MARCO ANTONIO FREITAS DO EGITO COELHO

PROGRAMAÇÃO EM MÁQUINAS PARALELAS NÃO-RELACIONADAS, SUJEITAS A DIVISÃO DE TAREFAS

Este exemplar corresponde à redação final da tese defendida por Marco A. F. do Egito Coelho e aprovada pela Comissão Julgadora em 8 / 11 / 96.

Orientador

CAMPINAS

1996

FACULDADE DE ENGENHARIA ELÉTRICA
E COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

PROGRAMAÇÃO EM MÁQUINAS PARALELAS NÃO-RELACIONADAS, SUJEITAS A DIVISÃO DE TAREFAS

Autor: MARCO ANTONIO FREITAS DO EGITO COELHO
Orientador: PAULO MORELATO FRANÇA

Tese apresentada à Faculdade de Engenharia Elétrica e Computação da UNICAMP (FEEC - UNICAMP) como parte dos requisitos exigidos para a obtenção do título de DOUTOR em ENGENHARIA ELÉTRICA

CAMPINAS
1996

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

C65p

Coelho, Marco Antonio Freitas do Egito
Programação em máquinas paralelas não-relacionadas, sujeitas a divisão de tarefas / Marco Antonio Freitas do Egito Coelho.--Campinas, SP: [s.n.], 1996.

Orientador: Paulo Morelato França.
Tese (doutorado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Programação heurística. 2. Otimização combinatória. 3. Planejamento da produção. 4. Pesquisa operacional. I. França, Paulo Morelato. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

à minha família,

Ana Maria, Rodrigo e Rafael

ABSTRACT

Scheduling in unrelated parallel machines subject to job splitting

In this thesis new methods and procedures to solve the multimachine scheduling problem with early and tardy costs are proposed. The machines are unrelated, job splitting is allowed, the jobs may have different due dates and the changeover time to process a new job on a machine depends on the job previously processed on the same machine.

This problem is new, hard to solve and no references have been found in especialized literature. However, it has many applications in the manufacturing.

Two linear mixed-integer programming models one stated when job splitting is allowed, as well as similar model is proposed for the case without job splitting.

Since the problem is **NP**-hard, the models can only be used to solve small test problems. Based on the mixed-integer formulation, a tree search method is developed to be used in a Branch & Bound and in a Filtered Beam Search procedures. Two lower bounds are developed for the Branch & Bound and four bounds for the Filtered Beam Search. Some properties of the original problem and an efficient decomposition method to solve the LP model derived from the mixed-integer problem are presented. Many test problems with up to 120 jobs and 6 machines has been used to show the performance of the proposed methods.

Keywords: *multiprocessor scheduling, mixed integer models, tree search*

AGRADECIMENTOS

Ao Prof. Paulo Morelato França, pelo estímulo, pelo apoio, pela amizade.

Ao Prof. Hermano Tavares,

Ao Prof. Luís Gimeno Latre,

Ao Prof. Evandro Emílio M. Lima,

Ao Departamento de Engenharia Elétrica da Universidade de Brasília,

Ao Programa de capacitação de docentes - CAPES

À Faculdade de Engenharia Elétrica e de Computação da UNICAMP,

À Universidade Estadual de Campinas (UNICAMP),

Aos colegas e amigos da UNICAMP.

RESUMO

Programação em máquinas paralelas não-relacionadas, sujeitas a divisão de tarefas

Nesta tese novos métodos e procedimentos para resolver o problema de programação de tarefas em múltiplas máquinas paralelas com custos de atraso e adiantamento são propostos. As máquinas são não-relacionadas, a divisão de tarefas é permitida, as tarefas podem ter datas previstas de entrega diferentes e os tempos de preparação de máquina para executar uma tarefa depende da tarefa anterior processada na mesma máquina.

Este é um problema novo, de difícil resolução e não foram encontradas referências na literatura especializada. Apesar disso, tem muitas possíveis aplicações na manufatura.

O problema é formulado através de um modelo linear de programação inteira mista para o caso sem divisão de tarefas e, posteriormente, dois outros modelos lineares são propostos para o caso onde a divisão de tarefas é permitida. Como o problema de minimização foi mostrado ser **NP**-completo, os modelos podem apenas ser utilizados para resolver pequenos problemas de teste. A partir do modelo de programação inteira mista, um método de busca em árvore é desenvolvido para uso com procedimentos *Branch & Bound* e Busca em Feixe Filtrada. Dois limitantes inferiores são desenvolvidos para o *Branch & Bound* e quatro limitantes para a Busca em Feixe Filtrada. Algumas propriedades e teoremas do problema original e um método de decomposição eficiente para resolver o modelo linear derivado do modelo de programação inteira mista, também são apresentados. Muitos problemas de teste com até 120 tarefas e 6 máquinas são utilizados para mostrar o desempenho dos métodos desenvolvidos aqui.

ÍNDICE

1. Introdução

Problemas de programação de tarefas (*scheduling*) podem ser entendidos, de forma ampla, como problemas de alocação de recursos no tempo para a execução de um conjunto de tarefas. Por recursos, entendem-se meios arbitrários que são objeto de competição entre as tarefas. Estes podem ser tão diversos como: mão-de-obra, capital, processadores (máquinas), energia ou ferramentas. Também as tarefas podem ser interpretadas de várias maneiras, desde produtos acabados saindo de uma linha de produção industrial até o processamento de informações em sistemas computacionais. Às tarefas estão associados diversos atributos como datas previstas de início e/ou término, tempos de processamento e tempos de preparação. É possível ainda que as características sejam função da alocação decidida ou da tarefa precedente executada pelo mesmo recurso. Define-se programa como uma determinada escolha de alocações das tarefas nos recursos disponíveis, seqüenciamento das tarefas em cada recurso e definição precisa do instante de início (ou término) de execução de cada tarefa. Para medir a qualidade de um certo programa, existem também vários critérios.

A importância prática deste assunto é tão clara, que ele tem sido objeto de pesquisas desde a década de 1950. Atualmente existe uma quantidade grande de literatura dedicada a

esta área. Como referências gerais, pode-se citar o livro clássico de CONWAY, MAXWELL & MILLER (1967), os textos introdutórios de BAKER (1974), além de revisões como LAWLER et al (1993), BLAZEWICZ et al (1993) e MORTON & PENTICO (1993).

1.1 Limitação do problema

Devido à enorme variedade de tipos de problemas de programação de tarefas, existe a necessidade de caracterizar claramente o problema tratado neste trabalho. Há duas hipóteses iniciais:

1) Tipos de recursos e atividades: Uma máquina ou processador é um recurso capaz de executar no máximo uma atividade de cada vez e a qualquer instante. As atividades são normalmente chamadas de tarefas, que poderão ser divididas e executadas em mais de uma máquina num mesmo instante de tempo.
2) Problemas determinísticos: Todas as informações que definem uma instância do problema são conhecidas com certeza. Para tratar os casos em que novas tarefas surgem durante a execução das atuais, é preciso formular um novo problema incluindo as novas tarefas junto com as que ainda não foram executadas.

O ponto de vista abordado é estritamente operacional e não opina sobre decisões táticas como determinação de datas previstas para a finalização das tarefas e número ou características das máquinas a serem adquiridas. O objetivo é minimizar um critério de otimalidade simples (função objetivo) que inclui atrasos e adiantamentos das tarefas em relação às suas datas previstas, além de custos de troca entre tarefas que podem ser dependentes da seqüência destas em cada máquina.

1.2 Atrasos e adiantamentos de tarefas

Até pouco tempo atrás, a literatura de programação de tarefas era dominada por medidas de desempenho simples conhecidas como medidas regulares. Uma medida regular **Z** é definida como: 1) o objetivo da programação é minimizar **Z**. 2) **Z** só pode aumentar se pelo menos um dos tempos de término do programa aumentar. Tempo de fluxo médio, atraso médio e número de tarefas atrasadas são alguns exemplos de medidas regulares (BAKER, 1974).

O adiantamento de tarefas é claramente uma medida de desempenho não-regular, visto que esta pode diminuir com o aumento de um ou mais tempos de término de tarefas do

programa. No entanto, é importante considerar os adiantamentos de tarefas, visto que estes estão associados a custos de armazenamento e depreciação dos produtos.

A introdução de atrasos e adiantamentos dentro da função objetivo é uma prática relativamente recente, devido ao crescente interesse na filosofia *Just-In-Time* (*JIT*) da produção que desencoraja tanto os adiantamentos como os atrasos das tarefas. Cabe ressaltar que a filosofia *JIT* não se limita a especificar que atrasos e adiantamentos devem ser minimizados, mas do ponto de vista da programação de tarefas é uma aproximação razoável.

A idéia de penalizar tanto o atraso como o adiantamento torna o problema mais complexo, visto que a medida de desempenho não-regular exige o desenvolvimento de novos métodos para a resolução do problema. A introdução de tempos ociosos entre as tarefas a serem executadas em cada máquina, aumenta ainda mais a complexidade do problema, porém, devido à penalização de adiantamentos de tarefas, estes não devem ser desprezados. A abordagem que será feita nesta tese baseia-se na função objetivo básica de um programa S (BAKER & SCUDDER, 1990),

$$f(S) = \sum_{j \in T} [\alpha_j e(j) + \beta_j z(j)] \qquad (1.1)$$

onde:

T = conjunto de tarefas a serem executadas.
α_j = coeficiente de custo de adiantamento da tarefa j na função objetivo (diferente para cada tarefa).
β_j = coeficiente de custo de atraso da tarefa j na função objetivo (diferente para cada tarefa).
$a(j)$ = instante de término da tarefa j.
d_j = data prevista (*due date*) para o término da tarefa j (diferente para cada tarefa).
$z(j) = max(0, a(j) - d_j)$ = atraso da tarefa j.
$e(j) = max(0, d_j - a(j))$ = adiantamento da tarefa j.

Mais adiante será introduzido mais um termo na função objetivo para medir o custo total de trocas entre tarefas, mas serão os atrasos e adiantamentos os principais motivadores do método desenvolvido.

1.3 Definição do problema

Apesar da riqueza da literatura de programação em máquinas paralelas, pouco pode ser encontrado sobre programação em máquinas paralelas não relacionadas (*unrelated*), principalmente quando existe a necessidade de inserir entre as tarefas tempos de troca

(*changeover*) dependentes da seqüência, isto é, dependentes da última tarefa processada. Além disso, é permitida a divisão de cada tarefa entre as várias máquinas da planta (*job splitting*), considera-se que as tarefas têm diferentes datas previstas de término (*due dates*), é possível (e desejável) inserir tempos ociosos (*idle times*) entre as tarefas com o objetivo de minimizar uma função objetivo composta pelos atrasos (*tardiness*), pelos adiantamentos (*earliness*) e também por custos de preparação de máquina para executar uma nova tarefa (custos de troca). O problema considerado neste trabalho possui todas estas características, porém não permite que as tarefas sofram preempção em uma única máquina, isto é, uma vez iniciado o processamento de uma tarefa ou fração em uma máquina, não é permitida a interrupção do processamento para a execução de uma segunda tarefa para posteriormente continuar o processamento da primeira. Além disso, como já explicado, o problema é determinístico e supõe que todas as tarefas estão disponíveis no momento inicial ($t = 0$).

Uma programação de tarefas que considere estas características é conveniente para muitas indústrias, como as de produção de papel, de produção de tecidos, plásticos, químicas ou de produção de alimentos. Muitos trabalhos relativos à programação em máquinas paralelas são dirigidos à aplicação industrial mas, quase sempre, são feitas várias hipóteses restritivas com o intuito de simplificar o modelo, talvez para adequá-lo a uma limitada capacidade de resolução disponível. Modelos com algumas das características enfocadas nesta tese podem ser encontrados em SERAFINI & SPERANZA (1992) e GUINET (1991).

A tese está organizada da seguinte forma:

- Capítulo 1: Definição, caracterização e justificativa do problema enfocado.

- Capítulo 2: Explicação simplificada sobre complexidade de problemas de decisão e otimização e avaliação da complexidade do problema enfocado.

- Capítulo 3: Utilizando um modelo de programação inteira-mista desenvolvido por GUINET (1991), como ponto de partida, este é melhorado com a distinção dos tempos de troca entre as máquinas, acrescido com tempos ociosos entre as tarefas e adicionados o adiantamento e tempo de troca entre tarefas na função objetivo. Em seguida, a divisão das tarefas entre as máquinas, como descrito por SERAFINI & SPERANZA (1992), é introduzida. Também é apresentado um segundo modelo com divisão de tarefas como um aperfeiçoamento do primeiro.

- Capítulo 4: Os modelos de programação inteira-mista com divisão de tarefas do capítulo 3 são separados em um gabarito para a alocação e sequenciamento de frações de tarefas e um novo modelo sem variáveis inteiras. Algumas propriedades relativas ao problema de programação definido em 1.3 são demonstradas com a utilização do modelo sem variáveis inteiras.

- Capítulo 5: É desenvolvida uma busca em árvore do tipo *Branch and Bound* para preencher o gabarito e dois limitantes inferiores desenvolvidos a partir do modelo sem variáveis inteiras, o que leva à resolução ótima de problemas de pequenas dimensões

- Capítulo 6: É desenvolvida uma busca em árvore do tipo Busca em Feixe Filtrada (*Filtered Beam Search*) para preencher o gabarito, e quatro limitantes desenvolvidos a partir do modelo sem variáveis inteiras, o que leva à resolução sub-ótima de problemas de porte médio.

- Capítulo 7: São feitas experiências computacionais com os modelos de programação inteira-mista e várias outras com a busca em feixe filtrada, que atestam o bom desempenho do método em problemas com até 120 tarefas em 6 máquinas.

- Capítulos 8 e 9: Conclusões sobre o método, resultados obtidos e bibliografia.

- No apêndice são dados detalhes das técnicas desenvolvidas para a resolução do modelo de programação linear sem variáveis inteiras de maneira rápida e econômica.

2. Complexidade do problema

Entende-se como função de complexidade temporal de um algoritmo A resolvendo um problema Π, como a função que mapeia cada comprimento de entrada de uma instância I de Π no máximo número de passos elementares (ou unidades de tempo) de um computador, que são necessários para resolver uma instância daquele tamanho pelo algoritmo A. Esta função não estará bem definida sem que o esquema de codificação e o modelo de computador estejam precisamente definidos.

Do ponto de vista de complexidade computacional, existem dois tipos principais de algoritmos que são os de tempo polinomial e de tempo exponencial, isto é, o tempo de computação cresce polinomialmente ou exponencialmente com o tamanho da entrada. Neste sentido, todos os modelos reais de computadores são equivalentes, pois embora possam apresentar velocidades diferentes na resolução desta ou daquela instância, uma vez determinado que o algoritmo A resolve o problema Π em um tempo polinomial, isto é válido para qualquer computador capaz de resolver o mesmo problema com o algoritmo A (BLAZEWICZ et al, 1993). Um problema de tempo polinomial pode ser considerado "fácil" em comparação com um problema exponencial ("difícil"), pois mesmo que o primeiro seja mais

lento com entradas suficientemente pequenas, com o crescimento do tamanho da entrada, fatalmente o problema de tempo exponencial se tornará tão lento que a sua execução será impraticável. Evidentemente, o tempo de computação também pode crescer rapidamente nos problemas de tempo polinomial mas, comparativamente, o crescimento é muito menor.

Para a análise da dificuldade computacional dos problemas de otimização é conveniente transformá-los em problemas de decisão, onde a saída pode ser apenas "sim" ou "não". Um problema de minimização f pode ser visto como uma pergunta: $f(x) \leq k$ para qualquer k dado? Se o problema de decisão for "fácil", é possível obter um algoritmo de tempo polinomial para f, aplicando busca binária sobre k. A classe de problemas de decisão que pode ser resolvida em tempo limitado acima por um polinômio no comprimento da entrada é chamada de **P**. O computador tomado como referência é a máquina de Turing determinística, mas isto não afeta a generalidade das definições, visto que todos os modelos de computadores são relacionados polinomialmente.

Existe uma classe maior de problemas de decisão na qual não são conhecidos algoritmos de tempo polinomial, porém, uma resposta positiva pode ser verificada em tempo polinomial desde que haja alguma informação adicional. Esta característica de verificação em tempo polinomial em vez de resolução, é capturada pela máquina de Turing não-determinística (BLAZEWICZ et al, 1993).

A classe de problemas de decisão **NP** é definida como consistindo de todos os problemas de decisão que podem ser resolvidos em tempo polinomial pela máquina de Turing não-determinística. Pertencer à classe **NP**, não significa necessariamente o não conhecimento de algoritmos polinomiais capazes de resolver o problema, visto que existem problemas que pertencem à classe **NP** e sabe-se que podem ser resolvidos em tempo polinomial. Um dos maiores desafios da matemática moderna é estabelecer se ou quando **P** é igual a **NP**, pois no momento só existem conjecturas sobre este assunto.

Um problema **NP**-completo é, a grosso modo, um problema dos mais difíceis da classe **NP**, de tal maneira que se um destes problemas puder ser resolvido em tempo polinomial, então todos os problemas da classe **NP** poderão também ser resolvidos em tempo polinomial, isto é, **P** seria igual a **NP**. Determinar que um problema é **NP**-completo é uma forte evidência de que não existem algoritmos capazes de resolve-lo em um tempo polinomial. A principal noção na definição de um problema como **NP**-completo, é a da redução entre dois problemas

da classe **NP**. Considera-se que um problema P se reduz a Q (ou $P \alpha Q$) se existir uma função de tempo polinomial que transforma as entradas de P em entradas para Q de tal maneira que qualquer entrada de P depois de transformada fará Q responder exatamente da mesma forma que P. Um problema é **NP**-completo se for **NP** e todos os problemas em **NP** se reduzirem a ele. Um problema de otimização é chamado **NP**-difícil (**NP**-*hard*) se o problema de decisão associado for **NP**-completo. Existe ainda uma outra definição devido ao fato de existirem problemas **NP**-completos sobre determinada codificação da entrada (binária) e polinomiais sobre outra (unária). Problemas que permanecem **NP**-completos sobre codificação unária da entrada, são ditos fortemente **NP**-completos, o que sugere que é impossível encontrar sempre uma solução ótima rapidamente.

Para provar que um problema P é **NP**-completo, basta apenas provar que um outro problema Q é **NP** e que $P \alpha Q$, resultado conhecido como técnica de restrição (GAREY & JOHNSON, 1979). Para este objetivo, existem vários problemas básicos que são conhecidos serem **NP** ou **NP**-completos, como por exemplo, o problema clássico do caxeiro viajante ou mesmo alguns problemas básicos de máquinas simples.

GAREY, TARJAN & WILFONG (1988), provaram que o problema de minimização da discrepância total ($\alpha_j = \beta_j$ = constante para todo j em (1. 1)) em uma única máquina é NP-completo, resultado que se estende imediatamente para problemas em uma única máquina que utilizam a função objetivo básica (1. 1). O problema descrito no item 1.3, pode ser reduzido ao de uma única máquina simplesmente fazendo o parâmetro número de máquinas igual a 1 e os custos de troca nulos para todas as tarefas, o que é suficiente para considerá-lo como NP-completo (ou mais precisamente NP-difícil) e de resolução exata impraticável se as entradas tiverem dimensões grandes.

3. Modelos de programação matemática

Programação matemática é um rico formalismo que permite a construção de modelos de otimização aplicáveis tanto em programação de tarefas como em planejamento da produção. Recentemente tem havido um interesse crescente na descrição do problema por este enfoque, que fornece aos gerentes de produção uma melhor visão global do problema, permitindo a identificação de custos altos, além de fornecer sugestões no campo tático para aumento da produção (SHAPIRO, 1993).

No caso do problema tratado nesta tese esta descrição será utilizada apenas como ponto de partida na análise dos problemas de larga escala que ocorrem em indústrias reais, visto que, uma vez caracterizado o problema como **NP**-difícil, as soluções exatas dos modelos serão inatingíveis em exemplos médios ou grandes. Os modelos desenvolvidos a seguir, apresentam um grande número de variáveis binárias e também contínuas. A função objetivo e todas as restrições são lineares, caracterizando-os como modelos de programação linear inteira-mista. Existem vários *softwares* no mercado para a resolução de tais modelos, mas estes tem pouca utilidade para resolver problemas reais nos modelos a seguir.

3.1 *Modelo sem divisão de tarefas - modelo S*

O modelo abaixo baseia-se na proposta de GUINET (1991) para a resolução do problema de programação da produção onde não é permitida a divisão de tarefas, não existem tempos ociosos entre as tarefas e nem adiantamentos na função objetivo.

$$\textit{Minimize} \quad \sum_{j \in T} [\alpha_j e(j) + \beta_j z(j)] + \sum_{k \in P} \sum_{i \in K0_k} \sum_{\substack{j \in K_k \\ j \neq i}} c'_{ijk} . x(i,j,k) \tag{3.1}$$

Sujeito a:

$$\sum_{k \in P} \sum_{\substack{i \in K0_k \\ i \neq j}} x(i,j,k) = 1 \qquad \forall j \in T \tag{3.2}$$

$$\sum_{j \in K_k} x(0,j,k) \leq 1 \qquad \forall k \in P \tag{3.3}$$

$$\sum_{\substack{i \in K0_k \\ i \neq h}} x(i,h,k) - \sum_{\substack{j \in K0_k \\ j \neq h}} x(h,j,k) = 0 \qquad \forall k \in P, \forall h \in K_k \tag{3.4}$$

$$a(j) = a(i) + id(i,j) + \sum_{k \in P} [(c_{ijk} + p_{jk} + HV).x(i,j,k)] - HV \qquad \begin{matrix} \forall i \in T0 \\ \forall j \in T, j \neq i \end{matrix} \tag{3.5}$$

$$a(j) + e(j) - z(j) = d_j \qquad \forall j \in T \tag{3.6}$$

$$id(i,j) \geq 0, \quad a(i) \geq 0, \quad z(j) \geq 0, \quad e(j) \geq 0 \qquad \forall i \in T0, \forall j \in T$$

$$x(i,j,k) \in \{0,1\} \qquad \forall i \in T0, \forall j \in T0, \forall k \in P$$

onde:

P = conjunto de máquinas (ou processadores) disponíveis.

K_k = conjunto de tarefas executáveis pela máquina k. Não deverá conter a tarefa 0.

T = conjunto de tarefas a serem executadas ($T = K_1 \cup K_2 \cup .... \cup K_M$).

$K0_k = 0 \cup K_k$. conjunto de tarefas executáveis pela máquina k, incluindo a tarefa 0.

$T0 = 0 \cup T$. conjunto de tarefas a serem executadas, incluindo a tarefa 0.

d_j = data prevista (*due date*) para o término da tarefa j.

p_{jk} = tempo de processamento da tarefa j na máquina k.

c_{ijk} = tempo de troca da tarefa i para a j na máquina k (*changeover* ou *setup time*).

c'_{ijk} = custo de troca da tarefa i para a j na máquina k.

α_j = coeficiente de custo de adiantamento da tarefa j na função objetivo.

β_j = coeficiente de custo de atraso da tarefa j na função objetivo.

HV = limite superior de tempo (deve ser maior do que o maior $a(j)$ de qualquer tarefa em qualquer solução factível).

$x(i,j,k) = 1$ se a tarefa i é processada imediatamente antes da tarefa j na máquina k (i antecede j ou $i \rightarrow j$), ou 0 em caso contrário. Se i for nulo, a tarefa j é a primeira a ser executada; se j for nulo, a tarefa i é a última a ser executada.(variável binária)

$a(j)$ = instante de término da tarefa j (variável contínua)
$z(j)$ = atraso da tarefa j (variável contínua)
$e(j)$ = adiantamento da tarefa j (variável contínua)
$id(i,j)$ = tempo ocioso (*idle time*) entre a tarefa i e a tarefa j na máquina onde ocorrer a seqüência $i \rightarrow j$ (variável contínua).

Em relação ao modelo original de GUINET (1991), foram introduzidos os conjuntos $P,T,K_k,T0,K0_k$, que tornam mais fácil o manuseio, eliminando uma das máquinas da programação, ou proibindo uma determinada máquina de executar certas tarefas. A tarefa 0 (fictícia), citada acima, serve apenas para indicar que a tarefa seguinte é a primeira da seqüência, ou que a tarefa anterior é a última da seqüência na máquina. Além disso, foi introduzida a restrição (3. 3) que corrige uma falha da modelagem original, proibindo que exista mais de uma tarefa como primeira da seqüência em cada máquina, sem impor que todas as máquinas sejam utilizadas na solução.

A função objetivo utilizada no modelo **S** acima (3. 1) é certamente uma das mais completas entre as que já foram consideradas na literatura especializada (BAKER & SCUDDER, 1990) e com isto pretende satisfazer um grande número de aplicações. Ela visa a minimização conjunta de atrasos, adiantamentos e custos de troca entre tarefas.

A restrição (3. 2) determina que cada tarefa deve ser executada exatamente 1 vez e que só pode haver uma única tarefa antecessora. A restrição (3. 4) estabelece que cada tarefa deve ter um antecessor e um sucessor, mesmo que estes sejam a tarefa 0. Se existir uma 1ª tarefa não nula na máquina k, deverá haver uma tarefa sucessora (2ª). Se existir uma 2ª tarefa não nula na máquina k, deverá haver uma tarefa sucessora (3ª), etc.... Desta maneira, as restrições (3. 3) e (3. 4) garantem a existência de uma única seqüência na máquina k, iniciando e terminando com a tarefa nula. As restrições (3. 3) e (3. 4), não evitam que ocorra um ciclo como, por exemplo, $1 \rightarrow 2 \rightarrow 1$ ou $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$, supondo que as tarefas são representadas por números inteiros. No entanto, a restrição (3. 5) só permitirá tal situação se os tempos de troca e de processamento das tarefas do ciclo forem nulos.

A restrição (3. 5) determina que uma tarefa não pode se sobrepor a uma outra da mesma máquina. Os termos com a variável binária $x(i,j,k)$ e o parâmetro HV, fazem parte de um mecanismo que tornará a restrição sem efeito para o problema, se a tarefa i não for imediatamente anterior a j na máquina k. Se a tarefa i for imediatamente anterior a j na máquina k, os termos se transformarão em $c_{ik} + p_{ik}$. A restrição (3. 5) é uma adaptação da

restrição (20) de GUINET (1991), onde os tempos de troca foram feitos dependentes da máquina (o que é razoável, já que as máquinas são não-relacionadas) e a variável de folga da restrição original foi colocada explicitamente como $id(i,j)$. Porém, esta só será válida como tempo ocioso entre tarefas, se existir uma variável $x(i,j,k) = 1$, com os mesmos i, j e qualquer $k \in P$. Caso isto não aconteça, $id(i,j)$ cumprirá seu papel de variável de folga, isto é, seu valor não terá um significado físico e poderá ser ignorado. Finalmente, a restrição (3. 6) também original de GUINET (1991), sendo que a variável de folga da restrição foi definida como $e(j)$.

A variável $a(0)$, poderá ser feita igual a um valor fixo qualquer, significando o instante de início de execução das tarefas. É possível fazer uma distinção entre os instantes de início de operação entre as máquinas da planta, somando uma constante a todos os tempos iniciais de todas as tarefas em uma determinada máquina. Para considerar os termos constantes $a(0,A)$, $a(0,B)$,...$a(0,k)$ que são os instantes de disponibilidade inicial em cada máquina, estes são somados aos tempos de troca iniciais c_{0jk}, formando os novos parâmetros co_{ijk}. Isto é,

$$co_{ijk} = c_{ijk} \text{ para } i \in T\,(i \neq 0) \text{ e } co_{0jk} = c_{0jk} + a(0,k) \text{ para todo } j \in T;\ k \in P. \qquad (3.\ 7)$$

Exemplo:

Considere-se um exemplo com apenas 1 máquina e 2 tarefas. Os dados do exemplo são:

$P = \{A\}$; $K_A = T = \{1,2\}$; $HV = 100$; $a(0) = 0$

$$\left[c'_{ijA}\right] = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}; \quad \left[c_{ijA}\right] = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}; \quad \left[p_{jA}\right] = \begin{bmatrix} 3 \\ 2 \end{bmatrix}; \quad \left[d_j\right] = \begin{bmatrix} 3 \\ 4 \end{bmatrix}; \quad \left[\beta_j\right] = \begin{bmatrix} 1 \\ 1 \end{bmatrix}; \quad \left[\alpha_j\right] = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Desta forma, o problema no modelo **S** seria:

Minimize $z(1) + z(2)$

Sujeito a:

$$(3.\ 2) \quad \begin{cases} x(0,1,A) + x(2,1,A) = 1 \\ x(0,2,A) + x(1,2,A) = 1 \end{cases}$$

$$(3.\ 3) \quad \{x(0,1,A) + x(0,2,A) \leq 1$$

$$(3.\ 4) \quad \begin{cases} x(0,1,A) + x(2,1,A) - x(1,0,A) - x(1,2,A) = 0 \\ x(0,2,A) + x(1,2,A) - x(2,0,A) - x(2,1,A) = 0 \end{cases}$$

$$(3.5)\quad \begin{cases} a(1) = 0 + id(0,1) + f_{seq}(0,1,A) \\ a(1) = a(2) + id(2,1) + f_{seq}(2,1,A) \\ a(2) = 0 + id(0,2) + f_{seq}(0,2,A) \\ a(2) = a(1) + id(1,2) + f_{seq}(1,2,A) \end{cases}$$

$$(3.6)\quad \begin{cases} a(1) + e(1) - z(1) = 3 \\ a(2) + e(2) - z(2) = 4 \end{cases}$$

$$id(i,j) \geq 0, \quad a(i) \geq 0, \quad z(j) \geq 0, \quad e(j) \geq 0 \qquad \forall i \in T0, \forall j \in T$$
$$x(i,j,k) \in \{0,1\} \qquad \forall i \in T0, \forall j \in T0, \forall k \in P$$

onde $f_{seq}(i,j,k) = c_{ijk} + p_{jk}$ se a tarefa i for imediatamente anterior a j na máquina k, ou $f_{seq}(i,j,k) = -HV = -100$ em caso contrário. O exemplo acima possui a seguinte solução ótima com valor da função objetivo igual a 1:

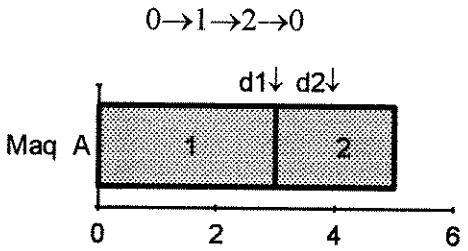


Fig 3. 1 - Exemplo sem divisão de tarefas

$$\begin{cases} x(0,1,A) = x(1,2,A) = x(2,0,A) = 1 \\ x(0,2,A) = x(2,1,A) = x(1,0,A) = 0 \\ a(1) = 3; z(1) = 0; e(1) = 0; \\ a(2) = 5; z(2) = 1; e(2) = 0; \\ id(0,1) = 0; id(1,2) = 0 \\ id(0,2) = 105; id(2,1) = 98 \end{cases}$$

Este exemplo serve para ilustrar a necessidade da restrição (3. 3), pois na ausência desta a seguinte solução, com valor da funcão objetivo igual a zero, seria factível e ótima:

$$0\to1\to0 \text{ e } 0\to2\to0 \begin{cases} x(0,1,A) = x(1,0,A) = x(0,2,A) = x(2,0,A) = 1 \\ x(1,2,A) = x(2,1,A) = 0 \\ a(1) = 3; z(1) = 0; e(1) = 0; id(0,1) = 0; id(1,2) = 99 \\ a(2) = 2; z(2) = 0; e(2) = 2; id(0,2) = 0; id(2,1) = 101 \end{cases}$$

É claro que a solução acima não pode ser implementada na prática, o que justifica a presença da restrição (3. 3) para torna-la infactível.

3.2 Divisão de tarefas através de programação linear

Muitos autores já analisaram o caso de máquinas paralelas sujeitas à preempção de tarefas (BAKER, 1974, LAWLER & LABETOULLE, 1978). Em máquinas paralelas, preempção significa que uma tarefa pode ter a sua execução interrompida antes do seu término e pode ser completada em uma outra máquina qualquer, ou mesmo poderá ser executada em

várias máquinas até o término de sua execução. Mas, a preempção exige que, a cada instante de tempo, a tarefa só esteja no máximo em uma máquina (não simultaneidade). No caso analisado aqui, esta limitação não se aplica, pois este é justamente o grande trunfo da divisão das tarefas, o processamento simultâneo em mais de uma máquina.

Enquanto a preempção de tarefas é admitida em muitos problemas clássicos, a divisão de uma ou mais tarefas entre as máquinas disponíveis é um assunto particularmente pouco estudado. A divisão de tarefas é considerada uma característica rara que depende da natureza particular do processo físico. SERAFINI & SPERANZA (1992) estudam um problema que se aproxima muito daquele definido no item 1.3, inclusive com divisão de tarefas via programação linear, mas não consideram adiantamentos de tarefas e nem tempos ociosos (o que eles chamam de tempos ociosos são, na realidade, tempos de troca). Também os autores utilizam uma modelagem especifica para a indústria têxtil com generalização difícil para outras indústrias.

LAWLER & LABETOULLE (1978), desenvolveram um modelo linear, capaz de minimizar o atraso máximo em um programa sujeito à preempção em máquinas paralelas não-relacionadas. Basta eliminar as restrições de não simultaneidade daquele modelo, para se chegar a um modelo de otimização linear capaz de fazer a divisão de tarefas em máquinas paralelas não-relacionadas. No entanto, uma generalização a partir do modelo **S** é mais conveniente devido, principalmente, à necessidade de considerar uma data prevista diferente para cada tarefa.

3.2.1 Modelo de programação inteira-mista com divisão de tarefas - modelo C

No modelo **S** as variáveis binárias $x(i,j,k)$ estão sub-utilizadas. Devido aos três índices, estas poderão se referir a uma fração de tarefa alocada à máquina k, pois mesmo que todas as tarefas $j \in T$ tenham uma fração alocada nesta máquina, as variáveis $x(i,j,k)$ poderão descrever o seqüenciamento destas frações corretamente.

Existe a necessidade de se definir uma nova variável $r(j,k)$, informando qual a fração (quantidade) da tarefa j alocada à máquina k e um novo parâmetro q_j informando qual a quantidade total da tarefa j. Supondo que o tempo de processamento da fração da tarefa j alocada à máquina k é proporcional a $r(j,k)$, constrói-se o seguinte modelo a partir do modelo **S**:

$$\textit{Minimize} \quad \sum_{j \in T} [\alpha_j . e(j) + \beta_j . z(j)] + \sum_{k \in P} \sum_{i \in K0_k} \sum_{\substack{j \in K_k \\ j \neq i}} c'_{ijk} . x(i,j,k) \qquad (3.8)$$

Sujeito a:

$$\sum_{k \in P} r(j,k) = q_j \qquad \forall j \in T \qquad (3.9)$$

$$r(j,k) \leq q_j . \sum_{\substack{i \in K0_k \\ i \neq j}} x(i,j,k) \qquad \forall k \in P, \forall j \in T \qquad (3.10)$$

$$\sum_{j \in K_k} x(0,j,k) \leq 1 \qquad \forall k \in P \qquad (3.11)$$

$$\sum_{\substack{i \in K0_k \\ i \neq h}} x(i,h,k) - \sum_{\substack{j \in K0_k \\ j \neq h}} x(h,j,k) = 0 \qquad \forall k \in P, \forall h \in K_k \qquad (3.12)$$

$$a(j,k) = a(i,k) + id(i,j,k) + c_{ijk} + s_{jk} . r(j,k) + [x(i,j,k) - 1] . HV \qquad \begin{matrix} \forall k \in P, \\ \forall i \in K0_k, \\ \forall j \in K_k, j \neq i \end{matrix} \qquad (3.13)$$

$$a(j,k) - d_j \leq z(j) \qquad \forall k \in P, \forall j \in K_k \qquad (3.14)$$

$$d_j - a(j,k) \leq e(j) \qquad \forall k \in P, \forall j \in K_k \qquad (3.15)$$

$$id(i,j,k) \geq 0, \quad r(j,k) \geq 0, \quad z(j) \geq 0, \quad e(j) \geq 0 \qquad \forall i \in T0, \forall j \in T, \forall k \in P$$

$$a(j,k) \geq 0, \quad x(i,j,k) \in \{0,1\} \qquad \forall i \in T0, \forall j \in T0, \forall k \in P$$

onde:

$x(i,j,k)$ = 1 se a fração da tarefa i é processada imediatamente antes da fração da tarefa j na máquina k (i antecede j ou $i \rightarrow j$), ou 0 em caso negativo. Se i for nulo, a fração da tarefa j é a primeira a ser executada; se j for nulo, a fração da tarefa i é a última a ser executada (variável binária).

$r(j,k)$ = fração (quantidade) da tarefa j alocada à máquina k (variável contínua).

$a(j,k)$ = instante de término da fração da tarefa j na máquina k (variável contínua).

$id(i,j,k)$ = tempo ocioso (*idle time*) entre a fração da tarefa i e a fração da tarefa j alocadas à máquina k (variável contínua).

q_j = quantidade total a ser produzida pela tarefa j

s_{jk} = tempo de processamento de uma quantidade unitária da tarefa j na máquina k.

Neste novo modelo **C** foram mantidos os conjuntos $P, T, K_k, T0, K0_k$, a função objetivo multi-critério e as restrições (3. 11) e (3. 12), que têm o mesmo significado das restrições (3. 3) e (3. 4) do modelo **S**. A restrição (3. 13) é uma adaptação que cumpre a necessidade de colocar um conjunto de restrições (3. 5) em cada uma das máquinas. A substituição de p_{jk} por $s_{jk} . r(j,k)$, deveria resultar em um produto de variáveis, se fosse mantido o mesmo formato das restrições (3. 5), mas neste caso é possível remover o produto, pois o último termo de (3. 13) tornará a restrição sem efeito no caso onde $x(i,j,k)$ é nulo. Analogamente, $id(i,j,k)$ só terá

significado de tempo ocioso entre frações de tarefas, se a variável correspondente $x(i,j,k)$ for igual a 1.

De forma semelhante ao que foi feito por SERAFINI & SPERANZA (1992), as restrições (3. 14) e (3. 15) estabelecem que o atraso ou adiantamento da tarefa j é o maior entre os atrasos ou adiantamentos de frações da tarefa j. Estas últimas poderiam facilmente ser alteradas juntamente com a função objetivo (3. 8), caso se desejasse penalizar o atraso/adiantamento de cada fração de tarefa em cada máquina.

As restrições (3. 9) e (3. 10), substituem a restrição (3. 2) do modelo **S**. Neste caso, não se deverá forçar a existência de uma fração de tarefa particular, mas será necessário apenas que as frações de tarefas produzam, no total, a quantidade requerida, o que leva à restrição (3. 9), originária de SERAFINI & SPERANZA, (1992). A restrição (3. 10) é apenas um artifício utilizado para forçar $r(j,k)$ a ser nulo, caso não exista uma fração da tarefa j alocada à máquina k, sem impor limites a $r(j,k)$ em caso contrário (apenas o limite já imposto por (3. 9)). Isto é necessário, sob pena de se chegar a soluções não implementáveis.

Caso se deseje dividir as tarefas em um número inteiro de quantidades fixas, basta fazer q_j igual ao número inteiro de padrões desejado para a tarefa j e definir as variáveis $r(j,k)$ como inteiras. Se uma determinada tarefa j não deve ser dividida, q_j pode ser 1 e $r(j,k)$ uma variável binária.

A variável $a(0,k)$, poderá ser feita igual a um valor fixo qualquer, significando o instante de início de execução das tarefas ou frações na máquina k, o que permite a distinção entre o instante de início de operação entre as máquinas da planta, não havendo a necessidade de todas estarem disponíveis no mesmo instante, quando se inicia a programação de tarefas.

3.2.2 Modelo com divisão de tarefas melhorado - modelo C'

Apesar do modelo **C** funcionar corretamente e, portanto, ser capaz de obter uma solução ótima implementável onde a divisão de tarefas é permitida, constatou-se que ele não é eficiente do ponto de vista computacional. Isto foi descoberto através da comparação do tempo de processamento e número de nós gerados, na solução de um exemplo com apenas 1 máquina disponível, quando se resolvem os modelos **S** e **C** usando o pacote AMPL (FOURER, GAY, & KERNIGHAN, 1993).

É claro que o modelo **C** tem, de fato, mais restrições e variáveis que o modelo **S**, mas no caso de apenas uma máquina disponível estas diferenças não justificam um aumento grande do tempo de processamento e nós gerados pelo método *Branch and Bound* usado pelo AMPL, como pode ser observado no primeiro exemplo do item 7.1.

Este efeito é atribuído às alterações introduzidas nas restrições que impõem limites às variáveis binárias. No caso do modelo **S**, a restrição (3. 2), que substitui (3. 9) e (3. 10), força a maioria das variáveis $x(i,j,k)$ para zero, como deve ocorrer em qualquer solução factível, mas no modelo **C** falta um mecanismo direto para isto.

Para forçar a maioria das variáveis binárias para zero as seguintes restrições podem ser definidas:

$$\sum_{\substack{i \in K0_k \\ i \neq h}} x(i,h,k) \leq 1 \qquad \forall k \in P, \forall h \in K_k \tag{3. 16}$$

$$\sum_{\substack{j \in K0_k \\ j \neq h}} x(h,j,k) \leq 1 \qquad \forall k \in P, \forall h \in K0_k \tag{3. 17}$$

Elas impõem que cada tarefa ou fração pode ter, no máximo, um antecessor e, no máximo, um sucessor. Nas restrições (3. 16) e (3. 17) acima as variáveis de folga terão seus valores completamente definidos:

$sla(h,k) = 0$ se existir uma fração de tarefa qualquer (mesmo que seja a tarefa 0), alocada à máquina k imediatamente anterior a h. Em caso contrário $sla(h,k) = 1$.

$slp(h,k) = 0$ se existir uma fração de tarefa qualquer (mesmo que seja a tarefa 0), alocada à máquina k imediatamente posterior a h. Em caso contrário $slp(h,k) = 1$.

Fazendo (3. 16) - (3. 17) com as variáveis de folga colocadas, chega-se a:

$$\sum_{\substack{i \in K0_k \\ i \neq h}} x(i,h,k) - \sum_{\substack{j \in K0_k \\ j \neq h}} x(h,j,k) + sla(h,k) - slp(h,k) = 0 \qquad \forall k \in P, \forall h \in K_k \tag{3. 18}$$

Substituindo (3. 12) em (3. 18), conclui-se que $sla(h,k) = slp(h,k)$ para quaisquer h,k. A restrição (3. 11) é idêntica a (3. 17) quando $h = 0$, portanto, é possível substituir as restrições (3. 11) e (3. 12) do modelo **C**, pelas restrições (3. 16) e (3. 17) definindo uma única variável de folga $sl(j,k)$. O novo modelo **C'** é:

$$\textit{Minimize} \qquad \sum_{j \in T} [\alpha_j e(j) + \beta_j z(j)] + \sum_{k \in P} \sum_{i \in K0_k} \sum_{\substack{j \in K_k \\ j \neq i}} c'_{ijk} . x(i,j,k) \tag{3. 19}$$

Sujeito a:

$$\sum_{k \in P} r(j,k) = q_j \qquad \forall j \in T \tag{3. 20}$$

$$r(j,k) + q_j.sl(j,k) \leq q_j. \qquad \forall k \in P, \forall j \in T \tag{3.21}$$

$$\sum_{\substack{i \in K0_k \\ i \neq j}} x(i,j,k) + sl(j,k) = 1 \qquad \forall k \in P, \forall j \in K_k \tag{3.22}$$

$$sl(i,k) + \sum_{\substack{j \in K0_k \\ j \neq i}} x(i,j,k) = 1 \qquad \forall k \in P, \forall i \in K0_k \tag{3.23}$$

$$a(j,k) = a(i,k) + id(i,j,k) + c_{ijk} + s_{jk}.r(j,k) + [x(i,j,k) - 1].HV \qquad \begin{matrix} \forall k \in P \\ \forall i \in K0_k, \\ \forall j \in K_k, j \neq i \end{matrix} \tag{3.24}$$

$$a(j,k) - d_j \leq z(j) \qquad \forall k \in P, \forall j \in K_k \tag{3.25}$$

$$d_j - a(j,k) \leq e(j) \qquad \forall k \in P, \forall j \in K_k \tag{3.26}$$

$$id(i,j,k) \geq 0, \quad r(j,k) \geq 0, \quad z(j) \geq 0, \quad e(j) \geq 0 \qquad \forall i \in T0, \forall j \in T, \forall k \in P$$

$$a(j,k) \geq 0, \quad sl(j,k) \geq 0, \quad x(i,j,k) \in \{0,1\} \qquad \forall i \in T0, \forall j \in T0, \forall k \in P$$

onde:

$sl(j,k)$ = variável auxiliar igual a zero quando uma fração da tarefa j está alocada à máquina k ou igual a 1 em caso contrário (tratada como variável contínua apesar de possuir valores binários).

