ESCALONAMENTO ÓTIMO DE TAREFAS PARA REDES DE CAMPO APLICADO A FOUNDATION™ FIELDBUS

Gabriel de Albuquerque Gleizer

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientadores: Liu Hsu
Danielle Zyngier

Rio de Janeiro
Janeiro de 2013

ESCALONAMENTO ÓTIMO DE TAREFAS PARA REDES DE CAMPO APLICADO A FOUNDATION™ FIELDBUS

Gabriel de Albuquerque Gleizer

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

Prof. Liu Hsu, Dr. d'Etat.

Prof. Dennis Brandão, D.Sc.

Prof. Nelson Maculan Filho, D.Habil.

Prof. Ramon Romankevicius Costa, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
JANEIRO DE 2013

Gleizer, Gabriel de Albuquerque

Escalonamento Ótimo de Tarefas para Redes de Campo Aplicado a FOUNDATION™ Fieldbus/Gabriel de Albuquerque Gleizer. – Rio de Janeiro: UFRJ/COPPE, 2013.

XXII, 163 p.: il.; 29, 7cm.

Orientadores: Liu Hsu

Danielle Zyngier

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2013.

Referências Bibliográficas: p. 159 – 163.

1. Redes de Campo. 2. FOUNDATION™ Fieldbus. 3. Escalonamento de Tarefas. 4. Programação Linear Inteira Mista. 5. Sistemas de Controle em Rede. 6. Otimização. I. Hsu, Liu *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

Agradecimentos

Toda seção de agradecimentos é injusta. Sempre esquecemos de registrar gratidão a alguém que foi fundamental na nossa jornada. Mas eu seria muito mais injusto se não escrevesse uma seção de agradecimentos.

Em primeiro lugar, como não poderia deixar de ser, agradeço a Deus. Não só por ser a causa primária de todas as coisas, mas pela vida maravilhosa que Ele está me concedendo e pelas pessoas fantásticas que colocou no meu caminho. A essas pessoas que quero agradecer agora.

Agradeço à minha família pelo apoio, por nunca ter tentado influenciar para que eu saísse do incerto caminho da pesquisa, por me conhecer melhor do que eu mesmo. Agradeço à minha mãe Luciana por ter sido mãe e pai, pelo carinho e paciência, pelo amor incondicional e por vezes imerecido, pelos valores que me passou. Agradeço ao meu falecido pai Adilson, que me ensinou e forjou boa parte do meu caráter, e que me incentivou tanto a trilhar o caminho da Engenharia. Ele certamente estaria orgulhoso. Agradeço ao meu irmão Bernardo, que é o melhor irmão que alguém pode ter, sempre um verdadeiro amigo e parceiro. Agradeço à minha avó Maria pelos conselhos e incentivos relacionados ao mundo acadêmico e à minha avó Dilce pelo carinho infinito e disponibilidade constante. Agradeço a meus primos e tios pela força e pelos momentos alegres que tanto ajudam a desembaraçar a mente. Agradeço demais à minha cunhada Thalita por ser minha companheira nas aflições e diversões acadêmicas, e agradeço muito à Aparecida, minha segunda mãe, pelo amor que me dedica até hoje.

Neste curto período de Mestrado várias mudanças ocorreram na minha vida. Uma delas foi a entrada da Eliza, minha namorada. Quero muito agradecer a ela por ser tão maravilhosa comigo e por me trazer tanta alegria. Ao mesmo tempo, ela sempre entendeu a necessidade que eu tive várias vezes de abdicar de passar mais tempo com ela para me dedicar a essa Dissertação. Em vez de reclamar, ela sempre me incentivava e me dava ideias para conciliar o estudo, o trabalho e a vida. Só posso agradecer e tentar retribuir agora que ela começou sua Dissertação. Ela certamente tornou essa caminhada mais leve e prazerosa.

Quero agradecer muito aos meus orientadores. Primeiro o Liu que me acolheu para o Mestrado, me ajudou a ter bom senso e maturidade na seara da pesquisa e me

deu a oportunidade de trabalhar no LEAD. Foi lá onde aprendi muito não só sobre Fieldbus, como também a trabalhar em equipe, a comunicar ideias e compartilhar um conhecimento que estávamos consolidando. Dentro do contexto de LEAD, quero agradecer aos meus parceiros Rodrigo Carneiro, Aurélio Lima e Priscilla Dinau, que foram tão importantes para esse aprendizado. Agradeço muito ao Miguel Borges, criador do projeto e um amigaço, pessoa da melhor qualidade. Devo quase tudo que sei sobre Fieldbus ao consultor Augusto Pereira, que conduziu com maestria nossos estudos e me ensinou muito sobre carreira profissional. Agradeço ao Celso Molinaro, sempre provocador, ao Danilo Prates, que nos ajudou a montar nosso curso FF – e com isso aprender mais ainda, ao Walter de Mattos, trabalhador incansável e pessoa de grande caráter. Agradeço à Telma Pará, que sempre nos deu muito suporte, ao Trevor Dobbin pelas longos e empolgantes debates. Agradeço aos seguranças e pessoal da limpeza, sempre muito prestativos e companheiros, em especial o Jesus, uma pessoa sensacional. E não posso deixar de agradecer ao nosso instrutor no curso da Fieldbus Foundation, Ian Verhappen – não só aprendi muito sobre FF com ele; foi conversando com ele que surgiu a ideia de eu fazer a Dissertação nesse tema.

Dentre as pessoas que Deus colocou na minha vida e que foram determinantes para esse trabalho, sem dúvida minha orientadora Danielle foi um presente. Ela passou pouco mais de um ano no Brasil, e nesse período tive a sorte de conhecê-la, de ela aceitar ser minha orientadora e ter paciência ao me introduzir ao vasto mundo da Pesquisa Operacional. Eu, controleiro de formação, me vi perdido ao tentar resolver o problema que será apresentado nessa Dissertação. A Danielle soube me guiar no aprendizado de algo totalmente novo, na condução dos meus estudos, no desenvolvimento das soluções e na clara apresentação dos resultados. Se não fui o melhor aluno, ela foi a melhor orientadora. Não posso agradecer o suficiente. Agradeço também ao pessoal do Lades pela amizade e pela força: novamente minha namorada Eliza, meus amigaços Leonardo Orenstein e Rafael Bendia, Evandro, Lizandro, Franklin, Simone, Danilo, Paiva, Trica e prof. Argimiro.

Também não posso agradecer o suficiente aos meus grandes amigos desde o tempo de graduação, muitos dos quais continuaram comigo na jornada do Mestrado – e alguns nomes se repetem: Alex Neves, Anderson Sangreman, André Barbosa, Aurélio Lima, Daniel Vaz, Diogo Nomiya, Leonardo Orenstein, Lucas Vargas, Marcelo Prado, Maurício Dias, Rafael Bendia e Thiago Camanho. Sem eles tudo teria sido bem sem graça e muito mais difícil. Agradeço às suas respectivas: Maria Fernanda, Cecília, Maria Pía, Monica e Gabriela pelas boas conversas. Agradeço ao filósofo Hugo Noberga e ao poeta Andrei Battistel pela sabedoria e pelas cervejas. E sou muito grato a eternos irmãos que infelizmente não tenho tido tanto contato quanto eu gostaria: Rafael Lofiego, Carol Valentim e Carlos Eduardo Santos, Thais Businaro, Leonardo Bazílio, Michel Cardozo, Daniel Xavier, Felipe Silva, Leonardo

Hernique, Jaqueline Lopes e Gabriela Gama.

Dentre as mudanças por que passei nesse período de Mestrado, uma muito grande foi quando eu deixei o LEAD e me juntei ao novo Centro de Pesquisa da GE. E quero muito agradecer ao pessoal que eu conheci ou reencontrei lá, que me incentivou a finalizar o Mestrado enquanto eu me adaptava a um ambiente totalmente novo. Não posso deixar de citar Bruno Leão, Rodrigo Salim, Álvaro Fialho, Débora Reis, Luiz Douat, Alessandro Dutra, Antonio Leite e Ítalo Oliveira pelas dicas e pela força. Agradeço também ao Ken Herd pelas palavras de apoio e por passar sua semelhante experiência. E agradeço muito ao Alexandre da Silva, meu primeiro incentivador a seguir a carreira da pesquisa e que hoje continua me dando uma assistência quase paternal nessa caminhada. Está apenas começando!

Por fim, quero agradecer aos membros da banca Ramon Costa, Nelson Maculan e Dennis Brandão por fazerem parte disso tudo; à Daniele do PEE/COPPE, por me salvar diversas vezes com as datas e documentações; aos professores que tive pela vida, por me ensinarem e me motivarem a aprender e descobri o que sei; ao governo brasileiro pelo investimento numa pós-graduação pública de qualidade; e ao povo brasileiro por financiar os estudos de uma parcela tão pequena desse mesmo povo. Espero contribuir à altura para meu país.

Muito, muito obrigado,

Gabriel de Albuquerque Gleizer.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

ESCALONAMENTO ÓTIMO DE TAREFAS PARA REDES DE CAMPO APLICADO A FOUNDATION™ FIELDBUS

Gabriel de Albuquerque Gleizer

Janeiro/2013

Orientadores: Liu Hsu
Danielle Zyngier

Programa: Engenharia Elétrica

Nesta dissertação de Mestrado, modelos baseados em Programação Linear Inteira Mista são aplicados no problema de escalonamento para FOUNDATION™ Fieldbus (FF). Problemas de otimização são formulados para minimizar um compromisso entre maximizar o agrupamento das comunicações, minimizar atrasos nas malhas de controle e minimizar o tempo final de execuções, tanto para o caso de ciclos de execução comuns quanto para o caso de ciclos de execução distintos. Resultados são comparados com algumas agendas de um Sistema Digital de Controle Distribuído (SDCD) comercial.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

OPTIMAL TASK SCHEDULING FOR FIELDBUSES APPLIED TO FOUNDATION™ FIELDBUS

Gabriel de Albuquerque Gleizer

January/2013

Advisors: Liu Hsu
Danielle Zyngier

Department: Electrical Engineering

In this Masters Degree dissertation, models based on Mixed Integer Linear Programming are applied for the FOUNDATION™ Fieldbus (FF) scheduling problem. An optimization problem is then formulated to minimize a trade-off between maximizing communications grouping, minimizing delays and minimizing the final time of executions, for both the case where execution cycles are equal and the case where the cycles are different. Results are compared with schedules obtained from a commercial Distributed Control System.

Sumário

Lista de Figuras

Lista de Tabelas

Nomenclatura

Índices .

c	Ordem de repetição de uma tarefa no macrociclo.
i	Tarefa.
j	Dispositivo.
m	Malha de controle.
n	Ponto de evento.
net	Dispositivo rede.
r	*Readback*.

Conjuntos .

C_i	Conjunto ordenado de execuções da tarefa i num macrociclo.
E	Pares de tarefas equivalentes entre si.
I	Tarefas.
I_m	Tarefas da malha de controle m.
I_j	Tarefas do dispositivo j.
I_{net}	Comunicações (CDs).
$Ia_{i,c}$	Execuções de tarefas necessariamente após (i, c).
$Ib_{i,c}$	Execuções de tarefas necessariamente antes de (i, c).
J	Dispositivos.
L_m	Ligações entre duas tarefas na malha de controle m.
$L_{j,m}$	Relações de precedência de qualquer ordem entre duas tarefas pertencentes ao dispositivo j na malha de controle m.
$L^1_{j,m}$	Relações de precedência de primeira ordem entre duas tarefas do dispositivo j na malha de controle m.
M	Malhas de controle.
N_j	Pontos de evento do dispositivo j.
P_j	Pares de tarefas do dispositivo j em que é possível haver sobreposição.
Q	Pares de execuções de CDs em que é possível não haver uma lacuna.
$\overline{Q}$	Pares de execuções de CDs entre os quais necessariamente há uma lacuna.
Q^*	Outro conjunto de pares de execuções de CDs em que é possível não haver uma lacuna.

RB Comunicações *readback*.

LR_r Tarefas enviante e recebente do *readback* r.

Parâmetros .

H Tempo do macrociclo, ou o horizonte de tempo.

H_i Tempo de ciclo da tarefa i.

R Período de tempo em que se podem executar CDs.

Te_i Tempo de execução da tarefa i.

Z Número total de execuções de CDs no macrociclo.

a_m Peso do atraso da malha de controle m.

d_m MDC entre os números de ciclos de todas as tarefas pertencentes à malha de controle m..

$d_{net,i,i'}$ MDC entre os números de ciclos dos CDs i e i', num macrociclo.

α Peso do critério de separação na função objetivo (caso Monociclo).

α' Peso dos critérios de lacunas e atrasos (caso Multiciclo).

β Peso do critério de atrasos na função objetivo (caso Monociclo).

γ Peso temporal de uma lacuna.

Variáveis .

D Somatório ponderado dos atrasos das malhas de controle.

D_m Critério de atraso para a malha de controle m.

F Função objetivo composta.

G Critério de lacunas entre CDs.

S Critério de separação de CDs.

TF Critério de tempo final das execuções.

$TfCDs$ Tempo final do último CD do macrociclo.

Tf_i Tempo final da tarefa i.

$\overline{Tf}_i$ Tempo final base de execução da tarefa i.

$Tf_{i,c}$ Tempo final da c-ésima execução da tarefa i.

$TsCDs$ Tempo inicial do primeiro CD do macrociclo.

Ts_i Tempo inicial da tarefa i.

$\overline{Ts}_i$ Tempo inicial base de execução da tarefa i.

$Ts_{i,c}$ Tempo inicial da c-ésima execução da tarefa i.

$\overline{c}_i$ Variável inteira que indica o ciclo base da tarefa i.

$g_{i,c,i',c'}$ Variável binária que determina se a lacuna entre c-ésima execução da CD i e a c'-ésima execução do CD i' terminar.

g_n Variável binária que indica a lacuna entre o n-ésimo e o $(n+1)$-ésimo CDs.

$o_{j,i,c,i',c'}$ Variável binária que determina se a c-ésima execução da tarefa i começa antes ou depois de a c'-ésima execução da tarefa i' terminar no dispo-

sitivo j.

$o_{j,i,i'}$ Variável binária que determina se a tarefa i começa antes ou depois de a tarefa i' terminar no dispositivo j.

$w_{i,c,j,n}$ Variável binária para o início de c-ésima execução da tarefa i no dispositivo j no ponto de evento n.

$w_{i,j,n}$ Variável binária que aloca o início de execução da tarefa i do dispositivo j no ponto de evento n.

y_r Variável binária que aloca o *readback* r antes da tarefa recebente ou depois da enviante.

Operadores .

π_k Projeção canônica de um produto cartesiano no seu k-ésimo conjunto.

$\times$ Produto cartesiano entre conjuntos.

$|\cdot|$ Cardinalidade de conjunto.

Lista de Acrônimos

AI Entrada Analógica, do inglês *Analog Input*.

AO Saída Analógica, do inglês *Analog Output*.

AP Atribuição de Prioridades.

AWG *American Wire Gauge*.

BFS Busca em Largura, do inglês *Breadth First Search*.

C/S Cliente/servidor.

CAN *Controller Area Network*.

CBC *Coin-OR Branch & Cut*.

CD Comunicação publicante/assinante, do inglês *Compel Data*.

CF Capability File.

DC Corrente contínua, do inglês *direct current*.

DDFC Diagrama Dispositivo-Função-Comunicação.

E/S Entrada/saída.

EDF *Earliest Deadline First*.

EP Elemento de Processamento.

F/P Fonte/poço.

FBPS Acoplador Fieldbus, do inglês *FieldBus Power Supply*.

FF FOUNDATION™ Fieldbus.

FF H1 FOUNDATION™ Fieldbus H1.

GLPK *GNU Linear Programming Kit.*

GMPL *GNU Mathematical Programming Language.*

ISEL Seletor de Entrada, do inglês *Input Selector*.

IT Integrador.

LAS Escalonador Ativo de Enlaces, do inglês *Link Active Scheduler*.

LP Programação Linear, do inglês *Linear Programming*.

MADB Máximos Limites de Atraso Permissíveis, do inglês *Maximum Allowable Delay Bounds*.

MDC Máximo Divisor Comum.

MILM Modelo Linear Inteiro Misto, do inglês *Mixed Integer Linear Model*.

MILP Programação Linear Inteira Mista, do inglês *Mixed Integer Linear Programming*.

MMA Mínimo Macrociclo Admissível.

MMC Mínimo múltiplo comum.

NCS Sistemas de Controle em Rede, do inglês *Network Control Systems*.

OSI *Open Systems Interconnection.*

P/A Publicante/assinante.

PID Proporcional-Integral-Derivativo.

QoC Qualidade de Controle, do inglês *Quality of Control*.

QoS Qualidade de Serviço, do inglês *Quality of Service*.

SCIP *Solving Constraint Integer Programs.*

SDCD Sistema Digital de Controle Distribuído.

SPLTR Divisor de Saída, do inglês *Output Splitter*.

Capítulo 1

Introdução

No atual estado da tecnologia, redes de dispositivos eletrônicos e/ou computadores são usadas em vários níveis de uma unidade industrial, desde o chão de fábrica até o nível corporativo. As redes usadas no chão de fábrica da indústria de processos são chamadas de redes de campo. Estas são usadas não apenas para dados de controle e monitoração, como também para configuração e diagnóstico. O uso de redes compartilhadas multipropósito para conectar elementos distribuídos espacialmente resulta em arquiteturas flexíveis e geralmente reduzem os custos de instalação e manutenção [20].

Redes de campo são usadas para conectar instrumentos, i.e., sensores e atuadores (neste caso, na maior parte das vezes transmissores e posicionadores de válvula), a uma interface para o controlador, que funciona como um módulo de entrada/saída (E/S). Alguns protocolos de redes de campo definem o compartilhamento de dados de E/S, configuração e diagnóstico apenas, como no caso do Profibus PA. Outros, como FOUNDATION™ Fieldbus H1 (FF H1), definem também Controle no Campo, permitindo que as funções de controle sejam executadas nos instrumentos, criando um sistema de controle distribuído malha por malha no campo. Para este último caso, surge a necessidade de um escalonamento de tarefas eficiente de forma a tirar total proveito das capacidades de redes e sistemas digitais, sem com isso degradar o desempenho das malhas de controle.

Um sistema de controle distribuído é tipicamente caracterizado pela presença de dois tipos de processos, os periódicos e os assíncronos [7]. Um processo periódico executa atividades altamente repetitivas a uma frequência fixa. É o caso da amostragem de um sinal analógico de um sensor. Esse tipo de processo é majoritariamente utilizado para controle. Um processo assíncrono, por outro lado, evolui no tempo de maneira imprevisível *a priori*. Ele é afetado por eventos que ocorram na planta, como alarmes, ou por eventos fora dela, como mudanças de configuração.

O problema de gerenciamento de uma rede de controle distribuído, como a FF H1, consiste em uma estratégia de escalonamento desses dois tipos de processos.

Tal estratégia deve considerar a confiabilidade temporal das tarefas periódicas, ao mesmo tempo garantindo a execução das tarefas assíncronas, conforme elas forem requisitadas, com algum nível de qualidade, ou seja, com poucas interrupções e com velocidade aceitável para o usuário. Enquanto as tarefas periódicas podem ser organizadas antes do tempo de execução, por serem configuradas offline, nada se pode fazer de antemão com as tarefas assíncronas, se não uma estratégia que aproveite o espaço de tempo ocioso da rede, quando esta não está sendo usada para comunicações periódicas.

O presente trabalho aborda a organização temporal ótima das tarefas periódicas de uma rede de controle distribuído, mais especificamente voltado para a rede FF H1. Essa organização temporal é chamada de escalonamento de tarefas e é um campo multidisciplinar: pode ser aplicado a gerenciamento de tarefas em processadores, programação da produção em indústrias, gerenciamento de tráfego no setor de transportes e gerenciamento de projetos.

1.1 Visão Geral do Funcionamento de um Rede FOUNDATION™ Fieldbus

O protocolo FF H1 define três tipos de comunicação. A comunicação periódica é chamada de publicante/assinante (P/A) e é usada apenas para dados de controle, ou seja, para valores que o usuário exige que sejam atualizados periodicamente. As comunicações assíncronas são de dois tipos: cliente/servidor (C/S), usada para configuração e dados requisitados pelo usuário, como variáveis de monitoração; e fonte/poço (F/P), que é usada para alarmes.

Como malhas de controle são configuradas antes que o sistema entre em operação, um escalonamento pré-execução precisa ser feito para cada função de cada dispositivo na rede. Assim, o dispositivo terá a informação das tarefas a executar, dos dados de que precisa, dos dados que deve enviar e do instante e periodicidade em que ele deve executar cada tarefa e receber/enviar cada informação. Essa periodicidade é chamada de ciclo, e cada tarefa tem o seu. O mínimo múltiplo comum (MMC) entre os ciclos das tarefas de uma rede é chamado de Macrociclo da rede. Muitos dos sistemas de controle comerciais obrigam que todos os ciclos numa rede sejam os mesmos e, portanto, iguais ao Macrociclo.

1.2 Motivação

O escalonamento de tarefas em redes de campo foi um ponto consideravelmente estudado nos anos 1990 [7, 17], época em que os padrões e protocolos foram de-

senvolvidos e começaram a se estabelecer. Com a maturidade das tecnologias, o assunto passou a ser menos abordado, já que boa parte dos protocolos define bem suas rotinas de escalonamento.

O assunto voltou a ser um ponto de interesse devido a recentes avanços na infraestrutura de redes de campo. Uma típica rede FF H1 não contém mais do que doze dispositivos: a queda de tensão na rede, que aumenta com o número de dispositivos, limitava a possibilidade de ampliação do número de dispositivos da rede. Com o surgimento de acopladores com tensão de saída mais alta (28V ou 32V, contra os 24V anteriores), isso deixou de ser um fator limitante e a banda de comunicação, de apenas 31.25 kbps, passou a impor restrições ao número de dispositivos. Se o escalonamento não for feito de uma maneira eficiente, poucas malhas de controle podem ser configuradas numa rede. Isso pode ser visto com mais detalhe no Capítulo A, Seção A.3.1. Em [15], o guia de engenharia de sistemas FF, recomenda-se no máximo 4 malhas de controle para uma rede com macrociclo de 1000 ms, 2 malhas para 500 ms e 1 malha para 250 ms em função das limitações de banda.

Com o aumento da tensão de saída dos acopladores, o uso de cabos com pares de fios 16AWG e o crescimento da confiabilidade das comunicações da rede, otimização do macrociclo passou a ser um dos pontos principais em FOUNDATION™ Fieldbus. Os principais softwares de Sistemas Digitais de Controle Distribuído (SDCDs) possuem uma opção para otimização do macrociclo ou equivalente para redes FF H1. Os algoritmos são proprietários e buscam fazer agendas mais eficientes. Além disso, poucos fabricantes publicaram resultados de otimização. Por exemplo, a Honeywell lançou um artigo branco [21] explicando a ideia do algoritmo de seu SDCD.

Um outro problema que surge com o uso de redes para sistemas de controle são os atrasos induzidos pela rede. Esses atrasos vão além do típico atraso de sistemas digitais, que é basicamente o tempo de processamento, mas incluem o tempo de comunicação e, num meio distribuído, eventuais esperas para a comunicação. Minimizar essa espera pode ser possível através de uma boa estratégia de escalonamento.

A principal motivação, portanto, deste trabalho vai além do mero interesse recente de fabricantes na tecnologia de otimização do macrociclo. Abordando o problema de forma direta e chegando a resultados de fato otimizados, torna-se possível ao usuário avaliar se seu projeto de rede de controle é viável, comportando todas as malhas de controle desejadas com bom desempenho e garantindo o máximo de qualidade de serviço na rede. Com a otimização, é possível aproveitar ao máximo a rede, maximizando o número de dispositivos a ela associados sem perda de qualidade. No fim, o usuário fará real proveito do uso de redes de campo com reduzido número de redes – o que leva a menores custos de infraestrutura de automação e, posteriormente, de manutenção.

1.3 Propostas e Contribuição do Trabalho

Neste trabalho, o problema de otimização do escalonamento de tarefas periódicas em sistemas de controle distribuído em rede é abordado de forma direta, através de uma modelagem matemática rigorosa e ferramentas de otimização. Os objetivos da otimização são diretamente voltados a problemas de uma rede de controle: qualidade de controle e qualidade de serviço, conceitos abordados em [18]. Neste trabalho, a qualidade de controle é alcançada através da minimização dos atrasos, enquanto a qualidade de serviço está atrelada à maximização do tempo disponível para comunicações assíncronas. Cabe ressaltar que, apesar de voltado ao problema de controle distribuído, o presente trabalho se aplica a uma configuração com controle centralizado no *host*.

O problema de escalonamento de tarefas FF é de natureza combinatória e NP-difícil. Em problema dessa natureza, a enumeração total das soluções torna-se inviável mesmo com um modesto número de variáveis. Levando isso em conta, propõe-se aqui a solução do problema via Programação Linear Inteira Mista (MILP, do inglês *Mixed Integer Linear Programming*), que é uma alternativa mais eficiente à simples enumeração. Alguns modelos matemáticos para o problema foram testados e comparados, advindos da literatura da Engenharia Química, para planejamento da produção, em particular de processos em batelada, e da Ciência da Computação, para programação paralela. Algumas simplificações foram aplicadas aos modelos, pois o problema de escalonamento FF é um caso particular de ambos os supracitados. Além disso, restrições e objetivos específicos para o problema FF foram considerados.

De aqui em diante a periodicidade de uma tarefa será denominada ciclo. O problema FF foi abordado em dois casos distintos: o caso monociclo e o caso multiciclo. O caso monociclo é aquele em que todas as malhas da rede possuem o mesmo ciclo. Neste caso, compararam-se os resultados com os de um sistema de controle comercial, o DeltaV 8.3, da Emerson Process Management [31]. O caso multiciclo, mais complexo, considera ciclos distintos numa rede de controle, com a apresentação dos resultados para testes de caso representativos.

1.4 Organização da Dissertação

Em seguida (Capítulo 2), uma revisão bibliográfica é descrita, levando em conta não apenas o problema de escalonamento FF, como também outra áreas que abordam problemas de escalonamento semelhantes. Essas áreas incluem sistemas digitais concentrados em um único processador, sistemas digitais de controle, sistemas de controle em rede, sistemas digitais distribuídos e processos em batelada. A área

de sistemas de controle em rede é também pesquisada fora do tema do escalonamento visando o entendimento dos problemas que podem ser mitigados com um bom escalonamento de tarefas.

No capítulo 3, descreve-se o problema de escalonamento para redes FF e similares. Em seguida, a modelagem baseada em Programação Linear Inteira Mista é descrita com duas formulações distintas: Intervalos de Tempo e Sobreposição. Em seguida, três critérios de otimização são então propostos e justificados. Por fim, discute-se um pré-processamento inicial que é recomendado para agilizar o processo de resolução da otimização.

O modelo é posto em prática em dois estudos de caso de tamanho industrial no capítulo 4. O desempenho da otimização é detalhado e as agendas resultantes são comparadas com as geradas por um SDCD comercial, o DeltaV 8.3.

O Capítulo 5 apresenta a descrição e a modelagem para o caso multiciclo. Reformulações na função objetivo são propostas, e mudanças nos modelos do Capítulo 3 são realizadas. Além disso, um modelo mais ajustado com a representação por Sobreposição é incluído.

Os resultados para o caso multiciclo são apresentados no Capítulo 6, com quatro estudos de caso representativos. Comparações entre os diferentes modelos são apresentadas e discutidas.

Por fim, análises finais, conclusões e propostas de trabalhos futuros são feitas no capítulo 7.

O Apêndice contém o Capítulo A, que introduz temas sobre FOUNDATION™ Fieldbus que são necessários para a compreensão do problema de escalonamento e de sua importância do ponto de vista do usuário. No Capítulo B do Apêndice, encontram-se os códigos usados para modelagem e entrada de dados dos problemas estudados.

Capítulo 2

Revisão Bibliográfica

O problema de planejamento e escalonamento de tarefas é um assunto multidisciplinar. Portanto, a pesquisa na literatura não poderia se limitar ao estudo do problema em FOUNDATION™ Fieldbus. Por mais que houvesse resultados importantes para tal protocolo de comunicação, seria importante visitar outras áreas de sistemas em tempo real e até mesmo outras áreas da Engenharia e da Matemática que estudam o problema. Dentre tais áreas, uma em que o problema de planejamento e escalonamento de tarefas é primordial é o de Pesquisa Operacional, que se permeia nos diversos ramos da Engenharia.

Para iniciar o estudo, as seguintes perguntas foram levantadas:

- O que já foi feito em termos de escalonamento de tarefas para FOUNDATION™ Fieldbus? E para redes de controle de uma maneira geral? E para outros tipos de sistemas distribuídos em tempo real? O que pode ser aproveitado destes trabalhos?
- Quais são as outras áreas da Engenharia em que o tema Escalonamento de Tarefas é abordado? Quais são as técnicas utilizadas nessas áreas?
- Dentre as diferentes técnicas para escalonamento de tarefas, qual é a mais adequada para o caso FF H1, com o objetivo de otimizar certos critérios?
- Em termos de otimização, quais critérios devem ser considerados? O que alterar o escalonamento das tarefas influencia num Sistema de Controle em Rede?

Para responder as perguntas acima, foi necessário e fundamental revisar a literatura para os seguintes temas:

- Escalonamento de tarefas

  na Engenharia Elétrica e na Ciência da Computação:

para redes FOUNDATION™ Fieldbus e redes de campo similares;

para sistemas distribuídos em geral;

em Pesquisa Operacional:

para Processos em Batelada;

- Sistemas de Controle em Rede.

As seções seguintes dividem a revisão bibliográfica nos temas acima. A seção 2.1 contém a descrição de trabalhos relacionados ao problema FF H1. Tais trabalhos incluem o problema de planejamento e escalonamento de tarefas em sistemas distribuídos em tempo real, incluindo sistemas em rede e sistemas multiprocessados. A revisão destes assuntos revelou poucos trabalhos visando a otimização de critérios, especialmente relacionados ao desempenho do sistema de controle. A procura por trabalhos que visam a otimização de sistemas distribuídos, relaxando-se a restrição de tempo real, é relatada na seção 2.2, incluindo sistemas de computação multiprocessados e sistemas de produção em batelada da Engenharia Química. Uma breve apresentação de Otimização Inteira Mista é feita na seção 2.3. Finalmente, a revisão da subseção 2.4 é direcionada a Sistemas de Controle em Rede, com seus problemas e desafios, buscando encaixar FOUNDATION™ Fieldbus nesse contexto.

2.1 Planejamento e Escalonamento de Tarefas em Sistemas Distribuídos em Tempo Real

Planejamento e escalonamento de tarefas é um assunto fundamental no projeto de sistemas em tempo real. Em sistemas dessa natureza, diferentes tarefas requisitam o uso de um ou mais processadores. Essas tarefas podem ou não ser independentes entre si e cada uma tem um um tempo de execução de pior caso e prazo para terminar sua execução. O objetivo consiste em alocar as tarefas a elementos de processamento e definir seus instantes de execução, a fim de se satisfazer algum objetivo de interesse. A revisão bibliográfica do tema inicia na década de 1990, quando houve os estudos iniciais mais importantes e a delineação das sub-áreas e desafios.

Nos casos em que as tarefas não podem ultrapassar seus prazos, trata-se de um sistema em tempo real rígido. Num trabalho seminal em sistemas mono-processador [29], Liu e Layland determinam os limites da estratégia de escalonamento usando prioridades fixas e propõem uma estratégia de atribuição de prioridades baseada no prazo mais próximo, que foi mais tarde chamada de *Earliest Deadline First* (EDF). A técnica de Atribuição de Prioridades (AP) online, isto é, à medida que as tarefas chegam ao processador, tornou-se muito utilizada no projeto de sistemas em tempo real, em função da eficiência, da simplicidade e da adaptabilidade.

No início dos anos 1990, Xu e Parnas [46, 47] tratam o problema de escalonamento considerando restrições de sequenciamento e exclusão, notando a necessidade da criação de agendas antes do tempo de execução. Essa maneira de escalonar tarefas é chamada de pré-tempo de execução e surge em contraponto às técnicas em tempo de execução, onde as estratégias de AP vigoram. Para a utilização desta técnica, é necessário algum conhecimento *a priori* da estrutura das tarefas a serem agendadas, mas como muitas das tarefas de um sistema em tempo real rígido são periódicas, isso torna-se viável. Esses autores são os primeiros a resolver problema de escalonamento para um processador considerando instantes de lançamento, prazos, relações de sequenciamento e de exclusão em [46]. Para tal, uma série de restrições lógicas é formulada e um algoritmo de busca em árvore é então utilizado para encontrar uma agenda viável e que minimize o atraso de cada sequencia de tarefas. O atraso nesse caso é definido como a diferença entre o tempo final de execução da tarefa em questão e seu prazo. Os mesmos autores discutem as vantagens do escalonamento pré-tempo de execução em [47]. Segundo eles, a previsibilidade do comportamento do sistema é a característica mais importante para satisfazer restrições temporais em sistemas em tempo real rígidos. Isso faz com que o escalonamento pré-tempo de execução seja desejável, por ser a única técnica capaz de prover tal previsibilidade num sistema complexo. O mesmo trabalho apresenta diversos algoritmos pré-tempo de execução existentes na literatura da época.

Com o advento, a evolução e a subsequente ampla utilização de sistemas distribuídos, surgiu a necessidade de se estudar o escalonamento e planejamento de tarefas para estes sistemas. Técnicas offline (ou pré-tempo de execução) e online (ou em tempo de execução) foram adaptadas das técnicas para sistemas mono-processador ou recriadas pensando nos desafios de ambientes multiprocessados. Xu estende seu trabalho do caso mono-processador em [46] para o caso multi-processador em [45]. Neste trabalho, Xu considera canais de comunicação dedicados entre os processadores e desconsidera atrasos de comunicação. Em [1], Abdelzaher e Shin consideram tais atrasos no que chamam de escalonamento combinado de tarefas e comunicações. Como no trabalho de Xu supracitado, ainda há a consideração de canais dedicados entre os processadores, o que impede aplicações desse tipo de algoritmo em sistemas em rede como uma rede de campo. Mais tarde, Peng et al [35] lançam para o mesmo problema um algoritmo de busca em árvore agora visando a minimização do que chamam de perigo do sistema. O perigo do sistema é definido como o máximo atraso (ou tempo de resposta), normalizado pelo prazo, entre todas as tarefas do sistema. A diferença para os trabalhos anteriores é que este se trata de um problema minimax que garante que todas as tarefas estão no prazo, enquanto minimizar uma soma ponderada de atrasos pode permitir alguma tarefa fora do prazo.

Ainda nos anos 1990, numa pesquisa de grande importância para a área de sis-

temas em tempo real em rede, Cardeira e Mammeri [6] avaliam a aplicabilidade das técnicas então existentes para sistemas em rede. Como Abdelzaher e Shin, eles notam as semelhanças entre tarefas e mensagens e a possibilidade de tratá-los de forma equivalente em termos de escalonamento. Os autores apresentam as novas restrições impostas por sistemas com transmissores/transdutores inteligentes em aplicações com redes de campo como o WorldFIP (protocolo posteriormente integrado ao FOUNDATION™ Fieldbus). Eles definem os algoritmos existentes adaptados ao problema. A maior parte dos algoritmos até então propostos considerava a possibilidade de preempção, isto é, interrupção de uma tarefa e a possível continuação da mesma posteriormente, mas os autores notaram a necessidade de algoritmos de escalonamento ótimos não-preemptivos para escalonamento periódico. Eles afirmam que a analogia feita entre algoritmos preemptivos e não-preemptivos não é permitida se os tempos de transmissão das mensagens for considerável. Os autores comentam que escalonamento não-preemptivo em um processador com prazos e tempos de lançamento arbitrários é um problema NP-completo.

