UNIVERSIDADE FEDERAL DO AMAZONAS
INSTITUTO DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

ESTRATÉGIAS ALGORÍTMICAS EXATAS E HÍBRIDAS PARA PROBLEMAS DE ESCALONAMENTO EM MÁQUINAS PARALELAS COM PENALIDADES DE ANTECIPAÇÃO E ATRASO

RAINER XAVIER DE AMORIM

Outubro de 2017

Manaus - AM

RAINER XAVIER DE AMORIM

ESTRATÉGIAS ALGORÍTMICAS EXATAS E HÍBRIDAS PARA PROBLEMAS DE ESCALONAMENTO EM MÁQUINAS PARALELAS COM PENALIDADES DE ANTECIPAÇÃO E ATRASO

Tese de Doutorado apresentada ao Programa de Pós-graduação em Informática do Instituto de Computação da Universidade Federal do Amazonas (PPGI/IComp, UFAM) como parte dos requisitos necessários à obtenção do título de Doutor em Informática .

Orientadora: Rosiane de Freitas Rodrigues, D.Sc.

Outubro de 2017

Manaus - AM

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Amorim, Rainer Xavier de
A524e Estratégias algorítmicas exatas e híbridas para problemas de escalonamento em máquinas paralelas com penalidades de antecipação e atraso / Rainer Xavier de Amorim. 2017
113 f.: il. color; 31 cm.

Orientadora: Rosiane de Freitas Rodrigues
Tese (Doutorado em Informática) - Universidade Federal do Amazonas.

1. algoritmos. 2. escalonamento Just-in-Time. 3. penalidad antecipação e atraso. 4. máquinas paralelas idênticas. 5. programação inteira. I. Rodrigues, Rosiane de Freitas II. Universidade Federal do Amazonas III. Título

FOLHA DE APROVAÇÃO

"Estratégia Algorítmica Híbrida Exata-Heurística para Problemas de Escalonamento em Máquinas Paralelas com Penalidades de Antecipação e Atraso"

RAINER XAVIER DE AMORIM

Tese de Doutorado defendida e aprovada pela banca examinadora constituída pelos Professores:

Profa. Rosiane de Freitas Rodrigues - PRESIDENTE

Prof. Raimundo da Silva Barreto - MEMBRO INTERNO

Profa. Eulanda Miranda dos Santos - MEMBRO INTERNO

Prof. Eduardo Uchoa Barboza - MEMBRO EXTERNO

Prof. José Luiz de Souza Pio - MEMBRO EXTERNO

Manaus, 06 de Outubro de 2017

A Deus,
aos meus professores,
aos meus colegas de curso e
aos meus familiares com muito amor,
dedico esta tese.

Agradecimentos

Agradeço a Deus pelo dom da vida, pelo poder do livre arbítrio, por estar sempre ao meu lado em todos os momentos da minha vida e por me guiar sempre por bons caminhos onde eu pudesse atingir todos os meus objetivos.

Aos meus pais, Mazionete Xavier de Amorim e Célio Figueiredo de Amorim, a minha irmã Kássia Cristina Xavier de Amorim e a minha esposa e companheira de todas as horas Lídia Mayara Almeida de Menezes pelo apoio e constante estímulo juntamente com os meus sogros Antônio José Magalhães da Silva e Angelita Souza Almeida.

A minha orientadora, Rosiane de Freitas Rodrigues, pela oportunidade de enriquecimento profissional, dedicação, lições e pela competência com que sempre orientou esta pesquisa.

Aos meus amigos e parceiros do grupo de pesquisa de Otimização, Algoritmos e Complexidade Computacional - ALGOX, que me ajudaram neste trabalho e aos professores e membros da banca examinadora com quem tive a oportunidade de estudar e que deram valiosas sugestões que contribuíram para o enriquecimento desta tese.

A Fundação de Amparo à Pesquisa do Estado do Amazonas (FAPEAM) por todo o suporte financeiro nestes quatro anos de pesquisa.

"Porque vivemos por fé, e não pelo que vemos"
2 Coríntios 5:7.

Resumo da Tese apresentada ao PPGI/IComp/UFAM como parte dos requisitos necessários para a obtenção do grau de Doutor em Informática.

ESTRATÉGIAS ALGORÍTMICAS EXATAS E HÍBRIDAS PARA PROBLEMAS DE ESCALONAMENTO EM MÁQUINAS PARALELAS COM PENALIDADES DE ANTECIPAÇÃO E ATRASO

RAINER XAVIER DE AMORIM

Outubro/2017

Orientadora: Profa. Rosiane de Freitas Rodrigues, D.Sc.

Esta pesquisa investiga problemas de escalonamento com penalidades de antecipação e atraso em ambiente mono e multiprocessado envolvendo máquinas paralelas. Este problema é também conhecido na literatura como escalonamento *Just-in-Time*, sistema amplamente utilizado em indústrias para reduzir estoques e os custos decorrentes, a fim de que o produto seja produzido de acordo com a demanda. Neste trabalho é proposta uma estratégia algorítmica híbrida exato-heurística, baseada em uma formulação de programação inteira *arc-time* e um algoritmo evolucionário fortemente baseado em busca local, para melhor resolver problemas clássicos de escalonamento em máquinas paralelas envolvendo penalidades de antecipação e atraso, com tarefas independentes e tempos de processamento arbitrários. Os arcos são selecionados das soluções ótimas locais obtidas pelo algoritmo genético fortemente baseado em busca local (GLS) com movimentos generalizados de troca de pares, que são fornecidos como entrada para a formulação arc-time, para gerar soluções melhores do que as obtidas por ambos os métodos quando utilizados isoladamente. Os experimentos computacionais apresentam resultados competitivos em relação à literatura. O método proposto também resolve instâncias de tamanho maior de até 500 tarefas em máquinas paralelas idênticas.

Palavras-chave: algoritmos, escalonamento *Just-in-Time*, penalidades de antecipação e atraso, máquinas paralelas idênticas, programação inteira.

Abstract of Thesis presented to PPGI/IComp/UFAM as a partial fulfillment of the requirements for the degree of Doctor in Informatics.

EXACT AND HIBRID ALGORITHMIC STRATEGIES FOR SCHEDULING PROBLEMS ON PARALLEL MACHINES WITH EARLINESS AND TARDINESS PENALTIES

RAINER XAVIER DE AMORIM

October/2017

Advisor: Prof. Rosiane de Freitas Rodrigues, D.Sc.

This research investigates scheduling problems with earliness and tardiness penalties on single and parallel machine environments. This problem is also known in the literature as Just-in-Time scheduling, system widely used in industries to reduce inventories and costs, in order to lead product to be produced according to demand. In this work we present a hybrid exact-heuristic algorithmic strategy, based on an arc-time indexed integer programming formulation and a generalized evolutionary heuristic based on a strong local search, to better solve classical parallel machine scheduling problems involving weighted earliness-tardiness penalties, with independent jobs and arbitrary processing times. Selected arcs from local optima solutions generated by a genetic algorithm based on a strong local search (GLS) with generalized pairwise interchanges are given as input to the arc-time formulation, to produce better solutions than those obtained by both methods when used isolated. Computational experiments present competitive results according to the literature. Our proposed method also solves large instances up to 500 jobs in identical parallel machines.

Keywords: algorithms, Just-in-Time scheduling, earliness and tardiness penalties, identical parallel machines, integer programming.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Neste capítulo serão apresentados o contexto e a descrição do problema, o que motivou esta pesquisa e as questões de investigação. São também apresentados os objetivos e a organização deste texto.

1.1 Contexto e descrição do problema

Os problemas de escalonamento em geral vêm sendo estudados desde a década de 1950 (DEFREITAS, 2009; BRUCKER, 2006), quando o ferramental matemático pôde ser usado para modelar problemas reais complexos com o auxílio do computador e questões econômicas motivaram cada vez mais o seu estudo em diversas áreas do conhecimento. Por isso, existe na literatura um vasto material sobre o assunto com inúmeras classes de complexidade com diferentes características, condições de execução, restrições e ambientes de processamento. Apesar do grande interesse de diversas áreas na investigação destes problemas, ainda existem muitos problemas em aberto, com pequenas diferenças em relação aos problemas já bem resolvidos ou já provados serem NP-Difíceis. Além disso, constantemente surgem novos problemas, conceitos, formas de modelagem, técnicas de resolução e aplicação, o que enriquece ainda mais a literatura e desperta cada vez mais o interesse em buscar novas e melhores soluções algorítmicas e novos modelos matemáticos cada vez mais eficientes para os problemas de escalonamento do mundo real (DEFREITAS, 2009).

Dentre os problemas de escalonamento existentes, os que consideram datas de término na execução de tarefas vêm sendo estudados desde a década de 1970 e constituem

uma das principais vertentes de pesquisa sobre a classe geral de problemas de escalonamento (EILON; CHOWDHURY, 1977; GORDON; STRUSEVICH, 1999; SOURD; KEDAD-SIDHOUM, 2008; JóZEFOWSKA, 2007; PINEDO, 2012; TANAKA, 2012b; AMORIM, 2013; KRAMER, 2015), tanto pela grande representatividade de situações reais, quanto pela maior complexidade teórica envolvida. Especialmente, o estudo de problemas de escalonamento que consideram simultaneamente penalidades de antecipação e atraso na execução de tarefas foi motivado pela adoção nas indústrias de processos produtivos com o conceito de produção sem folga (nem antes nem depois da data de término estipulada).

Esta ideia de produção sem folga ficou conhecida como *Just-in-Time* (JIT), tal termo surgiu no Japão na década de 70, na indústria automotiva, onde se buscava um sistema em que fosse possível coordenar uma determinada produção com a sua demanda específica e com o menor tempo de atraso possível. Este sistema tem como objetivo a melhoria do processo produtivo e fluxo de manufatura, eliminação de estoques e desperdícios. A aplicação deste sistema é muito ampla, envolvendo a produção de produtos perecíveis por exemplo, mas abrangendo qualquer sistema em que as tarefas devem ser finalizadas o mais próximo possível da data de término estipulada (LEUNG; KELLY; ANDERSON, 2004).

O conceito JIT também pode ser utilizado para gerir as atividades externas de uma empresa, como o processo de compra de insumos de outras fábricas, fabricação e montagem de novos produtos a partir desses insumos e distribuição ou exportação dos itens produzidos. JIT auxilia em todo esse processo, pois parte da ideia de "conduzir"a produção a partir de uma determinada demanda, ou seja, a fábrica produz somente o necessário e nas quantidades necessárias quando requerido, evitando assim a quebra da cadeia de suprimentos da empresa e o acúmulo de itens produzidos no estoque, pois esses produtos são logo enviados ao mercado consumidor.

Muitas vezes é bastante difícil atingir esses requisitos de produção, logo, a redução de custos de estoque e atraso na produção acabam se tornando grandes desafios. Se a demanda dos consumidores é conhecida no início do planejamento, é natural definir as datas de término dos produtos obedecendo a demanda de cada consumidor. Assim, a partir do cronograma principal de produção, onde as datas de término sugeridas dos consumidores estão definidas, as informações das demandas são propagadas para os níveis restantes de

produção. O objetivo é estipular as datas de término de cada etapa de produção e controle de matérias-primas, procurando assim atender ao cronograma principal de produção (JóZEFOWSKA, 2007).

Naturalmente, completar uma produção após a data de entrega não é algo desejável. No caso de produtos finalizados, o atraso não lida apenas com a insatisfação do cliente mas também pode resultar na perda de clientes. Em geral, os custos incorridos pela conclusão tardia de um pedido incluem penalizações devido a quebra de cláusulas contratuais, custos de expedição de tarefas que estão com atraso e custos de atualização do cronograma de produção. Por outro lado, quando uma produção é concluída antes da data de entrega, essa produção deve ser armazenada, o que pode gerar custos indesejáveis de estoque (JóZEFOWSKA, 2007).

O problema é formalizado da seguinte forma: seja $J = \{1, ..., n\}$ um conjunto de tarefas que podem ser processadas em um ambiente monoprocessado ou em um conjunto de máquinas paralelas idênticas $M = \{1, ..., m\}$, sem preempção. Cada tarefa deve ser processada em uma das máquinas, sendo que todas as tarefas devem ser processadas. Cada tarefa j possui um tempo de processamento positivo (*processing time*) p_j, um prazo de termino sugerido (*due date*) d_j e um peso positivo (*weight*) w_j (DEFREITAS, 2009). A antecipação de uma tarefa j é definida como $E_j = \max\{0, d_j - C_j\}$, por outro lado, o atraso de uma tarefa j é definido como $T_j = \max\{0, C_j - d_j\}$, onde C_j é o tempo de completude de cada tarefa (BRUCKER, 2006). O objetivo é minimizar $\sum_{j=1}^{n} \alpha_j E_j + \sum_{j=1}^{n} \beta_j T_j$, onde α_j é a penalidade aplicada a antecipação e β_j é a penalidade de atraso.

1.2 Motivação e Justificativa

Este trabalho justifica-se pela oportunidade de lidar com problemas de escalonamento de tarefas envolvendo a minimização das penalidades de antecipação e atraso através de um enfoque teórico-computacional. Sendo que, muitos problemas reais podem ser modelados como problemas de escalonamento e a literatura oferece várias ferramentas e técnicas de resolução para tais problemas.

Do ponto de vista teórico, a motivação encontra-se na dificuldade computacional de resolver tal problema de otimização. Isto se deve ao fato do problema de escalonamento em questão ser NP-Difícil, pois faz parte de uma generalização do problema $1||\sum T_j$ (na

notação de 3 campos de Graham et al. (1979), provado ser NP-Difícil por Du e Leung (1990). Existem muitos esforços na literatura para solucionar o problema de forma exata, até então realizados somente para instâncias pequenas para o problema.

Uma aplicação do problema no mundo real se dá nas organizações em geral, onde é importante que estratégias de alocação de recursos sejam bem definidas, como postos e máquinas da linha de produção de uma fábrica, organização do quadro de funcionários de uma empresa, fluxo de matérias-primas, de modo a minimizar os encargos de atraso ou antecipação nas atividades a fim de minimizar custos em geral e maximizar o lucro e a satisfação dos clientes.

Assim, tem-se uma área bastante desafiadora e promissora de pesquisa. Este trabalho busca soluções para instâncias de tamanho maior para os problemas de escalonamento que envolvem penalidades de antecipação e atraso em máquinas paralelas idênticas, ainda não existentes na literatura. Os resultados apresentados até então são somente para o ambiente monoprocessado ou de uma máquina de Tanaka (2012b) para até 300 tarefas.

O método híbrido exato-heurístico desenvolvido é capaz de lidar com o problema tanto para os ambientes mono e multiprocessado. O método heurístico apresentado é uma extensão do trabalho de mestrado de Amorim (2013), que se trata de um algoritmo evolucionário fortemente baseado em busca local e melhorado tanto no sentido de fornecer bons resultados quanto em se adaptar a problemas com funções não-regulares. Os resultados mostram que a estratégia híbrida proposta é capaz de obter soluções de alta qualidade quando comparadas com as melhores disponíveis na literatura.

1.3 Hipótese de Pesquisa

A seguinte hipótese motivou a presente pesquisa:

a. É possível resolver de maneira mais eficaz e eficiente o problema da minimização das penalidades de antecipação e atraso na execução das tarefas (JIT_Sched) usando estratégias exatas e heurísticas combinadas?

Assume-se eficaz no sentido se obter soluções ótimas ou melhor limites superiores. E eficiente é no sentido de resolver instâncias bem maiores e/ou com menores tempos de execução.

1.4 Objetivo

O principal objetivo desta tese consiste em elaborar uma estratégia algorítmica exata-heurística competitiva com o estado-da-arte, para a resolução do problema de escalonamento de tarefas independentes e ponderadas em máquinas paralelas idênticas, visando a minimização de penalidades de antecipação e atraso.

Sendo assim, como objetivos específicos a serem atingidos:

a. Analisar e adaptar formulações matemáticas de programação linear inteira e inteira mista existentes para o problema JIT_Sched.

b. Elaborar um método metaheurístico adaptativo fortemente baseado em busca local.

c. Resolver grandes instâncias do problema JIT_Sched, gerando ótimos ou melhores limites superiores.

1.5 Metodologia de pesquisa

A metodologia utilizada nesta pesquisa consistiu nos seguintes passos:

1. Revisão da literatura relacionada a problemas de escalonamento com penalidades de antecipação e atraso, com ênfase em formulações matemáticas e estratégias algorítmicas.

2. Proposição e desenvolvimento de algoritmos e métodos para os problemas pesquisados.

3. Realização de experimentos computacionais para avaliação dos algoritmos e comparação com resultados da literatura.

4. Publicação de artigos sobre resultados obtidos na pesquisa e participações em eventos.

5. Elaboração da tese e defesa da mesma.

1.6 Organização do documento

Com relação a organização, esta tese de doutorado encontra-se dividida em 06 (seis) capítulos. Neste Capítulo é apresentada a introdução desta tese, juntamente com a justificativa, objetivo geral e específicos, metodologia e organização da tese. No Capítulo 2 é apresentada a revisão da literatura realizada, onde é feita uma sumarização das principais publicações sobre o tema, juntamente com as suas respectivas estratégias de resolução, notações e definições. No Capítulo 3 são apresentas as formulações matemáticas de programação inteira e mista para o escalonamento E/T. No Capítulo 4 é detalhada a estratégia híbrida proposta nesta pesquisa. No Capítulo 5 são apresentados os resultados computacionais e uma análise empírica realizada sobre a estratégia proposta. E por fim, no Capítulo 6, são expostas as considerações finais do trabalho.

Capítulo 2

Escalonamento com penalidades de antecipação e atraso de tarefas

Neste capítulo são apresentados os trabalhos levantados na revisão da literatura realizada, com os artigos, livros e trabalhos de maior relevância sobre a classe de problemas de principal interesse deste trabalho.

2.1 Teoria de escalonamento

Nos problemas de escalonamento considerados neste trabalho, o número de tarefas e de máquinas é finito. O número de tarefas é denotado por n e o número de máquinas é denotado por m. Geralmente, j refere-se a uma tarefa enquanto que i refere-se a uma determinada máquina. Se uma tarefa requer um determinado tipo de processamento ou operação, então o par (i, j) refere-se ao processamento ou operação de uma tarefa j na máquina i (BRUCKER, 2006; PINEDO, 2012). Sejam m as máquinas $P_j(j = 1, ..., m)$ que devem processar n tarefas $J_i(i = 1, ..., n)$, um escalonamento é para cada tarefa uma alocação de um ou mais intervalos a uma ou mais máquinas. Esta seção baseia-se nas seguintes referências: deFreitas (2009), Brucker (2006) e Pinedo (2012).

Graham et al. (1979) introduziram um esquema de classificação para problemas de escalonamento em termos da notação de três campos $\alpha|\beta|\gamma$ onde α especifica o **ambiente de processamento** que possui somente uma entrada, β especifica os detalhes das **características das tarefas** e restrições de escalonamento, pode ter muitas ou nenhuma entrada e γ especifica a **função objetivo (FO)** a ser otimizada (*critério de otimização*), geralmente

tendo somente uma entrada. A notação de três campos é detalhada nas seções 2.1.1, 2.1.2 e 2.1.3.

2.1.1 Ambiente de processamento (α)

O primeiro campo α, referente às características das máquinas, pode ser representado pelas seguintes notações:

- **Ambiente monoprocessado:** quando há um único processador no sistema. A notação utilizada é 1;
- **Processadores ou máquinas paralelas idênticas:** quando existem m máquinas idênticas em paralelo. Utiliza-se a notação P_m, se o número de processadores é fixo (m é uma constante), e utiliza-se a notação P, se o número de processadores é parte da entrada (m é uma variável);
- **Processadores ou máquinas paralelas uniformes:** quando existem m máquinas paralelas com velocidades diferentes. A notação utilizada neste caso é Q e Q_m;
- **Processadores ou máquinas paralelas não-relacionadas:** quando existem m máquinas em paralelo com desempenhos dependentes da tarefa a ser executada. A notação utilizada neste caso é R e R_m;
- **Processadores ou máquinas de propósito geral:** quando as m máquinas são divididas em subconjuntos μ, a notação desses subconjuntos é $PMPM$ se as máquinas forem idênticas. Se as máquinas forem paralelas uniformes, a notação é $QMPM$. E, se as máquinas paralelas forem não-relacionadas, a notação é $RMPM$. A notação utilizada para os processadores de propósito geral é MPM;
- ***Flow shop*:** quando existe a mesma sequência de máquinas para cada tarefa. Dessa forma, considera-se m máquinas em série, cada tarefa deve ser processada em cada uma dessas m máquinas. Todas as tarefas devem seguir a mesma configuração de processamento, ou seja, elas devem ser processadas primeiramente na máquina 1, depois na máquina 2, e assim sucessivamente. Depois de finalizar a sua execução em uma máquina, a tarefa entra na fila para ser executada na próxima máquina. Geralmente, todas as filas são organizadas por ordem de chegada (*First In First Out - FIFO*). A notação utilizada é F e F_m;

- ***Job shop*:** quando existem máquinas diferentes e em ordem diferente em cada tarefa. A notação utilizada é J e J_m;

- ***Open shop*:** quando a tarefa é executada em cada máquina exatamente uma vez em cada sequência, esta sequência é diferente para cada tarefa. A notação utilizada é O e O_m.

2.1.2 Características das tarefas (β)

O campo β é utilizado para detalhar algumas características referentes ao processamento das tarefas. Este campo pode ser vazio, conter uma, ou múltiplas entradas. Cada **tarefa (do inglês, *job* ou *task*)** pode ter muitas restrições associadas, como as seguintes:

- **Data de chegada ou de disponibilidade (*release date*):** define o momento em que a tarefa pode começar a ser executada. A notação utilizada é r_j;

- **Data de término sugerida (*due date*):** indica o tempo que a tarefa deve ser finalizada, notação d_j, sob pena de sofrer alguma penalidade caso ultrapasse tal tempo;

- **Data de término obrigatória (*deadline*):** indica o tempo que cada tarefa deve ser finalizada, notação D_j, não sendo permitido ultrapassar tal tempo;

- **Tempos de processamento (*processing time*):** restringe o tempo de processamento de cada tarefa, notação p_j, por exemplo, quando $p_j = p$, significa que todas as tarefas possuem o mesmo tempo de processamento, igual a p, quando $p_j = 1$, significa que todas as tarefas têm o mesmo tempo de processamento igual a 1, mas quando a notação p_j é omitida da descrição do problema, as tarefas podem possuir tempos de processamento quaisquer;

- **Peso (*weight*):** que pode representar prioridades iguais ou diferentes. A notação utilizada é w_j;

- **Preempção (*preemption*):** indica se uma preempção (interrompe e retoma sua execução) é permitida ou não. A notação utilizada é *pmtn*;

- **Sem-espera (*no-wait*):** indica que a tarefa não pode ficar em estado de espera depois de iniciada sua execução. Somente para *flow shops*. A notação utilizada é *nwt*;

- **Restrição no número de tarefas:** restringe o número de tarefas, ou seja, $nbr_j = 5$ indica que existem no máximo 5 tarefas a serem processadas. A notação utilizada é *nbr*;

- **Restrição de precedência (*precedence constraints*):** essas restrições podem aparecer nos ambientes mono e multi-processado, sendo que uma ou mais tarefas devem ser completadas antes que outra tarefa possa começar o seu processamento. Existem casos especiais de restrição de precedência: se cada tarefa tem pelo menos um predecessor e pelo menos um sucessor, as restrições são definidas como *chains*. Se cada tarefa tem pelo menos um sucessor, as restrições são definidas como *intree*. Se cada tarefa tem pelo menos um predecessor as restrições são definidas como *outtree*. A notação utilizada é *prec*;

- **Sequência dependente de tempos de preparação (*sequence dependent setup times*):** onde s_{jk} representa uma sequência dependente de tempos de preparação das tarefas j e k, S_{0k} denota o tempo de preparação para a tarefa k se a tarefa k é a primeira na sequência e s_{j0} se a tarefa j é a última da sequência. Se o tempo de preparação entre as tarefas j e k depende da máquina, então o índice i é incluído, ou seja, s_{ijk}. Se s_{jk} não aparecer no problema, ou seja, se s_{jk} não aparecer no campo β, todos os tempos de preparação possuem o valor 0. A notação utilizada é s_{jk};

- **Processamento em lotes (*batch processing*):** onde uma máquina deve ser capaz de processar um número de tarefas, b, simultaneamente, ou seja, pode processar um lote de b tarefas ao mesmo tempo. Os tempos de processamento de todas as tarefas no lote não devem ser os mesmos e o lote todo é finalizado somente quando a última tarefa do lote é completada, implicando que o tempo de processamento de todo o lote é determinado pela tarefa com o maior tempo de processamento. Se $b = 1$, então o problema é reduzido ao problema convencional de escalonamento. Outro caso especial é o $b = \infty$, ou seja, não há limite no número de tarefas que a máquina pode processar ao mesmo tempo.

  Utilizam-se as notações $p-batch$, quando o comprimento do lote for igual ao maior dos tempos de processamento dentre todas as tarefas, e a notação $s-batch$, quando o comprimento do lote for igual a soma dos tempos de processamento de todas as tarefas.

2.1.3 Critério de otimização (γ)

O terceiro campo γ é destinado à função objetivo do problema. A seguir serão apresentados exemplos de funções regulares que consideram escalas de tempo, conhecidas como **critérios proporcionais**, que são (DEFREITAS, 2009):

- **Maior tempo de completude (*makespan*)** (Notação: C_{max}) refere-se ao tempo de completude da última tarefa a terminar sua execução, ou de outra forma, maior tempo de completude dentre todas as tarefas executadas. $C_{max} = \max\{C_1, ..., C_m\}$. Deseja-se minimizar C_{max}, pois um *makespan* mínimo geralmente significa uma maior utilização da máquina;

- **Máxima latência (*maximum lateness*)** (Notação: L_{max}) é definida como $L_j = C_j - d_j$, que será um valor positivo se a tarefa J_j for executada com atraso, será um valor negativo se a tarefa for antecipada ou, senão, será um valor nulo. A função de máxima latência (L_{max}), então, retorna o valor do maior atraso possível ou menor antecipação se não houver atrasos, considerando todas as tarefas executadas. $L_{max} = \max\{L_1, ..., L_n\}$, onde deseja-se minimizar o L_{max};

- **Tempo total de completude (*total completion time*)** (Notação: $\sum C_j$ ou $\sum w_j C_j$) soma de todos os tempos de completude das tarefas (ponderadas ou não), onde deseja-se minimizar $\sum C_j$ ou $\sum w_j C_j$;

- **Tempo total de atraso (*tardiness*) (Notação: $\sum T_j$ ou $\sum w_j T_j$)** trata-se da soma de todos os tempos de atraso das tarefas (ponderadas ou não). Dessa forma, o atraso assume um valor positivo se as tarefas forem atrasadas, caso contrário, o atraso assume o valor 0. Sendo $T_j = \max\{L_j, 0\}$, onde deseja-se minimizar $\sum T_j$ ou $\sum w_j T_j$.

Também existem funções objetivo que não dependem diretamente do tempo, sendo conhecidos como **critérios permanentes**, que são:

- **Número de tarefas tardias** (Notação: $\sum U_j$ ou $\sum w_j U_j$) é o número total de tarefas executadas com atraso (ponderadas ou não), onde U_j é uma penalidade unitária caso

a tarefa esteja atrasada (e nula caso contrário), como no exemplo a seguir:

$$U_j = \begin{cases} 1 \text{ se } C_j > d_j \\ \\ 0 \text{ caso contrário} \end{cases} \tag{2.1}$$

A função objetivo que envolve a minimização de atraso pertence a classe de funções objetivo que possui medidas **regulares** de desempenho. Uma função objetivo possui uma medida de desempenho regular quando é não-decrescente em $C_1, ..., C_n$. Pesquisas recentes consideram cada vez mais o estudo de funções objetivo que são **não-regulares**, como por exemplo, a que representa penalidades de antecipação (ponderadas ou não). A antecipação assume um valor positivo se as tarefas forem antecipadas, caso contrário, o valor de antecipação assume o valor 0. Desta forma, a penalidade de antecipação é não-crescente em C_j e pode ser formalmente descrita como segue:

- **Tempo total de antecipação (*earliness*) (Notação: $\sum E_j$ ou $\sum w_j E_j$)** trata-se da soma de todos os tempos de antecipação das tarefas (ponderadas ou não). Desta forma, a antecipação assume um valor positivo se as tarefas forem antecipadas, caso contrário, o valor de antecipação é valor 0. Sendo $E_j = \max\{d_j - C_j, 0\}$, onde deseja-se minimizar $\sum E_j$ ou $\sum w_j E_j$.

Para atender ao conceito surgido nas indústrias de produção sem folga, ou seja, de produto produzido o mais próximo possível da data de entrega (do inglês *Just-in-Time* - JIT), aplica-se uma combinação dos critérios de antecipação e atraso anteriormente descritos, e, desta forma, a função objetivo consiste na soma das tarefas escalonadas com antecipação e atraso, ponderadas ou não, tal como segue:

$$\sum_{j=1}^{n} \alpha_j E_j + \sum_{j=1}^{n} \beta_j T_j. \tag{2.2}$$

Observa-se que na formulação acima, as tarefas são ponderadas e o peso associado à antecipação da tarefa j (α_j) difere do peso associado ao atraso da tarefa j (β_j). Se os pesos forem iguais para ambas penalidades considerando a mesma tarefa, então, basta isolar a constante do somatório.

Na Figura 2.1 são apresentadas as reduções elementares para as funções objetivo,

incluindo as funções regulares e não-regulares para os problemas de escalonamento determinísticos.

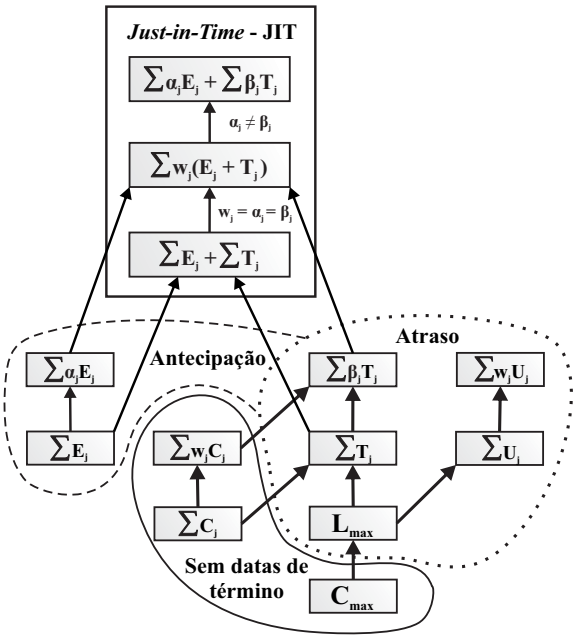


Figura 2.1 Hierarquias de complexidade dos problemas de escalonamento determinísticos com as funções objetivo clássicas incluindo os problemas de escalonamento de antecipação e atraso ou *Just-in-Time* (JIT).

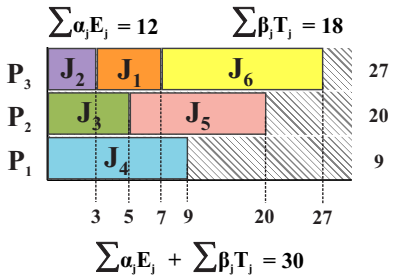
Um exemplo teórico para o escalonamento com penalidades de antecipação e atraso em ambiente de máquinas paralelas idênticas é apresentado na Figura 2.2 onde, na Figura 2.2 (a) é apresentado um exemplo de instância com 6 tarefas para o problema com seus respectivos valores para tempos de processamento (p_j), data de término sugerida (d_j), penalidades de antecipação (α_j), penalidades de atraso (β_j), tempo de completude (C_j) das tarefas, valores de antecipação (E_j) e atraso (T_j) das tarefas, e os valores da antecipação ponderada ($\alpha_j E_j$) e do atraso ponderado das tarefas ($\beta_j T_j$) e na Figura 2.2 (b) pode-se observar como fica a representação do escalonamento em três máquinas para essa instância.

O valor de antecipação é obtido pela diferença entre data de término sugerida e o tempo de completude, e o valor de atraso é obtido através de diferença entre o tempo de completude da tarefa e a data de término sugerida, caso o resultado de uma dessas diferenças seja negativo, é atribuído o valor 0 no lugar do resultado negativo. Ao final, o peso ou penalidade de cada tarefa é multiplicado com o seu respectivo valor de antecipação (E_j) ou atraso (T_j) e, depois esses resultados são somados obtendo assim o valor total de

antecipação ponderada e de atraso ponderado das tarefas.

J_j	p_j	d_j	α_j	β_j	C_j	E_j	T_j	$\alpha_j E_j$	$\beta_j T_j$
J_1	4	7	3	5	7	0	0	0	0
J_2	3	4	4	5	3	1	0	4	0
J_3	5	5	8	8	5	0	0	0	0
J_4	9	10	8	10	9	1	0	8	0
J_5	15	17	6	4	20	0	3	0	12
J_6	20	25	7	3	27	0	2	0	6
						$\sum\alpha_j E_j$ e $\sum\beta_j T_j$ =		12	18

(a)



(b)

Figura 2.2 (a) Exemplo de uma instância de 6 tarefas para o problema de escalonamento com antecipação e atraso (b) representação do escalonamento através do gráfico de Gantt orientado-a-máquina.

2.2 Literatura sobre problemas de escalonamento E/T

Os problemas de escalonamento com antecipação e atraso (escalonamento E/T) representam situações importantes do mundo real, principalmente com o surgimento motivado pela adoção nas indústrias de processos produtivos com o conceito de produção sem folga (nem antes nem depois da data de término sugerida), ou seja, de produtos com produção concluída no momento em que devem ser entregues (do inglês, *Just-in-Time* (JIT)) (LEUNG; KELLY; ANDERSON, 2004).

De acordo com Xiao e Li (2002), decisões na determinação de datas de término sugeridas de tarefas são importantes no planejamento e controle de muitas operações de uma organização. Uma cotação rápida de entrega de um determinado trabalho pode ser muito atrativa para o cliente, mas pode aumentar a chance de entregas atrasadas. Por outro lado, a demora na cotação de um trabalho pode diminuir a chance de um trabalho ser entregue com atraso, mas será penalizado com o risco de se perder um determinado negócio em potencial. A Tabela 2.1 apresenta uma classificação dos problemas de escalonamento de tarefas com antecipação e atraso investigados nesta seção.

Na literatura, os problemas de escalonamento que envolvem penalidades de antecipação e atraso são classificados em duas categorias principais, segundo os tipos de datas de término sugeridas, de acordo como abordado em Hassin e Shani 2005:

- **Problemas de escalonamento com datas de término iguais (*common due dates*)**, $d_j = d$, denotados como CDD. Estes problemas são divididos em **datas de término**

restritas e **datas de término irrestritas**. O problema de escalonamento é restrito, quando nenhuma tarefa pode inicializar seu processamento antes do instante zero e o due date d é justo. O problema de escalonamento é irrestrito, quando a data de término sugerida d é suficientemente grande, por exemplo $d \geq \sum_{j=1}^{n} p_j$. Uma solução ótima para problemas de escalonamento com datas de término iguais deve satisfazer as seguintes propriedades (BAKER; SCUDDER, 1990; HASSIN; SHANI, 2005):

1. não há tempo ocioso inserido (se a tarefa j segue imediatamente a tarefa i no escalonamento, $C_j = C_i + p_j$);
2. a sequência deve possuir uma função de crescimento em **forma de V** (*V-shaped*), ou seja, tarefas antecipadas ($C_j \leq d$) quando a solução possui uma sequência de tarefas em ordem não-crescente (em relação ao tempo de completude das tarefas) e tarefas atrasadas ($C_j > d$) quando a solução possui uma sequência de tarefas em ordem não-decrescente (em relação ao tempo de completude das tarefas), não permitindo tempo ocioso entre as tarefas;
3. a $b-$ésima tarefa da sequência é concluída precisamente em sua data de término sugerida, onde b é o menor valor inteiro que satisfaz a inequação $\sum_{j=1}^{b} \alpha_j \geq \sum_{i=b+1}^{n} \beta_j$ ($C_j = d$ para alguma tarefa j).

Revisões da literatura e uma classificação dos problemas de escalonamento com penalidades de antecipação e atraso com datas de término iguais podem ser consultados em Baker e Scudder (1990), Hall e Posner (1991), Hall, Kubiak e Sethi (1991) e Gordon, Proth e Chu (2002).

- **Problemas de escalonamento com datas de término distintas (*distinct due dates*)**, d_j - denotados como DDD e cujas três propriedades acima descritas, que definem uma solução ótima para problemas de escalonamento com datas de término iguais, não necessariamente podem ser utilizadas para problemas de escalonamento com datas de término distintas. Baker e Scudder (1990) afirmam que nenhuma das propriedades descritas para datas de término iguais pode ser aplicada a problemas com datas de término distintas, pois **um escalonamento ótimo para datas de término distintas pode requerer tempo ocioso entre as tarefas**. Logo, a primeira propriedade é violada e a segunda também, uma vez que tais propriedades não permitem tempo ocioso ou a utilização de datas de término distintas. A terceira

propriedade não se aplica, uma vez que múltiplas datas de término não podem ser acomodadas de modo a terminarem todas na mesma data de término sugerida.

Para resolver isso, James e Buchanan (1997) redefiniram tais propriedades para os problemas de escalonamento com datas de término distintas. Eles definiram um bloco como sendo um conjunto máximo de tarefas que são escalonados sem tempo ocioso, onde qualquer solução ótima para o problema satisfaz as três propriedades descritas acima com modificações simples, no que diz respeito a blocos de tarefas. Uma abordagem heurística híbrida para o problema, sem tempo ocioso, envolvendo algoritmo genético, busca local e recozimento simulado pode ser consultado em M'Hallah (2007). Kanet e Sridharan (2000) apresentaram uma revisão de problemas com tempos ociosos inseridos (IIT), onde é definida uma taxonomia dos problemas com IIT.

Existem casos especiais de problemas irrestritos com penalidades de antecipação e atraso unitários, ou seja, $\alpha = \beta = 1$, para $1 \leq j \leq n$, onde deseja-se minimizar $\sum_{j=1}^{n}(E_j + T_j)$, dessa forma, o problema de escalonamento que envolve antecipação e atraso ponderados de tarefas é uma generalização desse problema, onde $\alpha_j = \alpha$ e $\beta_j = \beta$ para $1 \leq j \leq n$ e deseja-se minimizar $\sum_{j=1}^{n}(\alpha_j E_j + \beta_j T_j)$. Verma e Dessouky (1998) consideraram o problema em ambiente monoprocessado considerando tempos de processamento unitários e casos de pesos de antecipação e atraso unitários, onde $\alpha_i = \beta_i = 1$ $\forall i \in J$, pesos de antecipação e atraso idênticos, onde a taxa α_i/β_i ou a diferença $\alpha_i - \beta_i$ é o mesmo para todas as tarefas, penalidades idênticas de antecipação, $\alpha_i = \alpha$ $\forall i \in J$ e penalidades idênticas de atraso, $\beta_i = \beta$ $\forall i \in J$.

Shabtay e Steiner (2012) apresentam uma revisão da literatura sobre problemas de escalonamento JIT, onde é apresentado o problema de maximização do número ponderado de tarefas que são finalizadas exatamente em suas datas de término sugeridas e também são apresentados vários ambientes de escalonamento com tempos de processamento fixos e variáveis. Mais revisões da literatura sobre o tema, envolvendo os ambientes mono e multiprocessado, casos especiais, provas e teoremas podem ser consultados nos trabalhos de Hall, Kubiak e Sethi (1991), Cheng e Chen (1994), Tuong e Soukhal (2010), Chung e Choi (2013), Wan e Yuan (2013), Zhao et al. (2014), Janiak et al. (2015), Awasthi et al. (2015) e Shabtay (2016).

Nas próximas sub-seções serão apresentadas as principais abordagens da literatura so-

bre problemas de escalonamento de tarefas com penalidades de antecipação e atraso, com tempo ocioso e não-ocioso, juntamente com trabalhos relacionados que envolvem ambientes monoprocessados e máquinas paralelas idênticas, uniformes e não-relacionadas. Também é apresentada uma aplicação na indústria automobilística localizada no Polo Industrial de Manaus.