A variável auxiliar $sl(j,k)$, também foi utilizada na restrição (3. 21) (que pode ser deduzida a partir de (3. 22) e (3. 10)) e com isto ajudou a reduzir o número de termos em relação à restrição original.

Uma vez demonstrada a equivalência do modelo **C** para **C'**, a equivalência de **C'** para **C** é óbvia, o que garante a plena equivalência dos dois modelos. A relaxação do modelo **C'** é melhor do que a de **C**, produzindo limitantes de melhor qualidade e maior eficiência computacional do método *Branch and Bound* utilizado pelo AMPL.

Este novo modelo conserva todo o potencial do modelo **C**, porém a introdução das restrições (3. 22) e (3. 23) torna o modelo com divisão de tarefas bastante mais eficiente computacionalmente, como é mostrado nos exemplos do capítulo 7.

4. Modelo com seqüenciamento e alocações fixas - modelo F.

Apesar dos aperfeiçoamentos introduzidos nas modelagens anteriores, o grande número de variáveis 0-1 presentes nos modelos de programação inteira-mista limita a sua utilização a problemas de pequenas dimensões, como será demonstrado computacionalmente mais adiante no item 7.1.

Analisando o modelo de programação inteira-mista **C'**, é possível idealizar a sua resolução em 2 fases: 1) seqüenciamento e alocação de frações de tarefas, feito pelas variáveis 0-1 e 2) definição de frações de tarefa, tempos ociosos, tempos de término das frações, atrasos e adiantamentos, feito pelas variáveis contínuas.

A alternativa que será empregada aqui é a utilização de um algoritmo de busca em árvore para a execução da fase 1, dispensando a resolução do modelo inteiro-misto. A busca, do tipo informada, será guiada por funções heurísticas de avaliação (ou limitantes) baseadas em um modelo onde as variáveis 0-1 podem ser descartadas. GAREY, TARJAN & WILFONG (1988), BAKER & SCUDDER (1990) e outros já empregaram este enfoque de decomposição em problemas de uma única máquina com custos de atraso e adiantamento. DAVIS & KANET (1993), utilizam o método *Branch and Bound* para a minimização de

atrasos e adiantamentos em uma única máquina de maneira bastante semelhante à proposta acima, embora não utilizem um modelo linear como o que é mostrado abaixo.

A partir do modelo **C** ou **C'**, as restrições relativas às variáveis 0-1 e termos auxiliares dependentes destas são removidos, o que fornece o modelo de Programação Linear (PL) **F** relativo à fase 2:

$$\textit{minimize} \quad \sum_{j \in T} [\beta_j . z(j) + \alpha_j . e(j) + \eta_j] \tag{4. 1}$$

Sujeito a:

$$a(j,k) - a(i,k) - id(j,k) - s_{jk} . r(j,k) = c_{ijk}; \quad \forall j \in T, \forall k \in M_j, \forall i \in A_{jk} \tag{4. 2}$$

$$\sum_{k \in M_j} r(j,k) = q_j; \quad \forall j \in T \tag{4. 3}$$

$$a(j,k) - z(j) \leq d_j; \quad \forall j \in T, \forall k \in M_j \tag{4. 4}$$

$$a(j,k) + e(j) \geq d_j; \quad \forall j \in T, \forall k \in M_j \tag{4. 5}$$

$$a(j,k) \geq 0, r(j,k) \geq 0, id(j,k) \geq 0, e(j) \geq 0, z(j) \geq 0; \quad \forall j \in T, \forall k \in M_j$$

onde:

P = conjunto de máquinas (ou processadores) disponíveis.

T = conjunto de tarefas a serem executadas.

M_j = conjunto de máquinas onde existe uma fração da tarefa j alocada.

A_{jk} = conjunto unitário contendo a tarefa i anterior a j na máquina k.

$a(j,k)$ = instante de término da fração da tarefa j na máquina k.

$r(j,k)$ = fração (quantidade) da tarefa j alocada à máquina k.

$id(j,k)$ = tempo ocioso anterior à execução da fração da tarefa j na máquina k. Note que é possível retirar o índice da tarefa anterior a j, pois esta dependerá do seqüenciamento decidido para a máquina na fase 1.

$z(j)$ = atraso da tarefa j.

$e(j)$ = adiantamento da tarefa j.

q_j = quantidade total a ser produzida pela tarefa j.

d_j = data prevista (*due date*) para o término da tarefa j.

α_j = coeficiente de custo de adiantamento da tarefa j na função objetivo.

β_j = coeficiente de custo de atraso da tarefa j na função objetivo.

s_{jk} = tempo de processamento de uma quantidade unitária da tarefa j na máquina k.

c_{ijk} = tempo de troca (*changeover*) entre a tarefa i e a tarefa j na máquina k.

η_j = total dos custos de troca para a execução da tarefa j (constante para uma determinada escolha de seqüenciamentos e alocações da tarefa j e anteriores).

No modelo acima os conjuntos M_j e A_{jk} são fixados na fase 1.

4.1 Resolução do sistema linear

O modelo acima, como um PL, pode ser resolvido pelos métodos tradicionais como

Simplex, Dual Simplex ou Simplex Revisado. Entretanto, apesar de dramaticamente menor, menos complexo e de resolução mais rápida do que o modelo de programação inteira-mista, o modelo acima representa ainda um PL relativamente grande. Em um exemplo com 100 tarefas em 6 máquinas, se todas as tarefas possuírem frações em todas as máquinas, o PL resultante do modelo **F** possuirá $3\times(6\times100) + 2\times100 = 2000$ variáveis e $3\times(6\times100) + 100 = 1900$ restrições. Além disso, o modelo deverá ser utilizado em uma busca em árvore calculando limitantes, o que significa que muitos problemas de diversos tamanhos devem ser resolvidos até o final da busca. A resolução eficiente e rápida do PL resultante do modelo **F** é de extrema importância para viabilizar a aplicação dos métodos e procedimentos desenvolvidos nos capítulos 5 e 6 a exemplos de porte médio em aplicações industriais.

Uma das principais contribuições deste trabalho de tese é o desenvolvimento de técnicas capazes de resolver o PL acima em um tempo bastante curto se comparado com os métodos tradicionais. Isto pode ser feito explorando certas características do modelo **F** que os métodos tradicionais não consideram, justamente porque são métodos de âmbito geral. As descrições detalhadas destas técnicas, que são um tanto extensas, foram colocadas no apêndice, mas basicamente são:

1) Adição de tarefas a uma solução parcial: utilizando um quadro Simplex Revisado para representar uma solução parcial (onde faltam tarefas), uma outra solução parcial (com tarefas diferentes) pode ser adicionada, chegando-se a um novo quadro contendo as duas soluções parciais a partir dos dois anteriores (Apêndice A.1.2).
2) Decomposição do modelo: Aproveitando a estrutura que se aproxima a uma bloco diagonal da matriz de bases B, aplica-se a decomposição de Dantzig-Wolfe ao modelo acima, gerando um problema principal formado pelas restrições (4. 2) e N (número de tarefas presentes no problema) subproblemas formados pelas restrições (4. 3), (4. 4) e (4. 5). Existe uma solução analítica capaz de resolver os subproblemas de maneira rápida e exata, sem a necessidade de quadros ou iterações. O problema principal é representado em um quadro Simplex Revisado (Apêndice A.1.3).

4.2 Representação das alocações e seqüenciamentos.

Com o modelo **F**, é possível deixar a decisão sobre que quantidades serão produzidas de cada tarefa em cada máquina, para um cálculo posterior, voltando, então, a atenção para o problema do seqüenciamento e alocação das tarefas. Ele pode ser esquematizado assim:

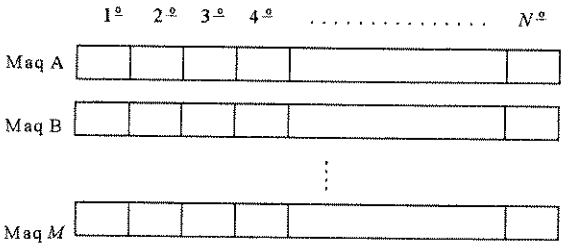


Fig. 4. 1 Gabarito de seqüências e alocações

No esquema acima, cada máquina terá N (= número de tarefas a serem executadas) posições para alocar frações de tarefas. Estas posições serão chamadas de *slots* virtuais, pois se referem apenas ao seqüenciamento de frações na máquina, uma vez que ainda não foi definida qual a quantidade a ser executada na máquina e, portanto, não sendo possível fixar ainda o início, duração e término de cada *slot* (isto será feito pelo modelo **F**). O *slot* mais à esquerda em uma máquina qualquer representará a primeira fração a ser executada; o *slot* contíguo será ocupado pela segunda fração a ser executada, e assim até o *slot* mais à direita, que deverá conter a última fração de tarefa a ser executada naquela máquina.

Para representar um programa qualquer, cada *slot* será preenchido com um número inteiro de 1 a N representando a tarefa que possui uma fração alocada naquela posição, ou 0 se a posição não estiver ocupada. Este último 0 servirá para indicar que o slot deverá ser pulado quando na determinação de c_{ijk} e c'_{ijk} e não deve ser confundido com a tarefa 0 utilizada nos modelos inteiro-mistos. Um esquema desse tipo será chamado de gabarito.

4.3 Programas de permutação.

Nos modelos inteiro-mistos, no gabarito da Fig. 4.1 e no modelo **F**, cada máquina pode ter um seqüenciamento diferente para as suas frações, o que implica em um número grande de combinações possíveis nos gabaritos. Este número de combinações será muito menor se apenas os programas de permutação forem considerados. Nestes, uma mesma tarefa sempre ocupa a mesma posição em todas as máquinas, isto é, se a tarefa j foi programada como a primeira da seqüência, suas frações estarão sempre nos primeiros *slots* de todas as máquinas. Se a tarefa j não estiver alocada à máquina k, o primeiro *slot* desta máquina deverá ser preenchido com 0.

Embora não seja possível garantir que os gabaritos ótimos dos programas completos correspondam sempre a programas de permutação, eles certamente serão uma boa escolha, pois são a melhor maneira de garantir que os tempos de término das frações componentes de

uma mesma tarefa estarão sempre próximos uns dos outros para todas as tarefas. Isto é importante em uma boa solução, pois o objetivo é minimizar atrasos e adiantamentos de cada tarefa.

As duas cartas de Gantt abaixo se referem a um mesmo exemplo de duas tarefas com frações em duas máquinas. As áreas mais claras são tempos de troca entre tarefas e as áreas mais escuras se referem à execução de frações.

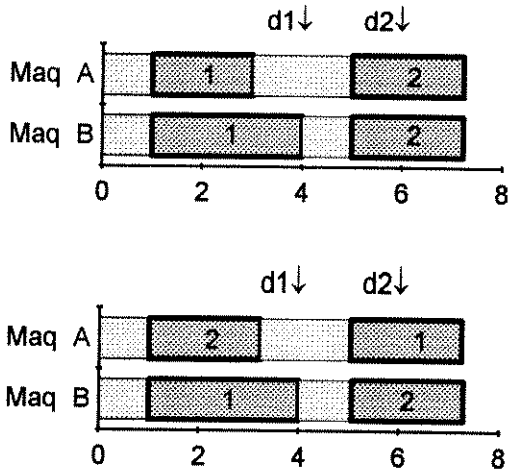


Fig. 4. 2 Programas com e sem permutação

O primeiro gráfico, é um programa de permutação e as diferenças entre os tempos de término das frações de uma mesma tarefa é pequeno.

O segundo gráfico não é um programa de permutação e as diferenças entre os tempos de término das frações é consideravelmente maior, o que se reflete como maiores atrasos e/ou adiantamentos das tarefas.

Nos exemplos do item 7.1, os programas ótimos do modelo inteiro-misto, que não se restringem aos programas de permutação, sempre estão nesta classe nos exemplos rodados, embora não seja garantido que sempre estarão.

Com os programas de permutação, pode-se considerar que todas as máquinas terão uma mesma seqüência, mesmo que algumas posições desta não estejam ocupadas em uma ou mais máquinas. Para definir em que máquinas estarão as frações de uma determinada posição da seqüência, existe a alocação da posição ou da tarefa que ocupa a posição. Isto torna o problema de programação em máquinas paralelas uma generalização do problema de uma única máquina. Neste último, só existe uma alocação possível para cada posição da seqüência, o que torna desnecessário defini-la.

4.4 Propriedades das soluções do modelo F

Com o problema modelado matematicamente, é possível deduzir algumas propriedades básicas que serão fundamentais para o cálculo das funções heurísticas de avaliação (ou limitantes) utilizados durante a busca em árvore desenvolvida nos capítulos 5 e 6.

4.4.1 Grupos, gabaritos e soluções

Definimos grupo de tarefas como um subconjunto qualquer de T. O grupo será chamado de grupo completo se possuir todas as tarefas do problema original, isto é, N tarefas. Um grupo incompleto possui menos do que N tarefas e o grupo complementar correspondente possui as tarefas do problema original não pertencentes ao grupo incompleto. O conceito de grupo é importante pois no método de busca em árvore uma solução é construida passo a passo, com a adição sucessiva de novas tarefas à seqüência daquelas que já estão definidas. A busca termina quando o grupo complementar torna-se vazio.

Chamamos de gabarito de um grupo de tarefas, não apenas ao seqüenciamento das tarefas dentro do grupo, mas também às alocações de suas frações às máquinas., Em outras palavras seqüenciar e alocar um determinado grupo é colocar as frações das suas tarefas no gabarito do item 4.1 acima. A definição do gabarito corresponde, assim, à fase 1 da solução do problema.

Uma solução para um determinado grupo, é a definição exata de tempos ociosos, tempos de término, atrasos e adiantamentos de cada fração de tarefa. A solução é encontrada durante a fase 2 de resolução do problema. Uma solução só pode ser definida depois de definido o gabarito do grupo e, para cada gabarito, haverá infinitas soluções distintas. O modelo **F** permite determinar a solução ótima sobre um dado gabarito, isto é, uma solução com função objetivo mínima possível para o gabarito dado.

É preciso algum cuidado para não confundir solução ótima de um grupo, solução ótima de um gabarito e gabarito ótimo de um grupo. Qualquer gabarito de grupo, seja ou não ótimo, possui pelo menos uma solução ótima que pode ser determinada pela resolução do PL correspondente ao modelo **F**. É possível que um mesmo gabarito tenha várias soluções ótimas. O gabarito ótimo do grupo é definido da seguinte maneira:

Se $\mathbf{g}^*$ é um gabarito ótimo de um grupo **G**, então para um outro gabarito **g** de **G**:

$$fo^*(\mathbf{G}, \mathbf{g}^*) \leq fo^*(\mathbf{G}, \mathbf{g}) \qquad (4.6)$$

onde:

$fo^*(\mathbf{G}, \mathbf{g}^*)$ = valor da função objetivo da solução ótima do grupo **G** no gabarito $\mathbf{g}^*$.

$fo^*(\mathbf{G}, \mathbf{g})$ = valor da função objetivo da solução ótima do grupo **G** no gabarito **g**.

A solução ótima do grupo é a solução ótima do gabarito ótimo do grupo. O sobrescrito * significa ótimo. Como nas soluções ótimas, poderá haver diversos gabaritos ótimos para um

mesmo grupo. O procedimento de busca *Branch & Bound* desenvolvido mais adiante, permite chegar a um gabarito ótimo do grupo completo e fornece a solução ótima deste gabarito, isto é, a solução ótima do grupo completo que possui a menor função objetivo possível de se obter para o problema original.

4.4.2 Parcelas da função objetivo

A função objetivo pode ser separada em duas parcelas, uma devido a atrasos e adiantamentos, que depende da solução do modelo **F**, e outra devido a custos de trocas, que não depende dela, mas apenas do grupo e do gabarito. A partir da função objetivo do modelo **F**, a função objetivo de uma solução qualquer de um grupo **G** em um gabarito **g** é:

$$fo(\mathbf{G}, \mathbf{g}) = \sum_{j \in T} [\beta_j . z(j) + \alpha_j . e(j)] \quad + \quad \sum_{j \in T} \eta_j \qquad (4.7)$$

Definindo,

$$fo^{ez}(\mathbf{G}, \mathbf{g}) = \sum_{j \in T} [\beta_j . z(j) + \alpha_j . e(j)] \qquad (4.8)$$

$$fo^{ch}(\mathbf{G}, \mathbf{g}) = \sum_{j \in T} \eta_j \qquad (4.9)$$

temos:

$$fo(\mathbf{G}, \mathbf{g}) = fo^{ez}(\mathbf{G}, \mathbf{g}) + fo^{ch}(\mathbf{G}, \mathbf{g}) \qquad (4.10)$$

As propriedades abaixo, são demonstradas para dois grupos, mas podem ser prontamente estendidas a um número qualquer de grupos não superior ao número de tarefas consideradas (N).

4.4.3 Acoplamento entre grupos

Sejam $\mathbf{G}_1$ e $\mathbf{G}_2$ grupos de tarefas seguidas com gabaritos $\mathbf{g}_1$ e $\mathbf{g}_2$. Se as tarefas em $\mathbf{G}_2$ forem todas diferentes das tarefas em $\mathbf{G}_1$ e os grupos forem colocados juntos no modelo de seqüenciamentos e alocações fixas (modelo **F**), devido à soma de termos para o cálculo da função objetivo, pode-se escrever:

$$fo^*(\mathbf{G}_1\mathbf{G}_2, \mathbf{g}_1\mathbf{g}_2) = fo(\mathbf{G}_1, \mathbf{g}_1) + fo(\mathbf{G}_2, \mathbf{g}_2) \qquad (4.11)$$

onde:

$fo^*(\mathbf{G}_1\mathbf{G}_2, \mathbf{g}_1\mathbf{g}_2)$ = valor ótimo da função objetivo avaliada no "grupo composto" $\mathbf{G}_1\mathbf{G}_2$ no "gabarito composto" $\mathbf{g}_1\mathbf{g}_2$.

$fo(\mathbf{G}_1, \mathbf{g}_1)$, $fo(\mathbf{G}_2, \mathbf{g}_2)$ = parcelas do valor ótimo da função objetivo para $\mathbf{G}_1$ e $\mathbf{G}_2$ nos gabaritos $\mathbf{g}_1$ e $\mathbf{g}_2$, respectivamente.

Observe que:

$$fo^{ez}(\mathbf{G}_1, \mathbf{g}_1) \geq fo^{ez*}(\mathbf{G}_1, \mathbf{g}_1) \quad \text{e} \quad fo^{ez}(\mathbf{G}_2, \mathbf{g}_2) \geq fo^{ez*}(\mathbf{G}_2, \mathbf{g}_2) \quad (4.12)$$

pois $fo^{ez*}(\mathbf{G}_1, \mathbf{g}_1)$ e $fo^{ez*}(\mathbf{G}_2, \mathbf{g}_2)$ são relativos às soluções ótimas de seus respectivos grupos e gabaritos quando considerados sozinhos no modelo **F**. Estes últimos resultados podem ser estendidos às funções objetivo completas, desde que os custos de troca na presença do outro grupo e os custos de troca com o grupo sozinho sejam idênticos, para manter $fo^{ch}(\mathbf{G}_1, \mathbf{g}_1)$ e $fo^{ch}(\mathbf{G}_2, \mathbf{g}_2)$ inalterados. O efeito causado na solução de um grupo devido à presença de outros grupos, será chamado de acoplamento entre grupos.

Na figura abaixo, ilustra-se o efeito acima, em um problema com 2 máquinas e 3 tarefas:

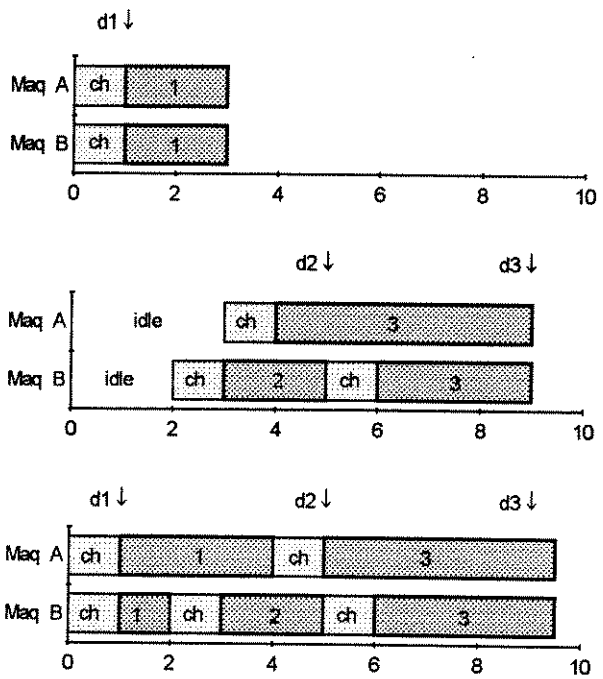


Fig. 4. 3 Exemplo de grupos acoplados

Considere $\mathbf{G}_1 = \{1\}$ e $\mathbf{G}_2 = \{2,3\}$ com os gabaritos definidos na figura ao lado. Os dois primeiros gráficos são as cartas de Gantt das soluções ótimas de $\mathbf{G}_1$ e $\mathbf{G}_2$, nos seus gabaritos, quando colocados sozinhos no modelo **F**.

Quando $\mathbf{G}_1$ e $\mathbf{G}_2$ são colocados juntos, são forçados a se deslocar de suas soluções ótimas, como é mostrado no 3º gráfico. Observe que aumentou o atraso da tarefa 1 em $\mathbf{G}_1$ e também a tarefa 3 em $\mathbf{G}_2$ passou a ter um pequeno atraso, mesmo com as novas soluções contidas na solução ótima de $\mathbf{G}_1\mathbf{G}_2$.

Quando os grupos, em seus respectivos gabaritos e soluções, podem ser colocados juntos sem que as suas funções objetivo se desviem dos valores ótimos obtidos com cada grupo sozinho, os grupos são ditos mutuamente desacoplados. Portanto, define-se que se os grupos $\mathbf{G}_1$ e $\mathbf{G}_2$ nos gabaritos $\mathbf{g}_1$ e $\mathbf{g}_2$ respectivamente, são mutuamente desacoplados, ou simplesmente desacoplados, então:

$$fo^{ez}(\mathbf{G}_1, \mathbf{g}_1) = fo^{ez*}(\mathbf{G}_1, \mathbf{g}_1) \quad \text{e} \quad fo^{ez}(\mathbf{G}_2, \mathbf{g}_2) = fo^{ez*}(\mathbf{G}_2, \mathbf{g}_2) \quad (4.13)$$

Se os custos de troca na presença do outro grupo e os custos de troca com o grupo sozinho forem idênticos:

$$fo^*(\mathbf{G_1G_2}, \mathbf{g_1g_2}) = fo^*(\mathbf{G_1}, \mathbf{g_1}) + fo^*(\mathbf{G_2}, \mathbf{g_2}) \quad (4.\ 14)$$

Nas três figuras abaixo, exemplifica-se o desacoplamento entre grupos utilizando um exemplo que se aproxima bastante do anterior:

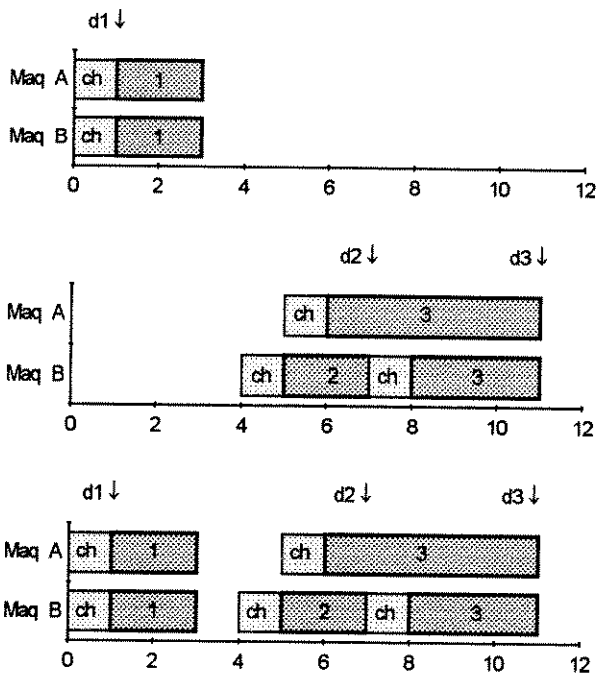


Fig. 4. 4 Exemplo de grupos desacoplados

Nos gráficos ao lado, os grupos e gabaritos são idênticos aos do 1º exemplo, exceto pelas datas previstas das tarefas 2 e 3 do grupo $\mathbf{G_2}$ que tiveram 2 unidades de tempo adicionadas.

A solução ótima de $\mathbf{G_1}$ isolado se "encaixa" nos tempos ociosos da solução ótima de $\mathbf{G_2}$ isolado, sem ter que ser modificada. Como na solução do grupo composto as soluções de $\mathbf{G_1}$ e $\mathbf{G_2}$ são idênticas às obtidas com estes grupos sozinhos (exceto pelos tempos ociosos de $\mathbf{G_2}$), pode-se afirmar que a função objetivo do grupo composto é a soma das funções objetivo ótimas de seus grupos componentes, dado que não se penalizam tempos ociosos.

No exemplo apresentado acima o desacoplamento é óbvio porque os grupos não se tocam. Entretanto, existem casos em que a presença do outro grupo força uma mudança na solução obtida com o grupo sozinho e mesmo assim os grupos podem ser considerados desacoplados, basta que a nova solução do grupo em que a mudança ocorreu tenha a mesma função objetivo ótima.

4.4.4 Propriedade da substituição de tempos de troca entre tarefas.

Seja **G** um grupo de tarefas no gabarito **g** e $fo^0{}^*(\mathbf{G}, \mathbf{g})$ o valor da função objetivo na solução ótima de **G** em **g**. Seja $c^0{}_{ijk}$ o tempo de troca entre as frações de tarefas i e j na máquina k. A substituição de $c^0{}_{ijk}$ por $c^1{}_{ijk}$ tal que $c^1{}_{ijk} \leq c^0{}_{ijk}$ implica em um novo valor da função objetivo ótima $fo^1{}^*(\mathbf{G}, \mathbf{g})$ tal que $fo^1{}^*(\mathbf{G}, \mathbf{g}) \leq fo^0{}^*(\mathbf{G}, \mathbf{g})$.

Prova:

1) Parcela $fo^{ez}(\mathbf{G})$

Se c^1_{ijk} é menor ou igual a c^0_{ijk}, então, após a substituição, a solução original pode conservar todos os tempos de término originais, isto é, o valor da função objetivo não se altera na nova solução, como é mostrado na figura abaixo que mostra apenas as duas frações afetadas pela substituição.

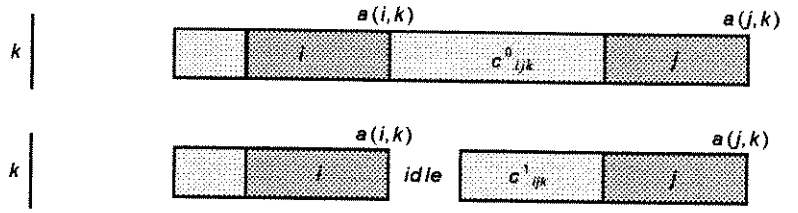


Fig. 4. 5 Substituição de tempos de troca

Escrevendo a restrição (4. 2) do modelo **F** para a fração *j* da máquina *k*, antes e após a substituição, tem-se:

$$a(j,k) - a(i,k) - id^0(j,k) - s_{jk}.r(j,k) = c^0{}_{ijk} \qquad (4.\ 15)$$

$$a(j,k) - a(i,k) - id^1(j,k) - s_{jk}.r(j,k) = c^1{}_{ijk} \qquad (4.\ 16)$$

onde $id^0(j,k)$ e $id^1(j,k)$ se referem aos tempo ocioso original e após a substituição respectivamente. Se as outras variáveis e parâmetros não se alteram, a subtração (4. 15) - (4. 16) fornece:

$$id^1(j,k) = id^0(j,k) + c^0{}_{ijk} - c^1{}_{ijk} \qquad (4.\ 17)$$

Como $id^0(j,k) \geq 0$ e $c^1_{ijk} \leq c^0_{ijk}$, (4. 17) implica em $id^1(j,k) \geq 0$, isto é, existirá sempre uma solução factível com os mesmos tempos de término e mesmo valor da função objetivo da solução original. Porém, uma restrição e uma variável do PL foram alterados e não é mais possível garantir que a nova solução é ótima, isto é, uma nova otimização reduzirá o valor da função objetivo obtida imediatamente após a substituição.

Desta forma conclui-se que:

$$fo^{1ez*}(\mathbf{G}, \mathbf{g}) \leq fo^{0ez*}(\mathbf{G}, \mathbf{g}) \qquad (4.\ 18)$$

2) Parcela $fo^{ch}(\mathbf{G}, \mathbf{g})$

Esta não se altera, pois não depende da solução do grupo **G** no gabarito **g.**

$$fo^{1ch*}(\mathbf{G}, \mathbf{g}) = fo^{0ch*}(\mathbf{G}, \mathbf{g}) \qquad (4.\ 19)$$

Juntando os itens 1) e 2) acima:

$$fo^1*(\mathbf{G}, \mathbf{g}) \leq fo^0*(\mathbf{G}, \mathbf{g}) \qquad (4.\ 20)$$

A propriedade também pode ser usada no sentido inverso, isto é, a substituição de c^0_{ijk} por c^1_{ijk} tal que $c^1_{ijk} \geq c^0_{ijk}$ implica em um novo valor da função objetivo ótima $fo^1*(\mathbf{G}, \mathbf{g})$ tal que $fo^1*(\mathbf{G}, \mathbf{g}) \geq fo^0*(\mathbf{G}, \mathbf{g})$.

Prova: Percorrer a demonstração acima no sentido inverso.

4.4.5 Propriedade da substituição de custos de troca entre tarefas.

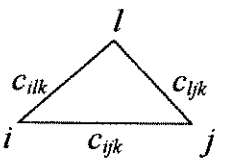
A substituição de c'^0_{ijk} por c'^1_{ijk} tal que $c'^1_{ijk} = c'^0_{ijk} + h$ implica que o novo valor da função objetivo ótima $fo^1*(\mathbf{G}, \mathbf{g})$ será tal que $fo^1*(\mathbf{G}, \mathbf{g}) = fo^0*(\mathbf{G}, \mathbf{g}) + h$.

Prova: Decorre imediatamente do fato da parcela $fo^{ez}(\mathbf{G}, \mathbf{g})$ não se alterar neste caso e da parcela $fo^{ch}(\mathbf{G}, \mathbf{g})$ ser formada pelo somatório de todos os custos de trocas envolvidos.

4.4.6 Desigualdade triangular dos tempos e custos de troca

A desigualdade triangular de tempos e custos de troca é uma hipótese utilizada para tornar o problema "bem comportado", isto é, mais fácil de se encontrar boas soluções. Isto ficará mais claro nos próximos itens, quando algumas propriedades do problema exigirão o uso desta hipótese.

Seja l uma tarefa do grupo $\mathbf{G}_2$ com fração na máquina k e i,j tarefas do grupo $\mathbf{G}_1$ que também possuem frações na máquina k. Desigualdade triangular entre tempos de troca, significa que:



$$c_{ijk} \leq c_{ilk} + c_{ljk} \qquad (4.\ 21)$$

Esta hipótese é aparentemente respeitada pela maioria dos processos no mundo real. Além disso, se existir um tempo de troca que desrespeita a desigualdade triangular, então este tempo poderá ser substituído pela menor soma de dois outros que levem de i a j, o que será menor que o tempo de troca direto. O mesmo pode ser dito em relação aos custos de troca. Porém, é possível supor que o tempo de troca direto, mesmo sendo maior, possui um pequeno custo e poderá ser empregado na prática, daí a necessidade de informar se a desigualdade triangular é ou não respeitada em um problema específico.

4.4.7 Propriedade do acoplamento entre grupos de tarefas intercalados

Com a suposição da desigualdade triangular e limitando os custos de troca a serem independentes da seqüência de tarefas, é possível estender as propriedades do acoplamento entre grupos ao caso em que as tarefas de um grupo se misturam às tarefas do outro. Isto é, poderá haver frações de tarefas de $\mathbf{G}_2$ entre duas frações de $\mathbf{G}_1$, e vice-versa, embora o seqüenciamento relativo às tarefas do grupo nos gabaritos de $\mathbf{G}_1$ e $\mathbf{G}_2$ não possa ser alterado.

A necessidade de impor que os custos de troca sejam independentes da seqüência se justifica porque a parcela $fo^{ch}(\mathbf{G}, \mathbf{g})$ não deve se alterar ou deve apresentar alterações no mesmo sentido de $fo^{ez}(\mathbf{G}, \mathbf{g})$. Como se deseja que as intercalações possam ocorrer simultaneamente com vários grupos, o melhor é impor a limitação de custos de troca independentes da seqüência.

PROPOSIÇÃO: Se a desigualdade triangular de tempos de troca entre grupos se verificar e os custos de troca do grupo $\mathbf{G}_1$ forem independentes da seqüência, a função objetivo ótima de $\mathbf{G}_1$ em $\mathbf{g}_1$, $fo^*(\mathbf{G}_1, \mathbf{g}_1)$, não será diminuída se tarefas de um outro grupo $\mathbf{G}_2$ forem colocadas entre as tarefas de $\mathbf{G}_1$. Isto é, $fo(\mathbf{G}_1, \mathbf{g}_1) \geq fo^*(\mathbf{G}_1, \mathbf{g}_1)$.

Prova: Basta fazer a demonstração para uma única fração de $\mathbf{G}_2$ entre frações de $\mathbf{G}_1$, que o caso com várias frações ficará evidente.

Em um primeiro momento o efeito de acoplamento entre grupos descrito no item 4.4.3 é desprezado para isolar o efeito da intercalação. Seja l uma tarefa de $\mathbf{G}_2$ com alocação na máquina k e i,j tarefas de $\mathbf{G}_1$ com alocação na mesma máquina k. A colocação da fração de l entre as frações i e j, implica em substituir o tempo de troca original c^0_{ijk} pelo equivalente a um tempo de troca c^1_{ijk} tal que:

$$c^1_{ijk} = c_{ilk} + s_{lk}.r(l, k) + c_{ljk} \qquad (4.\ 22)$$

Pela desigualdade triangular, para qualquer valor não negativo de $r(l, k)$, tem-se:

$$c^0_{ijk} \leq c^1_{ijk} = c_{ilk} + s_{lk}.r(l, k) + c_{ljk} \qquad (4.\ 23)$$

A propriedade da substituição de tempos de troca garante que, nestes casos, o valor da função objetivo após a substituição $fo^{ez}(\mathbf{G}_1, \mathbf{g}_1)$, será maior ou igual ao valor da função objetivo original $fo^{ez*}(\mathbf{G}_1, \mathbf{g}_1)$. Isto é:

$$fo^{ez}(\mathbf{G}_1, \mathbf{g}_1) \geq fo^{ez*}(\mathbf{G}_1, \mathbf{g}_1) \qquad (4.\ 24)$$

Uma vez que o efeito de acoplamento produz precisamente o mesmo efeito acima e como, neste caso, a parcela $fo^{ch}(\mathbf{G}_1, \mathbf{g}_1)$ não se altera, a conclusão se estende até a função objetivo completa:

$$fo(\mathbf{G}_1, \mathbf{g}_1) \geq fo^*(\mathbf{G}_1, \mathbf{g}_1) \qquad (4.25)$$

Observe que a desigualdade triangular é exigida entre duas tarefas de $\mathbf{G}_1$ e uma de $\mathbf{G}_2$, ou o oposto, para haver simetria. A rigor, esta não é necessária entre as tarefas pertencentes a $\mathbf{G}_1$ e nem entre as de $\mathbf{G}_2$. Entretanto, a desigualdade triangular entre as tarefas de um mesmo grupo possibilita a remoção de um efeito indesejado que é o aparecimento de frações nulas na solução ótima de um grupo ótimo, como é mostrado no item seguinte.

4.4.8 Propriedade da remoção de frações nulas

Uma fração nula ($r(j,k) = 0$), costuma ser indesejada em soluções finais, pois corresponde a dois tempos/custos de troca seguidos sem o processamento de uma fração de tarefa entre eles. Como o bom senso sugere, o custo e o tempo de troca de uma fração nula devem ser removidos para corrigir o efeito do seu aparecimento. Isto melhora o valor da função objetivo, desde que a desigualdade triangular seja respeitada.

PROPOSIÇÃO: Se a desigualdade triangular se verificar para tempos e custos de troca entre tarefas de um grupo **G**, a remoção de uma fração do gabarito **g** correspondente a uma fração nula em uma solução ótima de **G** em **g**, produzirá um gabarito **g'** tal que, $fo^*(\mathbf{G}, \mathbf{g}') \leq fo^*(\mathbf{G}, \mathbf{g})$.

Prova:

1) Parcela $fo^{ez}*(\mathbf{G}, \mathbf{g})$

Se a fração nula correspondente à tarefa l alocada à máquina k, não é a mais adiantada nem a mais atrasada dentre as frações de l alocadas às demais máquinas, a fração nula não determina o atraso nem o adiantamento de l. Neste caso, a remoção da fração nula não pode influenciar diretamente a função objetivo. A remoção corresponde à troca $c_{ilk} + c_{ljk} \rightarrow c_{ijk}$. Pela desigualdade triangular, $c_{ilk} + c_{ljk} \geq c_{ijk}$ e pela propriedade da substituição de tempos de troca, $fo^{ez}*(\mathbf{G}, \mathbf{g}') \leq fo^{ez}*(\mathbf{G}, \mathbf{g})$.

Se a fração nula corresponder à mais adiantada ou à mais atrasada dentre as frações de l, a remoção causará uma redução imediata no atraso ou adiantamento de l e,

conseqüentemente, em $fo^{ez*}(\mathbf{G}, \mathbf{g'})$ em relação a $fo^{ez*}(\mathbf{G}, \mathbf{g})$. A este efeito se soma o efeito demonstrado no parágrafo anterior, causando uma redução ainda maior em $fo^{ez*}(\mathbf{G}, \mathbf{g'})$.

2) Parcela $fo^{ch*}(\mathbf{G}, \mathbf{g})$

A desigualdade triangular entre custos de troca, $c'_{ilk} + c'_{ljk} \geq c'_{ijk}$, pode ser colocada como $c'_{ilk} + c'_{ljk} = c'_{ijk} + h$, onde $h \geq 0$. Pela propriedade da substituição dos custos de troca, $fo^{ch*}(\mathbf{G}, \mathbf{g}) = fo^{ch*}(\mathbf{G}, \mathbf{g'}) + h$, que é equivalente a $fo^{ch*}(\mathbf{G}, \mathbf{g}) \geq fo^{ch*}(\mathbf{G}, \mathbf{g'})$.

Combinando os itens 1) e 2) acima:

$$fo^{*}(\mathbf{G}, \mathbf{g'}) \leq fo^{*}(\mathbf{G}, \mathbf{g}) \qquad (4.26)$$

Corolário 1: Se a desigualdade triangular for válida, sempre haverá um gabarito ótimo cuja solução ótima não apresenta frações nulas.

Prova: Seja um grupo **G** que possui um gabarito ótimo **g*** tal que a solução ótima de **G** em **g*** tem frações nulas. Pela propriedade da remoção de frações nulas, o gabarito **g*'** (**g*** com as frações nulas removidas) não poderá apresentar uma solução ótima cujo valor da função objetivo é maior que $fo^{*}(\mathbf{G}, \mathbf{g^{*}})$. Portanto, **g*'** também será um gabarito ótimo sem que sua solução ótima apresente frações nulas.

Corolário 2: Se a desigualdade triangular for válida e os custos de troca forem não-nulos, a solução ótima de um gabarito ótimo jamais apresentará frações nulas.

Prova: Suponha, por absurdo, que existe um gabarito ótimo cuja solução ótima apresenta frações nulas. Pelo corolário anterior, a remoção das frações nulas gera um novo gabarito ótimo cuja solução ótima não tem frações nulas. Entretanto, a remoção das frações nulas representa uma diminuição de custos de troca, se estes são não-nulos. Portanto, o gabarito sem frações nulas possui um menor valor da função objetivo ótima, o que significa que o gabarito com frações nulas não é ótimo.

5. Busca em árvore

Com o modelo de seqüenciamentos e alocações fixas (modelo **F**), o problema de obter uma solução a partir de um grupo, está resolvido e de forma ótima. Resta então, a obtenção de um gabarito para o grupo cuja solução ótima se aproxime o mais possível da solução ótima do grupo. O procedimento proposto aqui para resolver este segundo problema, é a busca em árvore do tipo *Branch and Bound* para chegar a um programa ótimo, ou *Filtered Beam Search* para chegar a um programa cuja função objetivo da solução ótima se aproxima do menor valor possível.

Considerando somente os programas de permutação, existirão apenas duas decisões a serem tomadas; seqüenciamento global e alocação das tarefas às máquinas disponíveis. Partindo de um programa vazio, o caminho natural a ser seguido, é o preenchimento simultâneo de todos os *slots* em uma determinada posição da seqüência (nível). Na Fig 5.1 abaixo, está representada uma arborização que se inicia com os primeiros *slots* da seqüência e prossegue linearmente até os últimos, embora pudesse iniciar pelos últimos. Desta maneira tem-se uma árvore uniforme de profundidade N, com um grau de ramificação maior nos primeiros nós e menor próximo às folhas (PEARL, 1984).

A cada nível, ambas as decisões (seqüenciamento e alocação) são tomadas simultaneamente, o que permite chegar a qualquer combinação possível envolvendo tais decisões. Em cada nó da árvore, o programa parcial relativo àquelas posições na seqüência pode ser representado por apenas dois números. O primeiro representa a tarefa que foi designada para a posição da seqüência e o número inteiro entre parênteses representa a alocação decidida para aquela tarefa. Este último poderá variar entre 1 e 2^M-1 (binário de M bits), onde M é o número de máquinas disponíveis. Quando colocado no formato binário, cada dígito informa se a máquina correspondente recebe (1) ou não (0) uma fração da tarefa.

Um programa completo corresponde a uma cadeia de N nós que inicia no nó raiz.