Em 1998, Ola Redell analisou planejamento e escalonamento globais para sistemas distribuídos visando aplicações de controle em tempo real em [36]. Ele discute que objetivos devem ser considerados em termos de otimização, notando o conflito entre objetivos como finalização das execuções e ocupação da rede. Dentre os algoritmos pesquisados por Redell, destacam-se heurísticas, *Branch & Bound*, programação de restrições e meta-heurísticas como algoritmos genéticos, pesquisa tabu e arrefecimento simulado. Destas técnicas, apenas técnicas de *Branch & Bound* podem garantir um ótimo global. No entanto, o poder computacional na época era muito limitado e a natureza NP-completa de problemas de escalonamento dificultava o uso dessas ferramentas. Além disso, a ausência de uma estrutura comum para os algoritmos que surgiam tornava pouco viável a aplicação e a adaptação das ferramentas.

Os anos 1990 revelaram o surgimento de muitos trabalhos na área de escalonamento e planejamento de tarefas para sistemas distribuídos em tempo real. O paralelismo de execuções, a impossibilidade de preempção nas comunicações, bem como a indesejabilidade de preempção na maior parte dos casos – o que levaria a uma sobrecarga dos canais de comunicação – e as relações de precedência e exclusão entre tarefas e comunicações fizeram este um problema difícil de ser resolvido e certamente não houve (nem há ainda) um consenso sobre a melhor ferramenta a ser utilizada. Como em outros problemas de natureza combinatória, discussões sobre quando usar ferramentas de otimização global ou heurísticas/meta-heurísticas não revelam uma conclusão, se não pelo compromisso entre obter de fato o ótimo e a rapidez para fazê-lo.

Fora a questão da ferramenta ou técnica utilizada, a abordagem dos problemas

na época ainda era razoavelmente generalizada. Tentava-se enquadrar diferentes sistemas numa mesma estrutura, sem muita consideração sobre as peculiaridades de cada sistema. Isso se vê principalmente nos objetivos de otimização, que ainda não eram muito apropriados a nenhuma aplicação específica, e na verdade apenas representavam uma folga com relação ao problema de viabilidade. Para sistemas de controle distribuídos, estes critérios são pouco apropriados porque não levam em conta o desempenho do controle ou a qualidade da comunicação acíclica. As tecnologias de redes de campo eram incipientes na época – o FOUNDATION™ Fieldbus, primeiro a permitir controle distribuído, foi lançado em 1994. Além disso, após contato direto com diversos contatos industriais e no fórum da Fieldbus Foundation, chegou-se à conculsao de que o uso de controle concentrado num controlador remoto continuou sendo majoritariamente usado no ambiente FF, e que a migração para o controle distribuído apenas passou a ser significativa nos últimos 5 anos.

A teoria de Sistemas de Controle em Rede começou a ser amplamente explorada e pesquisada também no século atual e, portanto, seus problemas ainda eram pouco conhecidos pela comunidade científica. A revisão da área pode ser vista na seção 2.4.

O problema de planejamento e escalonamento de tarefas para sistemas distribuídos em tempo real continuou sendo amplamente pesquisado no século XXI, com os focos mais bem determinados. Com base na pesquisa dos trabalhos inciais e no estado da arte, as próximas revisões no tema estão dividas da seguinte forma: na seção 2.1.1, é descrita a revisão bibliográfica para escalonamento de tarefas em redes de campo, com atenção particular em FF H1. Na subseção 2.1.2, a revisão é estendida para o estado da arte em planejamento e escalonamento de tarefas para outros tipos de sistemas distribuídos. Por fim, conclusões e um resumo comparativo da literatura revista são feitos na subseção 2.1.3.

2.1.1 Escalonamento de Tarefas em Redes de Campo

No início da utilização do protocolo FOUNDATION™ Fieldbus, alguns poucos trabalhos trataram o tema. Em [7], um algoritmo de escalonamento foi criado para reduzir o tamanho da agenda quando alguns ciclos de transmissão são relativamente primos ou no caso geral em que o MMC entre os períodos é muito grande. Em [17], Franco considera dependências temporais entre os dados e utiliza a metodologia de Gerber para propor um algoritmo de escalonamento para ambos dados periódicos e assíncronos.

Conforme a tecnologia avançou, esses desafios perderam a importância ou se tornaram desatualizados. Com relação ao trabalho de [7], os sistemas de controle comerciais não costumam permitir muitos ciclos diferentes numa mesma rede, espe-

cialmente primos entre si. Na prática, um período de 500 ms não é muito diferente de um de 600 ms no que tange desempenho de controle. Em muitos sistemas [31],[23], existe uma lista de ciclos configuráveis divisíveis entre si. Por exemplo, em FF H1, ciclos de 250, 500, 1000, 2000 e 5000 ms são configuráveis no sistema Centum CS3000, da Yokogawa. No que diz respeito ao trabalho em [17], FF H1 já considera prioridades entre as modalidades de comunicação. Comunicações P/A têm a mais alta prioridade, seguidas por F/P e, por último, C/S. Assim, qualquer outra comunicação que não P/A é imediatamente cancelada se for o momento para uma comunicação P/A. Por isso o nome da comunicação P/A é *Compel Data* (traduzindo, algo como "Forçar Dados") na terminologia FF H1.

Em [25], os autores chegam a um importante resultado para Sistemas de Controle em Rede (NCS, do inglês *Network Control Systems*), o dos Máximos Limites de Atraso Permissíveis (MADB, do inglês *Maximum Allowable Delay Bounds*). Um método de escalonamento para redes de controle em rede em que múltiplas malhas de controle são executadas no controlador foi então proposto, baseado nesse resultado. O algoritmo é apenas aplicável em sistemas com controle centralizado e pode ser adaptado para FF H1, desde que o controle seja configurado dessa forma. Para tal, seria necessária uma adaptação no algoritmo, pois este não considera o readback, que é comum em malhas de controle de processo. Para o presente trabalho, o algoritmo não se aplica, pois o objetivo é fazer um método aplicável a redes com controle distribuído, que é um caso mais complexo estruturalmente.

O tratamento de questões de NCS para o escalonamento passou a ser mais ativo a partir deste século. Uma estratégia de AP para redes *Controller Area Network* (CAN) foi proposta em [43]. A fim de garantir que as restrições temporais sejam atendidas e manter elevada a flexibilidade da rede, Wu *et al* [43] propuseram a atribuição de prioridades baseada no prazo de entrega das mensagens. A estratégia é fortemente baseada em características particulares ao CAN, cujo protocolo prevê prioridades às mensagens. Mesmo podendo ser adaptável a FF H1, os ganhos em flexibilidades são pouco desejáveis uma vez que raramente ocorrem mudanças numa rede FF H1 a partir do momento em que ela passa a operar.

Em [21], métodos heurísticos são propostos para aprimorar o escalonamento de taregas em FF H1, incluindo métricas específicas para este tipo de sistema. Os objetivos propostos por Hodson [21] foram utilizados neste trabalho. O agrupamento de comunicações para aumentar a disponibilidade da rede e os atrasos de controle são levados em conta em seu trabalho e serão também explorados neste. Contudo, como se trata de um artigo branco, não há detalhes do algoritmo, que é protegido. Além disso, não é realizada uma otimização plena, o que é o objetivo deste trabalho. Heurísticas como as propostas em [21] costumam ser muito eficazes para um conjunto de casos conhecidos, mas podem não ser eficientes para casos mais complexos. Com

o avanço da tecnologia, espera-se que mais dispositivos pertençam a uma mesma rede, e assim mais malhas simples e algumas malhas complexas e/ou multivariáveis poderão ser realizadas nos dispositivos de campo, numa mesma rede.

Cicillini [8] propõe um algoritmo offline de escalonamento de tarefas e mensagens para FOUNDATION™ Fieldbus. O algoritmo se baseia em prioridades estáticas, escolhidas pela estrutura das malhas de controle, e pelas restrições de recurso e de precedência. Um processo iterativo aloca as tarefas segundo as prioridades e restrições, objetivando a obtenção de uma agenda viável. Os resultados mostram, no entanto, que algumas restrições de recursos não foram obedecidas, havendo sobreposição de tarefas no mesmo dispositivo. Além disso, o algoritmo não tenta otimizar algum objetivo, em contraponto ao proposto no presente trabalho. Novamente uma abordagem por prioridades é usada para redes de campo em [44]. Neste trabalho, Xiang-Li propõe uma atribuição de prioridades às tarefas em função da profundidade da tarefa no grafo a que ela pertence e no prazo. Novamente, otimização não é considerada.

Poucos trabalhos consideraram escalonamento ótimo em um ambiente de rede de campo. Num trabalho que se aproxima deste [39, 40], Song usa programação com restrições para enumerar todas as possibilidades de alocação de blocos de função em dispositivos e escolhe a configuração que leva ao mínimo número de comunicações e agenda mais curta (mínimo *makespan*). Song faz uma série de considerações para a temporização das tarefas. Em uma delas, assume-se que não há atrasos entre blocos consecutivos numa malha de controle, o que pode ser indesejável ou impossível dependendo do caso (como várias malhas complexas em paralelo). Song afirma que agendas ótimas são obtidas, mas no entanto seus resultados ainda precisam ser verificados através de estudos de caso numéricos.

Outras abordagens para sistemas de controle em rede visam o planejamento dos períodos de execução em vez de o planejamento dos instantes de execução. Essa abordagem é mais aplicável para casos de controle concentrado no controlador. No entanto, a análise e o planejamento dos períodos de execução é inerente ao projeto das configurações de controle. Em [34], Peng *et al* propõem a escolha ótima da periodicidade de cada malha de controle num NCS levando em conta o máximo desempenho das malhas e a qualidade de serviço. Abordagem semelhante é feita em [28], em que Li *et al* observam que minimizar os atrasos de controle e o consumo de banda são critérios conflitantes, e propõem uma otimização multi-objetivo para o problema de gerenciamento dos períodos de execução.

2.1.2 Escalonamento de Tarefas em Sistemas Distribuídos em Tempo Real

Em sistemas distribuídos em tempo real, a maior parte dos problemas abordados são de escalonamento em tempo de execução, em que não necessariamente se conhece uma configuração previamente estruturada de antemão e muitas vezes apenas uma estimativa conservadora dos tempos de execução é conhecida. Para tais problemas, algoritmos de Atribuição de Prioridades são comumente utilizados, como foi feito em [11, 12]. A interdependência de tarefas é considerada explicitamente em [12], transformadas em atrasos no tempo de lançamento. Em [11], o algoritmo de Determinação de Prioridade Ótima é estendido para um sistema em tempo real multiprocessador. A técnica também leva em conta a conversão de interdependências em atrasos nos tempos de lançamento. De uma maneira geral, algoritmos de AP podem ser usados para a criação de tabelas de agendamento offline, como as do problema FF H1, como já foi visto na seção anterior. No entanto, elas dificilmente são capazes de otimizar certos critérios para um caso geral, especialmente havendo interdependências e conflito de recursos, uma vez que não se usam todas as informações pertinentes ao problema de escalonamento. Dito isto, a não ser em casos em que há alguma garantia de otimização um determinado critério, trabalhos concernentes a algoritmos de AP não serão mais descritos nesta dissertação.

Uma abordagem via Redes de Petri, ferramenta para modelagem de sistemas a eventos discretos, foi proposta em [2] para o escalonamento pré-tempo de execução de tarefas em sistemas em tempo real rígidos. O problema de alocação é transformado em uma Rede de Petri que executa internamente um algoritmo de busca em profundidade. Segundo Barreto *et al*, o uso de uma Rede de Petri facilitaria a geração automática de código ao mesmo tempo em que se determina esse escalonamento. Além disso, a mesma Rede de Petri é aplicável à própria simulação do sistema em tempo real. Infelizmente, a técnica se limita a encontrar um escalonamento viável, novamente ficando pouco adaptável a um caso em que se deseja otimizar critérios mais específicos ao problema FF H1.

Em [27], o algoritmo proposto visa a otimizar o escalonamento para tarefas paralelas com dependências, aplicado a sistemas multiprocessador. O algoritmo possivelmente poderia ser adaptado ao problema FF H1, mas o tamanho da agenda não é o objetivo mais importante em um sistema de controle em rede. Além disso, o trabalho trata de processadores homogêneos e a alocação é o principal grau de liberdade para a otimização nesses casos. No caso FF H1, como a maior parte das alocações é predeterminada (ver capítulo 3), o principal grau de liberdade disponível são os tempos de execução.

2.1.3 Conclusões

A partir da revisão da literatura de escalonamento de tarefas para sistemas em tempo real, especialmente os distribuídos, as principais conclusões que podem ser tiradas com relação ao caso FF H1 são:

1. Muitos trabalhos abordam tarefas desconhecidas *a priori*, que são conhecidas em tempo de execução. Isso dificulta a realização de um escalonamento ótimo segundo objetivos especificados. O caso FF H1 permite o conhecimento total de todas as tarefas relacionadas a controle antes do tempo de execução, i.e., relações de precedência, tempo de execução e periodicidade. Dependendo da abordagem, a alocação pode também ser predeterminada. O caso FF H1 é, portanto, mais suscetível a uma análise global visando a otimização do desempenho do sistema.

2. É possível tratar tarefas e comunicações de maneira equivalente, como em [1], assim como tratar um barramento como dispositivo.

3. A observação de Cardeira e Mammeri em [6] é especialmente pertinente para o caso FF H1, em que os tempos de comunicação são comparáveis aos de execução. Ademais, como há de se perceber nos estudos de caso realizados neste trabalho, a rede é o "dispositivo" que mais se mantém ocupado dentro de um ciclo e é, portanto, o gargalo do sistema.

4. Muitos dos trabalhos vistos são para sistemas em tempo real de utilidade genérica, ou seja, sem consideração explícita das tarefas realizadas no sistema. Alguns dos trabalhos aplicados a NCS apenas adaptam técnicas de sistemas em tempo real sem preocupação com o desempenho do controle. Para um NCS como o FF H1, precisa-se considerar problemas relacionados a controle.

5. Notou-se ao longo da literatura revista a ausência de uma descrição matemática completa de um sistema em tempo real distribuído, de forma que essa descrição possa ser aplicável a um problema de otimização. De uma maneira geral, falta tratar o escalonamento de sistemas em tempo real como um problema de otimização de fato. Alguns trabalhos propõem algoritmos específicos de Ramificação e Poda ou Meta-heurísticas, mas que se tornam pouco adaptáveis a outros casos.

6. Os critérios que podem ser considerados na otimização para FF H1 incluem minimização do consumo de banda [28, 39, 40], minimização do período de execução das malhas [34], minimização dos atrasos de controle [21, 26, 28], agrupamento das comunicações [21] e tempo final de execuções [11, 27].

7. Devem-se considerar as relações de precedência e exclusão entre tarefas. O prazo de cada tarefa está relacionado à sua periodicidade.

8. Talvez a principal conclusão que se pode retirar de toda a revisão feita na literatura de sistemas distribuídos em tempo real é que, dependendo das características do sistema, não é necessário tratá-lo como um sistema em tempo real. No caso FF H1 em particular, com o completo conhecimento prévio das tarefas a serem executadas bem como suas periodicidades, é possível considerar o problema como um problema de escalonamento num sistema distribuído qualquer. Portanto, forçosa é a necessidade de se estender a revisão bibliográfica para outros tipos de sistema distribuído, em especial os que consideram otimização. É o que será feito na próxima seção deste capítulo.

2.2 Escalonamento Ótimo de Tarefas em Sistemas Distribuídos

Como foi observado na seção anterior, o problema FF H1 pode ser encarado como um problema de escalonamento geral, sem se preocupar com as características de tempo real do sistema. Isso acontece porque o tratamento das mensagens pelo Escalonador Ativo de Enlaces (LAS, do inglês *Link Active Scheduler*) é definido no protocolo e, para o gerenciamento das comunicações periódicas, o mesmo LAS recebe uma tabela de agendamento feita antes do tempo de execução. Esta tabela é válida para um macrociclo e a repetição desta tabela ciclicamente garante o tempo real e a periodicidade das tarefas. Para um macrociclo, o escalonamento é semelhante ao de qualquer sistema distribuído – mudam algumas restrições e os objetivos. Portanto, torna-se necessário rever um pouco da literatura para outros tipos de sistemas distribuídos, não necessariamente em tempo real. Como o objetivo deste trabalho é de otimizar o escalonamento, esta revisão limita-se aos trabalhos que abordam escalonamento ótimo.

Nesta área, dois campos de aplicação que se destacam são o de processos em batelada e o de sistemas multiprocessados. O sistema FF H1 é altamente heterogêneo como são processos em batelada, i.e., diversas tarefas têm alocação limitada a um determinado equipamento. Por outro lado, a preocupação com tempo é mais comum para sistemas de computador, com o lucro sendo o objetivo principal em processos. Independente das características individuais de cada aplicação, a caracterização matemática do problema e as técnicas aplicadas muitas vezes encontram semelhanças entre as diferentes áreas, e consequentemente podem ser avaliadas para adaptação ao caso FF H1.

Para se otimizar o escalonamento de fato, ou seja, encontrar garantidamente

uma solução ótima, uma técnica amplamente utilizada em ambas as áreas é a de Programação Inteira Mista. Trata-se de uma área da Programação Matemática em que as variáveis podem ser tanto contínuas como inteiras. Uma breve introdução deste assunto encontra-se na seção 2.3. Com isto, descrever-se-ão nesta dissertação as referências de escalonamento ótimo que usam esta técnica. Na subseção 2.2.1, a revisão é feita no campo de Processos em Batelada. Na subseção 2.2.2, estudam-se os casos de aplicação a Sistemas Multiprocessados. Na subseção 2.2.3, encontra-se um resumo das técnicas aplicáveis ou adaptáveis baseado na literatura revista em ambos os campos supracitados.

2.2.1 Escalonamento de Tarefas para Processos em Batelada

Como se trata de um problema multidisciplinar, planejamento e escalonamento de tarefas não é apenas abordado na Engenharia Elétrica ou na Ciência da Computação. Na Engenharia Química, o problema é amplamente estudado e é de grande importância. Segundo Mendez *et al* [32], a Programação da Produção é crítica para processos em batelada, já que ela é crucial para aumentar o desempenho da produção. Ainda de acordo com os autores, nos últimos 20 anos houve um progresso significativo na área de escalonamento de curto prazo para processos em batelada, incluindo a solução de problemas de tamanho industrial.

Para o escalonamento de curto prazo para processos em batelada, modelos são criados frequentemente através de Programação Linear Inteira Mista (MILP). Revisões do assunto para futura leitura podem ser encontradas em [13, 14, 32]. Uma breve visão geral está presente em [19].

Há uma forte correlação entre o problema de escalonamento para a rede FF H1 e para processos em batelada. Dispositivos de campo podem ser tratados como unidades que executam tarefas especificadas e produzem dados para outras tarefas. Com isso, a abordagem através de MILP é muito adequada para isto e os modelos encontrados na literatura da Engenharia Química podem ser adaptados para o caso FF H1.

Dentre as associações entre o caso FF H1 e processos em batelada, destacam-se:

1. Dispositivos de campo e a rede são equivalentes aos equipamentos da indústria química. São os executores de tarefas, sendo que algumas várias tarefas estão associadas a equipamentos/dispositivos específicos. Da mesma forma que uma mistura de produtos deve ser executada no misturador que a recebe, a leitura de uma variável de temperatura só pode ser feita pelo transmissor de temperatura associado.

2. Tarefas nos dois casos têm um tempo para serem executadas (esse tempo pode ser variável na Engenharia Química) e dependem de certas condições

para poderem ser ativadas. Estas condições em geral estão relacionadas às execuções de tarefas precedentes, direta ou indiretamente.

3. Relações de exclusão também existem por dispositivo ou equipamento. Um dispositivo/equipamento não pode executar duas tarefas distintas ao mesmo tempo.

4. Um macrociclo para o caso FF H1 equivale a uma batelada. É o período em que todas as tarefas serão executadas para um dado fim. Por mais que o macrociclo se repita, isso não faz diferença enquanto se faz o escalonamento das tarefas dentro de um macrociclo.

Existem duas principais abordagens no que tange à representação temporal para a programação da produção de processos em batelada: tempo discreto e tempo contínuo. Na representação por tempo discreto, o tempo é dividido em intervalos de duração fixa e as execuções das tarefas são alocadas nestes intervalos. Na representação por tempo contínuo, os tempos de início e fim de execução de cada tarefa são variáveis contínuas. Como será visto adiante, os objetivos tratados neste trabalho envolvem os instantes de execução diretamente. Para processos em batelada, as abordagens de tempo discreto ou contínuo competem em tempo para resolver e acurácia dos resultados. No caso do problema para FF H1, a abordagem por tempo contínuo é mais apropriada, pois permite tratar os objetivos de maneira direta, sem adaptações.

Uma modelagem eficiente usando a abordagem de tempo contínuo foi criada por Ierapetritou e Floudas em [22]. Neste artigo, os autores atingem bons resultados em escala e tempo para resolver. O modelo envolve o conceito de pontos de evento específicos para cada unidade. Trata-se de pontos na linha do tempo para um número pré-fixado de eventos que podem acontecer, onde eventos incluem o início e o fim de execuções. É importante notar que os instantes dos eventos não são predeterminados, mas variáveis, com a restrição de que um evento sempre ocorre antes que um evento subsequente, desde que referentes ao mesmo equipamento – não existe associação de eventos para equipamentos distintos. Variáveis binárias indicam o começo e o término de cada execução de uma tarefa num dado ponto de evento. Segundo Ierapetritou e Floudas, a contribuição que permitiu os bons resultados foi a desassociação dos eventos de tarefas dos eventos de unidades, introduzindo restrições de sequenciamento temporal e garantindo linearidade.

Dentre esses trabalhos mais recentes, destaca-se [38], onde uma abordagem de modelagem unificada é proposta para escalonamento de curto prazo para processos em batelada. Para tal, mesclam-se problemas envolvendo ou não restrições de recursos, criando uma estrutura genérica comum. Alguns avanços com relação ao trabalho

seminal de Ierapetritou e Floudas encontram-se neste artigo, melhorando o desempenho da otimização. Além disso, não existem restrições de recursos (no sentido de matéria-prima) para o caso FF H1, senão por meio das relações de precedência.

Por fim, nota-se que existe uma grande correlação entre os temas. Na verdade, pode-se encarar o caso FF H1 como um caso particular de processo em batelada, onde algumas simplificações existem:

- Os tempos de todas as tarefas são fixos.
- O número de execuções de cada tarefa é fixo e conhecido. Em processos em batelada, podem haver múltiplas campanhas que cheguem a um resultado ótimo, e seu número é desconhecido de antemão.
- Com essas duas características, o número de pontos de evento no caso FF H1 pode ser predeterminado com exatidão, o que não acontece em processos em batelada, onde estimativas têm que ser feitas.
- Não há diferenciação entre materiais nem quantidades. Em processos, há requisitos de quantidade de certos materiais, chamados de estados. No caso FF H1, a relação é meramente de existência ou não dos dados da tarefa precedente.
- Da mesma forma, as capacidades dos equipamentos pode ser desconsiderada.
- Custos e receitas podem ser desconsiderados.

Com essas simplificações, restam a atribuição de tarefas a pontos de evento e a determinação dos instantes de execução de cada tarefa. A atribuição a pontos de evento de certa forma se resume a um problema de ordenação se a alocação de tarefas for conhecida, como se verá no capítulo 3. Por outro lado, num caso em que múltiplos ciclos diferentes coexistem num mesmo macrociclo, o caso FF H1 ganha uma característica de periodicidade em que tarefas de ciclos distintos são realizadas num mesmo dispositivo. Essa característica não foi encontrada nos trabalhos aqui estudados.

2.2.2 Planejamento e Escalonamento Ótimos de Tarefas em Ambientes Multiprocessados

Planejamento e escalonamento de tarefas para ambientes multiprocessados heterogêneos é parte integrante do co-projeto de software e hardware de sistemas embarcados [3, 10]. De acordo com Davare *et al* [10], técnicas com a Programação Linear Inteira Mista têm a vantagem de serem adaptáveis a problemas de alocação e escalonamento com peculiaridades. Além disso, segundo Grajcar e Grass, o uso de MILP garante

a otimalidade do resultado e, mesmo em casos em que ótimo global do problema não é necessário, MILPs proporcionam um limite inferior para o ótimo (em caso de minimização – em caso de maximização, trata-se de um limite superior), o que meta-heurísitcas não proporcionam. Nesta área, o problema de alocação e escalonamento visa a determinar os Elementos de Processamento (EPs) que vão executar cada tarefa e a temporização das tarefas a fim de alcançar o menor tempo de execução (*makespan*, no inglês) e reduzir custos de processamento e comunicação. Muitas vezes consideram-se EPs heterogêneos, ou seja, com tempos de execução distintos entre si para a mesma tarefa ou com capacidade de executar apenas algumas tarefas específicas. Ao mesmo tempo, relações de precedência, que são fundamentais para esta dissertação, são consideradas em alguns trabalhos. De acordo com Davare *et al*, não existem boas aproximação que resolvam esse problema em tempo polinomial.

Assim como para o problema de processos em batelada, existem representações temporais discretas e contínuas para o problema de sistemas multiprocessados. Pelo motivo já comentado na subseção acima, nesta revisão apenas a representação por tempo contínuo será abordada. Davare *et al* [10] propõem uma taxonomia para as diferentes formulações usadas neste tipo de problema. As três formulações são a por sequenciamento, por *slots* e por sobreposição. Davare *et al* comentam que a formulação por sobreposição alcança melhores resultados ao mesmo tempo em que têm menos variáveis e restrições, mostrando resultados que comprovam isso para o problema abordado por eles. Uma vez que uma grande adaptação deveria ser feita para o caso FF H1, é importante adaptar as diferentes abordagens e comparar os resultados para tomar a decisão sobre as mesmas.

Uma formulação usando a representação por sequenciamento foi proposta por Bender em [3]. Em tal representação, para cada par de tarefas i, j e EP m existem uma variável binária $y_{m,i,j}$ que indica que, se ambas as tarefas são executadas em m, então i termina antes de j começar para $y_{m,i,j} = 1$ e j termina antes de i começar se $y_{m,i,j} = 0$. Dadas as restrições do modelo, a variável y perde o significado caso i e j sejam alocados em EPs distintos, podendo assumir qualquer valor sem influenciar o problema. Esse conceito tem uma diferença sutil com relação à representação por sobreposição, encontrada em [33]. Nesta representação, a variável $z_{i,j}$ é 0 se i termina depois de j começar. Quando $z_{i,j} = z_{j,i} = 0$, existe sobreposição entre as tarefas. O cerne da formulação em [33] é resumido em [10].

Uma representação conceitualmente distinta das duas descritas acima encontra-se em [30]. Em seu trabalho, Maculan *et al* propõem um conjunto ordenado de intervalos de tempo para cada EP. Diferente da formulação por tempo discreto, esses intervalos de tempo têm duração e tempos inicial e final desconhecidos *a priori*. Sua função é de indicação de ordem absoluta. Ou seja, se uma tarefa é escalonada no n-ésimo intervalo de tempo de um EP, ela será a n-ésima tarefa executada por ele.

Essa representação tem a vantagem de informar a posição absoluta de cada tarefa, mas possui a desvantagem de não ser possível saber de antemão o número de tarefas que cada EP executa e, portanto, torna-se necessária uma estimativa possivelmente conservadora para este número. Essa formulação tem particularmente um diálogo interessante com a formulação por pontos de evento para Processos em Batelada, pois nesta última os pontos de evento também indicam a ordem absoluta das tarefas numa determinada unidade.

Novamente, pode-se ver que o problema FF H1 é um caso particular do problema para múltiplos processadores. Neste caso, o problema é mais próximo do FF H1, se comparado ao de Processos em Batelada, mas já foi possível notar certas correlações entre o problema na Engenharia Química e o problema na Ciência da Computação. Destas correlações, destacam-se a heterogeneidade de unidades/EPs, as relações de precedência e exclusão (numa mesma unidade/EP) e as diferentes representações temporais, que em nenhum dos casos há um consenso sobre qual é a melhor.

Das simplificações do problema multiprocessador para o problema FF H1, destacam-se:

- A alocação de tarefas é predeterminada.
- Em razão disso, o número de execuções de tarefas em cada dispositivo ou na rede pode ser predeterminado com exatidão.
- Cada comunicação é conhecida de antemão.

Uma das principais diferenças entre esses problemas é novamente o objetivo da otimização. O *makespan* não seria o principal objetivo em FF H1, ao passo que costuma ser o único objetivo em multiprocessadores quando o sistema já é dado, ou pelo menos um dos objetivos quando trata-se do co-projeto do sistema – neste último caso, o custos de hardware e de comunicação devem ser levados em conta.

2.2.3 Observações Finais

Com a revisão realizada na área de planejamento e escalonamento ótimo de tarefas em sistemas distribuídos, concluiu-se que a ferramenta de Programação Linear Inteira Mista é muito apropriada para o objetivo deste trabalho. Além disso, há formulações adaptáveis ao caso FF H1 em diversas áreas da Engenharia, com destaque para Sistemas Multiprocessados. Na Engenharia Química, o problema de planejamento de curto-prazo da produção para Processos em Batelada também apresenta forte interação com o problema FF H1.

Como será discutido adiante, a alocação de tarefas a dispositivos pode ser considerada predeterminada no caso FF H1, o que simplifica o problema para este

trabalho. Talvez a principal mudança entre os trabalhos encontrados e este está nos objetivos. No caso FF H1, existem diversas malhas de controle compartilhando o mesmo barramento e há a preocupação de diminuírem-se os atrasos em cada uma das malhas. Ao mesmo tempo, como a rede tem outros propósitos além do controle, deve se haver uma preocupação com a Qualidade de Serviço (QoS, do inglês *Quality of Service*) dessa rede, garantindo banda para outras comunicações. Felizmente, a formulação por MILP também tem a vantagem de ser facilmente adaptável a problemas particulares – as restrições que definem o modelo podem ser as mesmas, mudando-se apenas a função objetivo.

O caso geral FF H1 considera múltiplos ciclos de execução numa mesma rede e isso dá uma nova particularidade do problema FF H1 frente aos demais casos. Nesse caso, a adaptação do problema se torna mais complexa, exigindo uma nova série de propostas de modelagem.

Por fim, uma vez que a abordagem via MILP foi identificada com sendo a mais propícia para este trabalho, é importante uma breve introdução ao assunto, que será feita a seguir.

2.3 Programação Linear Inteira Mista

Programação Linear Inteira Mista, ou MILP, é uma das sub-áreas de Programação Matemática. Linear porque considera equações lineares como restrições e função objetivo. Inteira Mista porque trata com variáveis tanto contínuas quanto inteiras. Ela é usada tipicamente em problemas NP-difíceis ou NP-completos, ou seja, problemas que não podem ser resolvidos em tempo polinomial com relação ao número de variáveis. Problemas combinatórios são exemplos de problemas NP-completos e para poucas variáveis inteiras (por exemplo, 30 variáveis binárias, o que daria cerca de 1.000.000.000 de possibilidades) a enumeração de todas as soluções já é impraticável.

Um Problema Linear Inteiro Misto pode ser enunciado da seguinte forma:

$$\begin{aligned} & \min_{x,y} c_1^T x + c_2^T y \\ \text{sujeito a} \quad & A_1 x \leq b_1, \\ & A_2 y \leq b_2, \\ & x \geq 0, \\ & y \geq 0, \\ \text{tal que} \quad & x \in \mathbb{R}^n, y \in \mathbb{Z}^m \end{aligned} \tag{2.1}$$

A representação de problemas através de MILPs ocorre pela natureza das variáveis de decisão. Variáveis inteiras normalmente representam escolhas dentro de um conjunto finito de elementos. Variáveis contínuas representam quantidades que podem variar dentro de uma determinada faixa. Por exemplo, a escolha de elementos dentro de um conjunto ou a ordem de execução de tarefas podem ser representados por variáveis inteiras. A corrente de um motor, a vazão de um líquido ou o instante de execução de uma tarefa podem ser representados por variáveis contínuas.

Por ser uma área ampla e consolidada, não será aqui feita uma revisão do tema. O leitor pode referir-se a livros como [42], [37] e [24]. Problemas inteiros tipicamente não podem ser resolvidos em tempo polinomial, mas várias técnicas conseguem encontrar garantidamente a solução ótima sem ter que resolver o problema contínuo resultante da enumeração de cada possibilidade de solução dos inteiros. Dentre essas técnicas, destacam-se:

- Método dos Planos de Corte, que realiza sucessivos cortes no hiperpoliedro correspondente ao problema relaxado, i.e., cujas variáveis inteiras são tratadas como reais, até que a solução ótima do problema cortado seja inteira. Os cortes são inequações que não estão explícitas no modelo original, mas que não eliminam qualquer solução inteira.

- *Branch & Bound*, técnica que enumera sistematicamente as soluções candidatas numa estrutura em árvore, eliminando grandes conjuntos de soluções subótimas ao usar estimativas de limites inferior e superior da função objetivo.

- *Branch & Cut*, híbrido das duas técnicas acima. Realiza cortes a cada etapa do *Branch & Bound*, criando cortes globais (i.e., válidos para todas as soluções inteiras viáveis) e locais (válidos para um subproblema do *Branch & Bound* em particular).

Não é desenvolvido aqui qualquer algoritmo para resolver MILPs, sendo utilizados *solvers* para este fim. Tais *solvers* combinam as técnicas apresentadas acima com *solvers* de problemas lineares contínuos, como o Simplex, heurísticas para encontrar boas soluções inteiras com rapidez e algoritmos para determinar direções de busca ou maneiras de *branching* eficazes.

2.4 Sistemas de Controle em Rede

No contexto de controle, um sistema controlado com instrumentos interligados através de uma rede de campo compõe um Sistemas de Controle em Rede. NCSs são sistemas espacialmente distribuídos em que a comunicação entre sensores, atuadores

e controladores ocorre através de uma rede de comunicação digital compartilhada e de banda limitada [20].

Numa recente revisão sobre o tema, Gupta e Chow [18] propõem a divisão dos desafios impostos por NCSs em dois tipos: Qualidade de Controle (QoC, do inglês *Quality of Control*) e Qualidade de Serviço (QoS). QoC refere-se a reduzir a degradação no desempenho que imperfeições e restrições dos canais de comunicação podem impor ao controle da planta. QoS, por sua vez, está relacionado a taxas de transmissão, taxas de erro e outras características que podem ser medidas, melhoradas e, até um certo ponto, garantidas.

Um dos tópicos recentes de pesquisa em NCS avaliados por Gupta e Chow, alocação de banda e escalonamento de tarefas é motivado pelo objetivo de se reduzirem custos no dimensionamento das redes garantindo o fluxo de dados necessário ao controle a às demais funções da rede. Dentre os trabalhos citados nesta revisão, a maior parte propõe heurísticas em tempo de execução, como as já revisadas na Seção 2.1.

Em outra revisão da área, voltada para o problema de desenvolvimento de estratégias de controle, Hespanha *et al* [20] classificam os problemas ao sistema de controle causados pela introdução de uma rede:

- Canais de banda limitada, que aborda os problemas do limite de informação que a rede impões. Isso leva a uma avaliação de importância relativa entre o tempo de amostragem e a taxa de quantização.

- Amostragem e atraso, que se refere à degradação do controle devido aos intervalos entre a passagem sucessiva de informações para realimentação – tempo de amostragem ou tempo de ciclo – e ao tempo que essa informação leva para ser processada e transmitida ao atuador – atraso.