2.2.1 Aplicações do escalonamento *Just-in-Time*

De acordo com Józefowska (2007), a filosofia *Just-in-Time* para processos produtivos foi inicialmente proposta pelos japoneses em 1950. Esta filosofia pode ser brevemente definida como a eliminação de desperdícios e contínua melhoria da produção. Existem muitas fontes diferentes de desperdício em um sistema de manufatura. Por isso, muitas atividades necessitam ser realizadas na fábrica a fim de efetivamente implementar a filosofia *Just-in-Time*. Tempo de espera, superprodução e inventário são as fontes de desperdícios que podem ser eliminadas pelo planejamento e escalonamento de produção apropriado. Esta seção baseia-se nos trabalhos de Józefowska (2007) e Moraes e Nogueira (2014).

Os objetivos do escalonamento *Just-in-Time* diferem dos objetivos considerados no escalonamento de produção tradicional. Por isso, os novos problemas de escalonamento têm sido definidos na teoria de escalonamento para atender as necessidades de soluções práticas. Dois objetivos de otimização são considerados no contexto do escalonamento *Just-in-Time*. O primeiro é a minimização de variação de produção, o que significa que a mesma quantidade de qualquer saída deve ser produzida todos os dias, ou o mesmo a cada hora. O segundo objetivo examinado no escalonamento *Just-in-Time* está na minimização total dos custos de antecipação e atraso. Esta minimização expressa o objetivo de reduzir custos de inventário e, simultaneamente, satisfazer as demandas dos clientes com a entrega dos itens no tempo estabelecido. Este objetivo dá origem a medidas de desempenho não-regulares e, portanto, tem-se assim, novos problemas metodológicos no projeto de algoritmos para escalonamento.

Problemas de escalonamento com duas funções objetivo, ou seja, a minimização de variação de produção e custos de antecipação e atraso, o que são considerados no planejamento e controle de produção *Just-in-Time*, possuem enormes aplicações em sistemas computacionais. A classe mais importante de tais sistemas, que trabalham no ambiente *Just-in-Time*, são os denominados sistemas de tempo real. O requisito principal de um sis-

tema de tempo real está no tempo de resposta a um dada entrada, que não deve ultrapassar um determinado intervalo de tempo. Sistemas que operam com restrições de tempo são aplicáveis em diversas áreas como sistemas embarcados, controle de veículos, automação de instalações industriais, robótica, áudio e vídeo, monitoramento e bolsa de valores.

Um bom exemplo de um sistema de tempo real é um teclado e monitor de um computador pessoal. Um usuário deve receber a resposta visual de cada tecla pressionada dentro de um período razoável de tempo. Se o usuário não conseguir ver qual tecla foi acionada em um determinado período de tempo, o software com certeza poderá não ser mais utilizado. Como os critérios de otimização são os mesmos para as duas aplicações supracitadas, os algoritmos para problemas de escalonamento *Just-in-Time* podem ser utilizados tanto para sistemas baseados em manufatura como para sistemas de tempo real.

Outra variação do escalonamento *Just-in-Time* é a que considera a possibilidade de reduzir o tempo de processamento das tarefas a fim de atender as datas de término sugeridas, esta variação do problema é a considerada nesta pesquisa. Reduzir o tempo de processamento das tarefas pode adicionar custos adicionais, por exemplo, mais recursos devem ser requeridos. Este custo é adicionado à função de antecipação e atraso, e o objetivo é a minimização do custo total.

Os problemas de minimização do custo total de antecipação/atraso ocorrem em muitas áreas, não apenas nos sistemas de produção *Just-in-Time*. Muitas atividades como compras, transporte, manutenção ou operações militares requerem o controle *Just-in-Time*. Uma das áreas mais importantes que esta abordagem lida é a redução significativa de custos de logística. A logística é a técnica de gerenciar e controlar o fluxo de bens, energia, informações e outros recursos como produtos, serviços e pessoas, desde a fonte de produção até ao mercado consumidor. Assim, a logística deve ter a quantidade certa no momento e no preço certo. Seguindo esta definição que a abordagem *Just-in-Time* atende aos objetivos estabelecidos na logística.

Seguindo esta filosofia, de acordo com Moraes e Nogueira (2014), a Moto Honda da Amazônia, no Polo Industrial de Manaus, teve o seu desempenho de atividade aumentado por conta das formas de organização da produção e reestruturação empresarial. Em sua prática de produção e no relacionamento com seus fornecedores, a Honda adotou o sistema *Just-in-Time*. Este sistema, introduzido inicialmente pela Toyota, no Japão, foi criado especialmente para reduzir estoques e custos decorrentes. Com o *Just-in-Time*, o

produto ou matéria prima chega ao local de utilização somente no momento exato em que for necessário, com exceção dos insumos nacionais e internacionais.

Para a Honda, os produtos devem ser fabricados ou entregues a tempo de serem montados. Para isso, os fornecedores locais foram treinados para estarem com seus produtos na plataforma de produção, minutos antes de serem utilizados, Enquanto o estoque de fornecedores do exterior corresponde em média a 10 dias da produção e o de nacionais, a 3 dias, não existe estoque para os componentes fornecidos localmente.

Esta política tem repercussões geográficas significativas, pois embora apenas 22 dos 118 fornecedores da empresa estejam localizados em Manaus, eles respondem por 60% do total de insumos utilizados pela empresa; por sua vez, a Honda abastece com mais de 70%, o mercado nacional de motos. Assim, qualquer desarmonia no abastecimento local corresponde a tribulações para o cumprimento de seus planos de produção, induzindo à aproximação constante com os fornecedores locais.

A Figura 2.3 apresenta uma aplicação do conceito JIT, muito utilizada na Moto Honda da Amazônia, a principal empresa do polo de duas rodas, dentre os vinte polos de produção do distrito industrial da Zona Franca de Manaus - ZFM. Pois a empresa necessita que os produtos sejam fabricados ou entregues a tempo de serem montados, para assim atender aos mercados nacional e internacional (MORAES; NOGUEIRA, 2014).

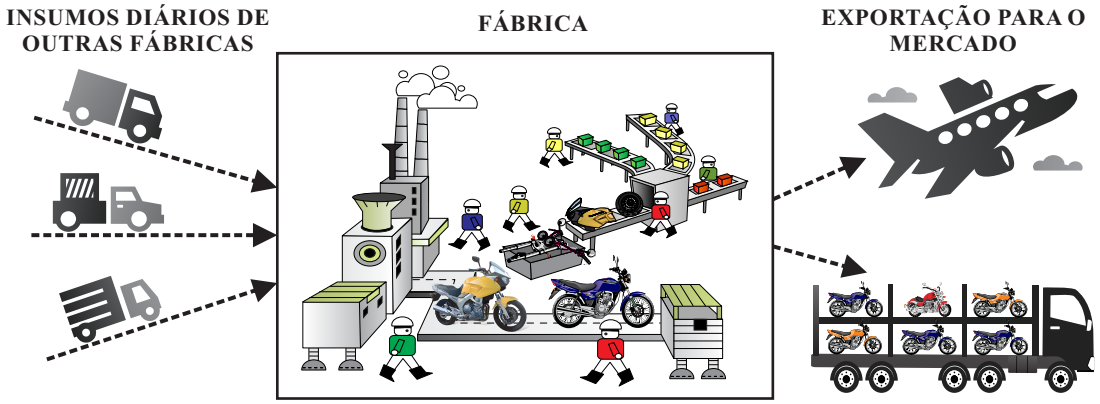


Figura 2.3 Aplicação do conceito JIT em uma empresa automotiva do Pólo Industrial de Manaus.

2.2.2 Tempo ocioso × não-ocioso

Como visto na seção anterior, um escalonamento com datas de término iguais não possui tempo ocioso em um escalonamento ótimo, antes que a última tarefa do escalonamento

seja completada. Entretanto, no caso das tarefas com datas de término distintas, pode ser vantajoso segurar a produção por algum tempo a fim de reduzir os custos decorrentes da antecipação de tarefas, assim, é possível que se tenha tempo ocioso no escalonamento. Esta seção baseia-se na seguinte referência: Józefowska (2007).

A Figura 2.4 apresenta dois exemplos de escalonamento, sem tempo ocioso (Figura 2.4 (a)) e com tempo ocioso (Figura 2.4 (b)), considerando duas tarefas com tempo de processamento igual a 6, datas de término sugeridas $d_1 = 8$ e $d_2 = 17$, em ambiente monoprocessado. Se o tempo ocioso não é permitido (Figura 2.4 (a)), a primeira tarefa inicia o seu processamento no instante 0, e possui um tempo de completude $C_1 = 6$. Dessa forma, a segunda tarefa inicia no instante 6 e finaliza seu processamento no instante 12, $C_2 = 12$. No escalonamento da Figura 2.4 (a) as tarefas são completadas antes da data de término sugerida, o valor de antecipação associado a cada tarefa é obtido desde que $d_1 - C_1 = 2 > 0$ e $d_2 - C_2 = 5 > 0$. No escalonamento da Figura 2.4 (b) dois períodos de tempo ocioso ocorrem, como resultado, as tarefas são completadas exatamente em suas datas de término sugeridas. Assim, os custos de antecipação e atraso são nulos.

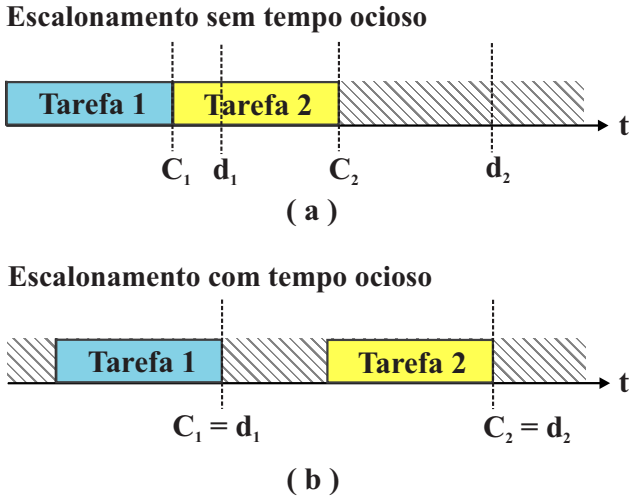


Figura 2.4 Escalonamento com tempo ocioso × não-ocioso.
Fonte: Józefowska (2007).

A inserção de tempo ocioso no escalonamento resulta em uma subutilização da máquina. A maximização na utilização da máquina geralmente entra em conflito com o valor de função objetivo JIT, o que pode impactar diretamente nos custos de estoque de uma empresa. Apesar do custo decorrente da existência de tempo ocioso na máquina não ser desprezível, ele é compatível com o conceito JIT quando se quer obter o menor valor de antecipação e atraso possível. Mas se o custo do tempo ocioso na máquina for elevado,

pode ser necessária a aplicação de uma restrição adicional a fim de evitar tempo ocioso no escalonamento. Assim, tem-se dois grupos principais de problemas de escalonamento: problemas com tempo ocioso permitido e problemas sem tempo ocioso. O primeiro caso é mais consistente com a ideia do escalonamento JIT, enquanto que a versão sem tempo ocioso pode ter seu uso justificado em outras aplicações.

Dependendo do problema de antecipação e atraso considerado, tem problemas de escalonamento que não permitem tempo ocioso entre as tarefas devido as restrições do problema, como as linhas de produção de uma fábrica, por exemplo, que não podem parar e devem seguir um fluxo contínuo de produção. Outro exemplo é quando se tem máquinas muito caras, que precisam ser utilizadas em uma determinada aplicação, logo, a sua utilização deve ser maximizada. Dessa forma, a restrição que evita tempo ocioso, apesar de não ser muito compatível com o conceito JIT, é também considerada na literatura.

Por outro lado, há problemas que permitem tempo ocioso no escalonamento, também devido as restrições do problema, ou seja, existem problemas que requerem que as tarefas terminem o mais próximas da data de término possível, a fim de atender a um determinado cronograma de produção ou atender as necessidades de um cliente, por exemplo. Dessa forma, com um número razoável de tarefas terminando na data de término sugerida, os custos das penalidades de antecipação e atraso tendem a ser reduzidos. Assim, pode ser mais lucrativo considerar tempo ocioso em alguns problemas.

Os problemas de escalonamento com e sem tempo ocioso podem ser encontrados na literatura com as seguintes variações de datas de término sugeridas:

- Escalonamento com tempo ocioso:

  a) Pesos arbitrários (*arbitrary weights*): $\alpha \neq \beta$.

  b) Pesos proporcionais (*proportional weights*): $\alpha_i = \alpha p_i$ e $\beta_i = \beta p_i$, $i = 1, ..., n$, $\alpha, \beta \geq 0$.

- Escalonamento sem tempo ocioso:

  a) Pesos arbitrários (*arbitrary weights*): $\alpha \neq \beta$.

  b) Pesos independentes de tarefa (*job independent weights*): $\alpha_i = \alpha$ e $\beta_i = \beta$, $i = 1, ..., n$.

2.2.3 Ambiente monoprocessado com antecipação & atraso

O problema de Escalonamento E/T em máquina única consiste em encontrar uma sequência ótima de um conjunto de n tarefas a serem escalonadas em uma única máquina. Cada tarefa J_j possui um tempo de processamento inteiro p_j e uma data de término sugerida inteira d_j. Caso as tarefas estejam prontas para serem executadas a partir do instante zero e não houver preempção das tarefas, não é permitido tempo ocioso porque a capacidade da máquina é limitada, comparada à demanda, e não é tecnologicamente viável deixar a máquina parada por um longo tempo (M'HALLAH, 2007).

Sourd e Kedad-Sidhoum (2003) lidaram com problemas envolvendo datas de término sugeridas, custos de antecipação e custos de atraso distintos e, com o objetivo de determinar o custo mínimo do problema, eles propuseram novos limites inferiores através da decomposição de cada tarefa em operadores unários, que por sua vez são atribuídos a cada instante de tempo, o que fornece um escalonamento preemptivo. Assim, dependendo de onde esses operadores unários são atribuídos na janela de tempo, gera uma custo associado no escalonamento. A atribuição que gerar o menor custo para o escalonamento é assumida como o limite inferior válido.

Um algoritmo *branch-and-bound* é proposto baseado neste limite inferior, obtendo resultados rápidos para instâncias envolvendo 8, 10, 15, 20, 25 e 30 tarefas. Em outro trabalho, os autores Sourd e Kedad-Sidhoum (2008) apresentam um novo algoritmo *branch-and-bound* capaz de resolver instâncias com até 50 tarefas. O algoritmo é baseado na combinação da relaxação lagrangiana de restrição de recursos e novas regras de dominância.

Uma meta-heurística híbrida foi proposta por M'Hallah (2007) para o problema $1||\sum E_j + \sum T_j$. Essa heurística é baseada em população, envolvendo Algoritmo Genético e a exploração do espaço de busca envolve Busca Local e Recozimento Simulado. A heurística baseada em população objetiva uma otimização global enquanto que a heurística de busca local se esforça para a otimização de um ótimo local. A otimização global equipara-se a evolução, enquanto que a otimização local equipara-se à aprendizagem.

O autor afirma que a sincronização da evolução com aprendizagem gera uma heurística híbrida eficiente. O problema $1||\sum E_j + \sum T_j$ é NP-difícil, sendo uma generalização de $1||\sum T_j$ que também é NP-difícil. O algoritmo proposto possui dois níveis de hibridização, que são: baixo nível (onde é verificada a habilidade do Algoritmo Genético, de

atingir boas soluções, substituindo a operação genética de mutação pela busca local) e o alto nível (onde são utilizadas três heurísticas gulosas para gerar uma população inicial, e a população é refinada a cada geração pelo algoritmo de Recozimento Simulado). A abordagem além de gerar resultados promissores em um tempo razoável, atingindo o ótimo na maioria dos casos testados, também pode ser estendido para outros problemas que envolvem antecipação e atraso.

Kedad-Sidhoum, Solis e Sourd (2008) propuseram duas novas famílias de limites inferiores para o problema de escalonamento em que as tarefas possuem datas de término distintas com custos de antecipação e atraso. Primeiro foi feita uma generalização de duas atribuições de limites inferiores para o problema com máquinas monoprocessadas e máquinas paralelas, e segundo, foi investigada uma formulação indexada por tempo para o problema a fim de se obter eficientes limites inferiores através da geração de colunas e relaxação lagrangiana. Para a geração de limite superior, foi apresentado um algoritmo de busca local. A relaxação lagrangiana também é utilizada no trabalho de Detienne, Pinson e Rivreau (2010) para o problema $1|r_j|\sum_j \alpha_j E_j + \beta_j T_j$, apresentando resultados para até 70 tarefas sem datas de início e 40 tarefas com datas de início.

Um algoritmo *branch-and-bound* para o problema de escalonamento com penalidades de antecipação e atraso pode ser consultado em Tanaka, Sasaki e Araki (2003), onde o procedimento começa com uma solução inicial e obtida através do algoritmo de troca de pares adjacentes (*adjacent pairwise interchange* - API). Esse algoritmo sequencia a solução inicial ordenando as tarefas a partir da menor para a maior data de término sugerida, seguido de movimentos API, até que a solução não possa mais ser melhorada. O algoritmo *branch-and-bound* também foi proposto nos trabalhos de Chang (1999), Liaw (1999), Rabadi, Mollaghasemi e Anagnostopoulos (2004), Rebai e Kacem (2008) e Kianfar e Moslehi (2012).

Estratégias baseadas em programação dinâmica foram propostas por Mondal e Sen (2001), Tanaka e Fujikuma (2008), Tanaka e Sato (2013). No trabalho de Tanaka, Fujikuma e Araki (2009) foi utilizada a programação dinâmica para solucionar de até 300 tarefas em ambiente monoprocessado. Uma outra versão desse algoritmo é apresentada em Tanaka (2012b), onde o algoritmo começa o seu processamento através da relaxação lagrangeana do problema original e, então, restrições são adicionadas até que a diferença entre o limite inferior e o superior se torne zero. As relaxações são resolvidas pela progra-

mação dinâmica, e os estados desnecessários do algoritmo de programação dinâmica são eliminados no decorrer da execução do algoritmo, a fim de evitar o crescimento excessivo de estados, causado pela adição de restrições.

Diferentes estratégias aproximadas envolvendo o ambiente monoprocessado podem ser consultadas em Sourd (2006), que considerou o método de busca em vizinhança chamado de *dynasearch* ou busca em vizinhança muito grande, onde a vizinhança é obtida através da composição de várias operações de troca de posições do escalonamento e, como a vizinhança é exponencial, um algoritmo de programação dinâmica foi proposto a fim de se obter a solução ótima da vizinhança. Júnior e Carvalho (2007), considerando janelas de entrega e tempo de preparação da máquina dependente da sequência de produção, propuseram um método heurístico baseado em GRASP, busca local iterada e descida em vizinhança variável (*variable neighborhood descent*), onde a heurística possui dois passos: o primeiro é a determinação da sequência de tarefas e o segundo passo é a determinação da data ótima de início de processamento de cada tarefa na sequência.

Penna et al. (2012) consideraram o escalonamento de tarefas com penalidades de antecipação e atraso com janelas de entrega distintas (onde há um período para a conclusão de cada tarefa), onde utilizou-se um algoritmo heurístico baseado em três fases. A primeira delas diz respeito ao algoritmo GRASP para a geração de uma solução inicial viável, a segunda fase é baseada em busca tabu, a fim de que seja refinada a solução encontrada; e a terceira fase conta com o algoritmo de reconexão de caminhos como estratégia de pós-otimização. O algoritmo foi testado com até 75 tarefas e foi comparado com algoritmos da literatura, superando todos eles. Um algoritmo genético em ambiente monoprocessado foi proposto por Yousefi e Yusuff (2012), onde foi considerado um estudo de caso de uma aplicação do algoritmo na indústria, com tarefas que possuem datas de término iguais, e o algoritmo é aplicado para o escalonamento de produtos em uma linha de produção.

2.2.4 Máquinas paralelas com antecipação & atraso

Os problemas de escalonamento que envolvem restrições de antecipação e atraso consistem em achar a sequência ótima para um dado conjunto de n tarefas $N = \{J_1, J_2, ..., J_n\}$ com tempos de processamento $p_j (j = 1, 2, ..., n)$ a serem processadas em: **máquinas paralelas idênticas** P (cada máquina possui a mesma velocidade, e tempo de processamento para a tarefa j é p_j); **máquinas paralelas uniformes** Q (cada máquina i possui

a sua própria velocidade s_i, e o tempo de processamento da tarefa j nessa máquina será $p_{ij} = p_j/s_i$) ou **máquinas paralelas não-relacionadas** R (a velocidade da máquina é dependente da tarefa a ser executada, e o tempo de processamento da tarefa j na máquina i é $p_{ij} = p_j/s_{ij}$). Os casos de *job shop* (máquinas diferentes e em ordem diferente para cada tarefa), *flow shop* (onde se tem uma sequência de máquinas para cada tarefa) e *open shop*, similar ao *flow shop*, diferenciando-se pelo fato de que cada tarefa com múltiplas operações tem associada qualquer ordem de processamento (cada máquina opera exatamente uma vez em cada sequência de tarefas, diferentes para cada tarefa) também são detalhados na literatura (LAUFF; WERNER, 2004). Kravchenko e Werner (2011) apresentam uma revisão da literatura envolvendo problemas de escalonamento em máquinas paralelas.

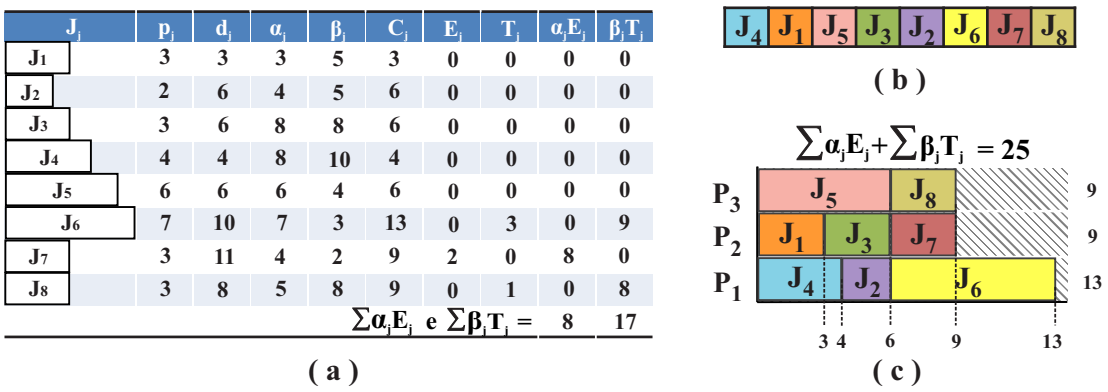
2.2.4.1 Máquinas paralelas idênticas

Nesta subseção, são apresentados os principais trabalhos encontrados na literatura envolvendo ambientes de máquinas paralelas idênticas (P). Ressalta-se que os trabalhos encontrados que envolvem máquinas paralelas uniformes e não-relacionadas também se adequam, a menos de algumas adaptações necessárias em alguns casos, ao caso de ambiente em máquinas paralelas idênticas. Ainda assim, não foram encontrados muitos trabalhos para este caso específico somente, e os dois principais encontrados são apresentados a seguir.

Kedad-Sidhoum, Solis e Sourd (2008) resolvem o problema $P|r_j|\sum_j \alpha_j E_j + \beta_j T_j$ utilizando programação inteira, através de uma estratégia algorítmica combinando geração de colunas, relaxação lagrangeana e heurística de busca local.

Existem na literatura abordagens para o ambiente de máquinas paralelas idênticas, mas que também consideram o ambiente monoprocessado, como pode ser observado em deFreitas et al. (2008), onde foi proposto um algoritmo heurístico para o problema $P||\sum w_j T_j$ em que o escalonamento tanto em uma máquina quanto em máquinas paralelas é representado por uma lista sequencial de tarefas (Figura 2.5 (b)). A Figura 2.5 (c) mostra representação de um escalonamento em máquinas paralelas idênticas através do **gráfico de Gantt**.

Para este mesmo problema de escalonamento (considerando somente máquinas paralelas idênticas), Croce, Garaix e Grosso (2012) apresentaram uma heurística de melhoria cujos resultados computacionais apresentados são melhores que os resultados existentes



J_j	p_j	d_j	α_j	β_j	C_j	E_j	T_j	$\alpha_j E_j$	$\beta_j T_j$
J_1	3	3	3	5	3	0	0	0	0
J_2	2	6	4	5	6	0	0	0	0
J_3	3	6	8	8	6	0	0	0	0
J_4	4	4	8	10	4	0	0	0	0
J_5	6	6	6	4	6	0	0	0	0
J_6	7	10	7	3	13	0	3	0	9
J_7	3	11	4	2	9	2	0	8	0
J_8	3	8	5	8	9	0	1	0	8
						$\sum\alpha_j E_j$ e $\sum\beta_j T_j$ =		8	17

(a)

Figura 2.5 (a) Exemplo de uma instância de 8 tarefas para o problema de escalonamento com antecipação e atraso. Exemplos de escalonamento através da representação por (b) lista sequencial de tarefas e (c) máquinas paralelas idênticas utilizando o gráfico de Gantt orientado-a-máquina.

na literatura, incluindo os apresentados por deFreitas et al. (2008). Um algoritmo exato pode ser consultado em Pessoa et al. (2010), onde foi proposto um método *branch-cut-and-price* para o problema $P||\sum w_j T_j$, considerando o ambiente monoprocessado, além de máquinas paralelas idênticas.

Tabela 2.1 Classificação dos problemas de escalonamento com antecipação e atraso de tarefas.

Ambiente	Problemas	Estratégias de resolução	Referências
1	$1\|s-batch,d_j = d\|\sum(\alpha_j E_j+\beta_j T_j+Ky_j)$, Ky_j é o custo da entrega do lote	Programação dinâmica	Herrmann e Lee (1993)
	$1\|\|\sum\alpha_i E_i+\sum\beta_i T_i$	Programação dinâmica	Tanaka e Fujikuma (2008), Tanaka, Fujikuma e Araki (2009), Tanaka (2012b)
	$1\|ST_{sd}\|\sum w_j E_j+\sum w_j T_j$	Busca tabu	Kolahan e Liang (1998)
	$1\|ST_{sd}\|\sum w_j E_j+\sum w_j T_j$	Método de busca *Dynasearch*	Sourd (2006)
	$1\|\|\sum\alpha_j E_j+\sum\beta_j T_j$	*branch-and-bound*	Sourd e Kedad-Sidhoum (2003), Sourd (2009)

Continua na próxima página

Tabela 2.1 – continuação da página anterior

Ambiente	Problemas	Estratégias de resolução	Referências
	$1\|r_j\|\sum\alpha_jE_j+\sum\beta_jT_j$	*branch-and-bound*	Sourd e Kedad-Sidhoum (2008)
	$1\|d_j\|\sum E_j+\sum T_j$	Algoritmo genético híbrido envolvendo busca local e recozimento simulado	M'Hallah (2007)
	$1\|ST_{sd},s_{ij}\|\sum_{j=1}^{n}(\alpha_iT_j+\beta_jE_j)$	GRASP, busca local iterada e descida em vizinhança variável	Júnior e Carvalho (2007)
	$1\|d_i\|\sum w_iE_i+\sum h_iT_i$	*branch-and-bound*	Li (1997), Rebai e Kacem (2008)
	$1\|\|\sum\alpha_iE_i+\sum\beta_iT_i$	*branch-and-bound*	Tanaka, Sasaki e Araki (2003)
	$1\|ST_{sd},E_i \leq d_i \leq T_i\|\sum_{i=1}^{n}(\alpha_iE_i+\beta_iT_i)$, onde $E_i \leq d_i \leq T_i$ indica a janelas de entrega distintas	GRASP, busca tabu e reconexão de caminhos	Penna et al. (2012)
	$1\|d_j=d\|\sum_{i=1}^{n}(\alpha_iE_i+\beta_iT_i)$	Algoritmo genético	Yousefi e Yusuff (2012)
	$1\|r_{ij},DW,batch\|\sum_{ij}\beta_{ji}E_{ij}+\sum_{ij}\alpha_{ij}T_{ij}$. Onde, $r_{ij}=$ datas de início, $DW=$ janela de entrega (*due window*), tal que, d_{ij}^{l} e d_{ij}^{u} representam as datas de término de antecipação e atraso, respectivamente, da tarefa i pertencente ao cliente j	Formulação matemática e um algoritmo híbrido imperialista competitivo baseado em busca local (*imperialist competitive algorithm* - ICA)	Ahmadizar e Farhadi (2014)
	$1\|d_j=d\|\sum_{i=1}^{n}(\alpha_iE_i+\beta_iT_i)$	Algoritmo exato com Recozimento Simulado	Awasthi, Lässig e Kramer (2013)

Continua na próxima página

Tabela 2.1 – continuação da página anterior

Ambiente	Problemas	Estratégias de resolução	Referências
	$1\|r_j\|\sum\alpha_jE_j+\sum\beta_jT_j$	*branch-and-cut* via CPLEX da formulação matemática inteira mista proposta, baseada na decomposição de intervalo de tempo (*interval-indexed formulation*)	Baptiste e Sadykov (2009)
	$1\|\|\sum(E_i+T_i)$	*branch-and-bound*	Chang (1999)
	$1\|r_j\|\sum_j\alpha_jE_j+\beta_jT_j$	relaxação lagrangiana	Detienne, Pinson e Rivreau (2010)
	$1\|TW\|\sum_j w_j(E_j+T_j)$, onde TW representa a janela de tempo em que cada tarefa pode ser processada	algoritmo de tempo $O(n^{m+1}max_i\{\alpha_i\}^m)$	Detienne, Pinson e Rivreau (2010)
	$1\|\|\sum(h_jE_j^2+w_jT_j^2)$	*branch-and-bound*	Kianfar e Moslehi (2012)
	$1\|\|\sum_{i=1}^n(\alpha_iE_i+\beta_iT_i)$	*branch-and-bound*	Liaw (1999)
	$1\|d_j=d\|\sum_j(\alpha E_i+\beta T_i)$	programação dinâmica e *branch-and-bound*	Mondal e Sen (2001)
	$1\|d_j=d,S_{ij}\|\sum_j(E_i+T_i)$	*branch-and-bound*	Rabadi, Mollaghasemi e Anagnostopoulos (2004)
	$1\|d_j=d\|\sum_j(\alpha_iE_i+\beta_iT_i)$	*branch-and-bound*	Ronconi e Kawamura (2010)
	$1\|\|\sum_j(\alpha_iE_i+\beta_iT_i)$	*branch-and-bound*	Sourd (2005)
	$1\|prec,noidle\|\sum(\alpha_iE_i+\beta_iT_i)$	programação dinâmica	Tanaka e Sato (2013)
	$1\|d_j=d\|\sum_i^n\alpha_iE_i+\beta_iT_i$	formulação matemática	Alvarez-Valdes et al. (2012)
	$1\|d_j=d\|\sum_i^n\alpha_iE_i+\beta_iT_i$	heurística e formulação matemática	Benmansour, Allaoui e Artiba (2011)
	$1\|\|\sum_i^n\alpha_iE_i+\beta_iT_i$	busca tabu	James e Buchanan (1997)

Continua na próxima página

Tabela 2.1 – continuação da página anterior

Ambiente	Problemas	Estratégias de resolução	Referências
	$1\|S_{ij}\|\sum_i^n \alpha_i E_i + \beta_i T_i$	algoritmo genético	Ribeiro et al. (2010)
	$1\|d_j = d\|\sum_{j=1}^n \alpha_j E_j + \beta_j T_j$	algoritmo auto-evolutivo	Weng e Fujimura (2008)
	$1\|d_{res}\|\sum_{j=1}^n \alpha_j E_j + \beta_j T_j$	heurística baseada em busca de vizinhança variável (*variable neighborhood search*)	Liu e Zhou (2013)
	$1\|\|\sum_{j=1}^n \alpha_j E_j + \beta_j T_j$	algoritmo genético	Yu et al. (1999)
	$1\|\|\sum(u_i E_i + w_i T_i)$	busca tabu	Wodecki (2009)
	$1\|d_i = d\|\sum(\alpha_i E_i + \beta_i T_i)$	busca tabu	Alharkan, Aziz e Alhaag (2015)
	$1\|\|\sum(\alpha_i E_i + \beta_i T_i)$	busca local iterada	Kedad-Sidhoum e Sourd(2010)
	$1\|\|\sum_{j=1}^n (E_j + T_j^2)$	Busca local iterada	Qin et al. (2015)
	$1\|d_j = d\|\sum_{j=1}^n (\alpha_j E_j + \beta_j T_j)$	busca dispersa	Talebi et al. (2009)
	$1\|d_j = d\|\sum_{j=1}^n (\alpha_j E_j + \beta_j T_j)$	algoritmo evolutivo baseado em Storn e Price (1997)	Nearchou (2006)
	$1\|pmtn\|\sum \alpha_i E_i + \beta_i T_i$	algoritmo baseado no tempo (*timing algorithm*) e busca local	Runge e Sourd (2009)
	$1\|d_j = d\|\sum(E_j + T_j)$	heurística baseada em bloco de tarefas	Mason, Jin e Jampani (2005)
	$1\|s_{ij}\|\sum_{i=1}^n (\alpha_i E_i + \beta_i T_i)$	algoritmo genético	Wang e Wang (1997)
	$1\|d_j = d\|\sum_{j=1}^n (\alpha_j E_j + \beta_j T_j)$	heurística evolucionária baseada em busca local	Tasgetiren et al. (2007)
P	$P\|\|nP_D d + \sum_{j=1}^n (P_E E_j + P_T T_j)$, $nP_D d$ é uma penalidade aplicada quando há datas de término iguais	abordagem heurística baseada em formulação matemática	Xiao e Li (2002)

Continua na próxima página

Tabela 2.1 – continuação da página anterior

Ambiente	Problemas	Estratégias de resolução	Referências
	$P\|r_j\|\sum_j \alpha_j E_j + \beta_j T_j$	programação inteira, geração de colunas, relaxação lagrangeana e busca local	Kedad-Sidhoum, Solis e Sourd (2008)
	$P\|d_j = d\|\sum_{i=1}^{n} (\alpha_i E_i + \beta_i T_i)$	algoritmo exato com recozimento simulado	Awasthi, Lässig e Kramer (2013)
	$P\|TW\|\sum_j w_j(E_j + T_j)$, onde TW representa a janela de tempo em que cada tarefa pode ser processada	algoritmo de tempo $O(n^{m+1} max_i\{\alpha_i\}^m)$	Detienne, Pinson e Rivreau (2010)
	$P\|\|\sum_j \max(E_j, T_j)$	estratégia híbrida baseada em recozimento simulado	Tamaki, Murao e Kitamura (2003)
	$P\|\|\sum_{j=1}^{n} (\alpha_j E_j + \beta_j T_j)$	algoritmo genético híbrido	Amorim (2013)
	$P_m\|\|\sum_{j=1}^{n} \alpha_j E_j + \beta_j T_j$	heurística baseada em busca de vizinhança variável (*variable neighborhood search*)	Batista e Batista (2015)
	$P\|d^i = d^r\|\sum_i (\alpha_i E_i + \beta_i T_i)$, onde $d^i = d^r$ significa que todas as tarefas compartilham a mesma data de término restritiva	busca local, reconexão de caminhos e busca dispersa	Alvarez-Valdes, Tamarit e Villa (2014)
	$P\|\|\sum_{j=1}^{n} (\alpha_j E_j + \beta_j T_j)$	heurística unificada baseada em busca local	Kramer e Subramanian (2015)

2.3 Resumo do capítulo

Neste capítulo foi apresentada a revisão da literatura realizada envolvendo problemas de escalonamento E/T. Foram apresentadas as definições e conceitos principais e notação clássica. Na literatura existe uma ampla e crescente quantidade de trabalhos que consideram problemas de escalonamento com antecipação e atraso, dentre eles destacam-se as revisões da literatura apresentadas por Baker e Scudder (1990), Gordon, Proth e Chu (2002) e Shabtay e Steiner (2012). Este capítulo foi estruturado com base na análise realizada sobre o problema, e os trabalhos e resultados encontrados.

Capítulo 3

Formulações inteira e mista para o escalonamento E/T

Neste capítulo são detalhadas as principais formulações matemáticas existentes na literatura para o problema de escalonamento clássico com penalidades de antecipação e atraso (tarefas independentes, com tempos arbitrários de processamento, sem preempção permitida, com datas de chegada iguais e datas de término distintas). Sendo assim, duas questões ajudam na classificação das formulações matemáticas existentes:

1. permitem tempo ocioso no escalonamento ou não;
2. consideram escalonamento em máquinas paralelas ou somente em uma única máquina.

Kanet e Sridharan (2000) apresentaram uma revisão da literatura sobre problemas de escalonamento com tempos ociosos inseridos, onde são considerados problemas de escalonamento que envolvem penalidades de antecipação e atraso, com resultados gerais sobre quando considerar ou não tempo ocioso em problemas de escalonamento no geral. Assim, afirmam que não é necessário considerar ou tratar tempo ocioso nos dois seguintes casos:

1. Problemas de escalonamento em ambiente monoprocessado.
2. Problemas de escalonamento em máquinas paralelas, quando todas as tarefas do problema estão simultaneamente disponíveis (quando se tem datas de chegadas iguais) e quando o problema de escalonamento apresentar medidas regulares de performance.

3.1 Formulação de programação linear inteira mista para o escalonamento em máquinas paralelas com tempo ocioso (PIM-TO)

A primeira formulação considerada é o modelo de programação linear inteira mista (PIM) proposto por Arenales et al. (2007) para o problema de escalonamento em máquinas paralelas. A formulação proposta assume que: todas as tarefas devem estar disponíveis no instante zero (datas de chegadas iguais, $r_j = 0$, sem perda de generalidade); cada tarefa possui a sua própria data de término sugerida (d_j distintos); qualquer máquina pode processar qualquer tarefa; cada tarefa pode ser executada somente uma vez; a preempção de uma tarefa não é permitida; cada máquina pode processar exatamente uma tarefa em um dado instante de tempo; o número de tarefas e máquinas são fixos e os tempos de processamento também são fixos. E apresenta tempos de preparação relacionados a máquina e tarefas consecutivamente escalonadas.

A variável de decisão inteira é binária e tri-indexada, sendo:

$$x_{ijk} = \begin{cases} 1, \text{ se a tarefa } i \text{ precede imediatamente a tarefa } j \text{ na máquina } k \\ 0, \text{ caso contrário} \end{cases}$$

As outras variáveis, que podem receber valores reais, são: C_{ik} que é o instante de término do processamento da tarefa i na máquina k; e, T_i e E_i representam o atraso e a antecipação da tarefa i, respectivamente.

A formulação matemática também apresenta os seguintes parâmetros não-negativos:

- p_{ik} é o tempo de processamento da tarefa i na máquina k (para máquinas paralelas idênticas a notação muda para p_j);
- s_{ijk} é o tempo de preparação da máquina k para processar a tarefa j imediatamente após a tarefa i (caso o problema não tenha tempos de preparação, assume-se o valor zero para o tempo de preparação da máquina);
- d_j é a data de entrega da tarefa j; e, M é uma constante suficientemente grande (conhecida pelo termo em inglês, *big M*).