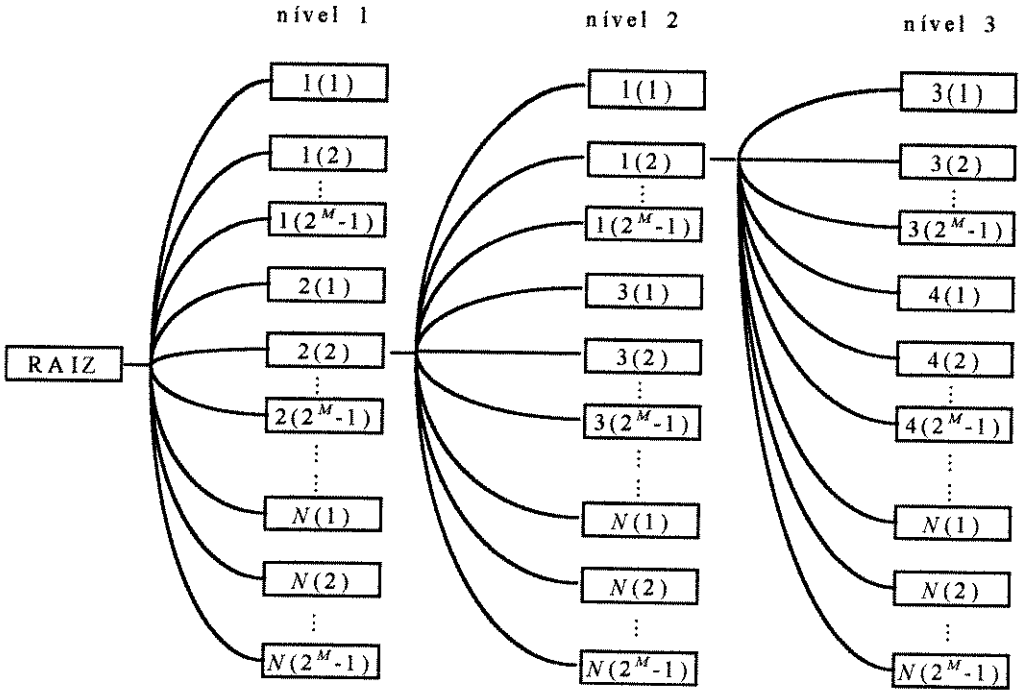


Fig 5. 1 Árvore de Busca

A figura acima representa apenas uns poucos nós da árvore em somente três posições da seqüência, pois a árvore completa é muito grande mesmo em pequenos exemplos. A árvore de busca completa corresponde à enumeração de todas as seqüências e alocações possíveis.

5.1 Busca Branch and Bound.

Uma forma segura de se chegar ao gabarito ótimo do problema é enumerar e testar todas as combinações possíveis ou, de forma equivalente, examinar todos os nós de nível N na árvore de busca. O nó que apresentar a menor medida de qualidade, ou valor da função objetivo da solução ótima, corresponde ao gabarito ótimo procurado.

O método *Branch and Bound* permite simplificar a enumeração, eliminando ramos inteiros da árvore que jamais levarão ao gabarito ótimo. Isto pode ser feito com as funções heurísticas de avaliação ou limitantes, que fornecem uma estimativa da medida de qualidade possível de se obter a partir de um nó qualquer da árvore (*bounding procedure*). O limitante inferior é uma função heurística de avaliação que garante um limite mínimo para a menor medida de qualidade possível de se obter a partir do nó analisado. Isto é, se $Z(n)$ é a menor medida de qualidade que pode ser obtida por um nó descendente do nó n e $b(n)$ é um limitante inferior calculado em n, então $b(n) \leq Z(n)$.

Suponhamos que em um determinado estágio da busca seja conhecido um nó terminal (nível N) cuja medida de qualidade seja Z. Qualquer nó de nível menor que apresentar um limitante inferior b tal que $b > Z$, jamais poderá levar a um nó terminal com medida de qualidade menor ou igual a Z. Desta forma é possível simplificar grandemente a busca, pois o conjunto inteiro de possíveis combinações pode ser enumerado implicitamente (BAKER, 1974).

O gabarito completo ou nó terminal usado nas comparações que permitem a eliminação de alguns ramos da árvore de busca (gabarito de tentativa), pode ser obtido por um procedimento heurístico ou pode ser encontrado no curso da busca e atualizado sempre que um nó de nível N com uma medida de qualidade menor surgir.

Uma tática bem conhecida utilizada para percorrer a árvore é chamada de *jumptracking* (BAKER, 1974 e BLAZEWICZ et al, 1993) ou *best-first* (OW & MORTON, 1988). Ela consiste em procurar na árvore o nó de nível inferior a N com o menor limitante inferior e expandi-lo (*branching*) gerando novos nós descendentes. Depois que todos os novos nós tiverem sido avaliados pelo limitante inferior, o processo é repetido. Quando não existirem mais nós de nível inferior a N com limitante inferior menor do que a medida de qualidade do gabarito de tentativa atual, a busca pode ser terminada, pois não será possível encontrar nós terminais melhores.

A tática de *jumptracking*, utilizada nesta tese, pulará frequentemente entre os vários ramos abertos na árvore e deverá manter todos os nós gerados durante a busca. Apenas os nós com limitante inferior maior do que a medida de qualidade do gabarito de tentativa atual serão ignorados, o que significa uma exigência considerável de espaço de armazenamento. Quanto maior (ou mais forte) for o limitante inferior utilizado durante a busca, menor será o número de

retornos a níveis inferiores (*backtracks*), menor o espaço de armazenagem necessário e maior a velocidade da busca (BAKER, 1974)

5.1.1 Limitante Inferior 1

Seja $\mathbf{G_1G_2}$ um grupo completo (com todas as tarefas a serem programadas), $\mathbf{G_1}$ um grupo incompleto associado a um nó da árvore a ser avaliado e $\mathbf{G_2}$ o grupo complementar de $\mathbf{G_1}$ e $\mathbf{g_1g_2}$, $\mathbf{g_1}$ e $\mathbf{g_2}$ os gabaritos respectivos. A expressão (4.11) do acoplamento entre grupos, permite escrever:

$$fo^*(\mathbf{G_1G_2}, \mathbf{g_1g_2}) = fo(\mathbf{G_1}, \mathbf{g_1}) + fo(\mathbf{G_2}, \mathbf{g_2}) \qquad (5.1)$$

Um limitante inferior à solução ótima $fo^*(\mathbf{G_1G_2}, \mathbf{g_1g_2})$, jamais poderá ser superior ao menor valor possível com $\mathbf{g_1}$ fixo e $\mathbf{g_2}$ livre. Como o gabarito para $\mathbf{G_2}$ ($\mathbf{g_2}$) é desconhecido, assim como suas influências sobre $\mathbf{G_1}$, uma boa opção para avaliar a primeira parcela da expressão acima é o cálculo de $fo^{ez*}(\mathbf{G_1}, \mathbf{g_1})$, pois pelas propriedades de acoplamento entre grupos,

$$fo^{ez*}(\mathbf{G_1}, \mathbf{g_1}) \leq fo^{ez}(\mathbf{G_1}, \mathbf{g_1}) \qquad (5.2)$$

Como $\mathbf{G_1}$ é o primeiro grupo, as suas tarefas nunca serão intercaladas com tarefas de outros grupos, portanto, nunca haverá diferenças entre $fo^{ch*}(\mathbf{G_1}, \mathbf{g_1})$ e $fo^{ch}(\mathbf{G_1}, \mathbf{g_1})$, permitindo estender a expressão (5. 2) até a função objetivo completa sem a necessidade de impor a desigualdade triangular entre tempos de troca ou que os custos de troca sejam independentes da seqüência.

$$fo^*(\mathbf{G_1}, \mathbf{g_1}) \leq fo(\mathbf{G_1}, \mathbf{g_1}) \qquad (5.3)$$

A avaliação da parcela $fo(\mathbf{G_2}, \mathbf{g_2})$, será feita de forma relaxada, pois o gabarito de $\mathbf{G_2}$ não é conhecido no momento da avaliação. Uma alternativa é avaliar apenas os custos de troca, alocando as tarefas de $\mathbf{G_2}$ unicamente na máquina com o menor custo de troca para cada tarefa. Desta forma, definimos:

$$LI1 = fo^*(\mathbf{G_1}, \mathbf{g_1}) + \sum_{j \in \mathbf{G_2}} \min_{\substack{i \in I_k \\ i \neq j \\ k \in P}} (c'_{ijk}) \qquad (5.4)$$

onde:

$I_k = \mathbf{G_2} \cup \{$última tarefa de $\mathbf{G_1}$ com uma fração alocada em $k\}$

Para facilitar o cálculo, se a tarefa j não puder ser executada na máquina k ($j \notin K_k$), deve-se fazer seus custos infinitos nesta máquina, o que provocará sempre a sua rejeição. O limitante LI1 não exige que a desigualdade triangular de tempos ou custos de troca seja

obedecida pelas tarefas do grupo completo $\mathbf{G_1G_2}$, assim como os custos de troca entre tarefas podem ser dependentes da seqüência.

5.1.2 Limitante Inferior 2

É possível estabelecer um limitante inferior de melhor qualidade considerando o acoplamento entre $\mathbf{G_1}$ e uma das tarefas de $\mathbf{G_2}$. Seja $\mathbf{G_1JG_2}$ um grupo completo, $\mathbf{G_1}$ um grupo incompleto representando o nó da árvore a ser avaliado, $\mathbf{JG_2}$ o grupo composto complementar de $\mathbf{G_1}$ e $\mathbf{g_1}$, $\mathbf{j}$ e $\mathbf{g_2}$ os gabaritos respectivos. $\mathbf{J}$ representa um grupo de uma única tarefa. A expressão (4.11) do acoplamento entre grupos, garante que:

$$fo^*(\mathbf{G_1JG_2}, \mathbf{g_1jg_2}) = fo(\mathbf{G_1J}, \mathbf{g_1j}) + fo(\mathbf{G_2}, \mathbf{g_2}) \qquad (5.5)$$

Devido às propriedades envolvendo o acoplamento entre grupos e analogamente ao que foi feito para LI1,

$$fo^*(\mathbf{G_1J}, \mathbf{g_1j}) \leq fo(\mathbf{G_1J}, \mathbf{g_1j}) \qquad (5.6)$$

A idéia aqui é permitir alguma influência das tarefas ainda não programadas sobre o grupo já programado. Como não é conhecida a posição exata da tarefa do grupo $\mathbf{J}$ na seqüência, pode-se utilizar a propriedade do acoplamento entre grupos de tarefas intercaladas (item 4.4.7) para garantir que, se algumas tarefas de $\mathbf{G_2}$ se intercalarem entre $\mathbf{G_1}$ e $\mathbf{J}$, a inequação (5. 6) continuará válida. Porém, a utilização daquela propriedade exige que a desigualdade triangular seja obedecida para os tempos de troca e que os custos de troca entre tarefas sejam independentes da seqüência.

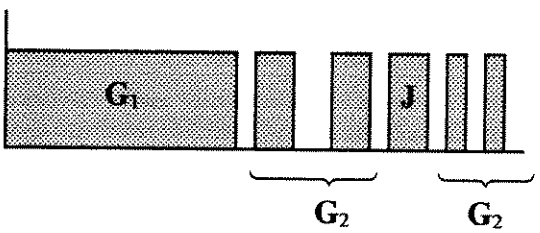


Fig 5. 2 Seqüência das tarefas nos grupos G_1, J e G_2.

Finalmente, como a alocação das frações da tarefa em $\mathbf{J}$ também é desconhecida, será necessário testar todas as alocações possíveis para escolher a que produzir o menor valor de $fo^*(\mathbf{G_1J}, \mathbf{g_1j})$. Assim, o limitante LI2 se escreve:

$$LI2 = fo^*(\mathbf{G_1J}, \mathbf{g_1j^*}) + \sum_{j \in \mathbf{G_2}} \min_{\substack{i \in I_k \\ i \neq j \\ k \in P}} (c'_{ijk}) \qquad (5.7)$$

onde:

$I_k = \mathbf{J} \cup \mathbf{G}_2 \cup \{$última tarefa de $\mathbf{G}_1$ com uma fração alocada em $k\}$

$\mathbf{j}^*$ é definido como um gabarito de $\mathbf{J}$ tal que,

$$fo^*(\mathbf{G}_1\mathbf{J}, \mathbf{g}_1\mathbf{j}^*) \leq fo^*(\mathbf{G}_1\mathbf{J}, \mathbf{g}_1\mathbf{j}) \qquad (5.8)$$

para qualquer outro gabarito $\mathbf{j}$ de $\mathbf{J}$. A tarefa contida em $\mathbf{J}$ poderá ser qualquer uma que não esteja em $\mathbf{G}_1$, mas quanto maior a sua influência sobre $\mathbf{G}_1$, maior (e melhor) será a avaliação feita pelo novo limitante. O ideal seria testar todas as tarefas restantes e escolher aquela que produzir o maior limitante. Isto porém, deve dispender um esforço computacional considerável. Parece ser mais razoável escolher a tarefa com a menor data prevista ou menor folga entre as do grupo complementar de $\mathbf{G}_1$.

O limitante LI2 é superior a LI1, o que se traduz como um número esperado menor de nós a serem pesquisados até a chegada ao grupo completo ótimo. LI2 é mais restritivo, pois exige que a desigualdade triangular de tempos de troca seja obedecida pelas tarefas envolvidas, assim como os custos de troca entre tarefas devem ser independentes da seqüência. Também o esforço computacional para o cálculo de LI2 é significativamente maior. É possível, pois, que em alguns casos o esforço computacional total do método utilizando LI1 seja menor do que utilizando LI2, apesar da melhor qualidade deste.

6. Busca em Feixe (*Beam Search*)

Este tipo de Busca, desenvolvida a partir de meados de 1970, tem apresentado bons resultados em problemas de programação (FOX, 1983), quando o objetivo não é encontrar um programa ótimo, mas trabalhar com problemas de maior porte obtendo programas de boa qualidade. OW & MORTON (1988) fizeram um estudo rigoroso do desempenho deste método em comparação com a busca em vizinhança e introduziram uma técnica de filtragem capaz de melhorar ainda mais o desempenho da Busca em Feixe. Os autores concluíram que o método com a utilização de filtragem adequada supera facilmente o desempenho da busca em vizinhança. Os problemas analisados em OW & MORTON (1988) tratam da minimização de atrasos e adiantamentos em máquinas simples, sem tempos ociosos. Eles se aproximam do problema tratado neste trabalho, quando apenas uma máquina é considerada. Os tempos ociosos presentes no modelo **F** tornam mais complexo o problema, porém podem reduzir os custos globais computados pela função objetivo.

Resumidamente, a busca em feixe utiliza a mesma árvore de busca empregada no Capítulo 5, porém sem a necessidade de utilização de limitantes inferiores. Basta a definição de um limitante que avalie a função objetivo, sem a necessidade dele ser inferior. A busca é executada em apenas uma direção, isto é, não são permitidos retornos (*backtracks*) a um nó de

nível mais baixo. Considere a Fig 6. 1 onde é mostrada uma árvore de busca com os valores dos limitantes calculados em cada nó. Quando é feita a primeira expansão a partir do nó raiz, não apenas um mas os *Bm* melhores nós, segundo a avaliação do limitante, são escolhidos. Na expansão seguinte, todos os nós escolhidos são expandidos e novamente os *Bm* melhores entre todos do novo nível são escolhidos. No caso estudado aqui, cada nível corresponde a uma nova tarefa adicionada, o que significa que, após *N* níveis, a programação estará completa e poderá ser escolhido o programa com menor valor da função objetivo como gabarito e a solução deste no modelo **F** será a solução final.

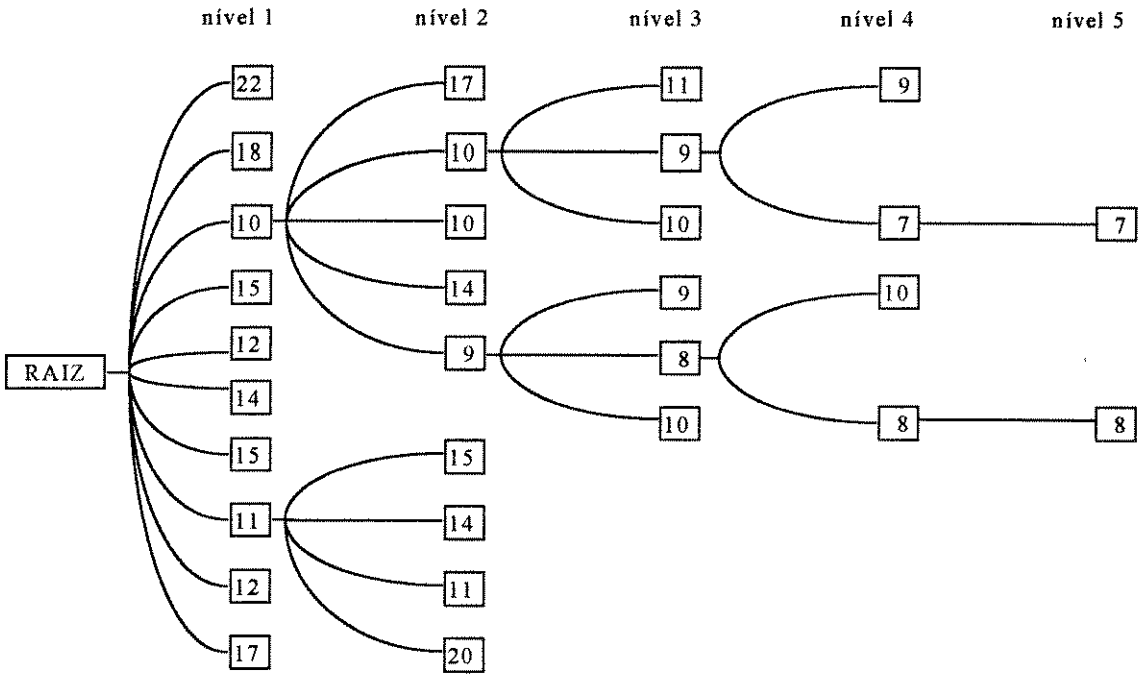


Fig 6. 1 - Busca em Feixe com *N* = 5 e largura de feixe (*Bm*) = 2

Apesar das simplificações em relação ao *Branch and Bound*, a busca em feixe tem mostrado um desempenho razoável, mesmo sem utilizar funções de avaliação perfeitas. Os erros de avaliação pelos limitantes podem fazer com que, em um determinado nível, o caminho da solução ótima seja perdido e nunca mais recuperado, pois não existem retornos a níveis anteriores. É por este motivo que são mantidos alguns ramos da árvore (largura de feixe) escolhidos entre os mais promissores. Quanto maior a largura de feixe (*Bm*), maior o esforço computacional e maior também a segurança de se chegar a um bom resultado final. Há, portanto, um compromisso entre a largura de feixe escolhida e a qualidade do programa final obtido.

6.1 Implementação da Busca em Feixe Filtrada

A filtragem proposta por OW & MORTON (1988) tem o objetivo de evitar que um limitante computacionalmente caro seja aplicado a todos os nós descendentes, uma vez feita a expansão dos *Bm* nós em um determinado nível. No exemplo da figura acima existe um número relativamente grande de nós expandidos nos níveis 1 e 2. A filtragem é feita por um critério de qualidade inferior e computacionalmente barato, que faz uma pré-seleção dos nós a serem avaliados pelo limitante caro.

As funções de avaliação aplicadas aos nós da árvore são de dois tipos básicos:

Funções globais tem validade em qualquer nível, permitindo comparações entre níveis diferentes ou entre ramos diferentes da árvore. Limitantes inferiores e limitantes superiores são exemplos de funções globais. Em geral estas funções demandam maior esforço computacional.

Funções locais tem validade apenas local, isto é, de maneira diferente das funções globais, os nós a serem comparados devem ser todos de um mesmo nível e possuir um mesmo nó pai. Um exemplo de função local é a folga imediata em problemas de uma única máquina. Esta computa a diferença entre a data prevista da tarefa e o instante de término da última tarefa processada. Por isto não tem sentido comparar folgas imediatas de nós em níveis ou ramos diferentes da árvore. Funções locais são, em geral, mais baratas do que as globais do ponto de vista computacional.

A função adotada por OW & MORTON (1988) para a filtragem possui validade local, não permitindo comparações entre nós de ramos diferentes. Por este motivo, os autores consideram apenas um nó de cada expansão, embora reconheçam que isto não é o ideal. Na Fig 6. 1 a escolha no nível 2 não poderia ser feita pela filtragem dos autores, devido à necessidade de considerar um único nó de cada uma das expansões feitas no nível 1. Naquele caso, a conseqüência seria a perda do caminho que leva até o nó final com medida de qualidade igual a 7, supondo que a função local fornece a mesma avaliação do limitante em todos os nós.

As funções responsáveis pela filtragem utilizada nesta tese (item 6.2.2 com todas as tarefas), possuem uma validade global, permitindo uma comparação equivalente em qualquer nível e em qualquer ramo da busca. Embora sejam funções globais de baixa qualidade, estas poderão selecionar quaisquer novos nós gerados, até mesmo todos de um mesmo pai, como é

o caso da escolha feita no nível 2 da Fig 6. 1. Na realidade a filosofia da filtragem se caracteriza por uma função global 1 (barata) para a filtragem e outra global 2 (mais cara) para o limitante.

Outra diferença fundamental em relação ao trabalho original de OW & MORTON (1988), é que não existe um número fixo de nós selecionados pela filtragem, que poderá variar amplamente durante a busca.

As tarefas selecionadas pela filtragem deverão satisfazer a:

$$cg(j) \leq cg_{\min} + Ft \,.\, \Delta cg \qquad (6.\ 1)$$

onde:

$cg(j)$ = característica global da tarefa j. Função global utilizada pela filtragem (número real).

$cg_{\min} = \min_{j \in G_2}\{cg(j)\}$, menor característica global entre as tarefas restantes (número real).

Ft = Filtragem, número real entre 0 e 1 (0% e 100%).

$\Delta cg = \max_{j \in T}\{cg(j)\} - \min_{j \in T}\{cg(j)\}$, variação máxima das características globais (número real).

Desta forma, muitos nós serão escolhidos quando a característica global das tarefas correspondentes aos nós mostrar pequenas diferenças e haverá apenas um ($cg_{\min}$) se as diferenças forem grandes, o que parece ser razoável. A filtragem seleciona apenas tarefas e não alocações específicas, isto é, para cada tarefa selecionada haverá nós com todas as alocações possíveis. A característica global de cada tarefa poderá ser a data prevista ou a folga global como calculada adiante no item 6.2.2.

6.2 *Tipos de limitantes*

A equação (4.11) do acoplamento entre grupos mostra claramente que uma avaliação precisa sobre a função objetivo ótima de um grupo completo precisa considerar o grupo complementar $\mathbf{G}_2$, que foi avaliado de maneira bastante pobre no cálculo dos limitantes inferiores. A busca em feixe não exige limitantes inferiores, mas funções de avaliação com boa capacidade de discriminação entre nós promissores e nós com alta probabilidade de levar a resultados finais ruins. Com mais liberdade de escolha, é possível desenvolver limitantes muito mais precisos e discriminantes do que os utilizados na busca *Branch and Bound*. Isto será possível através da montagem das tarefas restantes não programadas, sobre o nó a ser avaliado, segundo uma ordem determinada por uma regra de despacho e alocações baseadas em heurísticas que serão desenvolvidas adiante.

Dois limitantes (L3 e L5) podem ser chamados de limitantes superiores, pois serão a própria função objetivo do programa completo formado a partir do nó sob avaliação. Nestes casos, teoricamente, o grupo final (completo) nunca possuirá uma função objetivo superior aos limitantes calculados durante a busca. Na prática, porém, em alguns raros casos esta regra poderá ser violada devido a arredondamentos de ponto flutuante durante o cálculo.

Os limitantes L4 e L6 não garantem que o resultado seja um limitante superior, devido às simplificações utilizadas no cálculo de ambos, porém, são de baixo custo computacional se comparados com L3 e L5, o que permitirá a sua aplicação a problemas de maior porte.

Para uma avaliação razoável das tarefas restantes (grupo $\mathbf{G_2}$) é necessário colocá-las em um gabarito provisório. Nos itens a seguir, o seqüenciamento e alocação de tarefas de forma provisória para o cálculo dos limitantes será descrito:

- Itens 6.2.2, 6.2.3, 6.2.3.1e 6.2.3.2: formação do gabarito provisório.
- Itens 6.2.4 e 6.2.5: Descrição do cálculo dos limitantes.

6.2.1 Cálculo das variáveis da tarefa correspondente ao nó sob avaliação

Os limitantes L4 e L6 não fazem reotimizações pelo modelo **F** a cada nova tarefa adicionada, o que exige a utilização de uma heurística especial para um cálculo aproximado das variáveis relativas ao nó sob avaliação.

Como uma primeira aproximação, considere o caso onde j (a tarefa correspondente ao nó sob avaliação) esteja atrasada. Neste caso, não haverá tempos ociosos anteriores às suas frações e, para minimizar o atraso, os tempos de término das frações de j deverão se igualar. A partir da restrição (4.2) escrita para todas as tarefas, com $id(j,k) = 0$, $c_{ijk} + a(0,k) = co_{ijk}$ e $a(j,k) = a(j)$ para todo $k \in M_j$, somando todas e utilizando (4.3) chega-se a:

$$a(j) = \frac{\sum_{k \in M_j} \frac{co_{ijk} + a(i,k)}{s_{jk}} + q_j}{\sum_{k \in M_j} \frac{1}{s_{jk}}} \qquad (6.2)$$

Considerando que as tarefas anteriores a j estão fixas (não possuem acoplamento com j) e que, no momento do cálculo das frações, j é a última tarefa da seqüência, o adiantamento pode sempre ser anulado calculando tempos ociosos adequados antes das frações. O atraso será mínimo se os tempos ociosos forem anulados e os tempos de término das frações igualado, com a maior divisão possível para a tarefa. No procedimento abaixo a melhor solução possível para j (considerando as hipóteses acima) é obtida.

1) Calcula-se $co\_max = \max_{k \in M_j}(co_{ijk} + a(i,k))$ (6. 3)

2) Calcula-se $a(j)$ por (6. 2).

3) Calcula-se o menor tempo de término possível para a tarefa na alocação dada:

$$a\_min(j) = max\{co\_max,\ a\ (j)\} \qquad (6.\ 4)$$

4) Com o cálculo de $a\_min(j)$, as variáveis da tarefa j são calculadas pelo procedimento abaixo escrito em uma linguagem hipotética semelhante à linguagem C, onde são permitidos os símbolos matemáticos e notações utilizadas aqui. Os comandos são separados por " ; " e os procedimentos "FOR", "IF" e "ELSE" iniciam com "{" e terminam com "}", a, ac e fr e são variáveis de ponto flutuante (armazenam números reais).

IF ($a\_min(j) < d_j$) { $a\_min(j) = d_j$; }
$z(j) = a\_min(j) - d_j$;
$e(j) = 0$;
ac = q_j ;
FOR($k \in M_j$)
{
  fr = $(a\_min(j) - co_{ijk} - a(i,k))/s_{jk}$;
  IF (ac < fr) { fr = ac ; }
  ac = ac - fr ;
  $r(j,k)$ = fr ;
  a = $a(i,k) + co_{ijk} + s_{jk}$*fr ;
  IF (a > d_j) { $a(j,k)$ = a ; $id(j,k) = 0$; }
  ELSE { $a(j,k) = d_j$; $id(j,k) = d_j$ - a ; }
}

onde as instruções IF e ELSE das últimas linhas foram incluídas para minimizar os tempos de término de algumas frações sem alterar o custo da tarefa.

6.2.2 Seqüenciamento das tarefas restantes

Utilizam-se funções de baixo custo computacional (regras de despacho) para proceder o seqüenciamento das tarefas restantes. Regras como a escolha da tarefa com menor data prevista de entrega (EDD) ou menor folga (LL) como a próxima da seqüência, parecem ser boas escolhas devido à complexidade do problema global e à falta de pesquisas anteriores sobre este tema específico. No futuro, funções mais precisas poderão surgir, melhorando a avaliação dos limitantes. No caso do critério EDD, não há necessidade de maiores considerações mas, no caso do critério LL, é necessário definir o que se entende por folga da tarefa.

A folga é definida como a diferença entre a data prevista da tarefa e o menor tempo de término possível, supondo que a tarefa está sozinha no modelo **F**. A folga pode ser calculada com um algoritmo semelhante ao utilizado para obter uma solução ótima para uma tarefa no item anterior, porém eliminando as frações que se mostrarem nulas:

1) Parâmetros $\overline{d}_j = 0, \overline{co}_{ijk} = \frac{\sum_{i \in T0} co_{ijk}}{N} \quad \forall j \in T, \forall k \in M_j$, substituindo d_j e co_{ijk}. $\rightarrow$ 2)

2) M_j preenchido com todas as máquinas aptas à execução da tarefa j. $\rightarrow$ 3)

3) Executa-se o algoritmo do item 6.2.1 com $a(i,k) = 0$. $\rightarrow$ 4)

4) Se algum $r(i,k) = 0$, Remove-se a fração nula de M_j, $\rightarrow$ 3)
Se não $\rightarrow$ 5)

5) $folga(j) = d_j - z(j)$, $\rightarrow$ Fim

6.2.3 Alocação das tarefas restantes

Duas funções diferentes serão combinadas para a formação dos quatro limitantes. Em ambos os casos, as heurísticas estarão sempre sendo aplicadas à última tarefa da seqüência no momento. Em uma solução ótima, a última tarefa da seqüência não poderá estar adiantada, mesmo porque seus adiantamentos poderiam ser facilmente anulados com a introdução de tempos ociosos anteriores às suas frações. Desta forma, é razoável desprezar os adiantamentos

da tarefa atual nas duas heurísticas de alocação. Também em ambas, as tarefas anteriores são consideradas fixas, isto é, despreza-se a influência da tarefa atual sobre as anteriores.

6.2.3.1 Heurística de alocação em uma máquina (sem splitting)

Neste caso, a hipótese é que não haverá divisão da tarefa que está sendo alocada, o que significa que sua execução ocorrerá em uma única máquina. Em muitos casos, freqüentemente na maioria deles, a alocação de tarefas em uma única máquina é uma possibilidade atraente, como se observou em quase todos os gabaritos ótimos obtidos nos exemplos de programação inteira-mista (veja no item 7.1).

A máquina escolhida será aquela que apresentaria o menor custo, caso a tarefa estivesse alocada apenas a ela. Seja $C_1(j,k)$ o custo caso a tarefa estivesse alocada apenas à máquina k, sem considerar o adiantamento.

$$C_1(j,k) = \beta_j . z(j) + c'_{ijk}$$

onde:

$$z(j) = \begin{cases} 0 & \text{se } a(j,k) \le d_j \\ a(j,k) - d_j & \text{se } a(j,k) > d_j \end{cases} \qquad (6.5)$$

$$a(j,k) = a(i,k) + co_{ijk} + q_j . s_{jk}$$

A máquina escolhida k_{esc} será tal que $C_1(j,k_{esc}) \le C_1(j,k)$ para $k \in M_j$. Como se trata apenas de um artifício para avaliar o custo de uma tarefa pertencente ao grupo complementar $\mathbf{G}_2$, isto não implicará em que todas as tarefas de $\mathbf{G}_2$ sejam necessariamente alocadas a uma única máquina ou mesmo na máquina escolhida por esta heurística. Não existe impedimento para que um nó com tarefas divididas em muitas máquinas possua uma pequena avaliação pelo limitante e seja escolhido como parte do programa final.

6.2.3.2 Heurística de minimização de custos (com splitting)

Aqui a hipótese é que haverá divisão de tarefas na maior extensão que o custo permitir. Isto é, fixando a posição das tarefas já alocadas, o objetivo é encontrar a alocação que minimiza o custo da tarefa que está sendo alocada no momento, com tempos ociosos nulos e desprezando adiantamentos. Isto é feito com o seguinte procedimento:

1) Determinar k_{esc} pelo procedimento do item anterior. $\rightarrow$ 2)

2) se $z(j) = 0 \rightarrow$ Fim (a inclusão de frações não pode reduzir o custo associado à alocação de j).

se $z(j) > 0 \rightarrow 3)$

3) Procura-se entre as máquinas restantes aquela capaz de causar a maior redução da função objetivo determinada pelas alocações já decididas. Isto pode ser feito calculando-se inicialmente o tempo de término mínimo (por (6. 2)) para a alocação atual em M_j.

$$a(j) = \frac{\sum_{k \in M_j} \frac{a(i,k) + co_{ijk}}{s_{jk}} + q_j}{\sum_{k \in M_j} \frac{1}{s_{jk}}} \qquad (6.\ 6)$$

Com a inclusão de uma fração na máquina $k^n \notin M_j$:

$$a^n(j,k^n) = \frac{\sum_{k \in M_j} \frac{a(i,k) + co_{ijk}}{s_{jk}} + \frac{a(i,k^n) + co_{ijk^n}}{s_{jk^n}} + q_j}{\sum_{k \in M_j} \frac{1}{s_{jk}} + \frac{1}{s_{jk^n}}} \qquad (6.\ 7)$$

Seja e a redução na função objetivo causada pela introdução de uma fração na máquina k^n. Então, de (4.1):

$$\begin{aligned} &\beta_j \cdot \{a(j) - d_j\} + \eta_j - \beta_j \cdot \{a^n(j,k^n) - d_j\} - \eta_j - c'_{ijk^n} = e \\ &a^n(j,k^n) + \frac{c'_{ijk^n}}{\beta_j} = a(j) - \frac{e}{\beta_j} \end{aligned} \qquad (6.\ 8)$$

Como, neste caso, $a(j)$ é constante, a máquina k_{nova} deverá ser aquela que minimiza o lado esquerdo de (6. 8). De (6. 7) e (6. 8), define-se a função $F(j,k^n)$ como:

$$F(j,k^n) = a^n(j,k^n) + \frac{c'_{ijk^n}}{\beta_j} \qquad k^n \notin M_j \qquad (6.\ 9)$$

A máquina escolhida k_{nova} será tal que $F(j,k_{nova}) \leq F(j,k^n)$ para $k^n \in P$, $k^n \notin M_j$. $\rightarrow 4)$

4) Se $F(j,k_{nova}) \geq a(j) \rightarrow$ Fim (não pode haver melhoramento no custo de j).

Se $F(j,k_{nova}) < a(j)$, k_{nova} é incluída em M_j, isto é, $M_j = M_j \cup \{ k_{nova} \}$.

$$z(j) = \begin{cases} 0 & \text{se } a^n(j,k_{nova}) \leq d_j \\ a^n(j,k_{nova}) - d_j & \text{se } a^n(j,k_{nova}) > d_j \end{cases} \qquad (6.\ 10)$$

$\rightarrow 2)$

6.2.4 Limitantes L3 e L5 (com reotimização)

A partir do nó a ser avaliado, as tarefas restantes são ordenadas segundo o critério EDD ou LL (item 6.2.2). No caso de L3, a alocação será feita tarefa a tarefa usando a heurística de minimização de custos (item 6.2.3.2) e no caso de L5 com a heurística de alocação em uma única máquina (item 6.2.3.1). A cada nova tarefa adicionada, executa-se uma reotimização do quadro Simplex com a nova tarefa incluída. Com isso leva-se em conta o efeito da tarefa incluída sobre as anteriores, o que produzirá uma melhor avaliação da solução ótima. Neste caso, o custo computacional será elevado, mas os limitantes serão de alta qualidade.

6.2.5 Limitantes L4 e L6 (sem reotimização)

A partir do nó a ser avaliado, as tarefas restantes são ordenadas segundo o critério EDD ou LL (item 6.2.2). No caso de L4, a alocação será feita tarefa a tarefa usando a heurística de minimização de custos (item 6.2.3.2) e no caso de L6 com a heurística de alocação em uma única máquina (item 6.2.3.1). Nestes casos, não serão feitas reotimizações a cada nova tarefa adicionada, para reduzir o esforço computacional na geração dos limitantes. As minimizações pelo modelo **F** só ocorrerão durante a busca nos nós escolhidos para expansão. Portanto, não haverá um quadro minimizado do nó a ser avaliado, mas apenas do nó pai deste. Seguem alguns detalhes de como são implementados os cálculos dos limitantes L4 e L6.

Seja $\mathbf{G_1JG_2}$ um grupo completo, $\mathbf{G_1}$ um grupo incompleto representando os nós antecessores do nó sob avaliação, **J** um grupo de uma única tarefa representando o nó sob avaliação, $\mathbf{G_2}$ o grupo complementar de $\mathbf{G_1J}$ e $\mathbf{g_1}$, **j** e $\mathbf{g_2}$ os gabaritos correspondentes. A expressão do item 4.4.3 (acoplamento entre grupos), garante que:

$$fo^*(\mathbf{G_1JG_2}, \mathbf{g_1jg_2}) = fo(\mathbf{G_1}, \mathbf{g_1}) + fo(\mathbf{J}, \mathbf{j}) + fo(\mathbf{G_2}, \mathbf{g_2}) \qquad (6.\ 11)$$

A primeira parcela de (6. 11), pode ser avaliada, de forma aproximada, por $fo^*(\mathbf{G_1}, \mathbf{g_1})$ que é conhecido do quadro minimizado do nó pai. Isto é,

$$fo(\mathbf{G_1}, \mathbf{g_1}) = fo^*(\mathbf{G_1}, \mathbf{g_1}) \qquad (6.\ 12)$$

5. A tarefa em **J**, já possui seqüenciamento e alocação definidos pelo procedimento de busca, faltando apenas uma solução adequada para a avaliação de $fo(\mathbf{J}, \mathbf{j})$. Fixando a

posição das tarefas em $\mathbf{G}_1$, o algoritmo do item 6.2.1 será utilizado para obter uma solução para **J** em **j**. Portanto, já que o algoritmo não permite adiantamento,

$$fo(\mathbf{J}, \mathbf{j}) = \beta_j.z(j) + \eta_j \qquad (6.\ 13)$$

Para a última parcela de (6. 11), $fo(\mathbf{G}_2, \mathbf{g}_2)$, os dois limitantes propõem abordagens diferentes:

L4) A alocação das tarefas pertencentes a $\mathbf{G}_2$ será feita pela heurística de minimização de custos (item 6.2.3.2) com as tarefas sendo adicionadas uma após a outra, procurando sempre a minimização de atrasos e custos de troca. Como os tempos ociosos da heurística são sempre nulos, os adiantamentos das tarefas serão desprezados, mesmo que a diferença entre a data prevista de uma tarefa e seu tempo de término seja grande. Isto pode ser considerado razoável, pois os adiantamentos deverão ser bastante reduzidos quando for feita a minimização final sobre o grupo completo, além do que o custo de adiantamento costuma ser sensivelmente menor que o custo dos atrasos (OW & MORTON, 1988). Ainda assim, a omissão dos adiantamentos em $\mathbf{G}_2$ e o desprezo do acoplamento entre $\mathbf{G}_1$ e $\mathbf{JG}_2$ enfraquecerá L4, especialmente nos casos em que as datas previstas das tarefas estiverem relativamente distantes umas das outras.

L6) A alocação das tarefas pertencentes a $\mathbf{G}_2$ será feita pela heurística de alocação em uma única máquina (item 6.2.3.1), mas a cada nova adição será aplicado o procedimento de minimização de atrasos e adiantamentos em máquina simples (item 6.2.5.1), uma vez que, estando cada tarefa em apenas uma máquina, o problema pode ser decomposto e o procedimento executado independentemente em uma máquina de cada vez. L6 é computacionalmente superior a L5, mas despreza o acoplamento entre $\mathbf{G}_1$ e $\mathbf{JG}_2$ e, portanto, é esperado que ele apresente um comportamento mais fraco.

6.2.5.1 Minimização de atrasos e adiantamentos em máquina simples

GAREY, TARJAN e WILFONG (1988) mostraram ser possível minimizar atrasos e adiantamentos com um procedimento mais simples e rápido que a resolução do problema de programação linear, embora o problema resolvido pelos autores apresentasse custos de atraso e adiantamento simétricos, isto é, iguais para todas as tarefas. DAVIS & KANET (1993) apresentam um procedimento semelhante para a minimização em uma única máquina com custos de atraso e adiantamento diferentes para cada tarefa. O procedimento TIMETABLER

desenvolvido por DAVIS & KANET (1993), estaria próximo do ideal para o caso de L6, não fosse o fato de seguir uma ordem inversa na adição das tarefas ao programa. O procedimento apresentado abaixo, é equivalente ao TIMETABLER, seguindo uma ordem crescente na adição das tarefas e foi desenvolvido com os conceitos presentes em ambos os artigos citados acima.

O modelo **F** aplicado a uma única máquina fornece uma maneira simples e precisa de entender o que ocorre com os custos das tarefas submetidas aos deslocamentos (*shifts*) a que se referem os autores acima.

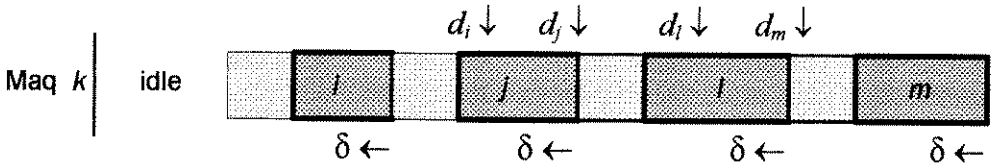


Fig 6. 2 - Bloco de tarefas

A figura acima representa um típico bloco de tarefas segundo a definição de GAREY, TARJAN e WILFONG (1988), isto é, um certo número de tarefas com tempo ocioso nulo entre elas. No bloco, as tarefas *l* e *m* estão atrasadas, *i* está adiantada e *j* termina exatamente sobre a data prevista. Nestas condições, e removendo os custos de troca da função objetivo, o custo do bloco será:

$$fo_0^{ez}(\mathbf{B},\mathbf{b}) = \alpha_i(d_i - a(i,k)) + \alpha_j(d_j - a(j,k)) + \beta_l(a(l,k) - d_l) + \beta_m(a(m,k) - d_m) \qquad (6.\ 14)$$

onde **B** é o grupo com as tarefas do bloco, **b** é o gabarito da seqüência acima e $fo_0^{ez}(\mathbf{B},\mathbf{b})$ o custo inicial do grupo. Observe que o termo referente à tarefa *j* é nulo e poderia ser omitido, mas o procedimento a seguir justifica a sua presença. Suponhamos que o bloco acima seja deslocado para a esquerda de δ unidades de tempo. O deslocamento corresponde a subtrair δ dos tempos de término das tarefas, com o seguinte efeito sobre a função objetivo do grupo:

$$fo_\delta^{ez}(\mathbf{B},\mathbf{b}) = fo_0^{ez}(\mathbf{B},\mathbf{b}) + \delta.(\alpha_i + \alpha_j - \beta_l - \beta_m) \qquad (6.\ 15)$$

onde $\delta \geq 0$ e $fo_\delta^{ez}(\mathbf{B},\mathbf{b})$ é o custo do grupo deslocado. Com (6. 15) fica claro que haverá uma redução na função objetivo do grupo se $\alpha_i + \alpha_j < \beta_l + \beta_m$. Neste caso, note também que quanto maior δ, menor $fo_\delta^{ez}(\mathbf{B},\mathbf{b})$. Entretanto, δ não pode crescer arbitrariamente, pois logo chegará o momento em que o tempo de término de uma outra tarefa (p.ex. *l*) atinge sua data prevista, ou o tempo ocioso anterior ao bloco se anula, o primeiro que ocorrer. No primeiro caso, a equação (6. 14) deverá ser refeita para avaliar se um novo deslocamento pode reduzir ainda

mais a função objetivo. Se o tempo ocioso se anular porque o bloco atingiu o início ($t = 0$), nada mais pode ser feito, mas se o bloco encontrou um outro anterior, o conjunto de ambos é um novo bloco passível da mesma análise.

Seguindo o procedimento básico de GAREY, TARJAN e WILFONG (1988), a primeira tarefa da seqüência é programada, se possível com o tempo de término sobre a data prevista, ou ficará atrasada mas com o tempo ocioso nulo. Desta maneira, tem-se um programa ótimo para esta primeira tarefa. Da mesma forma, a segunda tarefa é programada, se possível, com o tempo de término sobre a data prevista, ou ficará atrasada mas com o tempo ocioso nulo. Neste ponto, o bloco contendo a segunda tarefa poderá sofrer um deslocamento para a esquerda de maneira a otimizar a solução do grupo incompleto atual. Procedendo desta forma com todas as outras tarefas do grupo, ao final a solução será ótima, pois não será possível deslocar qualquer bloco ou tarefa com redução do valor da função objetivo.