- Perda de pacotes, que diz respeito a eventuais perdas de informação durante a trasmissão de dados de realimentação da malha de controle – e os efeitos que isso pode causar.

- Arquitetura de sistemas, que está relacionado aos protocolos de comunicação, buscando maior eficiência para a aplicação destino – o controle.

Dentre esses temas, para FOUNDATION™ Fieldbus e protocolos de comunicação semelhantes, apenas "Amostragem e atraso" é significativo. A quantização típica nesses protocolos é de 16 bits [16], que gera um erro de cerca de 0,0015%. Perda de pacotes costuma ser mais relevante em redes sem fio e acontece com mais frequência em redes com fio quando há problemas na instalação e deficiência na blindagem ao ruído. A mudança da arquitetura do sistema não é o objetivo desta dissertação.

Os atrasos induzidos por rede, por sua vez, são pertinentes a redes como FF H1 e vem sendo amplamente estudados [25], [5]. De acordo com Kim *et al* [25], tais atrasos, provenientes dos tempos de transmissão e de *overhead*, podem degradar o desempenho do controle e chegar a desestabilizar o sistema controlado.

Branicky *et al* [5] analisam explicitamente os efeitos combinados dos atrasos e do tempo de ciclo de malhas de controle para NCSs, determinando critérios para o sistema realimentado ser estável. O critério pode ser aplicado encontrando-se regiões de estabilidade no plano atraso x tempo de ciclo. De uma maneira geral, o resultado comprova que atrasos e tempos de ciclo pequenos o suficiente não interferem na estabilidade e que, quanto maior o tempo de ciclo, menor pode ser o atraso induzido pela rede.

2.4.1 Conclusões

A literatura de NCS é algo recente no que diz respeito aos efeitos negativos que a aplicação de uma rede a sistemas de controle pode causar. Dentre os temas mais estudados na área, e o que merece especial atenção por ser relevante para protocolos como FOUNDATION™ Fieldbus, a consideração dos atrasos induzidos pela rede se destaca. É um desafio diretamente relacionado à Qualidade de Controle e que pode ser mitigado com estratégias de escalonamento eficientes, que busquem minimizar esses atrasos. Além disso, é fundamental não sacrificar a Qualidade de Serviço e permitir ao usuário final o uso das funcionalidades que uma rede de campo proporciona com o máximo de rapidez e sem interrupções. Obviamente, esses dois objetivos devem ser buscados sem se sacrificar os aspectos econômicos e, portanto, obter recursos de rede em excesso não é uma solução considerada. Portanto, neste trabalho, propõem-se estratégias de escalonamento de tarefas para atingir bons níveis de QoC e QoS aproveitando os recursos disponíveis. Ao fim, espera-se que, com os resultados aqui alcançados, seja conluído que é possível diminuir o número de recursos de rede sem comprometer a qualidade do sistema através de um escalonamento de tarefas eficiente.

2.5 Considerações Finais

A revisão da literatura nos temas de interesse foi fundamental para situar o problema de escalonamento FF H1 dentro de uma área maior: a de pesquisa operacional. Várias técnicas foram revisadas e, para os propósitos deste trabalho, o método de Otimização Inteira Mista foi escolhido. Tal método tem a vantagem de ser possível representar o problema com fidedignidade, além de que a existência de *solvers* amplamente testados é um fator que contribui para testar e aplicar a técnica.

O estudo de trabalhos na área de NCS contribui para a compreensão dos problemas que o uso de uma rede podem causar ao controle e quais critérios devem ser levados em conta no escalonamento.

Dentro de MILP, observou-se que há diferentes maneiras de modelar o problema, que envolvem a representação temporal e a representação da ordenação das tarefas. Torna-se necessário testar diferentes formulações para encontrar a mais apropriada ou a de melhor desempenho.

Capítulo 3

Escalonamento Ótimo de Tarefas para FOUNDATION™ Fieldbus - Caso Monociclo

Neste capítulo define-se o problema de escalonamento de tarefas para FOUNDATION™ Fieldbus no caso monociclo, seguido da definição dos modelo matemáticos. O caso monociclo é o caso em que todos os dispositivos da rede que executam funções relacionadas a controle possuem o mesmo ciclo de execução para suas tarefas, bem como para as mensagens que enviam. Trata-se de um caso mais simples, mas que tem grande aplicação, visto que diversos sistemas não permitem múltiplos ciclos na mesma rede.

Na seção a seguir (Seção 3.1), há uma descrição textual do problema de escalonamento ótimo para a rede FF H1. A Seção 3.2 apresenta o Modelo Linear Inteiro Misto (MILM, do inglês *Mixed Integer Linear Model*) usando uma formulação baseada numa mistura de intervalos de tempo com pontos de evento – que convergem para uma formulação comum no caso FF. Na seção 3.3, um MILM baseado na formulação por sobreposição – que se confunde com a formulação por sequenciamento. Os critérios de otimização, comuns às duas formulações, são apresentados, justificados e explicados na Seção 3.4. Por fim, alguns ajustes e cortes extras no modelo são explicados na Seção 3.5.

3.1 Descrição do Problema

O escalonamento de tarefas já foi explicado na Seção A.3. Em função das malhas de controle e das características dos dispositivos da rede, o algoritmo de escalonamento deve gerar uma tabela de escalonamento contendo os instantes de execução de cada tarefa. Todas as malhas de controle devem ser executadas com tempo total inferior

ao Macrociclo. Além disso, as regras de ocupação de banda em [15], já comentadas na Seção A.2.2, devem ser satisfeitas. Como critérios de desempenho desse escalonamento, deseja-se minimizar a separação entre as comunicações, o que melhora a qualidade da comunicação acíclica na rede, ao mesmo tempo que os atrasos de controle induzidos pela rede sejam minimizados.

Já foi comentado que alocação é um fator chave em termos de escalonamento. Em termos de otimização do Macrociclo, os blocos de função deveriam ser alocados de forma que o mínimo número de comunicações seja requerido. Neste trabalho, a alocação de blocos de função a dispositivos é realizada *a priori*, já que disponibilidade do sistema e segurança são objetivos mais relevantes para a alocação de tarefas. Por exemplo, numa configuração de Proporcional-Integral-Derivativo (PID) com transmissor redundante, o bloco Seletor de Sinal que se segue ao Entrada Analógica (AI, do inglês *Analog Input*) não deve ser alocado em um dos transmissores, pois isso acarretaria na falha da redundância. De qualquer forma, a alocação ótima considerando disponibilidade da rede e desempenho de controle será abordada num estudo futuro.

Considera-se que uma tarefa nesse trabalho é um bloco de função ou um comunicação publicante/assinante (CD, do inglês *Compel Data*), aproveitando a semelhança de ambos em termos de escalonamento já observada em [6]. Essa semelhança se torna equivalência quando a alocação de tarefas a dispositivos for conhecida *a priori*. Um CD passa a ser uma tarefa do dispositivo rede, com relações de precedência bem definidas.

Isso pode ser observado quando se faz uma transformação dos diagramas de blocos usualmente existentes nos ambientes de configuração de SDCDs em diagramas que explicitam cada comunicação. Um exemplo de um diagrama de blocos de função simplificado pode ser visto na Figura 3.1. Neste trabalho, uma representação modificada da malha de controle FF, doravante chamada de Diagrama Dispositivo-Função-Comunicação (DDFC), é usada para explicitar as comunicações. A representação por DDFC identifica os blocos de função alocados e identifica unicamente cada comunicação. Os blocos de função são representados por retângulos, enquanto os CDs o são por círculos, enquanto os polígonos de linha tracejada que envolvem os blocos de função representam os dispositivos que os executam. Um DDFC geral é encontrado na mesma Figura 3.1, representando a malha da mesma figura. Um CD identificado unicamente é incluído entre dois dispositivos sempre que há comunicação entre eles. Naturalmente, não há CDs quando a comunicação é realizada entre duas funções no mesmo dispositivo.

Com todas essas considerações, o problema de escalonamento para uma rede FF H1 aqui abordado é sumarizado a seguir: dados

- Uma lista de dispositivos na rede;

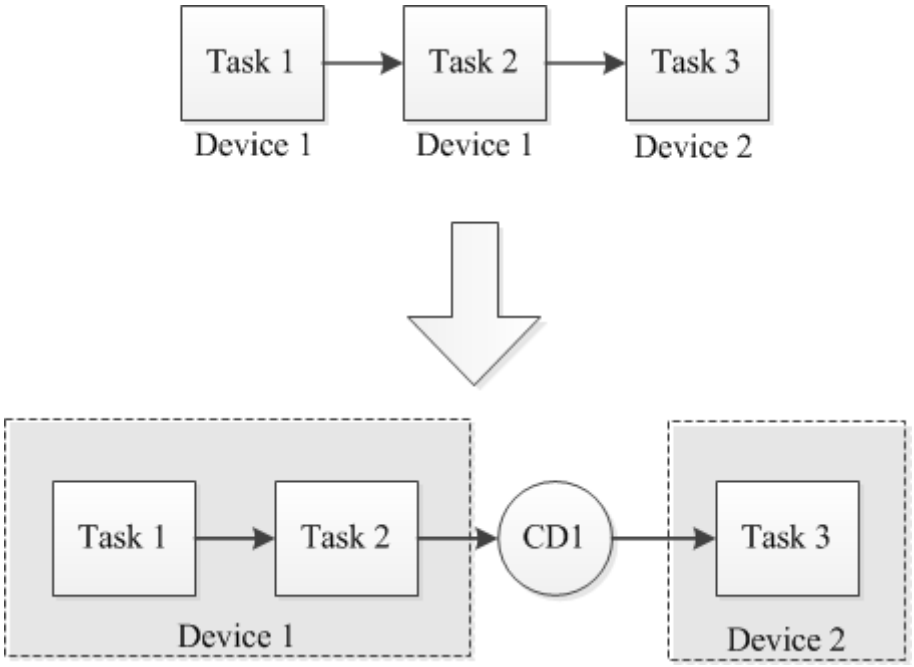

Figura 3.1: Um diagrama de bloco de função simplificado (acima) e o Diagrama Dispositivo-Função-Comunicação correspondente (abaixo).

- As malhas de controle configuradas tais que pelo menos uma função seja executada em algum dispositivo da rede [1];
- O tempo de execução de cada tarefa;
- O tempo de execução configurado para o Compel Data;
- A duração de macrociclo requerida;

o algoritmo de escalonamento deve determinar os instantes de execução de cada função e CD considerando que cada dispositivo (incluindo a rede) só pode realizar uma tarefa de cada vez. O próximo passo é descrever o problema matematicamente.

3.2 Modelo MILP com Representação Temporal por Pontos de Evento

Um modelo para o problema de escalonamento FF H1 monociclo é descrito nesta seção. Este modelo consiste de uma série de restrições de igualdade e desigualdade que descrevem o problema de escalonamento por completo. Uma vez que o objetivo é usar o modelo para resolver o problema de escalonamento, o modelo também

[1] Em [15] recomenda-se que malhas de controle tenham todos os seus blocos de função executados em dispositivos pertencentes à mesma rede.

precisa ser computacionalmente eficiente. Este modelo é baseado nos trabalhos em [22, 30, 38], usando a representação por pontos de evento.

Para este modelo, comunicações de *readback* são consideradas abertas na malha de controle, e não como ciclos fechados. Isso significa que não há verdadeiros reciclos no macrociclo. Uma vez que a agenda se repete a cada macrociclo, a tarefa que precisa do *readback* a receberá no macrociclo seguinte. Considerando isso, há apenas ligações diretas com início e fim para considerar e o diagrama passa a ser representado por um grafo direcionado acíclico. No entanto, o *readback* tem duas possibilidades num macrociclo: ou ele é agendado antes da tarefa que o recebe ou depois da tarefa que o envia. Portanto, surge a necessidade da determinação do ponto de abertura do reciclo durante a otimização. A figura 3.2 exemplifica este problema.

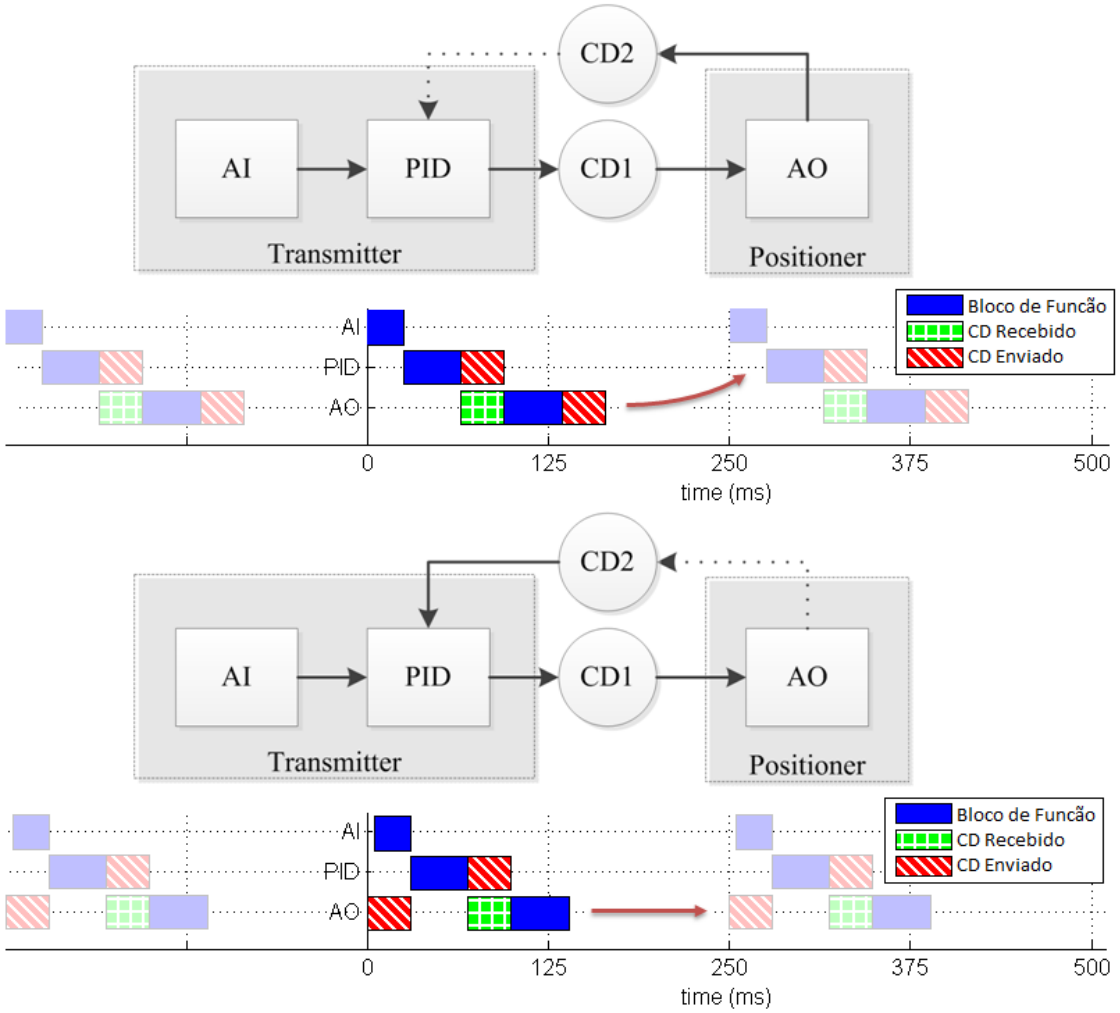

Figura 3.2: DDFC de um PID no transmissor e um escalonamento correspondente com *readback* no fim da agenda (acima), em contraste com DDFC e escalonamento correspondente com *readback* no início da agenda (abaixo).

A descrição rigorosa de um modelo requer inicialmente a definição dos conjuntos, que representam uma classe de elementos. Além disso, alguns parâmetros precisam ser definidos. A partir dessas definições, as restrições que descrevem o problema são formuladas e explicadas. É importante que o modelo seja geral, ou seja, só seja preciso escolher os elementos dos conjuntos e os parâmetros para aplicá-lo a um problema. Isso facilita a aplicação e garante que não há casos não considerados que a formulação não atende. Por outro lado, para que a generalidade não impactar no desempenho, restrições específicas para o caso FF são adicionadas. Para a aplicação em casos semelhantes, elas podem ser desconsideradas.

Antes de enunciar o modelo, algumas definições precisam ser feitas. Blocos de função e mensagens CD são tratadas da mesma maneira para o escalonamento. Um dispositivo é qualquer unidade física capaz de executar tarefas, incluindo transmissores, atuadores e a rede. Uma tarefa é qualquer execução de bloco de função ou mensagem CD. Uma malha de controle, ou simplesmente malha, é referida aqui como um conjunto de blocos de função e comunicações interconectados, formando um grafo acíclico direcionado, com a exceção dos *readbacks*, cuja ordem de execução com relação às tarefas enviante e recebente em um macrociclo não é conhecida de antemão. Um ponto de evento é um indicador do momento em que uma determinada tarefa começa a ser executada num dispositivo. Um conjunto de pontos de evento é um conjunto ordenado: por exemplo, se a tarefa A é executada no terceiro ponto de evento e a tarefa B o é no quarto, isso significa que A é a terceira tarefa a ser executada no dispositivo, enquanto B é a quarta.

Definições de conjuntos, elementos de conjuntos, parâmetros e variáveis encontram-se na seção de Nomenclatura no início deste documento.

O modelo é enunciado abaixo. Para o código na linguagem *GNU Mathematical Programming Language* (GMPL), refera-se à Seção B.1.1.

3.2.1 Conjuntos Pré-Determinados

$N_j = 1, 2, ..., |I_j|, j \in J$, onde N_j é o conjunto de pontos de evento do dispositivo j, I_j é o conjunto de tarefas do mesmo dispositivo j e $|.|$ é o operador cardinalidade. Há tantos pontos de evento para um dispositivo quanto há tarefas para ele executar. Isso ocorre porque cada tarefa é executada uma e apenas uma vez no ciclo, o que permite conhecer de antemão o número de execuções de cada dispositivo.

3.2.2 Definições

$L^1_{j,m}$ representa o conjunto de pares de tarefas que possuem entre si uma relação de precedência de primeira ordem. Precedência de primeira ordem num dispositivo j para uma malha de controle m é definida abaixo:

Definição 3.1 *Uma tarefa i tem precedência de primeira ordem com relação a uma tarefa i' em um dispositivo j numa malha de controle m se e somente se:*

- $i, i' \in I_j$,
- $i \in \pi_1(L_m)$,
- $i' \in \pi_2(L_m)$,
- *exista pelo menos um caminho direcionado da tarefa i à tarefa i',*
- *nenhum caminho direcionado da tarefa i à tarefa i' contenha uma terceira tarefa que pertença a I_j,*

onde π_k é a projeção canônica de um produto cartesiano no seu k-ésimo conjunto.

3.2.3 Restrições

Nesta subseção são definidas e explicadas as restrições que descrevem o problema de escalonamento.

Limites das Variáveis

Esses limites triviais garantem que os tempos inicial e final de cada tarefa esteja dentro do horizonte máximo H do macrociclo.

$$0 \leq Ts_i, Tf_i \leq H, \forall i \in I. \tag{3.1}$$

Restrições de Uso

Esta inequação significa que, para um dado intervalo de tempo n, um dispositivo pode executar no máximo uma única tarefa [22].

$$\sum_{i \in I_j} w_{i,j,n} \leq 1, \forall j \in J, n \in N_j.$$

Como o número de intervalos de tempo é conhecido com exatidão – e há exatamente um intervalo de tempo para cada tarefa – e cada tarefa terá que ser executada uma vez no ciclo, essa restrição pode ser transformada na seguinte:

$$\sum_{i \in I_j} w_{i,j,n} = 1, \forall j \in J, n \in N_j. \tag{3.2}$$

Restrições de Execução

Cada tarefa deve ser executada uma e apenas uma vez durante o macrociclo [30].

$$\sum_{n \in N_j} w_{i,j,n} = 1, \forall j \in J, i \in I_j. \tag{3.3}$$

Restrições de Sequenciamento

Para um dispositivo j e para tarefas que ocorram em intervalos de tempo consecutivos, o tempo final da próxima tarefa i' deve ser maior ou igual que o tempo final da tarefa atual i [22]. Essa inequação linear representa isso porque, quando duas tarefas ocorrem em sequencia, então $w_{i,j,n} = w_{i',j,n+1} = 1$, o que elimina a segunda parcela do lado direito da desigualdade. Quando uma dessas variáveis é igual a zero a equação é trivialmente satisfeita, pois o resultado do lado direito será no máximo $H - H = 0$.

$$\begin{aligned} Ts_{i'} \geq Tf_i - H\left(2 - w_{i,j,n} - w_{i',j,n+1}\right), \\ \forall j \in J, i, i' \in I_j, n \in N_j, n < |N_j|, i \neq i'. \end{aligned} \tag{3.4}$$

Se um dispositivo i' vem depois de outro, i, numa malha de controle m, então o tempo inicial de i' deve ser maior ou igual ao tempo final da tarefa i. Essa restrição é uma simplificação da encontrada em [30].

$$Ts_{i'} \geq Tf_i, \forall m \in M, (i, i') \in L_m. \tag{3.5}$$

Se a tarefa i precede i', ambas são executadas no mesmo dispositivo j e as tarefas estão numa mesma malha de controle m, então i' deve ser alocada num ponto de evento após aquele em que i está alocada. Essa restrição constitui um corte e não é necessária para o modelo representar o problema matematicamente com fidelidade. No entanto, ela torna o modelo mais justo, melhorando o desempenho computacional durante a otimização.[2]

$$\begin{aligned} \sum_{n' \in N_j : n' > n} w_{i',j,n'} \geq w_{i,j,n}, \\ \forall j \in J, m \in M, (i, i') \in L^1_{j,m}, n \in N_j. \end{aligned} \tag{3.6}$$

[2]Um modelo justo é um modelo em que as restrições do problema relaxado formam um poliedro que se aproxima do menor poliedro que engloba todas as soluções inteiras. Em geral, portanto, restrições aparentemente redundantes se apenas o espaço das variáveis inteiras for considerado podem ser relevantes para aproximar o problema relaxado e o problema inteiro.

Restrições de Tempo Final

Uma vez que o tempo de execução é fixo, essa restrição de igualdade pode ser simplificada da encontrada em [22]. Verdadeiramente, Tf_i poderia ser suprimida das variáveis, mas é deixada aqui intencionalmente para fins de claridade.

$$Tf_i = Ts_i + Te_i, \forall i \in I \tag{3.7}$$

Limites de Compel Data

As variáveis $TsCDs$ e $TfCDs$ são limites para todos os CDs da rede. Elas serão usadas na restrição seguinte.

$$\begin{aligned} TsCDs &\leq Ts_i, \forall i \in I_net \\ TfCDs &\geq Tf_i, \forall i \in I_net \end{aligned} \tag{3.8}$$

Restrições de Projeto

Essa restrição é específica para FF, mas poderia ser aplicada a outras redes.

$$TfCDs - TsCDs \leq R, \tag{3.9}$$

onde $R = \rho H$, sendo ρ uma fração do macrociclo explicada no próximo parágrafo.

De acordo com [15], não se recomenda que o tempo gasto em comunicações publicante/assinante exceda 50% ou 60% do tempo total do macrociclo para um novo projeto de rede, e 70% no caso geral (normalmente deixa-se um espaço para futuras ampliações da rede). Rigorosamente, isto é alcanaçado quando

$$\sum_{i \in I_{net}} Te_i \leq r'H, \tag{3.10}$$

onde $r' = 0.5$ ou $r' = 0.6$ para novos projetos e $r' = 0.7$ para o caso geral. Isso não depende da agenda, mas sim das malhas de controle configuradas. É, portanto, uma condição inicial de viabilidade. Como os tempos de Compel Data para uma rede são os mesmos para a maior parte dos sistemas, a equação 3.10 é imediatamente satisfeita ao se manter suficientemente baixo o número total de CDs. Por exemplo, um macrociclo requerido de 1000 ms permite no máximo 16 a 20 CDs de 30 ms cada numa nova rede.

É importante ressaltar que a banda FF H1 é relativamente pequena: a velocidade de 31.25kbps significa que em um milissegundo não é possível a trasmissão de 4 octetos (32 bytes). Sendo assim, uma operação como a passagem de *tokens* da comunicação acíclica já leva tempo significativo, que aumenta com o número de

dispositivos. Portanto, quando duas comunicações são realizadas muito próximas uma da outra, mas não são consecutivas, o tempo deixado entre elas é insuficiente para outras comunicações. Logo, este intervalo de tempo é desperdiçado e não pode ser aproveitado para comunicações sob demanda, como a requisição de dados pelo operador. Como o referido critério em [15] tem a intenção de assegurar que haja tempo para as outras comunicações, estes intervalos de tempo também deveriam ser considerados. Para o problema de agendamento, a equação 3.9 contrai a restrição ao tentar manter os CDs juntos um do outro. A restrição mais justa acontece quando $r = r'$. Valores de r levemente superiores ao de r' podem ser usados, desde que a restrição da equação 3.10 seja satisfeita também.

Observe que, para a restrição de agrupamento enunciada na subseção 3.2.3, a condição real que deve ser alcançada é

$$\max_{i \in I_{net}} Tf_i - \min_{i \in I_{net}} Ts_i \leq rH, \tag{3.11}$$

que corresponde exatamente ao agrupamento de CDs. Contudo, isso é alcançado se a equação 3.9 também o for: da equação 3.8, segue-se que

$$TfCDs \geq \max_{i \in I_{net}} Tf_i$$

e

$$TsCDs \leq \min_{i \in I_{net}} Ts_i.$$

Logo,

$$\max_{i \in I_{net}} Tf_i - \min_{i \in I_{net}} Ts_i \leq TfCDs - TsCDs \leq rH. \tag{3.12}$$

Restrições de *Readback*

Essas duas equações determinam se o *readback* r será transmitido antes da mensagem que o recebe (Eq. 3.13) ou depois da mensagem que o envia (Eq. 3.14).

$$Ts_{i_R} \geq Tf_r - Hy_r, \forall r \in RB, (i_R, i_S) \in LR_r. \tag{3.13}$$

$$Ts_r \geq Tf_{i_S} - H(1 - y_r), \forall r \in RB, (i_R, i_S) \in LR_r. \tag{3.14}$$

Se $y_r = 0$ a Equação 3.13 se torna $Ts_{i_R} \geq Tf_r$ ao passo que a Equação 3.14 é trivialmente satisfeita: dessa forma, tem-se apenas que i termina antes de i_R, a mensagem que o recebe, começa. Analogamente, se $y_r = 1$, a Equação 3.13 é trivialmente satisfeita enquanto a Equação 3.14 se torna $Ts_r \geq Tf_{i_S}$: assim, o *readback* é transmitido após o término do bloco que o envia. Lembre-se que essa distinção só é válida para um dado macrociclo. Considerando a repetição do

macrociclo periodicamente, cada *readback* estará entre a tarefa que o envia e a que o recebe.

3.3 Modelo MILP com Representação Temporal por Sobreposição

Nesta seção, uma formulação alternativa à da seção anterior é apresentada. A formulação representa o mesmo problema, mas com uma abordagem diferente para o problema de sequenciamento das tarefas em cada dispositivo: desta vez há variáveis binárias para cada par de tarefas, indicando se uma antecede ou sucede a outra. À primeira vista essa abordagem é a baseada em [3], mas optou-se por usar a de sobreposição conforme [10, 33] por esta ser mais compacta e resultar no mesmo conceito para o problema FF H1. Além disso, em [10], o autor compara ambos e observa um desempenho melhor do modelo por sobreposição nos casos estudados.

Como poderá ser observado, há poucas alterações no modelo com representação temporal por sobreposição com relação àquele com representação por intervalos de tempo. O mesmo conjunto de dados pode ser usado para os dois modelos, isto é, os conjuntos e parâmetros são os mesmos, com a exceção do conjunto de intervalos de tempo N_j, que deixa de existir aqui. Mesmo assim, todos os conjuntos, parâmetros, variáveis e restrições para este modelo serão aqui apresentados para facilitar a compreensão e a implementação. As mesmas definições sobre dispositivos, tarefas e malhas de controle feitas na seção anterior valem para esse modelo. Também são válidas para este caso as considerações sobre o *readback*.

Apenas para clarificar a representação por sobreposição, cabe aqui uma explicação. A variável binária $o_{i,i'}$ indica se i começa depois de i' terminar ($o_{i,i'} = 0$) ou antes ($o_{i,i'} = 1$). A sobreposição acontece quando $o_{i,i'} = o_{i',i} = 1$. O modelo não pode permitir que haja sobreposição num mesmo dispositivo.

Dadas as explicações iniciais, segue o enunciado do modelo. O código em GMPL encontra-se na Seção B.1.1

3.3.1 Restrições

Nesta subseção são definidas e explicadas as restrições que descrevem o problema de escalonamento. Essas restrições se adicionam às Equações 3.1, 3.5, 3.7, 3.8, 3.9, 3.13 e 3.14 afim de se obter o modelo completo.

Restrições de Sequenciamento

A primeira restrição de sequenciamento é a que segue. Ela é equivalente à equação 3.4 do modelo por Pontos de Evento.

$$Tf_{i'} - Ts_i \leq Ho_{j,i,i'}, \forall j \in J, i, i' \in I_j : i \neq i'. \tag{3.15}$$

Para um dispositivo j, se $o_{j,i,i'} = 0$ então a equação se torna $Tf_{i'} \leq Ts_i$, o que implica que a tarefa i' termina antes da tarefa i começar [10, 33]. Se $o_{j,i,i'} = 1$, a equação é trivialmente satisfeita, pois H é o limite superior dos tempos iniciais e finais de todas as tarefas.

Se a tarefa i precede i', ambas são executadas no mesmo dispositivo j e as tarefas estão numa mesma malha de controle m, então certamente i precede i'. Essa restrição constitui um corte, fixando algumas variáveis binárias e diminuindo a necessidade de averiguar alguns nós da árvore de busca:

$$o_{j,i,i'} = 1, \forall j \in J, m \in M, (i, i') \in L_{j,m}, n \in N_j. \tag{3.16}$$

Restrições de Sobreposição

A restrição abaixo, adaptada de [10, 33], impede a sobreposição num dispositivo, que é o caso em que $o_{j,i,i'} = o_{j,i',i} = 1$.

$$o_{j,i,i'} + o_{j,i',i} \leq 1, \forall j \in J, i, i' \in I_j : i \neq i'.$$

Complementarmente, nota-se que a situação $o_{j,i,i'} = o_{j,i',i} = 0$ não é desejável, pois implica que i' termina antes da tarefa i começar ($o_{j,i,i'} = 0$) e i termina antes de i' começar ($o_{j,i',i} = 0$). São dois eventos mutuamente contraditórios e, portanto, essa possibilidade também pode ser eliminada. Resta assim a seguinte equação:

$$o_{j,i,i'} + o_{j,i',i} = 1, \forall j \in J, i, i' \in I_j : i \neq i'. \tag{3.17}$$

Essa restrição constitui um corte muito poderoso, pois diminui pela metade o número de variáveis binárias.

3.4 Critérios de Otimização

Os modelos apresentados, ainda que sem uma função objetivo, levam a um problema de viabilidade que é de grande importância na fase de projeto de redes FF H1. Se este problema for inviável, então dispositivos correspondentes a uma malha de controle escolhida devem ser realocados em outra rede. Por outro lado, na operação das redes é importante que o desempenho tanto das comunicações das redes quanto do

controle dos processos seja elevado. Nesta seção, três critérios de otimização são propostos.

Na Subseção 3.4.1, apresenta-se o critério de maximização do agrupamento dos CDs. Na prática, para manter todos os critérios como minimizações, ele é enunciado como minimização da separação de CDs. Trata-se de um problema minimax, que será convertido a um conjunto de restrições lineares mais um critério de otimização linear. Na Subseção 3.4.2, o critério de minimização dos atrasos internos das malhas de controle é apresentado e justificado à luz da teoria de sistemas de controle em rede. Um terceiro critério, de minimização do tempo final de execuções é descrito na Subseção 3.4.3. Este critério tem por fim remover um grau de liberdade ao mesmo tempo que dá um indício do menor macrociclo possível para um dado problema. Por fim, como os critérios propostos são conflitantes entre si, uma composição das três funções objetivo é feita em 3.4.4.

3.4.1 Maximização do Agrupamento de Compel Data

Como já foi discutido no Capítulo 1 e neste capítulo, na Seção 3.2.3, é desejável que pouco ou nenhum tempo seja deixado entre duas comunicações programadas. O caso ideal é aquele em que todos os CDs estão agrupados por completo, de forma que mais e maiores conjuntos de dados possam ser enviados em sequência sem que haja interrupções nas comunicações assíncronas. Vários pequenos intervalos de tempo entre CDs levariam a muitas interrupções de comunicações assíncronas devido à maior prioridade do CD. O agrupamento pode ser definido como

$$G = 1 - \frac{\max\limits_{i \in I_{net}} Tf_i - \min\limits_{i \in I_{net}} Ts_i}{H}, \tag{3.18}$$

Onde $G \in \mathbb{R}$ é o fator de agrupamento, variando de 0 a $1 - \sum_{i \in I_{net}} Te_i/H$. Maximizar G é equivalente a minimizar o fator de separação de CDs S' definido como

$$S' = \max_{i \in I_{net}} Tf_i - \min_{i \in I_{net}} Ts_i. \tag{3.19}$$

Como é desejável manter linearidade, a função objetivo a ser usada é

$$S = TfCDs - TsCDs. \tag{3.20}$$

Pelo resultado obtido na equação 3.12, segue-se que $\inf S = S'$ e logo minimizar S é equivalente a minimizar S'.

3.4.2 Minimização de Atrasos

Um dos aspectos mais importantes em NCSs é a existência de atrasos inerentes. Num NCS geral, tempos de execução das tarefas e tempos de comunicação resultam em atrasos entre a aquisição dos sinais de entrada e a ação no sistema. Esses atrasos podem degradar o desempenho dos sistemas de controle projetados sem considerar tais atrasos e até desestabilizar o sistema [5]. Portanto, um bom algoritmo de escalonamento deveria buscar minimizar os atrasos da rede.

Para uma malha de controle m, o atraso de fim-a-fim é definido como

$$\max_{(i,i')\in L_m} Tf_{i'} - \min_{(i,i')\in L_m} Ts_i. \tag{3.21}$$

Essa equação não é linear e não pode ser usada no modelo. O critério de atraso para uma malha de controle m é definido como

$$\sum_{(i,i')\in L_m} (Ts_{i'} - Ts_i). \tag{3.22}$$

Observe que o valor obtido na equação 3.22 nem sempre é igual ao verdadeiro valor do atraso, mas funciona como uma métrica de atraso. Se a malha de controle tem apenas um caminho de fim-a-fim, o resultado é o mesmo que o atraso entre a primeira execução e a última. No caso geral, a equação 3.22 dará a soma temporal de todos os ramos do diagrama. Note que esse critério é mais adequado que o da equação 3.21 quando múltiplas malhas de controle estiverem interconectadas, uma vez que ele considera todos os extremos do diagrama.

Muitas malhas de controle são tipicamente executadas em uma mesma rede e minimizar os atrasos de todas as malhas de controle pode ser conflitante se o macrociclo requerido não for longo o suficiente. Assim, para o critério de atraso geral uma soma ponderada pode ser aplicada. Para um conjunto de malhas de controle M, o critério de atraso D é definido como

$$D = \sum_{m\in M,(i,i')\in L_m} a_m(Ts_{i'} - Tsi), \tag{3.23}$$

onde $a_m \in \mathbb{R}^+$ é o peso para a minimização do atraso da malha de controle m.