A seguir é apresentada a formulação matemática em programação inteira mista, com restrições e função objetivo lineares.

$$\text{Min} \sum_{j=1}^{n} (E_j + T_j) \tag{3.1}$$

$$\text{S. a.} \sum_{k=1}^{m} \sum_{i=0}^{n} x_{ijk} = 1 \qquad j = 1,2,...,n. \tag{3.2}$$

$$\sum_{j=1}^{n} x_{0jk} \leq 1 \qquad k = 1,2,...,m. \tag{3.3}$$

$$\sum_{i=0,\ i\neq h}^{n} x_{ihk} - \sum_{j=0,\ j\neq h}^{n} x_{hjk} = 0 \qquad h = 1,...,n \text{ e } k = 1,...,m. \tag{3.4}$$

$$C_{0k} = 0 \qquad k = 1,...,m. \tag{3.5}$$

$$C_{jk} \geq C_{ik} - M + (p_{jk} + s_{ijk} + M)x_{ijk} \quad i = 0,...,n; j = 1,...,n \text{ e } k = 1,...,m. \tag{3.6}$$

$$E_i \geq d_i - C_{ik} \qquad i = 1,...,n \text{ e } k = 1,...,m. \tag{3.7}$$

$$T_i \geq C_{ik} - d_i \qquad i = 1,...,n \text{ e } k = 1,...,m. \tag{3.8}$$

$$T_i \geq 0, E_i \geq 0 \qquad i = 1,...,n. \tag{3.9}$$

$$x_{ijk} \in \{0,1\} \qquad i,j = 0,...,n \text{ e } k = 1,...,m. \tag{3.10}$$

Na formulação acima, a função objetivo 3.1 minimiza a antecipação e o atraso das tarefas. As restrições 3.2, 3.3 asseguram uma sequência exclusiva de tarefas para cada uma das máquinas. Sendo que o conjunto de restrições 3.2 indica que cada tarefa j na máquina k tem apenas uma tarefa precedente. O conjunto de restrições 3.3 garante que cada máquina k, se usada, tenha uma sequência exclusiva de tarefas. O conjunto de restrições 3.4 define que cada tarefa j tenha uma única tarefa imediata sucessora, com exceção da tarefa 0 que estabelece o início e o fim de uma sequência de tarefas na máquina k. Para a tarefa 0, o conjunto de restrições 3.5 estabelece que o instante de término desta tarefa nas máquinas deve ser igual a zero. No conjunto de restrições 3.6, são verificados os instantes de conclusão das tarefas nas máquinas onde são executadas. Nas restrições 3.7 e 3.8, são definidos, respectivamente, a antecipação e o atraso associado a cada uma das tarefas. As restrições 3.9 e 3.10 indicam o domínio das variáveis utilizadas na formulação.

3.2 Formulação de programação linear inteira mista para o escalonamento em máquinas paralelas sem tempo ocioso (PIM-STO)

A segunda formulação considerada neste trabalho é o modelo de PIM proposto por Tsai e Wang (2012) para máquinas paralelas não-relacionadas ($R||\sum_{i=1}^{n} T_j + \sum_{i=1}^{n} E_j$) baseado na formulação apresentada por Rabadi, Moraga e Al-Salem (2006) para a minimização do *makespan* em máquinas paralelas não-relacionadas. A formulação proposta por Tsai e Wang (2012) também assume que: todas as tarefas devem estar disponíveis no instante zero (datas de chegadas iguais, $r_j = 0$, sem perda de generalidade); cada tarefa possui a sua própria data de término sugerida (d_j distintos); qualquer máquina pode processar qualquer tarefa; cada tarefa pode ser executada somente uma vez; a preempção de uma tarefa não é permitida; cada máquina pode processar exatamente uma tarefa em um dado instante de tempo; o número de tarefas e máquinas é fixo e os tempos de processamento também são fixos e apresenta tempos de preparação relacionados à máquina e tarefas consecutivamente escalonadas.

Apresenta a mesma variável de decisão inteira binária e tri-indexada, sendo:

$$x_{ijk} = \begin{cases} 1, \text{ se a tarefa } j \text{ é processada diretamente depois da tarefa } i \text{ na máquina } k \\ 0, \text{ caso contrário} \end{cases}$$

e

$$x_{0ik}/x_{i0k} = \begin{cases} 1, \text{ se a tarefa } i \text{ é a primeira/última a ser processada na máquina } k \\ 0, \text{ caso contrário} \end{cases}$$

As variáveis contínuas são: C_j, tempo de completude da tarefa j; T_j, atraso da tarefa j, e E_j, antecipação da tarefa j. A formulação matemática também apresenta os seguintes parâmetros não-negativos: n**:** número de tarefas; m**:** número de máquinas; p_{jk}**:** tempo de processamento da tarefa j na máquina k; d_j**:** data de término sugerida da tarefa j; e, M**:**

uma constante suficientemente grande.

A seguir é apresentada a formulação matemática em programação inteira mista, com restrições e função objetivo lineares.

$$\text{Min} \sum_{j=1}^{n} E_j + \sum_{j=1}^{n} T_j \tag{3.11}$$

$$\text{S. a.} \sum_{i=0,\ i\neq j}^{n} \sum_{k=1}^{m} x_{ijk} = 1 \qquad j = 1,2,...,n. \tag{3.12}$$

$$\sum_{i=0,\ i\neq h}^{n} x_{ihk} - \sum_{j=0,\ j\neq h}^{n} x_{hjk} = 0 \qquad h = 1,2,...,n;\ \ k = 1,2,...,m. \tag{3.13}$$

$$C_i + \sum_{k=1}^{m} x_{ijk}(p_{jk}) + M(\sum_{k=1}^{m} x_{ijk} - 1) \leq C_j \qquad i = 0,1,...,n;\ \ j = 1,2,...,n. \tag{3.14}$$

$$\sum_{j=0}^{n} x_{0jk} = 1 \qquad k = 1,2,...,m. \tag{3.15}$$

$$T_j \geq C_j - d_j \qquad j = 1,2,...,n. \tag{3.16}$$

$$E_j \geq d_j - C_j \qquad j = 1,2,...,n. \tag{3.17}$$

$$C_0 = 0 \tag{3.18}$$

$$C_j, E_j, T_j \geq 0 \qquad j = 1,2,...,n. \tag{3.19}$$

$$x_{ijk} \in \{0,1\} \quad i,j = 1,2,...,n;\ \ k = 1,2,...,m. \tag{3.20}$$

$$C_0 + \sum_{k=1}^{m} x_{0jk}(p_{jk}) \geq C_j + M(\sum_{k=1}^{m} x_{0jk} - 1) \qquad j = 1,2,...,n. \tag{3.21}$$

$$C_i + \sum_{k=1}^{m} x_{ijk}(p_{jk}) \geq C_j + M(\sum_{k=1}^{m} x_{ijk} - 1) \qquad i = 0,1,...,n;\ \ j = 1,2,...,n. \tag{3.22}$$

Para adaptar o modelo para o problema envolvendo máquinas paralelas idênticas (e não o caso geral, de não-relacionadas), $P||\sum_{i=1}^{n} \alpha_j E_j + \sum_{i=1}^{n} \beta_j T_j$, a notação referente ao tempo de processamento das tarefas deve ser mudada de p_{jk} (que indica que a tarefa j é processada na máquina k) para somente p_j (já que as máquinas são idênticas e não possuem desempenhos dependentes da tarefa a ser executada) e as penalidades de antecipação e atraso devem ser adicionadas na função objetivo, que fica como a seguir:

$$\text{Min} \qquad \sum_{j=1}^{n} \alpha_j E_j + \sum_{j=1}^{n} \beta_j T_j \tag{3.23}$$

Na formulação acima, a função objetivo 3.11 minimiza a antecipação e o atraso das ta-

refas. O conjunto de restrições 3.12 assegura que uma tarefa seja escalonada somente uma vez e processada por somente uma máquina. O conjunto de restrições 3.13 assegura que cada tarefa *i* não seja nem precedida e nem sucedida por mais de uma tarefa *j*. O conjunto de restrições 3.14 é utilizado para calcular o tempo de completude das tarefas e assegurar que nenhuma tarefa *i* deve preceder e suceder ao mesmo tempo uma determinada tarefa *j* no escalonamento.

O conjunto de restrições 3.15 assegura que não mais que uma tarefa seja escalonada primeiro em uma máquina. Os valores para atraso e antecipação das tarefas são calculados nas restrições 3.16 e 3.17, respectivamente. O conjunto de restrições 3.18 define que o tempo de completude da tarefa fictícia 0 seja zero. O conjunto de restrições 3.19 assegura que os valores para tempos de completude, antecipação e atraso sejam não-negativos. O conjunto de restrições 3.20 assegura que a variável de decisão x_{ijk} seja binária.

Como as restrições acima não excluem a ocorrência de tempo ocioso entre as tarefas, Tsai e Wang (2012) propuseram mais duas novas restrições para solucionar o problema. A primeira proposta é o conjunto de restrições 3.21, que assegura que o tempo de completude da primeira tarefa no escalonamento seja igual ao seu tempo de processamento.

O segundo conjunto de restrições 3.22 assegura que o tempo de completude de uma tarefa *j* seja igual à soma do tempo de completude da tarefa que foi processada anteriormente a ele (tarefa *i*), com o seu tempo de completude.

3.3 Formulação indexada pelo tempo para o escalonamento em ambiente monoprocessado (PI-STO-IT)

A terceira formulação considerada neste trabalho é um modelo de programação linear inteira (PI) proposto por Tanaka, Fujikuma e Araki (2009) para a minimização das penalidades de antecipação e atraso em ambiente monoprocessado (baseado nas formulações propostas por Pritsker, Watters e Wolfe (1968), Dyer e Wolsey (1990), Sousa e Wolsey (1992) e Akker, Hoesel e Savelsbergh (1999)).

Apresenta uma variável de decisão inteira binária e bi-indexada x_{it} $(i \in N, 1 \leq t \leq T)$, tal que:

$$x_{it} = \begin{cases} 1, \text{ se a tarefa } i \text{ é completada no instante } t(t = C_i) \\ 0, \text{ caso contrário} \end{cases}$$

Apresenta os seguintes parâmetros não-negativos: p_j, indicando o tempo de processamento da tarefa j; e, T uma constante indicando a soma dos tempos de processamento. As tarefas devem ser processadas no intervalo de tempo entre 1 e T.

Desta forma, o problema pode ser formulado como segue:

$$\text{Min} \sum_{i=1}^{n} \sum_{t=1}^{T} f_i(t) x_{it} \tag{3.24}$$

$$\text{S. a.} \sum_{i=1}^{n} \sum_{s=\max\{t,p_i\}}^{min(T,\ t+p_i-1)} x_{is} = 1 \quad 1 \leq t \leq T \tag{3.25}$$

$$\sum_{t=p_i}^{T} x_{it} = 1 \quad i \in N \tag{3.26}$$

$$x_{it} \in \{0,1\}, \quad i \in N, \quad 1 \leq t \leq T \tag{3.27}$$

O conjunto de restrições (3.25) define a restrição de capacidade da máquina, onde cada máquina deve processar exatamente uma tarefa em um determinado instante de tempo. O conjunto de restrições (3.26) indica o número de ocorrências das tarefas no escalonamento. Para a função objetivo (3.24), minimização da antecipação e atraso, tem-se: $f_j(t) = \alpha_i \cdot \max\{d_i - t, 0\} + \beta_i \cdot \max\{t - d_i, 0\}$.

3.4 Formulação clássica geral indexada pelo tempo para o escalonamento em ambiente monoprocessado (PI-TO-IT)

A quarta formulação considerada se baseia na formulação geral proposta para problemas de escalonamento com funções objetivo regulares (não-decrescentes em relação aos tempos de completude das tarefas) proposta por Dyer e Wolsey (1990).

As variáveis de decisão são binárias bi-indexadas, como segue:

$$x_j^t = \begin{cases} 1, \text{ se } j \text{ inicia o seu processamento no instante } t \text{ em alguma máquina} \\ 0, \text{ caso contrário} \end{cases}$$

A formulação também apresenta os seguintes parâmetros não-negativos: p_j, indicando o tempo de processamento da tarefa j; e, T uma constante indicando o limite superior para o intervalo de tempo em que as tarefas devem ser escalonadas (neste caso, intervalo de tempo entre 0 e T.

Desta forma, o problema pode ser formulado como segue:

$$\text{Min} \sum_{j \in J} \sum_{t=0}^{T-p_j} f_j(t+p_j) x^t{}_j \tag{3.28}$$

$$\text{S. a.} \sum_{t=0}^{T-p_j} x^t{}_j = 1 \ \ (\forall j \in J) \tag{3.29}$$

$$\sum_{j \in J} \sum_{s=\max\{0,t-p_j+1\}}^{min\{t,T-p_j\}} x^s{}_j \leq 1 \ \ (t = 0, ..., T-1) \tag{3.30}$$

$$x^t{}_j \in \{0,1\} \ \ (\forall j \in J; t = 0, ..., T-1) \tag{3.31}$$

Na função objetivo (3.28), a função $f_j(t+p_j)$ é definida para cada tarefa, onde $t+p_j$ é o instante onde cada tarefa finaliza o seu processamento na linha do tempo. Se a função objetivo é para a minimização do atraso ponderado, por exemplo, $f_j(t+p_j)$ é igual a $\beta_j \times \max\{0, x-d_j\}$. O conjunto de restrições (3.29) determina que a tarefa deve ser processada exatamente uma vez. O conjunto de restrições (3.30) determina que a máquina pode processar no máximo uma tarefa em um dado instante de tempo, sendo que a soma de tal instante com o seu respectivo tempo de processamento não pode ser maior que o máximo tempo de execução.

Somente 1 tarefa pode ser executada ao mesmo tempo (na única máquina disponível. Esta formulação pode ser adaptada para ambiente multiprocessado, trocando-se para m o lado direito desta restrição, indicando que no pior caso, somente m tarefas podem ser executadas ao mesmo tempo, onde m é o número de máquinas disponíveis. O conjunto de restrições (3.31) definem os tipos de variáveis utilizadas.

Uma vantagem importante da formulação indexada pelo tempo é que essa formulação pode ser utilizada para modelar vários critérios de desempenho, do tipo regulares (funções objetivo regulares) para problemas de escalonamento. A mudança na formulação se dá através da mudança dos custos da função objetivo. A desvantagem na utilização deste modelo é no tamanho: onde existem $n+T$ restrições e aproximadamente nT variáveis. T é a soma de todos os tempos de processamento $\sum_{j=1}^{n} p_j$ (para ambiente monoprocessado).

Considerando máquinas paralelas, Pessoa et al. (2010) definiu o valor de T como $\lfloor(\sum_{j=1}^{n} p_j - p_{max})/m\rfloor + p_{max}$, onde p_{max} é o maior tempo de processamento da tarefa, este valor é válido, pois, se uma tarefa i é completada após $\lfloor(\sum_{j=1}^{n} p_j - p_i)/m\rfloor + p_i$ em uma certa máquina, pode-se concluir que pelo menos uma outra máquina estará disponível no instante $\lfloor(\sum_{j=1}^{n} p_j - p_i)/m\rfloor$, dessa forma, a tarefa pode ser movida para essa outra máquina, reduzindo assim o tempo de completude do escalonamento. Desta forma, instâncias com muitas tarefas, ou tarefas com um grande tempo de processamento, demandam muito tempo para solução.

A formulação indexada pelo tempo contou com uma adaptação para o problema de escalonamento com funções não-regulares em máquinas paralelas sem tempo ocioso. Apesar da formulação não permitir tempo ocioso na minimização do atraso ponderado de tarefas (funções regulares), é necessário que algumas restrições sejam adicionadas para evitar tempo ocioso na minimização da antecipação e atraso ponderados (funções não-regulares). As restrições a serem adicionadas seriam as que não permitissem tempo ocioso no início e no meio da sequência.

Assim, considerando o problema de escalonamento desta pesquisa, sem tempo ocioso, uma formulação de programação inteira (que se baseia na adaptação da formulação clássica geral de escalonamento) foi proposta em Amorim (2013). A formulação proposta permite a utilização de funções objetivos não-regulares (não-crescentes em relação aos tempos de completude das tarefas) sem tempo ocioso entre as tarefas para os ambientes mono e multiprocessado.

As máquinas também não podem processar mais que uma tarefa em um dado instante. Todas as tarefas podem ser processadas a partir do instante 0 (zero). Portanto, as tarefas devem ser processadas no intervalo de tempo $[0,T]$, onde $T = \sum_{j=1}^{n} p_j$.

As variáveis de decisão utilizadas são binárias, como segue:

$$x^t_j = \begin{cases} 1, \text{ se a tarefa } j \text{ inicia o seu processamento no instante } t \text{ em alguma máquina} \\ 0, \text{ caso contrário} \end{cases}$$

Desta forma, o problema pode ser formulado como segue (Formulação indexada pelo tempo baseada no modelo de fluxo em redes para o escalonamento com funções não-regulares em máquinas paralelas sem tempo ocioso - *PI-NetworkFlow*):

$$\text{Min} \sum_{t=0}^{T} \sum_{j=1}^{n} f_j(t+p_j) x^t{}_j \tag{3.32}$$

$$\text{S. a.} \sum_{t=0}^{T} x^t{}_j = 1 \ \ (j = 1,2,...,n) \tag{3.33}$$

$$\sum_{j=1}^{n} x_j^{t-p_j} - \sum_{j=0}^{n} x_j^t = 0 \ \ (t = 1,...,T) \tag{3.34}$$

$$\sum_{j=1}^{n} x_j^0 = m \tag{3.35}$$

$$x^t{}_j \in \{0,1\} \ \ (j = 1,2,...,n; t = 0,...,T) \tag{3.36}$$

A função objetivo $f_j(t+p_j)$ (3.32) desta formulação é baseada no problema $P||\sum \alpha_j E_j + \sum \beta_j T_j$. Assim, o valor de função objetivo pode ser calculado como segue: $f_j(t+p_j) = \sum \alpha_j \cdot \max\{0, d_j - t + p_j\} + \sum \beta_j \cdot \max\{0, t + p_j - d_j\}$. O conjunto de restrições de atribuição (3.33) determina que a tarefa deve ser processada exatamente uma vez.

O conjunto de restrições de fluxo em redes (3.34) determina a representação do escalonamento através de fluxo em redes, o que garante que não se tenha tempo ocioso entre as tarefas no escalonamento, neste caso, o tempo ocioso é permitido somente quando todas as tarefas em cada máquina são processadas, o tempo ocioso é representado pela tarefa fictícia 0 (zero), que é utilizada no final da sequência de tarefas em cada máquina, como esta tarefa fictícia é a última da sequência de tarefas, ela é direcionada ao instante final do escalonamento, essa restrição é ilustrada na Figura 3.1 (a). O conjunto de restrições de capacidade (3.35) define o instante de início do escalonamento. No início do escalo-

namento é assegurado que a primeira tarefa da sequência em cada máquina comece no instante 0 (zero).

A Figura 3.1 (b) apresenta uma solução do escalonamento em máquinas paralelas idênticas utilizando a representação de fluxo em redes, para uma instância de 8 tarefas apresentada na Figura 2.5 (a). Cada caminho é representado por diferentes setas, que indica uma sequência de tarefas em cada máquina.

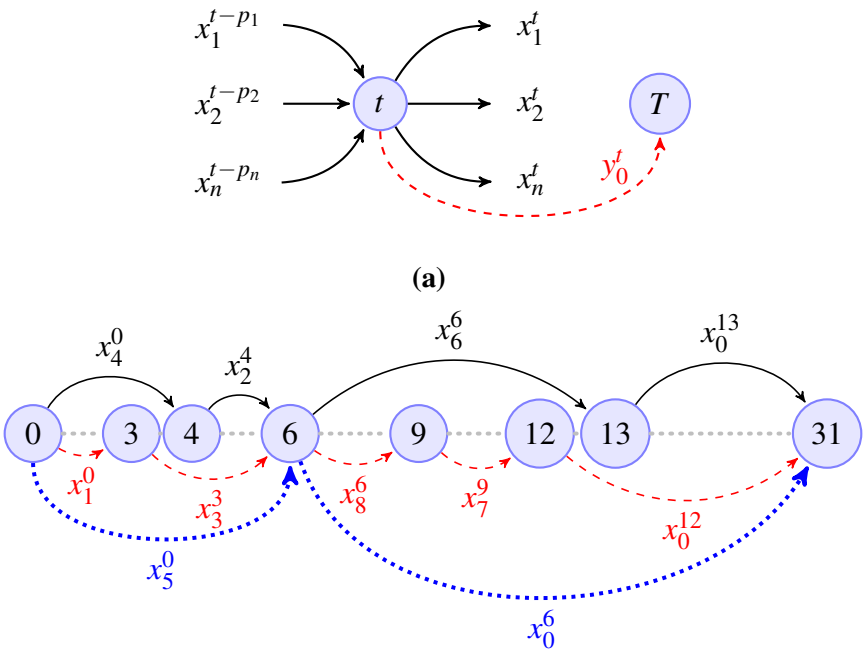


Figura 3.1 (a) Esquema do conjunto de restrições (3.34) proposta para a determinação da sequência de tarefas através de fluxo em redes. (b) representação do escalonamento no modelo de fluxo em redes para a formulação clássica do problema.

3.5 Formulação baseada no modelo de fluxo em redes e roteamento de veículos para o escalonamento em máquinas paralelas - *Arc-time indexed formulation* (PI-STO-ArcTime)

A quinta formulação, proposta por Pessoa et al. (2010) para a minimização do atraso ponderado de tarefas em ambiente mono e multiprocessado, baseia-se no modelo de fluxo em redes e roteamento de veículos (*Arc-time indexed formulation*) considera o escalonamento

no intervalo de tempo de 0 a T, onde as máquinas estão ociosas no instante 0 e devem estar novamente ociosas no instante T. As variáveis binárias x_{ij}^t, $i \neq j$, indicam que a tarefa i termina sua execução e a tarefa j inicia a sua execução no instante t, em alguma máquina.

A variável x_{0j}^t indica que a tarefa j inicia seu processamento no instante t em alguma máquina que estava ociosa no intervalo $t-1$ a t, em particular, a variável x_{0j}^0 indica que a tarefa j inicia em alguma máquina no instante 0. A variável x_{i0}^t indica que a tarefa i finaliza o seu processamento no instante t em uma máquina que ficará ociosa nos instantes t a $t+1$, a variável x_{i0}^T indica que a tarefa i será finalizada no instante final do intervalo de tempo. As variáveis x_{00}^t indicam o número de máquinas que estavam livres, no intervalo $t-1$ a t, irão permanecer ociosas no intervalo t a $t+1$. Seja $J_+ = \{0,1,...,n\}$ e $p_0 = 0$. A formulação é apresentada a seguir:

$$\text{Minimizar} \sum_{i \in J_+} \sum_{j \in J \setminus \{i\}} \sum_{t=p_i}^{T-p_j} f_j(t+p_j) x_{ij}^t \tag{3.37}$$

$$\text{Sujeito a} \sum_{i \in J_+ \setminus \{j\}} \sum_{t=p_i}^{T-p_j} x_{ij}^t = 1 \quad (\forall j \in J) \tag{3.38}$$

$$\sum_{j \in J_+ \setminus \{i\},\ t-p_j \geq 0} x_{ji}^t - \sum_{j \in J_+ \setminus \{i\},\ t+p_i+p_j \leq T} x_{ij}^{t+p_i} = 0 \quad (\forall i \in J; \tag{3.39}$$

$$t = 0,...,T-p_i)$$

$$\sum_{j \in J_+,\ t-p_j \geq 0} x_{j0}^t - \sum_{j \in J_+,\ t+p_j+1 \leq T} x_{0j}^{t+1} = 0 \quad (t = 0,...,T-1) \tag{3.40}$$

$$\sum_{i \in J_+} x_{0j}^0 = m \tag{3.41}$$

$$x_{ij}^t \in Z_+ \quad (\forall i \in J_+; \forall j \in J_+ \setminus \{j\}; t = p_i,...,T-p_j) \tag{3.42}$$

$$x_{00}^t \in Z_+ \quad (t = 0,...,T-1) \tag{3.43}$$

As restrições (3.39), (3.40) e (3.41) definem o fluxo em redes de m unidades (ou m máquinas) em um grafo acíclico $G = (V,A)$. Neste fluxo existem apenas uma origem e um destino, qualquer solução pode ser decomposta em um conjunto de m caminhos correspondentes a cada escalonamento (sequência de tarefas com seus tempos ociosos) de cada máquina. A restrição (3.38) define que cada tarefa deve estar em exatamente um caminho, e por isso, processada em somente uma máquina.

A Figura 3.2 apresenta o escalonamento utilizando a representação de fluxo em redes e roteamento de veículos, para a instância apresentada na Figura 2.5 (a), onde cada caminho na rede representa o escalonamento em uma máquina.

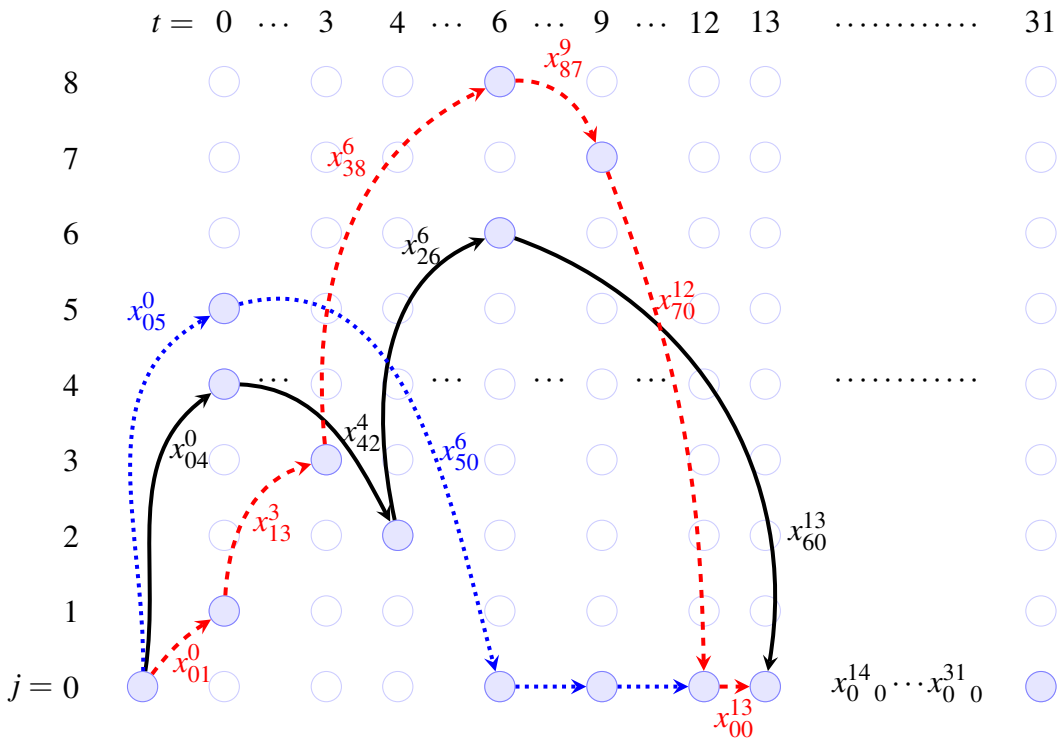


Figura 3.2 Representação do escalonamento no modelo de fluxo em redes.

Esta formulação foi adaptada para a minimização conjunta das penalidades de antecipação e atraso, sem tempo ocioso entre as tarefas, o tempo ocioso neste caso seria considerado somente após o fim da execução das tarefas nas máquinas.

A seguir são resumidas as principais características das formulações apresentadas. A Tabela 3.1 apresenta sete colunas: formulação matemática, número de restrições, número de variáveis, tipos de variáveis, ambiente de processamento, tipo de formulação e referência.

3.6 Resumo do capítulo

Neste capítulo foi apresentada uma análise das formulações matemáticas para os problemas de escalonamento com penalidades de antecipação e atraso. Dentre as formulações apresentadas, a formulação *PI-NetworkFlow* foi implementada (com a utilização da biblioteca *Concert Technology* do CPLEX para C/C++), com o objetivo de se obter os ótimos

para máquinas paralelas idênticas e assim servir de comparação com a estratégia algorítmica proposta nesta tese. Os resultados obtidos foram para 40, 50 e 100 tarefas. E uma versão relaxada da formulação PI-STO-ArcTime foi utiliza na estratégia híbrida proposta nesta tese, detalhada no Capítulo 4.

Tabela 3.1 Dimensão das formulações estudadas.

Formulação matemática	**Número de restrições**	**Número de variáveis**	**Tipos de variáveis**	**Ambiente de processamento**	**Tipo de formulação**	**Permite tempo ocioso**	**Referência**
PIM-TO	$O(n^2m)$	$O(n^2m)$	mono, bi e tri-indexado	mono e multi	PIM	✓	Arenales et al. (2007)
PI-TO-IT	$O(n+T)$	$O(nT)$	bi-indexado	monoprocessado	IP	✓	Dyer e Wolsey (1990) e Pessoa et al. (2010)
PIM-STO	$O(n^2)$	$O(n^2m)$	mono, bi e tri-indexado	mono e multi-processado	MIP	×	Tsai e Wang (2012)
PI-STO-IT	$O(n+T)$	$O(nT)$	bi-indexado	monoprocessado	PI	×	Tanaka, Fujikuma e Araki (2009)
PI-NetworkFlow	$O(n+T)$	$O(nT)$	bi-indexado	mono e multi-processado	PI	×	Amorim (2013)
PI-STO-ArcTime	$O(nT)$	$O(n^2T)$	tri-indexado	mono e multi-processado	IP	×	Pessoa et al. (2010)

Capítulo 4

Estratégia algorítmica proposta

Neste capítulo é apresentada uma estratégia híbrida, exato-heurística, para o problema de escalonamento com penalidades de antecipação e atraso. Nesta estratégia é utilizado o Algoritmo Genético com busca local (GLS) de Amorim (2013), melhorada nesta pesquisa com a inclusão de um critério de diversidade adaptativa (apresentada na seção 4.2.1.5), cruzamento seguido de busca local em toda a população e tratamento de escalonamento desbalanceado para funções não-regulares (seção 4.2.1.6). Na estratégia proposta é criado um conjunto de soluções ótimas locais, cada ótimo local é a melhor solução obtida de cada geração do GLS. Estes ótimos irão compor os arcos que são armazenados em uma tabela hash com endereçamento aberto com *hashing* duplo.

Ao final do passo heurístico, estes arcos serão utilizados para montar um modelo matemático relaxado (apresentado na seção 4.1) a ser resolvido pelo algoritmo *Branch-and-Cut* via CPLEX e assim verificar se é possível melhorar a solução obtida pelo GLS. O objetivo deste método híbrido é de obter resultados para instâncias de tamanho maior em máquinas paralelas, ainda não existentes na literatura. As seções a seguir detalham a estratégia híbrida proposta e os exemplos apresentados nestas seções utilizam a instância de teste de 8 tarefas apresentada na Figura 4.1.

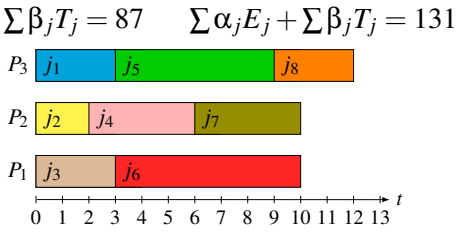
4.1 Formulação relaxada para o problema

A formulação arc-time proposta por Pessoa et al. (2010) é utilizada nesta pesquisa. Esta formulação assume um horizonte de tempo de 0 a T, onde as máquinas estão ociosas no instante 0 e devem estar ociosas novamente no instante máximo T. A variável binária x_{ij}^{t},

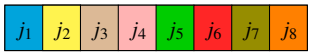
J	p_j	d_j	α_j	β_j	C_j	E_j	T_j	$\alpha_j E_j$	$\beta_j T_j$
j_1	3	3	3	5	3	0	0	0	0
j_2	2	6	4	5	2	4	0	16	0
j_3	3	6	8	8	3	3	0	24	0
j_4	4	4	8	10	6	0	2	0	12
j_5	6	6	6	4	9	0	3	0	27
j_6	7	10	7	3	10	0	0	0	0
j_7	3	11	4	2	10	1	0	4	0
j_8	3	8	5	8	12	0	4	0	48

$\sum \alpha_j E_j = 44$ e $\sum \beta_j T_j = 87$

(a)



(b)



(c)

Figura 4.1 Representação para uma (a) instância de 8 tarefas e (b) uma solução factível para o atraso e antecipação e atraso ponderados em máquinas paralelas idênticas juntamente com a sua respectiva (c) lista sequencial de tarefas.

$i \neq j$, indica que a tarefa i completa seu processamento e a tarefa j inicia no instante t, na mesma máquina. A variável x_{0j}^{t} indica que a tarefa j inicia no instante t na máquina que estava ociosa no intervalo de $t-1$ até t, em particular, a variável x_{0j}^{0} indica que a tarefa j inicia no instante 0 em alguma máquina. Já a variável x_{i0}^{t} indica que a tarefa i finaliza no instante t na máquina que permanecerá ociosa no intervalo de t a $t+1$, em particular, a variável x_{i0}^{T} indica que i termina no instante máximo de processamento. As variáveis x_{00}^{t} indicam que o número de máquinas que estavam ociosas no intervalo de $t-1$ a t, irão permanecer ociosas no intervalo de t a $t-1$.

Uma modificação foi feita nesta formulação com o objetivo de eliminar as variáveis binárias de tempo ocioso no término de uma sequência de tarefas em cada máquina (x_{00}^{t}). Assim, o conjunto de restrições (4.5) da formulação original foram removidas, destacadas em vermelho. A função objetivo $f_j(t+p_j)$ (4.1) foi modificada para a função de antecipação e atraso ponderados, $f_j(t+p_j) = \sum \alpha_j \times \max\{0, d_j - t + p_j\} + \sum \beta_j \times \max\{0, t + p_j - d_j\}$. O conjunto de restrições deste modelo define um fluxo em redes, que possui somente uma fonte e um só destino, a solução é decomposta em um conjunto de m caminhos, cada caminho corresponde a uma sequência de tarefas em uma máquina. O conjunto de restrições (4.2) define que cada tarefa deve estar em exatamente um dos caminhos (ou somente uma das máquinas disponíveis). Os conjuntos de restrições (4.3) a (4.7) definem um fluxo de m unidades (ou m máquinas) em um grafo acíclico

$G = (V, A)$.

$$\text{Minimizar} \sum_{i \in J_+} \sum_{j \in J \setminus \{i\}} \sum_{t=p_i}^{T-p_j} f_j(t+p_j) x_{ij}^t \tag{4.1}$$

$$\text{S. a.:} \sum_{i \in J_+ \setminus \{j\}} \sum_{t=p_i}^{T-p_j} x_{ij}^t = 1 \quad (\forall j \in J) \tag{4.2}$$

$$\sum_{j \in J_+ \setminus \{i\}, t-p_j \geq 0} x_{ji}^t - \sum_{j \in J_+ \setminus \{i\}, t+p_i+p_j \leq T} x_{ij}^{t+p_i} = 0 \tag{4.3}$$

$$(\forall i \in J; t = 0, ..., T - p_i) \tag{4.4}$$

$$\sum_{j \in J_+, \ t-p_j \geq 0} x_{j0}^t - \sum_{j \in J_+, \ t+p_j+1 \leq T} x_{0j}^{t+1} = 0 \quad (t = 0, ..., T-1) \tag{4.5}$$

$$\sum_{i \in J_+} x_{0j}^0 = m \tag{4.6}$$

$$x_{ij}^t \in Z_+ \quad (\forall i \in J_+; \forall j \in J_+ \setminus \{j\}; t = p_i, ..., T - p_j) \tag{4.7}$$

$$x_{00}^t \in Z_+ \quad (t = 0, ..., T-1) \tag{4.8}$$

4.2 Metaheurísticas GLS e ILS

Nas seções a seguir são apresentados os algoritmos genético e o algoritmo de busca local utilizados para o problema $P||\sum \alpha_j E_j + \sum \beta_j T_j$.

4.2.1 Algoritmo genético adaptativo com busca local intensiva (GLS)

O algoritmo genético deste trabalho utiliza a busca local de deFreitas et al. (2008). No algoritmo, o escalonamento multi-máquina é definido como uma lista sequencial de tarefas (Figura 4.1 (c)). Nesta pesquisa foi implementada uma estratégia populacional, desenvolvida com o algoritmo genético (GA) com busca local (LS), inicialmente proposto em Amorim (2013) e Amorim, deFreitas e Uchoa (2013), e melhorado a inclusão de um critério de diversidade adaptativa para a melhoria de novos indivíduos, tamanho de população e número de gerações.

Este algoritmo utiliza a representação da solução através de uma lista sequencial de tarefas proposta por deFreitas et al. (2008) (Figura 4.1 (c)), esta representação é tratada como um cromossomo onde os operadores genéticos trabalham (mutação e cruzamento baseado em posição). Cada gene representa uma tarefa e em cada iteração/geração uma nova população é criada, quando é necessário calcular a função de avaliação (*fitness*) ou

função objetivo de um cromossomo (solução), as tarefas são escalonadas nas máquinas (Figure 4.1 (b)) e o valor de antecipação e atraso são calculados sempre que necessário.

4.2.1.1 Função de avaliação (*fitness*)

Para o algoritmo genético proposto, uma avaliação de um escalonamento factível é representada pela seguinte função de avaliação (*fitness*): $f : \pi \rightarrow Z_+$, com a propriedade $f(\pi_i) \leq f(\pi_j)$ se π_i é melhor que π_j, onde π representa uma permutação (arranjo) de tarefas em uma lista sequencial (Figure 4.1 (c)). Com objetivo que se obter o custo do escalonamento, cada tarefa em π é distribuída em cada máquina disponível. A função de avaliação é calculada de acordo com a seguinte fórmula: $f(\pi_i) = \sum \alpha_j E_j + \sum \beta_j T_j$ (WET).

4.2.1.2 Inicialização

O algoritmo começa o seu processamento com k soluções aleatórias. Estas soluções são obtidas por n perturbações: $\pi_1, \pi_2, ..., \pi_n$. Cada solução possui um valor de função de avaliação $f_i = f(\pi_i)$ correspondente ao escalonamento em máquinas paralelas idênticas.

4.2.1.3 Seleção por torneio

A seleção por torneio é utilizada pelos operadores genéticos onde, a ideia principal é selecionar um indivíduo de um conjunto pequeno de indivíduos. Neste trabalho foi implementado a seleção por torneio considerando toda a população ao invés de somente um subconjunto de soluções. Em cada torneio, as soluções são comparadas entre si, sempre aos pares (π_i, π_j), onde i e j são escolhidos aleatoriamente na população. Assim, a solução ganhadora do torneio é a melhor solução escolhida para ser utilizada por um determinado operador genético.

4.2.1.4 Cruzamento

É apresentado neste trabalho uma adaptação do cruzamento baseado em posição de Liu, Abdelrahman e Ramaswamy (2005) que gera duas novas soluções na operação de cruzamento. Na adaptação deste trabalho, somente uma solução é gerada pela operação de cruzamento. Na Figura 4.2 é apresentado um exemplo de cruzamento, onde dois pais são selecionados (**Indivíduo 1** e **Indivíduo 2**) com o objetivo de gerar uma solução descendente. Para isso, o Indivíduo 1 possui $0,4 * n$ tarefas fixas ($0.4 * n$ posições aleatoriamente

escolhidas - valor obtido através de análise empírica dos experimentos computacionais). Essas tarefas permanecem fixas na nova solução descendente e as demais posições são preenchidas com as outras tarefas do indivíduo 2, respeitando a posição relativa das tarefas e sempre verificando se há repetição de tarefas no indivíduo gerado (para não gerar soluções infactíveis).

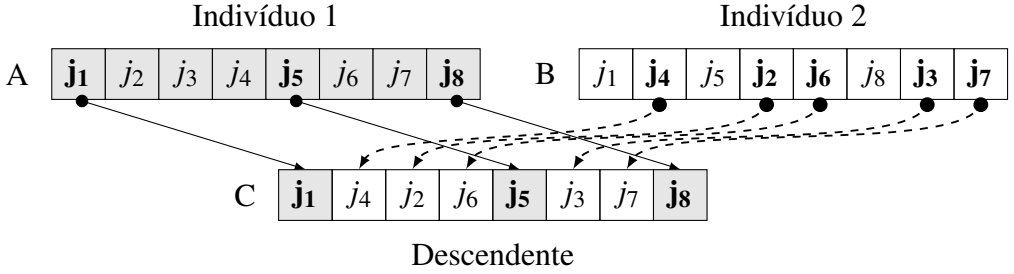


Figura 4.2 Cruzamento baseado em posição utilizado na geração de novas soluções.

4.2.1.5 Mutação

Na mutação, os genes (tarefas) do cromossomo (solução) são aleatoriamente escolhidos para serem trocados de posição e compor uma nova solução completamente diferente (perturbação). Cada mutação é repetida 3 vezes em cada cromossomo.