Observe que os deslocamentos serão sempre para a esquerda, pois uma tarefa no momento em que for programada, jamais estará adiantada. Quando isto ocorrer, será causado por um deslocamento devido a uma tarefa posterior, portanto, deslocamentos para a direita só serão razoáveis se tarefas já programadas forem removidas, o que nunca acontece.

O procedimento abaixo, resume a construção de uma solução ótima em um grupo com N tarefas. $a(j)$ é o tempo de término da tarefa j, p_j é o tempo de processamento de j, c_{ij} o tempo de troca entre i e j e $id(j)$ o tempo ocioso anterior a j.

1) $j = 0$, $\rightarrow$ 2)

2) j = próxima tarefa da seqüência.
Se $j = (N + 1)^a \rightarrow$ Fim
Se não $\rightarrow$ 3)

3) Se $p_j + c_{ij} + a(i) \leq d_j$, então $a(j) = d_j$, $id(j) = d_j - p_j - c_{ij}$, $\rightarrow$ 2)
Se não $\rightarrow$ 4)

4) $a(j) = p_j + c_{ij} + a(i)$, $id(j) = 0$, $\rightarrow$ 5)

5) l = primeira tarefa anterior a j com um tempo ocioso não nulo,

Se $l = 0$, $\rightarrow$ 2)

Se não, $\rightarrow$ 6)

6) Seja **B** o bloco formado pelas tarefas entre l e j, incluindo estas.

$\Sigma\alpha$ = somatório dos α das tarefas em **B** adiantadas ou sobre a data prevista,

$\Sigma\beta$ = somatório dos β das tarefas em **B** atrasadas, $\rightarrow$ 7)

7) Se $\Sigma\beta \leq \Sigma\alpha$, $\rightarrow$ 2)

Se não, $\rightarrow$ 8)

8) δ = mínimo entre atrasos das tarefas em **B** e $id(l)$, $\rightarrow$ 9)

9) Subtrair δ dos tempos de término de todas as tarefas em **B**, $\rightarrow$ 5)

No caso de L6, a seqüência será formada pelas tarefas de $\mathbf{JG}_2$ alocadas na máquina k e a tarefa possivelmente dividida em outras máquinas será sempre a primeira. No caso desta primeira, se a fração j,k não possuir tempo ocioso, não poderá sofrer deslocamentos para a esquerda. Se possuir tempo ocioso, seu custo de adiantamento α_j será considerado nulo, a menos que a fração possua o menor tempo de término entre todas da tarefa j, o que exige uma verificação antes da execução do procedimento.

7. Resultados Computacionais

Neste capítulo o desempenho computacional do método de busca desenvolvido nos Capítulos anteriores será avaliado em um conjunto variado de problemas. O item 7.1 será dedicado ao teste de desempenho dos modelos de programação inteira mista **S, C** e **C'** desenvolvidos no Capítulo 3, comparando os tempos de execução com os tempos do método *Branch and Bound* desenvolvido no Capítulo 5. Neste item, apenas pequenos problemas serão resolvidos, para comprovar a capacidade de resolução dos modelos, bem como os tempos de computação elevados em relação ao método de busca em árvore.

O restante do capítulo se refere à avaliação da busca em feixe filtrada desenvolvida no Capítulo 6. No item 7.2 são apresentados os métodos de geração aleatória utilizados e definição dos parâmetros que relacionam as datas de entrega de tarefas (item 7.3). Em seguida, no item 7.4 é descrito um procedimento de geração de exemplos cuja solução ótima é conhecida e as medidas de avaliação dos resultados são apresentadas no item 7.5. No item 7.6 é feita uma avaliação simplificada da sensibilidade à variação da largura de feixe e filtragem na busca em feixe.

Para que seja possível a comparação de alguns dos resultados obtidos com as soluções ótimas de grupos completos sem limitar o espectro dos testes, os exemplos serão divididos em duas partes:

1ª) Exemplos formados por grupos desacoplados com custos de troca entre tarefas independentes da seqüência, o que implica na possibilidade de prever gabaritos e soluções ótimas.

2ª) Não haverá qualquer garantia de desacoplamento entre grupos ou tarefas e os custos de troca serão dependentes da seqüência, o que implicará no desconhecimento de soluções ou gabaritos ótimos.

7.1 Exemplos com os modelos de programação inteira mista

Estes exemplos foram incluídos aqui apenas para facilitar a compreensão do problema e fazer uma comparação, embora superficial, do desempenho computacional destes modelos em relação à busca *Branch and Bound*.

- $N = 6; M = 1$

Como primeiro exemplo, será utilizado um problema com uma única máquina e 6 tarefas. Desta maneira não poderá haver divisão de tarefas, o que significa que os três modelos inteiro-mistos deverão fornecer a mesma solução.

Neste caso, pode ser feita uma analogia com o problema clássico do caxeiro viajante (CV), que deve visitar um certo número de cidades a diferentes distâncias umas das outras, percorrendo a distância total mínima possível.. Com uma pequena alteração no problema de programação, para a minimização do atraso máximo, os tempos de troca entre tarefas podem ser análogos às distâncias entre cidades do problema CV. Assim, se a data prevista de todas as tarefas forem iguais à soma dos tempos de processamento das tarefas, a solução do problema será idêntica à do problema CV.

A alteração será a introdução da variável $z_{max} = z(j), \forall j \in T$ nas restrições dos modelos e também nas funções objetivo, que passam a ter um único termo (z_{max}). Representando os parâmetros por matrizes, tem-se:

$P = \{A\}$; $K_A = T = \{1,2,3,4,5,6\}$; $HV = 500$; $a(0,A) = 0$

$$[c_{ijA}] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 6 & 20 & 30 & 15 \\ 5 & 0 & 15 & 15 & 40 & 20 \\ 15 & 8 & 0 & 15 & 15 & 20 \\ 25 & 20 & 20 & 0 & 10 & 5 \\ 40 & 30 & 10 & 10 & 0 & 20 \\ 10 & 6 & 5 & 30 & 15 & 0 \end{bmatrix} \quad [s_{jA}] = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \quad [q_j] = \begin{bmatrix} 5 \\ 6 \\ 4 \\ 3 \\ 10 \\ 15 \end{bmatrix} \quad [d_j] = \begin{bmatrix} 43 \\ 43 \\ 43 \\ 43 \\ 43 \\ 43 \end{bmatrix}$$

onde os custos de troca são todos nulos. Observe que $d_j = 43$ corresponde à soma dos tempos de processamento de todas as tarefas, portanto, na solução ótima, z_{max} deverá ser a menor soma possível dos tempos de troca entre tarefas . A matriz de tempos de troca é retangular (7X6), porque pode haver tempos de troca iniciais c_{0jA} (que neste caso são nulos), mas não existem tempos de troca finais c_{i0A}.

Os modelos **S, C** e **C'** alterados obtiveram a mesma solução para o problema acima: (valor da função objetivo = 32)

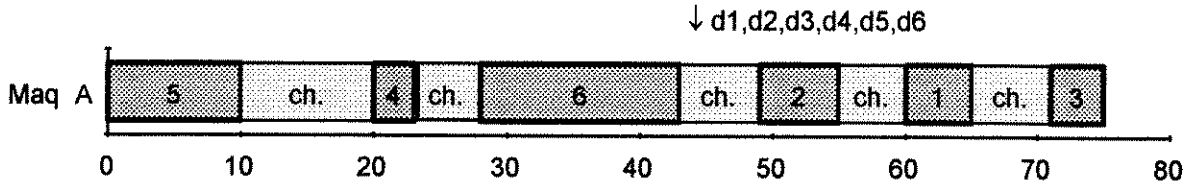


Fig 7. 1 - Exemplo com 6 tarefas em 1 máquina

As soluções dos modelos de programação inteira mista foram obtidas com o AMPL/CPLEX para MS-DOS da empresa norte americana AT&T (FOURER, GAY, KERNIGHAN, 1993) e a soluções do método *Branch and Bound* foram obtidas por um software específico desenvolvido em linguagem C e compilado com o IBM C/C++ para OS/2 (o método *Branch and Bound* permite divisão de tarefas). Todos os exemplos desta seção foram executados em um microcomputador 80486 DX2 66 Mhz em tempos de:

Modelo **S**. - 36,0 seg ,14.407 iterações inteiras (modelo sem divisão de tarefas).

Modelo **C**. - 405,6 seg, 113.740 iterações inteiras.

Modelo **C'**. - 44,6 seg, 18.482 iterações inteiras.

Branch and Bound com LI1 - 1,4 seg, 465 nós gerados. → Com a função objetivo padrão incluindo atrasos e adiantamentos,

$\alpha_j = 0$; $\beta_j = 1$. A solução ótima obtida é diferente, mas o tempo de processamento deve se aproximar ao que seria obtido com as modificações citadas acima.

- $N = 3; M = 2$

Com duas máquinas disponíveis, é possível haver divisão de tarefas, o que significa que o modelo sem esta característica poderá fornecer uma solução diferente. Com o objetivo de ilustrar como o tempo ocioso é inserido, serão definidas datas previstas relativamente grandes, mas todas iguais, para forçar uma disputa entre as tarefas. Representando os parâmetros por matrizes, tem-se:

$$P = \{A,B\};\quad K_A = K_B = T = \{1,2,3\};\quad HV = 100;\quad [a(0,k)] = [0 \quad 0]$$

$$[c_{ijA}] = [c_{ijB}] = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 3 & 1 \\ 2 & 0 & 2 \\ 3 & 1 & 0 \end{bmatrix};\quad [s_{jk}] = \begin{bmatrix} 3 & 2 \\ 5 & 3 \\ 4 & 3 \end{bmatrix};\quad [\alpha_j] = [\beta_j] = [q_j] = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix};\quad [d_j] = \begin{bmatrix} 8 \\ 8 \\ 8 \end{bmatrix}$$

O último termo da função objetivo (custos de troca entre tarefas) foi removido para acelerar a computação. No problema sem divisão de tarefas, a matriz de tempos de processamento foi feita igual à matriz de tempos unitários de processamento dos outros dois modelos.

A solução ótima sem divisão de tarefas, foi: (valor da função objetivo = 4)

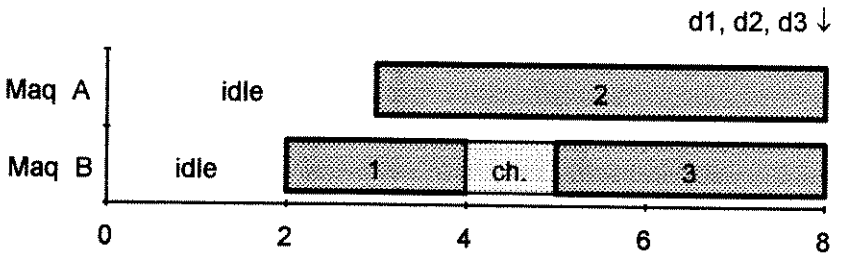


Fig 7. 2 - Exemplo com 3 tarefas em 2 máquinas - sem divisão de tarefas

A solução ótima com divisão de tarefas, foi: (valor da função objetivo = 3,143)

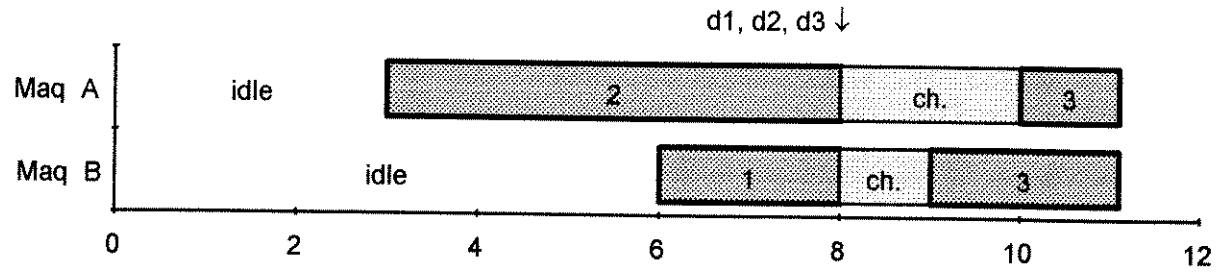


Fig 7. 3 - Exemplo com 3 tarefas em 2 máquinas - com divisão de tarefas

Os tempos de execução foram:

Modelo **S** - 0,4 seg, 93 iterações inteiras (modelo sem divisão de tarefas).

Modelo **C** - 7,4 seg, 2.568 iterações inteiras.

Modelo **C'** - 4,7 seg, 1.697 iterações inteiras.

Branch and Bound com LI1 - 0,125 seg, 65 nós gerados.

- $N = 3; M = 3$

Novamente poderá haver divisão de tarefas, o que significa que o modelo sem esta característica poderá fornecer uma solução diferente. Desta vez, serão definidas datas previstas diferentes, mas próximas. Representando os parâmetros por matrizes, tem-se:

$$P = \{A,B,C\};\quad K_A = K_B = K_C = T = \{1,2,3\};\quad HV = 200;\quad [a(0,k)] = [0\ \ 0\ \ 0]$$

$$[c_{ijA}] = [c_{ijB}] = \begin{bmatrix} 25 & 10 & 15 \\ 0 & 25 & 20 \\ 10 & 0 & 10 \\ 15 & 15 & 0 \end{bmatrix};\quad [s_{jk}] = \begin{bmatrix} 4 & 3 & 6 \\ 2 & 5 & 3 \\ 3 & 2 & 2 \end{bmatrix};\quad [\alpha_j] = [\beta_j] = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix};\quad [q_j] = \begin{bmatrix} 10 \\ 10 \\ 10 \end{bmatrix};\quad [d_j] = \begin{bmatrix} 30 \\ 35 \\ 40 \end{bmatrix}$$

Neste exemplo, os tempos de troca iniciais não são nulos e o termo da função objetivo relativo a custos de troca foi removido. No problema sem divisão de tarefas, a matriz de tempos de processamento foi feita igual à matriz de tempos unitários de processamento, multiplicada por 10, dos outros dois modelos.

A solução ótima sem divisão de tarefas, foi: (valor da função objetivo = 25)

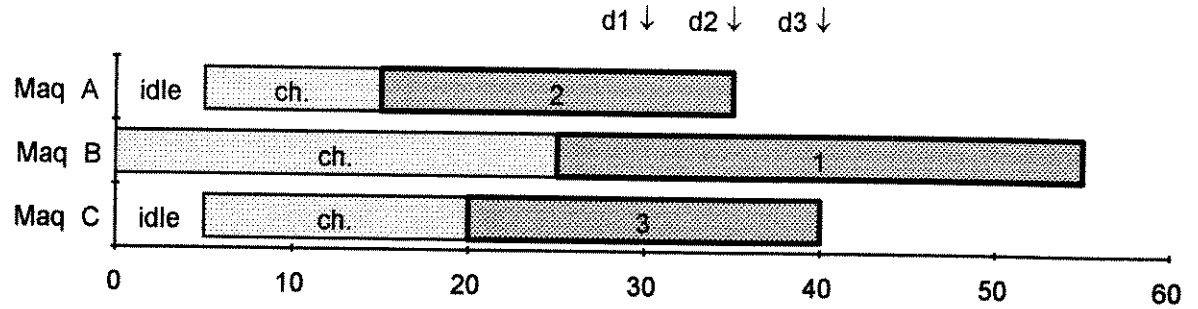


Fig 7. 4 - Exemplo com 3 tarefas em 3 máquinas - sem divisão de tarefas

A solução ótima com divisão de tarefas, foram: (valor da função objetivo = 20,714)

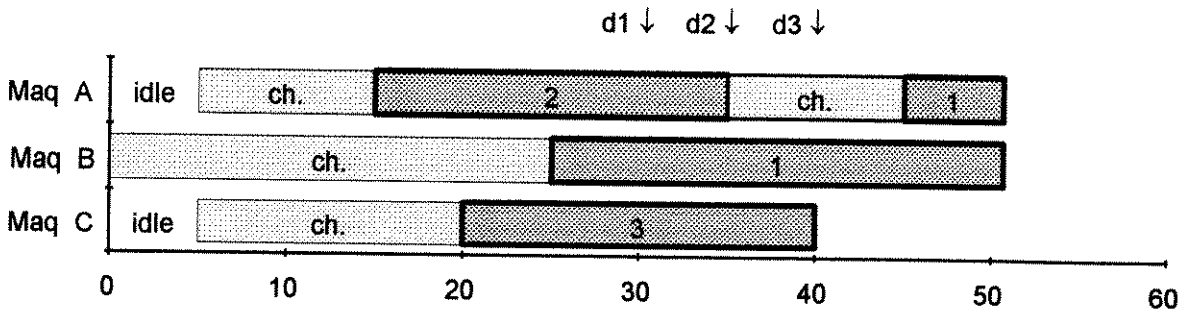


Fig 7. 5 - Exemplo com 3 tarefas em 3 máquinas - com divisão de tarefas

Os tempos de execução foram:

Modelo **S** - 1,2 seg, 396 iterações inteiras (modelo sem divisão de tarefas)

Modelo **C** - 60,2 seg, 16.983 iterações inteiras.

Modelo **C'** - 22,8 seg, 7.031 iterações inteiras.

Branch and Bound com LI1 - 1,1 seg, 186 nós gerados.

- $N = 4; M = 2$

Novamente poderá haver divisão de tarefas, o que significa que o modelo sem esta característica poderá fornecer uma solução diferente. Representando os parâmetros por matrizes, tem-se:

$$P = \{A,B\};\quad K_A = K_B = T = \{1,2,3,4\};\quad HV = 200;\quad [a(0,k)] = [0\ \ 0]$$

$$[c_{ijA}] = [c_{ijB}] = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 3 & 2 \\ 1 & 0 & 2 & 2 \\ 2 & 3 & 0 & 1 \\ 3 & 2 & 1 & 0 \end{bmatrix};\quad [s_{jk}] = \begin{bmatrix} 2 & 2 \\ 3 & 5 \\ 4 & 3 \\ 5 & 4 \end{bmatrix};\quad [\alpha_j] = \begin{bmatrix} 2 \\ 2 \\ 2 \\ 2 \end{bmatrix};\quad [\beta_j] = [q_j] = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix};\quad ;[d_j] = \begin{bmatrix} 3 \\ 3 \\ 4 \\ 6 \end{bmatrix}$$

Neste exemplo, os tempos de troca iniciais são nulos e último termo da função objetivo foi removido. No problema sem divisão de tarefas, a matriz de tempos de processamento foi feita igual à matriz de tempos unitários de processamento dos outros dois modelos.

A solução ótima com ou sem divisão de tarefas, foi: (valor da função objetivo = 11)

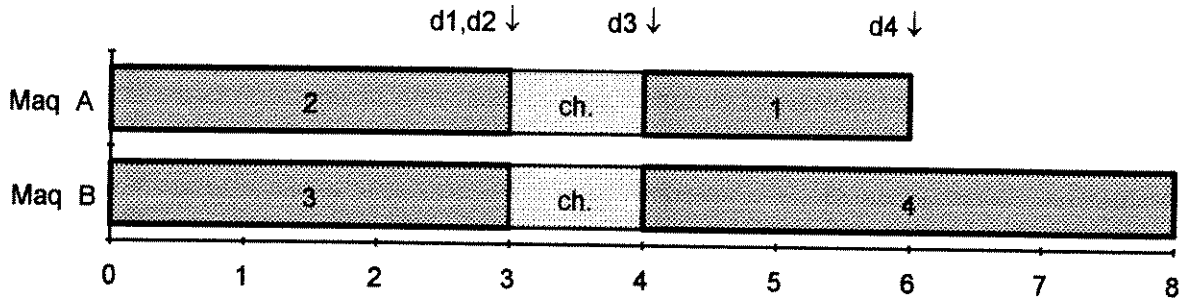


Fig 7. 6 - Exemplo com 4 tarefas em 2 máquinas

Os tempos de execução foram:

Modelo **S** - 3,5 seg, 1.558 iterações inteiras (modelo sem divisão de tarefas).

Modelo **C** - 231,6 seg, 68.643 iterações inteiras.

Modelo **C'**- 127,8 seg, 45.700 iterações inteiras.

Branch and Bound com LI1 - 1,4 seg, 383 nós gerados.

Os resultados destes pequenos exemplos comprovam a exatidão dos modelos desenvolvidos, bem como a sua flexibilidade, pois algumas alterações puderam ser introduzidas com grande facilidade (pelo menos nos modelos inteiro-mistos). Em todos os exemplos acima, confirmou-se também a superioridade computacional do modelo **C'** sobre o modelo **C**. O desempenho muito superior do método *Branch and Bound* em relação aos modelos de programação inteira-mista também merece ser apontada, porém, é preciso lembrar que o primeiro só examina programas de permutação, enquanto os modelos inteiro-mistos não têm esta limitação.

7.2 Geração dos parâmetros (Busca em Feixe Filtrada)

Devido à complexidade do modelo, alguns parâmetros serão mantidos fixos enquanto que os outros serão números inteiros escolhidos dentro de intervalos fixos. Por analogia com o problema clássico do Caxeiro Viajante, os tempos e custos de troca serão forçados a respeitar a desigualdade triangular de maneira a tornar a busca mais "comportada", isto é, facilitar a busca de uma boa "solução", aqui entendida como um grupo completo com seqüenciamentos e alocações definidos.

Em todos os problemas, β_j (coeficiente de atraso da tarefa j na função objetivo) será igual a 2 e α_j (coeficiente de adiantamento da tarefa j na função objetivo) será 25% do primeiro (0,5), para seguir um esquema semelhante ao de OW & MORTON (1988). Nos problemas da primeira parte, o custo de troca entre uma tarefa i e outra j será constante e numericamente igual à média de tempos de troca entre todas as tarefas e j. Na segunda parte dos exemplos, os custos de troca serão numericamente iguais aos tempos de troca respectivos entre duas tarefas. A tabela abaixo resume as escolhas dos parâmetros utilizados.

Tabela 7. 1 - Valores dos parâmetros dos exemplos

parâmetro	valor ou faixa de valores	forma de escolha
α_j	0,5	igual para todas as tarefas
β_j	2	igual para todas as tarefas
s_{jk}	[1, 10]	aleatório
c_{ijk}	[1, 5]	aleatório
q_j	[1, 10]	aleatório
c'_{jk}	$\frac{1}{(N-1)}\sum_{\substack{i\in T \\ i\neq j}} c_{ijk}$	independente da seqüência (1ª parte)
c'_{ijk}	c_{ijk}	dependente da seqüência (2ª parte)

As datas previstas d_j de cada tarefa também serão escolhidas aleatoriamente, porém dentro de faixas que dependerão dos parâmetros R e τ definidos a seguir.

7.3 *Parâmetros R e τ*

Estes parâmetros são amplamente utilizados para a geração controlada de conjuntos de problemas de programação envolvendo datas de entrega (OW & MORTON,1988, 1989 e DAVIS & KANET, 1993). Eles refletem aproximadamente o grau de dificuldade em encontrar boas soluções para um determinado problema. O parâmetro R é chamado de fator de faixa de datas previstas e mede a dispersão das datas previstas. O parâmetro τ, ou fator de atraso, fornece uma idéia da proporção de tarefas que deverão ficar atrasadas. Os parâmetros são definidos como:

$$R = \frac{\Delta d}{n\bar{p}} \qquad (7.1)$$

$$\tau = 1 - \frac{\bar{d}}{n\bar{p}} \qquad (7.2)$$

onde:

R = fator de faixa de datas previstas.

τ = fator de atraso.

Δd = diferença entre datas previstas máxima e mínima.

$\bar{d}$ = média das datas previstas.

$\bar{p}$ = média das tempos de processamento em uma única máquina.

n = número de tarefas em uma única máquina.

Para compreender melhor o significado das definições atuais, considere um exemplo em que todas as tarefas tem um tempo de processamento idêntico, as datas previstas estão igualmente afastadas e que as tarefas estão seqüenciadas segundo a ordem das datas previstas. Observe que $R = 0$ corresponde a datas previstas das tarefas iguais, $R = 1$ corresponde a uma folga entre datas previstas que deverá ser suficiente para que as tarefas as cumpram, desde que o instante inicial esteja razoavelmente afastado. Com $\tau = 0$ as tarefas poderão cumprir suas datas previstas e com $\tau = 1$, todas estarão atrasadas. É fácil mostrar que se $R + \tau > 1$ poderão aparecer datas previstas negativas, o que será evitado nos exemplos que serão desenvolvidos adiante.

Embora sejam definidos para uma única máquina, é razoavel generalizar as definições para o caso multimáquinas, uma vez que os métodos de solução dos programas multimáquinas

pode ser feito simplesmente substituindo o fator $n\bar{p}$ (número de tarefas vezes o tempo de processamento médio) presente nas duas definições, pelo somatório dos menores tempos de processamento possíveis para cada tarefa, utilizando ao máximo a divisão de tarefas. A justificativa é que nas boas soluções os tempos de processamento deverão se aproximar dos menores possíveis. Partindo das definições acima, definimos:

$$R^m = \frac{\Delta d}{\sum_{j \in T} a_{min}(j)} \qquad (7.3)$$

$$\tau^m = 1 - \frac{\bar{d}}{\sum_{j \in T} a_{min}(j)} \qquad (7.4)$$

onde:

R^m = fator de faixa de datas previstas para multimáquinas.

τ^m = fator de atrasos para multimáquinas.

Δd = diferença entre datas previstas máxima e mínima.

$\bar{d}$ = média das datas previstas.

$a_{min}(j)$ = menor tempo de processamento possível para a tarefa j, supondo-a sozinha no modelo. Igual a $z(j)$ calculado no item 6.2.2.

Portanto, uma vez definidos os valores dos parâmetros R^m e τ^m do exemplo, as expressões acima fornecem os valores de Δd e $\bar{d}$. A partir de um conjunto de N números aleatórios r_j, as datas previstas de cada tarefa podem ser calculadas pela seguinte transformação:

$$d_j = \frac{\Delta d}{\Delta r}(r_j - \bar{r}) + \bar{d} \qquad (7.5)$$

onde:

Δr = diferença entre o maior e o menor dos números aleatórios.

$\bar{r}$ = média dos números aleatórios.

Desta maneira, um exemplo gerado aleatoriamente poderá possuir precisamente os parâmetros fixados.

7.4 Exemplos formados a partir de grupos desacoplados

A idéia básica da utilização de grupos para a formação de um problema maior, é viabilizar a obtenção de soluções ótimas para problemas maiores, através da utilização do método *Branch and Bound* com LI1 ou LI2. Cada grupo possuirá 6 tarefas e nestes exemplos serão utilizadas apenas 2 ou 3 máquinas.

As tarefas de cada grupo possuirão as suas datas previstas de modo que as datas de um outro grupo deverão ser maiores que a maior do grupo ou menores do que a menor do grupo. Como foi mostrado no item 4.4.7 não é possível reduzir o valor da função objetivo ótima de um grupo, intercalando tarefas de um outro grupo (as desigualdades triangulares dos tempos de troca são respeitadas em todos os exemplos). Isto garante a obtenção de um gabarito ótimo e uma solução ótima para o problema completo, simplesmente juntando os gabaritos e soluções ótimas dos grupos que compõem o problema. Nestes exemplos, o custo de troca de cada tarefa não poderá depender da seqüência para que a propriedade de intercalação de tarefas seja válida. Com isso pode-se gerar um conjunto de problemas de teste com gabaritos e soluções ótimas conhecidas. Apesar de ser um conjunto particular, ele será de muita utilidade para a comparação da qualidade das heurísticas desenvolvidas na tese.

A seguir, um dos exemplos com 18 tarefas em 2 máquinas servirá como ilustração do procedimento empregado.

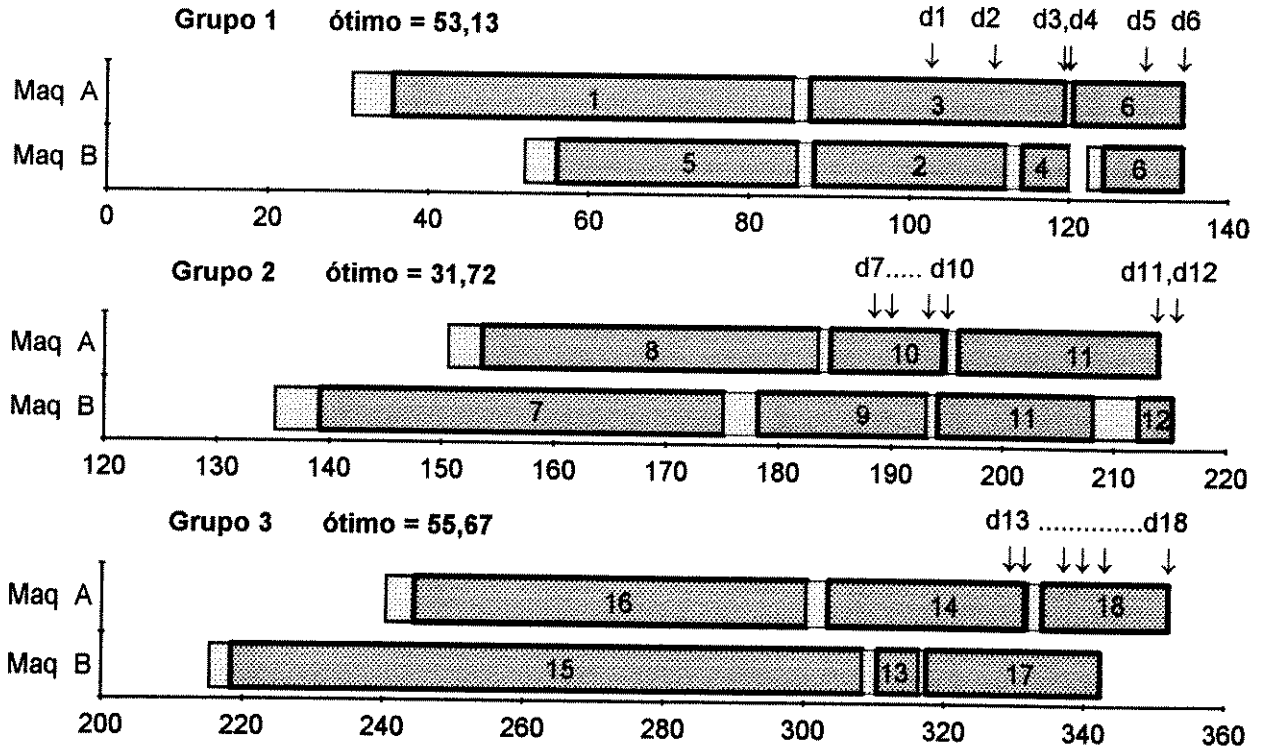


Fig 7. 7 - Grupos de tarefas componentes do exemplo 18X2 desacoplado

O primeiro dos grupos não tem restrições quanto às datas previstas de suas tarefas, mas nos outros grupos, estas devem ser suficientemente grandes para forçar o desacoplamento com t = 0, que é o instante de disponibilidade das máquinas em todos os exemplos. Uma vez obtidos os gabaritos ótimos e soluções ótimas, as datas previstas das tarefas de um mesmo grupo serão somadas ou subtraídas de uma constante de tal forma que as tarefas de grupos vizinhos fiquem bastante próximas sem se tocar (grupos desacoplados) quando montadas para a formação da solução ótima do grupo completo de tarefas. Somente o primeiro grupo manterá as datas prevista determinadas originalmente.

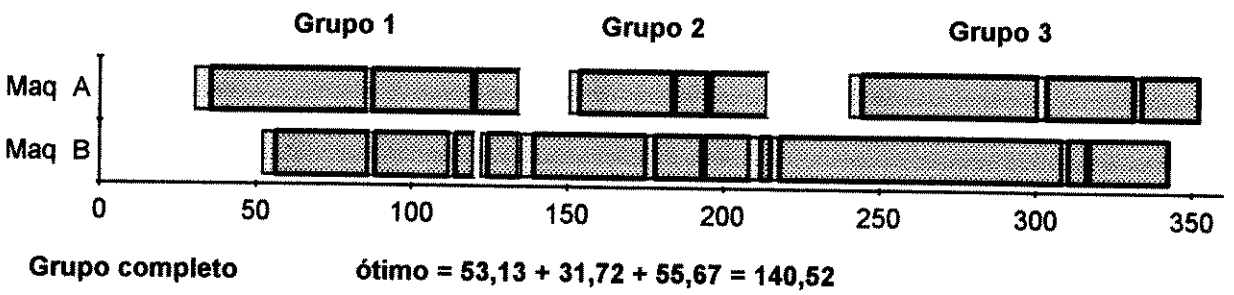


Fig 7. 8 - Grupo completo 18X2

Observe que os grupos só estarão desacoplados quando nos gabaritos ótimos e soluções ótimas. Isto porque os grupos foram colocados muito próximos entre si e se um gabarito ou solução diferentes dos ótimos forem tentados os limites de cada grupo poderão ser

ultrapassados e o acoplamento fatalmente ocorrerá. Durante o cálculo dos limitantes no procedimento de busca, provavelmente os grupos estarão acoplados. Isto é desejável, porque desta forma a busca sobre os exemplos desacoplados se aproxima do caso geral onde não há qualquer garantia de desacoplamento entre grupos de tarefas.

O procedimento de deslocamento de grupos descrito acima, altera os valores dos parâmetros R^m e t^m anteriormente definidos para cada grupo componente, tornando complexo o controle preciso destes após a formação do grupo completo. Por este motivo, não será feito um controle rigoroso destes parâmetros na primeira parte dos exemplos.

Para os exemplos desacoplados completos, foram construídos problemas com 18, 30, 60 e 120 tarefas em 2 e 3 máquinas, em um total de 8 problemas, cada um com 5 instâncias diferentes geradas a partir de grupos aleatórios de 6 tarefas, como descrito acima. Os resultados fornecidos nos gráficos e tabelas abaixo se referem à média das 5 instâncias.

7.5 Medidas de desempenho

Os resultados obtidos em cada um dos exemplos, constituem uma lista longa com o seqüenciamento, alocações, instante de início e término de cada fração, valor da funcão objetivo da solução ótima do grupo encontrado (*fo*) e tempo de busca em segundos. Neste trabalho, apenas medidas comparativas de *fo* em relação ao valor da função objetivo da solução ótima do gabarito ótimo do grupo (*ot*) e também em relação ao limitante superior de menor folga (*ubLL*), serão utilizadas para avaliar a qualidade da solução. O cálculo de *ot* para os exemplos desacoplados já foi explicado acima e para *ubLL* é empregado o seguinte procedimento:

1) Todas as tarefas são seqüenciadas segundo o critério de menor folga.
2) A partir da primeira tarefa da seqüência e em ordem crescente, as tarefas são alocadas unicamente à máquina em que o aumento da função objetivo for mínimo. Cada tarefa é posicionada, se possível com o tempo de término sobre a data prevista; se não, com o menor atraso possível (tempo ocioso nulo). O modelo **F**, de seqüenciamentos e alocações fixas, não é utilizado, isto é, uma vez alocadas e posicionadas as tarefas não se deslocam mais.

O valor da função objetivo da solução obtida desta maneira é chamado de *ubLL*. Alternativamente, um outro limitante superior obtido da mesma forma, porém partindo de um

seqüenciamento segundo as menores datas previstas, é chamado de *ubEDD*. Nos testes iniciais e posteriormente em todos os exemplos, verificou-se que *ubLL* é menor que *ubEDD* em um maior número de casos, o que levou à adoção do seqüenciamento segundo o critério de menor folga como o padrão a ser utilizado em todos os exemplos. Também para a filtragem, a característica global utilizada foi a folga da tarefa e não a data prevista.

As medidas comparativas utilizadas para aferir a qualidade de um programa obtido são:

$$fo\_ot = \frac{fo - ot}{ot} \times 100\% \qquad (7.6)$$

$$fo\_LL = \frac{fo - ubLL}{ubLL} \times 100\% \qquad (7.7)$$

onde:

fo_ot = desvio percentual em relação ao ótimo

fo_LL = desvio percentual em relação a *ubLL*

fo = valor da função objetivo na melhor solução alcançada pelo método proposto.

ot = valor da função objetivo na solução ótima do problema.

ubLL = valor da função objetivo da solução obtida por uma regra de despacho baseada na seqüência de menor folga.

A primeira das medidas de qualidade, *fo_ot*, jamais será negativa, pois *fo* não pode ser menor do que *ot*. Esta medida é muito empregada na literatura, quando as soluções ótimas são conhecidas.

A segunda das medidas de qualidade, compara a solução (*fo*) com uma outra obtida por uma regra de despacho (*ubLL*). Esta medida foi desenvolvida com base no trabalho de OW & MORTON (1988), que destaca a importância de "descobrir o tamanho da vantagem em termos de custo que o esforço adicional de busca do método de busca em feixe representa, comparada com a busca deficiente do método de despacho." Neste caso, as boas soluções apresentarão *fo_LL* negativo, pois qualquer solução com esta medida positiva é pior que a obtida pela regra de despacho.

Outra medida utilizada é o *splitting* (*sp*) da solução, que avaliará a quantidade de tarefas que foram submetidas a divisão em mais de uma máquina e, portanto, fornece uma medida sobre a oportunidade de se divir as tarefas.

$$sp = \frac{nf - N}{N} \times 100\% \qquad (7.8)$$

onde:

sp = índice da divisão de tarefas (*splitting*) da solução.

nf = número total de frações do grupo completo.

N = número de tarefas.

Esta medida será igual a 0% se nenhuma tarefa for dividida em mais de uma máquina e chegará a 100% em um exemplo de 2 máquinas com todas as tarefas divididas em ambas as máquinas.

A última das medidas de desempenho é o tempo de busca em segundos. Este não inclui o tempo gasto com leitura do arquivo de dados e inicializações anteriores à busca. Nos exemplos de 60 e 120 tarefas o tempo é incrementado devido aos acessos a disco (arquivos RAMDISK ou SWAP), mas servem como comparação entre exemplos de mesmo tamanho e mesmo valor de Bm ou como referência sem muita precisão. Todos os exemplos foram executados em um microcomputador IBM PC Pentium 90 Mhz com 16 Mbytes de RAM, programados em linguagem C e compilados com o IBM C/C++ para OS/2 (32 bits).

7.6 Sensibilidade à variação dos parâmetros da busca.

Como um primeiro teste de desempenho, foi avaliada a sensibilidade aos parâmetros Bm (largura do feixe de busca) e Ft (filtragem) em cada um dos problemas-exemplo, para que os melhores ajustes sejam utilizados na segunda parte dos exemplos. Cada uma das instâncias foi submetida aos quatro limitantes (L3, L4, L5 e L6) desenvolvidos anteriormente para busca em feixe filtrada. Alguns dos problemas de 60 e 120 tarefas não foram executados com limitantes lentos (L3 e L5) ou ajustes de Bm grandes, devido ao tempo de processamento excessivo (acima de 1200 segundos).

Segundo OW & MORTON (1988) quanto maior a largura de feixe (Bm), maior a segurança quanto à obtenção de um bom resultado, mas no caso da filtragem (Ft) é preciso fazer uma investigação mais apurada. Para a padronização dos resultados, foram adotados os seguintes valores para Bm e Ft:

Bm - 1, 2, 4, 8, 16.

Ft - 100%, 40%, 20%, 10%, 5%, 2%, 1%.

As sensibilidades foram avaliadas de maneira aproximada da seguinte maneira:

1) Com o maior valor de Bm que não penalize em excesso o tempo de processsamento, Ft é testado em toda a faixa acima.
2) Bm é variado de 1 até um valor que não implique em um tempo de processamento excessivo. Nestes testes Ft é feito igual ao que forneceu o melhor dos resultados no item 1) acima.

Os testes de sensibilidade mostrados nos gráficos abaixo foram executados com os exemplos da primeira parte (grupos desacoplados) e no eixo vertical dos gráficos está o desvio percentual em relação ao valor da função objetivo ótima (*ot*). Todos os resultados se referem à média das 5 instâncias de cada problema.

Legenda:

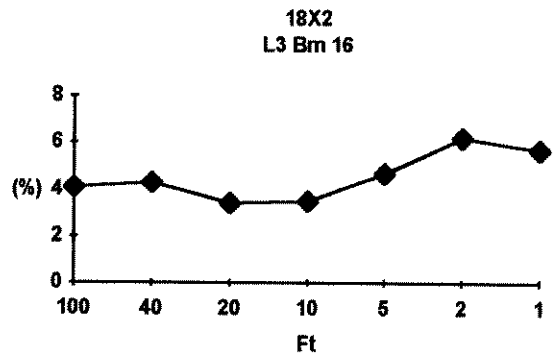
NXM - número total de tarefas X número de máquinas.

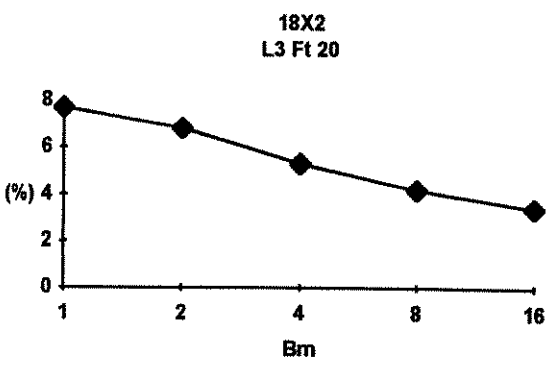
Bm - Largura de Feixe (*Beam*) .

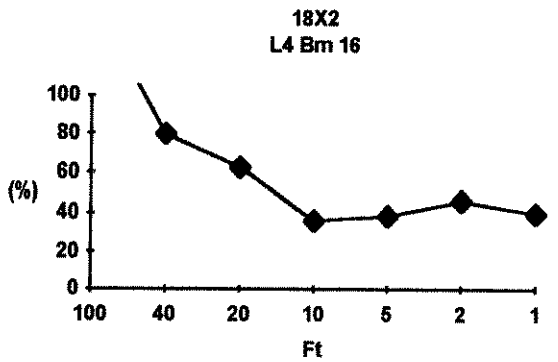
Ft - Filtragem como porcentagem da diferença entre o maior e o menor das características globais, como definido na expressão (6. 1).

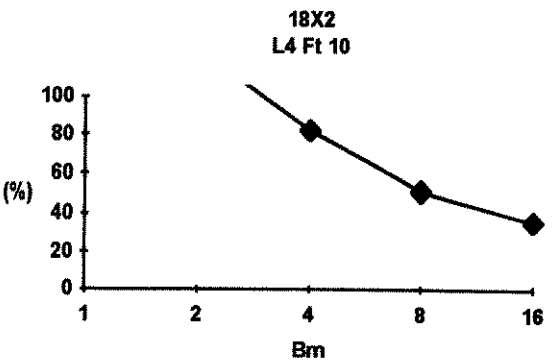
L3, L4, L5 e L6 - Limitante utilizado na busca.

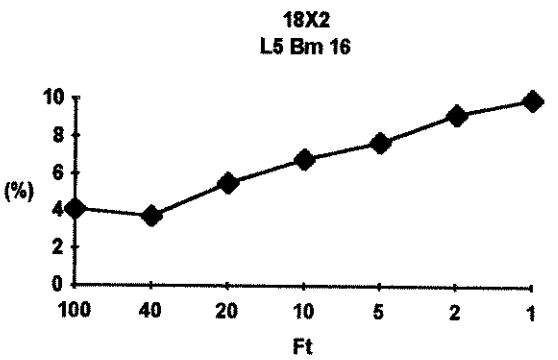
eixo vertical - desvio percentual médio em relação ao ótimo (*fo_ot*, expressão (7. 6)).