3.4.3 Minimização do Tempo Final

O último critério de otimização é a minimização do tempo final da tarefa mais tardia, o que é equivalente a diminuir a agenda. O tempo final é definido por

$$TF' = \max_{i\in I} Tf_i. \tag{3.24}$$

Mais uma vez, como isso torna o problema num problema minimax, uma variável auxiliar $TF \in \mathbb{R}^+$ é criada e o seguinte conjunto de inequações deve ser adicionado às formulas da subseção 3.2.3:

$$TF \geq Tf_i, \forall i \in I. \tag{3.25}$$

Assim, a função objetivo para o critério de tempo final é a própria TF.

Apesar desse critério não interferir por si só no desempenho do controle nem na utilização da rede, adicionar este objetivo dá uma indicação do mínimo macrociclo aceitável, além de remover um grau de liberdade, que é o tempo inicial de todas as execuções. Considerando apenas os objetivos das seções 3.4.2 e 3.4.1, o mesmo valor das funções objetivo seria obtido variando Ts_i e Tf_i de todas as tarefas pela mesma quantidade de tempo.

3.4.4 Composição da Função Objetivo

A fim de gerar uma única função objetivo composta, é proposta a soma ponderada das funções objetivo explicadas nas seções 3.4.1, 3.4.2 e 3.4.3. A função objetivo composta é uma combinação convexa dada por

$$F = \alpha S + \beta D + (1 - \alpha - \beta)TF. \tag{3.26}$$

A tabela abaixo mostra as restrições aplicáveis a cada modelo.

Restrições	Pontos de Evento	Sobreposição
Limites	Eq. 3.1	Eq. 3.1
Execução	Eq. 3.2 Eq. 3.3	
Sequenciamento	Eq. 3.4 Eq. 3.5 Eq. 3.6	Eq. 3.15 Eq. 3.5 Eq. 3.16
Tempo Final	Eq. 3.7 Eq. 3.25	Eq. 3.7 Eq. 3.25
Compel Data	Eq. 3.8	Eq. 3.8
Projeto	Eq. 3.9	Eq. 3.9
Readback	Eq. 3.13 Eq. 3.14	Eq. 3.13 Eq. 3.14

Tabela 3.1: Restrições aplicadas a cada modelo para o problema de otimização com função objetivo F

Os pesos $\alpha, \beta \in \mathbb{R}^+ : \alpha + \beta \leq 1$ são para os critérios de separação e atraso, respectivamente. Apesar de haver conflito entre os três objetivos, a prática mostra que a maior parte das combinações de três valores para os três pesos geram resultados

bons e semelhantes. Em geral, recomenda-se que o peso α seja o mais alto dentre α, β e $1 - \alpha - \beta$, já que é recomendável que o agrupamento de CDs seja priorizado para futuras ampliações das redes. A penalidade devido ao compromisso com o critério de atrasos pouco provavelmente afetaria o desempenho do controle, pois essa penalidade é da ordem de 10 a 30 ms. Recomenda-se que o peso para o tempo final seja o menor de todos - aproximadamente 1% do menor entre α e β.

3.5 Ajustes Finais do Modelo

Antes de implementar o modelo, alguns ajustes, como redução do modelo e cortes, devem ser feitos de forma a melhorar o desempenho da otimização. Na Subseção 3.5.1, é descrita uma restrição que configura um corte para diminuir a degeneração do problema de otimização.

3.5.1 Cortes por Equivalência

O problema de escalonamento para FF pode ter várias soluções equivalentes, uma vez que múltiplas tarefas em paralelo podem ter tempos de execução iguais e a troca de ordem entre as mesmas não faria a menor diferença em critérios de otimização que não privilegiem umas em detrimento de outras. Nesse caso, é possível aplicar um corte, arbitrando a ordem dentre elas. Trata-se de um corte inválido na nomenclatura de MILP, isto é, que reduz a região viável do problema. Isso não quer dizer que seja impróprio para aplicação, desde que haja conhecimento do problema.

Os pares de tarefas equivalentes encontram-se no conjunto E apresentado nas seções 3.2 e 3.3. É preciso então definir o que são tarefas equivalentes. Isso pode ser feito através da seguinte definição recursiva:

Definição 3.2 *Tarefas equivalentes Duas tarefas são equivalentes se e somente se:*

- *Possuem tempos de execução iguais,*
- *Pertencem a malhas de controle de importâncias iguais (i.e., pesos a_m iguais),*
- *Possuem períodos de execução iguais (no caso deste capítulo, isto sempre é verdade),*
- *São precedidas e sucedidas por:*

  Nenhuma outra tarefa ou

  A mesma tarefa e/ou

  Pares de tarefas equivalentes entre si.

É importante ressaltar que é possível que se formem subgrafos conexos de tarefas equivalentes, doravante chamados de subgrafos equivalentes. É importante manter a coerência: a tarefa de um mesmo subgrafo deve ser sempre o primeiro elemento da dupla, enquanto a tarefa equivalente do subgrafo equivalente é o segundo.

Encontrados os pares de tarefas equivalentes, a seguinte restrição pode ser aplicada:

$$Ts_i \geq Ts_{i'}, \forall (i, i') \in E \tag{3.27}$$

Essa restrição apenas arbitra uma ordem entre um par de tarefas equivalentes.

3.5.2 Redução do Modelo

Antes de aplicar o modelo ao problema, uma pequena redução do modelo é recomendada para eliminar algumas variáveis e tornar o modelo mais justo. Uma possibilidade é a concatenação de tarefas que ocorram em sequencia. Se existe um subconjunto I_j^s de I_j em que as seguintes condições são satisfeitas:

- Todas as tarefas em I_j^s são executadas na mesma malha de controle m e existe um caminho de conexões entre todos eles e apenas eles;
- No máximo uma tarefa $i^s \in I_j^s$ está pré-conectada a tarefas que não pertençam a I_j^s. Tal tarefa i^s não pode estar pré-conectada a qualquer tarefa pertencente a I_j^s;
- No máximo uma tarefa $i^s \in I_j^s$ está pós-conectada a tarefas que não pertençam a I_j^s. Tal tarefa i^s não pode estar pós-conectada a qualquer tarefa pertencente a I_j^s,

então pode-se converter o conjunto I_j^s em um novo conjunto I_j' que só possui uma tarefa i' tal que $Te_{i'} = \sum_{i \in I_j^s} Te_i$. A ordem de execução das tarefas originais deve ser escolhida manualmente ou por qualquer critério que respeite a sequência da malha de controle, uma vez que ela não interfere nos resultados da otimização.

Capítulo 4

Resultados e Comparações com Sistemas de Controle Comerciais

Neste capítulo, dois diferentes problemas de escalonamento são testados usando a formulação proposta. Ambos são retirados de malhas de controle FF práticas para redes de tamanho industrial, ou seja, números práticos de dispositivos e malhas de controle. Todas as informações dos dispositivos vêm de Capability Files de dispositivos comerciais. Os resultados são comparados com as agendas obtidas do sistema de controle Emerson DeltaV 8.4. Todas as execuções de otimização foram realizadas com $a_m = 1, \forall m$, $\alpha = 0.9$ e $\beta = 0.099$.

Existe um número considerável de ferramentas eficientes e amplamente testadas para resolver MILPs, incluindo comerciais e não-comercias. Linguagens para modelagem algébrica oferecem uma maneira direta de escrever modelos matemáticos e incluir dados. Além disso, eles podem ser incorporados e/ou utilizados em conjunto com outros softwares. A combinação dessas linguagens com os *solvers* faz com que modelagem via MILP seja facilmente aplicável na indústria e em desenvolvimento de software. O modelo também pode ser usado em redes semelhantes, que sejam determinísticas e de preferência permitam funções de controle distribuídas nos seus dispositivos.

O *solver* de otimização aqui usado foi o *GNU Linear Programming Kit* (GLPK) LP/MIP Solver, v4.46, que é um *solver* de código aberto. Ele foi usado com configurações padrão, com exceção do algoritmo de *Backtracking*, que foi configurado para Busca em Largura (BFS, do inglês *Breadth First Search*). O computador que executou o software é um notebook convencional, com Intel© Core™ 2 Duo T6500 com 2.10GHz para cada processador e uma memória de 4GB.

Em adição à comparação dos critérios, o Mínimo Macrociclo Admissível (MMA) para uma rede também será comparado. O MMA considera as restrições em [15], já descritas nas equações 3.1 and 3.11. O MMA é aqui definido como

$$\text{MMA} = \max\left(\frac{S'}{\rho}, TF\right). \tag{4.1}$$

4.1 Caso I – Rede com Quatro Malhas de Controle Simples

No primeiro caso, quatro malhas de controle simples são configuradas numa rede contendo cinco transmissores e cinco posicionadores de válvula. Os blocos de função configurados e seus tempos de execução estão descritos na tabela 4.1. O tempo configurado para o CD é de 30 ms, que é o padrão para o DeltaV 8.4. O macrociclo requerido é de 1000 ms. A restrição de projeto para comunicação publicante/assinante ρ é 0.5. Os diagramas de malhas de controle estão detalhados nas figuras 4.1, 4.2 e 4.3. O código para entrada de dados no GLPK pode ser visto na Seção B.1.2 do Apêndice.

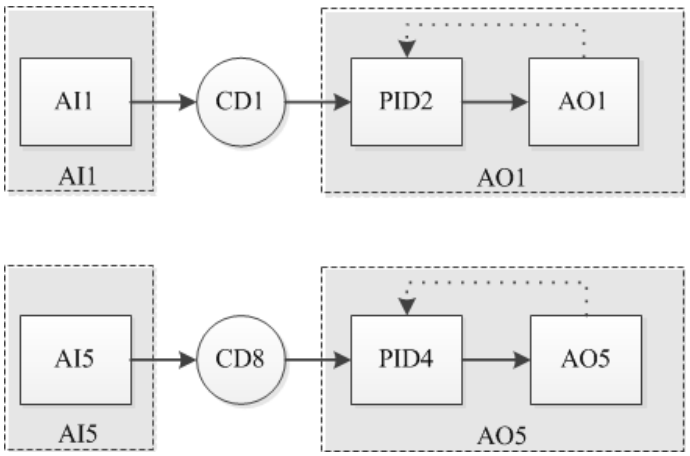

Figura 4.1: Diagramas de duas malhas PID simples configuradas na rede do Caso I.

Observe as transformações da Tabela 4.1 a fim de reduzir o modelo. Comparando com as malhas de controle das Figuras 4.1, 4.2 e 4.3, fica evidente que foram tarefas consecutivas de um mesmo dispositivo foram agrupadas. Para fins de escalonamento os blocos PID2 e AO1, assim como PID4 e AO5 (ambos na Figura 4.1) são como se fosse um único bloco cada. No caso de PID2 e AO1, o tempo inicial de AO1 seria o tempo inicial de PID2 somado ao tempo de execução de PID2: AO1 seria executado logo na sequência. Cabe ressaltar que tais simplificações só podem ser aplicadas porque não há outros blocos a serem executados no dispositivo AO5.

Tabela 4.1: Dispositivos, tarefas e tempos de execução para o Caso I.

j	i	Te_i (ms)
AI1	AI1	25
AI2	AI2	30
AI3	AI3	30
AI4	AI4	20
	PID1	25
	SPLTR1	20
AI5	AI5	30
AO1	PID2	50
	AO1	40
AO2	ISEL1	25
	PID3	30
	AO2	25
AO3	AO3	40
AO4	AO4	40
AO5	PID4	30
	AO5	25
net	CD1	
	⋮	30
	CD8	

(a) Dispositivos j, Tarefas i e seus tempos de execução correspondentes Te_i.

$\Longrightarrow$

j	i'	$Te_{i'}$ (ms)
AI1	AI1	25
AI2	AI2	30
AI3	AI3	30
AI4	AI41	45
	AI42	20
AI5	AI5	30
AO1	AO1	90
AO2	AO2	80
AO3	AO3	40
AO4	AO4	40
AO5	AO5	55
net	CD1	
	⋮	30
	CD8	

(b) Dispositivos j, Tarefas ajustadas i' e seus tempos de execução correspondentes $Te_{i'}$.

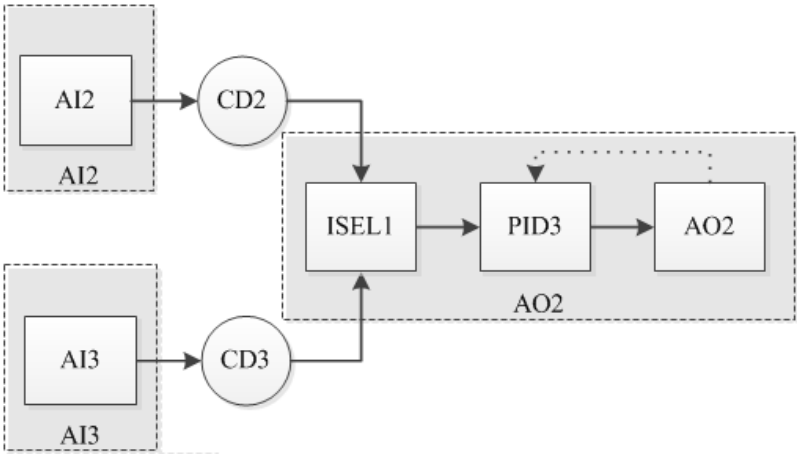

Figura 4.2: Diagrama de uma malha PID com transmissores redundantes configuradas na rede do Caso I.

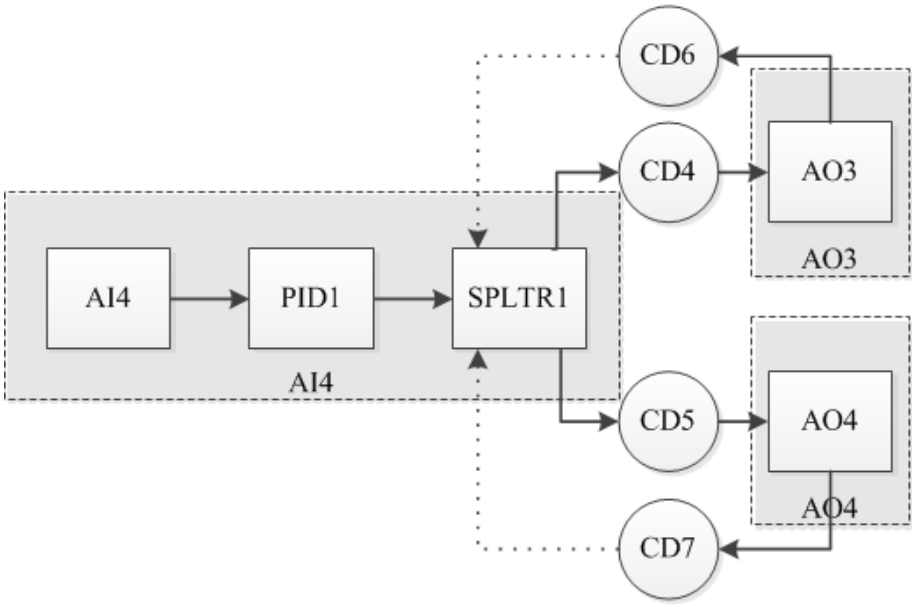

Figura 4.3: Diagrama de uma malha PID *split-range* configurada na rede do Caso I.

4.1.1 Representação da Configuração usando o Modelo

A fim de aplicar o modelo, os conjuntos e parâmetros devem ser determinados. Os conjuntos J, I e I_j, bem como os parâmetros Te_i dos blocos de função podem ser obtidos da tabela 4.1. Para as malhas de controle retratadas nas figuras 4.1, 4.2 e 4.3, os seguintes conjuntos podem ser determinados:

$$M = \{1, 2, 3, 4\}$$

$$L_1 = \{(AI1, CD1), (CD1, AO1)\},$$

$$L_2 = \{(AI2, CD2), (AI3, CD3), (CD2, AO2), (CD3, AO2)\},$$

$$L_3 = \{(AI4, CD4), (AI4, CD5), (CD4, AO3), (CD5, AO4)\},$$

$$L_4 = \{(AI5, CD8), (CD8, AO5)\}.$$

$$RB = \{CD6, CD7\},$$

$$LR_{CD6} = \{(AI42, AO3)\},$$

$$LR_{CD7} = \{(AI42, AO4)\}.$$

$$E = \{(AI2, AI3), (CD2, CD3), (CD4, CD5), (AO3, AO4)\}.$$

4.1.2 Estatísticas do Problema

Os mesmos conjuntos e parâmetros, aplicados aos modelos por Pontos de Evento e por Sobreposição, levam a problemas de otimização estruturalmente distintos. A Tabela 4.2 revela as estatísticas do problema sob ambas representações no GLPK. Observe que as estatísticas são quase todas equivalentes, com a exceção do número de linhas, ou restrições, que por sua vez interfere no número de não-zeros. Mesmo assim, a densidade de não-zeros também é maior na formulação por Pontos de Evento: 3.6% contra 2.3% na formulação por Sobreposição. Outro detalhe que chama a atenção foi o fato de que a etapa de pré-solução do GLPK quase nada conseguiu reduzir no modelo por sobreposição.

Tabela 4.2: Caso I: Estatísticas do Problema (entre parênteses, após pré-solução).

	Pontos de Evento	Sobreposição
Variáveis	120 (107)	101 (99)
Variáveis binárias	79 (68)	60 (60)
Colunas	120 (107)	101 (99)
Linhas	525 (488)	193 (190)
Não-Zeros	1972 (1874)	460 (442)

4.1.3 Resultados para o Caso I

A Tabela 4.3 mostra as estatísticas de solução do Caso I no GLPK para as dois modelos. O *gap* inicial é uma estimativa da lacuna de integralidade entre o modelo relaxado e o modelo inteiro, sendo definido pela seguinte fórmula: seja BI a primeira solução inteira encontrada e BR a primeira solução ótima do problema relaxado. O *gap* é dado por

$$gap = \frac{||BI|| - ||BR||}{||BI||}. \tag{4.2}$$

Observe que, apesar do problema parecer inicialmente com o mesmo grau de justeza para ambos os modelos, os tempos de solução foram significativamente diferentes entre eles. O número de nós buscados foi quatro vezes menor do que no modelo por Pontos de Evento. O número de iterações de Programação Linear (LP, do inglês *Linear Programming*) por segundo no modelo por Sobreposição, de cerca de 8600, também foi superior ao do modelo por Sobreposição, de 4700: um fator de quase dois. Isso indica um problema LP mais denso no caso do problema por Pontos de Evento, provavelmente por causa da quantidade maior de restrições e da densidade de não-zeros.

De qualquer forma, os tempos de solução foram absolutamente satisfatórios para ambos os modelos, por se tratar de uma otimização offline. Um tempo de solução de 5s como no caso do modelo por Sobreposição pode ser quase imperceptível para o usuário.

Tabela 4.3: Caso I: Estatísticas de Solução.

	Pontos de Evento	Sobreposição
Tempo de Solução	28.6s	5.0s
Nós Buscados	32667	8227
Iterações LP	134639	43006
Gap Inicial	59.6%	54.3%

No que diz respeito ao resultado, ambos os modelos chegaram ao mesmo valor. A Figura 4.4 mostra um gráfico de Gantt de uma agenda obtida no sistema DeltaV, enquanto a Figura 4.5 revela um gráfico de Gantt da agenda obtida pela otimização, para ambos os modelos. Pode-se ver que a otimização foi capaz de manter todas os CDs agrupados. No entanto, os atrasos de controle resultantes dessa agenda ótima não são mínimos em função do compromisso entre os objetivos. Se fosse desejado minimizar os atrasos de controle, seria preciso aumentar o ganho β na função objetivo. Como está sendo dada prioridade ao problema de agrupamento, este resultado se mostra extremamente satisfatório.

A Tabela 4.4 compara a agenda obtida no DeltaV com a agenda obtida pela otimização. Observe que há ganhos muito significativos em todos os critérios abordados. A separação foi mantida no mínimo ($8 \times 30 = 240$) e recuperou aproximados 55ms de comunicação potencialmente desperdiçada. Logo, enquanto a agenda obtida no DeltaV proporciona 705ms de comunicação ininterrupta, a comunicação ótima ocorre em 760ms – um ganho de 7.8% na qualidade de comunicação assíncrona.

No entanto, o que mais chama a atenção são os ganhos com o critério de atrasos de controle. O algoritmo de escalonamento do DeltaV 8.4 aparentemente determina tempo inicial igual a zero para todas as tarefas sem restrições de precedência – na maioria das vezes, os blocos AI. Isso provoca um efeito cascata nos atrasos conforme mais malhas de controle vão sendo adicionadas à rede. A primeira malha tem atraso mínimo, com critério da Eq. 3.21 igual a 55ms, mas os atrasos vão crescendo até chegar aos impressionantes 205ms da malha de controle 4, que é um simples PID, contra os 60ms mínimos alcançados na otimização. A melhoria chega a 70.7%.

Adicionalemnte, a comunicação otimizada traz uma melhoria de 17.2% no tempo final. Um ponto de melhoria muito relevante que vem a reboque da minimização do critério de Separação é o MMA obtido: 480ms. Ou seja, seria possível diminuir pela metade o Macrociclo, de 1000ms para 500ms sem com isso violar as restrições de

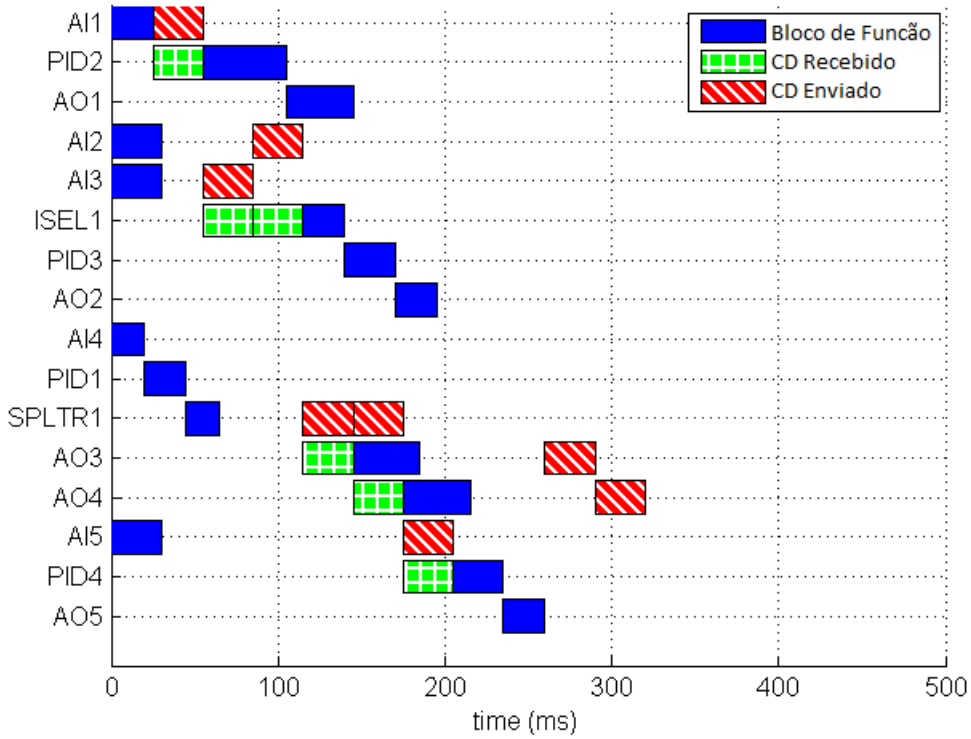

Figura 4.4: Gráfico de Gantt da agenda obtida no DeltaV para o Caso I.

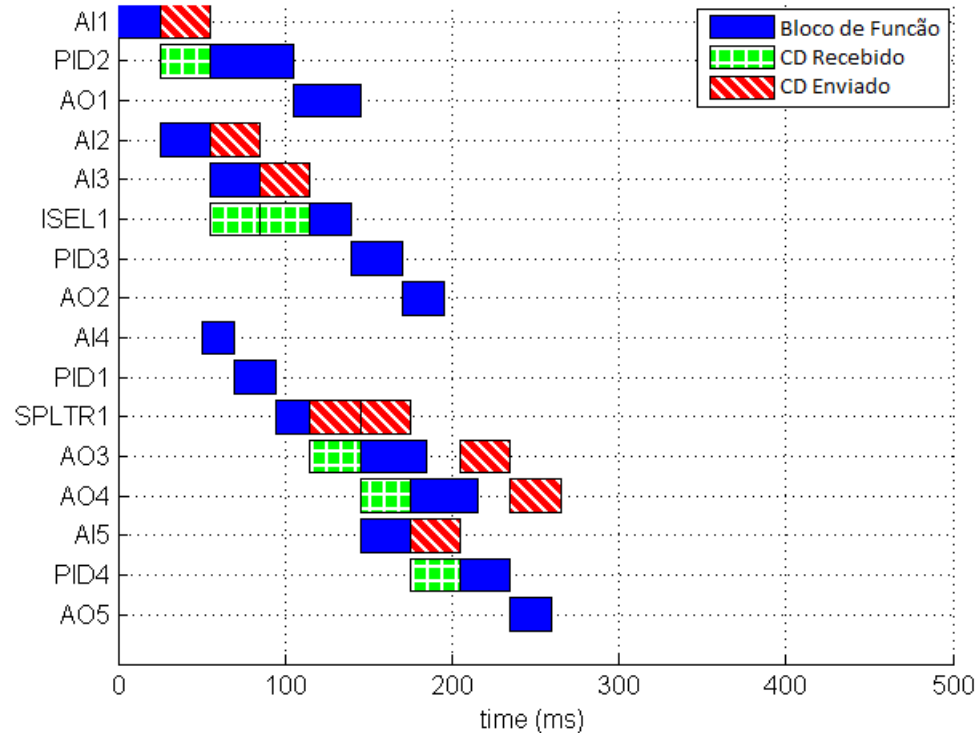

Figura 4.5: Gráfico de Gantt da agenda obtida pela otimização para o Caso I.

projeto em [15]. Trata-se de um caso real e provado de rede com quatro malhas de controle, sendo que duas não são simples PIDs, em que é possível um macrociclo de 500ms. Ainda haveria 52% do tempo livre para 260ms de comunicação sob demanda ininterrupta por ciclo.

Tabela 4.4: Caso I: Comparação de resultados entre a agenda do DeltaV e a ótima.

Critério		DeltaV	Otimização	Redução (%)
Separação (Eq. 3.19) (ms)		295	240	18.6
Atrasos (Eq. 3.21) (ms)	Malha 1	55	55	0
	Malha 2	170	120	29.4
	Malha 3	380	250	34.2
	Malha 4	205	60	70.7
Tempo Final (Eq. 3.24) (ms)		320	265	17.2
MMA (Eq. 4.1) (ms)		590	480	18.6

4.2 Caso II - Rede com Malha de Controle Complexa

No segundo caso, quatro malhas de controle são configuradas numa rede contendo cinco transmissores e cinco atuadores, mas, em contraponto com o Caso I, duas das malhas de controle compartilham o mesmo bloco de função, o que cria uma única malha de controle, maior, com diversas interconexões. Os blocos de função configurados e seus tempos de execução estão detalhados na tabela 4.5. O tempo de comunicação publicante/assinante configurado é de 30 ms. O macrociclo requerido é de 1000 ms. A restrição de projeto para Compel Data ρ é 0.5. Os diagramas de malhas de controle estão detalhados nas Figuras 4.6, 4.7 e 4.8. Nas malhas de controle das Figuras 4.6 e 4.7, há uma saída de PID proveniente de outra rede que entra no Seletor de Sinal desta. As malhas de controle da Figura 4.8 estão interconectadas: o primeiro Seletor de Entrada (ISEL, do inglês *Input Selector*), um seletor de transmissores redundantes, envia seu valor para um simples PID (topo) e um controlador *split-range* com um terceiro transmissor associado.

Para entrada de dados no GLPK, veja o código na Seção B.1.2.

4.2.1 Representação da Configuração usando o Modelo

Os conjuntos J, I e I_j, bem como os parâmetros Te_i dos blocos de função podem ser obtidos da tabela 4.5. Para as malhas de controle retratadas nas figuras 4.6, 4.7

j	i	Te_i (ms)
AI11	AI1	30
AI12	AI2	30
AI13	AI3	30
AI14	AI4	30
AI15	AI5	30
AI15	ISEL4	30
AO11	PID1	30
AO11	ISEL1	25
	AO1	25
AO12	PID2	30
AO12	ISEL2	25
	AO2	25
AO13	ISEL3	25
AO13	PID3	30
	AO3	25
AO14	PID4	30
AO14	SPLTR1	25
AO14	AO4	25
AO15	AO5	25
net	CD1 ⋮ CD10	30

(a) Dispositivos j, Tarefas i e os seus temos de execução correspondentes Te_i.

$\Longrightarrow$

j	i'	$Te_{i'}$ (ms)
AI11	AI1	30
AI12	AI2	30
AI13	AI3	30
AI14	AI4	30
AI15	AI5	30
AI15	ISEL4	30
AO11	PID1	30
AO11	AO1	50
AO12	PID2	30
AO12	AO2	50
AO13	ISEL3	25
AO13	AO3	55
AO14	PID4	30
AO14	SPLTR1	25
AO14	AO4	25
AO15	AO5	25
net	CD1 ⋮ CD10	30

(b) Dispositivos j, Tarefas ajustadas i' e seus tempos de execução correspondentes $Te_{i'}$.

Tabela 4.5: Dispositivos, Tarefas e seus tempos de execução para o Caso 2.

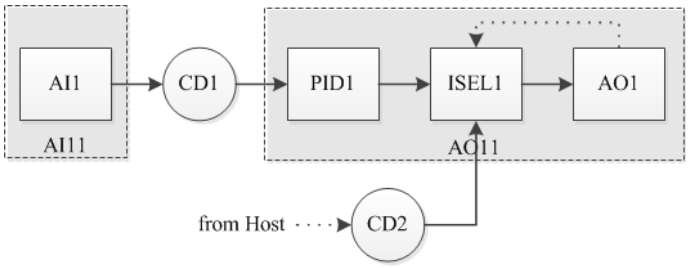

Figura 4.6: Diagrama de uma malha de controle PID *override* configurado na rede do Caso II. Uma das entradas do Seletor de Sinal vem de outra rede.

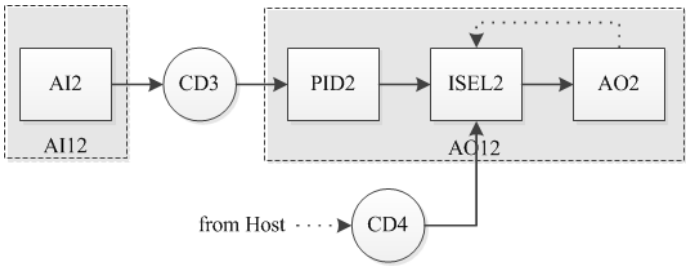

Figura 4.7: Diagrama de outra malha de controle PID *override* configurado na rede do Caso II. Uma das entradas do Seletor de Sinal vem de outra rede.

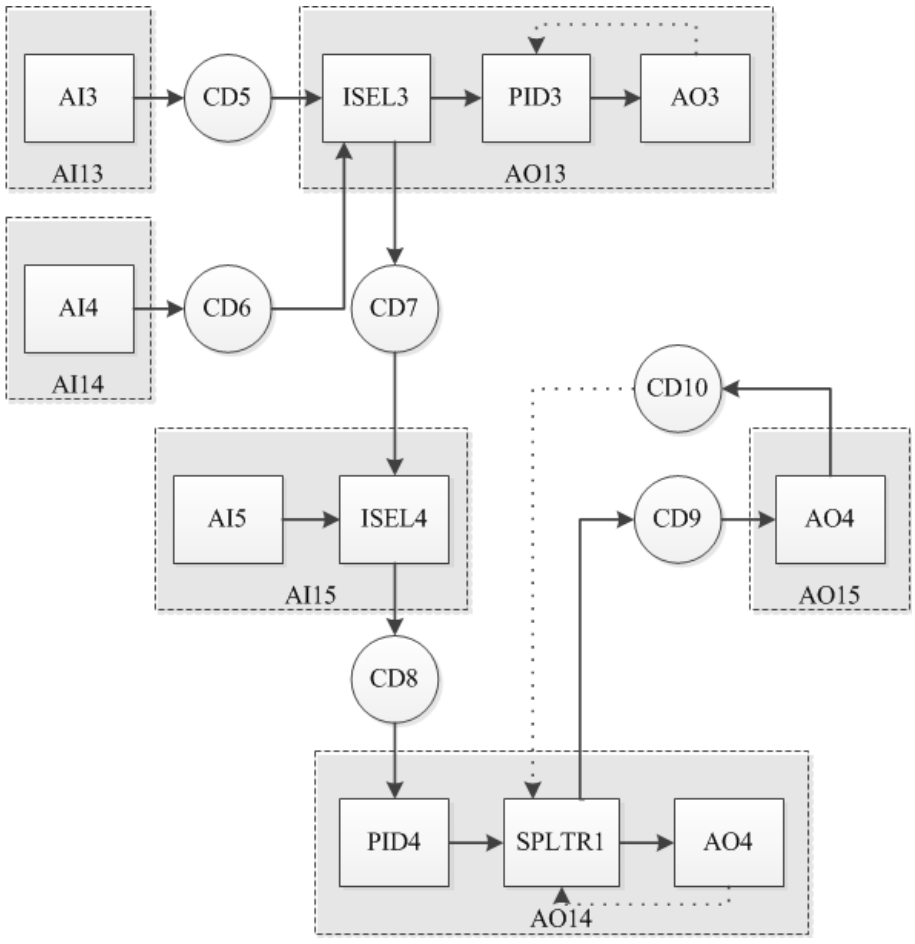

Figura 4.8: Diagrama de duas malhas de controle interconectadas, configuradas na rede do Caso II.

e 4.8, os seguintes conjuntos podem ser determinados:

$$M = \{1, 2, 3\}$$

$$\begin{aligned} L_1 &= \{(AI1, CD1), (CD1, AO11), (AO11, AO12), \\ &\quad (CD2, AO12)\}, \end{aligned}$$

$$\begin{aligned} L_2 &= \{(AI2, CD3), (CD3, AO21), (AO21, AO22), \\ &\quad (CD4, AO22)\}, \end{aligned}$$

$$\begin{aligned} L_3 &= \{(AI3, CD5), (AI4, CD6), (CD5, AO31), \\ &\quad (CD6, AO31), (AO31, AO32), (AO31, CD7), \\ &\quad (AI51, AI52), (CD7, AI52), (AI52, CD8), \\ &\quad (CD8, AO41), (AO41, AO42), (AO42, AO43), \\ &\quad (AO42, CD9), (CD9, AO5), (AO5, CD10)\}, \end{aligned}$$

$$L^1_{AI5,3} = \{(AI51, AI52\},$$

$$L^1_{AO1,1} = \{(AO11, AO12)\},$$

$$L^1_{AO2,2} = \{(AO21, AO22)\},$$

$$L^1_{AO3,3} = \{(AO31, AO32)\},$$

$$L^1_{AO4,3} = \{(AO41, AO42), (AO42, AO43)\},$$

$$L^1_{net,3} = \{(CD5, CD7), (CD6, CD7), (CD7, CD8), (CD8, CD9)\},$$

$$L^1_{j,m} = \emptyset \text{ para os demais } j, m.$$

$$RB = \{CD10\},$$

$$LR_{CD10} = \{(AI42, AO5)\}.$$

$$\begin{aligned} E &= \{(AI1, AI2), (CD1, CD3), (AO11, AO21), (AO12, AO22), \\ &\quad (CD2, CD4), (AI3, AI4), (CD5, CD6)\}. \end{aligned}$$

Nas representações propostas neste trabalho, uma malha de controle é um conjunto interligado de tarefas. Logo, ainda que haja quatro malhas do ponto de vista de controle, do ponto de vista de escalonamento existem apenas três. As malhas de controle da Figura 4.8 correspondem à malha de controle 3 do modelo.