No algoritmo genético, a mutação é aplicada se a diversidade da população cair, ou seja, se os cromossomos da população tiverem mais de 80% dos seus genes repetidos entre si. Pois a ideia é ter uma população com a maior quantidade de cromossomos com valores diferentes de função de avaliação. A diversidade é obtida através da seguinte fórmula:

$$f((a,j),(b,\gamma)) = \begin{cases} 1, \text{ se } \pi_j(a) = \pi_\gamma(b) \\ 0, \text{ caso contrário} \end{cases} \tag{4.9}$$

Sendo $\pi_j(a)$ representando uma tarefa na posição j do indivíduo a e, $\pi_\gamma(b)$ representando uma tarefa na posição γ do indivíduo b.

$$d(a,b) = \frac{\sum_{j=1}^{n} f((a,j),(b,j))}{n} \tag{4.10}$$

$$Diversidade = \sum_{(a,b)\in\prod\times\prod} \frac{d(a,b)}{(|\prod|(|\prod|-1))/2} \tag{4.11}$$

onde $|\prod|$ representa o tamanho da população.

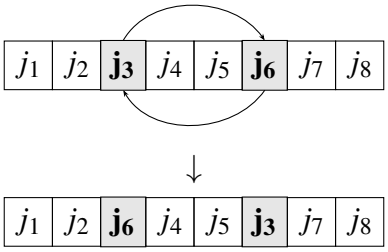


Figura 4.3 Operador genético de mutação.

A meta-heurística funciona da seguinte maneira: Dada uma população inicial de $\prod$ soluções, o algoritmo gera o dobro de soluções ($\prod^*$) em relação à população corrente (pois somente a metade dessas soluções serão copiadas para a população corrente - os melhores cromossomos que serão utilizados na próxima iteração do algoritmo), através da aplicação dos operadores genéticos de cruzamento e mutação, seguidos de busca local. As soluções geradas são mantidas em uma população auxiliar. Assim, a cada iteração ou geração, a população auxiliar é ordenada da melhor para a pior solução (valor de função objetivo), o que assegura o elitismo da população.

As soluções ordenadas na população auxiliar são copiadas para a população corrente, com um tratamento de soluções repetidas, ou seja, caso sejam encontrados dois escalonamentos com o mesmo valor de função objetivo, somente um deles é copiado para a população corrente. Desta forma, tem-se uma nova população e com indivíduos distintos entre si a cada geração do algoritmo (ver Figura 4.4).

Para a operação de cruzamento, dois indivíduos da população corrente são selecionados e assim é gerado um novo descendente. Dos pais escolhidos, somente alguns genes são mantidos intactos no descendente (obtidos do indivíduo 1) e os genes faltantes são preenchidos respeitando a posição relativa dos genes do indivíduo 2 (ver Figura 4.2), o cruzamento é aplicado em toda a população em cada geração (ver Figura 4.5). Já na operação de mutação, o critério de diversidade é de 80%, ou seja, a tolerância de genes repetidos dos indivíduos da população é de 80% (ver Seção 4.2.1.5). Assim, caso esta tolerância ultrapasse os 80%, toda a população é modificada (3 mudanças aleatórias de genes em cada cromossomo). O objetivo é fugir de ótimos locais e assim melhor explorar o espaço de busca. O Algoritmo 1 resume o método GLS.

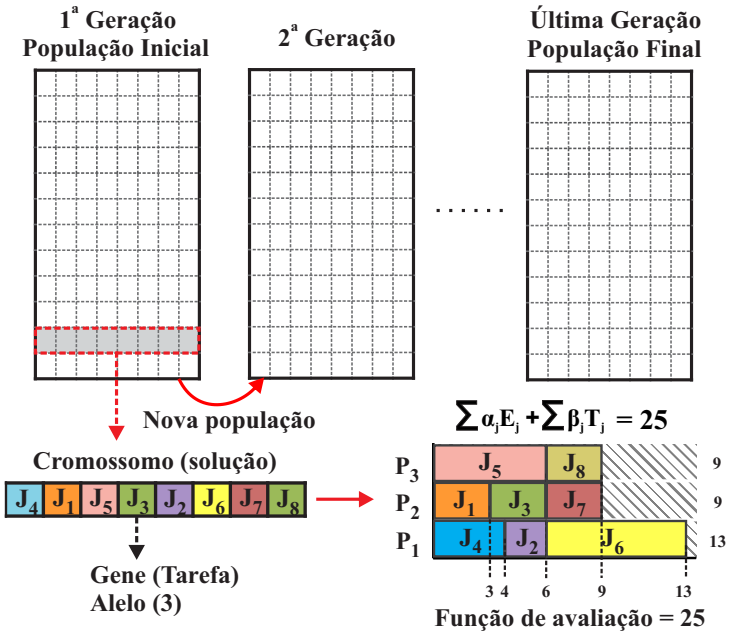


Figura 4.4 Estratégia utilizada no Algoritmo Genético com busca local, onde a cada iteração uma nova população é gerada.

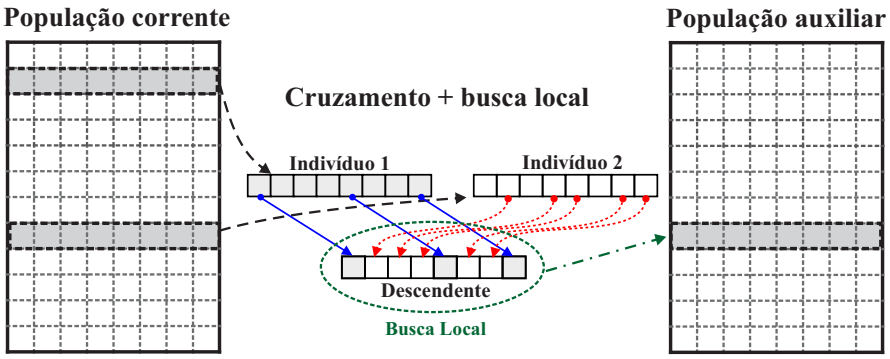


Figura 4.5 Esquema da operação de cruzamento seguida de busca local.

4.2.1.6 Assegurando escalonamento ótimo

Nesta pesquisa, a estratégia de alocar as tarefas nas máquinas disponíveis mais cedo, inicialmente proposta por deFreitas et al. (2008), não funcionou bem quando se tem funções objetivo não-regulares, como o caso do escalonamento com penalidades de antecipação e atraso em máquinas paralelas idênticas. Como a estratégia algorítmica GLS também utiliza a representação da solução como uma lista sequencial para múltiplas máquinas de deFreitas et al. (2008) (ver Figura 4.1), observou-se que esta estratégia pode gerar escalonamentos desbalanceados para algumas instâncias do problema investigado nesta pesquisa. Pois este tipo de alocação objetiva a minimização do tempo de completude - o que pode reduzir bastante o atraso de uma tarefa mas, por outro lado, pode aumentar

Algoritmo 1 (GLS) Algoritmo genético com busca local utilizando a ideia de lista sequencial de deFreitas et al. (2008) e operadores genéticos de Liu, Abdelrahman e Ramaswamy (2005).

entrada $\prod$ - uma população (conjunto de soluções), $\prod^*$ - uma população auxiliar, n - número de tarefas e N - número de gerações/iterações.

1: **função** GLS($\prod, \prod^*, n, N$)
2: $\prod \longleftarrow$ um conjunto de permutações aleatórias de tarefas;
3: Calcula a função de avaliação de cada permutação;
4: $i \longleftarrow 1$;
5: **enquanto** $i <= N$ **faça**
6: $j \longleftarrow 1$;
7: **enquanto** $j < 2 * \prod$ **faça**
8: Seleciona as soluções π e θ de $\prod$ utilizando a seleção por torneio (*Tournament Selection*);
9: $\pi^* \longleftarrow$ cruzamento baseado em posição (*Position-based Crossover*) utilizando π e θ;
10: Aplica a busca local em π^*;
11: $\prod^* \longleftarrow \pi^*$;
12: $j \longleftarrow j + 1$;
13: **fim enquanto**
14: Ordena $\prod^*$ a partir da melhor para a pior solução;
15: $\prod \longleftarrow$ metade das soluções de $\prod^*$; ▷ Elitismo
16: **se** ***Diversidade*** > 0.8 **então**
17: $\prod \longleftarrow$ mutação em todos os indivíduos seguida de busca local;
18: **fim se**
19: $i \longleftarrow i + 1$;
20: $\prod^{**} \longleftarrow$ a melhor solução da geração corrente ▷ Conjunto de soluções ótimas locais
21: **fim enquanto**
retorne $\prod^{**}$.
22: **fim função**

bastante a antecipação desta mesma tarefa.

Com a finalidade de resolver este comportamento foram adicionadas, no algoritmo genético com busca local de Amorim (2013), $m-1$ tarefas fictícias (*dummy jobs*) no escalonamento, onde m é o número de máquinas. Estas tarefas fictícias possuem valores de antecipação e atraso nulos e seus respectivos tempos de processamento são iguais a soma de todos os tempos de processamento de todas as tarefas da instância a ser testada (tarefas j_0 destacadas na cor branca nas Figuras 4.6, 4.7, 4.8, 4.9 e 4.10). Estas tarefas têm a função de bloquear a máquina em que ela for alocada (nenhuma outra tarefa será incluída nesta máquina após a inclusão da tarefa fictícia), permitindo assim a geração de escalonamentos balanceados.

A decisão de se utilizar $m-1$ tarefas fictícias foi com o objetivo de se ter pelo menos uma máquina que deverá receber tarefas sem restrições (para o caso de não se ter todas as máquinas bloqueadas e tarefas para serem escalonadas e não ter máquina disponível). Assim, o algoritmo assegura a distribuição das tarefas nas máquinas com o menor custo possível.

Assim, verificou-se que, com a inclusão de $m-1$ tarefas fictícias, para se chegar na solução ótima, dois tipos de escalonamento foram obtidos. Descritos a seguir:

a) Dada uma instância para o JIT_sched, existe um escalonamento ótimo que usa todas as m máquinas:

    1. Estes casos são apresentados nos exemplos das Figuras 4.6 e 4.9, para uma instância da literatura testada. Nas Figuras 4.6 (c) e 4.9 (b) são apresentas as sequências únicas que geraram os escalonamentos das Figuras 4.6 (b) e Figura 4.9 (a), respectivamente, através da distribuição das tarefas nas máquinas disponíveis mais cedo com a tarefas fictícias. Onde é possível notar que as tarefas fictícias são tratadas como tarefas normais na instância e, quando o valor de função objetivo é avaliado, elas não são consideradas. Além do mais, elas também têm a função de bloquear a máquina em que for alocada, ou seja, nenhuma outra tarefa é escalonada após a sua inclusão na máquina.

b) Dada uma instância para o JIT_sched, existe um escalonamento ótimo que utiliza apenas um subconjunto de máquinas, até mesmo apenas uma delas:

    1. Há casos cuja solução ótima utiliza apenas um subconjunto de máquinas, pois existem tarefas que se forem alocadas mais cedo, podem ter as suas penalidades de antecipação e atraso aumentadas. Sendo necessário a alocação de cada tarefa o mais próximo possível de sua respectiva data de término sugerida (ver Figura 4.7 (b) e 4.10 (a), para uma instância da literatura).
    2. Há casos excepcionais em que somente uma máquina é utilizada na solução ótima, sem a necessidade da utilização das outras máquinas disponíveis. Pois, se todas as tarefas fossem distribuídas em todas as máquinas, algumas poderiam ter suas penalidades de antecipação e atraso aumentadas. Pois parte delas não terminariam o seu processamento nas suas respectivas datas de término sugeridas (ver Figura 4.8 (b)).

Dessa forma, a inclusão das tarefas fictícias assegura que a solução do escalonamento tenha tarefas que terminam as suas execuções o mais próximo possível de suas datas de término sugeridas, reduzindo assim o valor da função objetivo.

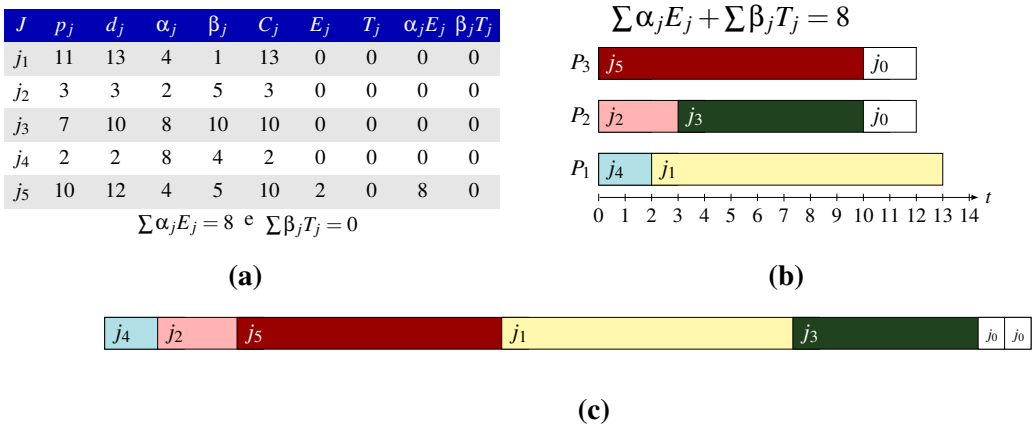


Figura 4.6 Instância de 5 tarefas (a), escalonamento utilizando todas as máquinas com solução igual a 8 (b) e a representação desta solução usando lista sequencial de tarefas (c).

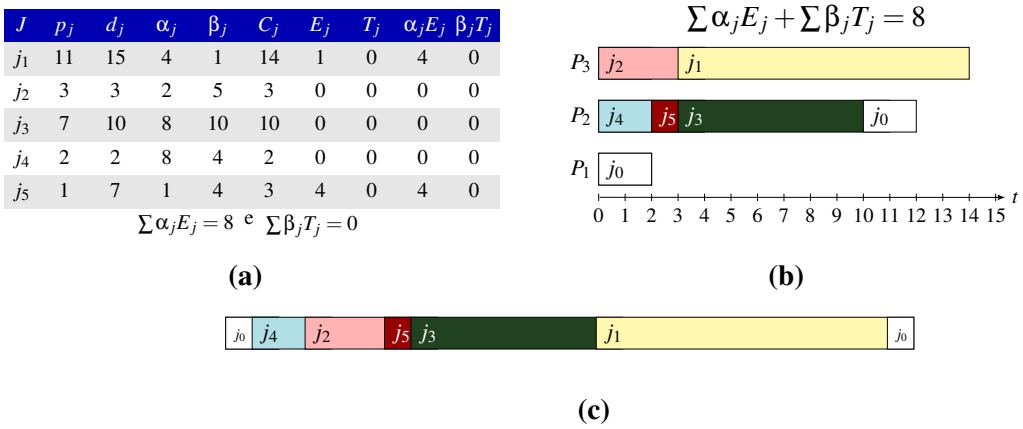


Figura 4.7 Instância de 5 tarefas (a), escalonamento utilizando somente um subconjunto de máquinas com solução igual a 8 (b) e a representação desta solução usando lista sequencial de tarefas (c).

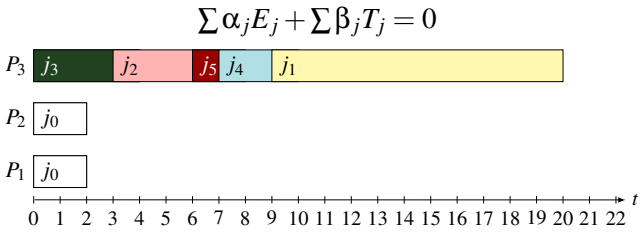
4.2.2 Algoritmo de busca local iterada

A busca local iterada funciona da seguinte forma (DEFREITAS et al., 2008): dada uma permutação π de tarefas, o algoritmo realiza uma busca local na vizinhança desta solução definida por movimentos generalizados de troca de pares (*Generalized Pairwise Interchange - GPI*).

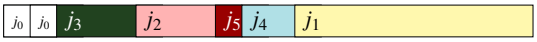
J	p_j	d_j	α_j	β_j	C_j	E_j	T_j	$\alpha_j E_j$	$\beta_j T_j$
j_1	11	20	4	1	20	0	0	0	0
j_2	3	6	1	2	6	0	0	0	0
j_3	3	3	5	2	3	0	0	0	0
j_4	2	9	3	1	2	0	0	0	0
j_5	1	7	1	4	7	0	0	0	0

$\sum \alpha_j E_j = 8$ e $\sum \beta_j T_j = 0$

(a)



(b)



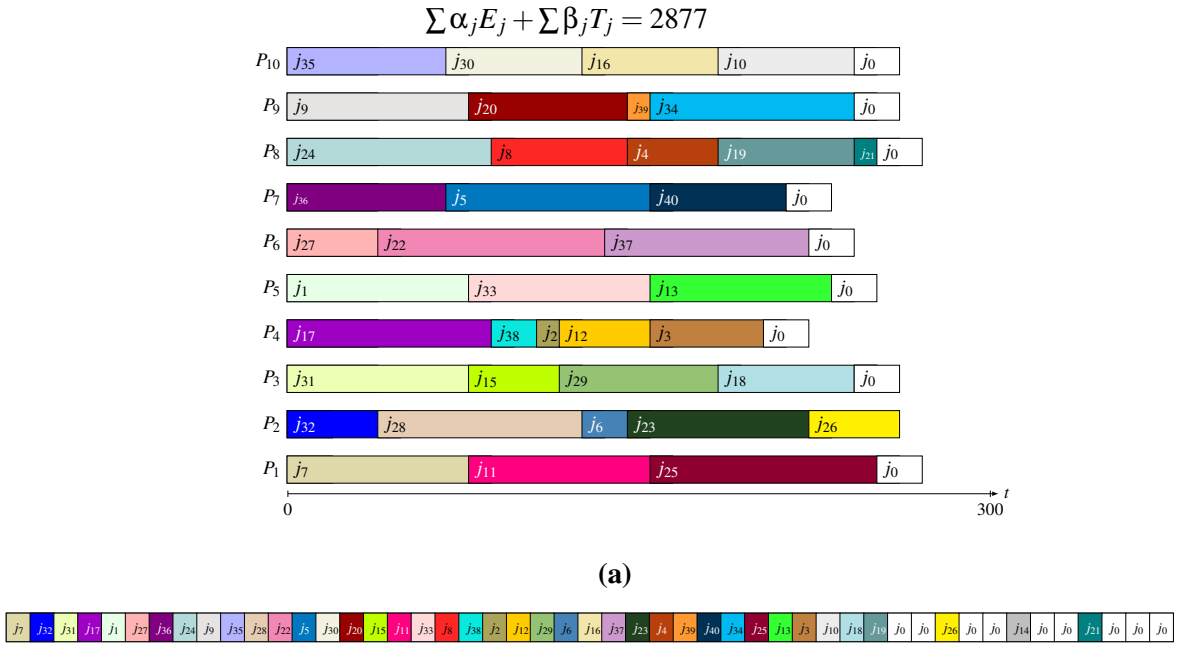
(c)

Figura 4.8 Instância de 5 tarefas (a), escalonamento utilizando somente uma máquina (sem a necessidade de utilizar as outras máquinas disponíveis) com solução igual a 0 (b) e a representação desta solução usando lista sequencial de tarefas (c).

Os movimentos GPI são definidos por dois tipos de movimentos, o primeiro deles é o movimento de troca (*swap*), onde a posição das duas tarefas i e j (não necessariamente adjacentes) na sequência é trocada em π, Figura 4.11 (a); e o segundo deles é o movimento de remoção (*move*), onde a tarefa i é removida de sua posição original e inserida na posição da tarefa j em π, a subsequência de tarefas entre as tarefas i e j são reposicionadas para dar espaço a tarefa i na sua nova posição, Figura 4.11 (b). Cada busca local é realizada até que não seja mais possível melhorar a solução corrente. Uma busca completa na vizinhança custa $O(n^2)$ movimentos GPI e o custo de transformar uma lista sequencial no escalonamento para máquinas paralelas é de $O(nlogm)$.

Seja uma solução inicial s para o problema $1||\sum \alpha_j E_j + \sum \beta_j T_j$, obtida por uma permutação inicial π de tarefas, $\pi = (\pi_1, \pi_2, ..., \pi_n)$, onde todas as máquinas estão ociosas no instante 0. Assim, a partir desta solução inicial π, uma solução factível é gerada através do sequenciamento de tarefas da menor para a maior data de término sugerida (*earliest due date* - EDD) e é armazenada em π^*. Na primeira iteração, é gerada uma permutação aleatória em π.

A busca na vizinhança é baseada em movimentos GPI e a busca na vizinhança da solução π termina quando o primeiro movimento de melhora é achado - neste caso, π é atualizado e uma nova busca é iniciada. Isto é feito até que uma busca seja completada sem melhorias. Se a solução achada na busca local for melhor que a melhor solução global atual, então a mesma será armazenada como π^*. Finalmente, k movimentos GPI escolhidos aleatoriamente são aplicados em uma tentativa de sair de regiões de ótimo local



(a)

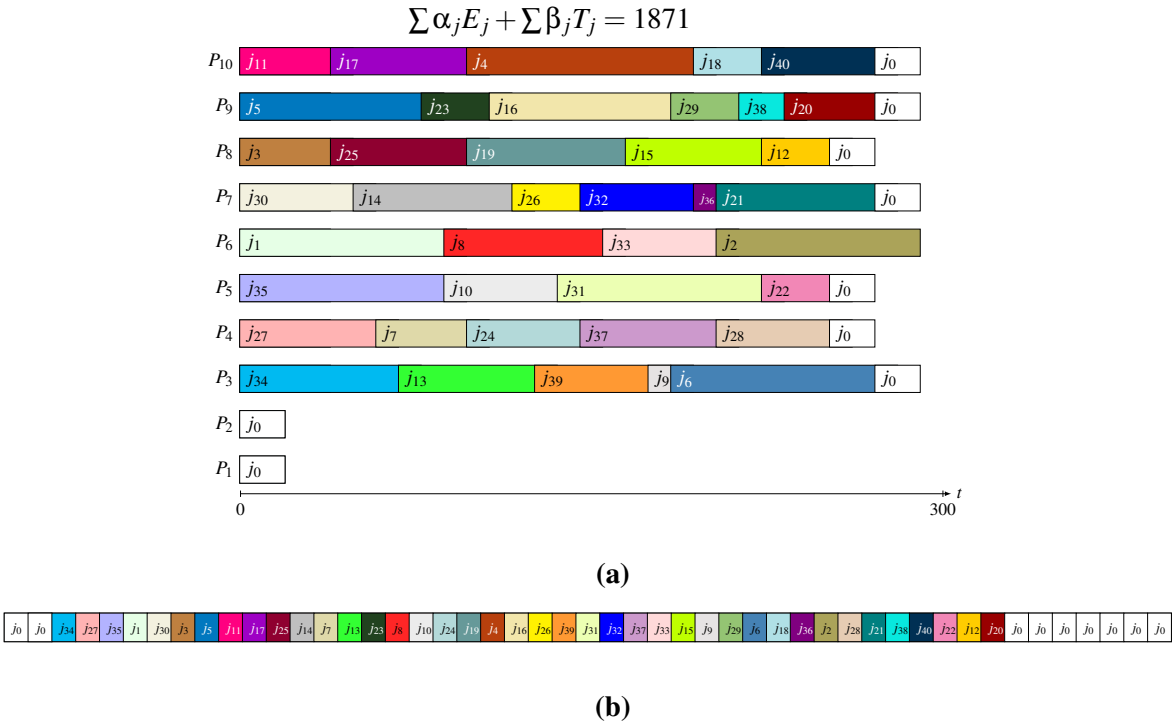
(b)

Figura 4.9 Escalonamento utilizando somente um subconjunto de máquinas: (a) Solução para a instância 51 de 40 tarefas com função objetivo no valor 2877 e a (b) representação desta solução usando lista sequencial de tarefas.

ruins. A cada r iterações, uma nova permutação totalmente aleatória substitui a solução gerada na iteração anterior. Uma busca completa na vizinhança requer um tempo de $O(n^2)$ movimentos, e a avaliação de cada movimento requer a construção do escalonamento usando a regra de despacho, o que requer tempo $O(n \log m)$ em uma implementação com filas de prioridade.

A busca local conta com a estratégia da melhor solução encontrada (*best improvement*), ou seja, a mudança é aceita somente quando a nova solução é melhor que a solução corrente. Entretanto, soluções ótimas locais geralmente possuem muitas soluções vizinhas com o mesmo custo. Dessa forma, para impedir uma busca prematura, ou seja, para que o algoritmo continue efetuando a busca por soluções melhores, o algoritmo possui um critério de desempate de soluções que define, dentre outras soluções de mesmo valor de função objetivo, qual deve ser considerada a melhor solução, obtido a partir da seguinte fórmula: $b(\pi) = \sum_{j=1}^{n} d_{\pi_j} x(n-j+1)$, onde d_{π_j} é a data de término sugerida das tarefas, n é o número de tarefas e j é o número da tarefa.

Este critério parte da ideia de que as tarefas sequenciadas da menor para a maior data de término sugerida (*earliest due date* - EDD) representam melhores soluções, pois



(a)

(b)

Figura 4.10 Escalonamento utilizando somente um subconjunto de máquinas: (a) Solução para a instância 101 de 40 tarefas com função objetivo no valor 1871 e a (b) representação desta solução usando lista sequencial de tarefas.

inicialmente esta estratégia foi proposta para o atraso ponderado de tarefas. Quando há algum empate entre máquinas, ou seja, quando se tem duas máquinas ociosas, é escolhida a máquina com o menor índice.

O processo de busca local é repetido durante x iterações, onde $x = k \cdot n \cdot m$ e k é uma dada constante, n é o número de tarefas e m é o número de máquinas. Quando o número da iteração corrente é um múltiplo de uma dada constante r, uma nova sequência randômica é gerada onde a busca local será aplicada novamente. Finalmente, k movimentos GPI escolhidos aleatoriamente são aplicados em uma tentativa de explorar outras sub-regiões do espaço de busca e fugir de ótimos locais. O Algoritmo 2 resume os passos da busca local iterada de deFreitas et al. (2008).

Por outro lado, a melhor estratégia exata existente na literatura foi proposta por Pessoa et al. (2010), um algoritmo Branch-and-Cut-and-Price para o problema $P||\sum w_j T_j$, que é baseado na formulação matemática apresentada na seção 4.1, também proposta por Pessoa et al. (2010), adotada na estratégia híbrida desta pesquisa. Esta formulação assume a execução de tarefas no horizonte do tempo de 0 a T, onde as máquinas estão ociosas no instante 0 e devem permanecer ociosas novamente no instante T.

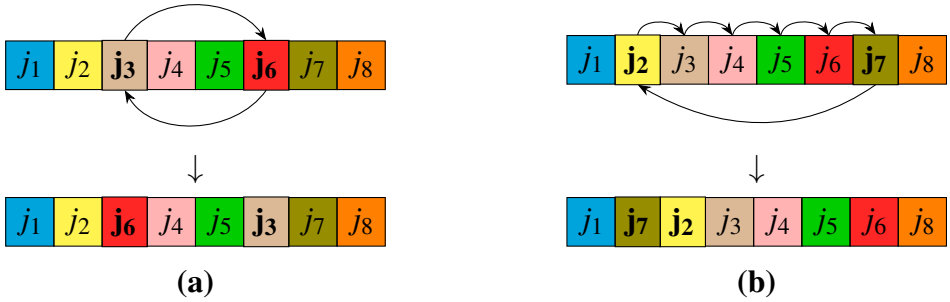


Figura 4.11 Movimentos generalizados de troca de pares: (a) troca (*swap*) e (b) remoção (*move*).

Algoritmo 2 (ILS) Algoritmo de busca local iterada de deFreitas et al. (2008).

entrada π - solução corrente, π^* - melhor solução global, n - número de tarefas e N - número de iterações.

1: **função** ILS(π, π^*, n, N)
2: $i \longleftarrow 1$;
3: $\pi^* \leftarrow$ uma permutação gerada pelo sequenciamento de tarefas da menor para a maior data de término sugerida;
4: **enquanto** $i < N$ **faça**
5: **se** i é um múltiplo de r **então**
6: $\pi \leftarrow$ uma permutação aleatória;
7: **fim se**
8: $\pi \longleftarrow$ movimentos GPI até que não seja possível melhorar a solução;
9: **se** $w(\pi) < w(\pi^*)$ **então**
10: $\pi^* \leftarrow \pi$;
11: **fim se**
12: Aplique k movimentos de troca escolhidos aleatoriamente em π;
13: $i \leftarrow i + 1$.
14: **fim enquanto**
retorne π^*
15: **fim função**

4.3 Algoritmo híbrido exato-heurístico (MathGLS-IP)

A estratégia algorítmica híbrida desta pesquisa para o problema $P||\sum \alpha_j E_j + \sum \beta_j T_j$ é baseada em dois passos: um passo meta-heurístico (utilizando o algoritmo genético com busca local (GLS)); e o passo exato (utilizando um resolvedor de programação matemática - CPLEX). A ideia principal é utilizar a melhor solução obtida de cada geração do GLS e prover um conjunto de arcos para ser utilizado na formulação matemática (apresentada na Seção 4.1). Este conjunto de arcos irá compor um modelo matemático relaxado, ou seja, esta formulação só terá os arcos dos ótimos locais obtidos (não terá todas as combinações de arcos possíveis e os arcos referentes a tempo ocioso no término de cada

sequência de tarefas). A ideia é obter uma melhor convergência e assim atingir a mesma qualidade de soluções geradas pelo GLS, em um tempo de execução curto, e também verificar se é possível obter melhores resultados em relação ao GLS (conseguiu-se provar pelos experimentos computacionais que é possível obter melhores resultados).

Assim, no primeiro passo, o Algoritmo Genético de (AMORIM; DEFREITAS; UCHOA, 2013) (GLS) é aplicado, que resolve uma instância do problema, e fornece um conjunto de ótimos locais obtidos de cada geração do GLS. Então, os arcos são gerados a partir de cada solução do conjunto e são armazenados em uma tabela hash (endereçamento aberto com *hashing* duplo ou re-hash), e ordenado de acordo com o valor das suas respectivas funções objetivo (da melhor para a pior solução). Finalmente, após a ordenação da tabela hash, os k melhores arcos são incluídos no passo exato (Figura 4.12 (a)), onde o algoritmo Branch-and-Cut do CPLEX é utilizado para a solução do modelo gerado. O Algoritmo 3 resume o método híbrido implementado.

Algoritmo 3 (MetodoHirido-Escalonamento) Algoritmo híbrido para resolver os arcos selecionados da formulação de programação inteira arc-time relaxada.

entrada $\prod$ - uma população (conjunto de soluções), $\prod^*$ - uma população auxiliar, $\prod^{**}$ - conjunto de soluções ótimas locais, n - número de tarefas, N - número de gerações/iterações, $nSols$ - número de soluções armazenadas do GLS, $HashArcos$ - tabela hash com os arcos.

1: **função** METODOHIRIDO-ESCALONAMENTO($\prod^{**}, nSols$)
2: $\prod^{**} \longleftarrow$ GLS($\prod, \prod^*, n, N$);
3: Ordena($\prod^{**}$); ▷ Ordena o conjunto de soluções do GLS
4: $i \longleftarrow 1$;
5: **enquanto** $i < nSols$ **faça**
6: **se** EstaCheio($HashArcos$) **então** ▷ Verifica se a tabela hash está cheia
7: interrompa;
8: **fim se**
9: $HashArcos \longleftarrow HashArcos \cup \prod^{**}[i]$ ▷ Constrói os arcos de cada solução e adiciona na tabela hash
10: $i \longleftarrow i + 1$;
11: **fim enquanto**
12: $LP \longleftarrow$ ControiIP-ArcTime($HashArcos$); ▷ Constrói a formulação utilizando os arcos da tabela hash (Figure 4.12 (b))
13: $CustoIP \longleftarrow$ Solucione(LP); ▷ Resolve com o algoritmo Branch-and-Cut do CPLEX;
retorne $CustoIP$
14: **fim função**

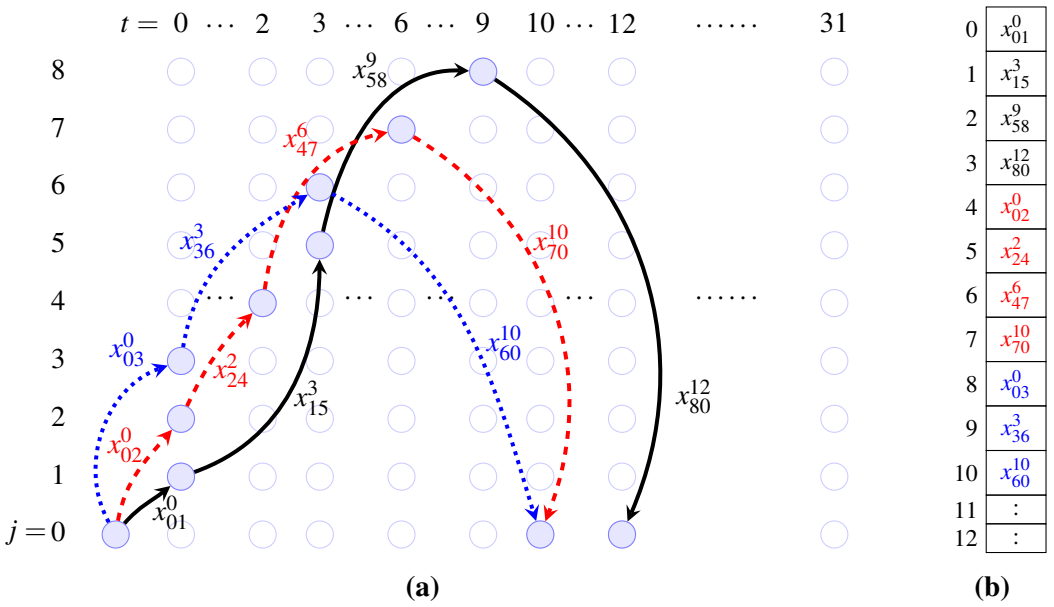


Figura 4.12 Estratégia híbrida proposta: (a) representação da solução para máquinas paralelas idênticas da formulação arc-time para o escalonamento da Figura 4.1 e (b) arcos armazenados desta solução em uma tabela hash (que guarda um conjunto de arcos obtidos dos ótimos locais - a melhor solução de cada geração do algoritmo).

4.3.1 Algoritmo Branch-and-Cut do CPLEX

O CPLEX utiliza o algoritmo *branch-and-cut* para solucionar problemas de programação linear mista. O procedimento de *branch-and-cut* lida com vários nós de uma árvore de busca, onde cada nó representa um subproblema a ser processado, ou seja, a ser checado a sua integralidade e se esse nó pode ou não ser podado. O CPLEX processa os nós da árvore de busca até que não se tenham mais nós ativos disponíveis ou se algum limite foi atingido.

Primeiramente, a árvore de *branch-and-cut* é inicializada a fim de que se tenha um nó raiz. O nó raiz da árvore representa o problema original, ignorando todas as restrições de integralidade. Os cortes são aplicados ao nó raiz com a finalidade de reduzir o espaço de busca. Se uma solução inteira for encontrada, ela será factível para o problema original e será considerada a melhor solução atualmente encontrada.

Um ramo (*branch*) é a criação de dois novos nós de um nó pai. Normalmente, um ramo ocorre quando os limites de uma única variável são modificados, esses novos limites são herdados pelos seus descendentes. Por exemplo, se um ramo ocorre em uma variável binária, ou seja, variável que possui o limite inferior 0 (zero) e um limite superior 1 (um), então o resultado serão dois nós, um nó com o limite superior 0 (o limite inferior do ramo

descendente dessa variável pode assumir somente o valor 0), e o outro nó com o limite superior modificado 1 (o limite superior do ramo descendente deve possuir somente o valor 1). Os dois novos nós da árvore terão assim dois novos domínios distintos. Uma restrição de corte é adicionada ao modelo.

O objetivo de se adicionar cortes ao modelo matemático é limitar o tamanho do espaço de soluções contínuas (obtidas pela relaxação linear do método - onde é removida a restrição de integralidade do problema), mas sem eliminar soluções inteiras. Onde deseja-se reduzir o número de ramos necessários para solucionar o problema.

Quando processa um nó, o CPLEX começa o seu processamento resolvendo a relaxação contínua do subproblema (problema sem as restrições de integralidade). O CPLEX pode adicionar um ou vários cortes no subproblema até solucioná-lo. Neste processo de adição de cortes, o nó é removido (podado) da árvore quando o subproblema se torna infactível. Caso contrário, o CPLEX verifica se a solução do nó satisfaz as restrições de integralidade, caso essa solução seja factível, e se o seu valor de Função Objetivo é melhor que a solução atual incumbente, a solução do nó é utilizada como nova solução incumbente. Senão, a ramificação irá ocorrer, mais primeiramente um método heurístico pode ser utilizado neste nó para verificar se uma nova solução incumbente pode ser inferida a partir deste nó. A ramificação, quando ocorre, é realizada em uma variável onde o valor da solução viola as restrições de integralidade. Esta prática resulta em dois nós que são adicionados na árvore para que seja processados posteriormente.

O algoritmo Branch-and-Cut do CPLEX foi utilizado para resolver a formulação, apresentada na seção 4.1 onde, dada uma instância para o problema $P||\sum \alpha_j E_j + \beta_j T_j$, ela é convertida em um formato de programação inteira para ser resolvido pelo CPLEX. Um algoritmo foi desenvolvido em C/C++ para converter as instâncias de teste do problema para este formato, `*.lp`. Este algoritmo com a ferramenta de programação matemática UFFLP[1]. O UFFLP é uma DLL (*Dynamic Link Library*) para Windows que pode ser utilizada com as seguintes linguagens: *Visual Basic for Applications* (VBA) e C/C++.

[1]Ferramenta de código aberto disponível em: <http://www.logis.uff.br/$\sim$artur/UFFLP/>

4.4 Resumo do capítulo

Neste capítulo foi apresentada uma estratégia híbrida exato-heurística, baseada na formulação de programação inteira arc-time e uma estratégia evolucionária com busca local. O objetivo deste método é solucionar instâncias de tamanho maior do problema de escalonamento com penalidades de antecipação e atraso, com tarefas independentes e tempos de processamento arbitrários. Os arcos são selecionados a partir de soluções ótimas locais obtidas de um algoritmo genético com busca local (GLS), que são utilizadas na construção de uma formulação matemática relaxada para o problema, arc-time, para produzir melhores resultados que os obtidos pelo GLS. Esta formulação relaxada, incluindo somente os arcos dos ótimos locais, é então resolvida pelo CPLEX.

Capítulo 5

Experimentos computacionais

Neste capítulo são apresentados os experimentos computacionais da pesquisa sobre problemas de escalonamento com penalidades de antecipação e atraso. Onde foram considerados os ambientes mono e multi-processado e as instâncias propostas por Tanaka (2012b), que são baseadas nas instâncias da *OR-Library* para ambiente monoprocessado, sendo aqui adaptadas para ambiente multiprocessado também. Um levantamento das formulações da literatura se fez necessário para a verificação de um modelo que se adequasse ao problema investigado nesta pesquisa. A formulação arc-time (ver seção 3.5) foi a formulação que se adequava ao problema, mas ela tem muitas variáveis e restrições, tornando-se impraticável, mesmo para instâncias de tamanho menor. Por isso, foi utilizada uma versão relaxada desta formulação neste trabalho (ver seção 4.1). Assim, foi proposto nesta pesquisa um modelo baseado no fluxo em redes, que se adaptou perfeitamente ao problema sem tempo ocioso entre as tarefas (*IP-NetworkFlow*). Os experimentos com o método híbrido apresentado serão comparados com a literatura e com os resultados da formulação *IP-NetworkFlow*, implementado no CPLEX.

5.1 Ambiente computacional

Os experimentos computacionais deste trabalho foram realizados em um computador com Windows 10 Pro, utilizando a linguagem de programação C/C++, ferramenta CPLEX e biblioteca UFFLP, para a resolução de problemas de programação inteira. A Tabela 5.1 apresenta o ambiente computacional utilizado. Os limites de tempo foram introduzidos pois haviam algumas instâncias que demoravam bastante para serem resolvidas. Assim,

os limites foram estimados com base em análise empírica dos experimentos, considerando amostras de instâncias de teste difíceis de serem resolvidas.