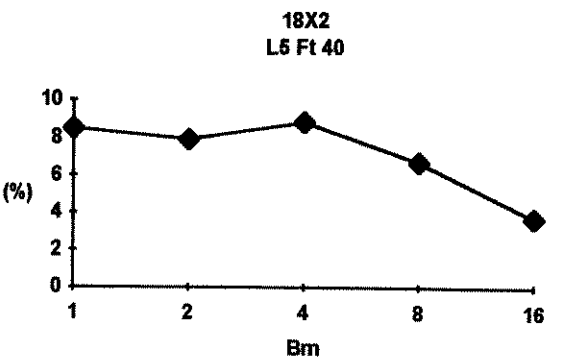

18X2
L3 Bm 16
(%)
8
6
4
2
0
100
40
20
10
5
2
1
Ft

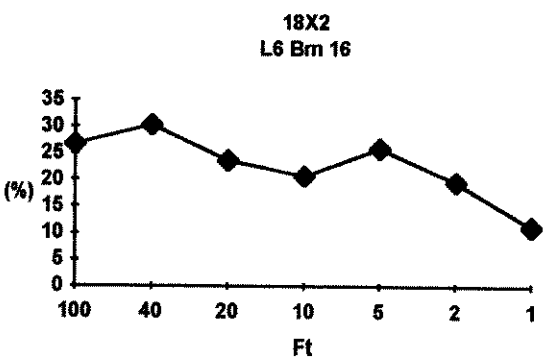

18X2
L3 Ft 20
(%)
8
6
4
2
0
1
2
4
8
16
Bm

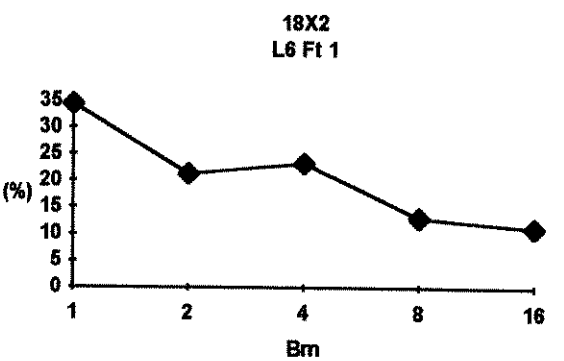

18X2
L4 Bm 16
(%)
100
80
60
40
20
0
100
40
20
10
5
2
1
Ft

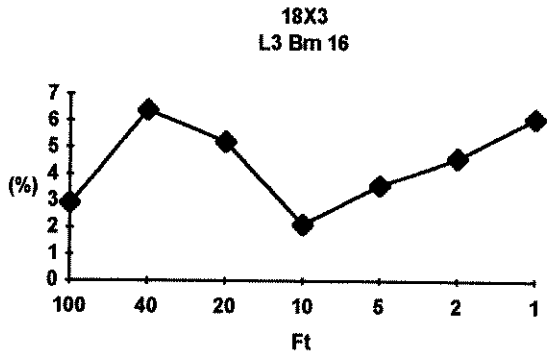

18X2
L4 Ft 10
(%)
100
80
60
40
20
0
1
2
4
8
16
Bm

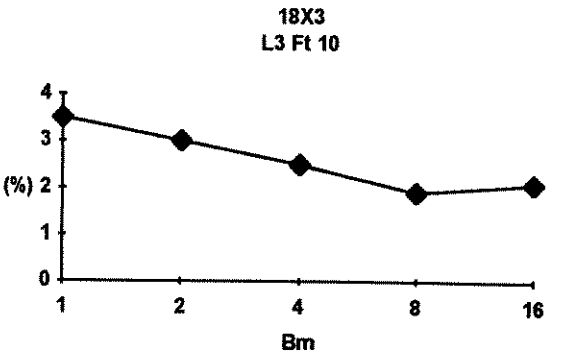

18X2
L5 Bm 16
(%)
10
8
6
4
2
0
100
40
20
10
5
2
1
Ft

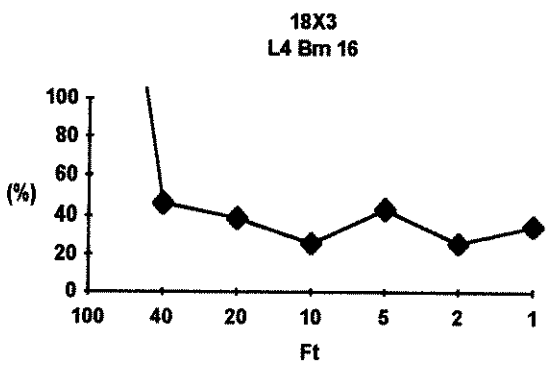

18X2
L5 Ft 40
(%)
10
8
6
4
2
0
1
2
4
8
16
Bm

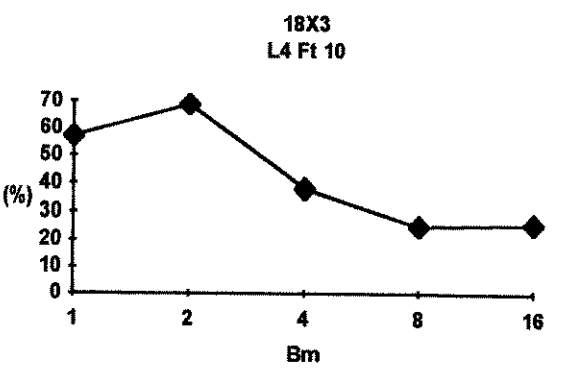

18X2
L6 Bm 16
(%)
35
30
25
20
15
10
5
0
100
40
20
10
5
2
1
Ft

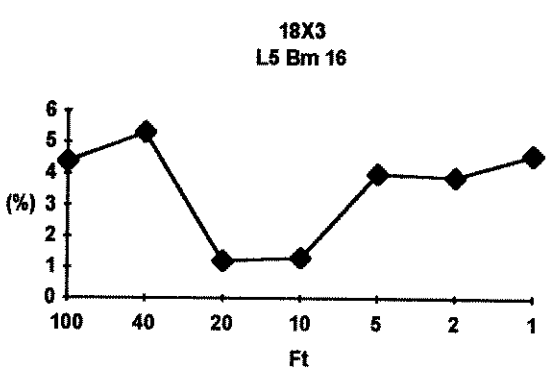

18X2
L6 Ft 1
(%)
35
30
25
20
15
10
5
0
1
2
4
8
16
Bm

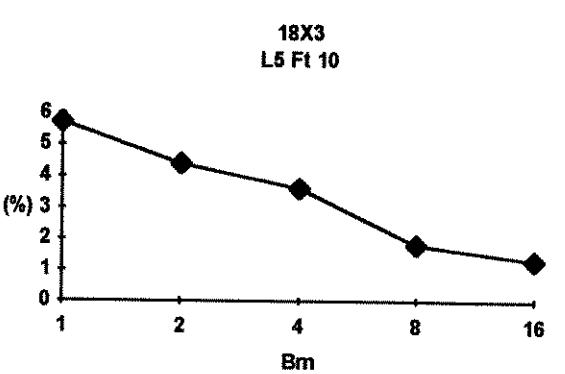

18X3
L3 Bm 16
(%)
0
1
2
3
4
5
6
7
100
40
20
10
5
2
1
Ft

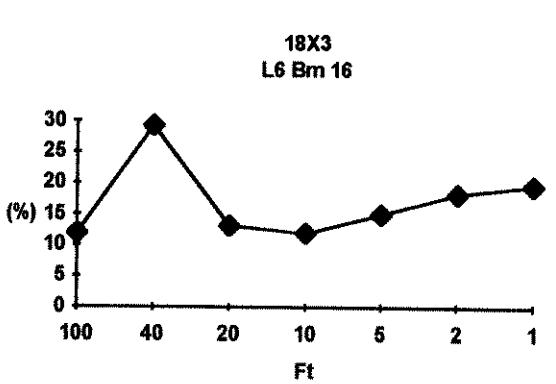

18X3
L3 Ft 10
(%)
0
1
2
3
4
1
2
4
8
16
Bm

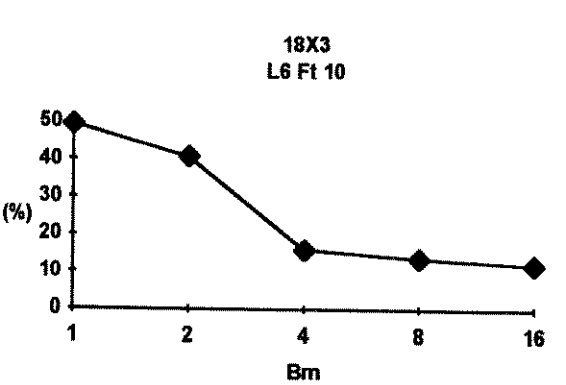

18X3
L4 Bm 16
(%)
0
20
40
60
80
100
100
40
20
10
5
2
1
Ft


18X3
L4 Ft 10
(%)
0
10
20
30
40
50
60
70
1
2
4
8
16
Bm

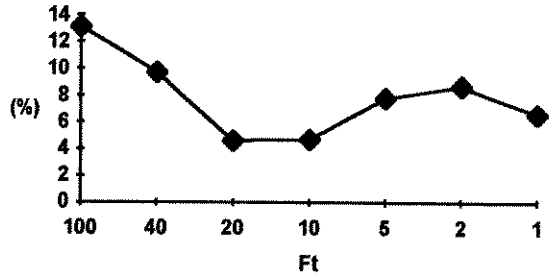

18X3
L5 Bm 16
(%)
0
1
2
3
4
5
6
100
40
20
10
5
2
1
Ft


18X3
L5 Ft 10
(%)
0
1
2
3
4
5
6
1
2
4
8
16
Bm

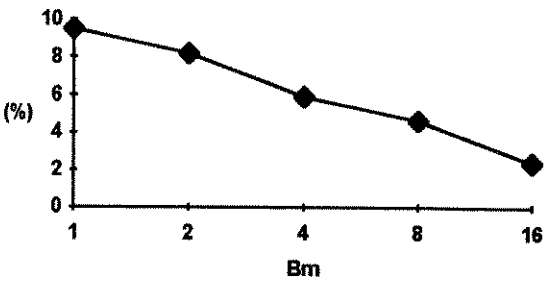

18X3
L6 Bm 16
(%)
0
5
10
15
20
25
30
100
40
20
10
5
2
1
Ft


18X3
L6 Ft 10
(%)
0
10
20
30
40
50
1
2
4
8
16
Bm

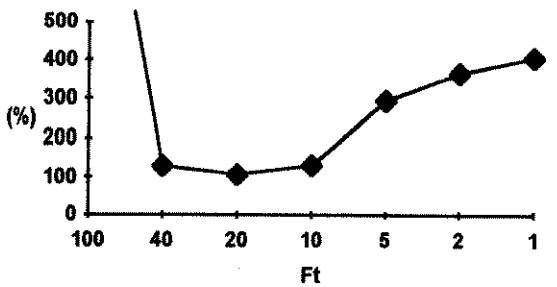
30X2
L3 Bm 8



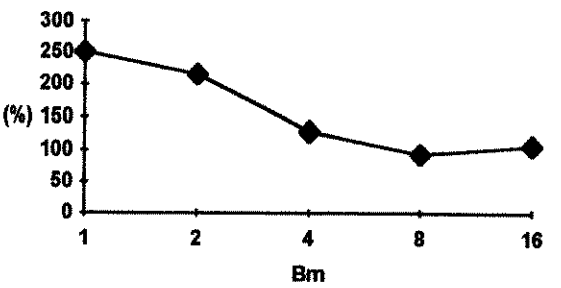
30X2
L3 Ft 20



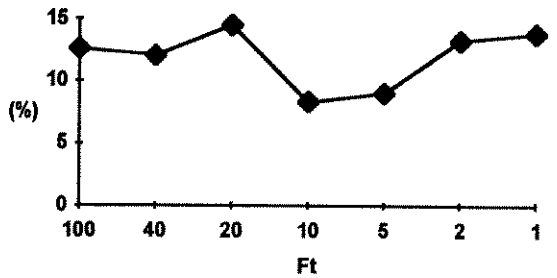
30X2
L4 Bm 16



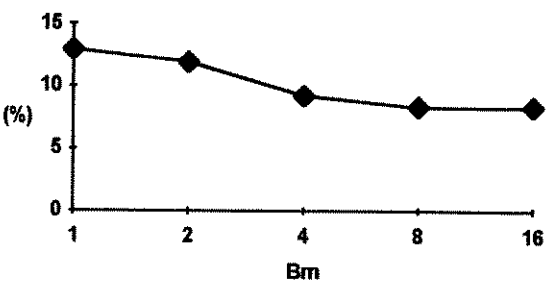
30X2
L4 Ft 20



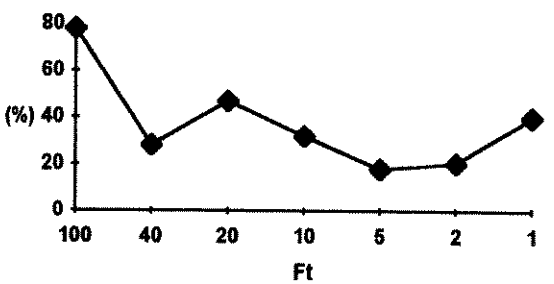
30X2
L5 Bm 8



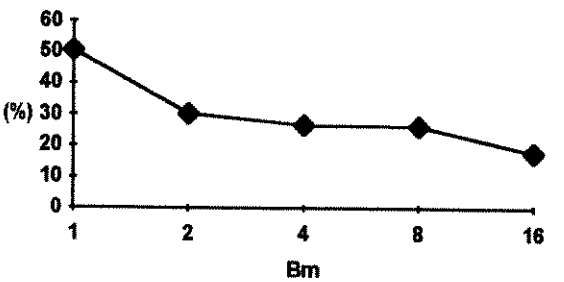
30X2
L5 Ft 10



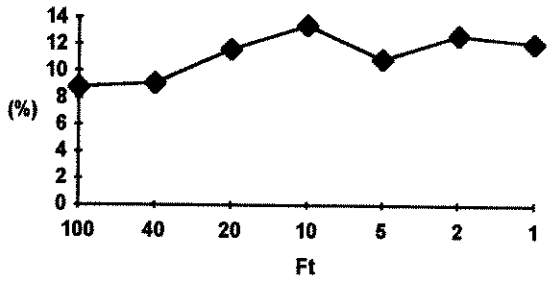
30X2
L6 Bm 16



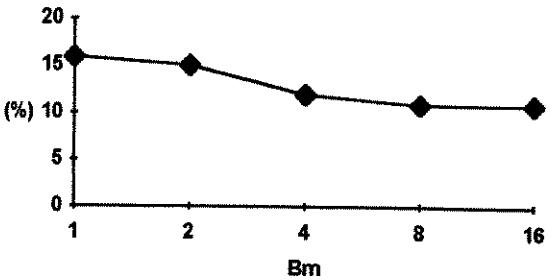
30X2
L6 Ft 5



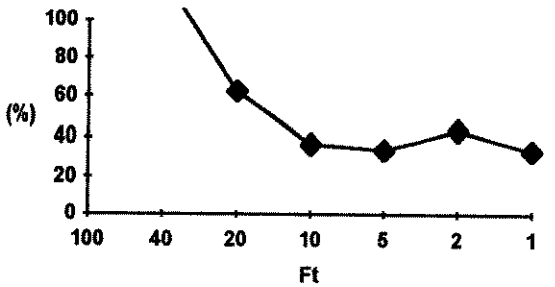
30X3
L3 Bm 8



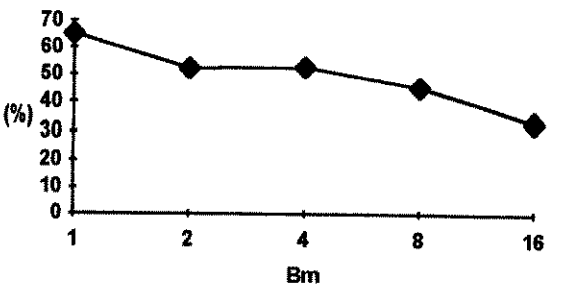
30X3
L3 Ft 5



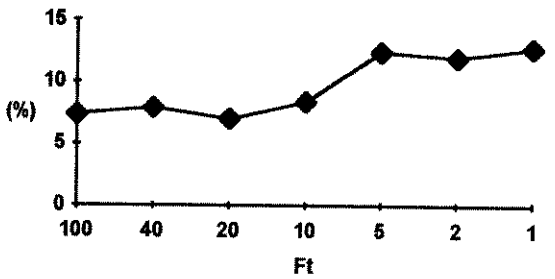
30X3
L4 Bm 16



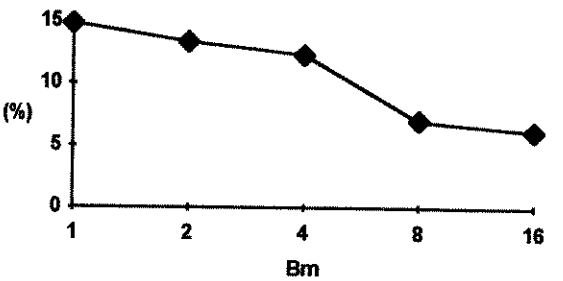
30X3
L4 Ft 1



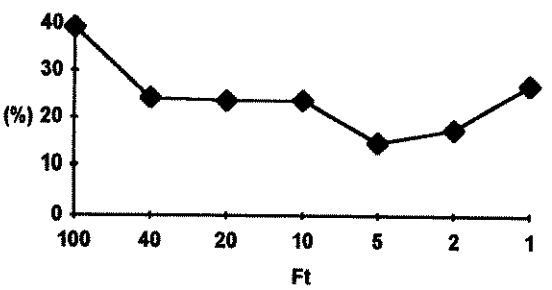
30X3
L5 Bm 8



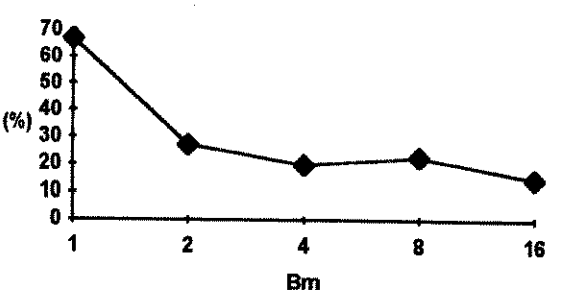
30X3
L5 Ft 20

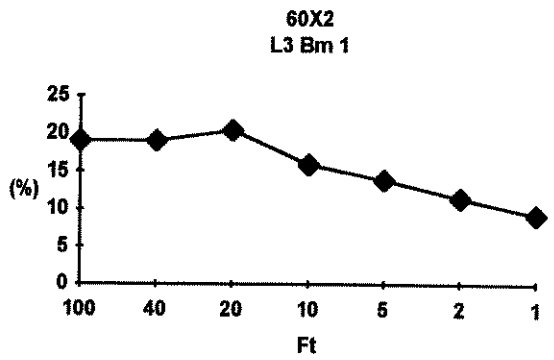


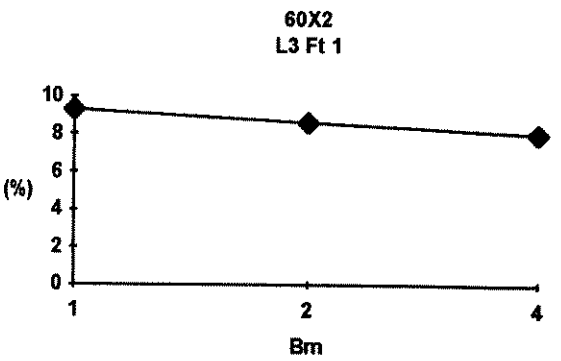
30X3
L6 Bm 16

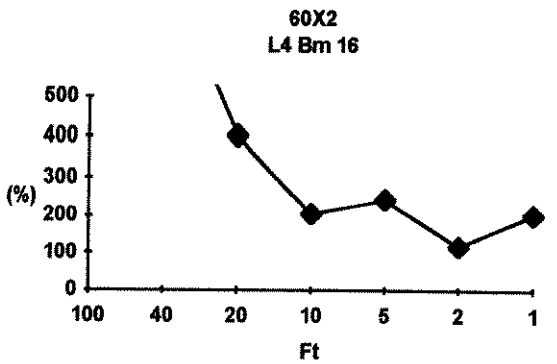


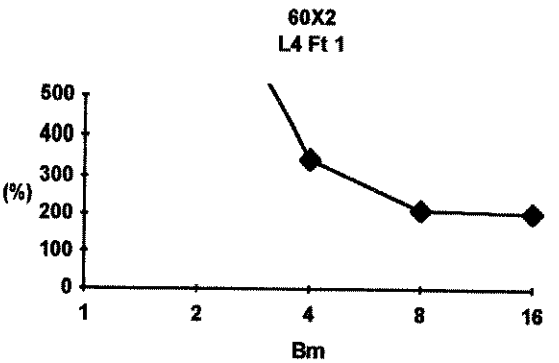
30X3
L6 Ft 5

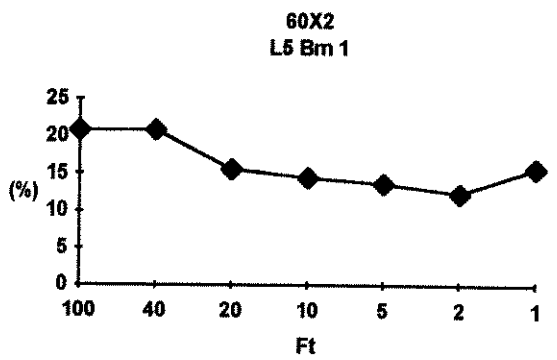


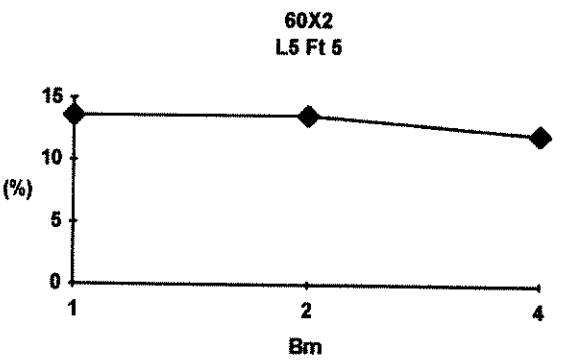

60X2
L3 Bm 1
25
20
15
10
5
0
(%)
100
40
20
10
5
2
1
Ft

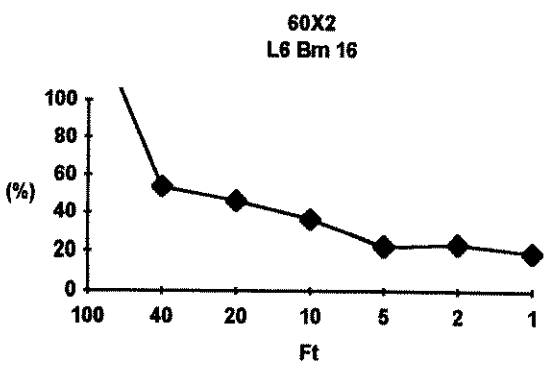

60X2
L3 Ft 1
10
8
6
4
2
0
(%)
1
2
4
Bm

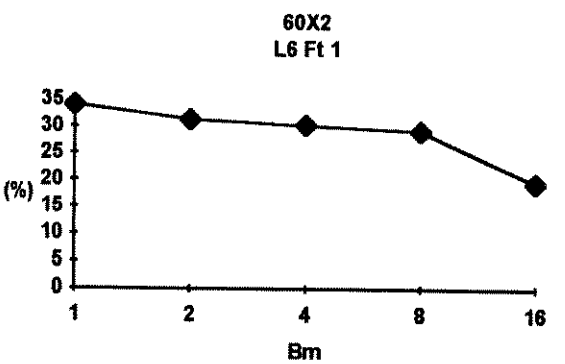

60X2
L4 Bm 16
500
400
300
200
100
0
(%)
100
40
20
10
5
2
1
Ft

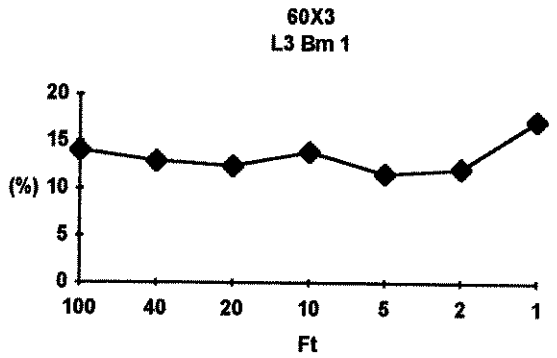

60X2
L4 Ft 1
500
400
300
200
100
0
(%)
1
2
4
8
16
Bm

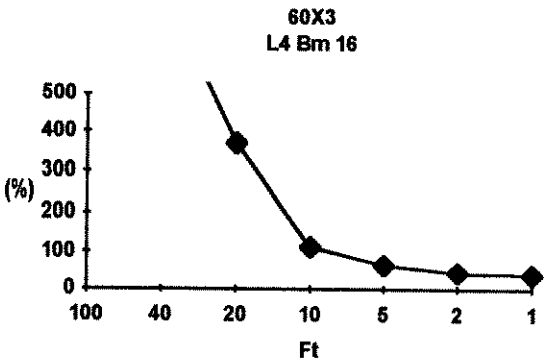

60X2
L5 Bm 1
25
20
15
10
5
0
(%)
100
40
20
10
5
2
1
Ft

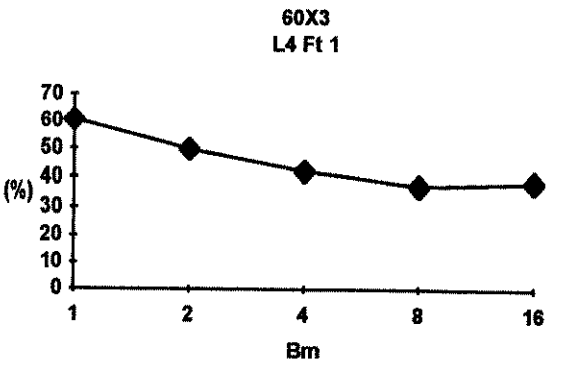

60X2
L5 Ft 5
15
10
5
0
(%)
1
2
4
Bm

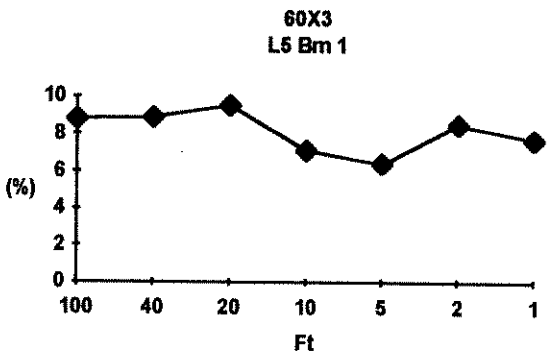

60X2
L6 Bm 16
100
80
60
40
20
0
(%)
100
40
20
10
5
2
1
Ft

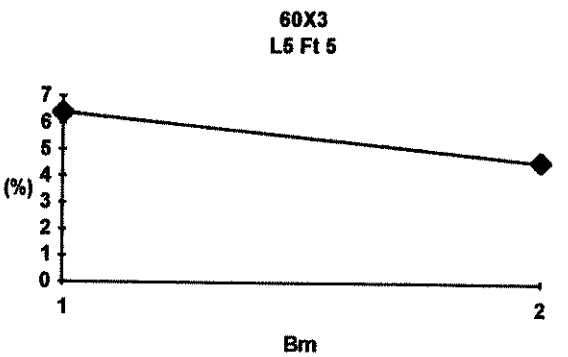

60X2
L6 Ft 1
35
30
25
20
15
10
5
0
(%)
1
2
4
8
16
Bm

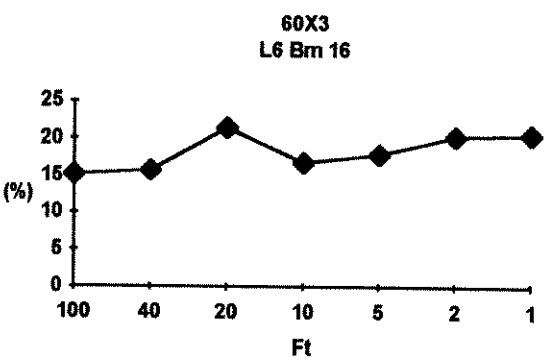

60X3
L3 Bm 1
20
15
(%) 10
5
0
100 40 20 10 5 2 1
Ft

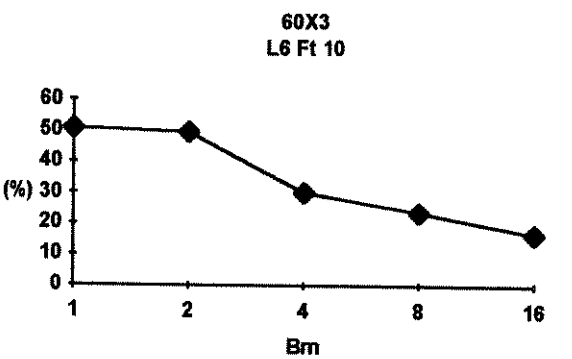

60X3
L4 Bm 16
500
400
300
(%) 200
100
0
100 40 20 10 5 2 1
Ft


60X3
L4 Ft 1
70
60
50
(%) 40
30
20
10
0
1 2 4 8 16
Bm

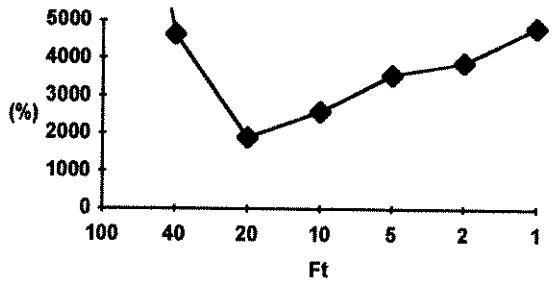

60X3
L5 Bm 1
10
8
(%) 6
4
2
0
100 40 20 10 5 2 1
Ft


60X3
L5 Ft 5
7
6
5
(%) 4
3
2
1
0
1 2
Bm

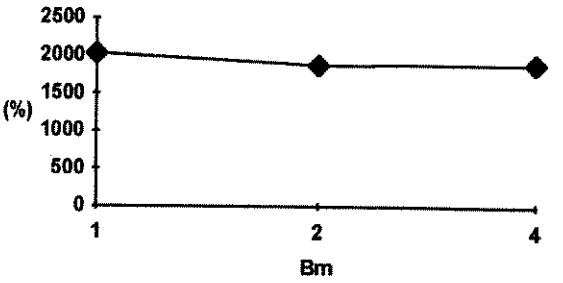

60X3
L6 Bm 16
25
20
15
(%) 10
5
0
100 40 20 10 5 2 1
Ft


60X3
L6 Ft 10
60
50
40
(%) 30
20
10
0
1 2 4 8 16
Bm

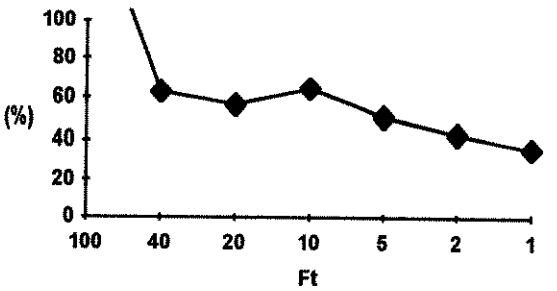
120X2
L4 Bm 4



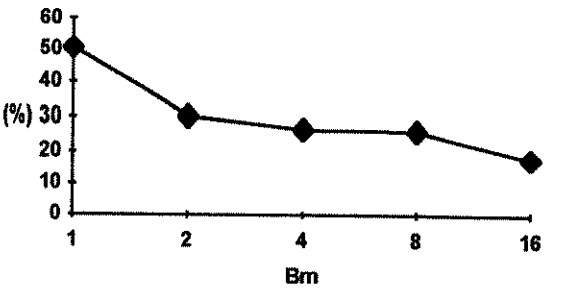
120X2
L4 Ft 20



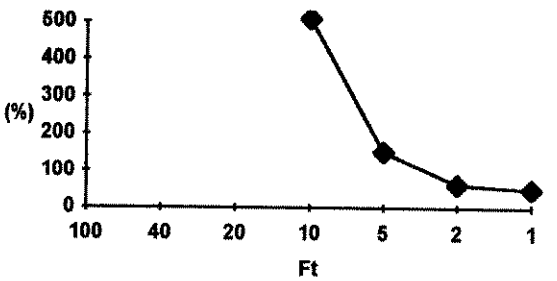
120X2
L6 Bm 4



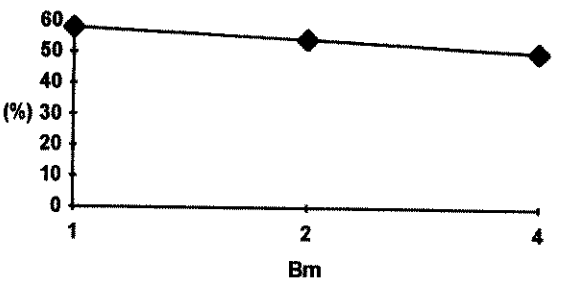
120X2
L6 Ft 1



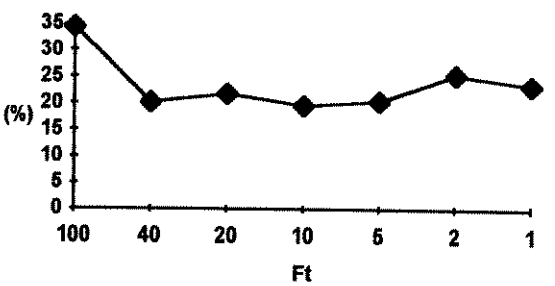
120X3
L4 Bm 4



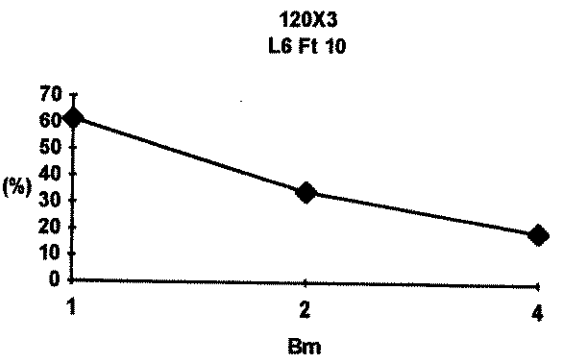
120X3
L4 Ft 1



120X3
L6 Bm 4



120X3
L6 Ft 10



Os gráficos mostram como é importante utilizar limitantes elaborados na busca, pois mesmo pequenas alterações na qualidade destes, como é o caso de L5 e L6, pode alterar significativamente os resultados. Simplificações maiores nos limitantes, como no caso de L4, podem ser desastrosas. Os gráficos também deixam claro que a filtragem melhora substancialmente o desempenho do método, ou reduzindo drásticamente o tempo de busca, ou mesmo melhorando a qualidade do resultado final. A melhoria de qualidade da solução com o aumento de Bm, já era esperada. O tempo de processamento cresce com o aumento de Bm e Ft, também como seria de se esperar.

7.7 Melhores resultados - 1ª parte

As tabelas abaixo resumem os melhores resultados para cada limitante em cada instância, incluindo informações sobre o tempo de processamento e "splitting" da solução encontrada.

Para L3:

EXEMPLO	18X2 Bm 16 Ft 20	30X2 Bm 16 Ft 20	60X2 Bm 4 Ft 1	
média *fo_ot*	3,4%	2,4%	8,0%	
média *fo_LL*	-74,3%	-81,2%	-76,7%	
média *sp*	5,6%	4,7%	2,7%	
Tempo médio (seg)	56,5	516,7	810,0	
EXEMPLO	18X3 Bm 8 Ft 10	30X3 Bm 8 Ft 5	60X3 Bm 1 Ft 5	
média *fo_ot*	1,9%	10,9%	11,6%	
média *fo_LL*	-64,3%	-59,4%	-63,1%	
média *sp*	1,1%	3,3%	4,3%	
Tempo médio (seg)	66,9	315,5	1079,6	

Para L5:

EXEMPLO	18X2 Bm 16 Ft 40	30X2 Bm 8 Ft 10	60X2 Bm 4 Ft 5	
média *fo_ot*	3,7%	8,3%	12,1%	
média *fo_LL*	-74,3%	-80,3%	-76,2%	
média *sp*	0,0%	4,0%	3,7%	
Tempo médio (seg)	40,2	89,1	868,6	
EXEMPLO	18X3 Bm 16 Ft 10	30X3 Bm 16 Ft 20	60X3 Bm 2 Ft 5	
média *fo_ot*	1,3%	6,2%	4,6%	
média *fo_LL*	-64,6%	-61,1%	-65,3%	
média *sp*	0,0%	2,0%	2,3%	
Tempo médio (seg)	58,6	538,2	1097,1	

Para L4:

EXEMPLO	18X2 Bm 16 Ft 10	30X2 Bm 8 Ft 20	60X2 Bm 16 Ft 2	120X2 Bm 2 Ft 20
média *fo_ot*	35,9%	93,4%	117,8%	1872,2%
média *fo_LL*	-66,3%	-66,2%	-54,2%	+245,9%
média *sp*	6,7%	2,0%	3,0%	0,0%
Tempo médio (seg)	1,1	2,1	26,6	41,1
EXEMPLO	18X3 Bm 8 Ft 10	30X3 Bm 16 Ft 1	60X3 Bm 8 Ft 1	120X3 Bm 4 Ft 1
média *fo_ot*	25,1%	33,6%	37,4%	50,2%
média *fo_LL*	-56,2%	-50,9%	-54,3%	-51,0%
média *sp*	14,4%	13,3%	19,7%	16,8%
Tempo médio (seg)	1,1	4,0	14,5	78,1

Para L6:

EXEMPLO	18X2 Bm 16 Ft 1	30X2 Bm 16 Ft 5	60X2 Bm 16 Ft 1	120X2 Bm 4 Ft 1
média *fo_ot*	11,1%	17,9%	19,7%	35,9%
média *fo_LL*	-72,4%	-78,6%	-74,2%	-75,6%
média *sp*	13,3%	6,7%	6,7%	10,7%
Tempo médio (seg)	0,8	3,3	26,2	62,4
EXEMPLO	18X3 Bm 16 Ft 10	30X3 Bm 16 Ft 5	60X3 Bm 16 Ft 10	120X3 Bm 4 Ft 10
média *fo_ot*	11,9%	14,8%	16,7%	19,5%
média *fo_LL*	-60,8%	-57,9%	-61,5%	-61,1%
média *sp*	11,1%	12,0%	6,3%	4,0%
Tempo médio (seg)	1,8	5,2	44,6	101,4

É interessante observar a baixa média de divisão de tarefas nas tabelas de resultados acima, como conseqüência da formação destes exemplos através de grupos de tarefas desacoplados. Como os grupos são relativamente pequenos em relação ao número de máquinas, a tendência é que as tarefas não concorram muito entre si, ficando alocadas em uma única máquina e bem distribuídas. Isto também foi observado nas soluções ótimas de cada um dos grupos, onde a divisão de tarefas é pequena nos grupos com 2 máquinas e praticamente nula nos grupos com 3 máquinas. No caso de L4 e L6, a divisão de tarefa foi ligeiramente maior, o que pode ser explicado pela qualidade inferior destes limitantes, levando o método a decidir de forma imprecisa pela divisão quando o melhor teria sido a não divisão das tarefas.

7.8 Exemplos acoplados - 2ª parte

Os exemplos a seguir foram gerados sem qualquer preocupação com o acoplamento entre grupos de tarefas, sendo o custo de troca entre duas tarefas dependente da seqüência. Como conseqüência, não se conhecem os gabaritos ou soluções ótimas. Nestes casos, apenas *fo_LL* será utilizada para avaliação da qualidade dos resultados obtidos. Algumas instâncias onde se previa que o tempo de processamento seria excessivamente longo não foram executadas. Cada instância foi executada com os melhores ajustes (Bm e Ft) verificados na 1ª parte para 2 máquinas e também com o ajuste para 3 máquinas, mesmo número de tarefas e mesmo limitante, isto é, normalmente dois ajustes para cada. Somente os resultados do melhor dos dois ajustes (que não é necessariamente o melhor possível), foi colocado nas tabelas abaixo, exceto em alguns casos onde o valor de Bm foi reduzido para que o tempo de processamento ficasse dentro de limites aceitáveis. Por exemplo, o problema 30X2 para L3 nas tabelas abaixo foi executado com Bm = 16 e Ft = 20, que é o ajuste para o problema 30X2 desacoplado para L3 (item 7.7) e também foi executado com Bm = 8 e Ft = 5, que é o ajuste para o problema 30X3 desacoplado para L3. No caso $\tau^m = 0$ e $R^m = 0{,}33$, o melhor ajuste foi Bm = 16, Ft = 20. No caso $\tau^m = 0$ e $R^m = 0{,}66$, o melhor ajuste foi Bm = 8, Ft = 5.

