está atrasada nessa posição. Portanto o custo marginal de seu movimento à direita é positivo.

3.2: $t = 2$;

3.3: $P = \{J_3\}$;

Passo 4: $P' \neq \emptyset$.

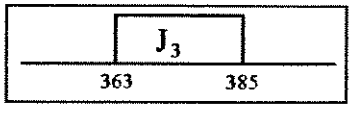

Figura C.2 - Estágio 1 do procedimento TIMETABLER.

Passo 2: $J_i = J_6$; $P' = \{J_4, J_8, J_1, J_7, J_2, J_5\}$;

Passo 3:

3.1: $S(3, P) = \{J_6, J_3\}$;

3.1.1: $\max\{0, r_6, \sum_{j \in P'} p_j\} = \max\{0, 270, 316\} = 316$

3.1.2: J_6 e J_3 estão atrasadas nessa posição. Portanto o custo marginal do movimento dessas tarefas, à direita, é positivo.

3.2: $t = 3$;

3.3: $P = \{J_6, J_3\}$;

Passo 4: $P' \neq \varnothing$.

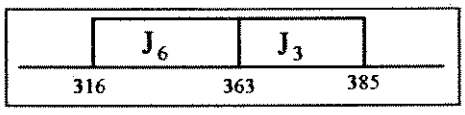

Figura C.3 - Estágio 2 do procedimento TIMETABLER.

Passo 2: $J_i = J_5$; $P' = \{J_4, J_8, J_1, J_7, J_2\}$;

Passo 3:

3.1: $S(4, P) = \{J_5, J_6, J_3\}$;

3.1.1: $\max\{0, r_5, \sum_{j \in P'} p_j\} = \max\{0, 62, 292\} = 292$

3.1.2: J_5, J_6 e J_3 estão atrasadas nessa posição. Portanto o custo marginal do movimento dessas tarefas, à direita, é positivo.

3.2: $t = 4$;

3.3: $P = \{J_5, J_6, J_3\}$;

Passo 4: $P' \neq \varnothing$.

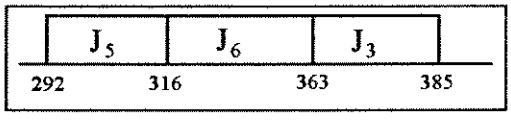

Figura C.4 - Estágio 3 do procedimento TIMETABLER.

Passo 2: $J_i = J_2$; $P' = \{J_4, J_8, J_1, J_7\}$;

Passo 3:

3.1: $S(5, P) = \{J_2, J_5, J_6, J_3\}$;

3.1.1: $\max\{0, r_2, \sum_{j \in P'} p_j\} = \max\{0, 166, 195\} = 195$

3.1.2: J_2 está adiantada e J_5, J_6 e J_3 estão atrasadas nessa posição.

O custo marginal do movimento dessas tarefas, à direita, é igual a $(-108 + 17 + 46 + 1) = -44$.

Portanto, deve ser feito um movimento à direita, fazendo com que J_2 termine de ser processada exatamente em sua data de entrega.

3.2: $t = 5$;

3.3: $P = \{J_2, J_5, J_6, J_3\}$;

Passo 4: $P' \neq \emptyset$.

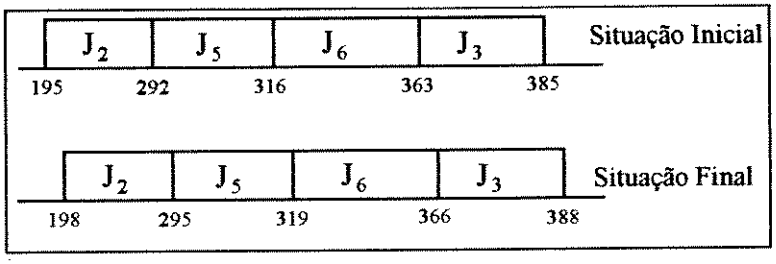

Figura C.5 - Estágio 4 do procedimento TIMETABLER.

Passo 2: $J_i = J_7$; $P' = \{J_4, J_8, J_1\}$;

Passo 3:

3.1: $S(6, P) = \{J_7, J_2, J_5, J_6, J_3\}$;

3.1.1: $\max\{0, r_7, \sum_{j \in P'} p_j\} = \max\{0, 310, 104\} = 310$

O posicionamento de J_7 fará com que todas as tarefas já em P sejam deslocadas à direita, como mostra a Figura C.6.

3.1.2: Todas as tarefas em P' estão atrasadas em sua posição atual, o que significa que o custo marginal de movimento à direita dessas tarefas é positivo

3.2: $t = 6$;

3.3: $P = \{J_7, J_2, J_5, J_6, J_3\}$;

Passo 4: $P' \neq \emptyset$.

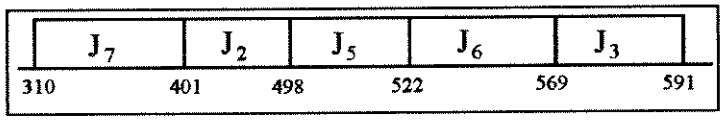

Figura C.6 - Estágio 5 do procedimento TIMETABLER.