Tabela 5.1 Ambiente computacional utilizado nos experimentos

Sistema Operacional	Windows 10 Pro
Processador	Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz
Memoria (RAM)	16 GB
Tipo de sistema	Sistema operacional de 64-bits
Softwares instalados	Visual Studio 2013 Professional, UFFLP e IBM/ILOG CPLEX
Linguagem de programação	C/C++
Limite de tempo	40, 50 e 100 tarefas - 1 hora (3600 s) 200 - 2 horas (7200 s) 300 - 3 horas (10800 s) 500 - 6 horas (21600 s)

5.2 Instâncias de teste

As instâncias utilizadas para teste são as instâncias geradas por Tanaka (2012a) para o problema de escalonamento com antecipação e atraso ponderados em ambiente monoprocessado sem tempo ocioso, onde, as instâncias de 40, 50 e 100 tarefas foram geradas a partir das instâncias da *OR-Library* (CONGRAM; POTTS, 1998) que contou apenas com a adição das penalidades de antecipação. As instâncias de 150, 200, 250 e 300 tarefas foram geradas de um modo similar à geração das instâncias da *OR-Library*. Para os experimentos em ambiente de máquinas paralelas, as instâncias de Tanaka (2012a) receberam uma modificação em suas datas de término sugeridas, ou seja, as datas de término das instâncias foram divididas pela quantidade de máquinas, pois, caso contrário, com uma data de término sugerida relativamente grande (compatível com ambiente monoprocessado) o valor de antecipação e atraso ponderados poderiam ser nulos[1]. Os ótimos disponibilizados por Tanaka (2012a) foram obtidos através da execução de um algoritmo exato baseado em programação dinâmica.

As instâncias da *OR-Library* foram geradas aleatoriamente de acordo com a configuração a seguir: para cada tarefa $j(j = 1,...,n)$, foi gerado um tempo de processamento p_j

[1] As instâncias adaptadas para os experimentos deste trabalho podem ser acessadas no seguinte endereço: http://algox.icomp.ufam.edu.br/benchmark-instances/weighted-earliness-tardiness-scheduling

a partir de uma distribuição uniforme no intervalo $[1,100]$ e um valor inteiro associado ao peso ou penalidade w_j foi gerado a partir e uma distribuição uniforme no intervalo $[1,10]$. Para se ter instâncias com complexidades distintas, as datas de término sugeridas d_j foram geradas através de diferentes distribuições uniformes. Desta forma, para um determinado intervalo de datas de término sugeridas $RDD(RDD = 0.2, 0.4, 0.6, 0.8, 1.0)$ e para uma determinada média correspondente ao fator de atraso $TF(TF = 0.2, 0.4, 0.6, 0.8, 1.0)$, uma data de término sugerida d_j para cada tarefa j é randomicamente gerada através da distribuição uniforme no intervalo $[P(1-TF-RDD/2), P(1-TF+RDD/2)]$, onde $P = \sum_{i=1}^{n} p_j$. Cinco instâncias foram geradas para cada 25 pares de valores de RDD e TF, obtendo assim 125 instâncias no total para cada valor de n (CRAUWELS; POTTS; WASSENHOVE, 1998). Um exemplo de instância é apresentado na Tabela 5.2.

Tabela 5.2 Exemplo de instância utilizada.

	1^a coluna	2^a coluna	3^a coluna	4^a coluna
1^a linha	8 (número de tarefas)			
2^a linha em diante	tempo de processamento	datas de término sugeridas	penalidades de antecipação	penalidade de atraso
	3	3	3	5
	2	6	4	5
	3	6	8	8
	4	4	8	10
	6	6	6	4
	7	10	7	3
	3	11	4	2
	3	8	5	8

5.3 Parametrização da metaheurística GLS

Além dos limites de tempo de execução e número de gerações como critério de parada, a metaheurística GLS de Amorim (2013), contou com melhorias na geração de novas soluções (nas operações de cruzamento e mutação - ver Seções 4.2.1.4 e 4.2.1.5), critério de diversidade (ver Seção 4.2.1.5) para garantir que toda a população seja a mais diversificada possível.

A população considerada nesta pesquisa foi de 40 indivíduos, obtida a partir de análise empírica de diversos experimentos realizados com diferentes tamanhos de população.

Primeiramente os testes foram realizados com populações maiores e variando com o tamanho da instância. Assim, testou-se com população de tamanho $m \times n$ indivíduos e depois foram realizados outros experimentos com n indivíduos como população, onde m é o número de máquinas e n o número de tarefas. Dessa forma, com base nos resultados, foi verificado que o método tinha um tempo de execução muito grande (problema inclusive reportado em Amorim (2013)). E o espaço de busca não era bem explorado, o algoritmo ficava preso em ótimos locais, logo, foi quando se teve a ideia de se trabalhar com critério de diversidade adaptativa, apresentada na Seção 4.2.1.5.

Nesta diversidade adaptativa, concluiu-se que era necessário aplicar mutação em toda a população quando os cromossomos da população tiverem mais de 80% dos seus genes repetidos entre si. Mas outros casos também foram testados, ou seja, foram realizados testes com critério de diversidade de 10% a 90%. Assim, chegou-se a conclusão, também com base em análise empírica que, aplicando mutação em toda a população somente quando os cromossomos da população tiverem mais de 80% dos seus genes repetidos entre si, geravam melhores resultados para o problema JIT_Sched.

Dessa forma, como populações de tamanho maior não estavam sendo efetivas, pensou-se em testar com tamanhos menores de população. Testou-se com população de 8, 10, 12 e 20 indivíduos, como apresentado na Tabela 5.3, que apresenta a comparação do método UILS, de Kramer e Subramanian (2015), e a primeiva versão do algoritmo genético de Amorim (2013). Verifica-se, nesta tabela, que os melhores resultados foram obtidos com a população de tamanho 40 para instâncias de 2-4 máquinas e 20 indivíduos para instâncias a partir de 10 máquinas, que foram os utilizados nos experimentos desta pesquisa.

Tabela 5.3 Análise de sensibilidade com 100 tarefas.

m	Ger.	UILS		GLS (8)		GLS (10)		GLS (12)		GLS (20)	
		Média soluções	**Média tempo**	**Média soluções**	**Média tempo**	**Média soluções**	**Média tempo**	**Média soluções**	**Média tempo**	**Média soluções**	**Média tempo**
2	20	0.008%	168.5	0.014%	33.5	0.010%	38.9	0.021%	50.4	0.004%	72.9
	40			0.007%	67.3	**0.004%**	**95.4**	0.004%	124.3	0.001%	178.6
	100			0.001%	168.1	0.001%	194.4	0.001%	253.2	0.000%	365.7
4	20	0.156%	181.0	0.103%	49.2	0.100%	61.4	0.088%	77.7	0.087%	111.6
	40			0.074%	97.3	**0.076%**	**151.1**	0.064%	190.9	0.072%	271.0
	100			0.049%	240.7	0.056%	299.7	0.040%	378.6	0.050%	536.5
10	10	0.202%	140.4	0.294%	42.4	0.260%	49.9	0.262%	67.1	0.247%	99.4
	20			0.192%	79.2	**0.175%**	**109.5**	0.165%	146.3	0.156%	215.7
	50			0.148%	187.5	0.132%	217.4	0.104%	291.3	0.118%	425.2
Todos	20 (10)	0.122%	163.3	0.137%	41.7	0.123%	50.1	0.124%	65.1	0.112%	94.6
	40 (20)			0.091%	81.3	**0.085%**	**118.7**	0.078%	153.8	0.077%	221.7
	100 (50)			0.066%	198.8	0.063%	237.1	0.048%	307.7	0.056%	442.4

Outra melhoria no algoritmo de Amorim (2013) foi a inclusão de $m-1$ tarefas fictícias, com o objetivo de garantir que o método obtenha soluções ótimas. Cada tarefa fictícia tem a função de bloquear a máquina que foi alocada, ou seja, nenhuma tarefa é incluída na máquina em que a tarefa fictícia foi alocada. Isto se dá por que a tarefa fictícia tem um tempo de processamento que é igual ao somatório de todos os tempos de processamento da instância a ser testada ($\sum_{j=1}^{n} p_j$). De modo que, se a próxima tarefa da sequência for alocada após esta tarefa fictícia, ela pode ficar com um custo muito alto, o que é automaticamente descartado pelo método proposto nesta tese. Logo, esta tarefa deverá ser, obrigatoriamente, alocada em outra máquina disponível mais cedo (sem tarefa fictícia).

Assim, a decisão de incluir $m-1$ tarefas fictícias na instância, é que pelo menos uma máquina deverá receber tarefas livremente, sem que se tenha problemas quando o método for alocar as tarefas nas máquinas.

5.4 Resultados para o $1||\sum \alpha_j E_j + \sum \beta_j T_j$

Nas Tabelas A.1-A.5 são apresentados os experimentos computacionais detalhados para até 200 tarefas. As tabelas citadas possuem as seguintes colunas: **Inst.** - para o número da instância de Tanaka (2012b), **Média custo GLS** e **Média custo PI** apresentam a média de custo obtido do método GLS e da formulação de programação inteira (PI) arc-time de 10 execuções com sementes diferentes, da mesma forma que são apresentadas as suas respectivas médias de tempo nas colunas **Média tempo GLS** e **Média tempo PI**. A seguir são apresentados o tempo mínimo, a média de tempo total e o tempo máximo de execução das duas partes do algoritmo (GLS + melhoria da solução com a formulação de PI arc-time) obtido das 10 execuções do algoritmo. Nas colunas **Melhor iter. GLS** e **Melhor tempo GLS** são apresentadas a iteração/geração e o instante em que o GLS obtém a melhor solução obtida, respectivamente.

Nas colunas **Melhor GLS** e **MathGLS-IP** estão as melhores soluções obtidas pelo GLS e o resultado da melhoria da solução obtida pelo GLS pela formulação de PI arc-time (dos arcos recebidos da heurística), em alguns casos pode acontecer da formulação não melhorar a solução do GLS. O resultado da coluna **MathGLS-IP** são comparados com a coluna Tanaka (2012b). A coluna **MSC** apresenta a Melhor Solução Conhecida na

Tabela 5.4 Resumos dos resultados computacionais para o problema $1||\sum \alpha_j E_j + \sum \beta_j T_j$ - 40, 50, 100, 200 tarefas e 1 máquina.

Grupo de instâncias	Tanaka (2012b)			MathGLS-IP				
	Melhor obtido		Média	Melhor obtido			Média	
	GAP (%)	MSC	Tempo (s)	GAP (%)	#	*	GAP (%)	Tempo (s)
wet40-1m	0.000	25	0.112	0.000	25	0	0.000	4.235
wet50-1m	0.000	25	0.245	0.000	25	0	0.000	7.755
wet100-1m	0.000	25	4.105	0.001	24	0	0.002	77.361
wet200-1m	0.000	25	57.695	0.001	23	0	0.009	637.960
wet300-1m	0.000	25	366.412	0.007	12	0	0.019	2044.159
Total	···	125	···	···	109	0	···	···
Média	0.000	···	85.714	0.002	···	···	0.006	554.294

MSC - Quantidade de soluções que são iguais às Melhores Soluções Conhecidas (MSC) da literatura.
- Quantidade de soluções iguais à MSC.
***** - Quantidade de soluções que foram melhoradas em relação a literatura.

literatura.

A Tabela 5.4 apresenta uma sumarização dos resultados para o ambiente monoprocessado, comparado com Tanaka (2012b). A intenção é mostrar que o método híbrido proposto nesta pesquisa consegue atingir soluções ótimas, também para o ambiente monoprocessado. Já que a maioria dos ótimos de Tanaka (2012b) são atingidos, com poucas instâncias que não se igualaram à literatura, a vantagem do método proposto em relação ao da literatura é que a estratégia híbrida proposta apresenta resultados para máquinas paralelas idênticas, ainda pouco explorado no estado da arte. Estes resultados são apresentados na próxima seção, que são bem competitivos em relação a trabalhos já publicados. O GAP (%) é calculado como segue: $GAP = \frac{\text{Solução} - MSC}{MSC}$, onde Solução é o melhor obtido do método a ser comparado e MSC é a Melhor Solução Conhecida.

5.5 Resultados para o $P||\sum \alpha_j E_j + \sum \beta_j T_j$

Nas Tabelas B.1-B.16 são apresentados os experimentos computacionais detalhados para até 500 tarefas em máquinas paralelas idênticas. Além das colunas citadas, o método hí-

brido proposto é comparado com os resultados de Kramer e Subramanian (2015), e a implementação da formulação *IP-NetworkFlow* implementada no CPLEX (com a utilização da biblioteca *Concert Technology*) e seu respectivo GAP (%) do CPLEX, apresentados nas colunas **IP-NFlow** e **IP-NFlow (GAP%)**, respectivamente. A coluna **MSC** apresenta a Melhor Solução Conhecida na literatura. Os melhores resultados obtidos em relação ao MSC foram destacados nas tabelas.

A Tabela 5.5 apresenta uma sumarização dos resultados para o ambiente de máquinas paralelas idênticas, comparado com os resultados do método UILS de Kramer e Subramanian (2015) e os resultados implementados no CPLEX da formulação *IP-NetworkFlow*. A intenção de comparar com a implementação de uma formulação no CPLEX é de mostrar que o método proposto desta pesquisa consegue também atingir os ótimos para máquinas paralelas. É possível observar na Tabela 5.5 que o método híbrido consegue fornecer mais soluções iguais a Melhor Solução Conhecida - MSC.

Na Tabela 5.5 são resumidos os resultados para 225 instâncias testas, onde: o método UILS de Kramer e Subramanian (2015) apresenta resultados para somente 85 instâncias, sendo que, 75 delas são as melhores soluções conhecidas (**MSC**). A formulação *IP-NetworkFlow* implementada, de Amorim (2013), considerando as 225 instâncias, conseguiu atingir as melhores soluções conhecidas para 193 delas. Já o método híbrido desta pesquisa, conseguiu atingir 200 melhores soluções conhecidas, também considerando as 225 mencionadas anteriormente, sendo que, 4 soluções são melhores que todos os métodos comparados para 100 tarefas e 4 máquinas.

Os resultados para instâncias de tamanho maior não puderam ser comparados, pois ainda não existem resultados na literatura para tais instâncias. Na Figura 5.1 são apresentados o comportamento do método híbrido proposto relacionado à melhora da formulação de PI arc-time com os arcos recebidos da meta-heurística GLS. O eixo das ordenadas apresentam os valores de função objetivo e o eixo das abscissas apresentam os números das instâncias de teste. As curvas em azul representam as soluções fornecidas pela metaheurística GLS e as curvas em azul são as soluções obtidas pelo algoritmo *Branch-and-Cut* do CPLEX, com os arcos obtidos das soluções do GLS, apresentando muitas melhorias, principalmente para 2 e 4 máquinas. Estima-se que sejam necessários mais testes para instâncias a partir de 10 máquinas para verificar se é possível obter melhorias com os arcos obtidos pelo método GLS.

Na Tabela 5.6 são sumarizados os resultados da melhoria do CPLEX para cada instância testada para 1, 2, 4 e 10 máquinas. Nesta tabela, as melhorias são classificadas em 4 (quatro) tipos, apresentadas a seguir: **A** - CPLEX não melhorou e terminou por tempo; **B** - CPLEX melhora a solução e terminou por tempo (melhor Limite Superior); **C** - CPLEX não melhorou mas termina no tempo (prova o ótimo) e **D** - CPLEX melhora e termina (fornece o ótimo). Os resultados obtidos ficaram basicamente nos tipos **C** e **D**, isto se deve ao fato da metaheurística GLS já fornecer boas soluções ou até mesmo o ótimo para algumas instâncias, principalmente para 40, 50 e 100 tarefas. A partir de 100 tarefas, o exato consegue melhorias significativas, principalmente em 2 e 4 máquinas (casos do tipo **D**). Para 10 máquinas quase não há melhoria, neste caso, são necessários mais testes para verificar se as soluções fornecidas pela metaheurística GLS são realmente as melhores, pois se forem, o método exato do CPLEX não teria muito o que melhorar. Assim, instâncias com um número de máquinas maior devem ser testadas com mais arcos para verificar se ainda é possível se obter melhores soluções pelo algoritmo *Branch-and-Cut* do CPLEX.

5.6 Resumo do capítulo

Este capítulo apresentou os experimentos computacionais realizados para estratégia híbrida proposta MathGLS-IP, considerando a minimização das penalidades de antecipação e atraso de tarefas, nos ambientes mono e multiprocessado (máquinas paralelas idênticas). Os resultados apresentados são competitivos em relação à literatura e o método proposto consegue atingir soluções ótimas na maioria dos casos testados. Sendo que o método MathGLS-IP proposto consegue melhorar 4 instâncias da literatura. Os demais resultados para instâncias maiores não puderam ser comparados por não haver resultados na literatura para comparação. Dada a comparação com soluções ótimas ou quase-ótimas conhecidas na literatura pode-se afirmar que os resultados para instâncias maiores são muito bons e podem ser competitivos com estratégias de outras pesquisas. O que ressalta a necessidade da utilização de estratégias exatas mais robustas para o problema.

Tabela 5.5 Resumos dos resultados computacionais para o problema $P||\sum \alpha_j E_j + \sum \beta_j T_j$ - 40, 50, 100 tarefas e 2-10 máquinas.

Grupo de instâncias	Kramer e Subramanian (2015)				*IP-NetworkFlow*			MathGLS-IP				
	Melhor obtido		Médias		Melhor obtido		Médias	Melhor obtido			Médias	
	GAP (%)	MSC	GAP (%)	Tempo (s)	GAP (%)	MSC	Tempo (s)	GAP (%)	#	*	GAP (%)	Tempo (s)
wet40-2m	0.000	12	0.000	5.592	0.000	22	448.314	0.000	24	0	0.001	6.420
wet40-4m	0.000	12	0.001	6.258	0.000	25	32.419	0.000	24	0	0.001	9.261
wet40-10m	0.000	5	0.000	4.080	0.000	22	3.175	0.000	25	0	0.002	8.882
wet50-2m	0.000	11	0.001	12.617	0.000	21	1199.022	0.000	23	0	0.000	13.623
wet50-4m	0.000	12	0.306	14.145	0.000	22	245.953	0.000	25	0	0.004	15.939
wet50-10m	0.000	5	0.014	9.320	0.000	23	5.839	0.000	24	0	0.038	13.686
wet100-2m	0.000	12	0.008	168.483	0.000	24	3000.365	0.000	24	0	0.004	166.087
wet100-4m	0.790	6	0.858	190.309	0.053	12	2129.438	0.000	23	4	0.055	114.163
wet100-10m	0.161	0	0.227	140.380	0.000	22	781.456	0.089	8	0	0.332	102.153
Total	···	75	···	···	···	193	···	···	200	4	···	···
Média	0.106	···	0.157	61.243	0.006	···	871.776	0.010	···	···	0.049	50.024

MSC - Quantidade de soluções que são iguais às Melhores Soluções Conhecidas (MSC) da literatura.
- Quantidade de soluções iguais à MSC.
***** - Quantidade de soluções que foram melhoradas em relação a literatura.

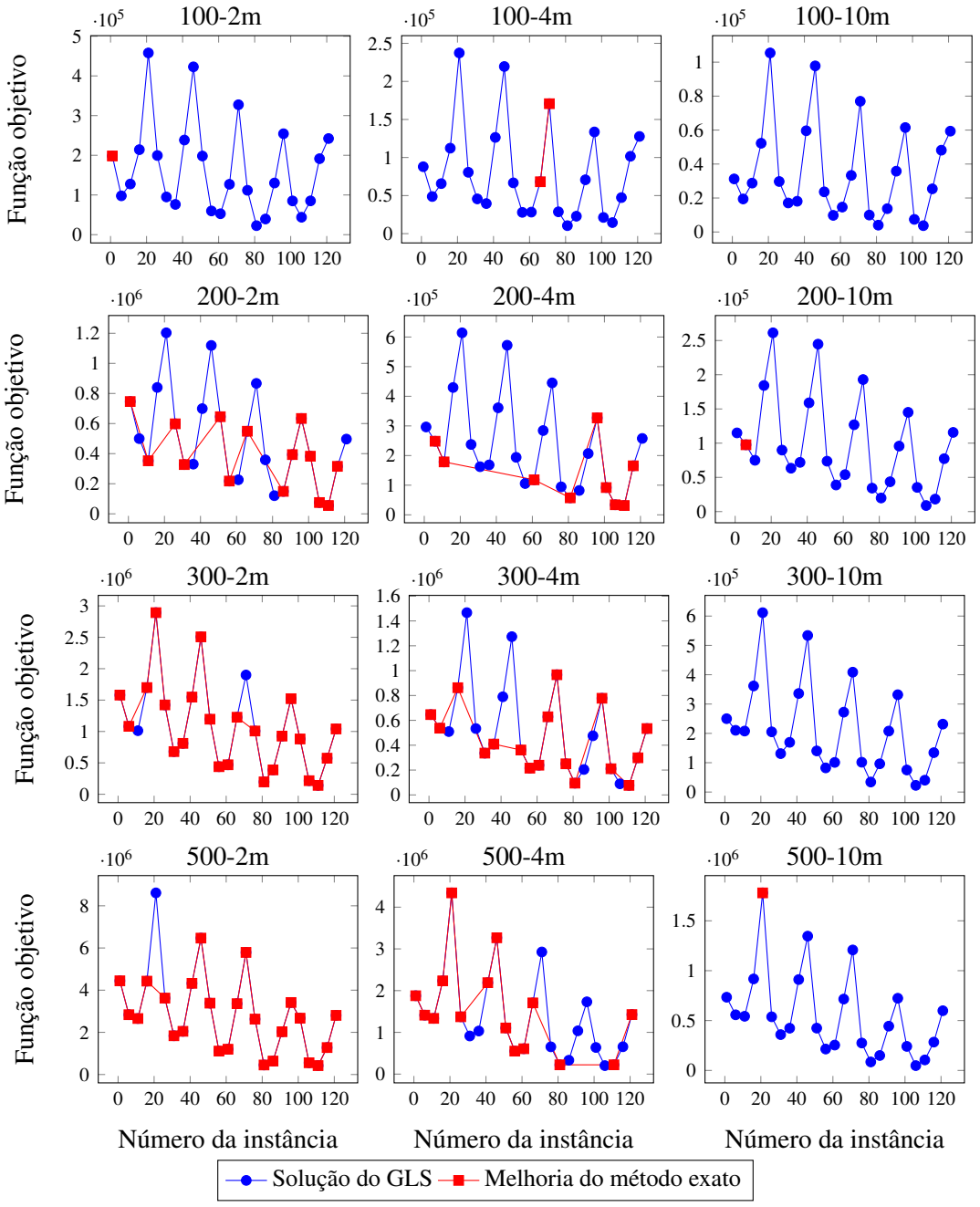


Figura 5.1 Melhoria da estratégia híbrida MathGLS-IP em relação ao método GLS para o escalonamento E/T em máquinas paralelas idênticas.

Tabela 5.6 Sumarização dos resultados computacionais do CPLEX da estratégia híbrida MathGLS-IP para o escalonamento E/T.

Intâncias	100 tarefas				200 tarefas				300 tarefas				500 tarefas		
	1m	2 m	4 m	10 m	1m	2 m	4 m	10 m	1m	2 m	4 m	10 m	2 m	4 m	10 m
1	C	**D**	C	C	C	**D**	C	C	C	**D**	**D**	C	**D**	**D**	C
6	C	C	C	C	C	C	**D**	**D**	**D**	**D**	**D**	C	**D**	**D**	C
11	C	C	C	C	C	**D**	**D**	C	C	C	C	C	**D**	**D**	C
16	C	C	C	C	C	C	C	C	C	**D**	**D**	C	**D**	**D**	C
21	C	C	C	C	C	C	C	C	**D**	**D**	C	C	C	**D**	**D**
26	C	C	C	C	C	**D**	C	C	C	**D**	C	C	**D**	**D**	C
31	C	C	C	C	C	**D**	C	C	C	**D**	**D**	C	**D**	C	C
36	C	C	C	C	C	C	C	C	C	**D**	**D**	C	**D**	C	C
41	C	C	C	C	C	C	C	C	C	**D**	C	C	**D**	**D**	C
46	C	C	C	C	C	C	C	C	**D**	**D**	C	C	**D**	**D**	C
51	C	C	C	C	C	**D**	C	C	**D**	**D**	**D**	C	**D**	**D**	C
56	C	C	C	C	C	**D**	C	C	C	**D**	**D**	C	**D**	C	C
61	C	C	C	C	C	C	**D**	C	**D**	**D**	**D**	C	**D**	**D**	C
66	C	C	**D**	C	C	**D**	C	C	**D**	**D**	**D**	C	**D**	**D**	C
71	C	C	**D**	C	C	C	C	C	C	C	**D**	C	**D**	C	C
76	C	C	C	C	C	C	C	C	**D**	**D**	**D**	C	**D**	C	C
81	C	C	C	C	C	C	**D**	C	C	**D**	**D**	C	**D**	C	C
86	C	C	C	C	C	**D**	C	C	C	**D**	C	C	**D**	C	C
91	C	C	C	C	C	**D**	C	C	C	**D**	C	C	**D**	C	C
96	C	C	C	C	**D**	**D**	**D**	C	**D**	**D**	**D**	C	**D**	C	C
101	C	C	C	C	C	**D**	**D**	C	C	**D**	**D**	C	**D**	C	C
106	C	C	C	C	**D**	**D**	**D**	C	C	**D**	C	C	**D**	C	C
111	C	C	C	C	C	**D**	**D**	C	C	**D**	**D**	C	**D**	**D**	C
116	C	C	C	C	C	**D**	**D**	C	**D**	**D**	**D**	C	**D**	C	C
121	C	C	C	C	**D**	C	C	C	C	**D**	**D**	C	**D**	**D**	C

A - CPLEX não melhorou e terminou por tempo.
B - CPLEX melhora a solução e terminou por tempo (melhor Limite Superior).
C - CPLEX não melhorou mas termina no tempo (prova o ótimo).
D - CPLEX melhora e termina (fornece o ótimo).

Capítulo 6

Considerações finais

Esta pesquisa considerou problemas de escalonamento com penalidades de antecipação e atraso em ambiente monoprocessado e ambiente de máquinas paralelas idênticas, com tarefas independentes, de tempos de processamento arbitrários e pesos distintos. Na fundamentação teórica foram apresentadas as definições, conceitos e notação clássica da literatura. Assim como as principais formulações matemáticas para os problemas de escalonamento E/T, incluindo a formulação para o problema de escalonamento com antecipação e atraso em ambientes mono e multiprocessado sem tempo ocioso entre as tarefas (*PI-NetworkFlow*) de Amorim (2013). Esta formulação foi baseada na formulação clássica geral indexada pelo tempo de Dyer e Wolsey (1990) e em um modelo de fluxo em redes.

Uma sumarização dos resultados da revisão da literatura foi apresentada na Tabela 2.1, que utilizou a notação de três campos $\alpha|\beta|\gamma$, amplamente utilizada para a representação dos problemas de escalonamento na literatura (GRAHAM et al., 1979; LAWLER et al., 1993), onde α descreve o ambiente de processamento e número de máquinas, β descreve os tipos de restrições e características das tarefas e γ descreve o critério de otimização ou função objetivo. Na Tabela 2.1 também foram apresentados os ambientes de processamento investigados nesta pesquisa (uma máquina - notação: 1; e máquinas paralelas idênticas - notação: P ou P_m), juntamente com as estratégias algorítmicas utilizadas e suas respectivas referências.

Entre as publicações identificados na revisão da literatura, a maioria delas são propostas para o ambiente monoprocessado, considerando restrições escalonamento em lotes (*batch processing*), tempos de preparação (*setups*), tempos de processamento arbitrários (p_j), datas de término em comum ($d_j = d$), datas de chegada (r_j), precedência (*prec*) e

preempção (*pmtn*). Já para o ambiente de máquinas paralelas idênticas, são consideradas as restrições de datas de término em comum ($d_j = d$), datas de chegada (r_j) e tempos de processamento arbitrários (p_j).

A literatura referente ao escalonamento de tarefas com penalidades de antecipação e atraso é muito ampla, incluindo os mais diversos ambientes de processamento e estratégias de resolução. Esta pesquisa se focou nos problemas que abordam os ambientes de uma máquina e máquinas paralelas. Uma estratégia híbrida, exato-heurística, foi proposta para o problema. Nesta estratégia foi utilizado o Algoritmo Genético com busca local (GLS) de Amorim (2013), melhorada com a inclusão de um critério de diversidade adaptativa, cruzamento seguido de busca local em toda a população e tratamento de escalonamento desbalanceado para funções não-regulares. Assim, um conjunto de soluções ótimas locais foi criado a partir do GLS, cada ótimo local é a melhor solução obtida de cada geração do GLS. Estes ótimos irão compor os arcos que são armazenados em uma tabela hash com endereçamento aberto com *hashing* duplo.

Ao final do passo heurístico, estes arcos são utilizados para montar um modelo matemático relaxado a ser resolvido pelo algoritmo *Branch-and-Cut* via CPLEX e assim verificar se era possível melhorar a solução obtida pelo GLS. O objetivo deste método híbrido foi de obter resultados para instâncias de tamanho maior em máquinas paralelas, ainda não existentes na literatura. Os resultados computacionais mostram que o passo exato do método melhora a solução recebida pela meta-heurística GLS, especialmente nos ambientes de 2 e 4 máquinas, e o método apresentado mostrou-se ser bastante competitivo em relação a literatura, atingindo os ótimos e melhores soluções conhecidas e até mesmo melhorando resultados existentes.

Foi possível também apresentar resultados para instâncias maiores para ambientes de máquinas paralelas, mas que ainda não puderam ser comparados por não existirem trabalhos com instâncias de tamanho maior, mas pode-se afirmar que os resultados para instâncias maiores também são competitivos para futuras comparações com outras pesquisas.

6.1 Hipótese da pesquisa e objetivos

O principal objetivo desta tese consistiu em elaborar uma estratégia algorítmica exata-heurística competitiva com o estado-da-arte, para a resolução do problema de escalonamento de tarefas independentes e ponderadas em máquinas paralelas idênticas, visando a minimização de penalidades de antecipação e atraso. Os objetivos específicos, que são: (a) analisar e adaptar formulações matemáticas de programação linear inteira e inteira mista existentes para o problema JIT_Sched; (b) elaborar um método metaheurístico adaptativo fortemente baseado em busca local e (c) resolver grandes instâncias do problema JIT_Sched, gerando ótimos ou melhores limites superiores, foram todos atingidos.

A contribuição desta pesquisa foi na melhoria da estratégia de Amorim (2013), chamada neste trabalho de metaheurística GLS. A melhoria contou com variação na geração de soluções aleatórias inicias, mutação adaptativa e ajuste critério de parada para a obtenção dos arcos. Outra contribuição foi a estratégia híbrida proposta, exato-heurística. O algoritmo exato utilizado foi o *Branch-and-Cut* via CPLEX para a resolução da formulação *arc-time* relaxada com os arcos obtidos do algoritmo GLS. A estratégia hibrida resolveu instâncias de até 500 tarefas em 1, 2, 4 e 10 máquinas, com soluções ótimas ou melhores limites superiores até então não existentes na literatura para máquinas paralelas idênticas. Assim, a hipótese levantada nesta pesquisa foi respondida de forma positiva. Pois os resultados obtidos também são competitivos quando comparados aos existentes na literatura, para 40, 50 e 100 tarefas.

6.2 Trabalhos futuros

A estratégia híbrida proposta ainda pode ser refinada através da melhoria do cálculo da função objetivo para funções não-regulares que, atualmente, as tarefas são alocadas nas máquinas disponíveis mais cedo, juntamente com $m-1$ tarefas fictícias (*dummy jobs*) para assegurar que o método possa atingir o ótimo para cada instância. Uma possível melhoria na alocação das tarefas nas máquinas poderia seguir uma estratégia gulosa. Onde cada tarefa seria alocada somente na máquina em que tivesse o menor custo nas penalidades de antecipação e atraso, ou seja, na máquina em que a tarefa pudesse finalizar o seu processamento o mais próximo possível de sua data de término sugerida.

Um aprofundamento no estudo e tratamento de soluções com empate e simétricas

também é necessário, de modo que os procedimentos algorítmicos que sejam capazes de detectar tais soluções equivalentes sejam elaborados. Assim, tais equivalências não precisam ser consideradas durante o processo de busca por melhores soluções, permitindo assim uma redução significativa do tempo de processamento do algoritmo. Por outro lado, é interessante também investigar novos ambientes de processamento como máquinas de propósito geral (*mpm*) e diferentes tipos de restrições, tais como: tempos de preparação (*setups*), datas de término iguais (*common due dates*) e preempção (*pmtn*).

6.3 Conferências e publicações

Os trabalhos apresentados/publicados estão listados abaixo em ordem cronológica do mais recente ao mais antigo:

a) **Artigo completo no CLEI 2017 - XLIII Conferencia Latinoamericana en Informática - Argentina/Córdova:** *Solving large instances applying meta-heuristics for classical parallel machine scheduling problems under tardiness and earliness penalties* (AMORIM; THOMAZ; DEFREITAS, 2017).

b) **Artigo completo no XLVI Simpósio Brasileiro de Pesquisa Operacional - SBPO 2014:** *Um estudo sobre formulações matemáticas para um problema clássico de escalonamento com antecipação e atraso em máquinas paralelas* (AMORIM; FREITAS, 2014).

c) **Artigo completo no XLV Simpósio Brasileiro de Pesquisa Operacional - SBPO 2013:** *A network flow IP formulation and exact/heuristics approaches for just-in-time scheduling problems on parallel machines without idle times* (AMORIM; DEFREITAS; UCHOA, 2013).

d) **Apresentação de trabalho na ELAVIO 2013 - XVII Escola Latino-iberoamericana de Pesquisa Operacional:** *Integer mathematical formulations and algorithms for scheduling problems with earliness and tardiness penalties on parallel machines* (AMORIM; FREITAS; UCHOA, 2013).

Referências

AHMADIZAR, F.; FARHADI, S. Single-machine batch delivery scheduling with job release dates, due windows and earliness, tardiness, holding and delivery costs. *Computers and Operations Research*, v. 53, p. 194–205, 2014. Cited By 0. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84907739030&partnerID=40&md5=bfdd02fdeb9c2ba79165070f3445dcc4>. Citado na página 27.

AKKER, J. M. van den; HOESEL, C. P. M. van; SAVELSBERGH, M. W. P. A polyhedral approach to single-machine scheduling problems. *Mathematical Programming*, v. 85, n. 3, p. 541, 1999. Disponível em: <http://link.springer-ny.com/link/service/journals/10107/bibs/9085003/90850541.htm>. Citado na página 36.

ALHARKAN, I.; AZIZ, T.; ALHAAG, M. M. Minimization of earliness & tardiness penalties with common due dates problem using tabu search. In: . [s.n.], 2015. Cited By 0. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84931030070&partnerID=40&md5=943f7ee10f9379a387fee7baf6fab056>. Citado na página 29.

ALVAREZ-VALDES, R. et al. Minimizing weighted earliness-tardiness on a single machine with a common due date using quadratic models. *TOP*, v. 20, n. 3, p. 754–767, 2012. Cited By 2. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84866565473&partnerID=40&md5=2b82afcf33180363e3868034cedd1a9b>. Citado na página 28.

ALVAREZ-VALDES, R.; TAMARIT, J.; VILLA, F. Minimizing weighted earliness-tardiness on parallel machines using hybrid metaheuristics. *Computers and Operations Research*, v. 54, p. 1–11, 2014. Cited By 1. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84907971027&partnerID=40&md5=e07ce866e977928d310a1eee0096eafd>. Citado na página 30.

AMORIM, R. X. *Um Estudo Sobre Formulações Matemáticas e Estratégias Algorítmicas para Problemas de Escalonamento em Máquinas Paralelas com Penalidades de Antecipação e Atraso*. 136 f. Dissertação (Mestrado em Informática) — Instituto de Computação – Universidade Federal do Amazonas, Manaus - AM, 2013. Citado 15 vezes nas páginas 2, 4, 30, 39, 44, 45, 47, 52, 65, 66, 67, 69, 74, 75 e 76.

AMORIM, R. X. d.; DEFREITAS, R.; UCHOA, E. A network flow IP formulation and exact/heuristics approaches for just-in-time scheduling problems on parallel machines without idle times. *Anais do XLV SBPO - Simpósio Brasileiro de Pesquisa Operacional - XLV SBPO*, v. 45, p. 1835–1846, 2013. ISSN 1518-1731. Citado 3 vezes nas páginas 47, 59 e 77.

AMORIM, R. X. d.; FREITAS, R. de. Um estudo sobre formulações matemáticas para um problema clássico de escalonamento com antecipação e atraso em máquinas paralelas. v. 46, p. 2514–2525, 2014. ISSN 1518-1731. Citado na página 77.

AMORIM, R. X. d.; FREITAS, R. de; UCHOA, E. Integer mathematical formulations and algorithms for scheduling problems with earliness and tardiness penalties on parallel machines. *ELAVIO 2013 - XVII Escola Latino-iberoamericana de Pesquisa Operacional*, p. 1–2, 2013. Citado na página 77.

AMORIM, R. X. d.; THOMAZ, M.; DEFREITAS, R. Solving large instances applying matheuristics for classical parallel machine scheduling problems under tardiness and earliness penalties. *Anais do CLEI 2017 - XLIII Conferencia Latinoamericana en Informática*, 2017. Citado na página 77.

ARENALES, M. et al. *Pesquisa Operacional: As Disciplinas da Execução da Estratégia*. Elsevier, 2007. ISBN 9788535251937. Disponível em: <http://books.google.com.br/books?id=sB__Fi8rprEC>. Citado 2 vezes nas páginas 32 e 44.

AWASTHI, A.; LÄSSIG, J.; KRAMER, O. Common due-date problem: Exact polynomial algorithms for a given job sequence. *CoRR*, abs/1311.2879, 2013. Disponível em: <http://arxiv.org/abs/1311.2879>. Citado 2 vezes nas páginas 27 e 30.

AWASTHI, A. et al. Common due-window problem: Polynomial algorithms for a given processing sequence. In: . [s.n.], 2015. p. 32–39. Cited By 0. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84922496837&partnerID=40&md5=e702d84b5cd3be318965766e3c8e286b>. Citado na página 16.

BAKER, K. R.; SCUDDER, G. D. Sequencing with earliness and tardiness penalties: a review. *Oper. Res.*, INFORMS, Institute for Operations Research and the Management Sciences (INFORMS), Linthicum, Maryland, USA, v. 38, p. 22–36, Fevereiro 1990. ISSN 0030-364X. Citado 2 vezes nas páginas 15 e 30.

BAPTISTE, P.; SADYKOV, R. On scheduling a single machine to minimize a piecewise linear objective function: A compact mip formulation. *Naval Research Logistics*, v. 56, n. 6, p. 487–502, 2009. Cited By 1. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-68949090174&partnerID=40&md5=3e6c5b94c3d3b3f19a37577b5eb95cd8>. Citado na página 28.

BATISTA, A.; BATISTA, L. A distance based variable neighborhood search for parallel machine scheduling. In: . [s.n.], 2015. p. 106–113. Cited By 0. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84922573107&partnerID=40&md5=01f57f945e2a2e9c56db015a379d86ed>. Citado na página 30.

BENMANSOUR, R.; ALLAOUI, H.; ARTIBA, A. Single machine scheduling problem in a just-in-time environment. In: *Logistics (LOGISTIQUA), 2011 4th International Conference on*. [S.l.: s.n.], 2011. p. 362–366. Citado na página 28.

BRUCKER, P. *Scheduling Algorithms*. 5th. ed. Secaucus, NJ, USA: Springer-Verlag New York, 2006. ISBN 3540415106. Citado 3 vezes nas páginas 1, 3 e 7.

CHANG, P. Branch and bound approach for single machine scheduling with earliness and tardiness penalties. *Computers and Mathematics with Applications*, v. 37, n. 10, p. 133–144, 1999. Cited By 22. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-0033130807&partnerID=40&md5=2555e46fee35e2387cd54c6827cde853>. Citado 2 vezes nas páginas 23 e 28.