Para L3 $\tau^m = 0$ $R^m = 0{,}33$

EXEMPLO/Ajuste	18X2 Bm 16 Ft 20	30X2 Bm 16 Ft 20	60X2 Bm 1 Ft 5
Média fo_LL	-87,30%	-85,40%	-84,90%
Média sp	7,80%	4,70%	10,70%
Média tempo	69,8	663,1	580,1
EXEMPLO/Ajuste	18X4 Bm 16 Ft 20	30X4 Bm 8 Ft 5	
Média fo_LL	-74,40%	-76,10%	
Média sp	11,10%	29,30%	
Média tempo	896,4	2612,7	
EXEMPLO/Ajuste	18X6 Bm 16 Ft 20	30X6 Bm 2 Ft 5	
Média fo_LL	-57,10%	-58,00%	
Média sp	11,10%	51,30%	
Média tempo	4274,2	4485,3	

Para L5 $\tau^m = 0$ $R^m = 0{,}33$

EXEMPLO/Ajuste	18X2 Bm 16 Ft 40	30X2 Bm 8 Ft 10	60X2 Bm 4 Ft 5
Média fo_LL	-86,90%	-82,9%	-84,30%
Média sp	10,00%	11,30%	10,30%
Média tempo	51,1	107,6	1046,7
EXEMPLO/Ajuste	18X4 Bm 16 Ft 40	30X4 Bm 8 Ft 10	60X4 Bm 2 Ft 5
Média fo_LL	-76,80%	-75,40%	-83,70%
Média sp	13,30%	16,70%	17,70%
Média tempo	283,9	604,9	2958,4
EXEMPLO/Ajuste	18X6 Bm 16 Ft 40	30X6 Bm 4 Ft 10	
Média fo_LL	-62,90%	-68,00%	
Média sp	7,80%	18,00%	
Média tempo	1324,6	1298	

Para L4 $\tau^m = 0$ $R^m = 0{,}33$

EXEMPLO/Ajuste	18X2 Bm 8 Ft 10	30X2 Bm 16 Ft 1	60X2 Bm 16 Ft 1	120X2 Bm 4 Ft 20
Média fo_LL	-24,90%	-60,70%	-15,80%	-7,60%
Média sp	0,00%	4,00%	0,00%	0,00%
Média tempo	0,5	2,5	28,1	92,4
EXEMPLO/Ajuste	18X4 Bm 16 Ft 10	30X4 Bm 8 Ft 20	60X4 Bm 8 Ft 1	120X4 Bm 4 Ft 20
Média fo_LL	-54,30%	-50,90%	-55,50%	-39,00%
Média sp	31,10%	40,70%	34,70%	26,80%
Média tempo	4,1	10,8	28,4	359,5
EXEMPLO/Ajuste	18X6 Bm 8 Ft 10	30X6 Bm 16 Ft 1	60X6 Bm 4 Ft 1	120X6 Bm 2 Ft 1
Média fo_LL	-53,70%	-44,70%	-40,50%	-38,90%
Média sp	32,20%	54,00%	75,00%	87,80%
Média tempo	9,5	40,5	59,2	256,5

Para L6 $\tau^m = 0$ $R^m = 0{,}33$

EXEMPLO/Ajuste	18X2 Bm 16 Ft 1	30X2 Bm 16 Ft 5	60X2 Bm 16 Ft 10	120X2 Bm 4 Ft 10
Média fo_LL	-84,60%	-75,30%	-83,90%	-83,10%
Média sp	12,20%	3,30%	6,70%	9,50%
Média tempo	0,9	3,1	41,1	100
EXEMPLO/Ajuste	18X4 Bm 16 Ft 1	30X4 Bm 16 Ft 5	60X4 Bm 16 Ft 10	120X4 Bm 4 Ft 10
Média fo_LL	-70,40%	-72,50%	-81,80%	-81,70%
Média sp	22,20%	23,30%	24,70%	37,20%
Média tempo	2,1	10,3	117,2	299,6
EXEMPLO/Ajuste	18X6 Bm 16 Ft 10	30X6 Bm 16 Ft 5	60X6 Bm 4 Ft 10	120X6 Bm 2 Ft 10
Média fo_LL	-57,60%	-62,70%	-72,50%	-75,60%
Média sp	22,20%	32,70%	41,70%	80,70%
Média tempo	13	40,7		

Para L3 $\tau^m = 0$ $R^m = 0{,}66$

EXEMPLO/Ajuste	18X2 Bm 8 Ft 10	30X2 Bm 8 Ft 5	60X2 Bm 2 Ft 1
Média fo_LL	-80,70%	-84,00%	-82,60%
Média sp	12,20%	12,00%	12,30%
Média tempo	27,7	148,3	622
EXEMPLO/Ajuste	18X4 Bm 16 Ft 20	30X4 Bm 8 Ft 5	
Média fo_LL	-60,70%	-68,80%	
Média sp	15,60%	14,70%	
Média tempo	661,4	1477,1	
EXEMPLO/Ajuste	18X6 Bm 16 Ft 20	30X6 Bm 2 Ft 5	
Média fo_LL	-32,70%	-46,70%	
Média sp	3,30%	16,00%	
Média tempo	2341,5	1101,9	

Para L5 $\tau^m = 0$ $R^m = 0{,}66$

EXEMPLO/Ajuste	18X2 Bm 16 Ft 40	30X2 Bm 8 Ft 10	60X2 Bm 2 Ft 5
Média fo_LL	-81,40%	-84,00%	-81,70%
Média sp	6,70%	11,30%	12,30%
Média tempo	51,3	90,2	520,2
EXEMPLO/Ajuste	18X4 Bm 16 Ft 40	30X4 Bm 8 Ft 10	60X4 Bm 2 Ft 5
Média fo_LL	-65,50%	-70,60%	-73,20%
Média sp	3,30%	9,30%	18,70%
Média tempo	274,8	443,9	2569,6
EXEMPLO/Ajuste	18X6 Bm 16 Ft 40	30X6 Bm 4 Ft 10	
Média fo_LL	-33,40%	-50,70%	
Média sp	3,30%	6,00%	
Média tempo	1186,4	1065,2	

Para L4 $\tau^m = 0$ $R^m = 0{,}66$

EXEMPLO/Ajuste	18X2 Bm 8 Ft 10	30X2 Bm 16 Ft 1	60X2 Bm 8 Ft 1	120X2 Bm 4 Ft 20
Média fo_LL	-29,00%	-5,20%	76,90%	71,00%
Média sp	1,10%	3,30%	0,00%	0,00%
Média tempo	0,6	2,4	8,6	90,4
EXEMPLO/Ajuste	18X4 Bm 8 Ft 10	30X4 Bm 16 Ft 1	60X4 Bm 16 Ft 1	120X4 Bm 4 Ft 1
Média fo_LL	-54,70%	-50,40%	-11,70%	11,80%
Média sp	17,80%	30,00%	42,70%	54,30%
Média tempo	1,8	8,4	77,4	185
EXEMPLO/Ajuste	18X6 Bm 16 Ft 10	30X6 Bm 16 Ft 1	60X6 Bm 4 Ft 1	120X6 Bm 2 Ft 1
Média fo_LL	-27,90%	-43,40%	-13,60%	5,10%
Média sp	12,20%	21,30%	76,00%	80,80%
Média tempo	15,4	31,1	51,1	211

Para L6 $\tau^m = 0$ $R^m = 0{,}66$

EXEMPLO/Ajuste	18X2 Bm 16 Ft 1	30X2 Bm 16 Ft 5	60X2 Bm 16 Ft 10	120X2 Bm 4 Ft 1
Média fo_LL	-77,00%	-77,60%	-69,60%	-82,30%
Média sp	14,40%	13,3%	3,30%	15,80%
Média tempo	0,9	3,3	38,8	75,7
EXEMPLO/Ajuste	18X4 Bm 16 Ft 1	30X4 Bm 16 Ft 5	60X4 Bm 16 Ft 1	120X4 Bm 4 Ft 1
Média fo_LL	-59,20%	-68,00%	-69,60%	-71,20%
Média sp	15,60%	20,00%	30,30%	39,70%
Média tempo	1,9	8,9	60,4	137
EXEMPLO/Ajuste	18X6 Bm 16 Ft 1	30X6 Bm 16 Ft 5	60X6 Bm 4 Ft 10	120X6 Bm 2 Ft 1
Média fo_LL	-32,40%	-50,70%	-59,00%	-50,00%
Média sp	8,90%	8,00%	15,00%	36,50%
Média tempo				

Para L3 $\tau^m = 0$ $R^m = 0{,}99$

EXEMPLO/Ajuste	18X2 Bm 16 Ft 20	30X2 Bm 8 Ft 5	60X2 Bm 2 Ft 1
Média fo_LL	-78,00%	-78,70%	-80,10%
Média sp	12,20%	14,00%	9,70%
Média tempo	50,3	119,7	376,6
EXEMPLO/Ajuste	18X4 Bm 16 Ft 20	30X4 Bm 8 Ft 5	
Média fo_LL	-34,30%	-55,50%	
Média sp	4,40%	12,00%	
Média tempo	230,6	897,6	
EXEMPLO/Ajuste	18X6 Bm 16 Ft 20	30X6 Bm 1 Ft 20	
Média fo_LL	-20,90%	-24,90%	
Média sp	5,60%	4,70%	
Média tempo	1286,2	911,8	

Para L5 $\tau^m = 0$ $R^m = 0{,}99$

EXEMPLO/Ajuste	18X2 Bm 16 Ft 10	30X2 Bm 8 Ft 10	60X2 Bm 2 Ft 5
Média fo_LL	-78,20%	-77,40%	-80,10%
Média sp	4,40%	9,30%	8,00%
Média tempo	18,4	88,3	484,2
EXEMPLO/Ajuste	18X4 Bm 16 Ft 40	30X4 Bm 8 Ft 10	60X4 Bm 2 Ft 5
Média fo_LL	-34,20%	-62,30%	-60,30%
Média sp	2,20%	9,30%	9,00%
Média tempo	233,3	449,7	2293,6
EXEMPLO/Ajuste	18X6 Bm 16 Ft 40	30X6 Bm 4 Ft 10	
Média fo_LL	-23,80%	-26,80%	
Média sp	0,00%	2,00%	
Média tempo	1079	975,0	

Para L4 $\tau^m = 0$ $R^m = 0{,}99$

EXEMPLO/Ajuste	18X2 Bm 16 Ft 10	30X2 Bm 8 Ft 20	60X2 Bm 16 Ft 1	120X2 Bm 4 Ft 1
Média fo_LL	-72,30%	-66,20%	214,10%	231,10%
Média sp	11,10%	4,00%	2,00%	5,00%
Média tempo	1	2,2	28,4	59,7
EXEMPLO/Ajuste	18X4 Bm 16 Ft 10	30X4 Bm 16 Ft 1	60X4 Bm 8 Ft 1	120X4 Bm 4 Ft 1
Média fo_LL	-31,40%	-56,70%	-31,80%	-30,70%
Média sp	7,80%	10,70%	28,00%	32,00%
Média tempo	2,9	6,7	23,5	114,2
EXEMPLO/Ajuste	18X6 Bm 16 Ft 10	30X6 Bm 16 Ft 1	60X6 Bm 4 Ft 1	120X6 Bm 2 Ft 1
Média fo_LL	-20,30%	-22,40%	-26,40%	-14,40%
Média sp	3,30%	10,00%	21,00%	47,00%
Média tempo	13,1	28,3	35,2	146,4

Para L6 $\tau^m = 0$ $R^m = 0{,}99$

EXEMPLO/Ajuste	18X2 Bm 16 Ft 10	30X2 Bm 16 Ft 5	60X2 Bm 16 Ft 10	120X2 Bm 4 Ft 1
Média fo_LL	-75,20%	-74,5%	-66,30%	-76,80%
Média sp	8,90%	12,00%	6,00%	11,00%
Média tempo	1	3,0	35,4	63,1
EXEMPLO/Ajuste	18X4 Bm 16 Ft 1	30X4 Bm 16 Ft 5	60X4 Bm 16 Ft 10	120X4 Bm 4 Ft 10
Média fo_LL	-33,70%	-58,30%	-58,70%	-57,10%
Média sp	6,70%	13,30%	9,00%	7,00%
Média tempo	1,7	8,3	85,8	182,2
EXEMPLO/Ajuste	18X6 Bm 16 Ft 10	30X6 Bm 16 Ft 5	60X6 Bm 4 Ft 10	120X6 Bm 2 Ft 10
Média fo_LL	-23,10%	-27,00%	-37,20%	-40,40%
Média sp	1,10%	5,30%	7,70%	7,50%
Média tempo				

Para L3 $\tau^m = 0,33$ $R^m = 0,33$

EXEMPLO/Ajuste	18X2 Bm 16 Ft 20	30X2 Bm 16 Ft 20	60X2 Bm 1 Ft 5
Média fo_LL	-88,30%	-85,80%	-84,10%
Média sp	11,10%	10,00%	12,70%
Média tempo	83,6	729,2	597,7
EXEMPLO/Ajuste	18X4 Bm 16 Ft 20	30X4 Bm 8 Ft 20	
Média fo_LL	-71,00%	-79,50%	
Média sp	13,30%	21,30%	
Média tempo	1001,5	5236,3	
EXEMPLO/Ajuste	18X6 Bm 16 Ft 20	30X6 Bm 1 Ft 20	
Média fo_LL	-68,00%	-71,10%	
Média sp	18,90%	57,30%	
Média tempo	4816	6460,7	

Para L5 $\tau^m = 0,33$ $R^m = 0,33$

EXEMPLO/Ajuste	18X2 Bm 16 Ft 40	30X2 Bm 8 Ft 10	60X2 Bm 4 Ft 5
Média fo_LL	-88,30%	-83,90%	-83,30%
Média sp	6,70%	8,70%	11,30%
Média tempo	50,8	103	1063,5
EXEMPLO/Ajuste	18X4 Bm 16 Ft 40	30X4 Bm 8 Ft 10	60X4 Bm 2 Ft 5
Média fo_LL	-71,10%	-82,00%	-77,50%
Média sp	4,40%	14,70%	20,70%
Média tempo	296,8	538,3	3322,1
EXEMPLO/Ajuste	18X6 Bm 16 Ft 40	30X6 Bm 4 Ft 10	
Média fo_LL	-70,70%	-67,50%	
Média sp	10,00%	12,00%	
Média tempo	1437,5	1245,6	

Para L4 $\tau^m = 0,33$ $R^m = 0,33$

EXEMPLO/Ajuste	18X2 Bm 16 Ft 10	30X2 Bm 16 Ft 1	60X2 Bm 16 Ft 1	120X2 Bm 4 Ft 20
Média fo_LL	-8,80%	-46,30%	40,30%	40,30%
Média sp	0,00%	6,70%	0,00%	0,00%
Média tempo	1,0	2,6	27,9	88,3
EXEMPLO/Ajuste	18X4 Bm 16 Ft 10	30X4 Bm 8 Ft 20	60X4 Bm 16 Ft 1	120X4 Bm 4 Ft 20
Média fo_LL	-55,70%	-35,70%	20,90%	28,90%
Média sp	25,60%	20,70%	7,00%	1,70%
Média tempo	4,1	10,5	56,7	314,8
EXEMPLO/Ajuste	18X6 Bm 16 Ft 10	30X6 Bm 16 Ft 1	60X6 Bm 4 Ft 1	120X6 Bm 2 Ft 20
Média fo_LL	-58,70%	-35,10%	-29,50%	-28,80%
Média sp	38,90%	38,70%	65,30%	63,80%
Média tempo	22,8	41,1	55,7	851,8

Para L6 $\tau^m = 0,33$ $R^m = 0,33$

EXEMPLO/Ajuste	18X2 Bm 16 Ft 10	30X2 Bm 16 Ft 5	60X2 Bm 16 Ft 10	120X2 Bm 4 Ft 10
Média fo_LL	-85,00%	-64,20%	-83,80%	-82,70%
Média sp	11,10%	8,00%	10,30%	12,80%
Média tempo	1,1	3,2	40,9	101,5
EXEMPLO/Ajuste	18X4 Bm 16 Ft 10	30X4 Bm 16 Ft 5	60X4 Bm 16 Ft 1	120X4 Bm 4 Ft 10
Média fo_LL	-67,50%	-79,90%	-74,60%	-81,30%
Média sp	18,90%	20,70%	29,30%	35,00%
Média tempo	3,4	9,9	64,7	297,5
EXEMPLO/Ajuste	18X6 Bm 16 Ft 1	30X6 Bm 16 Ft 5	60X6 Bm 4 Ft 1	120X6 Bm 2 Ft 10
Média fo_LL	-66,70%	-62,30%	-69,50%	-74,90%
Média sp	28,90%	27,30%	44,00%	66,20%
Média tempo				

Para L3 $\tau^m = 0{,}33$ $R^m = 0{,}66$

EXEMPLO/Ajuste	18X2 Bm 16 Ft 20	30X2 Bm 16 Ft 20	60X2 Bm 2 Ft 1
Média fo_LL	-81,30%	-89,10%	-84,50%
Média sp	12,20%	7,30%	14,00%
Média tempo	69,9	591,3	643,2
EXEMPLO/Ajuste	18X4 Bm 16 Ft 20	30X4 Bm 8 Ft 5	
Média fo_LL	-62,30%	-70,60%	
Média sp	10,00%	25,30%	
Média tempo	495	1615,6	
EXEMPLO/Ajuste	18X6 Bm 16 Ft 20	30X6 Bm 1 Ft 20	
Média fo_LL	-44,00%	-54,80%	
Média sp	4,40%	8,00%	
Média tempo	1868,6	1982,9	

Para L5 $\tau^m = 0{,}33$ $R^m = 0{,}66$

EXEMPLO/Ajuste	18X2 Bm 16 Ft 40	30X2 Bm 8 Ft 10	60X2 Bm 4 Ft 5
Média fo_LL	-80,00%	-88,40%	-85,00%
Média sp	10,00%	8,00%	10,70%
Média tempo	50,9	91,3	988,5
EXEMPLO/Ajuste	18X4 Bm 16 Ft 40	30X4 Bm 8 Ft 10	60X4 Bm 2 Ft 5
Média fo_LL	-60,90%	-72,20%	-71,60%
Média sp	10,00%	15,30%	14,70%
Média tempo	260,5	499,5	2417,5
EXEMPLO/Ajuste	18X6 Bm 16 Ft 40	30X6 Bm 4 Ft 10	
Média fo_LL	-44,00%	-56,70%	
Média sp	2,20%	6,00%	
Média tempo	1229,7	1114,4	

Para L4 $\tau^m = 0{,}33$ $R^m = 0{,}66$

EXEMPLO/Ajuste	18X2 Bm 16 Ft 10	30X2 Bm 16 Ft 1	60X2 Bm 8 Ft 1	120X2 Bm 4 Ft 20
Média fo_LL	48,60%	-55,40%	225,40%	207,30%
Média sp	1,10%	6,70%	0,00%	0,00%
Média tempo	0,9	2,5	8,5	90,6
EXEMPLO/Ajuste	18X4 Bm 16 Ft 10	30X4 Bm 8 Ft 20	60X4 Bm 16 Ft 1	120X4 Bm 4 Ft 1
Média fo_LL	-54,00%	-49,10%	8,70%	111,40%
Média sp	20,00%	17,30%	39,70%	70,30%
Média tempo	3,9	10,2	77,2	243
EXEMPLO/Ajuste	18X6 Bm 8 Ft 10	30X6 Bm 16 Ft 1	60X6 Bm 4 Ft 1	120X6 Bm 2 Ft 1
Média fo_LL	-38,40%	-42,10%	-37,60%	1,50%
Média sp	15,60%	24,70%	44,70%	68,00%
Média tempo	8,1	31,4	41,4	190,1

Para L6 $\tau^m = 0{,}33$ $R^m = 0{,}66$

EXEMPLO/Ajuste	18X2 Bm 16 Ft 1	30X2 Bm 16 Ft 5	60X2 Bm 16 Ft 10	120X2 Bm 4 Ft 1
Média fo_LL	-78,10%	-84,00%	-74,30%	-83,80%
Média sp	14,40%	13,3%	5,00%	18,20%
Média tempo	0,9	3,2	37,9	77,6
EXEMPLO/Ajuste	18X4 Bm 16 Ft 10	30X4 Bm 16 Ft 5	60X4 Bm 16 Ft 10	120X4 Bm 4 Ft 1
Média fo_LL	-60,00%	-66,60%	-68,20%	-64,20%
Média sp	18,90%	23,30%	12,30%	38,80%
Média tempo	3,1	9,1	98,7	128,1
EXEMPLO/Ajuste	18X6 Bm 16 Ft 1	30X6 Bm 16 Ft 5	60X6 Bm 4 Ft 10	120X6 Bm 2 Ft 1
Média fo_LL	-41,10%	-55,10%	-57,00%	-54,80%
Média sp	11,10%	10,00%	15,70%	33,00%
Média tempo				

Para L3 $\tau^m = 0,66$ $R^m = 0,33$

EXEMPLO/Ajuste	18X2 Bm 16 Ft 20	30X2 Bm 16 Ft 20	60X2 Bm 1 Ft 5
Média fo_LL	-54,20%	-57,80%	-59,80%
Média sp	5,60%	4,00%	10,30%
Média tempo	48,7	456,8	475,5
EXEMPLO/Ajuste	18X4 Bm 16 Ft 20	30X4 Bm 8 Ft 20	
Média fo_LL	-55,10%	-61,80%	
Média sp	30,00%	20,70%	
Média tempo	741,5	3988	
EXEMPLO/Ajuste	18X6 Bm 8 Ft 20	30X6 Bm 2 Ft 5	
Média fo_LL	-56,10%	-61,60%	
Média sp	42,20%	48,00%	
Média tempo	3252,2	4097	

Para L5 $\tau^m = 0,66$ $R^m = 0,33$

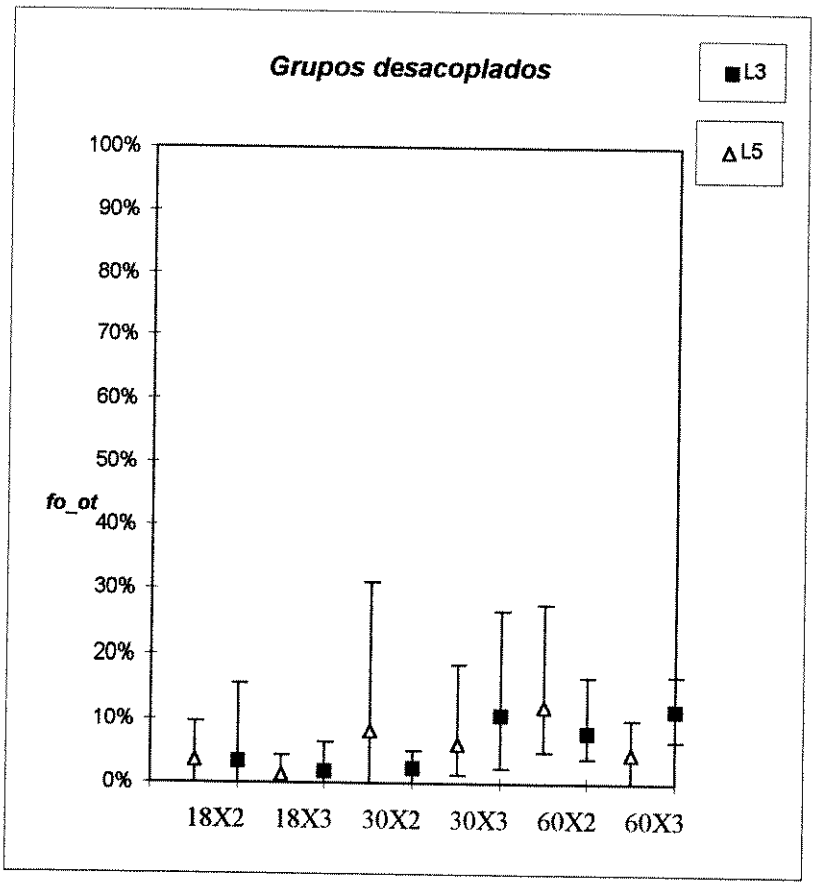
EXEMPLO/Ajuste	18X2 Bm 16 Ft 40	30X2 Bm 8 Ft 10	60X2 Bm 2 Ft 5
Média fo_LL	-53,60%	-53,00%	-57,10%
Média sp	16,70%	9,30%	9,30%
Média tempo	50,2	92	493,2
EXEMPLO/Ajuste	18X4 Bm 16 Ft 40	30X4 Bm 8 Ft 10	60X4 Bm 2 Ft 5
Média fo_LL	-57,80%	-63,40%	-79,20%
Média sp	16,70%	27,30%	15,30%
Média tempo	320,1	843,2	3148,2
EXEMPLO/Ajuste	18X6 Bm 8 Ft 40	30X6 Bm 4 Ft 10	
Média fo_LL	-57,60%	-71,40%	
Média sp	33,30%	19,3%	
Média tempo	1449,7	1458,5	

Para L4 $\tau^m = 0,66$ $R^m = 0,33$

EXEMPLO/Ajuste	18X2 Bm 16 Ft 10	30X2 Bm 16 Ft 1	60X2 Bm 16 Ft 1	120X2 Bm 4 Ft 20
Média fo_LL	-31,20%	-47,10%	39,00%	94,20%
Média sp	7,80%	4,70%	2,00%	0,00%
Média tempo	1,1	2,6	29,5	90,6
EXEMPLO/Ajuste	18X4 Bm 16 Ft 10	30X4 Bm 16 Ft 1	60X4 Bm 16 Ft 1	120X4 Bm 4 Ft 1
Média fo_LL	-39,00%	-2,70%	45,60%	93,20%
Média sp	23,30%	9,30%	9,00%	13,80%
Média tempo	4,3	12,5	61,2	111,7
EXEMPLO/Ajuste	18X6 Bm 16 Ft 10	30X6 Bm 16 Ft 1	60X6 Bm 4 Ft 1	120X6 Bm 2 Ft 1
Média fo_LL	-19,00%	-10,30%	16,70%	74,50%
Média sp	40,00%	17,30%	35,70%	39,80%
Média tempo	23,5	39	53,2	187,6

Para L6 $\tau^m = 0,66$ $R^m = 0,33$

EXEMPLO/Ajuste	18X2 Bm 16 Ft 10	30X2 Bm 16 Ft 5	60X2 Bm 16 Ft 10	120X2 Bm 4 Ft 10
Média fo_LL	-44,00%	-42,00%	-51,70%	-58,50%
Média sp	12,20%	8,70%	5,00%	8,50%
Média tempo	1	3,1	39	93,6
EXEMPLO/Ajuste	18X4 Bm 16 Ft 1	30X4 Bm 16 Ft 5	60X4 Bm 16 Ft 10	120X4 Bm 4 Ft 10
Média fo_LL	-50,50%	-53,60%	-68,40%	-76,10%
Média sp	27,80%	18,70%	22,30%	36,50%
Média tempo	2,1	13	111,3	285,8
EXEMPLO/Ajuste	18X6 Bm 16 Ft 1	30X6 Bm 16 Ft 5	60X6 Bm 4 Ft 1	120X6 Bm 2 Ft 1
Média fo_LL	-54,50%	-64,10%	-72,30%	-72,30%
Média sp	34,40%	26,00%	51,00%	72,30%
Média tempo	7,5	41,3		



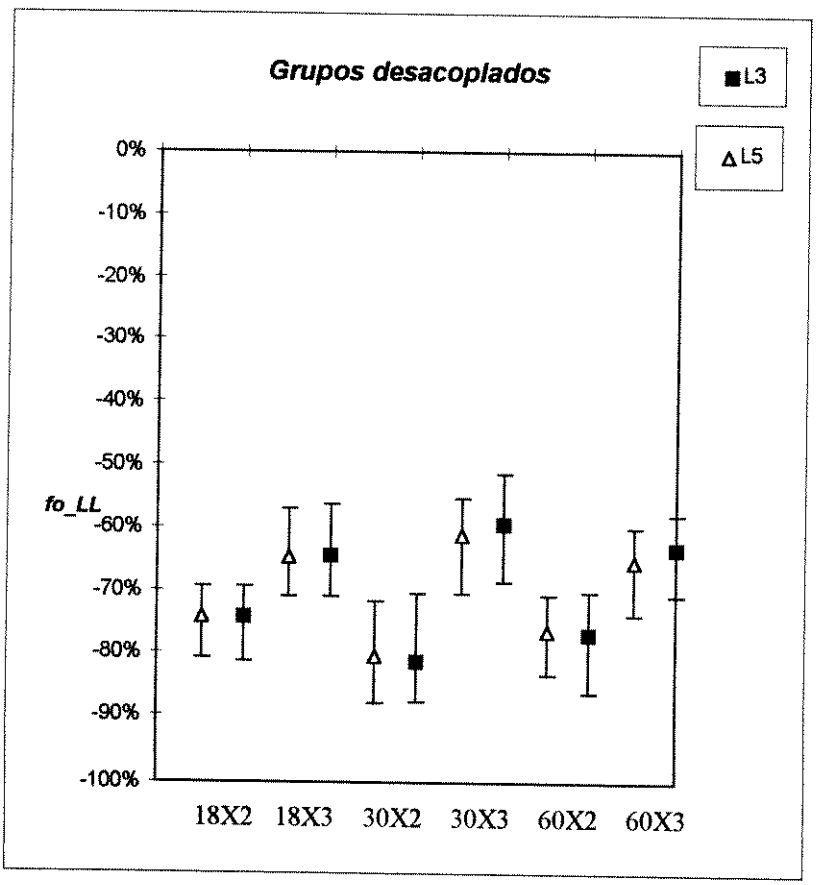
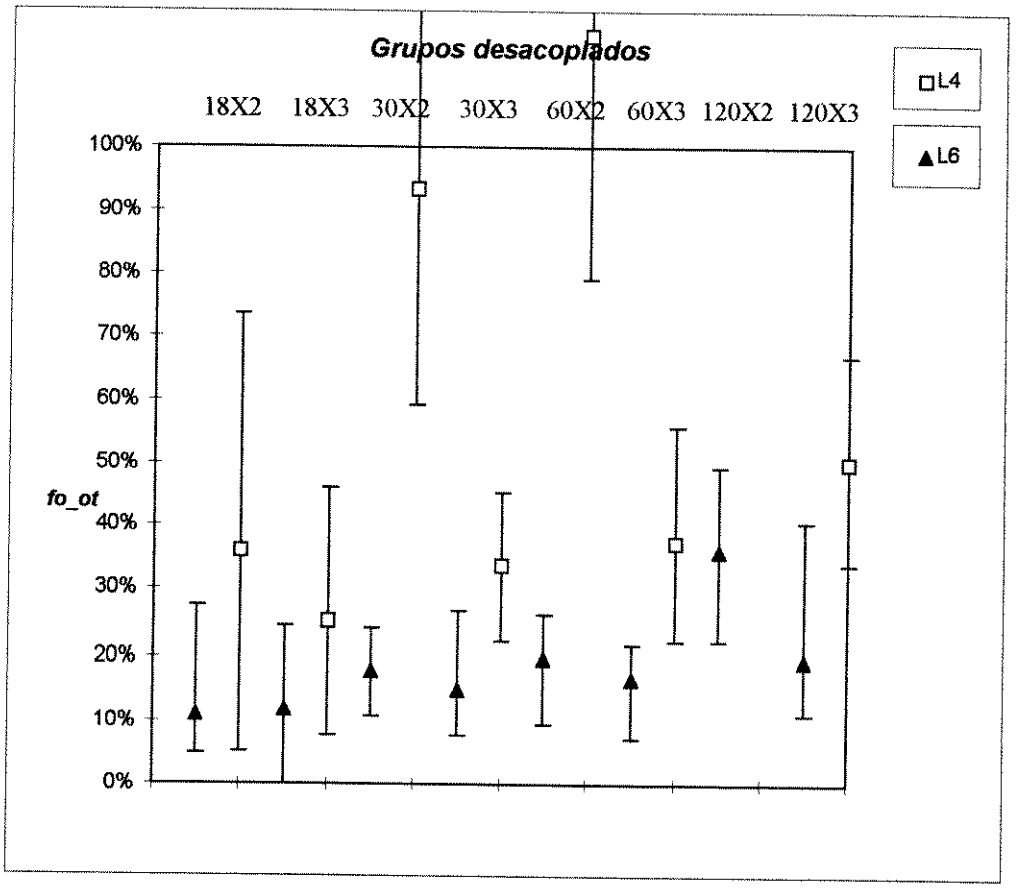


Gráfico de Dispersão 1 - quad/triang = média



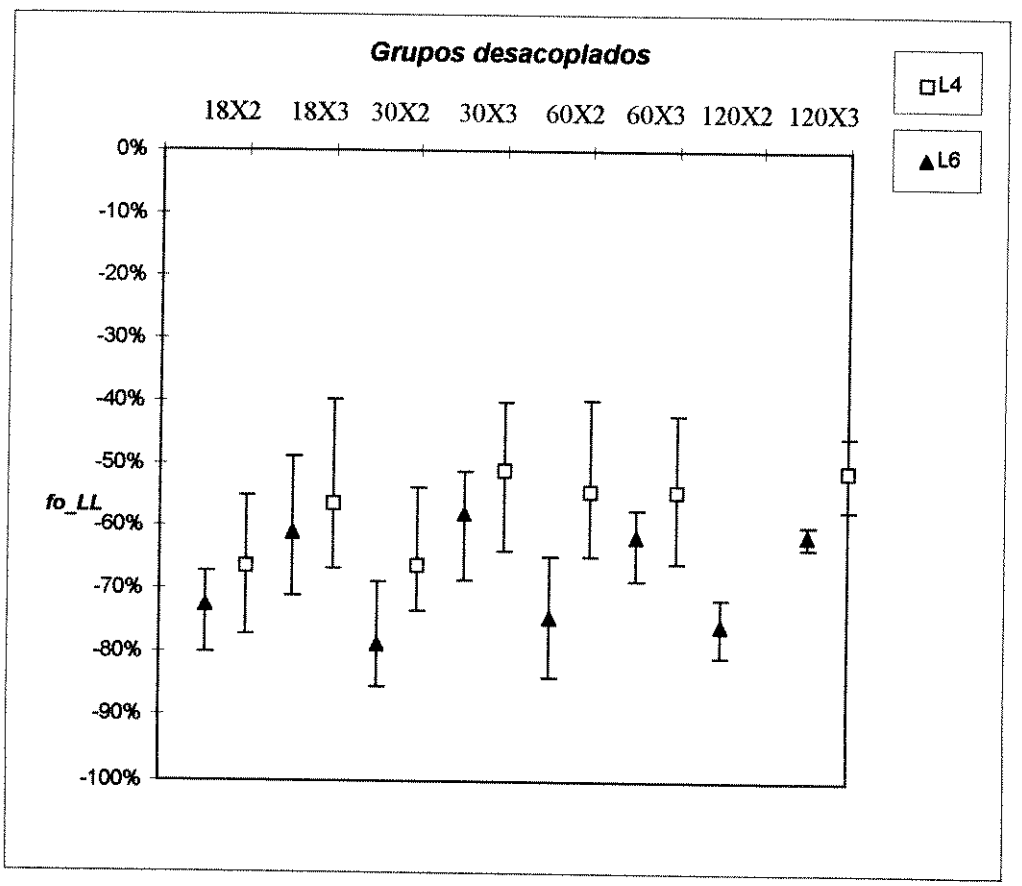
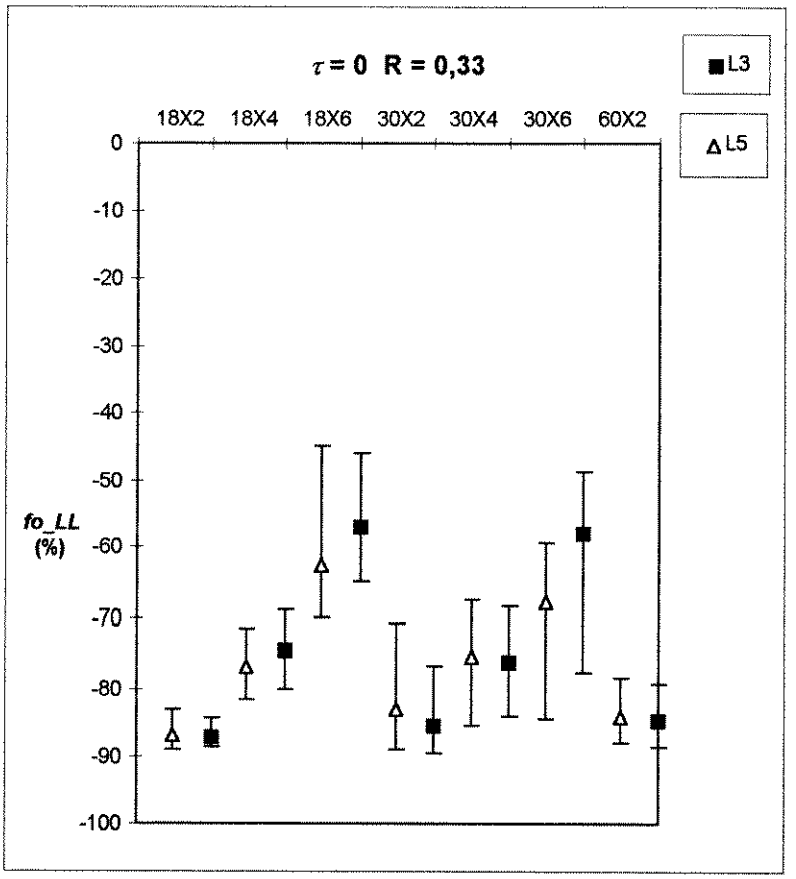


Gráfico de Dispersão 2 - quad/triang = média



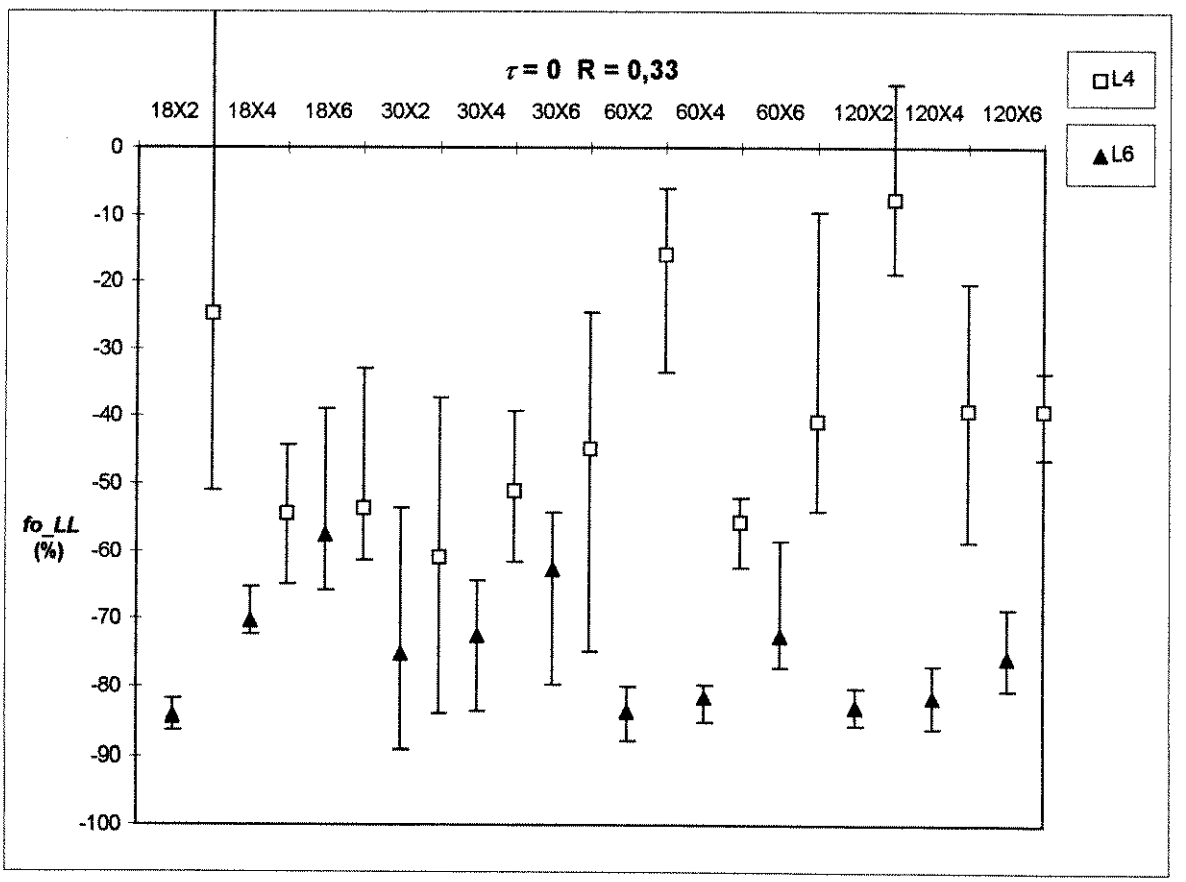
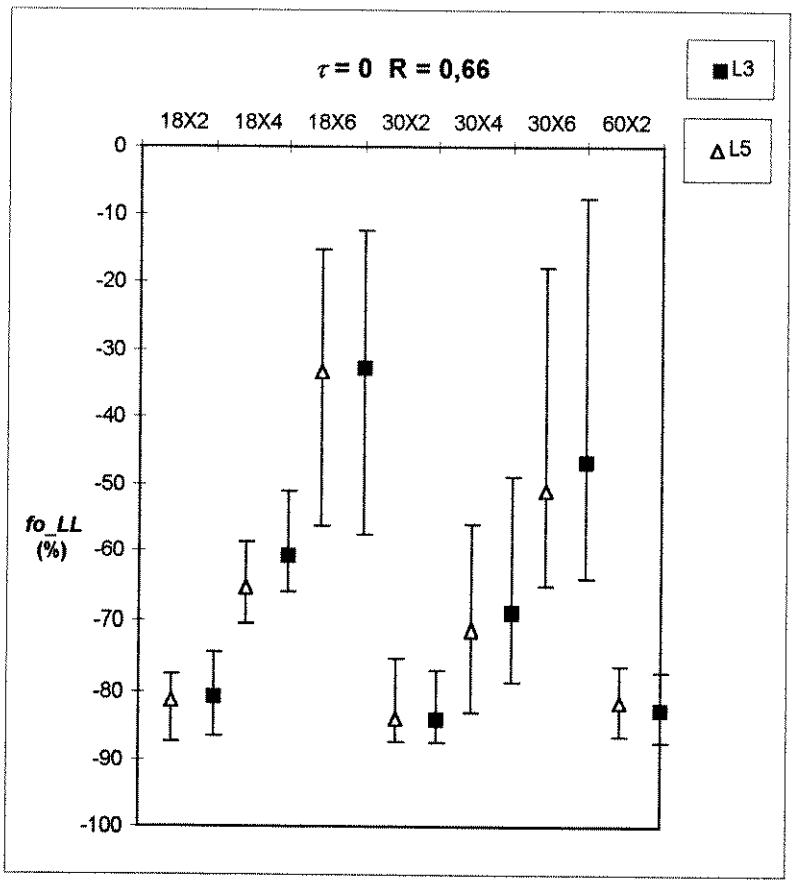


Gráfico de Dispersão 3 - quad/triang = média



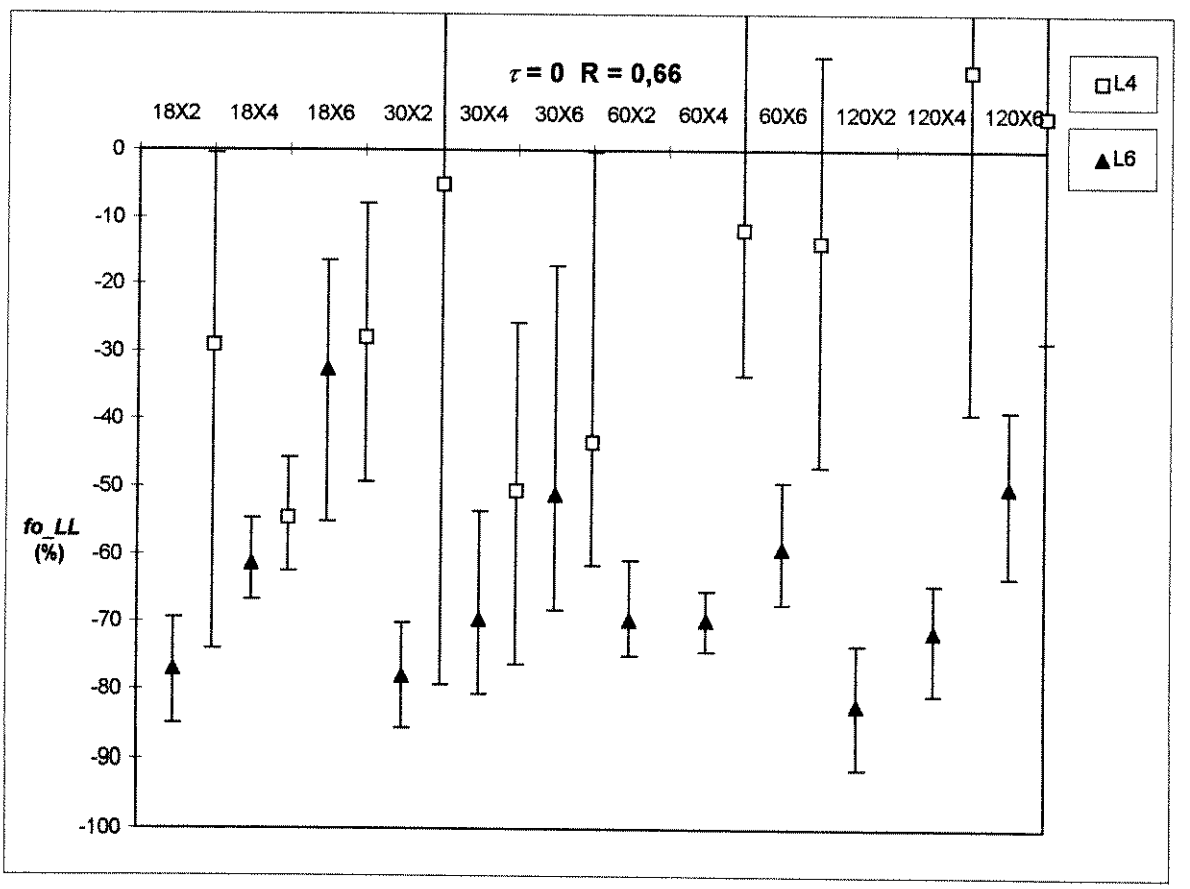
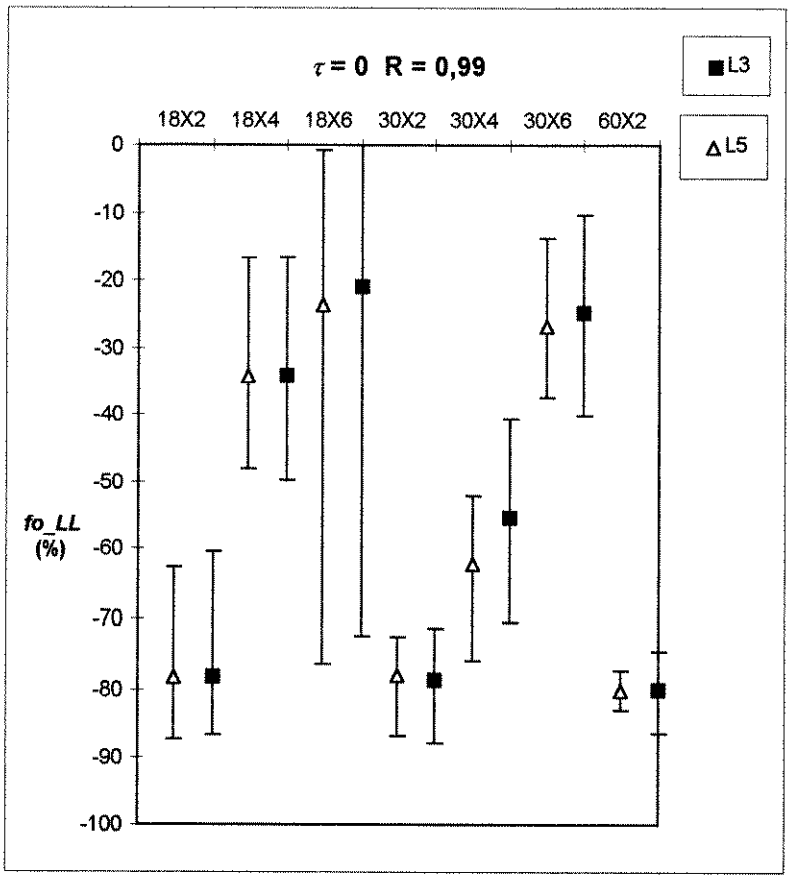