A Tabela 4.6 revela as estatísticas do problema sob ambas representações no GLPK. Novamente, as estatísticas são quase todas equivalentes, com a exceção do número de restrições. A densidade de não-zeros também é maior na formulação por Pontos de Evento: 2.8% contra 1.5% na formulação por Sobreposição. Em ambos os casos, trata-se de um problema menos denso que o do Caso I. Por outro lado, é um problema com mais variáveis, ao mesmo tempo em que existe uma maior proporção restrições/variáveis neste caso quando comparado ao Caso I, para ambas as formulações. Isso se deve principalmente à malha 3, que tem muitas relações de precedência e, com isso, restrições adicionais. Por fim, da mesma maneira que no Caso I, a etapa de pré-solução do GLPK reduziu muito pouco o modelo por sobreposição.

Tabela 4.6: Caso II: Estatísticas do Problema (entre parênteses, após pré-solução).

	Pontos de Evento	Sobreposição
Variáveis	186 (143)	160 (160)
Variáveis binárias	131 (89)	105 (105)
Colunas	186 (143)	160 (160)
Linhas	1071 (851)	313 (312)
Não-Zeros	4216 (3369)	747 (730)

4.2.2 Resultados para o Caso II

A Tabela 4.7 mostra as estatísticas de solução do Caso II no GLPK para as dois modelos. Observe que novamente há uma diferença de ordem de grandeza entre os tempos de solução para as duas formulações apresentadas. Enquanto no Caso I os *gaps* iniciais foram equivalentes, o *gap* inicial para o Caso II também foi menor (cerca de 30%) para o modelo por Sobreposição. O número de nós buscados foi sete vezes menor do que no modelo por Pontos de Evento e o número de iterações LP por segundo, de cerca de 6500 no modelo por Sobreposição, foi duas vezes e meia superior aos 2600 do modelo por Pontos de Evento. Isso indica um problema LP mais denso no caso do problema por Pontos de Evento, provavelmente por causa da quantidade maior de restrições e da densidade de não-zeros.

Como no Caso I, os tempos de solução foram absolutamente satisfatórios para ambos os modelos, especialmente no caso do modelo por Sobreposição, onde apenas 1s foi necessário. Pode parecer surpreendente à primeira vista o fato de que este caso, aparentemente mais complexo que o Caso I, tenha sido mais rápido para se resolver. No entanto, em se tratando de otimização combinatória, esse problema é mais simples. Isto porque existem menos possibilidades de ordenação em função

das muitas relações de precedência pré-determinadas pela estrutura da malha de controle 3. Assim, os modelos propostos neste trabalho se tornam especialmente relevantes quanto mais complexas forem as malhas de controle utilizadas.

Tabela 4.7: Caso I: Estatísticas de Solução.

	Pontos de Evento	Sobreposição
Tempo de Solução	14.5s	1.0s
Nós Buscados	8943	1225
Iterações LP	37968	6546
Gap Inicial	43.0%	29.9%

Como no Caso I, ambos os modelos chegaram ao mesmo resultado. A Figura 4.9 mostra um gráfico de Gantt de uma agenda obtida no sistema DeltaV, enquanto a Figura 4.10 ilustra um gráfico de Gantt da agenda obtida pela otimização proveniente de ambos os modelos. Pode-se ver que a otimização foi capaz de manter todas os CDs agrupados, com uma penalidade no atraso de controle da malha de controle 1.

A Tabela 4.8 compara a agenda obtida no DeltaV com a agenda obtida pela otimização. Os ganhos neste caso foram ainda mais significativos que os do Caso I, em todos os critérios abordados. A separação foi mantida no mínimo ($10 \times 30 = 300$) mesmo com a complexidade estrutural do problema, chegando a uma diferença de 130ms com relação à agenda do DeltaV. Esses 130ms, fragmentados entre vários CDs, teriam pouca utilidade para comunicação assíncrona. Logo, a agenda ótima proporciona 700ms de comunicação assíncrona ininterrupta por ciclo, 22.4% a mais que os 570ms do DeltaV.

Novamente, o que mais chama a atenção são os ganhos com o critério de atrasos de controle. O algoritmo de escalonamento do DeltaV 8.4 não deu prioridade às comunicações provenientes de outras redes, mas que tinha valores importantes de saída de PID. O sistema também não permitiu a mudança dessas prioridades. O resultado foi um atraso baseado na Eq. 3.21 cerca de 80% pior do que os obtidos na otimização. Na malha de controle 3, também melhoras relevantes de cerca de 40% foram alcançadas, principalmente pela estratégia de determinar tempo inicial zero para os AIs que é feita no DeltaV.

O tempo final novamente não chega a ser um problema em nenhum dos casos, mas a otimização também melhora esse aspecto: 29.3%. O gargalo para diminuir o Macrociclo ainda é a Separação, e a melhoria de 30.3% nesse aspecto leva a um MMA de 600ms para a agenda ótima, contra os 860ms da agenda do DeltaV. Nesta rede, portanto, seria difícil acrescentar malhas de controle sem uma otimização. Com a

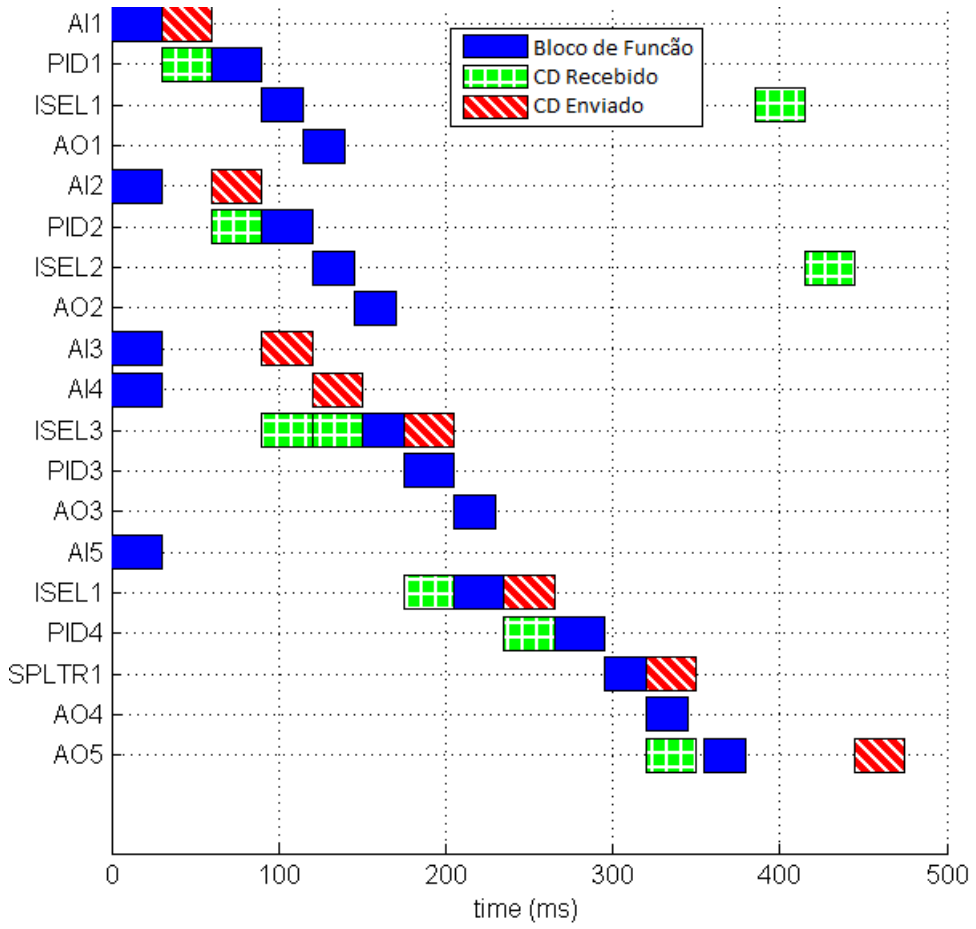

Figura 4.9: Gráfico de Gantt da agenda obtida no DeltaV para o Caso II.

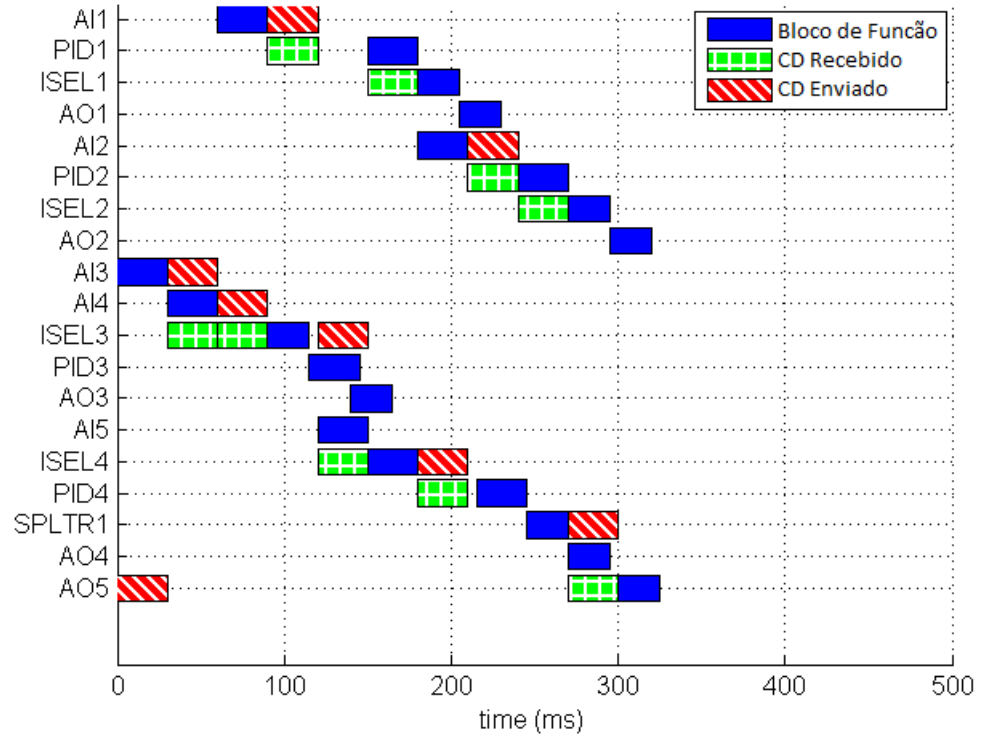

Figura 4.10: Gráfico de Gantt da agenda obtida pela otimização para o Caso II.

agenda obtida pelo DeltaV, a rede já opera próxima ao seu limite com apenas quatro malhas de controle de ciclo igual a 1000ms.

Tabela 4.8: Caso II: Comparação de resultados entre a agenda do DeltaV e a ótima.

Critério		DeltaV	Otimização	Redução (%)
Separação (Eq. 3.19) (ms)		430	300	30.3
Atrasos (Eq. 3.21) (ms)	Malha 1	750	150	80.0
	Malha 2	780	120	84.6
	Malha 3	760	435	42.8
Tempo Final (Eq. 3.24) (ms)		460	325	29.3
MMA (Eq. 4.1) (ms)		860	600	30.2

4.3 Discussões

Os estudos de caso abordados neste capítulo comprovam a eficiência da técnica apresentada neste trabalho. Ambos os estudos de caso são realistas, tratando de redes FF comparáveis às utilizadas industrialmente em termos de número de dispositivos e malhas de controle. Em ambos os casos, os modelos propostos obtiveram resultados ótimos provados, ou seja, após o *solver* convergir os limites superior e inferior, em tempos sempre inferiores a 30s. As tabelas de escalonamento resultantes foram superiores às obtidas no sistema DeltaV 8.3 em todos os objetivos abordados, com melhorias variando entre 15% até 30% em termos de agrupamento de CDs e até 80% para o critério de atraso de malha. A Tabela 4.9 sumariza as melhorias obtidas pela otimização frente às agendas obtidas no DeltaV 8.3, além das estatísticas de solução.

Esses excelentes resultados obtidos com a otimização do escalonamento das tarefas indicam um potencial para melhoria do desempenho das malhas de controle. Não só os atrasos internos induzidos pela rede puderam ser mitigados para níveis próximos do mínimo (i.e., se a rede fosse dedicada a uma malha específica), como também nos dois casos chegou-se à conclusão que o ciclo das malhas de controle poderia ser significativamente diminuído – até pela metade no Caso I – sem comprometer a qualidade de serviço. Outra alternativa, se não houver ganho relevante ao se diminuir o Macrociclo, seria aumentar o número de dispositivos e de malhas de controle na rede. Os resultados obtidos indicam que as recomendações em [15] podem estar sendo conservadoras por um fator de dois, isto é, seria possível configurar o dobro de malhas de controle por rede comparado ao recomendado nas diretrizes de projeto. Para tal, é necessária uma rotina de escalonamento que procure otimi-

Tabela 4.9: Resultados comparativos do problema Monociclo e estatísticas de solução.

Estudo de Caso I				
Critério		DeltaV	Otimização	Redução (%)
Separação (Eq. 3.19) (ms)		295	240	18.6
Atrasos (Eq. 3.21) (ms)	Malha 1	55	55	0
	Malha 2	170	120	29.4
	Malha 3	380	250	34.2
	Malha 4	205	60	70.7
Tempo Final (Eq. 3.24) (ms)		320	265	17.2
MMA (Eq. 4.1) (ms)		590	480	18.6
Estatísticas de Otimização		Pontos de Evento		Sobreposição
Tempo de Solução		28.6s		5.0s
Gap de Integralidade Final		0% (ótimo)		0% (ótimo)

Estudo de Caso II				
Critério		DeltaV	Otimização	Redução (%)
Separação (Eq. 3.19) (ms)		430	300	30.3
Atrasos (Eq. 3.21) (ms)	Malha 1	750	150	80.0
	Malha 2	780	120	84.6
	Malha 3	760	435	42.8
Tempo Final (Eq. 3.24) (ms)		460	325	29.3
MMA (Eq. 4.1) (ms)		860	600	30.2
Estatísticas de Otimização		Pontos de Evento		Sobreposição
Tempo de Solução		14.5s		1.0s
Gap de Integralidade Final		0% (ótimo)		0% (ótimo)

zar certas variáveis como a aqui proposta, em vez de buscar apenas uma tabela de escalonamento viável.

Os resultados deste capítulo revelaram um ponto interessante de comparação no que diz respeito aos modelos formulados. Trata-se da relevante diferença de tamanho e desempenho entre os modelos por Pontos de Evento e Sobreposição. O último teve um desempenho muito superior ao primeiro, com tempos de 5s e 1s para os casos aqui estudados, e uma avaliação do porquê se faz necessária.

Ambas as formulações trazem representações binárias de relações de precedência ou ordenação. A formulação por Pontos de Evento indica a posição absoluta de uma tarefa na lista de tarefas do seu dispositivo: se a tarefa tem a binária w ativada no ponto de evento número 3, isso quer dizer que ela é a terceira a ser executada em seu

dispositivo. Isso gera uma matriz de permutação para $w_{i,j,n}$, fixando-se o dispositivo j, uma vez que cada tarefa é executada uma e apenas uma vez. A formulação por Sobreposição, neste problema FF, relaciona apenas as precedências par a par, sem indicar uma posição absoluta. Ela aproveita explicitamente uma simetria do problema: se A precede B, então B sucede A. Isso leva a um corte imediato de metade do problema e, da matriz $o_{j,i,i'}$, fixando-se o dispositivo j, considera-se apenas uma parte triangular na otimização. Essa diferença de abordagem acaba gerando um modelo mais eficiente computacionalmente.

Outro motivo que pode ser relevante é o fato de que o problema FF trata basicamente de relações de precedência. Essa relação é tratada muito mais diretamente com a variável de sobreposição: basta que uma variável mude entre um e zero. Diferentemente, na formulação por Pontos de Evento, uma relação de precedência exige um conjunto de fórmulas. Se A precede B, compara-se toda a gama de posições relativas entre A e B: Se A é o primeiro, B é pelo menos segundo; se A é o segundo, B é pelo menos o terceiro, etc. Ou seja, N equações envolvendo somatórios de binárias para uma relação de precedência. Isso faz o número de restrições crescer sem que aumente a justeza do modelo. O problema fica com muitas restrições e até a relaxação LP fica prejudicada devido à densidade do problema.

A principal restrição que carrega esse problema é a definida na Eq. 3.4. Observe que ela vale para todo par de tarefas i, i' e todos os pontos de evento, exceto o último. Ou seja, para cada dispositivo j, há $|I_j|(|I_j|-1)^2$ equações – o número de restrições cresce com o cubo do número de tarefas. A restrição de sequenciamento equivalente na formulação por Sobreposição é a Eq. 3.15, que compara pares de tarefas entre si: $|I_j|(|I_j|-1)$ equações, com o número de restrições crescendo com o quadrado do número de tarefas apenas.

Com tudo isso, há um forte indicativo de que a formulação por Sobreposição é mais apropriada para este problema de escalonamento FF do que a formulação por Pontos de Evento. No que tange aos resultados, ambos são equivalentes e tiveram os mesmos nos estudos de caso deste Capítulo. No entanto, a diferença de desempenho foi da ordem de 10 vezes.

Capítulo 5

Escalonamento Ótimo de Tarefas para FOUNDATION™ Fieldbus - Caso Multiciclo

Até agora o problema de escalonamento ótimo de tarefas para uma rede FF H1 foi resolvido com sucesso, mas considerando um fator muito restritivo: todos os dispositivos da rede têm o mesmo ciclo de execução. Por mais que essa seja a realidade para muitos sistemas de controle comerciais, essa restrição dificulta o bom aproveitamento das redes. Se o modelo do Escalonador Ativo de Enlaces (LAS) ou o sistema de controle de uma planta impõem essa restrição, duas situações podem acontecer:

- Malhas de controle que poderiam ter ciclos longos de execução são configuradas com ciclos rápidos. Afinal, todas as malhas de controle deveriam ter o mesmo ciclo de execução que a malha de controle que exigisse o menor ciclo. Por exemplo, se uma malha de controle na rede exigir um ciclo de 500 ms, por controlar uma variável que se altera rapidamente, como vazão, todas as outras malhas de controle da rede deverão ter esse ciclo, mesmo que pudesse ter ciclos de 2, 5 ou 10 segundos. Naturalmente, quanto menores os períodos, maior o tempo em que a rede fica ocupada com comunicações cíclicas, o que limita o número de dispositivos na rede.

- O projeto das redes H1 pode levar em conta o critério de tentar colocar na mesma rede dispositivos de malhas de controle com o mesmo tempo de execução. Isso pode ter pouco valor para plantas com um número relativamente baixo de malhas de controle e pode complicar muito o projeto para grandes plantas com muitas malhas de controle. Além disso, aspectos geográficos podem ser impeditivos na hora de se agruparem malhas de controle de tempos de ciclo em comum, mas com dispositivos muito distantes entre si fisicamente.

O protocolo FOUNDATION™ Fieldbus prevê múltiplos ciclos numa mesma rede. Isso é uma característica muito importante da tecnologia, pois o fato de dispositivos compartilharem um meio de comunicação não deve impor limitações às particularidades de suas tarefas, especialmente o que concerne a controle. Afinal, este é o propósito principal dos dispositivos de campo. Para a configuração de uma rede multiciclo, basta ao usuário determinar ciclos distintos para malhas de controle diferentes, desde que o SDCD permita.

O caso aqui chamado de Multiciclo é exatamente o caso em que dispositivos executam tarefas com periodicidades distintas entre si. A descrição do problema é a mesma do caso anterior, encontrado na Seção 3.1, com algumas inclusões. Nesse caso, cada dispositivo tem um ciclo de execução, normalmente relacionado à malha de controle a que ele está associado. Como cada dispositivo possui a informação do instante em que ele pode enviar uma dada mensagem dentro de seu ciclo, todos os CDs são periódicos e com os mesmos períodos do dispositivo que os envia. Portanto, vários CDs na mesma rede podem ter ciclos diferentes entre si. O LAS recebe uma agenda referente ao macrociclo, que é o mínimo múltiplo comum (MMC) entre todos os ciclos da rede. Esse macrociclo, consequentemente, é o menor tempo em que se repetem todas as tarefas durante a operação. O LAS então envia, para cada dispositivo, a periodicidade de execução de suas tarefas, bem como o instante dentro do ciclo em que elas devem ser executadas e as mensagens enviadas.

O problema de escalonamento para uma rede FF H1 no caso multiciclo é sumarizado a seguir. Dados:

- Uma lista de dispositivos na rede;
- As malhas de controle configuradas tais que pelo menos uma função seja executada em algum dispositivo da rede;
- O tempo de execução de cada tarefa;
- O tempo de execução configurado para o comunicação publicante/assinante;
- A duração do ciclo de cada dispositivo;

o algoritmo de escalonamento deve determinar os instantes de execução de cada função e CD considerando que cada dispositivo (incluindo a rede) só pode realizar uma tarefa de cada vez. A etapa seguinte consiste na descrição matemática do problema.

Na seção a seguir (Seção 5.1), há uma discussão dos critérios de agrupamento e atraso de controle para o caso multiciclo. Na seguinte, Seção 5.2, discute-se a usabilidade dos modelos monociclo para o caso multiciclo. Na Seção 5.3, apresenta-se um modelo MILP que é a extensão da formulação da Seção 3.2, baseada em intervalos de

tempo. Na seção 5.4, a extensão do modelo MILP baseado na formulação por sobreposição da Seção 3.3 é enunciada e explicada. Por fim, na Seção 5.5, uma formulação completa com diversos cortes para melhorar o tempo de solução é apresentada.

5.1 Reestruturação dos Critérios de Otimização

Independente se há ciclos distintos na rede ou não, os objetivos para aumentar a eficiência da rede FF H1 através da otimização do escalonamento de tarefas são os mesmos. Por outro lado, esse caso geral de múltiplos ciclos leva a algumas dificuldades na formulação dos critérios de otimização. Neste cenário, encontra-se um macrociclo composto de diversos ciclos de malhas de controle diferentes e que podem ocasionalmente trocar dados entre si. Desta vez, encontrar uma agenda em que todas os CDs estejam agrupados num único aglomerado de comunicações não é geralmente viável: a mesma comunicação repetir-se-á diversas vezes num macrociclo, e haverá um grande intervalo – o próprio ciclo – entre duas execuções do mesmo CD. Entre CDs de ciclos distintos também é natural que ocorram lacunas em vários pares de ciclo. Além disso, uma malha de controle pode ter várias submalhas com ciclos distintos. Seria o caso de certas malhas de controle em cascata. Neste caso, como considerar o atraso?

Por mais que não seja possível agrupar todas as comunicações e nem seja interessante tentar minimizar o intervalo entre o primeiro e o último CDs, deve-se haver um critério que busque agrupar ao máximo CDs entre si. Ao passo que agrupar CDs no caso monociclo significa tentar colocar todos os CDs consecutivos num único grupo, no caso multiciclo pode haver um número desconhecido de grupos de CDs, sendo que em cada grupo as tarefas estão consecutivas umas às outras.

O que é proposto nesse trabalho é a contagem de lacunas entre CDs. Há uma lacuna entre dois CDs sempre que o tempo inicial do segundo não é igual ao tempo final do primeiro. Feita essa contagem, o objetivo passa a ser minimizar o número de lacunas.

No caso do critério de minimização de atraso nas malhas de controle, nada muda para malhas de controle cujas tarefas tenham todas o mesmo ciclo de execução: bastaria considerar o mesmo critério da Equação 3.22 para qualquer um dos ciclos dessa malha de controle. O problema é que, no caso de malhas de controle com mais de um ciclo de execução para suas tarefas, não se sabe de antemão em qual dos ciclos de cada tarefa deve-se considerar o tempo. Um exemplo simples pode ajudar a esclarecer isso. Suponha que uma malha de controle contenha duas tarefas, A e B, e um CD de A para B, chamado CD1. A e CD1 têm ciclo de execução de 100 ms e tempos de execução de 30 ms, ambas. B também tem tempo de execução de 30 ms, mas seu ciclo é de 50 ms. A e B estão em dispositivos distintos. Uma boa solução

seria $Ts_A = 0, Ts_{CD1} = 30$ e $Ts_{B,2} = 60$ (o segundo item do subscrito significa que é a segunda execução no macrociclo). Isso implica que $Ts_{B,1} = 60 - 50 = 10$. A figura 5.1 ilustra o Gráfico de Gantt relacionado. Repare que a base para considerar o atraso seria o segundo ciclo para a tarefa B. Neste caso de fato não haveria de ser diferente, pelo curto tamanho do ciclo de B, mas pode haver casos em que considerar outros ciclos que não o primeiro não seja necessário, mas seja bom para a otimização.

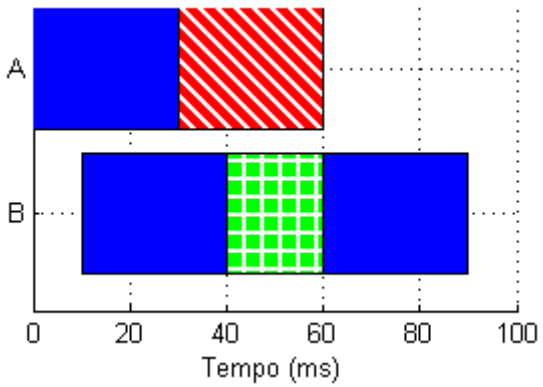

Figura 5.1: Gráfico de Gantt para um exemplo simples de malha de controle com múltiplos ciclos.

Mesmo que os ciclos de execução de todas as tarefas dentro de uma malha de controle em não sejam múltiplos entre si, ou harmônicos, haverá um momento dentro do macrociclo em que eles se sincronizam de forma que a malha de controle fica na sequência correta. O ponto-chave está em descobrir qual é o ciclo de cada tarefa que deve ser considerado. O que se propõe aqui é o uso de tempos base de início e fim de execução, $\overline{Ts_i}$ e $\overline{Tf_i}$. Esses tempos base são os tempos que se referem ao ciclo de execução em que a tarefa está sequenciada com a malha. O tempo base de uma tarefa pode não ser conhecido *a priori*. O ciclo base de cada tarefa também se torna uma variável, a ser descoberta durante a otimização. A partir dos tempos base, os tempos de todos os demais ciclos serão copiados, com o deslocamento devido ao período do ciclo, para os demais ciclos. Mas na fórmula do atraso, entram os tempos base apenas. Tendo em vista a repetição pelos ciclos, toda submalha com período de execução igual para suas tarefas terá preservado o atraso dos tempos base. Além disso, todo par de ciclos em que duas submalhas de ciclos distintos se "encontram", o atraso entre elas será mínimo.

Ilustrando esse conceito, na solução representada pela Figura 5.1 o ciclo base da tarefa B é o segundo ciclo, pois é o ciclo em que B fica idealmente sequenciado após a tarefa A e seu CD. Seu tempo base, portanto, é 60 ms. Os tempos base de A e CD são respectivamente 0 ms e 30 ms, por ser a única execução de ambos. O atraso é nulo, pois as tarefas estão sequenciadas consecutivamente. O tempo da primeira

execução de B é deduzido de seu tempo base: $Ts_{B,1} = \overline{Ts}_B - H_B$, onde $H_B = 50$ ms é o ciclo de B.

5.2 Adaptação do Modelo Anterior

Antes de buscar uma nova modelagem, pode ser útil avaliar se o modelo monociclo pode ser estendido para o multiciclo sem a necessidade de uma nova formulação. O caso geral em que os modelos monociclo geram tabelas de escalonamento viáveis para o caso multiciclo é o caso em que os ciclos de execução são harmônicos entre si. Em outras palavras, se o Máximo Divisor Comum (MDC) entre todos os ciclos for o menor ciclo da rede. Um exemplo é o de ciclos de 250, 500, 1000, 2000 ms, e assim por diante. Para este caso particular, o escalonamento é viável se todas as tarefas da rede puderem ser executadas dentro do menor ciclo.

Segue a análise que permite chegar a essa conclusão. Suponha uma rede com $|I|$ tarefas, sendo que algumas são executadas na periodicidade T, outras na periodicidade K_1T, outras na periodicidade K_1K_2T e assim por diante, sendo $K_1, K_2, ...$ inteiros. Suponha também que todas as tarefas podem ser programadas entre o instante 0 e o instante T dentro do macrociclo $H = T\prod_k K_k$. Isso quer dizer que as relações de precedência e exclusão (para o mesmo dispositivo) não são violadas no período.

As relações de precedência não são violadas nos períodos seguintes. Se as tarefas tiverem a mesma periodicidade, isso é trivial. Se tiverem periodicidades distintas, como nos casos comentados na seção anterior, então arbitrariamente existe um período de cada em que essa precedência deve ser satisfeita. Por exemplo, se a tarefa A é executada com periodicidade K_1T e a B, que se sucede a A, tem periodicidade T, existe pelo menos um $k \in \{1, 2, ..., K_1\}$ tal que a execução A ocorra antes da k-ésima de B, que é entre o instante $(k-1)T$ e kT. Para o caso aqui avaliado, esse k necessariamente seria 0. Os modelos monociclo encontrariam uma agenda para este caso, e a relação de precedência repetir-se-ia pelo menos para os ciclos seguintes de A.

As relações de exclusão também não são violadas nos períodos seguintes. Sejam as tarefas A e B com os mesmos ciclos do parágrafo acima, mas agora pertencendo ao mesmo dispositivo. Se na primeira execução de cada, entre 0 e T, não for conflitante, isso quer dizer que $Ts_{A,1} \geq Tf_{B,1}$ ou que $Ts_{B,1} \geq Tf_{A,1}$. Como todos esses valores são inferiores a T, as execuções seguintes de cada tarefa não entram em conflito com essas: certamente $Ts_{A,1} + T \geq T \geq Tf_{B,1}$ e $Ts_{B,1} + T \geq T \geq Tf_{A,1}$. Além disso, o próximo par de ciclos que essas tarefas poderiam "colidir" seria o período entre K_1T e K_1T+T, em que ocorreria a segunda execução de A e a (K_1+1)-ésima de B. Nesse caso, tem-se $Ts_{A,2} = Ts_{A,1} + K_1T$, $Tf_{A,2} = Tf_{A,1} + K_1T$, $Ts_{B,K_1+1} = Ts_{B,1} + K_1T$

e $Tf_{B,K_1+1} = Tf_{B,1} + K_1T$. Ou seja, os tempos são iguais aos iniciais, deslocados de K_1T. Portanto, as diferenças são mantidas e as relações de exclusão preservam-se.

A análise acima apenas garante que a agenda formada pelos modelos do Capítulo 3 gerariam soluções viáveis para o caso de ciclos harmônicos, mas nada diz sobre otimização. As lacunas entre CDs que surgem nos ciclos seguintes aos primeiros não são explicitamente avaliadas na modelagem do caso monociclo. Portanto, é de se esperar que os resultados obtidos tenham o número mínimo de lacunas se considerado apenas o primeiro ciclo de cada dispositivo, mas os demais ciclos podem ter várias lacunas entre seus CDs. Da mesma forma, considerando o critério de atraso, os tempos base são arbitrados como os primeiros, o que pode não ser o ótimo para malhas de controle com múltiplos ciclos internos. Mesmo assim, espera-se que no caso de ciclos harmônicos essa questão seja menos relevante, pois é um caso mais simples.

5.3 Modelo MILP com Representação Temporal por Pontos de Evento

No capítulo anterior, observou-se que a formulação por sobreposição era mais eficiente computacionalmente, com menor tempo para encontrar o ótimo e menos nós buscados. Mesmo assim, optou-se por tentar novamente as duas formulações e compará-las. O principal motivo é a mudança nos objetivos. A determinação das lacunas comentadas na seção anterior é facilitada na formulação por intervalos de tempo, pois nesta existe a informação da posição absoluta das tarefas. Assim, é direto determinar os pares de CDs consecutivos – que é onde se procura uma lacuna.

As mesmas considerações sobre *readback* feitas no Capítulo 3 são válidas neste capítulo, com a exceção de que o que antes valia para o macrociclo agora vale para o ciclo do CD em questão. Nesse caso usam-se novamente os tempos e ciclos base, já explicados na seção anterior.

Diferente do caso monociclo, as tarefas agora podem ter múltiplas execuções num macrociclo. O número de execuções de uma tarefa é dado pela razão entre o macrociclo e seu ciclo. O tempo de ciclo de um bloco de função é dado pelo tempo de ciclo do dispositivo que o executa e o de um CD, pelo tempo de ciclo do dispositivo que o envia. Sendo c o índice da c-ésima execução da tarefa i no macrociclo, o par (i, c) define unicamente uma execução de tarefa.

O modelo é enunciado abaixo. Para conjuntos, parâmetros e variáveis, referir-se à seção de Nomenclatura do início deste documento. Para código em GMPL, ver Seção B.2.1.

5.3.1 Conjuntos e Parâmetros Pré-Determinados

Os seguintes conjuntos e parâmetros são funções dos demais:

$C_i = 1, 2, ..., \frac{H}{H_i}, i \in I$ como o conjunto ordenado de execuções de i.O número de execuções da tarefa i é a razão entre o tempo total de Macrociclo H e o ciclo H_i da tarefa i.

$N_j = 1, 2, ..., \sum_{i \in I_j} |C_i|, j \in J$. Como neste caso cada tarefa tem $|C_i|$ execuções, o somatório conta o número total de execuções no dispositivo j.

$H = \text{MMC}_{i \in I}(H_i)$. O tempo total do Macrociclo H é o MMC entre os ciclos da rede.

5.3.2 Restrições

Nesta subseção são definidas e explicadas as restrições que descrevem o problema de escalonamento.