CHENG, T.; CHEN, Z.-L. Parallel-machine scheduling problems with earliness and tardiness penalties. *Journal of the Operational Research Society*, v. 45, n. 6, p. 685–695, 1994. Cited By 62. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-0028445470&partnerID=40&md5=8ff9a5e0087e33bb2e537e4f7118d0ba>. Citado na página 16.

CHUNG, D.-Y.; CHOI, B.-C. Just-in-time scheduling under scenario-based uncertainty. *Asia-Pacific Journal of Operational Research*, v. 30, n. 2, 2013. Cited By 0. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84874943289&partnerID=40&md5=b87098f43657a64cc5248332ff96f4c5>. Citado na página 16.

CONGRAM, R. K.; POTTS, C. N. *OR-Library - Benchmark instances the single-machine total weighted tardiness problem*. 1998. Disponível em: <http://people.brunel.ac.uk/~mastjjb/jeb/info.html>. Citado na página 64.

CRAUWELS, H. A. J.; POTTS, C. N.; WASSENHOVE, L. N. V. Local search heuristics for the single machine total weighted tardiness scheduling problem. *INFORMS J. on Computing*, INFORMS, Institute for Operations Research and the Management Sciences (INFORMS), Linthicum, Maryland, USA, v. 10, n. 3, p. 341–350, Março 1998. ISSN 1526-5528. Citado na página 65.

CROCE, F. D.; GARAIX, T.; GROSSO, A. Iterated local search and very large neighborhoods for the parallel-machines total tardiness problem. *Computers & Operations Research*, v. 39, p. 1213–1217, 2012. Citado na página 25.

DEFREITAS, R. *Caracterizações e algoritmos para problemas clássicos de escalonamento*. 83 p. Tese (Doutorado), Rio de Janeiro, 2009. Citado 4 vezes nas páginas 1, 3, 7 e 11.

DEFREITAS, R. et al. Heuristic algorithm for the parallel machine total weighted tardiness scheduling problem. *Relatórios de pesquisa em engenharia de produção*, v. 8, p. 1–12, 2008. Citado 8 vezes nas páginas 25, 26, 47, 51, 52, 54, 57 e 58.

DETIENNE, B.; PINSON, E.; RIVREAU, D. Lagrangian domain reductions for the single machine earliness-tardiness problem with release dates. *European Journal of Operational Research*, v. 201, n. 1, p. 45–54, 2010. Cited By 5. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-70249083786&partnerID=40&md5=d374b84f00f06d07f2def44fa5a62ef2>. Citado 3 vezes nas páginas 23, 28 e 30.

DU, J.; LEUNG, J. Y. Minimizing total tardiness on one machine is NP-Hard. *Mathematics of Operations Research*, v. 15, p. 483–495, 1990. Citado na página 4.

DYER, M. E.; WOLSEY, L. A. Formulating the single machine sequencing problem with release dates as a mixed integer program. *Discrete Appl. Math.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 26, n. 2-3, p. 255–270, Fevereiro 1990. ISSN 0166-218X. Disponível em:

<http://dx.doi.org/10.1016/0166-218X(90)90104-K>. Citado 4 vezes nas páginas 36, 37, 44 e 74.

EILON, S.; CHOWDHURY, I. G. Minimising waiting time variance in the single machine problem. *Management Science*, INFORMS, v. 23, p. 297–313, Fevereiro 1977. ISSN 567-575. Citado na página 2.

GORDON, V.; PROTH, J.-M.; CHU, C. A survey of the state-of-the-art of common due date assignment and scheduling research. *European Journal of Operational Research*, v. 139, n. 1, p. 1–25, 2002. Cited By 240. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-0037118177&partnerID=40&md5=bdf78ced2a4e18f10553fc8a7b59ae3c>. Citado 2 vezes nas páginas 15 e 30.

GORDON, V. S.; STRUSEVICH, V. A. Earliness penalties on a single machine subject to precedence constraints: SLK due date assignment. *Computers & OR*, v. 26, p. 157–177, 1999. Citado na página 2.

GRAHAM, R. L. et al. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, v. 5, p. 287–326, 1979. Citado 3 vezes nas páginas 4, 7 e 74.

HALL, N. G.; KUBIAK, W.; SETHI, S. P. Earliness-tardiness scheduling problems, II. deviation of completion times about a restrictive common due date. *Operations Research*, v. 39, n. 5, p. 847–856, 1991. Cited By 138. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-0026223827&partnerID=40&md5=05d7898ff4afdddced7d38063a4db58b>. Citado 2 vezes nas páginas 15 e 16.

HALL, N. G.; POSNER, M. E. Earliness-tardiness scheduling problems, I: Weighted deviation of completion times about a common due date. *Operations Research*, v. 39, n. 5, p. 836–846, 1991. Disponível em: <http://dx.doi.org/10.1287/opre.39.5.836>. Citado na página 15.

HASSIN, R.; SHANI, M. Machine scheduling with earliness, tardiness and non-execution penalties. *Computers & Operations Research*, v. 32, n. 3, p. 683 – 705, 2005. ISSN 0305-0548. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0305054803002624>. Citado 2 vezes nas páginas 14 e 15.

HERRMANN, J. W.; LEE, C.-Y. On scheduling to minimize earliness-tardiness and batch delivery costs with a common due date. *European Journal of Operational Research*, v. 70, p. 272–288, 1993. Citado na página 26.

JAMES, R.; BUCHANAN, J. A neighbourhood scheme with a compressed solution space for the early/tardy scheduling problem. *European Journal of Operational Research*, v. 102, n. 3, p. 513–527, 1997. Cited By 19. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-0031268318&partnerID=40&md5=7aa1f27835286ea54592477489303def>. Citado 2 vezes nas páginas 16 e 28.

JANIAK, A. et al. A survey on scheduling problems with due windows. *European Journal of Operational Research*, v. 242, n. 2, p. 347–357, 2015. Cited By 1. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84920645208&partnerID=40&md5=738cd509d2ca49caf5ae5d743b061768>. Citado na página 16.

JúNIOR, A. G.; CARVALHO, C. Um método heurístico híbrido para a resolução do problema de sequenciamento em uma máquina com penalidades por antecipação e atraso da produção. *Anais do XXXIX Simpósio Brasileiro de Pesquisa Operacional*, 2007. ISSN 1649-1661. Citado 2 vezes nas páginas 24 e 27.

JóZEFOWSKA, J. *Just-in-Time Scheduling: Models and Algorithms for Computer and Manufacturing Systems*. Springer, 2007. (International Series in Operations Research & Management Science). ISBN 9780387717173. Disponível em: <http://books.google.com.br/books?id=HE-pOFD6XVcC>. Citado 4 vezes nas páginas 2, 3, 17 e 20.

KANET, J. J.; SRIDHARAN, V. Scheduling with inserted idle time: problem taxonomy and literature review. *Operations Research*, v. 48, p. 99–110, 2000. Citado 2 vezes nas páginas 16 e 31.

KEDAD-SIDHOUM, S.; SOLIS, Y.; SOURD, F. Lower bounds for the earliness-tardiness scheduling problem on parallel machines with distinct due dates. *European Journal of Operational Research*, v. 189, n. 3, p. 1305–1316, 2008. Cited By 26. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-40849083523&partnerID=40&md5=deff7c94cbf1af1506c09b3378d258da>. Citado 3 vezes nas páginas 23, 25 e 30.

KEDAD-SIDHOUM, S.; SOURD, F. Fast neighborhood search for the single machine earliness-tardiness scheduling problem. *Computers and Operations Research*, v. 37, n. 8, p. 1464–1471, 2010. Cited By 10. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-73449095721&partnerID=40&md5=ee004f5e4702ed266464a16224f19d2b>. Citado na página 29.

KIANFAR, K.; MOSLEHI, G. A branch-and-bound algorithm for single machine scheduling with quadratic earliness and tardiness penalties. *Computers and Operations Research*, v. 39, n. 12, p. 2978–2990, 2012. Cited By 3. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84862991965&partnerID=40&md5=0306c59105d2b828f37ef13eea90d1ce>. Citado 2 vezes nas páginas 23 e 28.

KOLAHAN, F.; LIANG, M. An adaptive TS approach to JIT sequencing with variable processing times and sequence-dependent setups. *European Journal of Operational Research*, v. 109, n. 1, p. 142–159, Agosto 1998. Citado na página 26.

KRAMER, A.; SUBRAMANIAN, A. A unified heuristic and an annotated bibliography for a large class of earliness-tardiness scheduling problems. *Computing Research Repository - CoRR*, abs/1509.02384, 2015. Disponível em: <http://arxiv.org/abs/1509.02384>. Citado 13 vezes nas páginas 30, 66, 69, 71, 96, 97, 98, 99, 100, 101, 102, 103 e 104.

KRAMER, A. H. F. *Um método heurístico para a resolução de uma classe de problemas de sequenciamento da produção envolvendo penalidades por antecipação e atraso*. 144 f. Dissertação (Mestrado em Engenharia de Produção) — Universidade Federal da Paraíba, João Pessoa - PB, 2015. Citado na página 2.

KRAVCHENKO, S. A.; WERNER, F. Parallel machine problems with equal processing times: a survey. *J. Scheduling*, v. 14, n. 5, p. 435–444, 2011. Citado na página 25.

LAUFF, V.; WERNER, F. Scheduling with common due date, earliness and tardiness penalties for multimachine problems: a survey. *Mathematical and Computer Modelling*, v. 40, n. 5-6, p. 637 – 655, 2004. Citado na página 25.

LAWLER, E. L. et al. Sequencing and scheduling: algorithms and complexity. In: GRAVES, S. C.; KAN, A. H. G. R.; ZIPKIN, P. H. (Ed.). *Logistics of Production and Inventory*. Elsevier, 1993, (Handbooks in Operations Research and Management Science, v. 4). cap. 9, p. 445–522. Disponível em: <http://www.sciencedirect.com/science/article/B7P6G-4FKY22V-1B/2/e636a20dd1005f17be8e9704711ac821>. Citado na página 74.

LEUNG, J.; KELLY, L.; ANDERSON, J. H. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. Boca Raton, FL, USA: CRC Press, Inc., 2004. ISBN 1584883979. Citado 2 vezes nas páginas 2 e 14.

LI, G. Single machine earliness and tardiness scheduling. *European Journal of Operational Research*, v. 96, n. 3, p. 546 – 558, 1997. Citado na página 27.

LIAW, C.-F. A branch-and-bound algorithm for the single machine earliness and tardiness scheduling problem. *Computers and Operations Research*, v. 26, n. 7, p. 679–693, 1999. Cited By 76. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-0033165933&partnerID=40&md5=bacd99dadeaeed0d188415ec370ad1ed>. Citado 2 vezes nas páginas 23 e 28.

LIU, L.; ZHOU, H. Hybridization of harmony search with variable neighborhood search for restrictive single-machine earliness/tardiness problem. *Information Sciences*, v. 226, p. 68–92, 2013. Cited By 7. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84871771269&partnerID=40&md5=35b9ad6f26812f87b79ee2641ab26f74>. Citado na página 29.

LIU, N.; ABDELRAHMAN, M.; RAMASWAMY, S. A Genetic Algorithm for Single Machine Total Weighted Tardiness Scheduling Problem. *International Journal of Intelligent Control and Systems*, v. 10, p. 218–225, Setembro 2005. ISSN 0218-7965. Citado 2 vezes nas páginas 48 e 52.

MASON, S.; JIN, S.; JAMPANI, J. A moving block heuristic for minimizing earliness and tardiness on a single machine with unrestrictive common due dates. *Journal of Manufacturing Systems*, v. 24, n. 4, p. 328–338, 2005. Cited By 2. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-33947179014&partnerID=40&md5=786f76799a209c25c3041de2d591ae2e>. Citado na página 29.

M'HALLAH, R. Minimizing total earliness and tardiness on a single machine using a hybrid heuristic. *Computers and Operations Research*, v. 34, n. 10, p. 3126–3142, 2007. Citado 3 vezes nas páginas 16, 22 e 27.

MONDAL, S.; SEN, A. Single machine weighted earlines-tardiness penalty problem with a common due date. *Computers and Operations Research*, v. 28, n. 7, p. 649–669, 2001. Cited By 25. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-0035371947&partnerID=40&md5=6f36d8fef1bd90a874d3629b334ab473>. Citado 2 vezes nas páginas 23 e 28.

MORAES, E. d. O.; NOGUEIRA, R. J. B. Corporação em rede: um estudo sobre a empresa moto honda da amazônia. *CaderNAU – Cadernos do Núcleo de Análises Urbanas*, v. 7, p. 112–129, 2014. Disponível em: <http://www.seer.furg.br/cnau/article/view/4835>. Citado 3 vezes nas páginas 17, 18 e 19.

NEARCHOU, A. *An Efficient Meta-Heuristic for the Single Machine Common Due Date Scheduling Problem*. [s.n.], 2006. 431-435 p. Cited By 0. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84882903500&partnerID=40&md5=262048c897f28eedf8b3f937588f10d4>. Citado na página 29.

PENNA, P. H. V. et al. Uma heurística híbrida para minimizar custos com antecipação e atraso do sequenciamento da produção em uma máquina. *Produção*, Scielo Brasil, v. 22, n. 4, p. 766–777, Dezembro 2012. Citado 2 vezes nas páginas 24 e 27.

PESSOA, A. et al. Exact algorithm over an arc-time-indexed formulation for parallel machine scheduling problems. *Mathematical Programming Computation*, Springer-Verlag, v. 2, n. 3-4, p. 259–290, 2010. ISSN 1867-2949. Disponível em: <http://dx.doi.org/10.1007/s12532-010-0019-z>. Citado 6 vezes nas páginas 26, 39, 41, 44, 45 e 57.

PINEDO, M. L. *Scheduling: Theory, Algorithms, and Systems*. 4th. ed. [S.l.]: Springer Publishing Company, 2012. ISBN 0387789340, 9780387789347. Citado 2 vezes nas páginas 2 e 7.

PRITSKER, A.; WATTERS, L.; WOLFE, P. *Multiproject Scheduling with Limited Resources: A Zero-one Programming Approach*. Rand Corporation, 1968. (P (Rand Corporation), Nº 3800). Disponível em: <http://books.google.com.br/books?id=aU8xOAAACAAJ>. Citado na página 36.

QIN, T. et al. Iterated local search based on multi-type perturbation for single-machine earliness/tardiness scheduling. *Computers & Operations Research*, v. 61, n. 0, p. 81 – 88, 2015. ISSN 0305-0548. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0305054815000635>. Citado na página 29.

RABADI, G.; MOLLAGHASEMI, M.; ANAGNOSTOPOULOS, G. A branch-and-bound algorithm for the early/tardy machine scheduling problem with a common due-date and sequence-dependent setup time. *Computers and Operations Research*, v. 31, n. 10, p. 1727–1751, 2004. Cited By 42. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-1842762099&partnerID=40&md5=e6fb90695d7637b1517cf4f3acdcf934>. Citado 2 vezes nas páginas 23 e 28.

RABADI, G.; MORAGA, R. J.; AL-SALEM, A. Heuristics for the unrelated parallel machine scheduling problem with setup times. *Journal of Intelligent Manufacturing*, Kluwer Academic Publishers, v. 17, p. 85–97, 2006. ISSN 0956-5515. Disponível em: <http://dx.doi.org/10.1007/s10845-005-5514-0>. Citado na página 34.

REBAI, M.; KACEM, I. Minimizing the earliness-tardiness costs on a single machine. In: *Service systems and service management, on 2008 international conference*. [S.l.: s.n.], 2008. p. 1 –5. Citado 2 vezes nas páginas 23 e 27.

RIBEIRO, F. et al. An adaptive genetic algorithm to solve the single machine scheduling problem with earliness and tardiness penalties. In: *Evolutionary Computation (CEC), 2010 IEEE Congress on*. [S.l.: s.n.], 2010. p. 1–8. Citado na página 29.

RONCONI, D.; KAWAMURA, M. The single machine earliness and tardiness scheduling problem: Lower bounds and a branch-and-bound algorithm. *Computational and Applied Mathematics*, v. 29, n. 2, p. 107–124, 2010. Cited By 3. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-77956526909&partnerID=40&md5=28dc2a7adeca16074903ab7576a3d5a0>. Citado na página 28.

RUNGE, N.; SOURD, F. A new model for the preemptive earliness-tardiness scheduling problem. *Computers and Operations Research*, v. 36, n. 7, p. 2242–2249, 2009. Cited By 8. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-57649224341&partnerID=40&md5=8d402ac427d7a55ca6541104a0dc6417>. Citado na página 29.

SHABTAY, D. Optimal restricted due date assignment in scheduling. *European Journal of Operational Research*, v. 252, n. 1, p. 79 – 89, 2016. ISSN 0377-2217. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0377221715011820>. Citado na página 16.

SHABTAY, D.; STEINER, G. Scheduling to maximize the number of just-in-time jobs: A survey. *Springer Optimization and Its Applications*, v. 60, p. 3–20, 2012. Citado 2 vezes nas páginas 16 e 30.

SOURD, F. Earliness-tardiness scheduling with setup considerations. *Computers and Operations Research*, v. 32, n. 7, p. 1849–1865, 2005. Cited By 29. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-9644302529&partnerID=40&md5=d3806a52628a43c656e5f60115b339de>. Citado na página 28.

SOURD, F. Dynasearch for the earliness-tardiness scheduling problem with release dates and setup constraints. *Operations Research Letters*, v. 34, n. 5, p. 591 – 598, 2006. ISSN 0167-6377. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167637705001008>. Citado 2 vezes nas páginas 24 e 26.

SOURD, F. New exact algorithms for one-machine earliness-tardiness scheduling. *INFORMS J. on Computing*, INFORMS, Institute for Operations Research and the Management Sciences (INFORMS), Linthicum, Maryland, USA, v. 21, n. 1, p. 167–175, Janeiro 2009. ISSN 1526-5528. Disponível em: <http://dx.doi.org/10.1287/ijoc.1080.0287>. Citado na página 26.

SOURD, F.; KEDAD-SIDHOUM, S. The one-machine problem with earliness and tardiness penalties. *Journal of Scheduling*, Kluwer Academic Publishers, v. 6, p. 533–549, 2003. ISSN 1094-6136. Disponível em: <http://dx.doi.org/10.1023/A%3A1026224610295>. Citado 2 vezes nas páginas 22 e 26.

SOURD, F.; KEDAD-SIDHOUM, S. A faster branch-and-bound algorithm for the earliness-tardiness scheduling problem. *Journal of Scheduling*, v. 11, n. 1, p. 49–58, 2008. Cited By 22. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-38349141886&partnerID=40&md5=e96977a29fdf79dd4f232b4a4e664fa9>. Citado 3 vezes nas páginas 2, 22 e 27.

SOUSA, J. P.; WOLSEY, L. A. A time indexed formulation of non-preemptive single machine scheduling problems. *Math. Program.*, v. 54, p. 353–367, 1992. Disponível em: <http://dblp.uni-trier.de/db/journals/mp/mp54.html#SousaW92>. Citado na página 36.

STORN, R.; PRICE, K. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, Kluwer Academic Publishers, v. 11, n. 4, p. 341–359, 1997. ISSN 0925-5001. Disponível em: <http://dx.doi.org/10.1023/A%3A1008202821328>. Citado na página 29.

TALEBI, J. et al. An efficient scatter search algorithm for minimizing earliness and tardiness penalties in a single-machine scheduling problem with a common due date. *2009 IEEE Congress on Evolutionary Computation, CEC 2009*, p. 1012–1018, 2009. Cited By 1. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-70449889882&partnerID=40&md5=c35f87cffd230c9248fd4ae034a5a070>. Citado na página 29.

TAMAKI, H.; MURAO, H.; KITAMURA, S. A heuristic-based hybrid solution for parallel machine scheduling problems with earliness and tardiness penalties. v. 2, p. 239–244 vol.2, Sept 2003. Citado na página 30.

TANAKA, S. *Benchmark instances for the single-machine total weighted earliness-tardiness problem*. 2012. Disponível em: http://turbine.kuee.kyoto-u.ac.jp/∼tanaka/index-e.html. Disponível em: <http://turbine.kuee.kyoto-u.ac.jp/~tanaka/index-e.html>. Citado na página 64.

TANAKA, S. An exact algorithm for the single-machine earliness-tardiness scheduling problem. *Springer Optimization and Its Applications*, v. 60, p. 21–40, 2012. Citado 12 vezes nas páginas 2, 4, 23, 26, 63, 67, 68, 90, 91, 92, 93 e 94.

TANAKA, S.; FUJIKUMA, S. An efficient exact algorithm for general single-machine scheduling with machine idle time. *4th IEEE Conference on Automation Science and Engineering, CASE 2008*, p. 371–376, 2008. Cited By 4. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-54949151972&partnerID=40&md5=392066331ca235df4673698414f5ac35>. Citado 2 vezes nas páginas 23 e 26.

TANAKA, S.; FUJIKUMA, S.; ARAKI, M. An exact algorithm for single-machine scheduling without machine idle time. *Journal of Scheduling*, v. 12, n. 6, p. 575–593, 2009. Cited By 15. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-72249099711&partnerID=40&md5=7230362c9f735e4995b3bf82fd5993fc>. Citado 4 vezes nas páginas 23, 26, 36 e 44.

TANAKA, S.; SASAKI, T.; ARAKI, M. A branch-and-bound algorithm for the single-machine weighted earliness-tardiness scheduling problem with job independent weights. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, v. 2, p. 1571–1577, 2003. Cited By 3. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-0242660378&partnerID=40&md5=08f760845d564173aacfe1195c94b8ed>. Citado 2 vezes nas páginas 23 e 27.

TANAKA, S.; SATO, S. An exact algorithm for the precedence-constrained single-machine scheduling problem. *European Journal of Operational Research*, v. 229, n. 2, p. 345–352, 2013. Cited By 0. Disponível em: <http://www.scopus.com/inward/record.url?

eid=2-s2.0-84876960742&partnerID=40&md5=30b2b4aa64e450f2e13198ef01bfcc60>. Citado 2 vezes nas páginas 23 e 28.

TASGETIREN, M. F. et al. A discrete differential evolution algorithm for the total earliness and tardiness penalties with a common due date on a single-machine. In: *Computational Intelligence in Scheduling, 2007. SCIS '07. IEEE Symposium on*. [S.l.: s.n.], 2007. p. 271–278. Citado na página 29.

TSAI, C.-Y.; WANG, Y.-C. The sum of earliness and tardiness minimization on unrelated parallel machines with inserted idle time. *Latest Advances in Information Science and Applications*, WSEAS Press, p. 125–130, Maio 2012. ISSN 1790-5109. Disponível em: <http://www.wseas.us/e-library/conferences/2012/Singapore/ACCIDS/ACCIDS-20.pdf>. Citado 3 vezes nas páginas 34, 36 e 44.

TUONG, N. H.; SOUKHAL, A. Due dates assignment and jit scheduling with equal-size jobs. *European Journal of Operational Research*, v. 205, n. 2, p. 280–289, 2010. Cited By 9. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-76949092507&partnerID=40&md5=d48cfabff641d300410dde5eafaa299b>. Citado na página 16.

VERMA, S.; DESSOUKY. Single-machine scheduling of unit-time jobs with earliness and tardiness penalties. *Mathematics of Operations Research*, v. 23, p. 930 – 943, 1998. Citado na página 16.

WAN, L.; YUAN, J. Single-machine scheduling to minimize the total earliness and tardiness is strongly NP-hard. *Operations Research Letters*, v. 41, n. 4, p. 363–365, 2013. Cited By 2. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84877620879&partnerID=40&md5=ad0cf381dd32e410563e2b656aaef595>. Citado na página 16.

WANG, L.; WANG, M. Hybrid algorithm for earliness-tardiness scheduling problem with sequence dependent setup time. *Proceedings of the IEEE Conference on Decision and Control*, v. 2, p. 1219–1222, 1997. Cited By 8. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-0031348249&partnerID=40&md5=a6d9b941078844b50356f6fa48ece3e3>. Citado na página 29.

WENG, W.; FUJIMURA, S. Self evolution algorithm for common due date scheduling problem. *4th IEEE Conference on Automation Science and Engineering, CASE 2008*, p. 790–795, 2008. Cited By 1. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-54949123078&partnerID=40&md5=1e1d80f3aca9867ebcd4d20383de0e92>. Citado na página 29.

WODECKI, M. A block approach to earliness-tardiness scheduling problems. *International Journal of Advanced Manufacturing Technology*, v. 40, n. 7-8, p. 797–807, 2009. Cited By 8. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-59249094242&partnerID=40&md5=30bd4d09252c3935069f11a0aa331d0b>. Citado na página 29.

XIAO, W.-Q.; LI, C.-L. Approximation algorithms for common due date assignment and job scheduling on parallel machines. *IIE Transactions*, Springer Netherlands, v. 34, p. 467–477, 2002. Citado 2 vezes nas páginas 14 e 29.

YOUSEFI, M.; YUSUFF, R. M. *Minimizing Earliness and Tardiness Penalties in a Single Machine Scheduling Against Common Due Date using Genetic Algorithm*. [S.l.]: Maxwell Scientific Organization, 2012. v. 9. 1205-1210 p. ISBN 2040-7467. Citado 2 vezes nas páginas 24 e 27.

YU, H. et al. Genetic algorithm for single machine scheduling with general early-tardy penalty weights. *Proceedings of the American Control Conference*, v. 2, p. 885–889, 1999. Cited By 0. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-0033283332&partnerID=40&md5=247b2e74c5d627ff526d145020dc6a6a>. Citado na página 29.

ZHAO, C. et al. Single-machine scheduling and due date assignment with rejection and position-dependent processing times. *Journal of Industrial and Management Optimization*, v. 10, n. 3, p. 691–700, 2014. Cited By 6. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84890069031&partnerID=40&md5=a3941fa9727b3f5a071a5a28132e2147>. Citado na página 16.

Apêndice A

Resultados detalhados para

$1||\sum \alpha_j E_j + \sum \beta_j T_j$

Resultados detalhados para o escalonamento com penalidades de antecipação e atraso em ambiente monoprocessado (ver Tabelas A.1-A.5).

Tabela A.1 Resultados para o problema $1||\sum \alpha_j E_j + \sum \beta_j T_j$ com 40 tarefas e 1 máquina.

Inst.	Média custo GLS	Média custo PI	Média tempo GLS	Média tempo PI	Tempo mín.	Média tempo	Tempo máx.	Melhor iter. GLS	Melhor tempo GLS	Melhor arcos PI	Melhor tempo PI	Melhor GLS	MathGLS-IP	Tanaka (2012b)	MSC
1	54640.000	54640.000	4.124	0.063	4.082	4.186	4.317	0	0.066	40	0.006	54640	54640	54640	54640
6	40555.000	40555.000	3.677	0.045	3.581	3.722	3.987	0	0.005	40	0.004	40555	40555	40555	40555
11	29924.000	29924.000	3.774	0.035	3.697	3.809	4.065	0	0.006	40	0.003	29924	29924	29924	29924
16	73684.000	73684.000	3.644	0.025	3.586	3.669	3.752	0	0.006	40	0.003	73684	73684	73684	73684
21	77819.000	77819.000	3.295	0.021	3.237	3.316	3.436	0	0.015	40	0.005	77819	77819	77819	77819
26	51783.000	51783.000	3.662	0.051	3.539	3.713	3.986	0	0.004	40	0.005	51783	51783	51783	51783
31	23291.000	23291.000	4.565	0.160	4.527	4.725	5.333	0	0.007	40	0.003	23291	23291	23291	23291
36	25496.000	25496.000	4.455	0.109	4.374	4.564	5.184	0	0.010	40	0.078	25496	25496	25496	25496
41	59093.000	59093.000	4.521	0.153	4.474	4.674	5.532	0	0.005	40	0.071	59093	59093	59093	59093
46	64486.000	64486.000	3.450	0.718	3.425	4.168	9.647	0	0.004	40	0.175	64486	64486	64486	64486
51	47667.000	47667.000	4.771	0.066	4.633	4.837	5.047	0	0.006	40	0.007	47667	47667	47667	47667
56	27953.000	27953.000	5.007	0.112	4.975	5.119	5.386	0	0.013	40	0.043	27953	27953	27953	27953
61	21737.000	21737.000	4.592	0.163	4.501	4.755	5.248	0	0.018	40	0.307	21737	21737	21737	21737
66	65764.000	65764.000	3.788	0.347	3.614	4.135	6.571	0	0.023	40	0.029	65764	65764	65764	65764
71	90521.000	90521.000	3.434	0.138	3.309	3.572	4.306	0	0.004	40	0.175	90521	90521	90521	90521
76	43127.000	43127.000	4.478	0.026	4.277	4.504	4.897	0	0.013	40	0.003	43127	43127	43127	43127
81	12552.000	12552.000	3.721	0.026	3.519	3.748	3.928	0	0.082	40	0.004	12552	12552	12552	12552
86	15918.000	15918.000	4.706	0.025	4.384	4.731	5.046	0	0.013	40	0.022	15918	15918	15918	15918
91	48311.000	48311.000	5.010	0.029	4.945	5.039	5.162	0	0.007	40	0.004	48311	48311	48311	48311
96	126129.000	126129.000	3.301	0.028	3.276	3.329	3.405	0	0.005	40	0.004	126129	126129	126129	126129
101	85961.000	85961.000	4.067	0.025	4.038	4.092	4.199	0	0.009	40	0.004	85961	85961	85961	85961
106	18182.000	18182.000	4.401	0.042	4.271	4.443	4.619	0	0.017	40	0.085	18182	18182	18182	18182
111	32374.000	32374.000	3.916	0.039	3.846	3.955	4.113	0	0.006	40	0.024	32374	32374	32374	32374
116	47411.000	47411.000	4.200	0.037	4.117	4.238	4.488	0	0.015	40	0.003	47411	47411	47411	47411
121	122538.000	122538.000	4.822	0.021	4.374	4.843	5.817	0	0.011	40	0.004	122538	122538	122538	122538

Tabela A.2 Resultados para o problema $1||\sum \alpha_j E_j + \sum \beta_j T_j$ com 50 tarefas e 1 máquina.

Inst.	Média custo GLS	Média custo PI	Média tempo GLS	Média tempo PI	Tempo mín.	Média tempo	Tempo máx.	Melhor iter. GLS	Melhor tempo GLS	Melhor arcos PI	Melhor tempo PI	Melhor GLS	MathGLS-IP	Tanaka (2012b)	MSC
1	130548.000	130548.000	6.205	0.035	6.029	6.240	6.455	0	0.031	65	0.033	130548	130548	130548	130548
6	81229.000	81229.000	6.844	0.035	6.362	6.880	7.882	0	0.011	64	0.024	81229	81229	81229	81229
11	54167.000	54167.000	7.819	0.059	7.374	7.878	8.189	0	0.008	60	0.152	54167	54167	54167	54167
16	91507.000	91507.000	7.535	0.071	7.329	7.606	7.980	0	0.107	56	0.019	91507	91507	91507	91507
21	214555.000	214555.000	6.295	0.036	6.104	6.331	6.615	0	0.109	63	0.003	214555	214555	214555	214555
26	81285.000	81285.000	9.009	0.077	8.804	9.086	9.582	0	0.009	57	0.037	81285	81285	81285	81285
31	37565.000	37565.000	7.275	0.047	6.930	7.322	7.782	0	0.151	62	0.011	37565	37565	37565	37565
36	48923.000	48923.000	7.131	0.087	6.976	7.218	7.573	0	0.103	62	0.070	48923	48923	48923	48923
41	71949.000	71949.000	7.008	0.036	6.840	7.044	7.279	1	0.415	60	0.002	71949	71949	71949	71949
46	157588.000	157588.000	6.981	0.060	6.920	7.040	7.228	0	0.008	60	0.026	157588	157588	157588	157588
51	56208.000	56208.000	7.549	0.029	7.451	7.577	7.783	0	0.008	57	0.026	56208	56208	56208	56208
56	37492.000	37492.000	7.373	0.047	6.930	7.420	7.880	2	0.565	54	0.003	37492	37492	37492	37492
61	34640.000	34640.000	8.119	0.052	7.897	8.172	8.401	1	0.461	66	0.003	34640	34640	34640	34640
66	77080.000	77080.000	8.387	0.030	8.217	8.416	8.649	0	0.008	50	0.003	77080	77080	77080	77080
71	150978.000	150978.000	11.102	0.064	10.747	11.166	11.501	0	0.007	60	0.006	150978	150978	150978	150978
76	103652.000	103652.000	7.401	0.049	6.716	7.450	8.512	0	0.033	69	0.024	103652	103652	103652	103652
81	17433.000	17433.000	7.309	0.025	7.124	7.334	7.549	0	0.145	64	0.004	17433	17433	17433	17433
86	22733.000	22733.000	8.454	0.041	8.110	8.495	8.814	0	0.047	55	0.054	22733	22733	22733	22733
91	89715.000	89715.000	7.199	0.050	7.065	7.249	7.349	0	0.192	63	0.140	89715	89715	89715	89715
96	178101.000	178101.000	7.437	0.056	6.452	7.493	8.877	0	0.015	63	0.036	178101	178101	178101	178101
101	90880.000	90880.000	7.120	0.031	6.932	7.151	7.228	0	0.058	55	0.028	90880	90880	90880	90880
106	23348.000	23348.000	7.934	0.028	7.743	7.962	8.096	0	0.008	58	0.011	23348	23348	23348	23348
111	30170.000	30170.000	7.165	0.053	6.957	7.218	7.613	0	0.147	62	0.022	30170	30170	30170	30170
116	36704.000	36704.000	9.488	0.044	8.958	9.532	10.253	0	0.017	50	0.005	36704	36704	36704	36704
121	79007.000	79007.000	8.563	0.028	8.275	8.591	8.900	0	0.011	56	0.004	79007	79007	79007	79007

Tabela A.3 Resultados para o problema $1||\sum \alpha_j E_j + \sum \beta_j T_j$ com 100 tarefas e 1 máquina.

Inst.	Média custo GLS	Média custo PI	Média tempo GLS	Média tempo PI	Tempo mín.	Média tempo	Tempo máx.	Melhor iter. GLS	Melhor tempo GLS	Melhor arcos PI	Melhor tempo PI	Melhor GLS	MathGLS-IP	Tanaka (2012b)	MSC
1	405122.000	405122.000	62.901	0.059	57.522	62.960	66.386	20	30.886	514	0.007	405122	405122	405122	405122
6	197257.000	197257.000	66.531	0.049	62.792	66.581	72.585	2	4.263	498	0.007	197257	197257	197257	197257
11	252623.000	252623.000	64.855	0.052	59.342	64.907	69.233	14	23.570	482	0.009	252623	252623	252623	252623
16	419102.000	419102.000	81.650	0.041	79.294	81.691	83.153	0	0.077	509	0.016	419102	419102	419102	419102
21	898927.000	898927.000	71.788	0.039	70.094	71.828	72.684	0	0.054	553	0.005	898927	898927	898927	898927
26	408981.000	408981.000	78.010	0.048	76.704	78.058	78.651	0	0.227	547	0.005	408981	408981	408981	408981
31	193480.000	193480.000	48.432	0.049	45.649	48.481	52.039	0	0.084	512	0.004	193480	193480	193480	193480
36	149717.800	149717.800	58.100	0.047	52.915	58.147	64.773	11	19.717	490	0.008	149702	149702	149702	149702
41	463554.000	463554.000	77.455	0.040	76.976	77.495	78.618	0	1.602	572	0.005	463554	463554	463554	463554
46	829912.600	829912.600	70.078	0.044	69.699	70.121	70.558	0	0.073	526	0.008	829911	829911	829911	829911
51	444991.000	444991.000	85.273	0.047	80.394	85.319	88.767	0	0.269	514	0.004	444991	444991	444991	444991
56	122064.000	122064.000	77.758	0.044	71.104	77.802	82.568	0	1.388	495	0.006	122064	122064	122064	122064
61	102185.000	102185.000	98.338	0.048	91.556	98.387	105.405	2	6.729	471	0.005	102185	102185	102185	102185
66	244706.000	244706.000	82.893	0.048	80.167	82.941	87.073	5	11.555	443	0.006	244706	244706	244706	244706
71	640932.000	640932.000	72.461	0.041	72.145	72.502	73.284	0	0.177	511	0.045	640932	640932	640932	640932
76	312589.000	312589.000	93.122	0.047	89.435	93.169	95.699	0	2.482	483	0.020	312589	312589	312589	312589
81	47629.000	47629.000	63.817	0.046	57.503	63.863	68.402	1	2.876	463	0.007	47629	47629	47629	47629
86	74338.300	74338.300	89.549	0.052	84.887	89.601	94.740	7	19.958	504	0.009	74338	74338	74338	74338
91	249743.000	249743.000	81.302	0.046	80.368	81.348	81.890	0	2.467	474	0.006	249743	249743	249743	249743
96	495812.000	495812.000	76.649	0.035	76.120	76.684	77.522	0	1.046	479	0.004	495812	495812	495812	495812
101	356397.000	356397.000	89.597	0.053	87.895	89.650	91.181	3	7.283	470	0.010	356397	356397	356397	356397
106	114084.000	114084.000	83.369	0.050	80.864	83.418	84.599	1	4.095	496	0.005	114084	114084	114084	114084
111	161691.000	161691.000	89.104	0.048	87.982	89.152	89.857	3	8.559	518	0.004	161691	161691	**161642**	161642
116	370918.000	370918.000	84.475	0.041	83.882	84.516	85.148	0	1.554	484	0.005	370918	370918	370918	370918
121	471766.900	471762.800	85.353	0.047	84.730	85.399	86.127	28	60.050	498	0.006	471761	471761	471761	471761

Tabela A.4 Resultados para o problema $1||\sum \alpha_j E_j + \sum \beta_j T_j$ com 200 tarefas e 1 máquina.

Inst.	Média custo GLS	Média custo PI	Média tempo GLS	Média tempo PI	Tempo mín.	Média tempo	Tempo máx.	Melhor iter. GLS	Melhor tempo GLS	Melhor arcos PI	Melhor tempo PI	Melhor GLS	MathGLS-IP	Tanaka (2012b)	MSC
1	1508466.000	1508466.000	544.817	0.169	498.253	544.986	604.231	17	254.804	3975	0.022	1508466	1508466	1508466	1508466
6	1003138.000	1003138.000	523.217	0.149	506.705	523.366	549.587	3	47.290	4133	0.009	1003138	1003138	1003138	1003138
11	700455.600	700455.600	528.165	0.166	482.344	528.331	574.555	8	102.199	3661	0.011	700455	700455	700455	700455
16	1661656.000	1661656.000	624.834	0.141	591.500	624.974	653.220	1	19.198	4241	0.008	1661656	1661656	1661656	1661656
21	2381334.000	2381334.000	569.477	0.146	551.854	569.623	581.162	23	319.065	3994	0.008	2381330	2381330	2381330	2381330
26	1216262.000	1216262.000	638.475	0.143	624.964	638.618	647.807	15	229.733	3848	0.006	1216262	1216262	1216262	1216262
31	659204.800	659196.400	529.313	0.177	496.435	529.490	578.058	20	277.437	4178	0.010	659143	659143	659143	659143
36	655112.500	655112.500	665.698	0.152	617.690	665.849	710.098	6	92.822	4090	0.011	655055	655055	655055	655055
41	1375717.600	1375717.600	672.927	0.143	655.173	673.070	712.228	9	169.315	3906	0.024	1375693	1375693	1375693	1375693
46	2213475.200	2213475.200	613.178	0.149	587.912	613.327	642.236	9	141.663	4021	0.037	2213443	2213443	2213443	2213443
51	1508965.900	1508958.500	599.960	0.148	586.302	600.108	617.095	3	49.663	3814	0.009	1508957	1508957	1508957	1508957
56	442380.100	442350.400	545.859	0.160	511.370	546.018	572.177	19	261.450	4192	0.014	442088	442088	**442066**	442066
61	444916.000	444916.000	571.909	0.134	537.172	572.043	607.136	0	4.317	3956	0.004	444916	444916	444916	444916
66	1077908.100	1077869.100	593.287	0.151	554.106	593.438	632.655	39	556.302	4122	0.010	1077747	1077747	**1077545**	1077545
71	1710071.000	1710071.000	686.138	0.137	676.167	686.276	699.127	0	7.595	4096	0.007	1710071	1710071	1710071	1710071
76	1198536.000	1198508.600	670.397	0.140	652.072	670.537	687.184	9	156.890	3830	0.010	1198495	1198495	1198495	1198495
81	249999.900	249999.300	614.087	0.146	559.304	614.233	652.373	17	255.383	4188	0.009	249999	249999	249999	249999
86	283787.400	283777.000	672.157	0.136	627.660	672.293	706.861	28	412.423	3901	0.009	283765	283765	283765	283765
91	766694.900	766639.500	783.180	0.144	756.499	783.324	803.647	0	14.534	3887	0.008	766538	766538	766528	766528
96	1246339.700	1246339.600	653.722	0.149	624.220	653.871	689.506	22	370.757	3790	0.010	1246269	1246268	1246268	1246268
96	1246339.700	1246339.600	653.722	0.149	624.220	653.871	689.506	22	370.757	3790	0.010	1246269	1246268	1246268	1246268
101	1540505.000	1540505.000	751.902	0.147	737.318	752.049	763.197	19	358.515	3982	0.010	1540505	1540505	1540505	1540505
106	214955.100	214908.200	691.881	0.149	649.167	692.031	721.550	28	503.445	3968	0.011	214839	214759	214759	214759
111	102878.000	102878.000	778.523	0.134	747.072	778.658	808.505	1	22.705	3992	0.008	102878	102878	102878	102878
116	615275.400	615243.800	718.917	0.142	691.686	719.059	755.877	16	306.819	3890	0.010	615232	615232	615232	615232
121	973817.800	973807.300	703.295	0.141	682.576	703.437	729.868	14	248.669	3710	0.011	973771	973760	973760	973760

Tabela A.5 Resultados para o problema $1||\sum \alpha_j E_j + \sum \beta_j T_j$ com 300 tarefas e 1 máquina.