Gráfico de Dispersão 4 - quad/triang = média



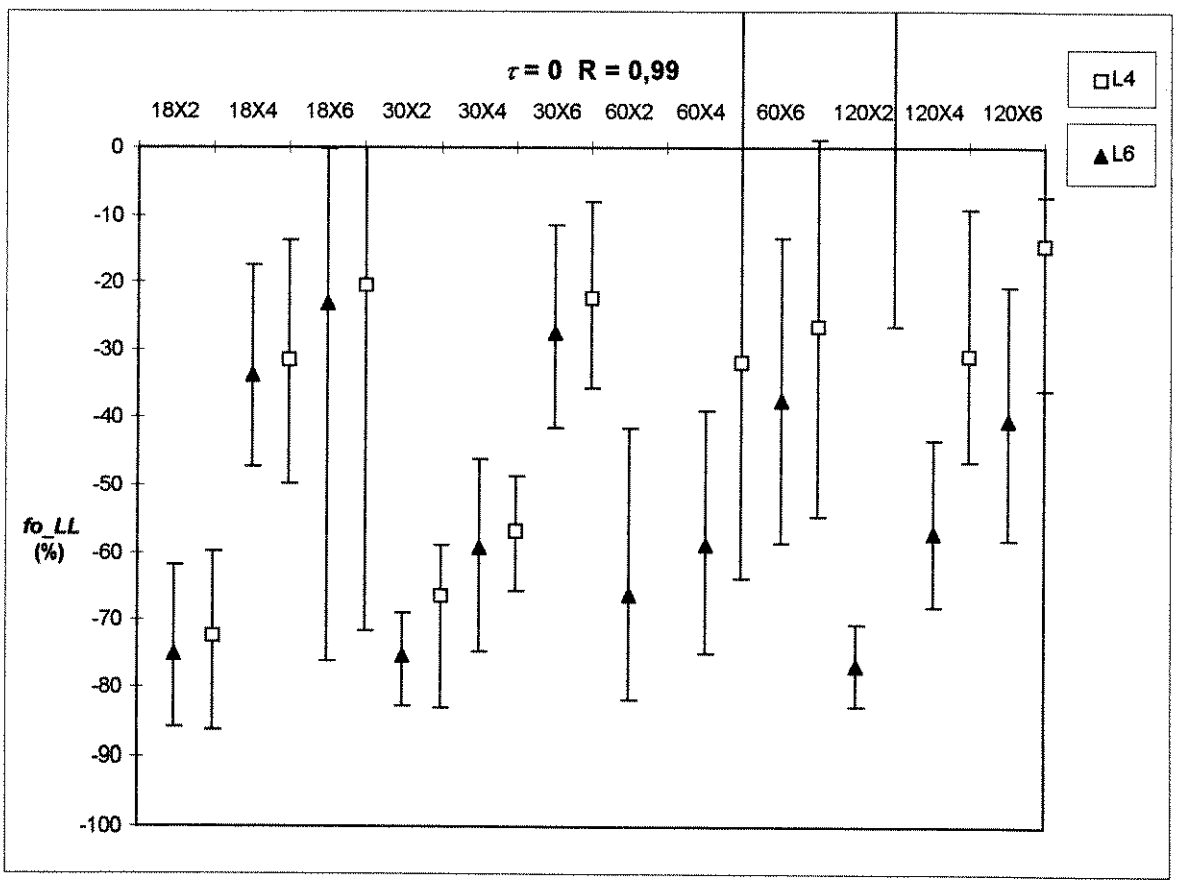
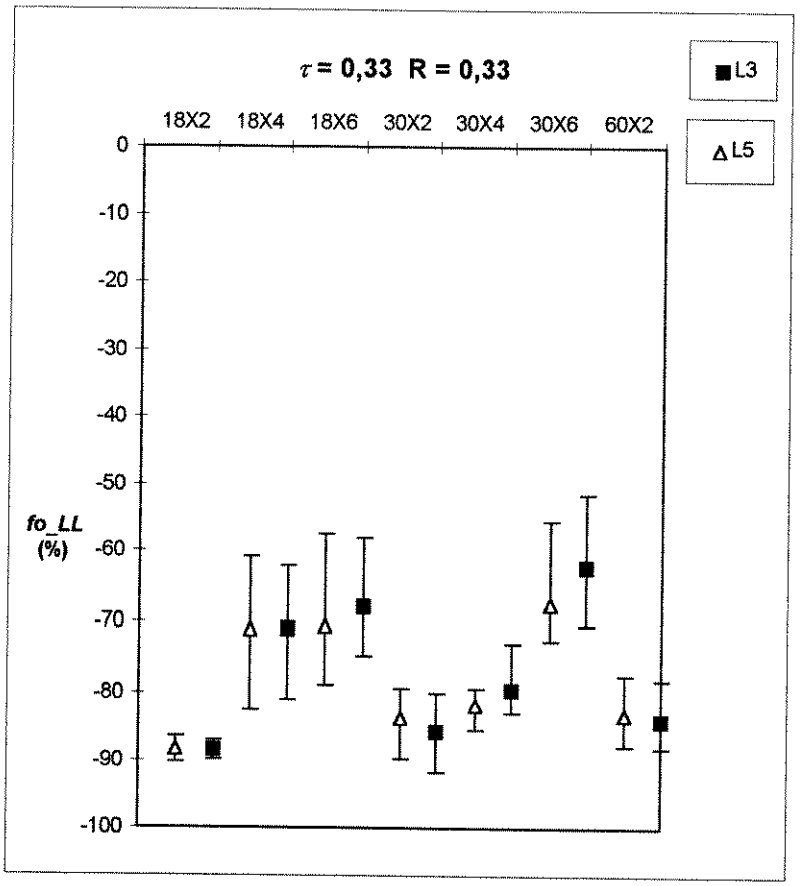


Gráfico de Dispersão 5 - quad/triang = média



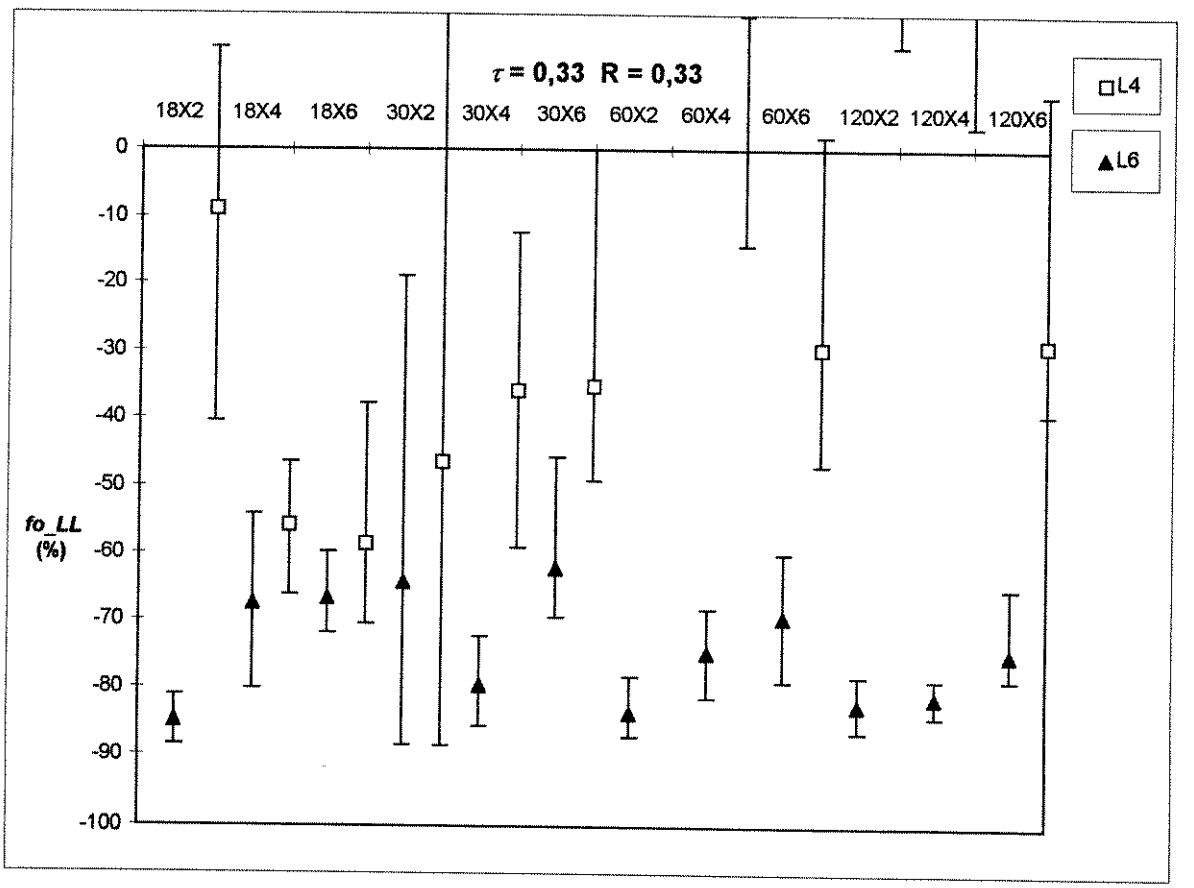
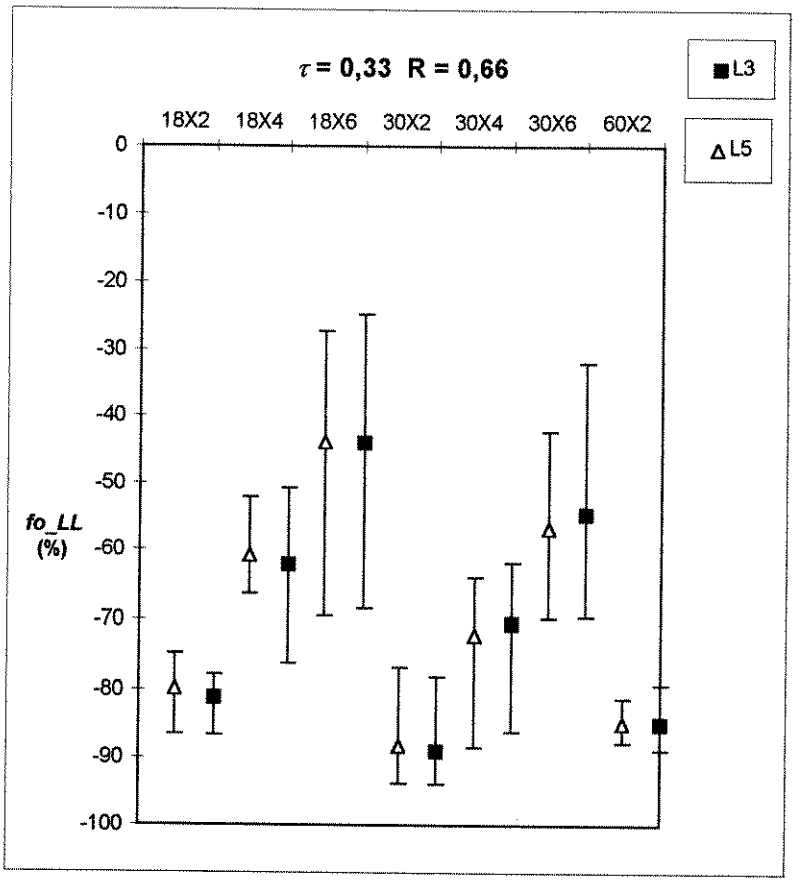


Gráfico de Dispersão 6 - quad/triang = média



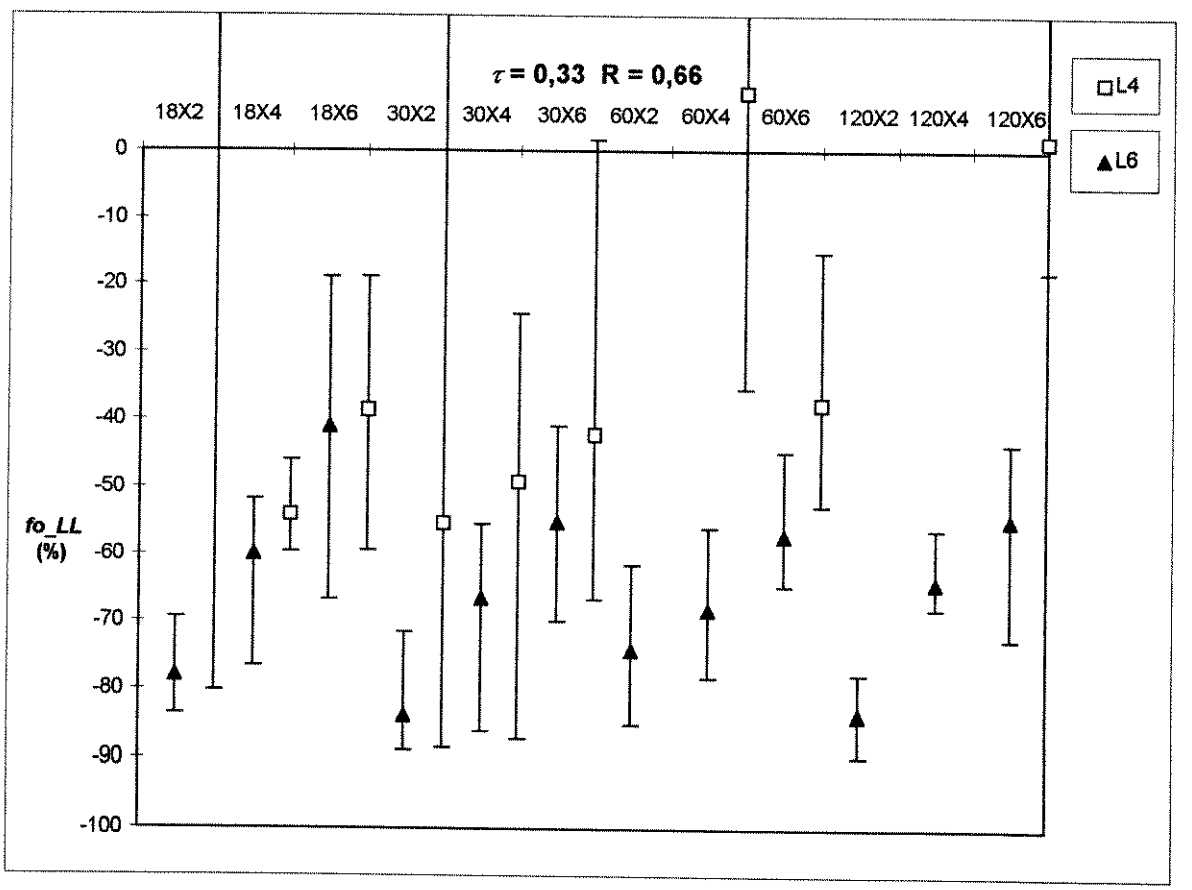
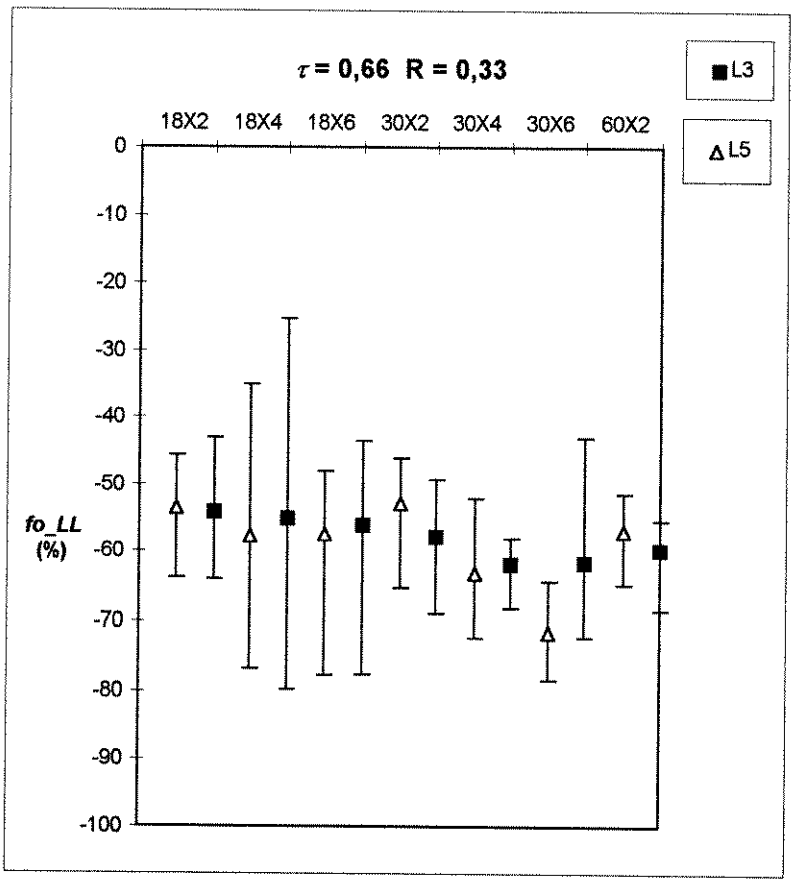


Gráfico de Dispersão 7 - quad/triang = média



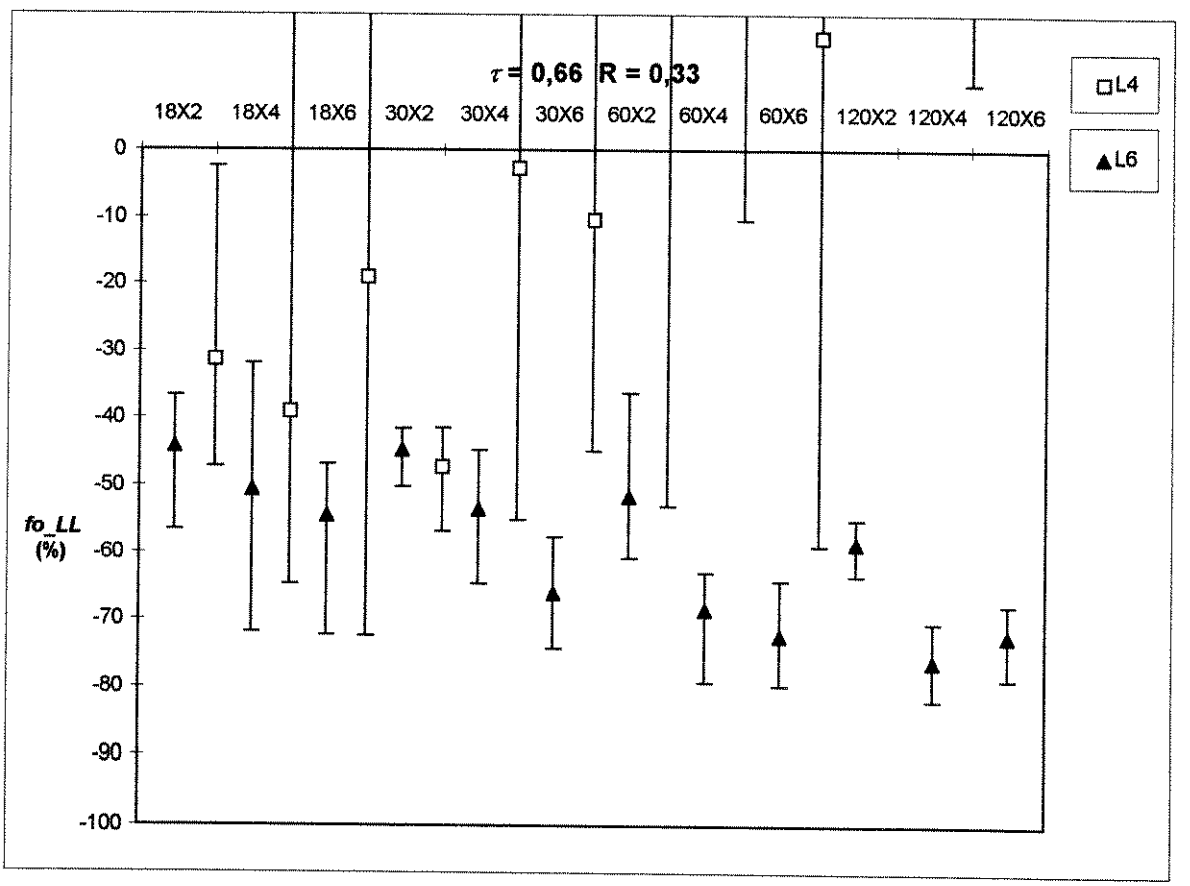


Gráfico de Dispersão 8 - quad/triang = média

7.9 Discussão dos resultados

Apesar dos poucos e pequenos exemplos apresentados no item 7.1, a superioridade do enfoque de busca proposto neste trabalho em relação à programação inteira-mista, fica evidente no exemplo de 4 tarefas em 2 máquinas, quando o tempo de processamento pelo método exato fica próximo a 1% do tempo necessário para o AMPL/CPLEX resolver o mais eficiente dos modelos de programação inteira mista. Isto não significa que estes modelos sejam dispensáveis sempre, mas apenas que a sua eficiência diminui drasticamente quando se aumenta o número de variáveis binárias.

Para a análise dos resultados da busca em feixe, utilizou-se um número maior de exemplos com dimensões maiores e uma variedade maior. Na parte 1 dos testes, onde são conhecidas as soluções ótimas, embora os custos de troca não sejam dependentes da seqüência e os parâmetros τ^m e R^m não sejam controlados de forma rígida, os resultados se parecem bastante com os da parte 2, sugerindo que o desacoplamento entre grupos não teve grande impacto no desempenho da busca. Isto sugere que o conjunto de problemas gerados de forma desacoplada não apresenta particularidades exclusivas.

O julgamento da qualidade da solução final de cada exemplo pela observação apenas da medida *fo_LL*, pode levar a erros substanciais. No item 7.7 o exemplo 30X2 com L4 fornece *fo_LL* = -66,2% e no exemplo 18X3 com L5 fornece *fo_LL* = -64,6%. Isto faz parecer que o primeiro resultado é melhor que o segundo, mas a observação dos *fo_ot* correspondentes (93,4% e 1,3%) mostra que o segundo resultado está muito mais próximo do ótimo. O motivo principal deste fato é a variação significativa do quociente entre *ot* e *ubLL* para duas instâncias distintas.

O fato de *ubLL* ser uma referência fraca também prejudica a análise dos gráficos de dispersão, pois uma parte grande, provavelmente a maior parte, da dispersão observada nas medidas de *fo_LL* se deve à variação do quociente entre *ot* e *ubLL*. Observe que no gráfico de dispersão 5 ($\tau^m = 0$ e $R^m = 0{,}99$), o problema 18X6 apresenta grande dispersão com todos os limitantes. Não é difícil de entender que neste caso onde não existe pressão de atrasos ($\tau^m = 0$) e as datas previstas das tarefas estão bem afastadas ($R^m = 0{,}99$), *ubLL* pode apresentar um resultado bastante próximo do ótimo em uma determinada instância, ou um resultado ruim em uma outra instância em que algumas das datas previstas aleatórias ficaram próximas. Por outro lado, todos os limitantes apresentaram resultados com médias e dispersões aproximadamente

iguais para o mesmo problema 18X6, sugerindo que são mais estáveis que *ubLL*. A variação relativa entre *ot* e *ubLL* tanto pode aumentar como diminuir as dispersões observadas nos gráficos. No gráfico 2, o problema 120X3 com L6, apresenta menor dispersão com *fo_LL* do que com *fo_ot*, já no gráfico 1, 0 problema 30X2 com L3 apresenta menor dispersão com *fo_ot* do que com *fo_LL*.

Presumindo que os resultados médios obtidos por L3 e L5 estejam sempre próximos ao ótimo na parte 2 dos exemplos, *ot* e *ubLL* se aproximam quando aumenta o número de máquinas ou o parâmetro R^m. Também o aumento do parâmetro τ^m parece produzir o mesmo efeito, embora em menor grau.

Um fato a primeira vista surpreendente é como a restrição imposta pela filtragem é capaz de melhorar a solução obtida pela busca em feixe. Isto pode ser explicado reconhecendo que todos os limitantes desenvolvidos possuem um certo grau de "miopia". Se a filtragem não impedir, as tarefas poderão ser seqüenciadas em posições inadequadas que não serão rejeitadas pela avaliação dos limitantes, o que resulta em soluções ruins quando Ft (filtragem) é de 100%. Isto fica claro observando os gráficos de variação de filtragem com o limitante L4 (o mais míope de todos pois despreza os adiantamentos em G_2), a melhoria na solução é dramática com menores valores de Ft em quase todos os gráficos do item 7.6.

O tempo de processamento varia com cada um dos limitantes, conforme mostrado nas tabelas do item 7.8, e cresce com o aumento de Bm e Ft, exatamente como seria de se esperar. Outro fato observado é o número de tarefas submetidas a divisão em mais de uma máquina, que é pequeno no caso dos exemplos desacoplados e nos exemplos acoplados aumenta com o número de máquinas, com a diminuição de R^m e parece sofrer pouca influência de τ^m. A experiência acima mostra que os limitantes mais precisos realmente tem um desempenho melhor no sentido de encontrar soluções de boa qualidade. Os limitantes L3 e L5 mostraram sua superioridade em relação a L6 mesmo quando utilizados com valores de Bm muito menores. No entanto, o tempo de computação requerido pelos limitantes L3 e L5 pode não ser razoável em muitos casos. É interessante observar que, embora equivalentes quanto aos resultados obtidos, L3 é sensivelmente mais lento que L5, como pode ser observado em alguns exemplos (30X4, $R^m = 0{,}33$ e $\tau^m = 0{,}66$, onde o tempo com L5 ficou próximo a 30% do tempo com L3, mesmo com Ft maior para L5). Isto certamente ocorre devido à tendência do limitante L3 de fazer muitas divisões de tarefas durante o seu cálculo, o que sobrecarrega o modelo **F** com muitas frações, tornando-o mais lento.

O limitante L4 mostra que as simplificações na modelagem dos problemas de programação de tarefas podem ser desastrosas. Em muitos exemplos com L4 chegou-se a valores positivos de *fo_LL*, isto é, soluções piores do que o método de despacho. Por outro lado, os resultados com os outros limitantes mostraram que o método de busca proposto é capaz de encontrar boas soluções em todos os exemplos gerados e que estes resultados superam de longe os resultados obtidos com o método de despacho.

8. CONCLUSÕES

Neste trabalho de tese o objetivo foi desenvolver um método para programação de tarefas em máquinas paralelas sujeitas a divisão de cada tarefa em várias máquinas. As máquinas consideradas são não-relacionadas, o tempo de troca entre tarefas pode depender da última tarefa processada e as tarefas podem ter suas próprias datas previstas. A programação deverá ser feita de forma a minimizar uma função objetivo que inclui atrasos e adiantamentos ponderados e também custos de troca entre tarefas.

Inicialmente o problema foi escrito como um modelo de programação linear inteira-mista. Para aumentar a capacidade de resolução, o modelo inteiro-misto serviu como base no desenvolvimento de um método de busca em árvore capaz de resolver o problema de forma exata e não-exata. Para tanto foram desenvolvidos 6 limitantes, sendo 2 deles limitantes inferiores para utilização em uma busca *Branch and Bound* e também 4 limitantes utilizados em uma busca em feixe filtrada. O método foi testado em um conjunto razoável de problemas diferentes, com resultados animadores.

O método de busca proposto neste trabalho pretende ser aplicado principalmente em indústrias de produção de papel, tecidos, plásticos, laminação de metais e possivelmente indústrias químicas e de produção de alimentos. Em ambientes industriais, pode haver não

apenas uma quantidade grande de pedidos de produtos diferentes, mas também pedidos de quantidades grandes de um mesmo produto. É claro que a maneira mais adequada de lidar com a produção de uma quantidade grande de um mesmo produto e data prevista relativamente próxima, é a divisão do trabalho entre as máquinas disponíveis. Esta e outras características básicas presentes nos problemas de programação da produção, como o tempo de troca entre tarefas dependente da seqüência, têm merecido pouca atenção na literatura de programação em máquinas paralelas. (Mc CARTY & LIN, 1993).

Alguns textos que tratam problemas semelhantes aplicáveis na programação de tarefas em indústrias têxteis (SERAFINI & SPERANZA, 1992 e ESPUÑA & PUIGJANER, 1989) utilizam, basicamente, seqüenciamento de tarefas segundo a prioridade e simulações. Os resultados obtidos pelo método aqui desenvolvido é sempre superior (excluindo L4) aos obtidos por *ubLL*, que também seqüencia as tarefas segundo a prioridade. Em quase todos os casos a diferença é grande a favor deste método, com o valor da função objetivo chegando a 10% do que seria obtido pela regra de despacho. Ao mesmo tempo, a velocidade de execução, embora inferior às obtidas pelos seqüenciamentos segundo a prioridade, é bastante grande (com L6) para permitir a resolução de problemas com mais de 100 tarefas em um microcomputador.

A eficiência do método de busca em feixe filtrada aplicado a problemas de programação de tarefas já havia sido demonstrada por OW & MORTON (1988) em comparação com a busca em vizinhança. Alguns testes de uma versão ainda em desenvolvimento do modelo sem divisão de tarefas, onde um modelo linear obtido a partir do modelo S (capítulo 3) e resolvido com os deslocamentos de blocos de tarefas descritos no item 6.2.4.1, mostram que a velocidade de execução pode ser ainda bastante aumentada. Este modelo em desenvolvimento mostrou que é possível obter soluções finais equivalentes e até superiores às da busca Tabu (esta também em fase de desenvolvimento) em problemas 80x1 (80 tarefas em 1 máquina) em tempos inferiores a 0,5% do tempo gasto pela busca Tabu.

A influência do parâmetro largura do Feixe (Bm) na qualidade da solução e no tempo de processamento ocorre, como já era de se esperar, com uma nítida melhoria da solução quando Bm é aumentado. Porém, o tempo de processamento também aumenta de forma aproximadamente proporcional a Bm, havendo a necessidade de avaliar caso a caso para uma escolha adequada deste parâmetro. O parâmetro filtragem (Ft) chega a ter uma influência surpreendente sobre a qualidade da solução final, pois inicialmente se esperava apenas reduzir

o tempo de processamento sem prejudicar muito a qualidade da solução. Além da melhoria no tempo de processamento, se verificou também uma melhoria, em muitos casos substancial, na qualidade da solução final, mostrando que a filtragem é mais do que importante nos métodos de busca em feixe.

Um ponto a ser melhor explicado, é a razão da utilização de 6 limitantes (2 limitantes inferiores e 4 limitantes sobre seqüências completas). Talvez apenas 1 limitante inferior (LI1) e dois limitantes sobre seqüência completa (L5 para boa qualidade da solução final e L6 para baixo tempo de processamento) já fossem suficientes. Uma justificativa para isto é que, como explicado na discussão dos resultados, a avaliação da qualidade da solução final freqüentemente é imprecisa se feita apenas pelas medidas do capítulo 7 (*fo_LL* e *fo_ot* quando possível). Neste sentido, uma comparação dos resultados obtidos com limitantes diferentes fornece uma maior segurança no julgamento das boas ou más soluções finais. As diferenças pequenas entre os valores médios de *fo_LL* obtidos com L3 e L5 (os limitantes mais lentos porém de resultados melhores), em quase todas as instâncias testadas, são uma indicação clara de que o método possui um bom desempenho médio. Os resultados com o limitante L4 foram em alguns casos muito fracos, piores até do que *ubLL*, mas servem para mostrar que os adiantamentos das tarefas não podem ser desprezados, mesmo que parcialmente como é feito em L4, sob pena de se chegar a soluções finais inaceitáveis. O limitante inferior LI2, foi desenvolvido numa tentativa de reduzir o tempo de processamento e número de nós gerados nas buscas *Branch & Bound* com LI1. De fato, o número de nós gerados com LI2 é significativamente menor e o tempo de execução na maioria dos casos é menor. As principais contribuições de LI2 foram os desenvolvimentos (necessários para justificá-lo) de algumas das propriedades colocadas no capitulo 5, que permitiram o desenvolvimento de soluções ótimas para alguns dos problemas do capítulo 7 e essenciais para uma adequada avaliação do enfoque proposto. Tais propriedades podem ser úteis no futuro para o aprofundamento das pesquisas neste tema.

O método também permite o tratamento de alterações na função objetivo ou mesmo no modelo de seqüenciamentos e alocações fixas, sem alterar as características básicas da busca ou dos limitantes empregados. No item 7.1, quando a função objetivo é substituída pelo atraso máximo, a alteração é fácil para o modelo de programação inteira-mista e, no caso da busca em feixe, os limitantes poderiam ser obtidos a partir de procedimentos semelhantes para a

geração da soluções utilizadas pelos limitantes L3 a L6, porém retornando o atraso máximo em vez da função objetivo (4.1)

O objetivo aqui não foi fazer uma análise detalhada do desempenho do método, trabalho este que será deixado para outros pesquisadores interessados em investigar este tema. É importante que mais exemplos sejam testados, explorando melhor a variação de alguns parâmetros que ficaram fixos nos testes aqui apresentados. Também é importante que exemplos obtidos a partir de casos reais, com tempos unitários de processamento reais e tempos de troca verdadeiros, sejam testados e as soluções obtidas comparadas com as tradicionalmente empregadas.

Como sugestões para possíveis aperfeiçoamentos neste método, deve-se citar a extensão da minimização de atrasos e adiantamentos em máquinas simples (item 6.2.4.1), para que mesmo que de maneira aproximada possa substituir o procedimento do apêndice 1 em $\mathbf{G}_1$ durante a busca. Isto poderia aumentar a velocidade da busca e também a qualidade da solução final, desde que o limitante utilizado seja semelhante a L5 e não a L6. Uma outra possibilidade é a investigação mais apurada da filtragem, com a utilização de outras funções para seqüenciamento das tarefas restantes e a comparação entre os resultados obtidos com várias filtragens diferentes.

9. BIBLIOGRAFIA

AZIZOGLU, M.; KONDAKCI, S.; KIRCA, Ö. : "Bicriteria Scheduling Problem Involving Total Tardiness and Total Earliness Penalties". **International Journal of Production Economics,** 23, p. 17-24, 1991.

BAKER, K. R.: "Introduction to Sequencing and Scheduling". **John Wiley & Sons,** 1974.

BAKER, K. R. & SCUDDER, G. D.: "Sequencing with Earliness and Tardiness Penalties: A Review". **Operations Research,** 38, p. 22-36, 1990.

BAZARAA, M. S. & JARVIS, J. J.: Linear Programming and Network Flows. **John Wiley & Sons**, 1977.

BLAZEWICZ, J.; DROR, M.; WEGLARZ, J.:"Mathematical Programming Formulations for Machine Scheduling: A Survey". **European Journal of Operational Research,** 51, p. 283-300, 1991.

BLAZEWICZ, J.; ECKER, K.; SCHMIDT, G.; WEGLARZ, J.:"Scheduling in Computer and Manufacturing Systems". **Springer-Verlag,** 1993.

CHENG, T. C. E.; GUPTA, M. C.: "Survey of Scheduling Research Involving Due Date Determination Decisions". **European Journal of Operational Research,** 38, p. 156-166, 1989.

COELHO, M. A. E. & FRANÇA, P. M.: "Heurísticas para Programação em Máquinas Paralelas Não-Relacionadas Sujeitas a Divisão de Tarefas". **ENEGEP- UFSCar**, São Carlos-SP, 1995.

COELHO, M. A. E. & FRANÇA, P. M.: "Busca em Árvore para o Problema de Programação em Multiprocessadores com Custos de Atraso e Adiantamento". **XI Congresso Brasileiro de Automática,** USP - São Paulo-SP, 1996.

CONWAY, R. W. , W. L. MAXWELL, L. W. MILLER: "Teory of Scheduling" , **Addison-Wesley Reading, MA**, 1967.

DAVIS, J. S.; KANET, J. J.: "Single-Machine Scheduling with Early and Tardy Completion Costs". **Naval Research Logistics,** 40, pp. 85-101, 1993.

ESPUÑA, A.; PUIGJANER, L.: 'Solving the Production Planning Problem for Parallel Multiproduct Plants", **Chem. Eng. Res. Des.,** Vol 67, Nov. 1989.

FOURER, R.; GAY, D. M.; KERNIGHAN, B. W.: "AMPL : A Modeling Language for Mathematical Programming". **The Scientific Press,** 1993.

FOX, M. S.;"Contraint-directed search: A case study of job-shop scheduling." Ph.D. Thesis, Carnegie-Mellon University, U.S.A. 1983.

GAREY, M. R.; JOHNSON, D. S.: "Computers and Intractability: a guide to the theory of NP-completeness". **W. H. Freeman, New York,** 1979.

GAREY, M. R.; TARJAN, R. E.; WILFONG, G. T.: "One-Processor Scheduling with Symmetric Earliness and Tardiness Penalties". **Mathematics of Operations Research,** 13/2, pp. 330-348, 1988.

GUINET, A.: "Textile Production Systems: a Succession of Non-identical Parallel Processor Shops". **Journal of the Operational Research Society,** 42, p. 665-671, 1991

GAUDIOSO, M. & PASQUALE, L.: "Linear Programming Models for Load Balancing". **Computers and Operational Research,** 18, p. 59-64, 1991.

HARIRI, A. M. A. & POTTS, C. N. : "Heuristics for Scheduling Unrelated Parallel Machines". **Computers and Operational Research,** 18, p. 323-331, 1991.

LAWLER, E. L. & LABETTOULLE, J. : "On Preemptive Scheduling of Unrelated Parallel Processors by Linear Programming". **Journal of the Association for Computing Machinery,** 25, p. 612-619, 1978.

LAWLER, E. L.; LENSTRA, J. K.; RINNOOY KAN, A. H. G.; SHMOYS, D. B.: "Sequencing and Scheduling: Algoritms and Complexity". **Logistics of Production and Inventory,** Chap 9, North-Holland, 1993.

LENSTRA, J. K. & SHMOYS, D. B. & TARDOS, E.: "Approximations Algorithms for Scheduling Unrelated Parallel Machines". **Mathematical Programming,** 46, p. 259-271, 1990.

Mc CARTHY, B. L.; LIN, J.: "Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling". **International Journal of Production Research,** 31/1, pp. 59-79, 1993.

MORTON, T. E. & PENTICO, D. W.: "Heuristic Scheduling Systems: With Applications to Production Systems and Project Management, **Wiley, New York,** 1993.

OW, P.S.; MORTON, T. E.: "The Single Machine Early/Tardy Problem". **Management Science**, 35/2, pp. 177-191, 1989.

OW, P.S.; MORTON, T. E.: "Filtered beam search in scheduling". **International Journal of Production Research,** 26/1, pp. 35-62, 1988.

PEARL, J.: "HEURISTICS: Intelligent Search Strategies for Computer Problem Solving. **Addison-Wesley**, 1984.

POURBABAI, B. : "A Short Term Production Planning and Scheduling Model". **Engineering Costs and Production Economics,** 18, p. 159-167, 1989.

SERAFINI, P. & SPERANZA, M. G.: "Production Scheduling Problems in a Textile Industry". **European Journal of Operational Research,** 58, p. 173-190, 1992.

SHAPIRO, J F: "Mathematical Programming Models and Methods for Production Planning and Scheduling". **Logistics of Production and Inventory,** Chap 8, North-Holland, 1993.

APÊNDICE

A.1. Técnicas de resolução rápida do modelo de seqüenciamento e alocações fixas

A.1.1 Justificativa

O Modelo **F** (capítulo 4) poderá ser utilizado para encontrar soluções em problemas de porte médio com, possivelmente, 1000 frações de tarefas (como será visto adiante, a dimensão do problema de programação linear cresce com o número de frações de tarefas e não exatamente com N ou M). Em problemas deste tamanho, o espaço de memória utilizado, velocidade de processamento e precisão numérica começam a ficar críticos, pelo menos nos computadores atuais. Além disso, pretende-se viabilizar a utilização de microcomputadores para que futuras pesquisas sobre este tema não se limitem a grandes corporações usuárias de máquinas poderosas.

A resolução do modelo **F** utilizando o método Simplex padrão, em duas fases, é ineficiente para problemas com mais do que 20 frações, limitando severamente o tamanho dos problemas a serem resolvidos. Chegou-se a desenvolver um método capaz de gerar uma solução dual factível rearranjando as restrições do modelo e posteriormente resolvendo com um quadro Dual Simplex, o que evita a primeira fase com suas variáveis artificiais. Apesar da significativa melhora em relação ao Simplex padrão, continuava havendo forte limitação à resolução de problemas maiores. Optou-se então por um procedimento que apela para a decomposição de Dantzig-Wolfe aplicada ao problema.

Os métodos desenvolvidos nos próximos itens deste apêndice, utilizam particularidades do modelo **F**, na tentativa de atingir o objetivo acima. O caminho básico é:

- Utilizar o quadro Simplex de um grupo incompleto representando um nó da árvore de busca, para gerar os quadros dos grupos descendentes.
- Utilizar as características bloco diagonal do modelo, para compactar o quadro Simplex que representa a solução.

A maior complexidade deste enfoque será amplamente compensada com: velocidade de execução muito superior nos problemas de maior porte, espaço de armazenagem dramaticamente inferior e precisão numérica dentro de limites aceitáveis.

A.1.2 Adição de Tarefas ao Modelo

A Matriz de bases **B** do problema é formada pelas colunas da Matriz de coeficientes **A**, relativas às variáveis básicas. Partindo de uma solução parcial com n tarefas cuja matriz de bases é $\mathbf{B}^n$, a adição de uma nova tarefa em uma posição posterior às tarefas da solução atual, forneceria:

$$\mathbf{B}^{n+1} = \left[\begin{array}{c|c} \mathbf{B}^n & \mathbf{0} \\ \hline \mathbf{C}^{n+1} & \mathbf{B}^0 \end{array}\right] \qquad \text{(A. 1)}$$

onde:

$\mathbf{B}^n$ = Matriz de bases da solução parcial antes da adição da nova tarefa.