Limites das Variáveis

Esses limites garantem que os tempos base inicial e final de cada tarefa estejam dentro do macrociclo.

$$0 \leq \overline{Ts}_i, \overline{Tf}_i \leq H, \forall i \in I. \tag{5.1}$$

Esses limites garantem que o início e o término da c-ésima execução da tarefa i ocorram dentro dos limites do c-ésimo ciclo.

$$H_i(c-1) \leq Ts_{i,c}, Tf_{i,c} \leq H_i c, \forall i \in I. \tag{5.2}$$

Restrições de Uso

Esta inequação significa que, para um dado ponto de evento n, um dispositivo pode executar exatamente uma única tarefa uma única vez. Essa equação é equivalente à Eq. 3.2.

$$\sum_{i \in I_j, c \in C_i} w_{i,c,j,n} = 1, \forall j \in J, n \in N_j. \tag{5.3}$$

Restrições de Execução

Cada execução da tarefa deve ser escalonada uma e apenas uma vez durante o macrociclo. Estende a Equação 3.3.

$$\sum_{n \in N_j} w_{i,c,j,n} = 1, \forall j \in J, i \in I_j, c \in C_i. \tag{5.4}$$

Restrições de Sequenciamento

Para um dispositivo j e para tarefas que ocorram em intervalos de tempo consecutivos, o tempo final da c'-ésima execução da próxima tarefa i' deve ser maior ou igual que o tempo final da c-ésima execução da tarefa atual i. É uma extensão da Equação 3.4. Observe que a condição ($i \neq i'$ ou $c \neq c'$) representa a diferença entre a execução (i, c) e a execução (i', c').

$$Ts_{i',c'} \geq Tf_{i,c} - H\left(2 - w_{i,c,j,n} - w_{i',c',j,n+1}\right),$$
$$\forall j \in J, i, i' \in I_j, c \in C_i, c' \in C_{i'}, n \in N_j : n < |N_j|, (i \neq i' \text{ ou } c \neq c'). \tag{5.5}$$

Se um dispositivo i' vem depois de outro, i, numa malha de controle m, então o tempo base inicial de i' deve ser maior ou igual ao tempo base final de i. Trata-se de uma extensão da Eq. 3.5. Observe que aqui usam-se os tempos base, como discutido na seção anterior. Esses tempos serão idênticos aos de alguma execução das tarefas correspondentes.

$$\overline{Ts}_{i'} \geq \overline{Tf}_{i}, \forall m \in M, (i, i') \in L_m. \tag{5.6}$$

Se a tarefa i precede i', ambas são executadas no mesmo dispositivo j e as tarefas estão numa mesma malha de controle m, então i' deve ser alocada num ponto de evento após aquele em que i está alocada. Essa restrição constitui um corte e é equivalente à Eq. 3.6. No entanto, aqui ela só vale para tarefas que executam o mesmo número de vezes e, portanto, têm o mesmo tempo de ciclo.

$$\sum_{n' \in N_j : n' > n} w_{i',c,j,n'} \geq w_{i,c,j,n},$$
$$\forall j \in J, m \in M, (i, i') \in L^1_{j,m}, c \in C_i, n \in N_j : |C_i| = |C_{i'}|. \tag{5.7}$$

Esse corte limita o conjunto de intervalos de tempo de uma dada execução de tarefa. Existem execuções de tarefas que certamente serão antes e outras que serão depois de uma dada execução.

$$\sum_{|N_j| - |Ia_{i,c}| \geq n > |Ib_{i,c}|} w_{i,c,j,n} = 1,$$
$$\forall j \in J, i \in I_j, c \in C_i, n \in N_j. \tag{5.8}$$

O conjunto $Ia_{i,c}$ é o conjunto de execuções de tarefas que certamente são exe-

cutadas após (i, c). Trata-se de todas as execuções cujo limite inferior do tempo de início é maior ou igual ao limite superior do tempo final de (i, c):

$$Ia_{i,c} := \{i' \in I, c' \in C'_i : (i \neq i' \text{ ou } c \neq c'), H_i c \leq H_{i'}(c' - 1)\}. \tag{5.9}$$

Por sua vez, $Ib_{i,c}$ é o conjunto das execuções que são antes de de (i, c). Analogamente,

$$Ib_{i,c} := \{i' \in I, c' \in C'_i : (i \neq i' \text{ ou } c \neq c'), H_i(c - 1) \geq H_{i'} c'\}. \tag{5.10}$$

Como existem pelo menos $|Ib_{i,c}|$ execuções antes de (i, c), o primeiro ponto de evento que (i, c) poderia ser alocado seria $|Ib_{i,c}|+1$. Da mesma forma, como existem pelo menos $|Ia_{i,c}|$ após (i, c), os $|N_j| - |Ia_{i,c}| + 1$ intervalos finais não podem ser ocupados por (i, c), conforme imposto pela Eq.5.8.

Restrições de Tempo Final

As duas equações a seguir são meras extensões da Eq. 3.7, válidas para cada execução bem como para os tempos base.

$$Tf_{i,c} = Ts_{i,c} + Te_i, \forall i \in I \tag{5.11}$$

$$\overline{Tf}_i = \overline{Ts}_i + Te_i, \forall i \in I \tag{5.12}$$

A restrição de tempo final vem a seguir. Optou-se aqui por deixar a restrição de tempo final para as primeiras execuções apenas, uma vez que sua função é apenas de antecipar ao máximo as tarefas. As outras execuções terão seu tempo inicial reduzido por consequência das restrições de periodicidade (ver Eq. 5.19).

$$TF \geq Tf_{i,1}, \forall i \in I. \tag{5.13}$$

Restrições de *Readback*

Essas equações são extensões das Eqs. 3.13 e 3.14, usando os tempos base.

$$\overline{Ts}_{i_R} \geq \overline{Tf}_r - Hy_r, \forall r \in RB, (i_R, i_S) \in LR_r. \tag{5.14}$$

$$\overline{Ts}_r \geq \overline{Tf}_{i_S} - H(1 - y_r), \forall r \in RB, (i_R, i_S) \in LR_r. \tag{5.15}$$

As equações seguintes concernem os ciclos anterior (Eq. 5.16) e seguinte (Eq. 5.17). A Equação 5.16 garante que, se $y_r = 0$, o CD fique após o término da execução anterior de i_S, a tarefa que o envia. Se não houver execução anterior no macrociclo, o motivo está no fato de que esta se trata da primeira, fazendo com que $\overline{Tf}_{i_S} - H_{i_S} \leq 0$ e a equação seja trivialmente satisfeita. De forma equivalente,

a Equação 5.17 entra em vigor no caso em que $y_r = 1$, exigindo que o CD fique antes do início da execução da tarefa que o receberá, i_R, no próximo ciclo desta. Se não houver próxima execução, trata-se da última no macrociclo, resultando em $\overline{Ts}_{i_B} + H_{i_B} \geq H$ e, novamente, numa aceitação trivial da inequação.

$$\overline{Ts}_r \geq \overline{Tf}_{i_S} - H_{i_S} - Hy_r, \forall r \in RB, (i_R, i_S) \in LR_r. \tag{5.16}$$

$$\overline{Ts}_{i_B} + H_{i_B} \geq \overline{Tf}_r - H(1 - y_r), \forall r \in RB, (i_R, i_S) \in LR_r. \tag{5.17}$$

Restrições de Ciclo Base

As restrições a seguir referem-se à determinação dos ciclos base de cada tarefa.

$$H_i(\overline{c}_i - 1) \leq \overline{Ts}_i \leq H_i\overline{c}_i, \forall i \in I.$$

Observe que essa equação é muito semelhante à Eq. 5.2. No entanto, dessa vez os limites são dados pelas variáveis $\overline{c}_i$. Dependendo de como forem determinados $\overline{Ts}_i$ ou $\overline{Tf}_i$, haverá um único $\overline{c}_i$ correspondente. A equação pode ser reescrita para apenas o argumento central conter variáveis:

$$-H_i \leq \overline{Ts}_i - H_i\overline{c}_i \leq 0, \forall i \in I. \tag{5.18}$$

Restrições de Periodicidade

Uma restrição para garantir a periodicidade das tarefas seria:

$$Ts_{i,c+1} = Ts_{i,c} + H_i, \forall i \in I, c \in C_i : c < |C_i|.$$

Para cada execução, atrela-se seu instante inicial ao da execução anterior, somado ao seu ciclo. Contudo, a restrição abaixo a substitui e considera o ciclo base:

$$Ts_{i,c+1} = \overline{Ts}_i + H_i(c - \overline{c}_i), \forall i \in I, c \in C_i. \tag{5.19}$$

A equação acima copia o tempo base para os tempos reais, deslocando conformemente por $c - \overline{c}_i$ tempos de ciclo.

Restrições de Lacunas de Comunicação

Essa inequação tem formato semelhante à Eq. 5.5, mas com os tempos trocados, e restringe a solução somente aos casos em que $w_{net,i,c,n} = w_{net,i',c',n+1} = 1$ e $g_n = 0$, quando se torna $Tf_{i,c} \geq Ts_{i',c'}$. Nos outros casos, é trivialmente satisfeita. No entanto, se $w_{net,i,c,n} = w_{net,i',c',n+1} = 1$, então essas execuções são consecutivas: (i', c') é a execução que vem após (i, c) no dispositivo j. Isso implica (pela Eq.

5.5) em $Ts_{i',c'} \geq Tf_{i,c}$. Se $g_n = 0$, então $Ts_{i',c'} \geq Tf_{i,c}$ e $Tf_{i,c} \geq Ts_{i',c'}$. Essas duas condições juntas só são possíveis no caso marginal em que $Tf_{i,c} = Ts_{i',c'}$ – justamente quando não há uma lacuna entre as tarefas. Quando $g_n = 1$, é permitido que $Tf_{i,c} < Ts_{i',c'}$.

$$\begin{gathered} Tf_{i,c} \geq Ts_{i',c'} - H\left(2 - w_{net,i,c,n} - w_{net,i',c',n+1} + g_n\right), \\ \forall i, i' \in I_{net}, c \in C_i, c' \in C_{i'}, n \in N_{net} : n < |N_{net}|, (i \neq i' \text{ ou } c \neq c'). \end{gathered} \tag{5.20}$$

Restrições de Equivalência

A equação a seguir é similar à Eq. 3.27.

$$\overline{Ts}_i \geq \overline{Ts}_{i'}, \forall (i, i') \in E. \tag{5.21}$$

Restrição do Critério de Minimização das Lacunas

Trata-se do critério de minimização do número de lacunas. Minimizar G levará à tentativa de zerar os g_n, uma vez que $g_n \geq 0$.

$$G = \sum_{n \in N_{net} : n < |N_{net}|} g_n. \tag{5.22}$$

Restrições do Critério de Atraso

Essa equação é semelhante à Equação 3.22, no entanto para esta formulação compara-se o tempo final da tarefa seguinte com o tempo inicial da tarefa anterior. Então são apenas pesados os intervalos entre execuções, sem haver preocupação com os tempos de execução das tarefas.

$$D_m = \sum_{(i,i') \in L_m} \overline{Ts}_{i'} - \overline{Tf}_i, \forall m \in M. \tag{5.23}$$

A vantagem desta abordagem é que nesse caso um limite inferior é zero:

$$D_m \geq 0, \forall m \in M. \tag{5.24}$$

Este limite inferior vale para uma malha de controle que represente exatamente uma cadeia de tarefas. A temporização das tarefas que leva ao mínimo atraso é quando o tempo inicial da subsequente é igual ao tempo final da anterior. Havendo mais ramos na malha de controle, esse limite pode passar a ser conservador, pois pode não ser possível agrupar todas as tarefas da mesma malha de controle.

5.3.3 Critério de Otimização

Equivalentemente à composição de funções objetivo na Seção 3.4.4, aqui é feita a composição das funções objetivo G, D_m e TF. A diferença é que, enquanto D_m e TF têm valores temporais, G é adimensional, medindo uma quantidade. Propõe-se então um peso temporal a G, chamado γ. A função objetivo é dada por:

$$F = \alpha' \left(0.5\gamma G + 0.5 \sum_{m \in M} a_m D_m \right) + (1 - \alpha')TF, \tag{5.25}$$

onde $0 \leq \alpha' \leq 1$. Novamente, o peso para TF deve ser significativamente menor que os demais, em função da pouca importância relativa no objetivo final. O peso γ é um valor temporal que representa o compromisso entre os critérios de minimização de atrasos e de lacunas. Esse peso significa que a otimização busca diminuir, se possível, o número de lacunas sempre que isso não gere a contrapartida de aumentar em mais de γms o critério de atraso para cada lacuna. Por exemplo, para $\gamma = 50$ms, uma solução com uma lacuna a menos terá o valor da função objetivo inferior a uma solução com essa lacuna extra, porém com o critério de atraso somando até 50ms mais. Essa análise depende também dos pesos a_m, mas pode servir como um guia para a escolha dos pesos.

5.4 Modelo MILP com Representação Temporal por Sobreposição - Versão Básica

Nesta seção é apresentado, como alternativa ao modelo da seção anterior, um modelo com representação temporal por sobreposição. Trata-se então de uma extensão do modelo da Seção 3.3. Os conceitos de tempos base e ciclos base ainda se aplicam. O modelo resultante guarda muita semelhança com o da seção anterior exceto pela determinação das lacunas, além, é claro, da representação de ordenação.

Como as variáveis binárias de sobreposição não fornecem informação sobre ordem absoluta, não é possível à primeira vista uma variável g_n como a da seção anterior. Foram pensadas duas maneiras de contornar isso neste trabalho:

- Retirar a informação de tarefas consecutivas a partir das binárias de sobreposição. Percebeu-se que isso é possível sem se criarem variáveis inteiras adicionais.
- Calcular a lacuna entre cada par de CDs. É possível descontar as lacunas referentes a CDs não consecutivos. Essa abordagem se mostrou mais eficiente computacionalmente, e será a relatada aqui.

Dito isso, enuncia-se o modelo abaixo. A função objetivo é a mesma da modelagem anterior, avaliada através da Eq. 5.25. O código GMPL encontra-se na Seção

B.2.1

5.4.1 Parâmetros Pré-Definidos

Novamente, o horizonte de tempo é o Macrociclo, definido por $H = \text{MMC}_{i \in I}(H_i)$.O número total de execuções de CDs é encontrado por essa equação: $Z = \sum_{i \in I_{net}} |C_i|$.

5.4.2 Restrições

Nesta subseção são definidas e explicadas as restrições que descrevem o problema de escalonamento. Às restrições abaixo adicionam-se as Eqs. 5.1, 5.2, 5.6, 5.11, 5.12, 5.13, 5.14, 5.15, 5.16, 5.17, 5.18, 5.19, 5.21, 5.23 e 5.24.

Restrições de Sequenciamento

Para um dispositivo j se $o_{j,i,c,i',c'} = 0$ então a equação se torna $Tf_{i'} \leq Ts_i$, o que implica que (i', c') termina antes de (i, c) começar. Se $o_{j,i,c,i',c'} = 1$, a inequação é trivialmente satisfeita, pois H é o limite superior dos tempos iniciais e finais de todas as tarefas. É uma extensão da Equação 3.15. Observe que a condição $(i \neq i'$ ou $c \neq c')$ representa a diferença entre a execução (i, c) e a execução (i', c').

$$\begin{gathered} Tf_{i',c'} - Ts_{i,c} \leq Ho_{j,i,c,i',c'}, \\ \forall j \in J, i, i' \in I_j, c \in C_i, c' \in C_{i'} : (i \neq i' \text{ ou } c \neq c'). \end{gathered} \tag{5.26}$$

Se a execução (i, c) precede (i', c'), ambas são executadas no mesmo dispositivo j e as tarefas estão numa mesma malha de controle m, então (i', c') deve ser executada após o termino de (i, c). Esta restrição é equivalente à Eq. 5.7, novamente valendo apenas para tarefas de mesmo tempo de ciclo.

$$\begin{gathered} o_{j,i,c,i',c'} = 1, \\ \forall j \in J, m \in M, (i, i') \in L^1_{j,m}, c \in C_i : |C_i| = |C_{i'}|. \end{gathered} \tag{5.27}$$

Restrições de Sobreposição

Essa restrição impede a sobreposição num dispositivo, sendo uma extensão da Eq. 3.17.

$$o_{j,i,c,i',c'} + o_{j,i',c',i,c} = 1, \forall j \in J, i, i' \in I_j, c \in C_i, c' \in C_{i'} : i \neq i' \text{ ou } c \neq c'. \tag{5.28}$$

Restrições de Lacunas de Comunicação

Aqui entra uma grande diferença com relação à seção anterior.

$$Hg_{i,c,i',c'} \geq Ts_{i',c'} - Tf_{i,c},$$
$$\forall i, i' \in I_{net}, c \in C_i, c' \in C_{i'} : i \neq i' \text{ ou } c \neq c'. \tag{5.29}$$

Observe que essa inequação tem formato semelhante à Eq. 5.26, mas com os tempos trocados, de maneira equivalente à Eq. 5.20. Quando $g_{i,c,i',c'} = 0$, temos que $Tf_{i,c} \geq Ts_{i',c'}$. Caso contrário, a inequação é trivialmente satisfeita. Ou seja, aparentemente $g_{i,c,i',c'}$ é um indicativo de quando (i, c) termina antes de (i', c') começar. Mas existe um único caso em que $g_{i,c,i',c'} = g_{i',c',i,c} = 0$: quando não há lacuna entre eles. Por exemplo, se (i, c) é executado antes de (i', c'), naturalmente $Tf_{i',c'} > Ts_{i,c}$ e, logo, $g_{i,c,i',c'}$ pode ter valor nulo. Por outro lado, na ausência de uma lacuna entre eles, $Tf_{i,c} = Ts_{i',c'}$ e $g_{i',c',i,c}$ também pode ser anulado.

Restrição do Critério de Minimização das Lacunas

Opção 1:

$$G = \sum_{(i,i',c,c') \in \mathcal{D}} g_{i,c,i',c'}, \tag{5.30}$$

onde $\mathcal{D} = \{i, i' \in I_{net}, c \in C_i, c' \in C_{i'} : i \neq i' \text{ ou } c \neq c'\}$.

Esse critério serve minimiza o número de lacunas, mas inclui uma série de lacunas indesejáveis na conta: as lacunas entre tarefas não consecutivas. É possível retirar essas lacunas do somatório com uma conta simples. Seja Z o número de execuções de CD no macrociclo (ver Subseção 5.4.1). Existem $Z-1$ pares de CDs consecutivos. O número total de pares de execuções de CDs, sem considerar ordem, é de $\binom{Z}{2} = Z(Z-1)/2$. O número de pares não-consecutivos é, portanto, de $Z(Z-1)/2 - (Z-1) = (Z-1)/(Z-2)/2$. Para esses pares $(i, c), (i', c')$, certamente haverá um $g_{i,c,i',c'} = 1$, ao passo que $g_{i',c',i,c}$ pode ser 0. Finalmente, pode-se concluir que o somatório de g para os pares não-consecutivos é igual a $(Z-1)/(Z-2)/2$. Pode-se reescrever portanto a Eq. 5.30 através da Eq. 5.31, com os limites impostos pela Eq. 5.32:

$$G = \sum_{(i,i',c,c') \in \mathcal{D}} g_{i,c,i',c'} - \frac{(Z-1)(Z-2)}{2}. \tag{5.31}$$

$$0 \leq G \leq Z - 1. \tag{5.32}$$

5.5 Modelo MILP com Formulação por Sobreposição - Versão Completa

Existem oportunidades de melhorias em função da estrutura do problema multiciclo, a saber:

- No caso multiciclo, como há múltiplas execuções da mesma tarefa, aumentam o número de variáveis temporais e binárias. No entanto, o número de graus de liberdade temporais não se altera. Uma vez que o instante inicial da primeira execução de uma tarefa é determinado, os instantes de execução dos ciclos seguintes estão atrelados ao primeiro, pelas restrições de periodicidade.
- Existem diversos pares de execuções em que é desnecessário comparar sobreposição. Um caso trivial é o de duas execuções da mesma tarefa, porém isto também ocorre em diversos outros casos. Por exemplo, qualquer execução necessariamente entre 0ms e 250ms nunca pode ter sobreposição com outra que ocorra entre 500ms e 1000ms. Há oportunidade, portanto, para eliminar uma série de variáveis de sobreposição binárias. Raciocínio semelhante se aplica às lacunas, com um cuidado extra: pode não haver lacuna entre tarefas de ciclos adjacentes.
- A periodicidade das tarefas pode se aplicar às sobreposições e lacunas. Se duas execuções têm uma determinada relação de precedência temporal, as mesmas tarefas terão a mesma relação de precedência nos próximos ciclos que se sincronizarem. Por exemplo, a tarefa A, de ciclo 250ms e a B, de 500ms. Se a primeira execução de A ocorre antes da primeira de B, a terceira execução de A, entre 500ms e 750ms, também será antes da segunda de B, entre 500ms e 1000ms. Afinal, ambas serão deslocadas do mesmo valor, igual a 500ms.

Todas essas observações levam a oportunidades de redução e aumento da justeza do modelo, promovendo um potencial ganho na eficiência. Essa seção apresenta um modelo que leva em conta essas observações, sendo chamado de Versão Completa da Formulação por Sobreposição. O modelo é enunciado abaixo, ao passo que o código em GMPL pode ser encontrado na Seção B.2.1

5.5.1 Conjuntos e Parâmetros Novos

Todos os conjuntos e parâmetros usados no modelo da Seção 3.3 são válidos neste modelo. A eles adicionam-se os seguintes parâmetros e conjuntos, que podem ser determinados com base nos dados:

$d_{net,i,i'} = \text{MDC}(|C_i|, |C_i'|), i, i' \in I_net$. Trata-se do MDC entre os números de ciclos de dois CDs, num macrociclo. Esse parâmetro indica o número de sincronizações dentro do Macrociclo para um par de CDs.

$d_m = \text{MDC}_{i \in I_m} |C_i|$, como o MDC entre os números de ciclos das tarefas pertencentes à mesma malha de controle. Esse é o número de repetições de uma configuração temporal das tarefas de uma malha de controle no macrociclo.

P_j é o conjunto de pares de execuções em que é possível haver sobreposição. Além disso, esse conjunto desconsidera sobreposições repetidas em ciclos futuros após múltiplos do MMC entre seus tempos de ciclos. $P_j := \{i, i' \in I_j, c \in C_i, c' \in C_{i'} : i \neq i', H_{i'}c' > H_i(c-1), H_{i'}(c'-1) < H_i c, c \leq |C_i|/d_{j,i,i'}, c' \leq |C_{i'}|/d_{j,i,i'}\}$.

Q é um conjunto de pares de execuções de CDs em que é possível não haver uma lacuna. $Q := \{i, i' \in I_net, c \in C_i, c' \in C_{i'} : i \neq i', H_{i'}c' \geq H_i(c-1), H_{i'}(c'-1) \leq H_i c, c \leq |C_i|/d_{net,i,i'}, c' \leq |C_{i'}|/d_{net,i,i'}\}$.

Observe que aqui há a consideração dos casos $H_{i'}c' = H_i(c-1), H_{i'}(c'-1) = H_i c$, que não eram levados em conta na definição de P_j. Isso ocorre porque, mesmo não sendo possível uma sobreposição, é possível uma sequência imediata. Como em P_j, foram desconsideradas repetições da mesma lacuna, que devem ser contabilizadas na função objetivo.

O conjunto complementar a Q que contém outros pares de execuções de CDs em que é possível haver sequência imediata é Q^*: $Q^* := \{i, i' \in I_net, c \in C_i, c' \in C_{i'} : i \neq i', ((H_{i'}c' = H_i(c-1), c = |C_i|/d_{net,i,i'} + 1, c' = |C_{i'}|/d_{net,i,i'})$ ou $H_{i'}(c'-1) = H_i c, c = |C_i|/d_{net,i,i'}, c' = |C_{i'}|/d_{net,i,i'} + 1\}$. Nesse caso, incluem-se os pares em que uma tarefa está em seu último ciclo dentro dos $|C_i|/d_{net,i,i'}$ primeiros ciclos e a outra tarefa está logo após, no ciclo de número $|C_{i'}|/d_{net,i,i'} + 1$. Outras repetições dessa relação serão levadas em conta na função objetivo.

Os pares de CDs em que certamente existem lacunas é dado por $\overline{Q} := \{i, i' \in I_net, c \in C_i, c' \in C_{i'} : (i = i', c < c')$ ou $H_{i'}(c'-1) > H_i c, c \leq |C_i|/d_{net,i,i'}, c' \leq |C_{i'}|/d_{net,i,i'}\}$.

5.5.2 Mudança nas variáveis

Com a determinação dos conjuntos P_j, Q, Q^* e $\overline{Q}$, torna-se possível diminuir o número de variáveis o e g a composições desses conjuntos:

$$o_{j,i,c,i',c'} \in \{0,1\}, \forall j \in J, \{i, i', c, c'\} \in P_j \tag{5.33}$$

$$g_{i,c,i',c'} \in \{0,1\}, \forall Q \cup Q^* \tag{5.34}$$

5.5.3 Equações e Inequações

Nesta subseção são definidas e explicadas as restrições que descrevem o problema de escalonamento. Às equações listadas abaixo adicionam-se como restrições as Eqs. 5.6, 5.11, 5.12, 5.13, 5.14, 5.15, 5.16, 5.17, 5.18, 5.19, 5.21, 5.32, 5.23, 5.24 e 5.25.

Limites das Variáveis

Essa nova restrição limita a base da tarefa entre o primeiro e o segundo macrociclos da malha de controle em que ela participa. Isso ainda permite malhas de controle que comecem num ciclo e terminem em outro, que é o motivo pelo qual foram propostos os ciclos base.

$$0 \leq \overline{Ts}_i, \overline{Tf}_i \leq \min 2\frac{H}{d_m}, H \forall minM, i \in I_m. \tag{5.35}$$

Essa restrição limita o ciclo base conformemente:

$$1 \leq \overline{c}_i \leq \min 2\frac{|C_i|}{d_m}, |C_i|\} \forall m \in M, i \in I_m. \tag{5.36}$$

Restrições de Sequenciamento

Essa equação é uma adaptação da Eq. 5.26 para os novos conjuntos deste modelo:

$$\begin{gathered} Tf_{i',c'} - Ts_{i,c} \leq (H_{i'}c' - H_i(c-1))o_{j,i,c,i',c'}, \\ \forall j \in J, \{i, i', c, c'\} \in P_j. \end{gathered} \tag{5.37}$$

Da mesma forma, a equação a seguir é uma adaptação da Eq. 5.27.

$$\begin{gathered} o_{j,i,c,i',c'} = 1, \\ \forall j \in J, m \in M, (i, i') \in L^1_{j,m}, c \in C_i : |C_i| = |C_{i'}|, \{i, i', c, c'\} \in P_j. \end{gathered} \tag{5.38}$$

Restrições de Sobreposição

Essa equação é uma adaptação da Eq. 5.28:

$$o_{j,i,c,i',c'} + o_{j,i',c',i,c} = 1, \forall j \in J, \{i, i', c, c\} \in P_j. \tag{5.39}$$

Restrições de Lacunas de Comunicação

Essa equação é uma adaptação da Eq. 5.29:

$$(H_{i'}c' - H_i(c-1))g_{i,c,i',c'} \geq Ts_{i',c'} - Tf_{i,c},$$
$$\forall i, i', c, c' \in Q \cup Q^* \tag{5.40}$$

Restrição do Critério de Minimização das Lacunas

Esta equação constitui uma grande diferença com relação à versão original da contagem de lacunas no modelo por sobreposição. O objetivo é o mesmo: contar o número de lacunas. No entanto, com a mudança nos conjuntos e as considerações efetuadas, essa contagem é feita da seguinte forma:

$$G = \sum_{(i,i',c,c') \in Q} d_{net,i,i'} g_{i,c,i',c'} + \sum_{(i,i',c,c') \in Q^*} (d_{net,i,i'} - 1) g_{i,c,i',c'} + \\ + |\overline{Q}| - \frac{(Z-1)(Z-2)}{2} \tag{5.41}$$

Observe que há quatro parcelas. A última, como na Eq. 5.31, tem a função de desconsiderar todas os pares de tarefas não-consecutivos em que há lacunas. A diferença encontra-se nas três primeiras parcelas.

A primeira leva em conta pares de CDs que pertencem ao conjunto Q, que é o conjunto em que há sobreposição entre os limites de tempo de ambas as tarefas. Como esse conjunto não considera execuções em que as diferenças temporais se repetem, multiplica-se o número de lacunas obtido para cada par em Q pelo número de vezes que esse par ocorre. Por exemplo, os CDs CD1 e CD2 têm duas execuções no macrociclo cada um. Logo, se a primeira execução é em $Ts_{CD1,1} = 30$ e $Ts_{CD2,1} = 70$, a diferença entre o fim de CD1 e o início de CD2 será de 10 ms. Como ambos têm a mesma periodicidade, essa mesma diferença ocorrerá na segunda execução dos mesmos. Mas como existe essa repetição, apenas a primeira execução de CD1 e CD2 é considerada no conjunto Q. É, portanto, necessário multiplicar a booleana que representa a lacuna destes CDs por 2. No caso geral, o fator multiplicativo é $d_{net,i,i'}$.

A segunda parcela trata do conjunto Q^*. São execuções de CD que possuem limites temporais de execução não-sobreponentes, mas imediatos. Existe uma possibilidade marginal de não haver lacuna entre eles, mas que deve ser considerada. Para o mesmo exemplo acima, supondo um macrociclo de 1000 ms: as primeiras execuções compreendem-se entre 0 e 500 ms, enquanto as segundas, entre 500 ms e 1000 ms. Se $Tf_{CD1,1} = 500$ e $Ts_{CD2,2} = 500$, não há lacuna entre esse par de execuções. Neste caso, não há repetição entre esses pares de execução, ao passo que, caso o macrociclo fosse de 1500 ms, essa configuração repetir-se-ia entre $(CD1, 2)$ e

$(CD2, 3)$. Como regra, a configuração se repete $d_{net,i,i'} - 1$ vezes.

A terceira parcela apenas conta o número de elementos do conjunto $\overline{Q}$, uma vez que esse conjunto é o dos pares em que certamente existe lacuna.

Com isso, finalizam-se os enunciados dos modelos matemáticos para o caso multiciclo. No capítulo seguinte, os resultados para a aplicação destes modelos a diferentes estudos de caso serão apresentados, detalhados e avaliados.

Capítulo 6

Aplicação a Sistemas Reais com Múltiplos Ciclos

As formulações apresentadas no capítulo anterior serão avaliadas em quatro estudos de caso neste capítulo. Novamente, os estudos de caso são feitos em redes realistas de tamanho industrial, i.e., com tempos de execução retirados de folhas de dados de dispositivos FF reais, cerca de 10 dispositivos e malhas de controle realistas. Todavia, diferente do Capítulo 4, não há comparação com os resultados de um SDCD real, uma vez que o DeltaV 8.4 não possui como recurso a configuração de malhas de controle com ciclos de execução distintos na mesma rede.

Outra mudança foi realizada para o caso multiciclo: uma vez que os tempos de solução foram muito mais longos aos obtidos no caso monociclo, devido ao aumento da complexidade do problema, optou-se por se utilizar um *solver* mais eficiente que o GLPK. Após a avaliação de benchmarks (disponíveis no website http://plato.asu.edu/ftp/milpc.html), dois *solvers* foram testados: o *Coin-OR Branch & Cut* (CBC), versão 2.7.5, win64, intel11.1; e o *Solving Constraint Integer Programs* (SCIP), versão 2.1.1, win x64, com SoPlex 1.6.0. O SCIP foi mais eficiente em todos os casos testados e foi escolhido para os estudos de caso.

Os estudos de caso foram determinados objetivando ser representativos para diferentes níveis de complexidade que o caso multiciclo pode apresentar: a existência de múltiplos ciclos na mesma malha de controle e a existência de ciclos heterogêneos, ou não-harmônicos. O Caso III é o mais simples, sem nenhuma dessas complexidades adicionais: trata-se das mesmas malhas de controle do Caso I, mas com ciclos distintos entre si: uma malha de controle de 250ms de ciclo, uma de 500ms e duas de 1000ms. O Caso IV apresenta uma malha de controle composta de três submalhas, com tempos de 500, 1000 e 2000ms. O Caso V mistura os problemas anteriores, com a malha de controle composta do Caso IV e duas malhas de controle mais simples do Caso III. Finalmente, o Caso VI apresenta um caso de ciclos heterogêneos. São as mesmas malhas de controle dos Casos I e III, mas com ciclos de 200ms, 400ms e

1000ms.

Para todos os casos, os seguintes pesos da função objetivo foram utilizados: $\alpha' = 0.98$, $\gamma = 50$ms e $d_m = 1, \forall m$.

6.1 Caso III - Três Ciclos Distintos

Este caso possui a mesma rede que a do Caso I. No entanto, é feita uma modificação nos tempos de execução dos dispositivos da rede: a malha de controle 1 neste caso possui ciclo de 500ms e a malha de controle 4, de 250ms. As malhas de controle 2 e 3 continuam com tempos de ciclo de 1000ms. Observe que este primeiro caso do problema multiciclo representa um problema de certa forma simples: cada malha de controle tem apenas um ciclo e, além disso, os ciclos são harmônicos entre si.

Os blocos de função configurados e seus tempos de execução estão descritos na tabela 6.1. Como no capítulo 4, só são aqui mostradas as informações após a Redução do Modelo (ver Sec. 3.5.2. O tempo configurado para o CD é de 30 ms. O macrociclo é o MMC entre 250, 500 e 1000 ms: 1000 ms. Os diagramas de malhas de controle estão detalhados nas figuras 4.1, 4.2 e 4.3 e não serão repetidos aqui.

O código para entrada de dados no GLPK encontra-se na Seção B.2.2.

6.1.1 Representação da Configuração usando o Modelo

Os seguintes conjuntos podem ser determinados:

$$M = \{1, 2, 3, 4\},$$

$$I_1 = \{AI1, CD1, AO1\},$$

$$I_2 = \{AI2, CD2, AI3, CD3, AO2\},$$

$$I_3 = \{AI41, AI42, CD4, CD5, AO3, AO4, CD6, CD7\},$$

$$I_4 = \{AI5, CD8, AO5\}.$$

$$L_1 = \{(AI1, CD1), (CD1, AO1)\},$$

$$L_2 = \{(AI2, CD2), (AI3, CD3), (CD2, AO2), (CD3, AO2)\},$$

$$L_3 = \{(AI4, CD4), (AI4, CD5), (CD4, AO3), (CD5, AO4)\},$$

$$L_4 = \{(AI5, CD8), (CD8, AO5)\}.$$

$$RB = \{CD6, CD7\},$$

$$LR_{CD6} = \{(AI42, AO3)\},$$

Tabela 6.1: Dispositivos, tarefas, tempos de execução e tempos de ciclo para o Caso III.

j	i	Te_i (ms)	H_i (ms)
AI1	AI1	25	500
AI2	AI2	30	1000
AI3	AI3	30	1000
	AI41	45	1000
AI4	AI41	20	1000
AI5	AI5	30	250
AO1	AO1	90	500
AO2	AO2	80	1000
AO3	AO3	40	1000
AO4	AO4	40	1000
AO5	AO5	55	250
	CD1	30	500
	CD2	30	1000
	CD3	30	1000
net	CD4	30	1000
	CD5	30	1000
	CD6	30	1000
	CD7	30	1000
	CD8	30	250

$$LR_{CD7} = \{(AI42, AO4)\}.$$

$$E = \{(AI2, AI3), (CD2, CD3), (CD4, CD5), (AO3, AO4)\}.$$

Observe que a determinação dos conjuntos não difere muito do caso monociclo. De fato, apenas o conjunto I_m precisa ser gerado, enquanto os demais são exatemente os mesmos do Caso I.

6.1.2 Estatísticas do Problema

A Tabela 6.2 mostra as estatísticas do problema para o Caso III, geradas pelo SCIP, antes e depois da etapa de pré-solução.

Observe como o problema aumentou consideravelmente quando comparado ao Caso I, seu equivalente na versão monociclo. Enquanto o número de variáveis não-binárias não aumenta muito, uma vez que ele está atrelado linearmente ao número total de execuções no macrociclo – e este número subiu de 19 para 31, o número de variáveis inteiras aumenta consideravelmente, especialmente nas formulações por sobreposição. Isso ocorre porque o número de binárias cresce na proporção do qua-

Tabela 6.2: Caso III: Estatísticas do Problema (entre parênteses, após pré-solução).

	Pontos de Evento	Sobreposição	
		Básico	Completo
Variáveis	313 (163)	421 (210)	345 (193)
Variáveis inteiras	225 (137)	315 (184)	239 (167)
Variáveis binárias	206 (134)	296 (181)	220 (167)
Restrições	3551 (2185)	679 (272)	1555 (1163)

drado do número de execuções.

O motivo pelo qual o número de variáveis binárias é superior na formulação por sobreposição o acréscimo das variáveis representantes das lacunas. Na formulação por sobreposição, o número dessas variáveis cresce com o quadrado do número de execuções de CDs, ao passo que na formulação por pontos de evento, esse número é o número de execuções de CD menos um, configurando uma relação linear. Por outro lado, da mesma maneira que no caso multiciclo, o número de restrições é muito superior na formulação por pontos de evento, novamente porque as restrições que tratam das relações de precedência crescem com o cubo do número de execuções.