Inst.	Média custo GLS	Média custo PI	Média tempo GLS	Média tempo PI	Tempo mín.	Média tempo	Tempo máx.	Melhor iter. GLS	Melhor tempo GLS	Melhor arcos PI	Melhor tempo PI	Melhor GLS	MathGLS-IP	Tanaka (2012b)	MSC
1	3184403.200	3184403.200	1855.751	0.349	1789.516	1856.099	1938.393	17	830.899	13348	0.102	3184308	3184308	3184308	3184308
6	2171434.900	2171405.000	1725.421	0.323	1624.918	1725.744	1884.304	13	533.169	14005	0.032	2171259	2170960	**2170245**	2170245
11	2019080.800	2019080.800	1614.520	0.323	1497.428	1614.843	1787.591	21	966.443	13961	0.021	2018983	2018983	2018983	2018983
16	3373159.600	3373159.600	2239.746	0.289	2110.535	2240.035	2377.570	36	2008.194	13578	0.011	3373153	3373153	3373153	3373153
21	5744791.200	5744781.700	2065.325	0.317	2025.250	2065.642	2101.081	39	2049.172	13278	0.015	5744722	5744701	**5744651**	5744651
26	2879059.200	2879037.300	1936.794	0.299	1774.214	1937.093	2085.061	29	1384.179	13224	0.014	2879006	2879006	2879006	2879006
31	1366959.100	1366941.800	1984.679	0.323	1836.369	1985.002	2098.376	9	460.660	13714	0.012	1366741	1366741	1366741	1366741
36	1614219.900	1614173.900	1793.893	0.337	1677.066	1794.230	1949.167	27	1218.698	13400	0.016	1613335	1613335	**1613012**	1613012
41	3066643.400	3066580.000	1892.944	0.306	1805.029	1893.250	1967.722	39	1818.759	13006	0.016	3066261	3066261	**3066036**	3066036
46	4974550.900	4974461.500	2033.708	0.301	1960.108	2034.008	2081.471	34	1747.149	13612	0.014	4974477	4974336	4974336	4974336
51	2717846.300	2717806.200	2225.736	0.302	2133.786	2226.038	2317.724	33	1838.170	13173	0.013	2717719	2717673	**2717659**	2717659
56	887494.100	887438.600	2109.580	0.307	1946.265	2109.887	2220.660	21	1104.967	13572	0.009	887423	887423	887423	887423
61	933496.400	933358.900	1956.435	0.327	1816.875	1956.762	2044.972	24	1164.549	13645	0.020	932573	932564	**932495**	932495
66	2421574.400	2421544.900	2130.494	0.312	1941.300	2130.806	2299.561	4	298.408	13771	0.012	2421273	2421268	**2421261**	2421261
71	3766386.300	3766371.900	2195.188	0.288	2129.058	2195.476	2293.927	39	2247.776	13613	0.013	3766290	3766290	3766290	3766290
76	3272699.100	3272678.700	1975.425	0.295	1842.267	1975.720	2111.845	22	1156.526	13509	0.009	3272681	3272657	**3272613**	3272613
81	403264.100	403142.800	2062.669	0.310	1918.932	2062.980	2168.585	34	1788.965	13127	0.016	403105	403105	403105	403105
86	758028.700	758007.000	2167.019	0.290	2065.903	2167.308	2256.322	34	1846.941	12865	0.010	757993	757993	757993	757993
91	1826949.200	1826879.800	2117.874	0.301	2058.502	2118.175	2168.298	21	1092.860	14000	0.010	1826696	1826696	**1826552**	1826552
96	3013494.300	3013335.500	2017.161	0.313	1873.960	2017.475	2183.842	13	707.763	12960	0.013	3013357	3013110	**3013022**	3013022
101	3914715.100	3914634.300	2091.760	0.302	2017.458	2092.062	2159.578	9	500.217	13937	0.009	3914656	3914588	**3914559**	3914559
106	621873.300	621705.500	2429.205	0.304	2376.124	2429.509	2496.377	29	1793.001	13522	0.010	621592	621592	**621564**	621564
111	271307.200	271305.200	2411.257	0.285	2315.002	2411.542	2526.435	14	841.317	13515	0.010	271305	271305	271305	271305
116	1126412.700	1126223.700	2072.197	0.310	1990.426	2072.507	2164.266	30	1692.207	11741	0.014	1125841	1125741	**1124788**	1124788
121	2054401.200	2054401.200	1991.476	0.300	1887.482	1991.776	2090.996	37	1852.635	12744	0.011	2054397	2054397	2054397	2054397

Apêndice B

Resultados detalhados para

$P||\sum \alpha_j E_j + \sum \beta_j T_j$

Resultados detalhados para o escalonamento com penalidades de antecipação e atraso em máquinas paralelas idênticas (ver Tabelas B.1-B.18).

Tabela B.1 Resultados para o problema $P||\sum \alpha_j E_j + \sum \beta_j T_j$ com 40 tarefas e 2 máquinas.

Inst.	Média custo GLS	Média custo PI	Média tempo GLS	Média tempo PI	Tempo min	Média Tempo	Tempo máx.	Melhor iter. GLS	Melhor tempo GLS	Melhor arcos PI	Melhor tempo PI	Melhor GLS	MathGLS-IP	Kramer e Subramanian (2015)	*IP-NFlow*	*IP-NFlow* (GAP%)	MSC
1	26063.100	26063.100	5.673	0.022	5.497	5.695	6.146	3	0.450	47	0.005	26063	26063	26063	26063	0.000	26063
6	19805.000	19805.000	7.731	0.021	7.253	7.752	8.511	0	0.093	44	0.004	19805	19805	···	19805	0.000	19805
11	15451.000	15451.000	5.251	0.022	4.920	5.273	5.529	0	0.046	45	0.004	15451	15451	15451	15451	0.000	15451
16	39041.000	39041.000	8.606	0.025	8.202	8.631	8.968	0	0.051	46	0.003	39041	39041	···	39041	0.000	39041
21	41054.000	41054.000	6.298	0.018	5.894	6.315	6.833	0	0.012	41	0.003	41054	41054	41054	41054	0.000	41054
26	24497.000	24497.000	7.395	0.023	6.863	7.417	8.448	1	0.267	48	0.003	24497	24497	···	···	···	24497
31	11679.000	11679.000	5.395	0.018	5.178	5.413	5.761	0	0.017	52	0.002	11679	11679	11679	11679	0.000	11679
36	13166.000	13166.000	6.097	0.022	5.798	6.119	6.509	0	0.032	46	0.014	13166	13166	···	13166	0.000	13166
41	31678.000	31678.000	5.180	0.018	5.105	5.198	5.309	0	0.005	47	0.003	31678	31678	31678	31678	0.000	31678
46	34149.000	34149.000	5.319	0.020	4.670	5.338	5.885	0	0.005	41	0.004	34149	34149	···	**34148**	0.000	34148
51	20190.000	20190.000	8.193	0.020	7.957	8.213	8.951	0	0.108	56	0.003	20190	20190	20190	20190	12.370	20190
56	12820.000	12820.000	7.144	0.029	6.754	7.173	7.939	2	0.443	44	0.006	12820	12820	···	12820	0.000	12820
61	12472.000	12472.000	8.195	0.022	7.422	8.217	9.019	2	0.416	45	0.005	12472	12472	12472	12472	0.000	12472
66	35365.000	35365.000	5.546	0.020	5.276	5.565	5.816	0	0.007	47	0.005	35365	35365	···	35365	0.000	35365
71	47952.000	47952.000	8.524	0.020	8.183	8.544	9.120	0	0.007	41	0.003	47952	47952	47952	47952	0.000	47952
76	14425.000	14425.000	5.482	0.023	5.101	5.506	6.963	0	0.158	45	0.003	14425	14425	···	···	···	14425
81	5278.000	5278.000	5.559	0.020	5.286	5.579	6.019	5	0.807	42	0.002	5278	5278	5278	5278	0.000	5278
86	8399.000	8399.000	7.360	0.024	7.121	7.383	7.663	0	0.093	44	0.005	8399	8399	···	8399	0.000	8399
91	26309.000	26309.000	7.247	0.020	6.810	7.267	7.724	0	0.076	47	0.003	26309	26309	26309	26309	0.000	26309
96	66141.000	66141.000	6.309	0.022	6.160	6.330	6.746	0	0.021	48	0.004	66141	66141	···	66141	0.000	66141
101	22413.000	22413.000	4.551	0.021	4.421	4.572	4.756	0	0.025	51	0.003	22413	22413	···	22413	34.120	22413
106	4629.000	4629.000	5.935	0.024	5.362	5.959	6.444	5	0.828	49	0.004	4629	4629	···	···	···	4629
111	18495.000	18495.000	5.016	0.024	4.850	5.040	5.426	0	0.089	50	0.005	18493	18493	18493	18493	0.000	18493
116	26314.000	26314.000	5.434	0.030	5.176	5.464	5.849	6	0.851	44	0.004	26314	26314	···	26314	0.000	26314
121	64584.000	64584.000	6.522	0.019	6.181	6.541	6.963	0	0.035	50	0.003	64584	64584	64584	64584	0.000	64584

Tabela B.2 Resultados para o problema $P||\sum \alpha_j E_j + \sum \beta_j T_j$ com 40 tarefas e 4 máquinas.

Inst.	Média custo GLS	Média custo PI	Média tempo GLS	Média tempo PI	Tempo min	Média Tempo	Tempo máx.	Melhor iter. GLS	Melhor tempo GLS	Melhor arcos PI	Melhor tempo PI	Melhor GLS	MathGLS-IP	Kramer e Subramanian (2015)	*IP-NFlow*	*IP-NFlow* (GAP%)	MSC
1	11987.000	11987.000	8.376	0.053	8.145	8.429	8.761	7	1.438	58	0.007	11985	11985	11985	11985	0.000	11985
6	9363.000	9363.000	9.098	0.021	8.810	9.119	9.888	0	0.014	55	0.003	9363	9363	···	9363	0.000	9363
11	8206.000	8206.000	9.033	0.027	8.742	9.060	9.396	0	0.107	56	0.004	8206	8206	8206	8206	0.000	8206
16	22008.000	22008.000	8.646	0.018	8.011	8.665	9.134	0	0.031	57	0.004	22008	22008	···	22008	0.000	22008
21	22793.000	22793.000	7.648	0.020	7.377	7.668	7.954	0	0.090	51	0.003	22793	22793	22793	22793	0.000	22793
26	10801.400	10801.400	8.520	0.051	8.343	8.571	9.042	7	1.762	59	0.006	10801	10801	···	10801	0.000	10801
31	5950.000	5950.000	9.648	0.021	9.251	9.670	10.208	0	0.100	64	0.004	5950	5950	5950	5950	0.000	5950
36	7313.000	7313.000	11.221	0.025	10.228	11.246	15.836	0	0.035	56	0.004	7313	7313	···	7313	0.000	7313
41	18020.000	18020.000	11.418	0.018	11.148	11.436	11.733	0	0.035	58	0.004	18020	18020	18020	18020	0.000	18020
46	19124.000	19124.000	9.208	0.017	8.862	9.224	9.664	0	0.021	43	0.003	19124	19124	···	**19123**	0.000	19123
51	8360.000	8360.000	9.509	0.028	8.972	9.537	10.149	5	1.190	69	0.004	8360	8360	8360	8360	0.000	8360
56	5561.000	5561.000	9.030	0.026	8.721	9.057	9.456	3	0.868	54	0.003	5561	5561	···	5561	0.000	5561
61	7714.000	7714.000	9.568	0.019	9.100	9.587	10.021	0	0.172	55	0.003	7714	7714	7714	7714	0.000	7714
66	20300.000	20300.000	10.567	0.018	9.945	10.584	10.973	0	0.010	58	0.006	20300	20300	···	20300	0.000	20300
71	26740.000	26740.000	8.014	0.020	7.682	8.034	8.524	0	0.124	50	0.003	26740	26740	26740	26740	0.000	26740
76	5086.000	5086.000	8.958	0.021	8.012	8.979	9.889	0	0.159	55	0.004	5086	5086	···	5086	0.000	5086
81	2181.000	2181.000	10.680	0.022	9.973	10.703	11.185	2	0.790	52	0.003	2181	2181	2181	2181	0.000	2181
86	5611.000	5611.000	9.042	0.020	8.590	9.063	9.438	0	0.218	54	0.003	5611	5611	···	5611	0.000	5611
91	15678.000	15678.000	8.651	0.028	8.329	8.679	9.330	0	0.181	58	0.004	15678	15678	15678	15678	0.000	15678
96	36271.000	36271.000	7.351	0.025	7.152	7.376	7.649	0	0.031	59	0.003	36271	36271	···	36271	0.000	36271
101	6306.000	6306.000	9.021	0.026	8.739	9.047	9.475	3	0.708	62	0.005	6306	6306	···	6306	0.000	6306
106	1855.000	1855.000	10.943	0.022	10.496	10.965	11.552	2	0.493	61	0.004	1855	1855	···	1855	0.000	1855
111	11505.000	11505.000	10.643	0.024	10.182	10.667	10.781	0	0.053	61	0.004	11505	11505	11505	11505	0.000	11505
116	15694.000	15694.000	8.751	0.021	8.459	8.772	9.222	1	0.268	54	0.003	15694	15694	···	15694	0.000	15694
121	35783.000	35783.000	7.359	0.018	7.112	7.378	7.612	0	0.037	61	0.004	35783	35783	35783	35783	0.000	35783

Tabela B.3 Resultados para o problema $P||\sum \alpha_j E_j + \sum \beta_j T_j$ com 40 tarefas e 10 máquinas.

Inst.	Média custo GLS	Média custo PI	Média tempo GLS	Média tempo PI	Tempo min	Média Tempo	Tempo máx.	Melhor iter. GLS	Melhor tempo GLS	Melhor arcos PI	Melhor tempo PI	Melhor GLS	MathGLS-IP	Kramer e Subramanian (2015)	*IP-NFlow*	*IP-NFlow* (GAP%)	MSC
1	3988.000	3988.000	10.815	0.029	8.549	10.845	11.941	1	1.123	72	0.007	3988	3988	3988	3988	0.000	3988
6	3841.000	3841.000	9.649	0.025	7.949	9.674	10.427	0	0.274	68	0.003	3841	3841	···	3841	0.000	3841
11	5044.000	5044.000	8.053	0.030	6.737	8.083	8.930	0	0.229	69	0.003	5044	5044	···	5044	0.000	5044
16	12114.000	12114.000	9.689	0.022	8.203	9.711	10.454	0	0.035	71	0.003	12114	12114	···	12114	0.000	12114
21	12062.000	12062.000	7.752	0.020	6.112	7.773	8.607	0	0.119	63	0.003	12062	12062	···	12062	0.000	12062
26	3326.000	3326.000	10.501	0.030	8.499	10.531	11.194	2	1.569	74	0.005	3326	3326	···	3326	0.000	3326
31	3558.000	3558.000	10.939	0.021	8.851	10.961	11.774	0	0.143	80	0.003	3558	3558	3558	3558	0.000	3558
36	5392.000	5392.000	9.824	0.021	7.971	9.845	10.800	0	0.161	70	0.003	5392	5392	···	5392	0.000	5392
41	10624.000	10624.000	8.535	0.023	7.087	8.558	9.200	0	0.312	72	0.003	10624	10624	···	10624	0.000	10624
46	10348.000	10348.000	7.786	0.021	5.631	7.807	9.001	0	0.027	51	0.003	10348	10348	···	10348	0.000	10348
51	2878.200	2878.200	11.245	0.032	9.199	11.277	12.586	7	3.821	86	0.006	2877	2877	···	2877	0.000	2877
56	1797.000	1797.000	10.829	0.024	7.799	10.853	11.829	1	1.133	67	0.003	1797	1797	···	1797	0.000	1797
61	5985.000	5985.000	9.967	0.020	8.233	9.987	11.351	0	0.203	68	0.003	5985	5985	5985	5985	0.000	5985
66	11685.000	11685.000	8.210	0.021	5.841	8.232	9.153	0	0.234	72	0.003	11685	11685	···	11685	0.000	11685
71	14290.000	14290.000	7.210	0.020	5.541	7.230	7.820	0	0.085	62	0.003	14290	14290	···	14290	0.000	14290
76	1890.000	1890.000	10.191	0.033	8.161	10.223	11.158	12	6.692	69	0.011	1890	1890	···	···	···	1890
81	1886.000	1886.000	7.050	0.028	6.190	7.079	7.652	5	1.983	65	0.004	1886	1886	···	···	···	1886
86	5126.000	5126.000	9.297	0.024	7.522	9.320	10.458	4	2.155	68	0.003	5126	5126	···	5126	0.000	5126
91	9818.000	9818.000	9.529	0.023	7.934	9.552	10.231	0	0.313	72	0.004	9818	9818	9818	9818	0.000	9818
96	18587.000	18587.000	6.609	0.022	5.178	6.631	7.347	0	0.157	73	0.005	18587	18587	···	18587	0.000	18587
101	1871.000	1871.000	9.786	0.024	9.134	9.809	10.951	0	0.021	78	0.002	1871	1871	···	···	···	1871
106	1366.000	1366.000	8.077	0.022	7.808	8.100	8.347	1	0.590	76	0.006	1366	1366	···	1366	0.000	1366
111	8003.000	8003.000	7.202	0.025	6.422	7.226	7.759	0	0.040	76	0.004	8003	8003	···	8003	0.000	8003
116	9889.000	9889.000	6.842	0.022	6.431	6.863	7.189	0	0.300	67	0.006	9889	9889	···	9889	0.000	9889
121	18818.000	18818.000	5.859	0.022	5.520	5.881	6.139	0	0.297	76	0.003	18818	18818	18818	18818	0.000	18818

Tabela B.4 Resultados para o problema $P||\sum \alpha_j E_j + \sum \beta_j T_j$ com 50 tarefas e 2 máquinas.

Inst.	Média custo GLS	Média custo PI	Média tempo GLS	Média tempo PI	Tempo min	Média Tempo	Tempo máx.	Melhor iter. GLS	Melhor tempo GLS	Melhor arcos PI	Melhor tempo PI	Melhor GLS	MathGLS-IP	Kramer e Subramanian (2015)	*IP-NFlow*	*IP-NFlow* (GAP%)	MSC
1	62985.300	62985.200	12.249	0.037	11.274	12.287	13.233	10	3.192	100	0.006	62985	62985	62985	62985	14.980	62985
6	40008.000	40008.000	14.464	0.029	13.973	14.493	15.072	2	0.778	99	0.006	40008	40008	···	40008	0.000	40008
11	27871.000	27871.000	12.403	0.024	11.416	12.427	13.590	0	0.108	93	0.003	27871	27871	27871	**27870**	0.000	27870
16	47692.000	47692.000	10.820	0.026	10.146	10.846	11.710	0	0.396	86	0.003	47692	47692	···	47692	0.000	47692
21	111069.000	111069.000	12.725	0.020	12.537	12.745	13.270	0	0.046	97	0.005	111069	111069	111069	111069	0.000	111069
26	38133.000	38133.000	14.333	0.030	13.840	14.363	14.896	1	0.497	89	0.004	38133	38133	···	···	···	38133
31	18266.000	18266.000	15.013	0.029	14.253	15.042	16.297	1	0.683	96	0.005	18266	18266	18266	18266	0.000	18266
36	25205.000	25205.000	12.163	0.033	10.666	12.196	13.909	4	1.794	96	0.004	25205	25205	···	···	···	25205
41	38491.000	38491.000	12.629	0.022	12.112	12.652	13.875	6	2.303	92	0.005	38491	38491	38491	38491	0.000	38491
46	82134.000	82134.000	14.763	0.021	14.452	14.784	15.025	0	0.021	93	0.004	82134	82134	···	82134	0.000	82134
51	23697.000	23697.000	16.175	0.028	15.107	16.203	17.563	11	3.319	89	0.024	23697	23697	23697	23697	25.520	23697
56	17205.000	17205.000	13.120	0.027	12.496	13.147	14.698	1	0.471	83	0.006	17205	17205	···	17204	0.298	17204
61	17456.000	17456.000	14.030	0.026	12.858	14.056	14.801	3	1.154	102	0.003	17456	17456	17456	17456	0.000	17456
66	41153.000	41153.000	17.389	0.025	17.163	17.414	17.708	21	9.373	73	0.003	41153	41153	···	41153	0.000	41153
71	78612.000	78612.000	13.185	0.028	12.961	13.213	13.614	0	0.011	92	0.004	78612	78612	78612	78612	0.000	78612
76	39743.000	39743.000	14.506	0.025	13.337	14.530	15.454	0	0.102	106	0.003	39743	39743	···	···	···	39743
81	7903.000	7903.000	12.336	0.029	11.137	12.364	13.482	0	0.231	98	0.003	7903	7903	7903	7903	4.450	7903
86	13250.000	13250.000	17.093	0.026	16.574	17.119	17.444	1	0.370	86	0.005	13250	13250	···	13250	0.000	13250
91	47513.000	47513.000	13.490	0.023	13.039	13.513	13.989	1	0.302	97	0.005	47513	47513	47513	47513	0.000	47513
96	92899.000	92899.000	14.559	0.024	14.137	14.583	14.897	1	0.283	98	0.006	92899	92899	···	92899	0.000	92899
101	24748.000	24748.000	9.804	0.030	8.424	9.833	10.588	3	0.809	85	0.004	24748	24748	···	24748	40.950	24748
106	9500.000	9500.000	12.476	0.029	11.961	12.505	13.210	2	0.695	89	0.005	9500	9500	···	···	···	9500
111	17012.000	17012.000	13.584	0.023	13.013	13.607	14.397	0	0.040	96	0.004	17012	17012	17012	17012	0.000	17012
116	20022.000	20022.000	12.734	0.031	12.126	12.764	13.223	4	1.202	78	0.004	20022	20022	···	20022	0.000	20022
121	41989.000	41989.000	13.859	0.021	13.680	13.880	14.086	0	0.214	86	0.004	41989	41989	41989	41988	0.000	41988

Tabela B.5 Resultados para o problema $P||\sum \alpha_j E_j + \sum \beta_j T_j$ com 50 tarefas e 4 máquinas.

Inst.	Média custo GLS	Média custo PI	Média tempo GLS	Média tempo PI	Tempo min	Média Tempo	Tempo máx.	Melhor iter. GLS	Melhor tempo GLS	Melhor arcos PI	Melhor tempo PI	Melhor GLS	MathGLS-IP	Kramer e Subramanian (2015)	*IP-NFlow*	*IP-NFlow* (GAP%)	MSC
1	26311.000	26311.000	13.221	0.033	12.921	13.254	13.963	2	0.793	122	0.006	26311	26311	26311	26311	0.000	26311
6	19778.000	19778.000	17.571	0.034	16.799	17.605	18.657	0	0.031	121	0.006	19778	19778	···	19778	0.000	19778
11	14828.000	14828.000	14.516	0.031	13.967	14.548	15.061	6	2.554	113	0.005	14828	14828	14828	14828	0.000	14828
16	25908.000	25908.000	13.522	0.024	12.842	13.546	14.115	0	0.194	105	0.004	25908	25908	···	25908	0.000	25908
21	59441.000	59441.000	13.535	0.022	12.644	13.557	13.910	0	0.238	119	0.004	59441	59441	59441	59441	0.000	59441
26	16632.000	16632.000	20.002	0.034	19.327	20.036	20.712	3	1.550	108	0.007	16632	16632	···	···	···	16632
31	8714.900	8714.900	15.912	0.033	15.185	15.945	17.102	1	0.845	117	0.006	8709	8709	8709	8709	0.000	8709
36	13442.000	13442.000	15.563	0.029	14.597	15.592	16.598	3	1.522	117	0.006	13442	13442	···	13442	0.000	13442
41	22009.000	22009.000	15.665	0.022	14.601	15.687	16.253	0	0.038	112	0.003	22009	22009	22009	22009	0.000	22009
46	44469.000	44469.000	13.579	0.023	12.856	13.602	14.645	0	0.094	114	0.003	44469	44469	···	44469	0.000	44469
51	9057.000	9057.000	15.324	0.037	14.814	15.361	16.142	4	2.016	108	0.029	9057	9057	···	9057	3.980	9057
56	7851.000	7851.000	16.202	0.035	15.689	16.237	16.780	5	2.162	101	0.005	7851	7851	···	7851	0.000	7851
61	9376.000	9376.000	16.679	0.029	15.880	16.708	17.297	1	0.549	124	0.006	9376	9376	9376	9376	0.000	9376
66	23272.000	23272.000	15.434	0.029	14.646	15.463	16.566	6	2.879	88	0.006	23272	23272	···	23272	0.000	23272
71	42660.000	42660.000	15.880	0.024	15.216	15.904	16.560	0	0.125	112	0.004	42660	42660	42660	42660	0.000	42660
76	14970.000	14970.000	16.926	0.033	15.912	16.959	17.585	4	1.771	129	0.006	14970	14970	···	···	···	14970
81	3835.000	3835.000	17.930	0.033	16.563	17.964	19.380	3	1.483	120	0.008	3835	3835	3835	3835	0.000	3835
86	8897.000	8897.000	17.388	0.031	16.065	17.419	18.498	9	4.134	104	0.005	8897	8897	···	8897	0.000	8897
91	26660.800	26660.800	13.982	0.033	13.686	14.015	14.474	11	4.128	118	0.006	26659	26659	26659	26659	0.000	26659
96	50339.000	50339.000	14.694	0.029	14.173	14.723	15.215	5	2.065	119	0.006	50339	50339	···	50339	0.000	50339
101	6602.000	6602.000	17.900	0.027	17.482	17.928	18.421	0	0.214	104	0.003	6602	6602	···	6602	0.000	6602
106	3692.000	3692.000	20.532	0.034	19.754	20.566	21.619	0	0.412	108	0.004	3692	3692	···	···	···	3692
111	10695.200	10695.200	16.689	0.033	16.176	16.722	17.562	7	3.254	116	0.008	10694	10694	10694	10694	0.000	10694
116	11745.000	11745.000	16.346	0.032	15.708	16.379	17.370	1	0.673	95	0.007	11745	11745	···	11745	0.000	11745
121	23885.200	23885.200	12.727	0.032	11.952	12.758	13.227	28	8.962	105	0.008	23884	23884	23884	23884	0.000	23884

Tabela B.6 Resultados para o problema $P||\sum \alpha_j E_j + \sum \beta_j T_j$ com 50 tarefas e 10 máquinas.

Inst.	Média custo GLS	Média custo PI	Média tempo GLS	Média tempo PI	Tempo min	Média Tempo	Tempo máx.	Melhor iter. GLS	Melhor tempo GLS	Melhor arcos PI	Melhor tempo PI	Melhor GLS	MathGLS-IP	Kramer e Subramanian (2015)	*IP-NFlow*	*IP-NFlow* (GAP%)	MSC
1	8886.800	8886.800	13.578	0.044	13.184	13.622	14.295	11	8.566	148	0.008	8879	8879	8879	8879	0.000	8879
6	8140.000	8140.000	18.158	0.027	17.364	18.185	18.823	0	0.584	146	0.004	8140	8140	···	8140	0.000	8140
11	8023.000	8023.000	12.743	0.029	11.519	12.771	13.621	1	0.866	137	0.006	8023	8023	···	8023	0.000	8023
16	14236.000	14236.000	14.674	0.029	13.908	14.703	15.747	1	0.894	127	0.006	14236	14236	···	14236	0.000	14236
21	28666.000	28666.000	9.663	0.027	8.998	9.689	10.179	0	0.032	144	0.003	28666	28666	···	28666	0.000	28666
26	4972.700	4972.700	15.940	0.043	15.155	15.982	16.720	12	10.183	131	0.009	4971	4971	···	···	···	4971
31	3845.800	3845.800	14.084	0.038	12.629	14.122	15.132	14	9.602	142	0.010	3839	3839	3839	3839	0.000	3839
36	8369.200	8369.200	12.412	0.030	11.952	12.441	12.870	0	0.641	142	0.008	8365	8365	···	8364	0.000	8364
41	12671.000	12671.000	10.113	0.028	9.838	10.140	10.526	0	0.425	136	0.003	12671	12671	···	12671	0.000	12671
46	22166.000	22166.000	9.517	0.025	9.107	9.543	10.179	0	0.041	138	0.003	22166	22166	···	22166	0.000	22166
51	2831.300	2831.300	16.623	0.045	15.728	16.668	17.501	11	9.410	131	0.008	2826	2826	···	2826	0.000	2826
56	2599.000	2599.000	15.796	0.037	14.750	15.833	20.436	4	3.279	123	0.007	2599	2599	···	2599	0.000	2599
61	5856.000	5856.000	15.542	0.030	13.833	15.571	16.751	4	3.682	150	0.006	5856	5856	5856	5856	0.000	5856
66	12954.000	12954.000	11.460	0.028	10.716	11.488	12.670	0	0.187	107	0.003	12954	12954	···	12954	0.000	12954
71	21315.000	21315.000	10.066	0.028	9.521	10.094	10.757	0	0.150	136	0.004	21315	21315	···	21315	0.000	21315
76	4157.000	4157.000	16.411	0.040	14.832	16.452	17.697	18	14.925	157	0.010	4157	4157	···	4157	0.000	4157
81	2365.100	2365.100	17.213	0.035	16.464	17.248	18.476	4	3.758	145	0.006	2363	2363	···	2363	0.000	2363
86	6833.000	6833.000	15.075	0.034	14.333	15.110	15.724	4	3.833	126	0.008	6833	6833	···	6833	0.000	6833
91	14527.000	14527.000	12.049	0.031	10.726	12.080	12.962	4	3.325	143	0.006	14527	14527	14527	14527	0.000	14527
96	25103.000	25103.000	11.067	0.031	10.285	11.098	11.790	0	0.429	144	0.004	25103	25103	···	25103	0.000	25103
101	2073.400	2073.400	16.729	0.043	15.710	16.772	17.469	10	8.162	126	0.008	2071	2071	···	···	···	2071
106	1959.000	1959.000	13.897	0.041	13.457	13.938	14.603	4	3.710	132	0.008	1955	1955	···	1955	0.000	1955
111	7841.000	7841.000	14.911	0.026	14.179	14.937	16.066	1	1.211	141	0.006	7841	7841	···	7841	0.000	7841
116	7119.000	7119.000	12.773	0.031	12.072	12.804	14.465	1	1.441	115	0.005	7119	7119	···	7119	0.000	7119
121	13346.000	13346.000	10.837	0.030	10.020	10.867	11.821	0	0.235	128	0.004	13346	13346	13346	13346	0.000	13346

Tabela B.7 Resultados para o problema $P||\sum \alpha_j E_j + \sum \beta_j T_j$ com 100 tarefas e 2 máquinas.

Inst.	Média custo GLS	Média custo PI	Média tempo GLS	Média tempo PI	Tempo min	Média Tempo	Tempo máx.	Melhor iter. GLS	Melhor tempo GLS	Melhor arcos PI	Melhor tempo PI	Melhor GLS	MathGLS-IP	Kramer e Subramanian (2015)	*IP-NFlow*	*IP-NFlow* (GAP%)	MSC
1	198307.400	198307.200	172.895	0.252	166.597	173.148	182.697	13	53.134	781	0.010	198290	198288	**198282**	198290	12.620	198282
6	97835.200	97835.200	173.221	0.183	166.808	173.403	181.084	5	23.043	757	0.075	97835	97835	⋯	97835	0.000	97835
11	127695.400	127695.000	161.459	0.160	152.986	161.619	177.681	5	21.921	732	0.006	127666	127666	127666	127666	0.167	127666
16	214158.400	214157.000	215.078	0.100	207.280	215.178	220.823	5	27.298	774	0.004	214155	214155	⋯	214155	0.000	214155
21	457865.000	457865.000	173.602	0.071	172.262	173.673	175.573	0	3.086	842	0.008	457865	457865	457865	457865	0.000	457865
26	199435.200	199435.100	164.977	0.149	159.672	165.126	168.641	12	50.863	832	0.011	199435	199435	⋯	199435	19.171	199435
31	95038.200	95038.100	159.204	0.174	151.862	159.378	168.249	6	27.291	779	0.036	95038	95038	95038	95038	0.806	95038
36	76244.700	76244.700	164.399	0.163	153.539	164.562	175.422	17	68.612	745	0.020	76243	76243	⋯	76243	0.292	76243
41	238416.000	238416.000	200.982	0.128	195.773	201.111	207.780	14	73.869	870	0.009	238416	238416	238416	238416	0.056	238416
46	422861.000	422861.000	200.661	0.109	186.432	200.770	216.969	1	6.997	800	0.006	422861	422861	⋯	422861	0.000	422861
51	198185.500	198185.200	204.073	0.171	196.214	204.245	215.384	34	167.819	782	0.016	198182	198182	198182	198182	30.677	198182
56	59736.200	59736.200	166.917	0.125	160.772	167.042	173.373	5	19.844	752	0.009	59730	59730	⋯	59730	1.247	59730
61	52739.000	52739.000	200.752	0.108	187.237	200.860	214.056	4	18.047	716	0.009	52739	52739	52739	52739	0.373	52739
66	126937.600	126936.800	140.718	0.115	134.406	140.833	145.810	2	8.097	674	0.006	126936	126936	⋯	126936	0.119	126936
71	327358.200	327358.000	153.858	0.098	143.826	153.956	161.872	13	49.102	777	0.008	327358	327358	327358	327358	0.006	327358
76	111782.500	111781.700	169.458	0.147	157.041	169.605	190.067	32	132.417	734	0.007	111761	111761	⋯	111761	46.525	111761
81	22726.000	22722.900	179.973	0.128	171.463	180.101	193.931	7	30.996	705	0.012	22720	22720	22720	22720	4.566	22720
86	39459.000	39459.000	207.437	0.135	199.920	207.571	216.485	5	28.098	766	0.007	39459	39459	⋯	39459	1.495	39459
91	130167.100	130166.900	187.218	0.114	175.280	187.332	193.474	2	10.934	720	0.005	130165	130165	130165	130165	0.194	130165
96	254326.300	254326.300	192.508	0.104	185.298	192.612	198.933	5	24.180	729	0.007	254323	254323	⋯	254323	0.050	254323
101	85072.500	85069.200	173.471	0.144	166.617	173.615	184.880	24	107.461	715	0.008	85058	85058	⋯	85058	49.235	85058
106	43882.000	43882.000	97.365	0.117	89.689	97.482	101.190	4	10.389	754	0.010	43882	43882	⋯	43882	48.71	43882
111	85173.900	85166.500	98.644	0.104	95.814	98.748	102.140	36	92.558	788	0.061	85160	85160	85160	85160	0.074	85160
116	191463.300	191459.700	103.362	0.276	100.715	103.638	106.953	17	41.691	737	0.011	191452	191452	⋯	191452	0.099	191452
121	242238.800	242237.500	86.490	0.076	82.662	86.566	92.868	22	50.575	758	0.054	242234	242234	242234	242234	0.039	242234

Tabela B.8 Resultados para o problema $P||\sum \alpha_j E_j + \sum \beta_j T_j$ com 100 tarefas e 4 máquinas.

Inst.	Média custo GLS	Média custo PI	Média tempo GLS	Média tempo PI	Tempo min	Média Tempo	Tempo máx.	Melhor iter. GLS	Melhor tempo GLS	Melhor arcos PI	Melhor tempo PI	Melhor GLS	MathGLS-IP	Kramer e Subramanian (2015)	*IP-NFlow*	*IP-NFlow* (GAP%)	MSC
1	87921.900	87921.900	94.854	0.143	88.779	94.997	106.047	30	69.288	931	0.023	87921	**87921**	94986	87923	3.830	87921
6	48771.900	48771.900	106.872	0.120	102.795	106.992	112.269	29	75.195	902	0.026	48758	48758	⋯	⋯	⋯	48758
11	65733.800	65733.200	110.616	0.104	103.283	110.720	114.845	16	49.531	872	0.019	65719	65719	65719	65719	0.000	65719
16	112299.000	112299.000	98.798	0.078	96.027	98.876	102.062	6	14.781	922	0.010	112299	112299	⋯	112299	0.000	112299
21	237398.300	237398.300	86.222	0.064	82.578	86.287	89.418	1	4.299	1002	0.007	237398	237398	237398	237398	0.000	237398
26	80633.800	80631.400	106.159	0.108	102.543	106.267	113.100	9	30.785	991	0.022	80582	80582	⋯	⋯	⋯	80582
31	45677.700	45677.700	127.997	0.123	124.073	128.120	131.673	23	76.393	928	0.021	45644	**45644**	**45664**	45838	0.710	45664
36	39510.100	39510.100	115.108	0.093	111.424	115.201	118.423	26	76.284	887	0.026	39509	39509	⋯	⋯	⋯	39509
41	126408.700	126408.700	102.924	0.071	99.492	102.995	106.283	2	7.407	1037	0.009	126408	126408	126408	126408	0.000	126408
46	219569.300	219569.300	83.161	0.074	79.741	83.234	85.849	6	12.645	953	0.012	219569	219569	⋯	219569	0.000	219569
51	66766.300	66765.200	126.791	0.105	121.424	126.896	132.970	39	130.764	931	0.024	66711	66711	⋯	**66699**	0.300	66699
56	27984.600	27984.600	132.324	0.106	124.789	132.430	138.161	10	33.892	896	0.015	27961	27961	⋯	⋯	⋯	27961
61	28273.000	28273.000	135.951	0.079	121.071	136.030	146.788	8	27.129	853	0.020	28273	**28273**	28290	**28273**	0.520	28273
66	68189.400	68185.800	100.513	0.075	96.959	100.588	103.849	27	66.090	803	0.008	68167	68154		68152	0.000	68152
71	170698.600	170698.300	93.534	0.084	90.459	93.618	96.769	19	44.093	925	0.023	170697	170696	170696	170696	0.000	170696
76	28793.800	28793.800	133.367	0.089	126.028	133.456	138.865	9	34.227	874	0.016	28676	28676	⋯	⋯	⋯	28676
81	10519.500	10518.100	133.023	0.121	128.416	133.144	136.618	39	130.146	840	0.028	10491	**10491**	10550	10517	3.760	10491
86	22893.600	22893.600	131.921	0.090	126.873	132.011	136.650	9	31.609	913	0.016	22893	22893	⋯	⋯	⋯	22893
91	70756.000	70756.000	109.579	0.078	106.803	109.657	113.088	13	35.994	858	0.015	70755	70755	70755	70755	0.000	70755
96	133603.200	133603.200	93.684	0.080	91.567	93.763	97.587	24	56.894	869	0.020	133602	133602	⋯	133602	0.000	133602
101	21291.300	21291.300	157.082	0.095	149.495	157.177	162.933	20	81.380	852	0.011	21269	**21269**	⋯	21321	2.730	21269
106	14540.300	14538.700	124.591	0.102	115.294	124.693	130.185	31	100.130	898	0.034	14518	14518	⋯	⋯	⋯	14518
111	47375.000	47374.300	135.453	0.102	131.431	135.555	138.787	39	131.838	939	0.023	47352	**47352**	47365	47364	0.330	47352
116	101640.700	101640.400	101.278	0.090	98.885	101.368	103.030	11	29.893	878	0.011	101637	101637	⋯	⋯	⋯	101637
121	127713.700	127712.800	109.891	0.100	102.074	109.992	134.545	37	105.134	903	0.037	127697	127697	127684	**127682**	0.013	127682

Tabela B.9 Resultados para o problema $P||\sum \alpha_j E_j + \sum \beta_j T_j$ com 100 tarefas e 10 máquinas.