Passo 2: $J_i = J_1$; $P' = \{J_4, J_8\}$;

Passo 3:

3.1: $S(7, P) = \{J_1, J_7, J_2, J_5, J_6, J_3\}$;

3.1.1: $\max\{0, r_1, \sum_{j \in P'} p_j\} = \max\{0, 69, 70\} = 70$

3.1.2: Como J_1 está adiantada nessa posição, o custo marginal de seu movimento à direita, até sua data de entrega é negativo. Como esse movimento não afeta as demais tarefas já programadas, não é necessário avaliar nenhum outro movimento de tarefas. Na Figura C.7, mostra-se o novo posicionamento de J_1.

3.2: $t = 7$;

3.3: $P = \{J_1, J_7, J_2, J_5, J_6, J_3\}$;

Passo 4: $P' \neq \emptyset$.

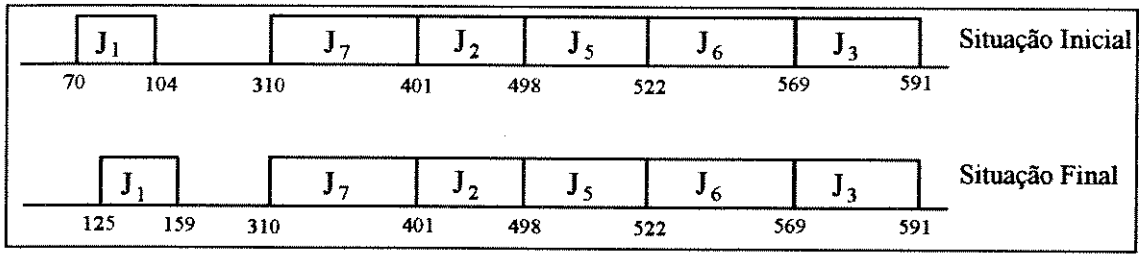

Figura C.7 - Estágio 6 do procedimento TIMETABLER.

Passo 2: $J_i = J_6$; $P' = \{J_4\}$;

Passo 3:

3.1: $S(8, P) = \{J_6, J_1, J_7, J_2, J_5, J_6, J_3\}$;

3.1.1: $\max\{0, r_6, \sum_{j \in P'} p_j\} = \max\{0, 55, 30\} = 55$

3.1.2: J_6 está adiantada nessa posição. O custo marginal de seu movimento à direita, até o instante de início de J_1, é igual a -73.

Como J_6 continua adiantada, deve-se avaliar o custo marginal do movimento das tarefas J_6 e J_1 à direita. Esse custo é igual $(-73 + 31) = -42$. Portanto, deve-se deslocar essas duas tarefas à direita até que o instante de término de J_6 coincida com sua data de entrega. Nenhuma outra tarefa foi afetada por esse movimento. Portanto, não é necessário considerar nenhum outro movimento nesse estágio. A Figura C.8 mostra o novo posicionamento de J_1 e de J_6.

3.2: $t = 8;$

3.3: $P = \{J_8, J_1, J_7, J_2, J_5, J_6, J_3\};$

Passo 4: $P' \neq \emptyset$.

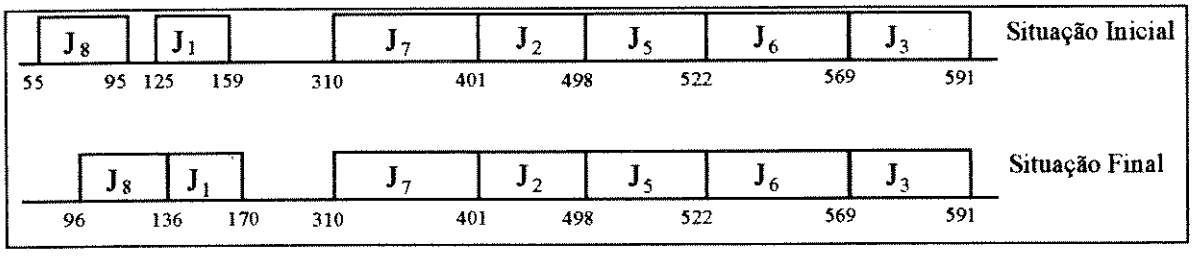

Figura C.8 - Estágio 7 do procedimento TIMETABLER.

Passo 2: $J_i = J_4$; $P' = \emptyset$;

Passo 3:

3.1: $S(9, P) = \{J_4, J_8, J_1, J_7, J_2, J_5, J_6, J_3\};$

3.1.1: $\max\{0, r_4, \sum_{j \in P'} p_j\} = \max\{0, 87, 0\} = 87.$

Para que J_4 possa ser iniciada nesse instante, as tarefas J_8 e J_1 devem ser deslocadas à direita, tornando o programa de produção factível.

3.1.2: Ainda assim, J_4 está adiantada. O custo marginal do movimento de J_4, J_8 e J_1 à direita é igual a

$(-6 + 69 + 31) = 94.$

Portanto, esse movimento não deve ser feito.