Finalmente, é interessante comparar a formulação por sobreposição completa com a básica. O número de variáveis, mais particularmente as inteiras, diminui consideravelmente na formulação completa, em função das reduções que são feitas no modelo. Após a pré-solução, deixam de existir as variáveis inteiras não-binárias – que são os ciclos base. Isso é previsto, uma vez que essa variável de ciclo base só se faz necessária quando há malhas de controle com múltiplos ciclos ou quando há ciclos não-harmônicos entre si. Como elas ficam limitadas ao valor 1, a pré-solução as elimina. Em contrapartida, o número de restrições aumenta consideravelmente por causa das diversas restrições adicionadas na formulação completa para tornar o modelo mais justo.

De uma maneira geral, nos três casos a pré-solução foi capaz de reduzir os modelos consideravelmente, tanto no número de variáveis quanto no de restrições. Por se tratar de um modelo mais justo, a formulação por sobreposição foi aquela em que, proporcionalmente, menos houve reduções pela pré-solução.

6.1.3 Resultados para o Caso III

A tabela abaixo mostra as estatísticas de solução do Caso III para as três formulações distintas.

O problema foi resolvido com menos de seis horas para as três formulações diferentes. Pode-se ver novamente que a formulação por pontos de evento teve um

Tabela 6.3: Caso III: Estatísticas de Solução.

	Pontos de Evento	Sobreposição	
		Básico	Completo
Tempo de Solução	8640s (2h24min)	1403s (23min23s)	659s (10min59s)
Nós Buscados	2040211	1549351	358612
Iterações LP	12894551	8085281	3206009
Ótimo encontrado em	66s (1h06min)	23.51s	48.88s

tempo muito superior aos das formulações por sobreposição. No entanto, comparando a formulação por pontos de evento com a por sobreposição versão básica, o número de nós buscados não é tão significativamente diferente, nem o número de iterações LP: ambos são cerca de 30-50% maiores, ao passo que o tempo foi seis vezes maior. Pode-se perceber que o principal problema da formulação por pontos de evento não é exatamente a complexidade do problema binário, mas também a densidade do problema relaxado: apenas cerca de 1490 iterações por segundo, contra 5760 para a formulação básica por sobreposição.

A formulação completa por sobreposição tem a densidade um pouco maior que a básica, com 4865 iterações por segundo, mas compensa ao provar a otimalidade num tempo mais que duas vezes menor, com um número de nós buscados 4,3 vezes menor – o que indica a reduzida complexidade do problema inteiro nessa formulação ajustada. Por outro lado, dentre as três formulações, a que levou à solução ótima mais rapidamente foi a por sobreposição básica. É importante lembrar, contudo, que não é possível dizer que a solução encontrada é a ótima antes de o *solver* provar a otimalidade. Ainda assim, isso pode ser um indicativo de que a formulação básica é melhor para encontrar soluções boas rapidamente.

A principal conclusão desta análise de estatísticas é que o problema multiciclo parece muito mais complexo e demorado de se resolver do que o monociclo. No entanto, um tempo de 10 minutos não é impeditivo uma vez que trata-se de uma otimização offline.

Os gráficos de Gantt das Figuras 6.1 e 6.2 referem-se aos resultados do Caso III com o modelo monociclo e o modelo multiciclo, respectivamente. Os três modelos multiciclo geraram o mesmo resultado. A solução para o Caso III com o modelo monociclo por sobreposição foi de apenas 8.8 segundos no GLPK. No entanto, pode-se ver que, por não considerar os múltiplos ciclos, o modelo monociclo não consegue remover uma lacuna que o modelo multiciclo remove. Ainda assim, por mais que seu resultado seja sub-ótimo, trata-se de um bom resultado.

A Tabela 6.4 compara os critérios de otimização – lacunas, atrasos e tempo final – gerados pelos modelos mono e multiciclo. Pode-se ver que a solução do problema

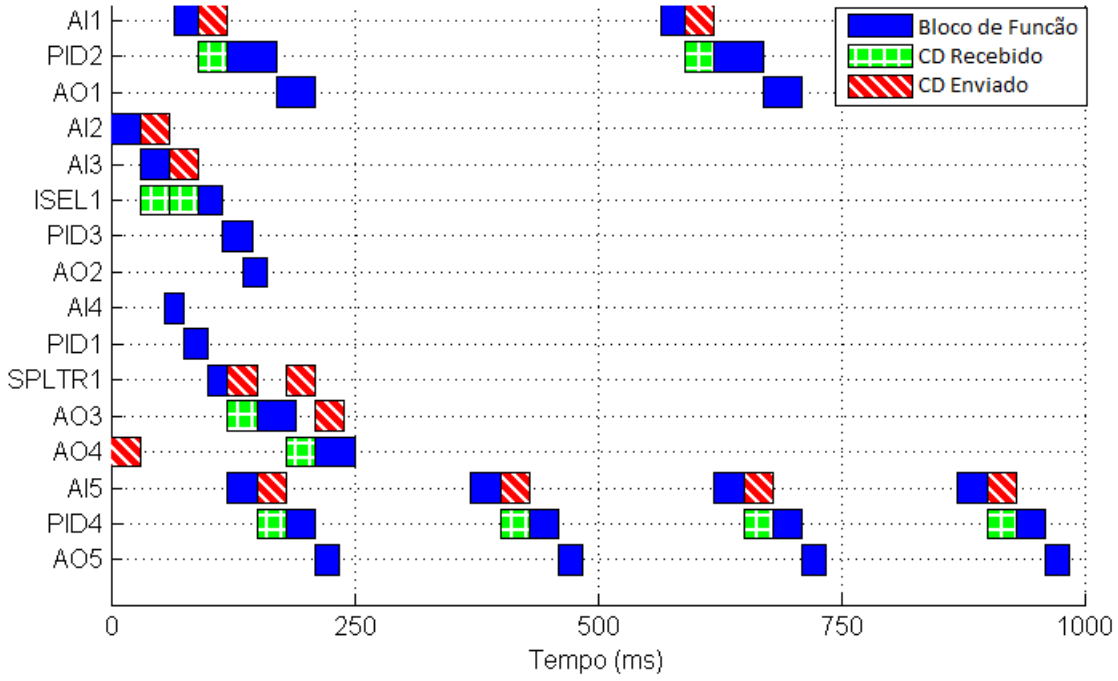

Figura 6.1: Gráfico de Gantt da agenda obtida pelo modelo monociclo para o Caso III.

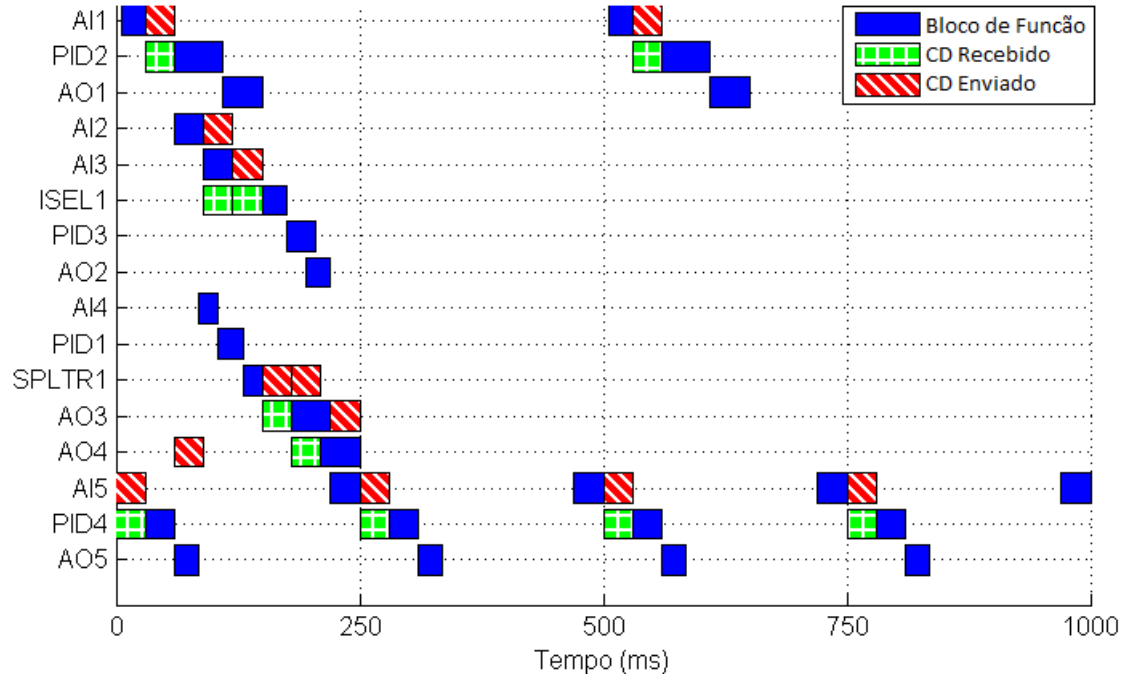

Figura 6.2: Gráfico de Gantt da agenda obtida pelos modelos multiciclo para o Caso III.

monociclo adaptada é um resultado sub-ótimo do problema multiciclo, que tem 25% menos lacunas (quatro contra três) e um atraso 50% inferior para a malha de controle 3.

Tabela 6.4: Caso III: Comparação de resultados entre a agenda do modelo monociclo e a do multiciclo.

Critério		Monociclo	Multiciclo	Redução (%)
Lacunas		4	3	25
Atrasos (Eq. 5.23) (ms)	Malha 1	0	0	0
	Malha 2	30	30	0
	Malha 3	60	30	50
	Malha 4	0	0	0
Tempo Final (Eq. 5.13) (ms)		250	250	0

6.2 Caso IV - Hodson (2005) - Três Submalhas de Ciclos Distintos

O caso a seguir é considerado em [21]. Trata-se de uma malha de controle formada por três submalhas, numa configuração de cascata tripla. Em aplicações típicas de malha de controle em cascata, o nível inferior constitui uma malha de controle para uma variável rápida, como vazão e pressão em processos químicos, ou torque ou velocidade em controle de máquinas. Nos níveis superiores, são controladas variáveis mais lentas, que são influenciadas por essas variáveis mais rápidas. É, portanto, natural que seja importante usar múltiplos ciclos para malhas de controle em cascata.

O Caso IV aqui reproduzido ilustra exatamente isso: a malha de controle no nível inferior tem ciclo de 500 ms. A do nível intermediário, 1000 ms. A do nível superior tem ciclo de 2000 ms. No total, três transmissores e um posicionador apenas compõem a rede. Trata-se de uma rede subutilizada, mas por se tratar de uma malha de controle de elevada complexidade, pode tornar o problema difícil de ser resolvido. Além disso, diferente do Caso III, esse caso leva em consideração o problema de múltiplos ciclos na mesma malha de controle. Para código, referir-se à Seção B.2.2.

Infelizmente, não é possível a comparação do método aqui apresentado com os resultados em [21], uma vez que esse artigo não apresenta os tempos de execução dos blocos de função nem o tempo de Compel Data. A malha de controle aqui tem exatamente a mesma estrutura, mas como tem os tempos diferentes, a comparação

não teria significado algum.

Os blocos de função configurados e seus tempos de execução estão descritos na tabela 6.5. O tempo de comunicação publicante/assinante configurado é de 30 ms. O diagrama de malha de controle encontra-se na Figura 6.3.

Tabela 6.5: Dispositivos, Tarefas e seus tempos de execução para o Caso IV.

j	i	Te_i (ms)	H_i (ms)
T1	AI1	25	2000
	PID1	60	2000
	AI4	20	2000
T2	AI2	30	1000
	PID2	55	1000
T3	AI3	30	500
P1	AR	40	500
	IN	35	500
	PID3	70	500
	AO	50	500
net	CD1	30	2000
	CD2	30	1000
	CD3	30	2000
	CD4	30	500
	CD5	30	1000
	CD6	30	500

6.2.1 Representação da Configuração usando o Modelo

Os conjuntos J, I e I_j, bem como os parâmetros Te_i dos blocos de função podem ser obtidos da tabela 6.5. Para a malha de controle retratada na figura 6.3, os seguintes conjuntos podem ser determinados:

$$M = \{1\}$$

$$\begin{aligned} I_1 \quad = \quad & \{AI1, PID1, CD1, PID2, CD2, PID3, AO, AI4, CD3, \\ & AR, IN, AI2, AI3, CD4, CD5, CD6\}, \end{aligned}$$

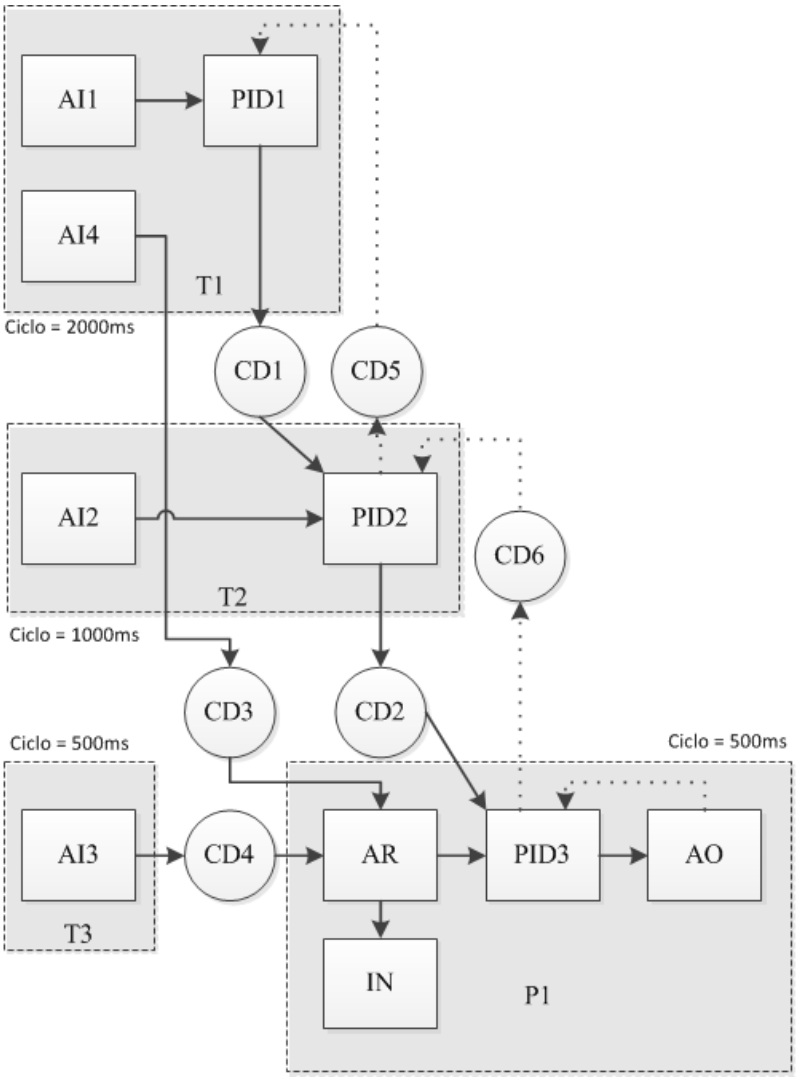

Figura 6.3: Diagrama da malha de controle usada no Caso IV. Retirado de [21]

$$
\begin{aligned}
L_1 \;=\; & \{(AI1, PID1), (PID1, CD1), (CD1, PID2),\\
& (PID2, CD2), (CD2, PID3), (PID3, AO), (AI4, CD3),\\
& (CD3, AR), (AR, IN), (AR, PID3),\\
& (AI2, PID2), (AI3, CD4), (CD4, AR)\},
\end{aligned}
$$

$$L^1_{T1,1} = \{(AI1, PID1)\},$$

$$L^1_{T2,1} = \{(AI2, PID2)\},$$

$$L^1_{T3,1} = \{(AR, IN), (AR, PID3), (PID3, AO)\},$$

$$L^1_{net,3} = \{(CD1, CD2)\},$$

$$L^1_{j,m} = \emptyset \text{ para os demais } j, m.$$

$$RB = \{CD5, CD6\},$$

$$LR_{CD5} = \{(PID2, PID1)\},$$

$$LR_{CD6} = \{(PID3, PID2)\}.$$

$$E = \emptyset.$$

6.2.2 Estatísticas do Problema

A Tabela 6.6 apresenta as estatísticas do Caso IV para as três formulações multiciclo. Algumas semelhanças ao que foi observado no caso III são encontradas: Um maior número de inteiras da formulação básica por sobreposição quando comparada à formulação por pontos de evento, mas um número muito maior de restrições na formulação por pontos de evento.

O que chama atenção nesse caso é como a formulação completa por sobreposição possui estatísticas muito superiores às demais: menos restrições e muito menos variáveis. Além disso, a pré-solução remove variáveis num fator de três para as outras formulações, ao passo que retira menos da metade das variáveis e cerca de um terço das restrições na formulação completa por sobreposição. Isso é um indicativo de que esta última é uma formulação mais justa especialmente para o caso em que há malhas de controle com múltiplos ciclos. De qualquer forma, ao fim da pré-solução as estatísticas não são muito diferentes, o que pode indicar que muitas das reduções feitas na formulação por sobreposição são também realizadas pela pré-solução do *solver*. Ainda assim, a redução nas variáveis da formulação básica para a completa, após a pré-solução, foi significativa: 36,7%.

Tabela 6.6: Caso IV: Estatísticas do Problema (entre parênteses, após pré-solução).

		Sobreposição	
	Pontos de Evento	Básico	Completo
Variáveis	641 (188)	769 (250)	319 (171)
Variáveis inteiras	524 (160)	652 (222)	202 (143)
Variáveis binárias	508 (153)	636 (215)	186 (136)
Restrições	11477 (1900)	1361 (281)	983 (609)

6.2.3 Resultados para o Caso IV

A Tabela 6.7 mostra as estatísticas de solução para o Caso IV, comparando as tr6es formulações. Desta vez, as diferenças de desempenho foram ainda maiores. O tempo de solução com a formulação por pontos de evento foi quase quarenta vezes maior que o levado usando a formulação por sobreposição básica, que por sua vez

demorou quatro vezes mais que a completa. Entre as formulações por sobreposição, a completa buscou cinco vezes menos nós que a básica, e teve cerca de 40% do número de iterações LP da mesma. Isso indica que cada nó na formulação completa leva mais iterações para fechar uma soluçãom ótima do subproblema relaxado. A densidade dos dois é comparável, sendo um pouco maior ainda na completa, o que é de se esperar pelo maior número de restrições.

Ao contrário do que aconteceu no Caso III, a formulação por sobreposição completa foi também a que proporcionou o menor tempo para encontrar a solução ótima, em apenas 8s. No geral, o tempo foi muito menor nesse caso quando comparado ao Caso III: cerca de 5% do melhor tempo do Caso III apenas. É um resultado esperado por dois motivos: o menor número de dispositivos e comunicações e o maior número de relações de precedência, que restringe o número de possibilidades e, por sua vez, limita a busca e a região viável do problema. O tempo levado na resolução com a adaptação do modelo monociclo foi de apenas 2.1s no GLPK.

Tabela 6.7: Caso IV: Estatísticas de Solução.

		Sobreposição	
	Pontos de Evento	Básico	Completo
Tempo de Solução	5100s (1h25min)	128s (2min8s)	35.11s
Nós Buscados	910647	109982	21043
Iterações LP	6804733	700995	289904
Ótimo encontrado em	630s (10min30s)	20.6s	8.2s

As Figuras 6.4 e 6.5 são os gráficos de Gantt dos resultados encontrados pelos modelos monociclo e multiciclo, respectivamente. As três formulações multiciclo levaram ao mesmo resultado. Pode-se ver que, nesse caso, o número de lacunas foi muito maior com o modelo monociclo: 6 contra 3. As três lacunas excedentes são devidas ao *readback* do PID3 ao PID2 nos três últimos subciclos de 500 ms. Não há lacunas no primeiro subciclo de 500 ms, pois este é o considerado na adaptação do modelo monociclo.

Os valores dos critérios de otimização para os modelos mono e multiciclo, encontrados na Tabela 6.8, mostram que apenas as lacunas sofrem alteração, ao passo que o atraso e o tempo final são os mesmos.

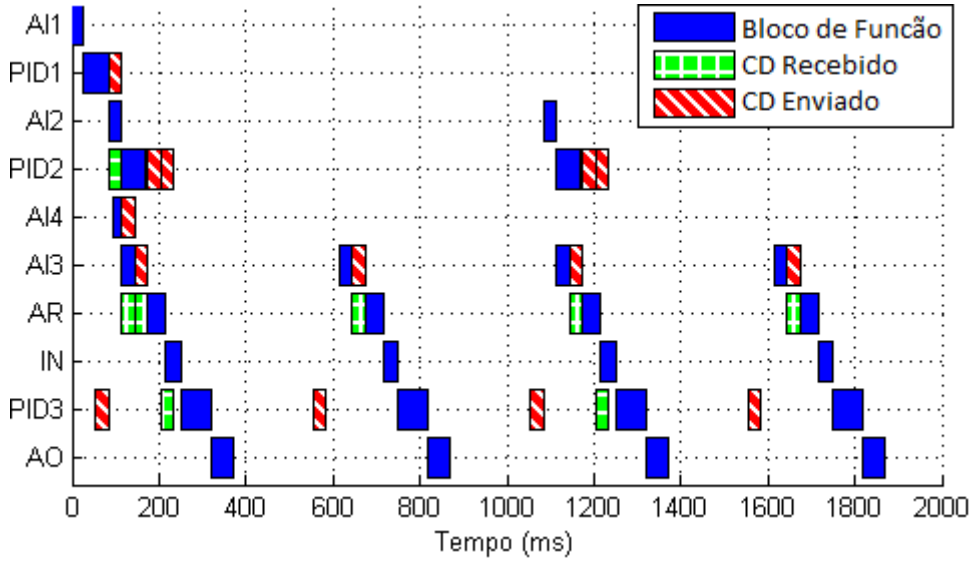

Figura 6.4: Gráfico de Gantt da agenda obtida pelo modelo monociclo para o Caso IV.

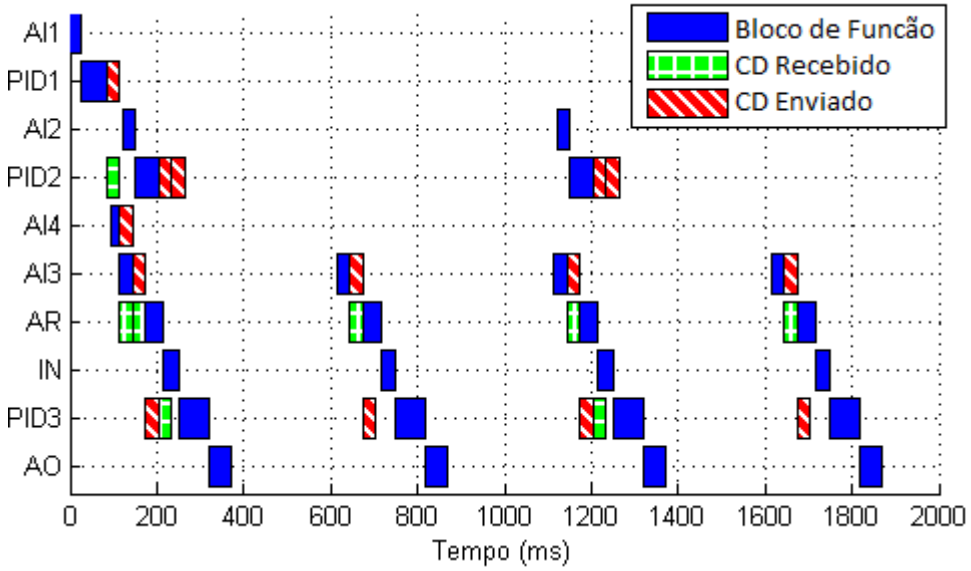

Figura 6.5: Gráfico de Gantt da agenda obtida pelos modelos multiciclo para o Caso IV.

Tabela 6.8: Caso IV: Comparação de resultados entre a agenda do modelo monociclo e a do multiciclo.

Critério	Monociclo	Multiciclo	Redução (%)
Lacunas	6	3	50
Atrasos da malha (Eq. 5.23) (ms)	115	115	0
Tempo Final (Eq. 5.13) (ms)	370	370	0

6.3 Caso V - Hodson (2005) com Três Malhas de Controle Independentes

O Caso V tem o objetivo de misturar o problema de múltiplos ciclos numa malha de controle, como o Caso IV, com a existência de um número considerável de dispositivos na rede, como o Caso III. Para isso, a rede do Caso V é composta pela malha de controle do Caso IV com as malhas de controle 1 e 2. No entanto, aqui a malha de controle 1 tem ciclo de 1000 ms e a 2, de 2000 ms. No total, 18 comunicações em 2000 ms, o que leva a uma ocupação de 27% da rede, com CDs de 30 ms. O código para entrada de dados no GLPK referente a esse estudo de caso pode ser visto na Seção B.2.2.

Os blocos de função configurados e seus tempos de execução estão descritos na tabela 6.9. O tempo de comunicação publicante/assinante configurado é de 30 ms. Os diagramas de malha de controle encontram-se nas Figura 6.3, 4.1 e 4.2 e não serão repetidos nesta seção.

6.3.1 Representação da Configuração usando o Modelo

Os conjuntos J, I e I_j, bem como os parâmetros Te_i dos blocos de função podem ser obtidos da tabela 6.9. Seguem os conjuntos obtidos a partir das malhas de controle:

$$M = \{1, 2, 3\}$$

$$\begin{aligned} I_1 \;=\; & \{AI1, PID1, CD1, PID2, CD2, PID3, AO, AI4, CD3, \\ & AR, IN, AI2, AI3, CD4, CD5, CD6\}, \end{aligned}$$

$$I_2 = \{AI5, CD7, AO2\},$$

$$I_3 = \{AI6, CD8, AI7, CD9, AO3\}.$$

$$\begin{aligned} L_1 \;=\; & \{(AI1, PID1), (PID1, CD1), (CD1, PID2), \\ & (PID2, CD2), (CD2, PID3), (PID3, AO), (AI4, CD3), \\ & (CD3, AR), (AR, IN), (AR, PID3), \\ & (AI2, PID2), (AI3, CD4), (CD4, AR)\}, \end{aligned}$$

$$L_2 = \{(AI5, CD7), (CD7, AO2)\},$$

$$L_3 = \{(AI6, CD8), (AI7, CD9), (CD8, AO3), (CD9, AO3)\}.$$

$$L^1_{T1,1} = \{(AI1, PID1)\},$$

Tabela 6.9: Dispositivos, Tarefas e seus tempos de execução para o Caso V.

j	i	Te_i (ms)	H_i (ms)
T1	AI1	25	2000
	PID1	60	2000
	AI4	20	2000
T2	AI2	30	1000
	PID2	55	1000
T3	AI3	30	500
P1	AR	40	500
	IN	35	500
	PID3	70	500
	AO	50	500
T4	AI5	25	1000
T5	AI6	30	2000
T6	AI7	30	2000
P2	AO2	90	1000
P3	AO3	80	2000
net	CD1	30	2000
	CD2	30	1000
	CD3	30	2000
	CD4	30	500
	CD5	30	1000
	CD6	30	500
	CD7	30	1000
	CD8	30	2000
	CD9	30	2000

$$L^1_{T2,1} = \{(AI2, PID2)\},$$

$$L^1_{T3,1} = \{(AR, IN), (AR, PID3), (PID3, AO)\},$$

$$L^1_{net,3} = \{(CD1, CD2)\},$$

$$L^1_{j,m} = \emptyset \text{ para os demais } j, m.$$

$$RB = \{CD5, CD6\},$$

$$LR_{CD5} = \{(PID2, PID1)\},$$

$$LR_{CD6} = \{(PID3, PID2)\}.$$

$$E = \{(AI6, AI7), (CD8, CD9)\}.$$

6.3.2 Estatísticas do Problema

A Tabela 6.10 mostra as estatísticas do Caso V para as diferentes formulações. Observe que esse problema é consideravelmente maior que os Casos III e IV, com cerca do dobro de variáveis e restrições, na média, para todas as formulações, principalmente após a etapa de pré-solução. Como no Caso IV, a formulação por sobreposição apresenta o menor número de variáveis inteiras, e número maior de restrições. Também como no Caso IV, a pré-solução se mostrou muito menos relevante para a formulação completa, quando comparada à básica e à por pontos de evento.

Tabela 6.10: Caso V: Estatísticas do Problema (entre parênteses, após pré-solução).

		Sobreposição	
	Pontos de Evento	Básico	Completo
Variáveis	823 (360)	1069 (432)	555 (330)
Variáveis inteiras	675 (321)	912 (393)	398 (291)
Variáveis binárias	651 (314)	888 (386)	374 (284)
Restrições	17722 (6869)	1820 (502)	2360 (1771)

6.3.3 Resultados para o Caso V

As estatísticas de solução encontram-se na Tabela 6.11. Os tempos de solução e demais estatísticas são significativamente maiores neste caso. Apenas com a formulação por sobreposição completa foi possível provar otimalidade global dentro de seis horas, mas ainda levou mais de duas horas para fazê-lo. Foram necessários milhões de nós buscados para provar otimalidade. Apesar deste tempo muito elevado, a solução ótima foi encontrada pela formulação completa em pouco mais de dois minutos e meio. Com a formulação básica, foi preciso mais de duas horas e meia para chegar nesse resultado, enquanto que este não foi encontrado com a formulação por pontos de evento nas seis horas em que a otimização foi executada.

Em função da elevada complexidade deste problema, foi introduzida uma nova estatística nessa tabela: o tempo levado para encontrar a primeira solução viável. Com a formulação completa por sobreposição, essa solução foi encontrada em menos de um segundo, contra três para a formulação básica. Em suma, a formulação completa por sobreposição foi não apenas mais eficiente que as demais, como foi a única capaz de resolver garantidamente o problema em menos de seis horas, e conseguiu um ótimo tempo para encontrar a solução ótima.

As Figuras 6.6 e 6.7 mostram os resultados do Caso V para os modelos monociclo e multiciclo, respectivamente, através de gráficos de Gantt. O tempo levado no

Tabela 6.11: Caso V: Estatísticas de Solução.

	Pontos de Evento	Sobreposição	
		Básico	Completo
Tempo de Solução	21600s* (6h)	21600s** (6h)	8100s (2h15min)
Nós Buscados	1742347	12821620	2416651
Iterações LP	24976131	51106625	4020071
Primeira solução em	42.8s	3.08s	0.23s
Ótimo encontrado em	-*	9300s (2h35min)	156s (2min36s)

*Solução interrompida após 6h, com um *Integrality Gap de 84%*. O ótimo não foi encontrado.

**Solução interrompida após 6h, com um *Integrality Gap de 43.4%*

GLPK, para o modelo monociclo adaptado, foi de 41min15s. Pelos gráficos e pela Tabela 6.12, vê-se que novamente as lacunas são diminuídas no comparativo entre mono e multiciclo, ao passo que os outros critérios se mantém iguais. O tempo final da solução monociclo é até inferior ao da solução macrociclo, mas vale lembrar que esse critério não é relevante para o desempenho da rede, mas apenas um fator de redução dos graus de liberdade do problema.

Tabela 6.12: Caso V: Comparação de resultados entre a agenda do modelo monociclo e a do multiciclo.

Critério		Monociclo	Multiciclo	Redução (%)
Lacunas		5	3	40
Atrasos (Eq. 5.23) (ms)	Malha 1	115	115	0
	Malha 2	0	0	0
	Malha 3	30	30	0
Tempo Final (Eq. 5.13) (ms)		345	375	-8

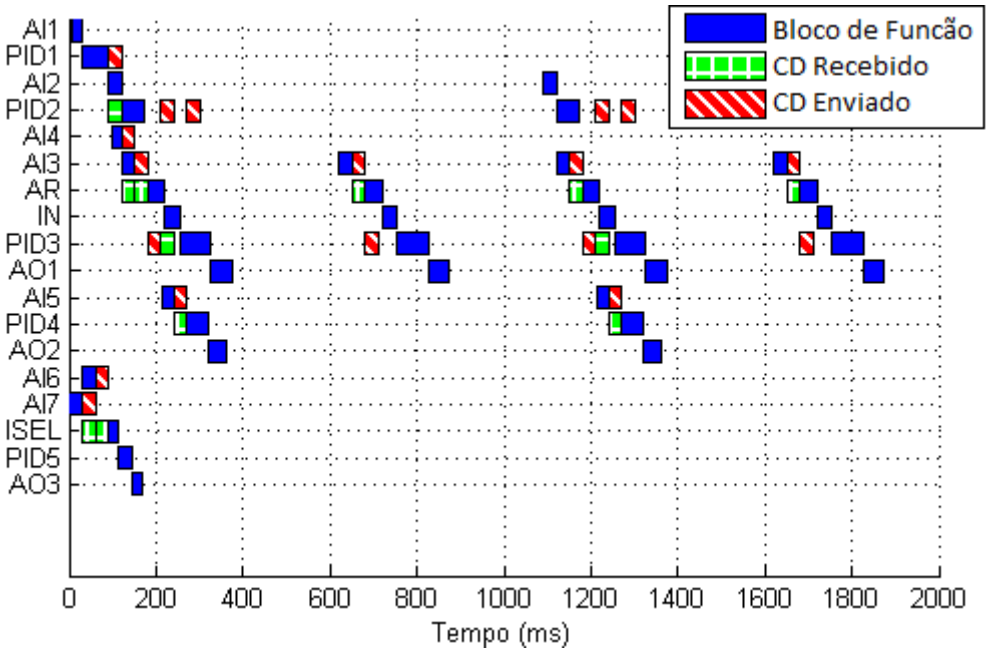

Figura 6.6: Gráfico de Gantt da agenda obtida pelo modelo monociclo para o Caso V.

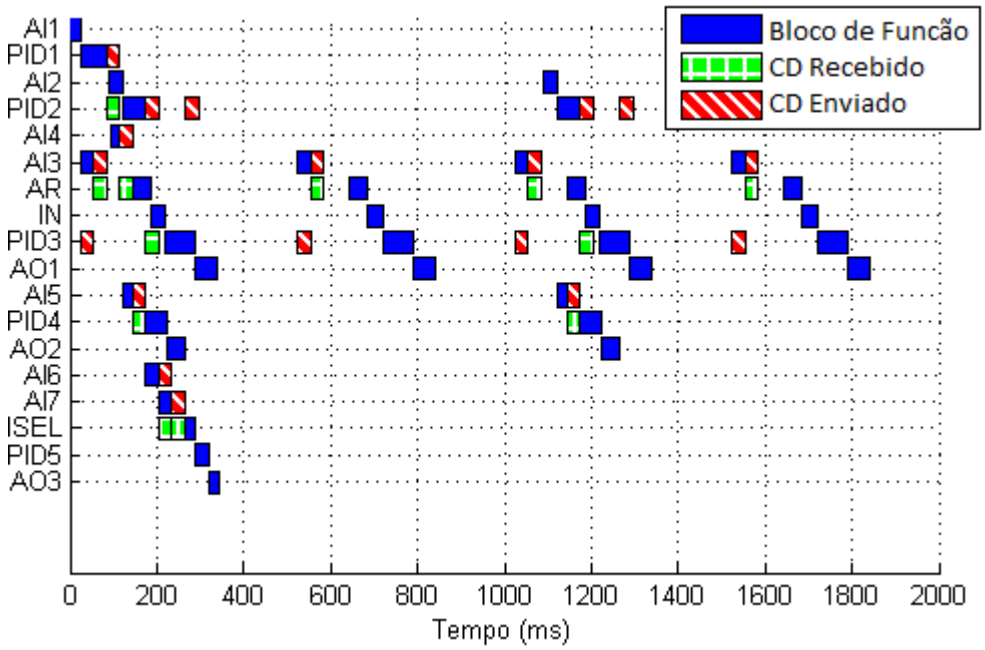

Figura 6.7: Gráfico de Gantt da agenda obtida pelos modelos multiciclo para o Caso V.