Inst.	Média custo GLS	Média custo PI	Média tempo GLS	Média tempo PI	Tempo min	Média Tempo	Tempo máx.	Melhor iter. GLS	Melhor tempo GLS	Melhor arcos PI	Melhor tempo PI	Melhor GLS	MathGLS-IP	Kramer e Subramanian (2015)	*IP-NFlow*	*IP-NFlow* (GAP%)	MSC
1	31255.700	31255.700	111.549	0.185	108.148	111.734	116.909	16	91.400	1071	0.053	31241	31241	31251	**31213**	0.000	31213
6	19457.300	19457.300	109.831	0.137	106.619	109.968	112.559	17	98.468	1038	0.119	19436	19436	···	**19399**	0.000	19399
11	28786.500	28786.500	87.425	0.101	79.852	87.525	99.254	13	57.854	1004	0.021	28772	**28772**	···	**28772**	0.000	28772
16	52245.000	52245.000	77.145	0.096	72.028	77.241	83.844	3	16.530	1060	0.010	52245	**52245**	···	**52245**	0.000	52245
21	105424.000	105424.000	59.972	0.071	56.385	60.043	62.304	5	18.109	1153	0.015	105424	**105424**	···	**105424**	0.000	105424
26	29766.600	29766.600	118.609	0.132	113.180	118.740	121.555	6	37.337	1140	0.022	29716	29716	···	**29652**	0.068	29652
31	17105.900	17105.900	117.138	0.182	112.767	117.320	124.925	18	109.434	1067	0.113	17082	17082	17072	**17012**	0.000	17012
36	18105.300	18105.200	98.647	0.104	82.475	98.751	109.061	9	57.294	1020	0.021	18085	18085	···	18085	0.000	18085
41	59595.400	59595.400	79.784	0.100	68.678	79.884	85.532	19	83.824	1192	0.029	59594	59594	···	**59593**	0.000	59593
46	97797.600	97797.600	63.442	0.073	60.167	63.515	65.920	3	13.739	1096	0.012	97797	97797	···	**97796**	0.000	97796
51	23725.300	23725.100	125.678	0.144	116.480	125.821	133.807	13	92.107	1071	0.018	23621	**23621**	···	···	···	23621
56	9898.000	9898.000	110.761	0.149	102.253	110.910	119.026	16	89.006	1031	0.045	9778	9778	···	**9777**	0.000	9777
61	14677.500	14677.500	111.016	0.116	105.521	111.132	117.695	10	61.136	982	0.019	14645	14645	14677	**14634**	0.000	14634
66	33319.900	33319.900	90.337	0.096	84.889	90.433	94.684	19	92.132	924	0.013	33313	33313	···	**33307**	0.000	33307
71	76966.800	76966.800	82.378	0.096	78.234	82.474	86.803	13	56.957	1064	0.020	76965	76965	···	**76964**	0.000	76964
76	10068.200	10068.200	116.468	0.096	109.089	116.564	125.234	19	118.528	1006	0.019	9960	9960	···	···	···	9960
81	4024.400	4024.400	135.877	0.133	130.171	136.011	142.299	19	125.257	966	0.020	4013	4013	···	**3989**	0.000	3989
86	13776.000	13776.000	116.287	0.100	113.690	116.387	119.479	11	70.738	1050	0.020	13721	13721	···	**13719**	0.000	13719
91	35792.800	35792.800	88.800	0.106	79.660	88.905	100.083	8	44.545	987	0.023	35790	35790	35788	35784	0.000	**35784**
96	61493.500	61493.500	84.701	0.094	69.287	84.795	90.234	13	60.066	999	0.042	61491	**61491**	···	**61491**	0.000	61491
101	7466.600	7466.600	130.012	0.137	127.629	130.149	132.487	17	113.302	980	0.024	7409	**7409**	···	···	···	7409
106	3752.500	3752.500	118.061	0.141	113.990	118.202	124.637	15	94.949	1033	0.024	3720	3720	···	**3705**	1.883	3705
111	25407.900	25407.900	118.636	0.134	115.471	118.770	122.411	14	88.840	1080	0.019	25394	25394	···	**25368**	0.000	25368
116	48163.700	48163.700	104.226	0.099	102.509	104.325	106.229	17	92.610	1009	0.026	48153	48153	···	**48132**	0.000	48132
121	59375.200	59374.300	94.116	0.122	90.120	94.237	97.104	19	91.389	1038	0.019	59365	59365	59361	**59346**	0.000	59346

Tabela B.10 Resultados para o problema $P||\sum \alpha_j E_j + \sum \beta_j T_j$ com 200 tarefas e 2 máquinas.

Inst.	Média custo GLS	Média custo PI	Média tempo GLS	Média tempo PI	Tempo mín.	Média tempo	Tempo máx.	Melhor iter. GLS	Melhor tempo GLS	Melhor arcos PI	Melhor tempo PI	Melhor GLS	MathGLS-IP
1	746240.300	746230.800	768.568	0.285	682.120	768.854	815.068	17	354.429	6002	0.029	746193	**746192**
6	499793.300	499792.200	751.501	0.222	711.222	751.724	785.143	39	754.671	6241	0.047	499788	499788
11	352612.300	352593.300	781.138	0.254	764.950	781.392	808.057	35	714.276	5528	0.050	352566	**352549**
16	839776.000	839776.000	797.725	0.174	747.284	797.899	878.983	4	96.238	6404	0.009	839776	839776
21	1203101.000	1203101.000	640.532	0.179	608.121	640.711	683.756	4	66.448	6031	0.007	1203101	1203101
26	598289.100	598284.900	802.201	0.227	754.082	802.428	837.923	23	473.540	5811	0.026	598234	**598233**
31	327414.700	327408.800	788.299	0.257	738.741	788.555	832.737	35	740.503	6309	0.020	327308	**327306**
36	330202.500	330197.200	704.062	0.215	657.771	704.277	746.058	17	304.699	6177	0.014	330175	330175
41	699320.800	699318.100	810.850	0.188	764.211	811.038	847.300	4	74.581	5899	0.015	699301	699301
46	1119000.500	1119000.200	659.628	0.206	623.203	659.833	682.561	6	90.887	6072	0.019	1118997	1118997
51	645896.000	645850.000	776.598	0.457	719.407	777.054	812.756	8	164.192	5760	0.017	645527	**645525**
56	217937.900	217907.300	749.093	0.258	713.716	749.351	772.035	37	733.113	6331	0.028	217842	**217763**
61	226486.900	226465.800	775.698	0.219	714.375	775.917	807.512	5	108.803	5974	0.015	226385	226385
66	548904.000	548885.500	698.353	0.247	663.342	698.600	727.397	18	312.800	6225	0.043	548793	**548790**
71	866929.700	866928.000	707.471	0.185	662.572	707.656	730.961	7	122.902	6186	0.019	866920	866920
76	358746.200	358744.100	774.529	0.245	739.597	774.775	830.369	38	765.261	5784	0.068	358715	358715
81	120725.400	120698.100	893.216	0.222	874.744	893.438	902.166	33	737.938	6324	0.021	120677	120677
86	149219.500	149200.700	832.039	0.201	815.844	832.240	860.831	39	824.146	5891	0.022	149201	**149174**
91	393654.300	393644.500	775.143	0.196	723.520	775.339	816.221	21	398.153	5870	0.020	393635	**393610**
96	633704.200	633697.400	735.492	0.195	668.702	735.687	773.879	19	313.486	5723	0.041	633656	**633636**
101	382755.800	382744.500	761.951	0.219	716.796	762.170	802.421	15	324.146	6013	0.025	382712	**382710**
106	75272.600	75241.300	802.274	0.208	745.241	802.482	850.321	19	397.043	5993	0.024	75183	**75143**
111	55415.100	55356.100	891.136	0.205	847.513	891.341	949.747	3	62.497	6028	0.013	55358	**55312**
116	315561.700	315535.900	777.601	0.214	760.999	777.815	810.713	10	196.706	5875	0.026	315374	**315362**
121	496741.900	496721.100	691.448	0.196	651.353	691.644	719.213	39	705.789	5602	0.020	496717	496717

Tabela B.11 Resultados para o problema $P||\sum \alpha_j E_j + \sum \beta_j T_j$ com 200 tarefas e 4 máquinas.

Inst.	Média custo GLS	Média custo PI	Média tempo GLS	Média tempo PI	Tempo mín.	Média tempo	Tempo máx.	Melhor iter. GLS	Melhor tempo GLS	Melhor arcos PI	Melhor tempo PI	Melhor GLS	MathGLS-IP
1	296635.500	296635.400	778.477	0.311	717.183	778.788	958.136	24	556.744	7076	0.024	296456	296456
6	248492.200	248491.700	968.747	0.290	916.867	969.038	1025.506	34	815.061	7357	0.123	248293	**248288**
11	178660.900	178660.000	933.549	0.462	855.791	934.011	1003.027	31	795.064	6517	0.247	178588	**178586**
16	429664.800	429664.700	771.830	0.239	747.097	772.069	796.125	13	270.484	7550	0.039	429651	429651
21	614222.900	614222.200	676.213	0.236	650.142	676.449	698.870	7	141.409	7110	0.042	614220	614220
26	237865.600	237851.400	1129.084	0.249	1043.110	1129.332	1246.386	34	1056.779	6850	0.031	237441	237441
31	162440.100	162438.100	1071.728	0.303	1019.041	1072.032	1114.539	34	989.325	7438	0.070	162333	162333
36	168423.400	168421.000	886.977	0.325	832.574	887.303	949.213	39	846.218	7282	0.038	168358	168358
41	361141.100	361140.900	814.522	0.257	764.722	814.779	855.880	23	476.187	6954	0.035	361130	361130
46	572212.900	572212.800	705.597	0.236	680.584	705.833	739.030	28	513.510	7158	0.037	572206	572206
51	194146.900	194141.900	1399.447	0.286	1374.376	1399.733	1455.823	30	1064.210	6791	0.093	193991	193991
56	105823.000	105817.700	1115.510	0.335	1073.801	1115.845	1133.587	35	973.722	7463	0.077	105604	105604
61	118219.800	118213.800	972.767	0.345	925.103	973.113	1040.960	30	738.381	7042	0.123	118179	**118147**
66	284514.200	284513.900	870.956	0.329	834.457	871.285	897.757	39	858.835	7339	0.075	284362	284362
71	445280.900	445280.200	757.274	0.272	705.973	757.546	789.990	35	662.156	7292	0.047	445254	445254
76	94560.900	94545.700	1485.944	0.293	1406.099	1486.237	1552.058	16	584.753	6819	0.031	94409	94409
81	57480.400	57466.200	1339.968	0.510	1288.215	1340.478	1405.727	20	703.071	7455	0.074	57260	**57241**
86	82232.800	82231.000	1033.897	0.226	1007.664	1034.123	1062.340	10	301.230	6945	0.023	82114	82114
91	206894.600	206888.000	1016.981	0.306	958.019	1017.286	1054.185	35	887.337	6921	0.107	206826	206826
96	327179.400	327179.200	910.706	0.306	886.204	911.012	943.124	19	439.918	6747	0.088	327080	**327078**
101	92290.300	92277.600	1519.642	0.268	1427.135	1519.910	1575.921	9	411.000	7089	0.017	91984	**91977**
106	34360.500	34356.400	1225.603	0.347	1165.917	1225.950	1259.171	4	178.683	7065	0.054	33948	**33910**
111	31560.900	31556.000	1139.818	0.303	1117.698	1140.121	1179.245	29	851.215	7107	0.050	31416	**31409**
116	165460.600	165457.100	1037.261	0.313	1010.340	1037.574	1089.517	31	840.948	6926	0.086	165405	**165390**
121	257814.100	257812.700	924.911	0.294	883.643	925.205	955.212	24	540.358	6604	0.056	257768	257768

Tabela B.12 Resultados para o problema $P||\sum \alpha_j E_j + \sum \beta_j T_j$ com 200 tarefas e 10 máquinas.

Inst.	Média custo GLS	Média custo PI	Média tempo GLS	Média tempo PI	Tempo mín.	Média tempo	Tempo máx.	Melhor iter. GLS	Melhor tempo GLS	Melhor arcos PI	Melhor tempo PI	Melhor GLS	MathGLS-IP
1	115048.300	115047.900	922.945	0.334	883.449	923.279	960.617	12	600.635	7913	0.096	115005	115005
6	97717.700	97714.500	905.694	0.567	865.685	906.261	953.350	17	778.393	8228	0.104	97656	**97641**
11	75031.000	75031.000	942.264	0.852	904.673	943.117	987.938	17	898.867	7288	0.999	74975	74975
16	184319.900	184319.900	754.137	0.445	635.914	754.582	787.475	16	564.201	8442	0.062	184287	184287
21	261289.900	261289.900	650.901	0.298	591.480	651.199	694.347	18	558.908	7951	0.102	261282	261282
26	89960.300	89960.300	1031.355	0.346	992.144	1031.701	1093.923	18	988.966	7660	0.025	89885	89885
31	63477.700	63477.700	990.851	0.717	960.113	991.567	1021.852	17	898.451	8317	0.077	63401	63401
36	71988.500	71988.500	867.099	0.904	842.069	868.004	920.837	19	834.605	8143	2.944	71940	71940
41	158961.800	158961.800	825.046	0.420	790.668	825.466	848.499	18	737.791	7776	0.036	158943	158943
46	244717.000	244717.000	700.481	0.298	642.643	700.779	741.723	15	536.976	8005	0.195	244703	244703
51	73773.500	73773.500	1092.361	0.305	1054.748	1092.667	1128.703	11	622.441	7593	0.071	73656	73656
56	38967.900	38967.900	1033.284	1.699	1012.702	1034.983	1072.106	18	928.456	8346	0.646	38902	38902
61	54088.900	54088.900	897.034	0.382	861.667	897.416	926.302	16	755.900	7875	0.054	54037	54037
66	126940.000	126940.000	854.744	0.637	836.166	855.382	898.852	15	654.637	8206	0.079	126899	126899
71	192969.700	192969.700	767.980	0.301	742.220	768.282	798.723	12	502.835	8154	0.041	192949	192949
76	34508.500	34508.500	1112.043	0.406	1083.401	1112.449	1165.264	19	1149.480	7625	0.062	34285	34285
81	19960.600	19960.600	1137.514	1.875	1093.395	1139.388	1174.972	19	1122.846	8337	0.049	19879	19879
86	43666.200	43666.200	966.931	0.393	933.241	967.325	1004.392	18	940.942	7766	0.148	43604	43604
91	95762.800	95762.800	893.444	0.696	879.826	894.140	914.282	19	891.565	7739	0.074	95719	95719
96	145080.300	145080.300	874.168	0.519	826.498	874.686	966.869	18	793.062	7544	0.057	145036	145036
101	35517.500	35517.500	1173.831	0.492	1146.392	1174.323	1213.587	17	980.242	7927	0.121	35366	35366
106	9133.700	9133.700	1117.348	1.936	1081.801	1119.284	1164.279	17	969.715	7900	0.064	8897	8897
111	18276.600	18276.600	1054.034	0.384	1023.408	1054.418	1079.905	18	992.660	7946	0.031	18226	18226
116	77371.100	77371.100	941.939	0.369	921.888	942.307	974.505	18	918.004	7745	0.022	77322	77322
121	115799.500	115799.500	891.167	0.815	841.154	891.981	933.152	18	843.024	7385	0.456	115726	115726

Tabela B.13 Resultados para o problema $P||\sum \alpha_j E_j + \sum \beta_j T_j$ com 300 tarefas e 2 máquinas.

Inst.	Média custo GLS	Média custo PI	Média tempo GLS	Média tempo PI	Tempo mín.	Média tempo	Tempo máx.	Melhor iter. GLS	Melhor tempo GLS	Melhor arcos PI	Melhor tempo PI	Melhor GLS	MathGLS-IP
1	1580231.800	1580199.500	2577.313	0.513	2509.202	2577.827	2684.513	17	1137.230	20111	0.030	1579828	**1579745**
6	1082160.400	1082156.000	2490.464	0.503	2379.808	2490.967	2573.629	29	1753.348	21101	0.037	1081653	**1081645**
11	1013148.000	1013140.200	2475.852	0.451	2363.255	2476.303	2593.209	10	633.262	21035	0.036	1012759	1012759
16	1699585.800	1699563.200	2272.650	0.497	2169.050	2273.147	2391.120	34	1919.528	20457	0.056	1699545	**1699536**
21	2892611.500	2892607.800	2033.741	0.425	1907.860	2034.166	2142.698	38	1930.349	20006	0.077	2892584	**2892582**
26	1423171.100	1423144.400	2727.898	0.458	2666.621	2728.356	2835.835	14	913.329	19925	0.074	1423112	**1423092**
31	680268.600	680226.500	2738.751	0.553	2662.780	2739.304	2786.836	27	1873.001	20663	0.107	679942	**679892**
36	810995.300	810962.600	2655.289	0.552	2579.781	2655.841	2795.412	38	2531.045	20191	0.124	810760	**810734**
41	1548338.000	1548311.500	2833.245	0.404	2703.870	2833.649	2971.277	19	1388.754	19596	0.029	1548121	**1548086**
46	2506984.800	2506975.900	2119.870	0.447	2042.235	2120.317	2238.809	4	237.359	20508	0.024	2506921	**2506908**
51	1195772.700	1195734.900	2820.911	0.506	2610.026	2821.417	2949.879	37	2640.838	19848	0.049	1195593	**1195582**
56	439280.600	439213.200	3030.056	0.460	2971.043	3030.516	3100.341	23	1818.336	20448	0.037	439234	**439179**
61	469851.800	469825.500	2526.871	0.495	2269.135	2527.366	2655.147	25	1534.381	20559	0.066	469796	**469766**
66	1226610.500	1226572.200	2364.891	0.458	2252.097	2365.349	2452.968	8	471.814	20749	0.030	1226477	**1226444**
71	1899658.100	1899649.000	2188.184	0.451	2136.630	2188.635	2252.069	14	803.831	20511	0.026	1899627	1899627
76	1007597.900	1007581.000	2466.451	0.427	2367.899	2466.878	2620.833	14	941.491	20354	0.028	1007502	**1007501**
81	197289.900	197189.100	2876.968	0.453	2795.074	2877.420	2952.624	10	708.390	19778	0.028	197170	**197078**
86	388142.000	388080.200	2856.418	0.433	2741.513	2856.851	3016.784	25	1903.651	19383	0.027	388062	**388023**
91	926849.800	926804.900	2444.348	0.493	2339.054	2444.841	2542.327	28	1704.830	21094	0.054	926678	**926582**
96	1522555.400	1522530.800	2230.685	0.425	2127.482	2231.111	2315.007	10	560.098	19528	0.036	1522518	**1522461**
101	881045.300	881044.200	2439.258	0.478	2337.379	2439.735	2597.478	38	2202.533	20999	0.092	880724	**880723**
106	216725.100	216644.000	3003.923	0.508	2955.209	3004.431	3120.552	39	2876.565	20373	0.054	216490	**216439**
111	141876.300	141822.500	3307.698	0.424	3202.532	3308.121	3426.299	9	770.271	20363	0.024	141818	**141775**
116	574302.200	574237.000	2496.451	0.587	2397.424	2497.038	2583.777	25	1606.244	17691	0.063	574132	**574092**
121	1041081.400	1041065.800	2418.375	0.436	2308.825	2418.812	2500.617	24	1436.860	19201	0.081	1040677	**1040619**

Tabela B.14 Resultados para o problema $P||\sum \alpha_j E_j + \sum \beta_j T_j$ com 300 tarefas e 4 máquinas.

Inst.	Média custo GLS	Média custo PI	Média tempo GLS	Média tempo PI	Tempo mín.	Média tempo	Tempo máx.	Melhor iter. GLS	Melhor tempo GLS	Melhor arcos PI	Melhor tempo PI	Melhor GLS	MathGLS-IP
1	646609.400	646603.700	2951.196	0.642	2760.865	2951.837	3135.789	38	2823.483	19322	0.116	646242	**646234**
6	537459.700	537452.200	3759.802	0.831	3590.956	3760.633	3901.389	18	1938.622	18159	0.080	536840	**536823**
11	509952.700	509951.100	3522.612	0.914	3393.735	3523.527	3712.329	24	2318.351	18237	0.604	509784	509784
16	862906.200	862899.400	2750.272	0.606	2527.863	2750.879	2838.908	33	2395.063	18915	0.072	862837	**862828**
21	1466694.900	1466693.900	2370.453	0.706	2181.582	2371.159	2583.310	34	1940.841	19444	0.108	1466692	1466692
26	534331.700	534324.300	3390.415	0.667	3062.869	3391.082	3735.572	35	3037.164	19541	0.122	534008	534008
31	336444.900	336415.400	4571.481	1.088	4456.239	4572.570	4761.539	25	3095.158	18673	0.168	336250	**336230**
36	409345.900	409339.400	4075.194	0.831	3909.716	4076.025	4275.004	36	3773.013	19228	0.266	409140	**409137**
41	789859.900	789852.000	2993.925	1.009	2828.602	2994.934	3214.390	36	2805.997	19928	0.165	789782	789782
46	1273662.100	1273660.200	2582.201	0.627	2428.738	2582.829	2755.508	3	309.732	18855	0.015	1273582	1273582
51	361895.100	361866.400	5092.298	0.656	4700.273	5092.954	5418.301	32	4390.069	19631	0.330	361299	**361281**
56	216166.300	216157.400	4895.841	0.913	4751.677	4896.754	5071.183	37	4544.077	18925	0.116	215523	**215510**
61	239436.800	239397.900	3684.635	0.901	3523.104	3685.536	3818.682	38	3421.102	18795	0.379	239273	**239196**
66	628796.600	628781.500	3318.857	1.018	3279.792	3319.875	3391.673	36	3136.732	18572	0.125	628727	**628695**
71	967022.000	967016.500	2783.707	1.150	2620.805	2784.858	3090.805	26	1864.401	18851	0.250	966973	**966972**
76	250842.300	250813.900	4284.775	0.572	4058.164	4285.347	4616.838	34	3832.398	19036	0.116	250298	**250247**
81	94861.200	94844.400	4659.379	0.991	4470.011	4660.369	5098.461	37	4377.267	19713	0.633	94602	**94575**
86	204987.500	204962.600	3847.983	0.759	3672.696	3848.742	3986.878	27	2804.279	20176	0.282	204633	204633
91	475865.200	475857.200	3484.814	0.577	3418.996	3485.391	3511.886	27	2441.635	18167	0.118	475772	475772
96	777688.400	777681.900	3251.402	0.632	3162.081	3252.034	3351.250	13	1179.639	20008	0.073	777647	**777618**
101	211216.600	211198.800	4640.997	0.579	4326.334	4641.577	4809.312	37	4526.861	18279	0.084	210756	**210721**
106	89699.300	89681.900	5187.144	0.939	4848.930	5188.082	5558.404	38	4988.353	19014	0.944	89249	89249
111	77065.200	77032.800	4615.604	0.803	4403.947	4616.407	4788.547	18	2191.730	19025	0.092	76851	**76817**
116	298929.900	298901.000	3776.553	0.972	3636.004	3777.525	3919.307	36	3452.390	20783	0.161	298797	**298756**
121	534244.000	534235.400	3349.047	1.052	3203.340	3350.099	3518.682	32	2797.510	20391	0.371	534057	**534036**

Tabela B.15 Resultados para o problema $P||\sum \alpha_j E_j + \sum \beta_j T_j$ com 300 tarefas e 10 máquinas.

Inst.	Média custo GLS	Média custo PI	Média tempo GLS	Média tempo PI	Tempo mín.	Média tempo	Tempo máx.	Melhor iter. GLS	Melhor tempo GLS	Melhor arcos PI	Melhor tempo PI	Melhor GLS	MathGLS-IP
1	250420.100	250420.100	3873.598	0.501	3701.706	3874.099	3996.314	19	3759.922	16783	0.061	250136	250136
6	210606.400	210606.400	3510.094	1.173	3361.225	3511.266	3631.300	18	3315.148	15495	0.248	210462	210462
11	207937.700	207937.700	3097.039	4.239	2956.764	3101.278	3243.868	18	2805.457	15581	0.139	207902	207902
16	361777.500	361777.300	2964.379	3.788	2890.779	2968.166	3154.775	13	1943.595	16332	0.091	361720	361720
21	611549.200	611549.200	2598.298	0.614	2407.583	2598.912	2711.202	5	715.447	16919	0.035	611539	611539
26	205628.700	205628.700	3851.271	0.728	3769.007	3851.999	3912.467	18	3568.687	17026	0.045	205270	205270
31	131168.300	131168.300	4207.322	4.946	4027.772	4212.268	4639.873	18	3787.957	16065	1.038	130981	130981
36	169609.800	169609.800	3458.560	2.895	3366.128	3461.455	3627.357	11	2052.180	16679	0.057	169488	169488
41	336070.700	336069.600	3134.830	3.768	3033.665	3138.598	3248.810	19	3130.172	17453	0.678	335969	335969
46	533870.900	533870.900	2684.756	1.959	2627.730	2686.715	2794.488	18	2469.490	16266	0.203	533852	533852
51	140545.400	140545.400	4443.182	0.616	4337.949	4443.798	4519.359	19	4376.993	17125	0.630	140234	140234
56	82458.300	82458.300	4227.072	6.896	4073.671	4233.968	4362.293	19	4019.195	16345	1.623	82356	82356
61	101207.900	101207.900	3730.819	2.055	3624.467	3732.874	3909.109	12	2352.718	16199	2.651	101009	101009
66	272372.900	272372.900	3299.718	1.459	3218.937	3301.176	3406.228	17	2770.358	15953	0.071	272275	272275
71	408720.300	408720.300	2908.694	3.720	2809.880	2912.414	3069.292	17	2426.491	16262	9.158	408683	408683
76	102318.400	102318.400	4029.380	0.384	3866.100	4029.764	4252.262	15	3282.391	16467	0.061	101604	101604
81	34504.300	34504.300	4452.943	4.100	4201.406	4457.043	4856.291	19	4434.471	17216	1.285	34139	34139
86	96416.500	96416.500	3710.672	1.764	3607.739	3712.435	3868.121	19	3678.440	17729	0.088	96304	96304
91	207923.900	207923.900	3594.152	0.596	3511.583	3594.749	3686.573	10	1927.899	15504	0.887	207825	207825
96	331868.900	331868.900	3292.332	0.562	3224.146	3292.893	3363.271	12	2068.702	17542	0.639	331813	331813
101	75679.400	75679.400	4103.774	0.654	3826.773	4104.428	4302.112	19	3694.165	15628	0.185	75233	75233
106	23533.000	23533.000	4465.501	2.043	4225.274	4467.544	4677.965	16	3883.034	16442	0.526	22754	22754
111	40444.600	40444.600	3969.867	0.945	3826.232	3970.812	4076.747	12	2671.549	16454	0.070	40321	40321
116	134416.900	134416.900	4066.302	2.256	3826.943	4068.558	4247.249	13	2573.646	19932	0.048	134276	134276
121	231630.200	231627.900	3332.895	2.112	3276.166	3335.008	3410.308	13	2293.888	17967	3.917	231528	231528

Tabela B.16 Resultados para o problema $P||\sum \alpha_j E_j + \sum \beta_j T_j$ com 500 tarefas e 2 máquinas.

Inst.	Média custo GLS	Média custo PI	Média tempo GLS	Média tempo PI	Tempo mín.	Média tempo	Tempo máx.	Melhor iter. GLS	Melhor tempo GLS	Melhor arcos PI	Melhor tempo PI	Melhor GLS	MathGLS-IP
1	4444401.000	4444357.200	12287.639	0.850	11949.754	12288.489	12825.157	12	4022.581	7465	0.091	4443768	**4443699**
6	2837615.600	2837586.200	12321.389	1.106	11863.610	12322.495	12855.379	30	10024.178	12018	0.091	2836762	**2836748**
11	2656082.000	2656040.400	12102.720	1.214	11459.765	12103.934	12968.200	32	9286.886	7074	0.310	2655737	**2655724**
16	4429685.200	4429648.900	10495.470	0.994	10023.939	10496.464	10859.084	29	7795.699	6927	0.148	4428494	**4428442**
21	8612913.800	8612908.500	9041.865	0.782	8811.838	9042.647	9242.837	31	7214.896	11709	0.035	8612899	8612899
26	3624372.900	3624309.100	14327.557	0.762	13910.812	14328.319	14926.559	39	14340.957	3182	0.067	3623825	**3623818**
31	1838741.600	1838694.200	13197.470	1.042	12614.911	13198.512	13735.330	15	5092.766	11532	0.074	1837880	**1837854**
36	2049963.800	2049853.200	12284.356	1.014	11777.591	12285.370	12790.179	20	6282.492	11836	0.173	2048882	**2048812**
41	4324653.600	4324563.600	12419.135	0.830	11892.924	12419.965	12931.351	27	8468.150	8721	0.100	4324306	**4324240**
46	6468145.200	6468070.900	9285.758	0.838	8943.352	9286.596	9686.764	33	7711.994	7809	0.112	6468034	**6467897**
51	3385296.800	3385166.600	14047.119	1.208	13653.716	14048.326	14578.918	16	5858.014	7936	0.104	3384539	**3384379**
56	1111867.000	1111784.900	14673.602	0.932	14403.362	14674.534	15240.401	32	11920.625	5452	0.164	1111047	**1110786**
61	1199978.600	1199812.800	12706.758	0.903	12191.695	12707.660	13156.791	28	8808.658	9743	0.134	1199684	**1199449**
66	3360088.500	3360017.900	12022.167	0.915	11646.121	12023.082	12639.537	33	10454.822	10565	0.296	3359651	**3359600**
71	5790520.667	5790469.667	10602.759	1.051	10596.389	10603.809	10613.116	25	6581.051	11091	0.109	5790187	**5790099**
76	2629636.333	2629623.667	12560.521	0.892	12037.248	12561.413	12857.464	39	11905.486	6606	0.132	2627587	**2627586**
81	453041.000	452705.667	14135.936	0.788	14066.431	14136.724	14203.677	19	7281.851	6808	0.084	452913	**452699**
86	638427.500	638188.000	13603.770	0.685	13254.774	13604.454	13954.135	38	12695.512	8564	0.049	638370	**638103**
91	2029864.500	2029794.500	12202.073	0.727	12080.690	12202.800	12324.910	28	8620.391	8569	0.075	2029531	**2029460**
96	3415788.000	3415733.000	11602.358	0.773	11596.229	11603.131	11610.033	39	11586.081	4645	0.112	3415735	**3415681**
101	2671139.000	2671084.000	13841.596	0.809	13719.764	13842.405	13965.046	20	6957.949	13682	0.041	2670976	**2670975**
106	558977.000	558452.000	14382.615	0.849	14345.907	14383.464	14421.021	36	13842.469	5691	0.179	558686	**558340**
111	423485.500	423246.500	13536.878	1.056	13404.862	13537.934	13671.007	14	5064.156	11764	0.098	423343	**423164**
116	1278284.500	1278192.000	13747.387	0.790	13705.076	13748.176	13791.277	19	6737.218	7882	0.109	1278170	**1278039**
121	2805258.000	2805140.500	12245.569	0.951	11952.995	12246.520	12540.045	26	8264.236	9309	0.208	2804886	**2804828**

Tabela B.17 Resultados para o problema $P||\sum \alpha_j E_j + \sum \beta_j T_j$ com 500 tarefas e 4 máquinas.

Inst.	Média custo GLS	Média custo PI	Média tempo GLS	Média tempo PI	Tempo mín.	Média tempo	Tempo máx.	Melhor iter. GLS	Melhor tempo GLS	Melhor arcos PI	Melhor tempo PI	Melhor GLS	MathGLS-IP
1	1877891.500	1877835.500	14186.823	1.362	14122.821	14188.185	14253.549	25	11033.225	19466	0.219	1877571	**1877550**
6	1412618.000	1412515.000	14198.452	2.243	14172.266	14200.694	14229.123	21	13750.001	14132	1.858	1412529	**1412510**
11	1334823.000	1334785.000	18341.984	8.923	18350.907	18350.907	18350.907	33	16954.853	19924	5.761	1334823	**1334785**
16	2234018.000	2233941.000	14052.175	2.732	14054.907	14054.907	14054.907	25	11244.592	20097	1.508	2234018	**2233941**
21	4342114.000	4342096.000	13160.704	1.159	13161.863	13161.863	13161.863	39	13017.434	14494	0.260	4342114	**4342096**
26	1373176.000	1373147.000	14237.399	0.598	14237.997	14237.997	14237.997	24	14224.227	18466	0.022	1373176	**1373147**
31	913235.000	913235.000	14505.026	0.548	14505.574	14505.574	14505.574	20	14103.609	14702	0.012	913235	913235
36	1032277.000	1032277.000	14231.556	0.703	14232.259	14232.259	14232.259	23	11959.356	14347	0.058	1032277	1032277
41	2190230.000	2190143.000	14172.888	1.344	14174.232	14174.232	14174.232	26	11940.490	17995	0.331	2190230	**2190143**
46	3266680.000	3266675.000	13957.717	15.732	13973.449	13973.449	13973.449	39	13694.219	19063	8.251	3266680	**3266675**
51	1105392.000	1105139.000	21795.046	0.765	21795.811	21795.811	21795.811	27	20569.300	18913	0.088	1105392	**1105139**
56	550056.000	549926.000	21630.990	2.113	21633.103	21633.103	21633.103	31	20522.316	21123	0.788	550056	549926
61	607570.000	607547.000	14107.337	0.718	14108.055	14108.055	14108.055	21	11897.809	16799	0.061	607570	**607547**
66	1706425.000	1706424.000	14155.829	1.655	14157.484	14157.484	14157.484	30	13675.829	15836	0.419	1706425	**1706424**
71	2925770.000	2925770.000	14377.998	0.827	14378.825	14378.825	14378.825	30	12506.503	15218	0.130	2925770	2925770
76	653183.000	653183.000	14278.476	0.147	14278.623	14278.623	14278.623	17	14174.602	20472	0.007	653183	653183
81	221917.000	221897.000	22149.539	1.292	22150.831	22150.831	22150.831	30	20547.811	20237	0.294	221917	**221897**
86	331769.000	331769.000	21712.909	1.482	21714.391	21714.391	21714.391	37	20640.662	18178	0.387	331769	331769
91	1036762.000	1036762.000	17712.996	1.577	17714.573	17714.573	17714.573	38	16978.183	18173	0.561	1036762	1036762
96	1731728.000	1731728.000	16882.794	1.349	16884.143	16884.143	16884.143	25	11198.020	20177	0.316	1731728	1731728
101	638069.000	638069.000	22118.269	0.546	22118.815	22118.815	22118.815	21	21739.114	12183	0.019	638069	638069
106	203461.000	203461.000	22404.000	0.874	22404.874	22404.874	22404.874	20	22032.941	21403	0.016	203461	203461
111	222303.000	222277.000	22135.831	1.387	22137.218	22137.218	22137.218	36	21964.181	14429	0.332	222303	**222277**
116	655158.000	655158.000	21689.463	0.935	21690.398	21690.398	21690.398	28	19994.707	18978	0.036	655158	655158
121	1426022.000	1426011.000	21928.759	1.176	21929.935	21929.935	21929.935	32	21177.478	17307	0.138	1426022	**1426011**

Tabela B.18 Resultados para o problema $P||\sum \alpha_j E_j + \sum \beta_j T_j$ com 500 tarefas e 10 máquinas.

Inst.	Média custo GLS	Média custo PI	Média tempo GLS	Média tempo PI	Tempo mín.	Média tempo	Tempo máx.	Melhor iter. GLS	Melhor tempo GLS	Melhor arcos PI	Melhor tempo PI	Melhor GLS	MathGLS-IP
1	734346.333	734154.000	15013.866	0.692	14782.756	15014.559	15214.831	6	11722.520	8660	0.105	733982	733982
6	557947.667	557850.667	14153.752	0.661	14046.755	14154.413	14328.580	7	13167.458	2798	0.163	557811	557811
11	542898.333	542898.333	14368.146	0.866	14040.212	14369.012	14820.754	5	8952.026	9164	0.080	542858	542858
16	917507.333	917398.333	14660.982	2.852	14304.127	14663.834	15054.022	9	11746.396	9352	0.270	917388	917388
21	1779903.000	1779900.500	14620.164	0.925	14386.283	14621.089	14855.895	9	9354.858	3198	0.140	1779894	**1779889**
26	538055.500	538055.500	14809.218	0.517	14506.128	14809.736	15113.343	7	11715.255	14173	0.033	537668	537668
31	359436.500	359131.000	14892.267	0.489	14846.862	14892.756	14938.651	7	13439.348	3424	0.077	359131	359131
36	423523.500	423523.500	14297.127	0.666	14089.855	14297.793	14505.732	8	12880.491	3034	0.143	423496	423496
41	911061.000	911061.000	14757.097	5.861	14759.437	14762.958	14766.478	10	13095.219	7042	5.059	911009	911009
46	1346326.000	1346326.000	14220.749	0.894	14063.512	14221.643	14379.774	11	12978.656	8217	0.021	1346319	1346319
51	424627.000	424167.000	14553.936	0.454	14196.248	14554.390	14912.533	6	13084.660	8053	0.016	424167	424167
56	214531.500	214531.500	15263.031	0.570	15087.445	15263.601	15439.757	6	10534.715	11253	0.080	214452	214452
61	254217.000	254217.000	18835.854	0.774	18836.628	18836.628	18836.628	10	14963.966	5728	0.015	254217	254217
66	715416.000	715416.000	14742.209	0.740	14742.949	14742.949	14742.949	8	11269.298	4670	0.025	715416	715416
71	1207786.000	1207786.000	15171.042	0.870	15171.912	15171.912	15171.912	10	12786.706	3994	0.078	1207786	1207786
76	274940.000	274940.000	15325.369	0.106	15325.475	15325.475	15325.475	6	12552.354	9767	0.009	274940	274940
81	85108.000	85108.000	14689.886	0.097	14689.983	14689.983	14689.983	6	12226.358	9506	0.004	85108	85108
86	149819.000	149819.000	22387.241	0.923	22388.164	22388.164	22388.164	8	13156.896	7245	0.039	149819	149819
91	443802.000	443802.000	22018.895	0.834	22019.729	22019.729	22019.729	11	15903.912	7240	0.035	443802	443802
96	723593.000	723593.000	15101.754	0.779	15102.533	15102.533	15102.533	10	13935.695	12292	0.014	723593	723593
101	241147.000	241147.000	14005.426	0.027	14005.453	14005.453	14005.453	3	6724.622	659	0.009	241147	241147
106	48441.000	48441.000	15413.988	0.101	15414.089	15414.089	15414.089	0	908.218	10945	0.004	48441	48441
111	105220.000	105220.000	15451.662	0.042	15451.704	15451.704	15451.704	7	15337.623	3128	0.002	105220	105220
116	283908.000	283908.000	22677.023	0.967	22677.990	22677.990	22677.990	13	19028.886	8126	0.086	283908	283908
121	598623.000	598623.000	14768.071	0.725	14768.796	14768.796	14768.796	8	11081.950	6288	0.010	598623	598623