$\mathbf{B}^{n+1}$ = Matriz de bases da solução parcial após a adição da nova tarefa.

$\mathbf{B}^0$ = Matriz de bases da nova tarefa sozinha no modelo.

$\mathbf{C}^{n+1}$ = Matriz de interligação entre a nova tarefa e as anteriores.

0 = Matriz nula devido à imposição da nova tarefa ser posterior às antigas.

Na Matriz $\mathbf{C}^{n+1}$ de interligação, os únicos elementos não nulos estarão na coluna $a(i,k)$ e linha correspondente à restrição (4.2) da fração (j,k), se a nova tarefa for chamada de j e i for a tarefa anterior a j na máquina k. Estes elementos não nulos serão iguais a -1 e poderá haver no máximo F_j (número de frações ativas da nova tarefa) elementos não nulos, pois algumas frações poderão ser as primeiras a serem executadas em uma determinada máquina.

Portanto, a inversa da nova matriz de bases é:

$$\left(\mathbf{B}^{n+1}\right)^{-1} = \left[\begin{array}{c|c} \left(\mathbf{B}^{n}\right)^{-1} & \mathbf{0} \\ \hline -\left(\mathbf{B}^{0}\right)^{-1}.\mathbf{C}^{n+1}.\left(\mathbf{B}^{n}\right)^{-1} & \left(\mathbf{B}^{0}\right)^{-1} \end{array}\right] \qquad \text{(A. 2)}$$

Esta facilidade de gerar a inversa da matriz de bases da nova solução ampliada a partir da inversa da matriz de bases da antiga solução sugere a utilização do método Simplex Revisado para a execução do procedimento de otimização do modelo. As outras sub-matrizes do novo quadro Simplex Revisado podem ser deduzidas a partir da inversa da matriz de bases e das sub-matrizes do antigo quadro:

$$\overline{\mathbf{b}}^{n+1} = \left(\mathbf{B}^{n+1}\right)^{-1}.\mathbf{b}^{n+1} = \left(\mathbf{B}^{n+1}\right)^{-1}.\left[\begin{array}{c} \mathbf{b}^{n} \\ \hline \mathbf{b}^{0} \end{array}\right] = \left[\begin{array}{c} \overline{\mathbf{b}}^{n} \\ \hline \overline{\mathbf{b}}^{0} - \left(\mathbf{B}^{0}\right)^{-1}.\mathbf{C}^{n+1}.\overline{\mathbf{b}}^{n} \end{array}\right] \qquad \text{(A. 3)}$$

$$\mathbf{w}^{n+1} = \mathbf{c}_{\mathrm{B}}^{n+1}.\left(\mathbf{B}^{n+1}\right)^{-1} = \left[\mathbf{c}_{\mathrm{B}}^{n} \;\vdots\; \mathbf{c}_{\mathrm{B}}^{0}\right].\left(\mathbf{B}^{n+1}\right)^{-1} = \left[\mathbf{w}^{n} - \mathbf{w}^{0}.\mathbf{C}^{n+1}.\left(\mathbf{B}^{n}\right)^{-1} \;\vdots\; \mathbf{w}^{0}\right] \qquad \text{(A. 4)}$$

$$\text{f.obj.} = \mathbf{c}_{\mathrm{B}}^{n+1}.\overline{\mathbf{b}}^{n+1} = \left[\mathbf{c}_{\mathrm{B}}^{n} \;\vdots\; \mathbf{c}_{\mathrm{B}}^{0}\right] \times \left[\begin{array}{c} \overline{\mathbf{b}}^{n} \\ \hline \overline{\mathbf{b}}^{0} - \left(\mathbf{B}^{0}\right)^{-1}.\mathbf{C}^{n+1}.\overline{\mathbf{b}}^{n} \end{array}\right] = \mathbf{c}_{\mathrm{B}}^{n}.\overline{\mathbf{b}}^{n} + \mathbf{c}_{\mathrm{B}}^{0}.\overline{\mathbf{b}}^{0} - \mathbf{c}_{\mathrm{B}}^{0}.\left(\mathbf{B}^{0}\right)^{-1}.\mathbf{C}^{n+1}.\overline{\mathbf{b}}^{n}$$

(A. 5)

As expressões acima empregam as mesmas notações utilizadas em BAZARAA (1977), com os superíndices indicando a que quadro cada uma das sub-matrizes se refere. Para gerar um quadro Simplex Revisado pelo procedimento acima, é necessário primeiro obter o quadro Simplex Revisado da nova tarefa, com a alocação desejada, sozinha no modelo. Ele pode ser obtido, sem a utilização de variáveis artificiais, através de uma escolha apropriada de bases, o que será feito mais adiante. Também nos itens seguintes será demonstrada a factibilidade da nova solução gerada pelo procedimento acima.

A.1.3 Decomposição do Modelo

A.1.3.1 Decomposição de Dantzig-Wolfe aplicada ao modelo de seqüenciamentos e alocações fixas

O modelo de seqüenciamentos e alocações fixas apresentado no capítulo 4, possui características de uma estrutura bloco-diagonal, isto é, à exceção da restrição (4.2), as restrições e variáveis de uma mesma tarefa podem ser agrupadas em blocos independentes uns dos outros. A teoria abaixo foi adaptada ao modelo **F** a partir do método de decomposição de Dantzig-Wolfe encontrado em BAZARAA (1977) (*The Decomposition Principle - Block Diagonal Structure)*, com pequenas alterações na notação empregada.

Se X é um poliedro ou conjunto representando todas as restrições do problema, este pode ser decomposto em N conjuntos $X_1, X_2, ..., X_N$, cada um envolvendo restrições relativas a uma mesma tarefa. Da mesma forma o vetor **x**, contendo todas as variáveis do problema, é decomposto nos vetores $\mathbf{x}_1$, $\mathbf{x}_2$, ..., $\mathbf{x}_N$, cada um contendo todas as variáveis relativas a uma mesma tarefa; o vetor **c** em $\mathbf{c}_1$, $\mathbf{c}_2$, ..., $\mathbf{c}_N$ e a matriz **A**, das restrições principais (4.2), nas matrizes $\mathbf{A}_1$, $\mathbf{A}_2$, ..., $\mathbf{A}_N$, fornecendo o seguinte problema:

$$
\begin{aligned}
&\textit{Minimize} \quad \mathbf{c}_1\mathbf{x}_1 + \mathbf{c}_2\mathbf{x}_2 + \cdots + \mathbf{c}_N\mathbf{x}_N \\
&\text{Sujeito a:} \\
&\qquad \mathbf{A}_1\mathbf{x}_1 + \mathbf{A}_2\mathbf{x}_2 + \cdots + \mathbf{A}_N\mathbf{x}_N = \mathbf{b} \\
&\qquad \mathbf{B}_1\mathbf{x}_1 \leq \mathbf{b}_1 \\
&\qquad\quad \mathbf{B}_2\mathbf{x}_2 \leq \mathbf{b}_2 \\
&\qquad\qquad \ddots \quad \vdots \\
&\qquad\qquad\quad \mathbf{B}_N\mathbf{x}_N \leq \mathbf{b}_N \\
&\qquad\qquad \mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_N \geq 0
\end{aligned}
\qquad \text{(A. 6)}
$$

onde:

$$X_j = \{\mathbf{x}_j : \mathbf{B}_j\mathbf{x}_j \leq \mathbf{b}_j, \mathbf{x}_j \geq 0\} \quad , j = 1,2,\ldots,N$$

Supondo que os conjuntos X_j são limitados, qualquer ponto $\mathbf{x}_j \in X_j$ pode ser representado como uma combinação convexa de um número finito de pontos extremos de X_j, isto é:

$$\mathbf{x}_j = \sum_{\ell=1}^{t_j} \lambda_{j\ell} \mathbf{x}_{j\ell}$$

$$\sum_{\ell=1}^{t_j} \lambda_{j\ell} = 1 \qquad \text{(A. 7)}$$

$$\lambda_{j\ell} \geq 0 \qquad \ell = 1,2,\ldots,t_j$$

onde $\mathbf{x}_{j\ell}$ são pontos extremos do conjunto X_j e $\lambda_{j\ell}$ são números reais. A substituição de (A. 7) no problema original, fornece o problema mestre ou principal:

$$\textit{Minimize} \quad \sum_{j=1}^{N} \sum_{\ell=1}^{t_j} (\mathbf{c}_j \mathbf{x}_{j\ell}) \lambda_{j\ell}$$

Sujeito a:

$$\sum_{j=1}^{N} \sum_{\ell=1}^{t_j} (\mathbf{A}_j \mathbf{x}_{j\ell}) \lambda_{j\ell} = \mathbf{b} \qquad \text{(A. 8)}$$

$$\sum_{\ell=1}^{t_j} \lambda_{j\ell} = 1 \quad j = 1,2,\ldots,N$$

$$\lambda_{j\ell} \geq 0 \quad j = 1,2,\ldots,N, \quad \ell = 1,2,\ldots,t_j$$

O algoritmo de decomposição pode ser visto como uma aplicação do Método Simplex Revisado ao problema mestre. Supondo que seja conhecida uma solução básica factível para o problema, cuja base é **B** e também que sejam conhecidos,

$$\overline{\mathbf{b}} = \mathbf{B}^{-1} \begin{bmatrix} \mathbf{b} \\ \hline \mathbf{1} \end{bmatrix} \quad \therefore \quad [\mathbf{w} \mid \mathbf{alfa}] = [w_1 \quad \cdots \quad w_f \mid alfa_1 \quad \cdots \quad alfa_N] = \hat{\mathbf{c}}_{\mathbf{B}} \mathbf{B}^{-1} \qquad \text{(A. 9)}$$

onde $\hat{\mathbf{c}}_{\mathbf{B}}$ é o custo relativo das variáveis básicas ($\hat{c}_{j\ell} = \mathbf{c}_j \mathbf{x}_{j\ell}$ para $\lambda_{j\ell}$). A solução será ótima se $z_{j\ell} - \hat{c}_{j\ell} \leq 0$ para cada variável não-básica (para as variáveis básicas $z_{j\ell} - \hat{c}_{j\ell} = 0$), isto é:

$$0 \geq z_{j\ell} - \hat{c}_{j\ell} = [\mathbf{w} \mid \mathbf{alfa}] \times \begin{bmatrix} \mathbf{A}_j \mathbf{x}_{j\ell} \\ \hline 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} - \mathbf{c}_j \mathbf{x}_{j\ell} = (\mathbf{w}.\mathbf{A}_j - \mathbf{c}_j)\mathbf{x}_{j\ell} + alfa_j \qquad \text{(A. 10)}$$

A condição acima pode ser verificada resolvendo o seguinte subproblema:

$$\textit{Maximize} \quad (\mathbf{w}.\mathbf{A}_j - \mathbf{c}_j)\mathbf{x}_j + alfa_j$$

sujeito a: (A. 11)

$$\mathbf{x}_j \in X_j$$

A solução do subproblema ($\mathbf{x}_{j\ell}$, que é um ponto extremo da região X_j), se tiver um valor de função objetivo positiva, indicará que $\lambda_{j\ell}$ é candidato a entrar na base. Quando nenhum subproblema possuir uma solução com valor da função objetivo positiva, a condição de otimalidade ($z_{j\ell} - \hat{c}_{j\ell} \leq 0$ para cada variável não-básica) terá sido atingida e, portanto, a solução do problema principal será ótima.

O método de decomposição não exige, mas nisso reside a sua potencialidade, porém o ideal é que os subproblemas possam ser resolvidos de uma forma que dispense os quadros Simplex. Felizmente, no caso do modelo **F**, existe uma solução analítica capaz de resolver os subproblemas de forma rápida e exata.

A.1.3.2 Particularização do problema

Seja N o número de tarefas e M o número de máquinas, F o número total de frações de tarefas existentes e F_j o número de frações existentes (ou ativas) na tarefa j. O modelo **F** possuirá:

- F restrições (4.2), (4.4), (4.5).
- N restrições (4.3).
- $3F + 2N$ variáveis.

A decomposição do item anterior, implicará em:

- vetores linha $\mathbf{c}_j \rightarrow 3F_j + 2$ componentes.
- vetores coluna $\mathbf{x}_j \rightarrow 3F_j + 2$ componentes.

$$\mathbf{c}_j = \begin{bmatrix} \alpha_j & \beta_j & 0 & \cdots & 0 \end{bmatrix} \quad \therefore \quad \mathbf{x}_j = \begin{bmatrix} e(j) \\ z(j) \\ a(j,k_1) \\ r(j,k_1) \\ id(j,k_1) \\ \vdots \\ a(j,k_{F_j}) \\ r(j,k_{F_j}) \\ id(j,k_{F_j}) \end{bmatrix} \qquad \text{(A. 12)}$$

onde $a(j,k_1), r(j,k_1), id(j,k_1)$ significam as variáveis relativas à primeira das máquinas onde existe uma fração da tarefa j alocada e $a(j,k_{F_j}), r(j,k_{F_j}), id(j,k_{F_j})$ as variáveis relativas à última máquina onde existe uma fração da tarefa j alocada.

- matrizes $\mathbf{A}_j \rightarrow (F)\mathrm{X}(3F_j + 2)$ componentes.

- vetor coluna $\mathbf{b} \rightarrow F$ componentes.

$$\mathbf{A}_j = \begin{bmatrix} 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \iddots & \vdots \\ 0 & \cdots & 1 & -s_{jk} & -1 & \cdots & 0 \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & \cdots & -1 & 0 & 0 & \cdots & 0 \\ \vdots & \iddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \quad \therefore \quad \mathbf{b} = \begin{bmatrix} c_{01A} \\ \vdots \\ c_{ijk} \\ \vdots \\ c_{jpk} \\ \vdots \\ c_{lNM} \end{bmatrix} \qquad \text{(A. 13)}$$

onde i é a tarefa anterior a j na máquina k, p é a tarefa posterior a j na máquina k, c_{01A} e c_{lNM} são tempos de troca da primeira e última frações entre todas consideradas. Cada linha da matriz $\mathbf{A}_j$ se refere à restrição (4.2) relativa a uma fração ativa qualquer. Como a tarefa j possui apenas F_j frações ativas, somente $4.F_j$ elementos de $\mathbf{A}_j$ serão não-nulos, considerando que todas as frações de j possuem frações posteriores, ou 4 elementos não nulos para cada $k \in M_j$.

- matrizes $\mathbf{B}_j \rightarrow (2F_j + 1)\mathrm{X}(3F_j + 2)$ componentes.

- vetores coluna $\mathbf{b}_j \rightarrow 2F_j + 1$ componentes.

$$\mathbf{B}_j = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & \cdots & 0 \\ 1 & 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & -1 & 1 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & -1 & 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \quad \therefore \quad \mathbf{b}_j = \begin{bmatrix} q_j \\ d_j \\ d_j \\ \vdots \\ d_j \\ d_j \end{bmatrix} \qquad \text{(A. 14)}$$

A primeira linha de $\mathbf{B}_j$ se refere à restrição (4.3), enquanto as outras se referem às restrições (4.5) e (4.4), nesta ordem.

O problema mestre será:

$$\textit{Minimize} \quad \sum_{j\in T}\sum_{\ell=1}^{t_j}(\beta_j.z_\ell(j)+\alpha_j.e_\ell(j))\lambda_{j\ell}$$

Sujeito a:

$$\sum_{\ell=1}^{t_j}(a_\ell(j,k)-id_\ell(j,k)-s_{jk}.r_\ell(j,k))\lambda_{j\ell}-\sum_{\ell=1}^{t_i}a_\ell(i,k)\lambda_{i\ell}=c_{ijk} \qquad \begin{matrix} k\in P \\ j\in K_k \\ i\in A_{jk}\end{matrix} \qquad \text{(A. 15)}$$

$$\sum_{\ell=1}^{t_j}\lambda_{j\ell}=1 \quad j\in T$$

$$\lambda_{j\ell}\geq 0 \quad j\in T, \quad \ell=1,2,\ldots,t_j$$

onde $z_\ell(j)$, $e_\ell(j)$, $a_\ell(j,k)$, $r_\ell(j,k)$, $id_\ell(j,k)$ são componentes do vetor $\mathbf{x}_{j\ell}$ e, portanto, são constantes neste problema.

Os subproblemas serão, para $j \in T$:

$$\textit{Maximize} \quad -\beta_j.z_\ell(j)-\alpha_j.e_\ell(j)+\sum_{k\in M_j}\{w(j,k)-w(p,k)\}.a_\ell(j,k)$$
$$-\sum_{k\in M_j}w(j,k).s_{jk}.r_\ell(j,k)-\sum_{k\in M_j}w(j,k).id_\ell(j,k)+alfa_j$$

Sujeito a:

$$\begin{aligned}
&\sum_{k\in M_j}r_\ell(j,k)=q_j \\
&a_\ell(j,k)-z_\ell(j)\leq d_j, \quad k\in M_j \\
&a_\ell(j,k)+e_\ell(j)\geq d_j, \quad k\in M_j \\
&a_\ell(j,k)\leq HV, \qquad k\in M_j \\
&id_\ell(j,k)\leq HV, \qquad k\in M_j \\
&z_\ell(j)\geq 0, e_\ell(j)\geq 0, a_\ell(j,k)\geq 0, r_\ell(j,k)\geq 0, id_\ell(j,k)\geq 0, \quad k\in M_j
\end{aligned} \qquad \text{(A. 16)}$$

onde $z_\ell(j)$, $e_\ell(j)$, $a_\ell(j,k)$, $r_\ell(j,k)$, $id_\ell(j,k)$ (componentes do vetor $\mathbf{x}_{j\ell}$) são as variáveis do subproblema. Os parâmetros $w(j,k)$ que surgiram na função objetivo, são as componentes do vetor **w**, colocadas em função da fração de tarefa a que se referem. Observe que foram introduzidas duas restrições, não presentes no modelo original, com o objetivo de não permitir o crescimento das variáveis $a_\ell(j,k)$ e $id_\ell(j,k)$ acima de valores razoáveis (HV). Isto torna o subproblema limitado e evita a necessidade de introduzir direções extremas, além dos pontos $\mathbf{x}_{j\ell}$ já introduzidos. Escolhendo valores suficientemente grandes para HV (maiores do que os valores de $a(j,k)$ e $id(j,k)$ nas soluções percorridas durante as iterações do problema mestre), as

restrições adicionais não alterarão os resultados do modelo decomposto em relação aos obtidos com o modelo padrão.

A.1.3.3 Solução analítica dos subproblemas

A variável $id_\ell(j,k)$ está presente em um termo da função objetivo dos subproblemas e em restrições onde não existem outras variáveis, o que implica em:

$id_\ell(j,k) = 0$ para $w(j,k) \geq 0$

$id_\ell(j,k) = HV$ para $w(j,k) < 0$ (A. 17)

Também a variável $r_\ell(j,k)$ está presente em um termo da função objetivo dos subproblemas e em restrições onde não existem outras variáveis, o que implica em:

$r_\ell(j,k) = q_j$ para o menor $w(j,k).s_{jk}$

$r_\ell(j,k) = 0$ para os demais $k \in M_j$ (A. 18)

Removendo os termos da função objetivo e restrições relativas às variáveis $id_\ell(j,k)$, $r_\ell(j,k)$ e $alfa_j$ resta o seguinte subproblema reduzido, para $j \in T$:

$$Maximize \quad -\beta_j.z_\ell(j) - \alpha_j.e_\ell(j) + \sum_{k \in M_j} \{w(j,k) - w(p,k)\}.a_\ell(j,k)$$

Sujeito a:

$$\begin{aligned} a_\ell(j,k) - z_\ell(j) &\leq d_j, \quad k \in M_j \\ a_\ell(j,k) + e_\ell(j) &\geq d_j, \quad k \in M_j \\ a_\ell(j,k) &\leq HV, \qquad k \in M_j \\ z_\ell(j) \geq 0, e_\ell(j) \geq 0, a_\ell(j,k) &\geq 0, \quad k \in M_j \end{aligned} \qquad \text{(A. 19)}$$

No subproblema reduzido acima, qualquer solução com $HV \geq a_\ell(j,k) \geq 0$ pode ser factível, o que facilita a análise através de hipóteses.

Se em uma determinada solução, $a_\ell(j,k)$ é o maior tempo de término entre as frações da tarefa e $a_\ell(j,k) \geq d_j$, o atraso da tarefa é determinado por esta fração. Se uma outra fração $a_\ell(j,k')$ possuir $\{w(j,k') - w(p,k')\} > 0$, a função objetivo do subproblema ficará maior com o crescimento de $a_\ell(j,k')$. Como esta última fração não determina a atraso da tarefa, estará livre para crescer e aumentar a função objetivo, mas não poderá ir além de $a_\ell(j,k)$, sob pena de carregar o atraso. A conclusão é que, na solução ótima do subproblema reduzido, todas as

frações com $\{w(j,k) - w(p,k)\} > 0$ possuirão o mesmo tempo de término, que será chamado de $a_{max}(j)$. É claro que $a_{max}(j) \geq d_j$, pois apenas o atraso e HV podem limitar o seu crescimento.

De forma análoga, como o adiantamento é calculado pela fração com menor tempo de término, todas as frações com $\{w(j,k) - w(p,k)\} < 0$, possuirão o mesmo tempo de término, que será chamado de $a_{min}(j) \leq d_j$.

Define-se então os seguintes parâmetros:

$$W^+(j) = \sum_{\substack{k \in M_j \\ w(j,k) > w(p,k)}} \{w(j,k) - w(p,k)\} \quad = \text{Somatório dos termos } \{w(j,k) - w(p,k)\} > 0. \quad \text{(A. 20)}$$

$$W^-(j) = \sum_{\substack{k \in M_j \\ w(j,k) < w(p,k)}} \{w(j,k) - w(p,k)\} \quad = \text{Somatório dos termos } \{w(j,k) - w(p,k)\} < 0. \quad \text{(A. 21)}$$

A substituição dos novos parâmetros na função objetivo do subproblema reduzido deve ser feita com hipóteses apropriadas, para lidar corretamente com atrasos e adiantamentos.

1) Todos $\{w(j,k) - w(p,k)\} = 0$.

Neste caso o máximo do subproblema reduzido será atingido com $z_\ell(j) = e_\ell(j) = 0$, o que implica em $a_\ell(j,k) = d_j$, para $k \in M_j$

2) Todos $\{w(j,k) - w(p,k)\} > 0$.

Neste caso, $a_\ell(j,k) = a_{max}(j)$, para $k \in M_j$, $e_\ell(j) = 0$ e $z_\ell(j) = a_{max}(j) - d_j$. Substituindo na função objetivo do subproblema reduzido, tem-se:

$$\textit{maximize} \quad \{W^+(j) - \beta_j\} a_{max}(j) + \beta_j . d_j \qquad \text{(A. 22)}$$

3) Todos $\{w(j,k) - w(p,k)\} < 0$.

Neste caso, $a_\ell(j,k) = a_{min}(j)$, para $k \in M_j$, $z_\ell(j) = 0$ e $e_\ell(j) = d_j - a_{min}(j)$. Substituindo na função objetivo do subproblema reduzido, tem-se:

$$\textit{maximize} \quad \{W^-(j) + \alpha_j\} a_{min}(j) - \alpha_j . d_j \qquad \text{(A. 23)}$$

4) Caso geral englobando todos os outros.

$a_\ell(j,k) = a_{min}(j)$, $a_\ell(j,k) = d_j$ ou $a_\ell(j,k) = a_{max}(j)$, conforme $\{w(j,k) - w(p,k)\}$ correspondente seja negativo, nulo ou positivo.

Juntando os casos 2) e 3), conclui-se que:

$$\begin{aligned} &a_{max}(j) = HV \text{ se } \{W^+(j) - \beta_j\} > 0 \\ &a_{max}(j) = d_j \text{ se } \{W^+(j) - \beta_j\} \leq 0 \end{aligned} \qquad \text{(A. 24)}$$

$$a_{min}(j) = d_j \text{ se } \{W^-(j) + \alpha_j\} \geq 0$$
$$a_{min}(j) = 0 \text{ se } \{W^-(j) + \alpha_j\} < 0 \qquad \text{(A. 25)}$$

$$z_\ell(j) = a_{\max}(j) - d_j \qquad \text{(A. 26)}$$

$$e_\ell(j) = d_j - a_{\min}(j) \qquad \text{(A. 27)}$$

A solução pelo procedimento indicado acima é ótima e pode ser obtida em um tempo bastante curto em comparação com o método Simplex aplicado aos subproblemas, o que viabiliza a utilização da decomposição em problemas de dimensão média.

A.1.3.4 Solução inicial do problema mestre

Considerando que o problema mestre será resolvido com um quadro Simplex Revisado e que o procedimento de adição de tarefas poderá ser utilizado, será suficiente encontrar uma solução inicial factível para o caso de apenas uma tarefa j e F_j frações ativas.

Apesar do método de decomposição utilizar pontos extremos oriundos dos subproblemas, não é necessário utilizá-los numa solução inicial. Nesta, qualquer ponto do conjunto X_j poderá estar na base, mesmo porque após algumas iterações estes serão substituídos pelos pontos extremos que entrarão na base e mesmo que não sejam, a factibilidade da solução final estará garantida se os pontos iniciais estiverem em X_j, como pode ser verificado em (A. 7).

Suponhamos que $\mathbf{x}_{joo}$ representa uma solução factível para a tarefa j com F_j frações ativas sozinhas no modelo. O problema mestre possui uma restrição para cada fração de tarefa ativa e mais uma para cada tarefa, o que neste caso totaliza $F_j + 1$ restrições, $F_j + 1$ pontos "extremos" e $F_j + 1$ variáveis λ correspondentes a cada ponto. Chamando de λ_{joo} a variável correspondente a $\mathbf{x}_{joo}$ e λ_{jok} a variável correspondente ao ponto $\mathbf{x}_{jok}$, é possivel criar uma solução factível para o problema mestre, onde $\lambda_{joo} = 1$ e $\lambda_{jok} = 0$ para $k \in M_j$. Como $\mathbf{x}_{joo}$ já representa uma solução factível para o modelo, a única exigência sobre os outros pontos iniciais é que $\mathbf{x}_{jok} \in X_j$, pois não causarão qualquer influência sobre a solução do modelo, como pode ser verificado em (A. 7). Esta liberdade será útil para fabricar uma solução inicial em que a matriz de bases **B** é particularmente simples.

As componentes dos pontos iniciais $\mathbf{x}_{jok}$ serão as seguintes:

fração j,k	outras frações
$a_{ok}(j,k) = \tau + s_{jk}.q_j$	$a_{ok}(j,k') = d_j$

$$
\begin{array}{llll}
r_{ok}(j,k) = q_j & & r_{ok}(j,k') = 0 & \\
id_{ok}(j,k) = 0 & & id_{ok}(j,k') = d_j & \text{(A. 28)} \\
z_{ok}(j) = \tau + s_{jk}.q_j - d_j \quad \text{se} \quad \tau + s_{jk}.q_j \geq d_j & \text{ou} & z_{ok}(j) = 0 & \text{caso contrário.} \\
e_{ok}(j) = 0 \quad \text{se} \quad \tau + s_{jk}.q_j \geq d_j & \text{ou} & e_{ok}(j) = d_j - \tau - s_{jk}.q_j & \text{caso contrário.}
\end{array}
$$

onde τ é uma constante com dimensão de tempo a ser definida posteriormente junto com HV. A solução em $\mathbf{x}_{jok}$ não precisa ser factível para o modelo completo (como não é). Precisa apenas satisfazer as restrições do subproblema j.

Para facilitar a compreensão, será considerado um caso em que existem apenas 3 frações ativas nas máquinas A, B, C. Substituindo $\mathbf{x}_{joo}$ e $\mathbf{x}_{jok}$ dados acima nas restrições do problema mestre (A. 15), tem-se:

$$
\begin{aligned}
&\tau.\lambda_{joA} + \{a_{oo}(j,A) - id_{oo}(j,A) - s_{jA}.\, r_{oo}(j,A)\}.\, \lambda_{joo} = co_{ijA} \\
&\tau.\lambda_{joB} + \{a_{oo}(j,B) - id_{oo}(j,B) - s_{jB}.\, r_{oo}(j,B)\}.\, \lambda_{joo} = co_{ijB} \\
&\tau.\lambda_{joC} + \{a_{oo}(j,C) - id_{oo}(j,C) - s_{jC}.\, r_{oo}(j,C)\}.\, \lambda_{joo} = co_{ijC} \\
&\lambda_{joA} + \lambda_{joB} + \lambda_{joC} + \lambda_{joo} = 1 \qquad \text{(A. 29)}
\end{aligned}
$$

onde os termos com $a_\ell(i,k)$ foram considerados nulos, uma vez que o tratamento adequado entre tarefas anteriores e posteriores é feito pelo procedimento de adição de tarefas no item A.1.2. Para considerar os termos constantes $a(0,A)$, $a(0,B)$ e $a(0,C)$ que são os instantes de disponibilidade inicial em cada máquina, estes são somados aos tempos de troca iniciais c_{0jk}, formando os novos parâmetros co_{ijk}. Isto é:

$$co_{ijk} = c_{ijk} \text{ para } i \in T\,(i \neq 0) \text{ e } co_{0jk} = c_{0jk} + a(0,k) \text{ para } j \in T\,;\, k \in P. \qquad \text{(A. 30)}$$

Os tempos de troca utilizados em (A. 29) não serão necessariamente os tempos de troca iniciais c_{0jk}, porque a tarefa considerada, não é necessariamente a primeira a ser executada, embora no momento esteja sozinha no modelo. Os coeficientes de λ_{joo} nas três primeiras equações, são iguais ao lado direito destas, pois $\lambda_{joo} = 1$ e $\lambda_{joA} = \lambda_{joB} = \lambda_{joC} = 0$, na primeira solução do problema mestre. A matriz $\mathbf{B}^0$ corresponde aos coeficientes das variáveis básicas nas restrições do problema e $\mathbf{b}^0$ ao lado direito das restrições. A partir de (A. 29):

$$\mathbf{B}^0 = \begin{bmatrix} \tau & 0 & 0 & co_{ijA} \\ 0 & \tau & 0 & co_{ijB} \\ 0 & 0 & \tau & co_{ijC} \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad \therefore \quad \mathbf{b}^0 = \begin{bmatrix} co_{ijA} \\ co_{ijB} \\ co_{ijC} \\ 1 \end{bmatrix} \qquad \text{(A. 31)}$$

Ou, no caso geral:

$$\mathbf{B}^0 = \left[\begin{array}{c|c} \tau.\mathbf{I} & [co_{ij}]^t \\ \hline [1] & 1 \end{array}\right] \qquad \therefore \qquad \mathbf{b}^0 = \left[\begin{array}{c} [co_{ij}]^t \\ \hline 1 \end{array}\right] \tag{A. 32}$$

onde [1] é um vetor linha com F_j componentes iguais a 1, $[co_{ij}]^t$ é um vetor coluna com F_j componentes co_{ijk} e **I** é uma matriz identidade de ordem F_j. A matriz inversa de $\mathbf{B}^0$, será:

$$(\mathbf{B}^0)^{-1} = \frac{1}{\Delta}\left[\begin{array}{c|c} \frac{1}{\tau}\left\{\Delta.\mathbf{I} + [co_{ij}]^t.[1]\right\} & -[co_{ij}]^t \\ \hline -[1] & \tau \end{array}\right] \qquad \therefore \qquad \overline{\mathbf{b}}^0 = \left[\begin{array}{c} [0]^t \\ \hline 1 \end{array}\right] \tag{A. 33}$$

onde $\Delta = \tau$ - [1]. $[co_{ij}]^t$ e $[0]^t$ é um vetor coluna com F_j componentes iguais a 0. Para o caso das três frações,

$$(\mathbf{B}^0)^{-1} = \frac{1}{\Delta}\begin{bmatrix} (\Delta + co_{ijA})/\tau & co_{ijA}/\tau & co_{ijA}/\tau & -co_{ijA} \\ co_{ijB}/\tau & (\Delta + co_{ijB})/\tau & co_{ijB}/\tau & -co_{ijB} \\ co_{ijC}/\tau & co_{ijC}/\tau & (\Delta + co_{ijC})/\tau & -co_{ijC} \\ -1 & -1 & -1 & \tau \end{bmatrix} \quad \therefore \quad \overline{\mathbf{b}}^0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

onde: (A. 34)

$\Delta = \tau - (co_{ijA} + co_{ijB} + co_{ijB})$

O vetor $\hat{\mathbf{c}}_{\mathbf{B}}^0$ corresponde aos coeficientes das variáveis básicas na função objetivo, que de acordo com o problema mestre (A. 15), cada componente será $fo_\ell = \beta_j.z_\ell(j) + \alpha_j.e_\ell(j)$, e o vetor $\mathbf{w}^0$ será:

$$\mathbf{w}^0 = \hat{\mathbf{c}}_{\mathbf{B}}^0.(\mathbf{B}^0)^{-1} = \left[[fo_{jok}] \;\middle|\; fo_{joo}\right].(\mathbf{B}^0)^{-1} = \left[\frac{1}{\tau}[fo_{jok}] - \frac{1}{\Delta}(fo_{joo} - \Sigma fc).[1] \;\middle|\; \frac{\tau}{\Delta}(fo_{joo} - \Sigma fc)\right]$$

onde:

$$\Sigma fc = \frac{[fo_{jok}] \times [co_{ij}]^t}{\tau} = \sum_{k \in M_j} \frac{fo_{jok}.co_{ijk}}{\tau}$$

(A. 35)

Finalmente, a função objetivo do problema mestre, como esperado, é:

$$\text{f.obj} = \hat{\mathbf{c}}_{\mathbf{B}}^0.\overline{\mathbf{b}}^0 = fo_{joo} = \beta_j.z_{oo}(j) + \alpha_j.e_{oo}(j) \tag{A. 36}$$

A.1.3.5 Algoritmo para a solução ótima de uma tarefa

Qualquer solução factível pode ser usada em $\mathbf{x}_{joo}$, mas como mostra (A. 36), se for possível gerar uma solução ótima por um algoritmo qualquer, o quadro mestre inicial do problema decomposto também será ótimo e dispensará iterações para a busca da otimalidade (estas serão necessárias apenas no quadro adicionado pelo procedimento do item A.1.2).

Como a tarefa é a única no modelo, não existe acoplamento de outras, significando que pelo menos o adiantamento pode sempre ser anulado calculando tempos ociosos adequados antes das frações. O atraso será mínimo se os tempos ociosos forem anulados e os tempos de término das frações igualado, com a maior divisão possível para a tarefa. O procedimento do item 6.2.1 com $a(i,k) = 0$ para todo $k \in M_j$ pode ser utilizado no cálculo das componentes de $\mathbf{x}_{joo}$, o que minimizará o valor da função objetivo em (A. 36).

A.1.3.6 Coluna de pivoteamento do problema mestre

Depois de resolver todos os subproblemas e selecionar o que possui maior função objetivo $fo_{j\ell}$ e ponto extremo correspondente $\mathbf{x}_{j\ell}$, a seguinte coluna é gerada:

$$(\mathbf{B}^n)^{-1} \times \begin{bmatrix} \mathbf{A}_j \mathbf{x}_{j\ell} \\ \hline 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix} = \sum_{k \in M_j} \left(\left\{ a_\ell(j,k) - id_\ell(j,k) - s_{jk} . r_\ell(j,k) \right\} . \begin{bmatrix} \text{coluna} \\ (j,k) \\ \text{de} \\ (\mathbf{B}^n)^{-1} \end{bmatrix} - a_\ell(j,k) . \begin{bmatrix} \text{coluna} \\ (p,k) \\ \text{de} \\ (\mathbf{B}^n)^{-1} \end{bmatrix} \right) + \begin{bmatrix} \text{coluna} \\ alfa_j \\ \text{de} \\ (\mathbf{B}^n)^{-1} \end{bmatrix} \tag{A. 37}$$

onde (p,k) é a fração posterior a (j,k). Se não existir uma fração posterior a (j,k), a coluna (p,k) será nula. A indicação das colunas de $(\mathbf{B}^n)^{-1}$ pela fração ou $alfa_j$ correspondente, é conveniente pois, na prática, estas colunas estarão ordenadas de uma forma complexa, devido ao procedimento de adição de tarefas e possuirão rótulos de identificação para uma correta correspondência. O ordenamento é o mesmo das linhas de $\mathbf{B}^n$ que recebem o rótulo da fração relativa a cada restrição, que é adicionada ao problema.

Para completar a coluna de pivoteamento, uma nova componente contendo $fo_{j\ell}$ é colocada na linha correspondente à $\mathbf{w}^n$ do quadro Simplex Revisado, normalmente a primeira componente da coluna.

A.1.3.7 Factibilidade na adição de tarefas

O procedimento de adição de tarefas, descrito no item A.1.2, pode gerar uma solução infactível ao transformar o quadro gerado pelos itens acima, o que pode ser evitado com uma correta escolha do parâmetro τ. O valor das variáveis básicas da nova solução adicionada, dado em (A. 3), não pode ser negativo, o que justifica um cálculo explícito destas. De (A. 15), conclui-se que:

$$\mathbf{C}^{n+1}.\overline{\mathbf{b}}^{n} = \left[\begin{array}{ccccccccc} 0 & \cdots & 0 & -a_1(i,k_1) & \cdots & -a_{t_i}(i,k_1) & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & -a_1(i,k_{F_j}) & \cdots & -a_{t_i}(i,k_{F_j}) & 0 & \cdots & 0 \\ \hline 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \end{array}\right] \times \left[\begin{array}{c} \vdots \\ \vdots \\ \vdots \\ \lambda_{i1} \\ \vdots \\ \lambda_{it_i} \\ \vdots \\ \vdots \\ \vdots \end{array}\right] = \left[\begin{array}{c} -a(i,k_1) \\ \vdots \\ -a(i,k_{F_j}) \\ \hline 0 \end{array}\right]$$

(A. 38)

onde a ordem das linhas e colunas na matriz $\mathbf{C}^{n+1}$ se torna diferente da descrição no item A.1.2, devido ao procedimento de adição de tarefas. Portanto, as novas componentes do vetor $\overline{\mathbf{b}}^{n+1}$ dado em (A. 3), serão:

$$\overline{\mathbf{b}}^{0} - (\mathbf{B}^{0})^{-1}.\mathbf{C}^{n+1}.\overline{\mathbf{b}}^{n} = \left[\begin{array}{c} 0 \\ \vdots \\ 0 \\ \hline 1 \end{array}\right] - (\mathbf{B}^{0})^{-1}.\left[\begin{array}{c} -a(i,k_1) \\ \vdots \\ -a(i,k_{F_j}) \\ \hline 0 \end{array}\right] = \frac{1}{\tau}\left[\begin{array}{c} a(i,k_1) + co_{ijk_1}.\Sigma a\Delta \\ \vdots \\ a(i,k_{F_j}) + co_{ijk_{F_j}}.\Sigma a\Delta \\ \hline \tau(1-\Sigma a\Delta) \end{array}\right]$$

onde: (A. 39)

$$\Sigma a\Delta = \frac{1}{\Delta}\sum_{k\in M_j} a(i,k)$$

Apenas a última componente pode se tornar negativa, portanto, a factibilidade do quadro adicionado ficará garantida se:

$$1 - \Sigma a\Delta \geq 0 \quad \Rightarrow \quad \tau \geq \sum_{k\in M_j}\left\{a(i,k) + co_{ijk}\right\} \qquad \text{(A. 40)}$$

A.1.3.8 Escolha de τ e *HV* e normalizações

As restrições que devem ser obedecidas por τ são:

- De (A. 33):

$$\Delta > 0 \quad \Rightarrow \quad \tau > \sum_{k\in M_j} co_{ijk} \qquad \text{(A. 41)}$$

- De (A. 40):

$$\tau \geq \sum_{k\in M_j}\left\{a(i,k) + co_{ijk}\right\} \qquad \text{(A. 42)}$$

Numericamente, existe a conveniência de manter Δ afastado de zero, o que não ocorreria se $a(i,k) = co_{ijk} = 0$ para $k \in M_j$. Portanto, é razoável inserir o tempo de processamento da tarefa j em (A. 42), para chegar a:

$$\tau \geq \sum_{k \in M_j} a(j,k) \qquad \text{(A. 43)}$$

Como se pretende utilizar τ como base para uma normalização dos tempos envolvidos, este deve ser comum a todas as tarefas, o que leva a:

$$\tau \geq \max_{j \in T} \left\{ \sum_{k \in M_j} a(j,k) \right\} \qquad \text{(A. 44)}$$

Se a tarefa com maior d_j possuir um grande α_j e for colocada como primeira da seqüência em um problema de 1 máquina, é possível que todas as outras se coloquem além de d_j. Portanto, é possível que o maior d_j ou o maior $a(0,k)$ seja o ponto de partida. Também é preciso considerar o maior tempo de troca possível para cada tarefa e que a alocação ocorrerá na máquina com maior tempo de processamento. Juntando isto a (A. 44), chega-se a:

$$\tau = M. \sum_{j \in T} \left\{ \max_{k \in P} (s_{jk}.q_j) + \max_{\substack{k \in P \\ i \in T0}} (c_{ijk}) \right\} + M. \max \left\{ \max_{j \in T} (d_j), \max_{k \in P} (a(0,k)) \right\} \qquad \text{(A. 45)}$$

onde o produto por M nos dois termos, ocorre devido à possibilidade de haver frações (mesmo que nulas) alocadas em todas as máquinas. O parâmetro HV, até poderia ser calculado por (A. 45) com $M = 1$, mas devido aos pontos iniciais contidos em $\mathbf{x}_{jok}$ (A. 28), a factibilidade estará sempre garantida se:

$$HV = \tau + \max_{\substack{j \in T \\ k \in P}} (s_{jk}.q_j) \qquad \text{(A. 46)}$$

O valor de τ calculado por (A. 45) logo após a entrada dos dados, servirá como base na normalização dos parâmetros envolvendo tempo. Isto é:

$$\tau^{\tau} = \frac{\tau}{\tau} = 1 \quad \therefore \quad HV^{\tau} = \frac{HV}{\tau}$$
$$s_{jk}^{\tau} = \frac{s_{jk}}{\tau} \quad \therefore \quad c_{ijk}^{\tau} = \frac{c_{ijk}}{\tau} \quad \therefore \quad d_j^{\tau} = \frac{d_j}{\tau} \qquad \text{(A. 47)}$$

A normalização simplifica as expressões utilizadas na decomposição, economizando operações de ponto flutuante e ainda melhorando a precisão numérica nos casos em que a unidades de tempo dos dados de entrada se afastam muito de 1. Na saída, todas as variáveis e a função objetivo, à exceção de $r(j,k)$, deverão ser multiplicados por τ, para que os verdadeiros valores sejam conhecidos. É recomendável que os parâmetros relativos à custos e quantidades,

também sejam normalizados com bases próprias, ou indicados em unidades tais que os valores não se afastem muito da unidade.

A.1.4 Exemplos usando decomposição

Os resultados de três exemplos de tamanhos diferentes, foram executados com o método Dual Simplex, e com a decomposição, fornecendo os seguintes resultados:

N	*M*	Frações	f. obj	tempo Dual	tempo Decomp
3	3	9	96.8549	.06 seg	.03 seg
6	3	18	35.1978	.66 seg	.03 seg
9	4	36	54.2488	15.19 seg	.28 seg

As duas técnicas de resolução foram programadas em linguagem C, (IBM C/C++ para OS/2) e executadas em um microcomputador tipo PC AT 486 DX2 66 Mhz. É importante esclarecer que os tempos acima se referem à resolução de apenas um grupo de cada problema onde todas as frações de todas as tarefas estão presentes (um nó da árvore de busca). No procedimento de busca os tempos seriam muito maiores, pois diversos grupos em diversos gabaritos precisam ser resolvidos.