3.2: $t = 9;$

3.3: $P = \{J_4, J_8, J_1, J_7, J_2, J_5, J_6, J_3\};$

Passo 4: $P' = \emptyset$. Pare.

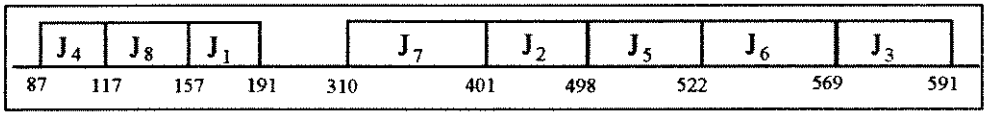

Figura C.9 - Estágio 8 do procedimento TIMETABLER.

REFERÊNCIAS BIBLIOGRÁFICAS

BAGCHI, Uttarayan, CHANG, Yih - Long, SULLIVAN, Robert S, (1987) - *Minimizing Absolute and Square Deviations of Completion Times with Different Earliness and Tardiness Penalties and a Common Due Date.* Naval Research Logistics, Vol 34, páginas 739 a 751.

BAKER, Kenneth R.., SCUDDER, Gary D., (1990) - *Sequencing with Ealiness and Tardiness Penalties: a review.* Operations Research, Vol 38, No.1, páginas 22 a 36.

DAVIS, J. Steve, KANET, John J., (1993) - *Single machine Scheduling with Early and Tardy Completion Costs.* Naval Research Logistics, Vol 40, No. 1, páginas 85 a 101.

DE, Prabuddha, GHOSH, Jay B., WELLS, Charles E., (1991) - *Scheduling to Minimize Weighted Earliness and Tardiness about a Common Due Date.* Computers & Operations Research, Vol 18, No. 5, páginas 465 a 475.

DE, Prabuddha, GHOSH, Jay B., WELLS, Charles E., (1993) - *On the General Solution for a Class of Early/Tardy Problems.* Computers & Operations Research, Vol 20, No. 2, páginas 141 a 149.

FRY, Timothy D., ARMSTRONG, Ronald D., BLACKSTONE, John H., (1987) - *Minimizing Weighted Absolute Deviation in Single Machine Scheduling.* IIE Transactions, Vol 19, No. 4, páginas 445 a 450.

GAREY, Michael R., TARJAN, Robert E., WILFONG, Gordon T., (1988) - *One-Processor Scheduling with Symmetric Earliness and Tardiness Penalties.* Mathematics of Operations Research, Vol 13, No. 2, páginas 330 a 348.

HALL, Nicholas G., POSNER, Marc E., (1991) - *Earliness-Tardiness Problems I: Weighted Deviation of Completion Times about a Common Due Date.* Operations Research, Vol 39, No. 5,páginas 836 a 846.

HALL, Nicholas G., KUBIAK, Wieslaw, SETHI, Suresh P., (1991) - *Earliness-Tardiness Scheduling Problems II: Deviation of Completion Times About a Restrictive Due Date*. Operations Research, Vol 39, No. 5, páginas 847 a 856.

HOOGEVEEN, J. A., OOSTERHOUT, H., VAN DE VELDE , S. L., (1994) - *New Lower and Upper Bounds for Scheduling Around a Common Due Date*. Operations Research, Vol 42, No. 1, páginas 102 a 110.

KAHLBACHER, Helmut G., (1993) - *Scheduling with Monotonous Earliness and Tardiness Penalties*.European Journal of Operational Research, No. 64, páginas 258 a 277.

KIM, Yeong Dae, YANO, Candance Arai, (1994) - *Minimizing Mean Tardiness and Earliness in Single-Machine Problems with Unequal Due Dates*. Naval Research Logistics, Vol 41, No. 7, páginas 913 a 933.

MANNUR, Narasimha R., ADDAGATLA, Jyothi Babu, (1993) - *Heuristic Algorithms for Solving Earliness-Tardiness Scheduling Problem with Machine Vacations*. Computers & Industrial Engineering, Vol 25, Nos. 1 - 4, páginas 255 a 258.

OW, Peng Si, MORTON, Thomas E., (1989) - *The Single Machine Early/Tardy Problem*. Management Science, Vol 35, No. 2, páginas 177 a 191.

POTTS, C. N., VAN WASSENHOVE, L. N., (1982) - *A Decomposition Algorithm for the Single Machine Total Tardiness Problem*. Operations Research Letters, No. 1, Vol 5, páginas 177 a 181.

SUNG, Chang S., JOO, U. G., (1992) - *A Single Machine Scheduling Problem with Earliness/Tardiness and Starting Time Penalties Under a Commom Due Date*. Computers & Operations Research, Vol 19, No. 8, páginas 757 a 766.

YANO, Candance Arai, KIM, Yeong Dae, (1991) - *Algorithms for a Class of Single Machine Weighted Tardiness and Earliness Problems*. European Journal of Operational Research, No. 52, páginas 167 a 178.