6.4 Caso VI - Ciclos Heterogêneos

O último subproblema do caso multiciclo é a existência de ciclos heterogêneos, ou seja, ciclos que não sejam múltiplos uns dos outros. Suponha que um estudo para as malhas de controle 4 e 1 do Caso III tenha levado à conclusão que seus ciclos máximos para um desempenho satisfatório não podem ser 250ms e 500ms, respectivamente, mas 210ms e 450ms. Se esses valores fossem usados como os ciclos das malhas de

controle na rede, obter-se-ia um macrociclo de 63000ms, que é o MMC entre 210ms, 450ms e 1000ms. Esse é um macrociclo muito elevado, que exigiria a consideração de 300 repetições da malha de controle 4, 140 da malha de controle 1 e 63 das malhas de controle 2 e 3.

Para trabalhar com uma quantidade tratável de variáveis e tornar a agenda mais compacta, é recomendável ajustar os ciclos. Um ajuste possível seria determinar ciclos de 400ms e 200ms para as malhas de controle 1 e 4. O macrociclo iria aumentar de 1000ms para 2000ms, e seriam consideradas 5 repetições da malha de controle 1, 2 das malhas de controle 2 e 3, e 10 repetições da malha de controle 4. Outro ajuste mais limitante, mas que evitaria a existência de ciclos heterogêneos, seria também ajustar os ciclos das malhas de controle 2 e 3 para 800ms, levando a um macrociclo de também 800ms. Um problema que esse tipo de corte gera é a ocupação maior da rede: nesse caso, a ocupação iria para 360ms em 800ms, ou 45% do tempo, para CDs. Com o ciclo em 1000ms para as malhas de controle 2 e 3, a ocupação é de 810ms a cada 2000ms: 40.5%.

Dadas as considerações acima, optou-se por configurar a rede com os ciclos de 400ms para a malha de controle 1, 1000ms para as malhas de controle 2 e 3, e 200ms para a malha de controle 4. Os blocos de função configurados e seus tempos de execução estão descritos na tabela 6.13. O tempo de Compel Data configurado é de 30 ms. Os diagramas de malha de controle encontram-se nas Figuras 4.1, 4.2 e 4.3 e não serão repetidos nesta seção. O código para entrada de dados no GLPK encontra-se na Seção B.2.2

6.4.1 Representação da Configuração usando o Modelo

Os conjuntos J, I e I_j, bem como os parâmetros Te_i dos blocos de função podem ser obtidos da tabela 6.13. Seguem os conjuntos obtidos a partir das malhas de controle:

$$M = \{1, 2, 3\}$$

$$\begin{aligned} I_1 \;=\; & \{AI1, PID1, CD1, PID2, CD2, PID3, AO, AI4, CD3, \\ & AR, IN, AI2, AI3, CD4, CD5, CD6\}, \end{aligned}$$

$$I_2 = \{AI5, CD7, AO2\},$$

$$I_3 = \{AI6, CD8, AI7, CD9, AO3\}.$$

Tabela 6.13: Dispositivos, tarefas, tempos de execução e tempos de ciclo para o Caso VI.

j	i	Te_i (ms)	H_i (ms)
AI1	AI1	25	400
AI2	AI2	30	1000
AI3	AI3	30	1000
	AI41	45	1000
AI4	AI41	20	1000
AI5	AI5	30	200
AO1	AO1	90	400
AO2	AO2	80	1000
AO3	AO3	40	1000
AO4	AO4	40	1000
AO5	AO5	55	200
	CD1	30	400
	CD2	30	1000
	CD3	30	1000
net	CD4	30	1000
	CD5	30	1000
	CD6	30	1000
	CD7	30	1000
	CD8	30	200

$$\begin{aligned} L_1 \;=\; & \{(AI1, PID1), (PID1, CD1), (CD1, PID2), \\ & (PID2, CD2), (CD2, PID3), (PID3, AO), (AI4, CD3), \\ & (CD3, AR), (AR, IN), (AR, PID3), \\ & (AI2, PID2), (AI3, CD4), (CD4, AR)\}, \end{aligned}$$

$$L_2 = \{(AI5, CD7), (CD7, AO2)\},$$

$$L_3 = \{(AI6, CD8), (AI7, CD9), (CD8, AO3), (CD9, AO3)\}.$$

$$L^1_{T1,1} = \{(AI1, PID1)\},$$

$$L^1_{T2,1} = \{(AI2, PID2)\},$$

$$L^1_{T3,1} = \{(AR, IN), (AR, PID3), (PID3, AO)\},$$

$$L^1_{net,3} = \{(CD1, CD2)\},$$

$$L^1_{j,m} = \emptyset \text{ para os demais } j, m.$$

$$RB = \{CD5, CD6\},$$

$$LR_{CD5} = \{(PID2, PID1)\},$$

$$LR_{CD6} = \{(PID3, PID2)\}.$$

$$E = \{(AI6, AI7), (CD8, CD9)\}.$$

6.4.2 Estatísticas do Problema

A Tabela 6.14 mostra as estatísticas do Caso VI para as diferentes formulações. Esse problema, pela existência de mais repetições de cada malha de controle devidas à heterogeneidade dos ciclos, é ainda maior em número de variáveis que o Caso V. Comparando ao caso base, o Caso III, ele chega a ter mais que o dobro de variáveis. Comparando as formulações básica e completa por sobreposição, vê-se que o número de variáveis inteiras cresceu num fator de cinco para a básica, ao passo que para a completa, o crescimento foi num fator de dois.

Mesmo depois da pré-solução, o problema resultante tem, no melhor caso, 344 variáveis inteiras para determinar. Neste caso heterogêneo, o modelo completo por sobreposição apresenta estatísticas muito superiores aos demais. Também se mostrou mais justo, com a pré-solução removendo apenas cerca de 20% das variáveis, enquanto eliminou mais da metade nos demais.

Tabela 6.14: Caso VI: Estatísticas do Problema (entre parênteses, após pré-solução).

		Sobreposição	
	Pontos de Evento	Básico	Completo
Variáveis	1248 (510)	1853 (727)	629 (383)
Variáveis inteiras	1062 (490)	1667 (688)	443 (344)
Variáveis binárias	1043 (490)	1648 (682)	424 (344)
Restrições	47512 (15200)	3013 (633)	2505 (1780)

6.4.3 Resultados para o Caso VI

As estatísticas de solução encontram-se na Tabela 6.15. Os números desse problema são comparáveis aos do Caso V: apenas com a formulação completa por sobreposição foi possível provar otimalidade com o *solver*, e isso levou três horas e meia para ser atingido. A maior razão entre iterações LP e nós buscados (10 contra 2, aproximadamente) indica que não apenas a busca em árvore foi mais longa, mas a convergência do problema relaxado também é mais demorada.

A formulação por pontos de evento teve um baixo desempenho neste problema. A primeira solução viável foi a única encontrada, com o valor da função objetivo duas

vezes maior que o ótimo, enquanto não conseguiu elevar o nível do limite inferior da solução desde a solução do problema base relaxado. Considerando o problema de encontrar uma solução viável, o modelo básico por sobreposição teve o melhor resultado, fazendo-o em quase 4 segundos, contra os 15 do modelo completo.

Apesar do tempo de solução ser muito longo, a formulação completa conseguiu evoluir suas soluções rapidamente a ponto de encontrar o ótimo após quase quatro minutos, coisa que com a formulação básica só foi possível após uma hora. Nota-se então, como no Caso V, que com a formulação completa é possível atingir o resultado ótimo em poucos minutos.

Tabela 6.15: Caso VI: Estatísticas de Solução.

	Pontos de Evento	Sobreposição	
		Básico	Completo
Tempo de Solução	21600s* (6h)	21600s** (6h)	12408s (3h27min)
Nós Buscados	178697	7338010	4093886
Iterações LP	25594841	43067626	39226734
Primeira solução em	9897s (2h45min)	3.84s	15.37s
Ótimo encontrado em	-*	4050s (1h7min30s)	237.86s (3min58s)

*Solução interrompida após 6h, com um *Integrality Gap de 99.3%*. O ótimo não foi encontrado.

**Solução interrompida após 6h, com um *Integrality Gap de 83.1%*

A Figura 6.8 mostra os resultados do Caso VI para o modelos multiciclo apenas, já que não é possível usar o modelo monociclo para o problema heterogêneo. A solução foi encontrada pelos dois modelos por sobreposição. Pelo gráficos e pela Tabela 6.16, vê-se que há um número significativamente maior de lacunas, se comparado aos casos anteriores. No entanto, é importante notar que há um maior número de comunicações nesse caso. No total, há 9 lacunas em 27 comunicações, correspondendo a apenas um terço das comunicações com lacunas entre si. No Caso III, por exemplo essa proporção foi de 3 lacunas em 12 comunicações, ou um quarto. Pelo gráfico, vê-se que a maior parte das lacunas está entre repetições do CD da malha de controle 4. No fim, há uma lacuna a cada intervalo de 200ms, o que é o melhor que podia ser feito.

As estatísticas de atraso permaneceram inalteradas na comparação com o Caso III. O tempo final foi maior, de 290ms contra 250ms do Caso III.

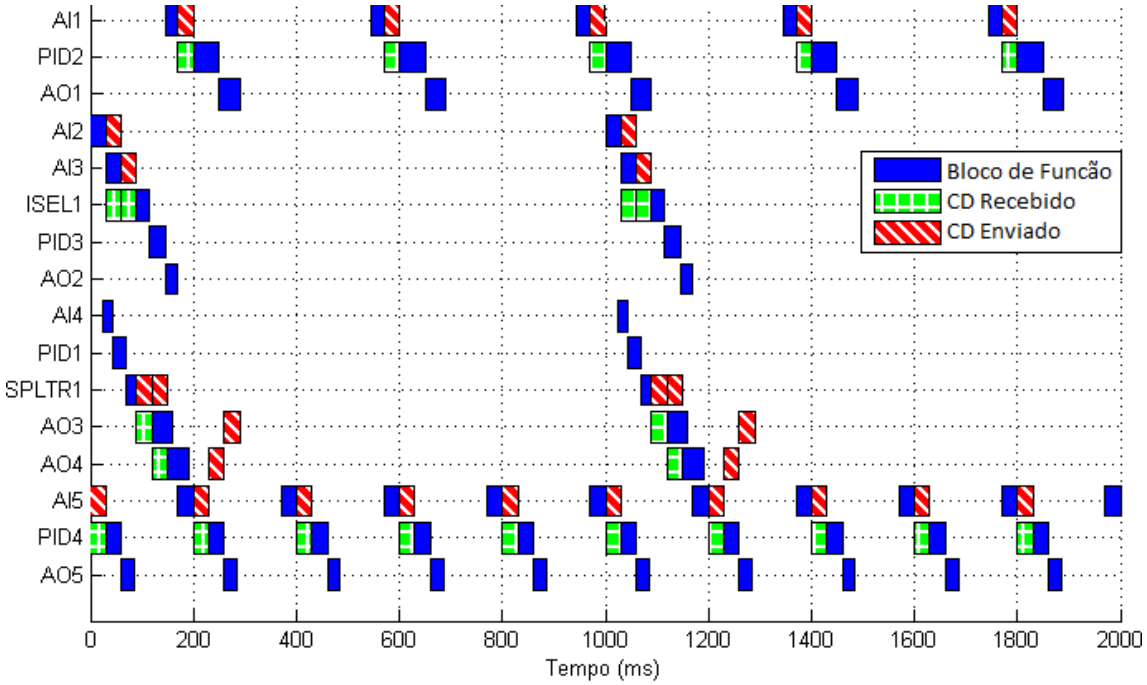

Figura 6.8: Gráfico de Gantt da agenda obtida pelo modelos multiciclo por sobreposição para o Caso VI.

Tabela 6.16: Caso VI: Resultados.

Critério		Multiciclo
Lacunas		9
Atrasos (Eq. 5.23) (ms)	Malha 1	0
	Malha 2	30
	Malha 3	30
	Malha 4	0
Tempo Final (Eq. 5.13) (ms)		290

6.5 Discussões

Neste capítulo foram aplicadas as formulações para o problema de escalonamento de tarefas numa rede de campo em que são possíveis múltiplos ciclos de execução, inclusive com múltiplos ciclos na mesma malha de controle. As três formulações atendem ao problema de otimização, mas com diferenças significativas de desempenho. A formulação por sobreposição completa teve o melhor desempenho em todas as instâncias, ao passo que a formulação por pontos de evento foi a que teve pior desempenho com tempos muito longos para solução e, nos casos mais complexos, falhando em obter a solução ótima num tempo de seis horas.

Para os problemas que envolviam ciclos harmônicos, é também possível adaptar

a modelagem do caso monociclo. A solução é encontrada muito mais rapidamente, mas por outro lado ela não é garantidamente ótima para o problema multiciclo. De fato, em todos os casos em que ela foi usada, os resultados foram sub-ótimos considerando número de lacunas e tempo de atraso das malhas de controle.

Conforme previsto, os tempos de solução foram muito superiores aos tempos do caso monociclo. Enquanto o último obteve tempos de no máximo 30s para provar a otimalidade da solução, no caso multiciclo os tempos chegaram a algumas horas, mesmo com o uso de um *solver* mais poderoso. Foi possível ver que o tamanho dos problemas, em número de variáveis e de restrições, cresceu significativamente quando comparado a redes de mesmo tamanho, mas com apenas um ciclo: enquanto o pior caso dos problemas monociclo tinha cerca de cem variáveis binárias, esse número chegou a mais de trezentos no caso multiciclo.

Por outro lado, esses longos tempos não são impeditivos pois trata-se de um problema de otimização offline. Além disso, a solução ótima foi encontrada em menos de cinco minutos para todos os casos com a formulação por sobreposição completa. Em problemas de programação inteira mista, é comum acontecer isso: a solução ótima é encontrada num tempo muito menor do que o levado para os limites superior e inferior convergirem e a prova ser completa. Com isso em vista, se para o usuário o tempo até o ótimo se tornar proibitivo, é possível que se obtenha uma solução viável num tempo razoável. Vale ressaltar que essa conclusão é baseada nos estudos de caso deste trabalho e não é possível prever de antemão o desempenho da otimização de problemas desse tipo.

Capítulo 7

Conclusões

Neste trabalho um novo procedimento para gerar agendas ótimas para redes de campo determinísticas com controle distribuído foi proposto. O procedimento se baseia em Programação Linear Inteira Mista, com um modelo baseado na literatura de Sistemas de Computação, mais especificamente Programação Paralela, e outro baseado na literatura da Engenharia Química, com modelos de Processos em Batelada. O método foi aplicado ao protocolo FOUNDATION™ Fieldbus, tanto para o caso monociclo quanto para o macrociclo. Até onde sabemos, este é o primeiro trabalho que aplica otimização *stricto sensu* para escalonamento de tarefas em redes de campo.

Além da modelagem, foram desenvolvidos objetivos aplicáveis ao problema FF. Os tradicionais objetivos usados em Sistemas em Tempo Real não são os mais relevantes em Sistemas de Controle em Rede, onde o problema dos atrasos e a escassez de banda disponível tornam-se mais críticos. Os modelos usados também contêm algumas variáveis e restrições específicas a FF. A modelagem do critério de agrupamento de comunicações para o caso multiciclo foi particularmente desafiadora, mas mostrou-se bem sucedida e é outra contribuição deste trabalho.

Estudos de caso mostraram que o método é eficiente para otimizar o escalonamento para redes de tamanho industrial. Os resultados são precisos, sem necessidade de ajustes, já que o modelo considera os critérios desejados de uma maneira direta e garante viabilidade. Os tempos de solução para sistemas reais foram da ordem de 10s a 1min para o caso monociclo, o que é muito satisfatório.

No caso multiciclo, os resultados foram satisfatórios e a otimização foi alcançada considerando ciclos heterogêneos e múltiplos ciclos na mesma malha. Após o aproveitamento de características estruturais do problema, foi possível aumentar significativamente o desempenho da otimização através do desenvolvimento de uma modelagem mais justa. Com o uso dessa modelagem, a solução ótima foi encontrada em todos os casos num tempo inferior a cinco minutos. Particularmente no caso de ciclos não-harmônicos, outros modelos matemáticos para problemas semelhantes não

foram encontrados na literatura.

A aplicabilidade do método é garantida uma vez que o modelo é totalmente parametrizado e o usuário apenas precisa configurar os parâmetros e os elementos dos conjuntos. Isso pode ser feito automaticamente via software, uma vez integrado o otimizador ao sistema de configuração.

O método proposto aqui se aplica a outras redes de controle de características semelhantes, ou seja, determinísticas e cíclicas. O valor desse modelo é maior para os casos de funcionalidade distribuída nos dispositivos, que agrega complexidade ao problema de escalonamento.

O presente trabalho, portanto, resolve o problema de otimização de redes FF H1 e, potencialmente, outras redes. O procedimento aqui apresentado é aplicável nas redes FF existentes atualmente sem necessidade de modificações e contribui para a redução do principal gargalo de projetos FF H1: a subutilização da banda pelo mau gerenciamento das tarefas. Isso proporciona maiores qualidades de serviço e de controle. Além disso, abre campo para redes FF melhor aproveitadas, com mais dispositivos do que o que se pratica nos dias de hoje.

7.1 Trabalhos Futuros

Para redes de campo, ainda há alguns desafios a serem encarados. Alguns protocolos são não-determinísticos e aleatoriedade não foi abordada neste modelo. A técnica deste trabalho poderia ser usada como uma abordagem de pior caso, mas abordagens estocásticas poderiam ser usadas para mudar o modelo e os critérios de otimização. Protocolos não-determinísticos incluem os *wireless*, que estão num momento de crescente aplicação em plantas industriais.

O modelo MILP aqui desenvolvido para o caso multiciclo apresentou um alto nível de complexidade, implicando no custo computacional elevado. Outras técnicas de otimização ou heurísticas / meta-heurísticas poderiam ser testadas para este caso e comparadas. Melhorias no modelo atual também poderiam ser buscadas.

Como foi visto neste trabalho, diferentes áreas de aplicação de Pesquisa Operacional, desde sistemas digitais em tempo real a processos químicos em batelada, possuem vários pontos em comum, tendo destaque a necessidade de uma representação temporal eficiente. Percebeu-se que o mesmo problema, com praticamente o mesmo modelo, pode ter seu desempenho enormemente alterado em função da representação temporal. Neste trabalho, apenas representações temporais contínuas foram testadas e a formulação por sobreposição se mostrou mais eficiente. Seria interessante tentar aplicar essa formulação no problema de programação de processos em batelada, da mesma forma que avaliar a efetividade de uma representação por tempo discreto no problema FF. Uma análise mais aprofundada que explique

a causa das enormes diferenças de desempenho teria imenso valor para o campo de PLIM.

Finalmente, um trabalho futuro fundamental é o desenvolvimento de um software integrado a um SDCD que aplique esta metodologia.

Apêndice A

Introdução a FOUNDATION™ Fieldbus H1

Este capítulo tem o objetivo de dar ao leitor uma visão geral sobre a tecnologia FOUNDATION™ Fieldbus do ponto de vista do usuário, além de garantir o conteúdo técnico necessário para o entendimento do problema de escalonamento e sua importância. Este texto não tem a função de ser uma fonte primária de conhecimento sobre FF. O leitor, no entanto, é convidado a conhecer boas referências na área. Documentos oficiais da Foundation Fieldbus se encontram disponíveis em sua página na internet (http://www.fieldbus.org, acessada em 11 de agosto de 2012), com destaque para o Technical Overview[16] e o Systems Engineering Guidelines[15]. Bons livros incluem Foundation Fieldbus, de Ian Verhappen e Augusto Pereira[41] e Fieldbuses for Process Control, de Jonas Berge[4]. Para maior aprofundamento, normas FF podem ser compradas no site da Fieldbus Foundation.

Em função das necessidades de integração de informações e do baixo custo de instalação dos componentes de conectividade, plantas industriais de processos possuem nos dias de hoje redes de comunicação em variados níveis: desde o chão de fábrica até o nível corporativo, passando pelos níveis operacional, de supervisão e de planta. A figura A.1 ilustra a esses diferentes níveis numa planta. As redes que proporcionam a comunicação entre os ativos de chão de fábrica, mais precisamente sensores e atuadores, e os sistemas de controle/supervisão são chamadas de redes industriais de campo, ou *fieldbuses*. Dentre as redes de campo predominantes na indústria de processos, destacam-se os protocolos HART, FF H1 e Profibus DP.

Historicamente, na era da comunicação por fios (i.e., após a introdução do sistema 4-20mA de comunicação, substituindo aos poucos o sistema pneumático), observou-se uma migração do sistema analógico para o digital. A possibilidade de usar os canais existentes para transmitir mais informação do que meramente as variáveis de processo levou ao surgimento à posterior ampla utilização do sistema 4-20mA + HART. O HART utiliza um sinal modulado em frequência por sobre o sinal 4-

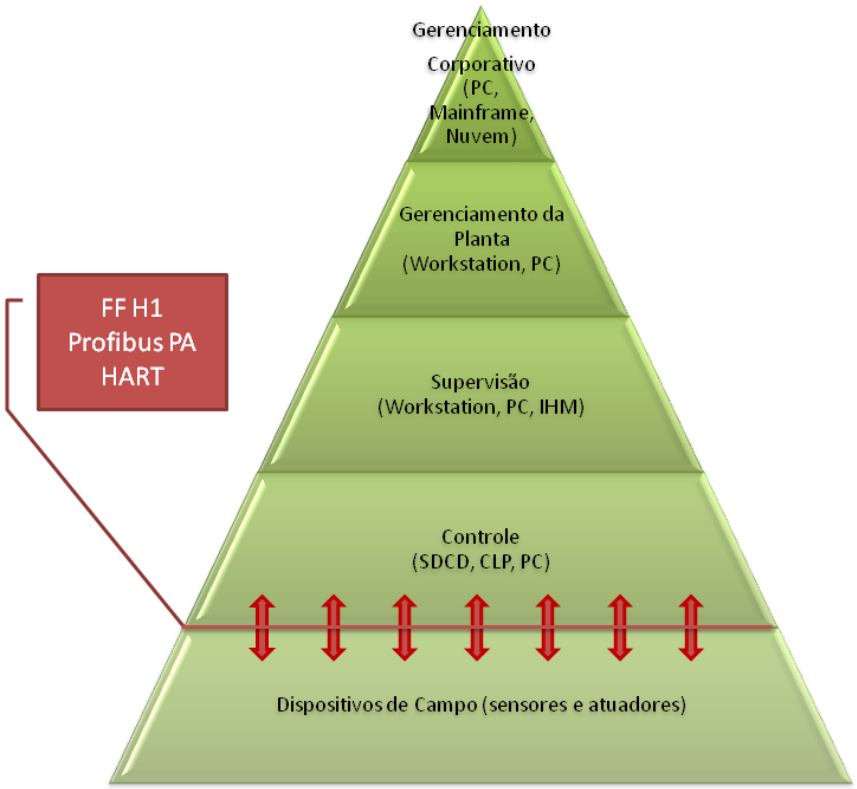

Figura A.1: Diagrama esquemático representando os níveis típicos a que pertencem os ativos de monitoramento, controle e informação numa organização industrial. Em destaque os protocolos de redes de campo, para plantas de processo.

20mA, de variabilidade lenta. Tal protocolo possibilitou a passagem de informações extras de/para os dispositivos de monitoração e controle, tais como parâmetros de configuração e variáveis de alarme. Essa nova possibilidade reduziu tempos de implantação da instrumentação e permitiu o começo do gerenciamento online dos ativos. Ou seja, tornou-se possível monitorar os dispositivos remotamente e sem a necessidade de intervenção na operação.

Apesar de ser um avanço inegável, a tecnologia HART possui diversas limitações. A velocidade máxima é de 1200bps, dificultando a transmissão de quantidades maiores de dados. A variável de processo permanece analógica e, portanto, sujeita a variações que não são diagnosticáveis pelo usuário. Por fim, mesmo possuindo a capacidade de prover alarmes de mau funcionamento, o protocolo não permite o aviso do dispositivo ao sistema: o sistema deve procurar por alarmes, varrendo todos os seus ativos de tempos em tempos. Além de sobrecarregar o sistema com requisições majoritariamente desnecessárias, o tempo necessário para fazer toda a varredura pode ser o suficiente para que uma eventual falha de dispositivo já tenha impactado o processo.

Com o redução de custos e a popularização dos equipamentos digitais, surgiram os dispositivos inteligentes, os SDCDs e as redes de campo puramente digitais. Os dispositivos inteligentes possuem eletrônica embarcada que possibilita um grande número de funções internas. Isso inclui desde conversão de sinais até filtragem, cálculos de funções de controle e diagnóstico avançado, tanto do dispositivo quanto de sua interface com o processo. Os SDCDs aumentaram o escopo de funções e a capacidade computacional dos sistemas de controle. Por sua vez, as redes de controle digitais não somente reduziram os cabeamentos de campo como possibilitaram o aproveitamento das capacidades dos dispositivos e removeram a sobrecarga dos sistemas de controle ao mesmo tempo que proporcionaram a estes sistemas informações e capacidades mais relevantes.

Baseados na norma IEC 61158, que define a camada física, os protocolos FF H1 e Profibus DP usam apenas comunicação digital para transmitir, nos dois sentidos, dados de processo, parâmetros de configuração, históricos (*trends*), alarmes, especificações dos dispositivos e informações de diagnóstico avançado. O protocolo FF H1 é mais sofisticado, implementando o conceito de Controle Distribuído e possibilitando funções básicas de controle regulatório como o controlador PID serem executadas nos dispositivos de campo. Dentre as vantagens desta estratégia, destacam-se a redução do número de comunicações (que impacta positivamente no atraso fim-a-fim da malha de controle), a redução da carga do SDCD e o aumento da disponibilidade da malha (no caso de falha de comunicação com o SDCD ou do próprio SDCD). Por fim, a implementação do *"report by exception"* permite que um dispositivo comunique um alarme imediatamente após sua ocorrência.

Há algumas diferenças marcantes entre as tecnologias FF H1 e Profibus PA. FF H1 é possui maior complexidade em termos de número de parâmetros e de configuração, possui a tecnologia de Controle no Campo e em geral permite um máximo de 16 dispositivos por rede. Por outro lado, o Profibus PA tem menor complexidade, com o controle centralizado no controlador e permite um máximo de 32 dispositivos por rede. Essa última foi um dos elementos motivadores deste trabalho. Os dois protocolos se baseiam na mesma camada física. No entanto, o Profibus PA permite o dobro de dispositivos por rede que o FF H1, que é de no máximo 16 dispositivos e não costuma, na prática, a ultrapassar 10 ou 12.

Para entender os motivos que levam a esse subaproveitamento da rede FF H1, é preciso verificar alguns tópicos de interesse. Na seção a seguir, há uma breve explicação sobre a norma IEC 61158, que define a camada física do protocolo FF H1. Em seguida, uma breve introdução ao tema Projeto de Redes FF na Seção A.2 ajuda a entender as variáveis e restrições que envolvem um projeto, à luz do guia de projeto em [15]. Por fim, a Seção A.3 apresenta a configuração no ambiente FF e seus impactos na rede.

A.1 Camada Física – IEC 61158

Como já foi dito anteriormente, a norma IEC 61158[9] define o funcionamento da camada física, de acordo com o modelo *Open Systems Interconnection* (OSI), dos protocolos FOUNDATION™ Fieldbus e Profibus DP, entre outros. Essa camada física tem como destaque as seguintes características:

- Comunicação via par trançado, com fios de espessura entre 22 e 14 *American Wire Gauge* (AWG);
- Sinal digital de 750mV a 1V pico-a-pico, com codificação Manchester, sobre o sinal corrente contínua (DC, do inglês *direct current*) de alimentação;
- Velocidade de 31.25kbps;
- Tensão de operação de dispositivos entre 32V e 9V;
- Até 32 dispositivos por rede.

Além dessas características, a norma define limites aceitáveis de ruído em diferentes faixas de frequência, além de um limite para o *jitter*, que é o intervalo de tempo entre o instante esperado para o sinal cruzar o zero e o instante em que isso ocorre de fato.

Como se pode ver, a camada física impõe algumas restrições ao tamanho da rede FF. Além do limite rígido de 32 dispositivos, a questão de tensão de operação deve ser observada. Quanto mais dispositivos na rede, maior o consumo de corrente e, portanto, maior a queda de tensão no cabeamento. Isso será mais detalhado na Seção A.2, a seguir. Outras preocupações com relação à camada física incluem a limitação do comprimento da rede e o uso de terminadores, devido a efeitos de desbalanceamento, a blindagem dos cabos devido ao ruído e o aterramento apropriado. Informações práticas para o usuário a respeito disso encontram-se em [15, 41].

A.2 Projeto de redes FF

Além das restrições impostas pela camada física, outras são impostas baseadas em boas práticas ou na capacidade de comunicação da rede ou de processamento dos elementos que a ela pertencem. As restrições serão aqui explicadas com base na topologia recomendada em [15, 41]: pé de galinha ou, numa generalização, tronco e ramificações. A Figura A.2 ilustra essa topologia.

Os elementos usados numa rede FF H1, ilustrados na Figura A.2, são os seguintes:

- *(*Host), ou Servidor, o elemento que se comunica com o controlador. Em geral é um *Linking Device* ou um cartão de SDCD específico;

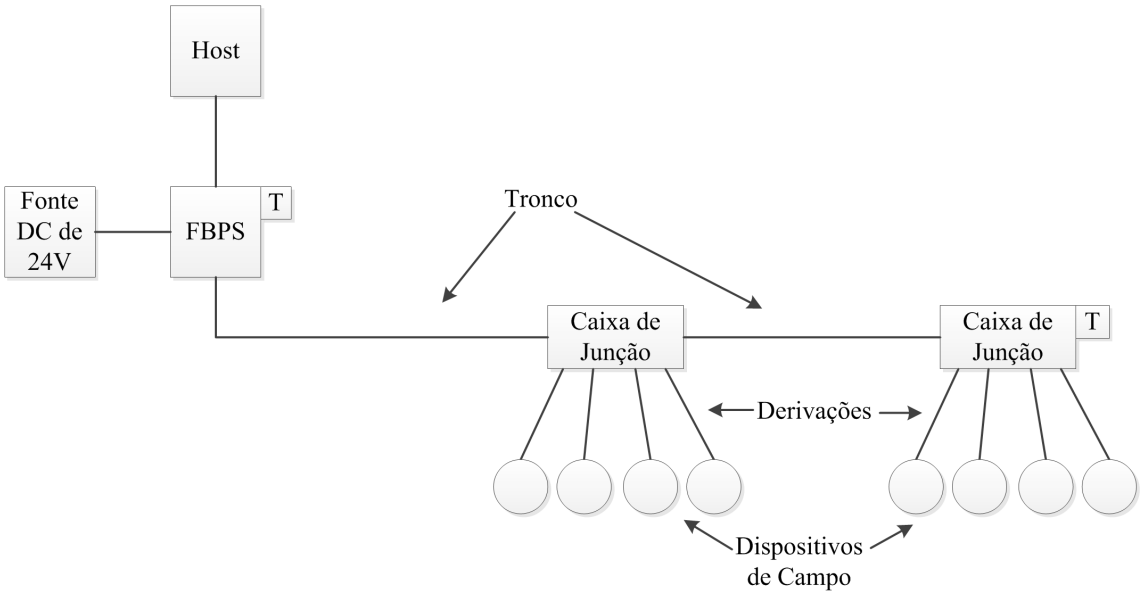

Figura A.2: Topologia recomendada na literatura para uma rede FF. O elemento com um "T" representa um terminador.

- Fonte de Alimentação DC;
- Acoplador Fieldbus (FBPS, do inglês *FieldBus Power Supply*);
- Caixas de Junção ou Protetores de Segmento (Caixas de Junção com proteção contra sobrecorrente);
- Barreiras de Segurança Intrínseca, quando em áreas classificadas. Áreas classificadas limitam significativamente a energia da rede e com isso restringem o número de dispositivos por rede;
- Dispositivos de Campo, como transmissores e posicionadores de válvula;
- Terminadores.

As restrições ao projeto, desconsiderando projetos em áreas classificadas, são [15]:

1. Até 16 dispositivos por rede;
2. Tensão e comprimento da rede;
3. Reserva;
4. Criticidade de válvulas;
5. Macrociclo.

A restrição de 16 dispositivos por rede se deve principalmente ao fato de que, segundo [41], muitos fabricantes providenciam servidores que apenas suportam 16 dispositivos por rede.

O item 2 apenas sugere que haja 20% de folga na rede para futuras ampliações. Ou seja, num projeto inicial com 16 dispositivos por rede, é recomendável considerar apenas 12 (16 - 3.2, aproximando para baixo) no projeto. O ítem 3 discorre sobre certas válvulas que, devido a sua importância no processo, devem ficar isoladas em redes dedicadas.

A.2.1 Queda de Tensão

No caso geral, redes FF são limitadas pelas restrições 1 e 4. A restrição 1 enuncia duas regras principais: a primeira, referente ao comprimento da rede, exige que o comprimento do tronco somado ao somatório dos comprimentos das ramificações, não deve ultrapassar 1900m, com cada ramificação limitada em 120m. O segundo, que a tensão de operação em cada dispositivo de campo seja superior à mínima exigida. Via de regra, o projeto considera folgas de cerca de 20%, com o valor mínimo definido por 11V ou 12V, superiores ao mínimo por norma, 9V.

Para se ter uma noção do número de dispositivos que pode haver numa rede considerando apenas a restrição 1, pode-se fazer uma análise de pior caso. Segundo [41], a tensão de operação num dispositivo i, numa topologia tipo pé de galinha, é dada por

$$V_D = V_{PS} - \left(2R_T L_T \sum_{j \in rede} I_{D_j} + 2R_S L_S I_{D_i} \right) > V_{min}, \tag{A.1}$$

onde V_D é a tensão DC disponível no dispositivo, V_min é a mínima aceitável, V_PS é a fornecida pela FBPS, I_{D_j} é a corrente consumida pelo dispositivo j, R_T e R_S são as resistências por comprimento do tronco e da ramificação, respectivamente, e L_T e L_S são seus comprimentos. Considerando o caso marginal de um tronco de 1900m, cabo 16AWG (13.17Ω/km) e ramificações de 0m, com uma FBPS de 28V e N dispositivos consumindo 17mA (corrente média de dispositivos FF segundo [15]), temos a seguinte equação:

$$V_D = 28V - 0.85N > V_{min}. \tag{A.2}$$

Considerando 12V de tensão mínima, isso resulta em 18 dispositivos, no máximo, que é superior a 16. Com uma rede menor, de 1000m, a equação se torna

$$V_D = 28V - 0.45N > V_{min} \tag{A.3}$$

e o número máximo de dispositivos já chegaria a 35, próximo do limite teórico definido pela norma IEC 61158. Obviamente, deve-se garantir que a FBPS providencie a corrente necessária para todos os dispositivos.

Conclui-se até o momento que a queda de tensão, com o uso de FBPS e cabo apropriados, não é fator limitante na grande maioria dos casos. O gráfico da Figura A.3 ilustra o número máximo de dispositivos numa rede em função do comprimento do tronco, baseado nessa análise de pior caso, considerando um FBPS de 500mA. Quedas de tensão nos tronco não costumam ultrapassar 0.1V e, na prática, podem ser absorvidas na margem de erro.

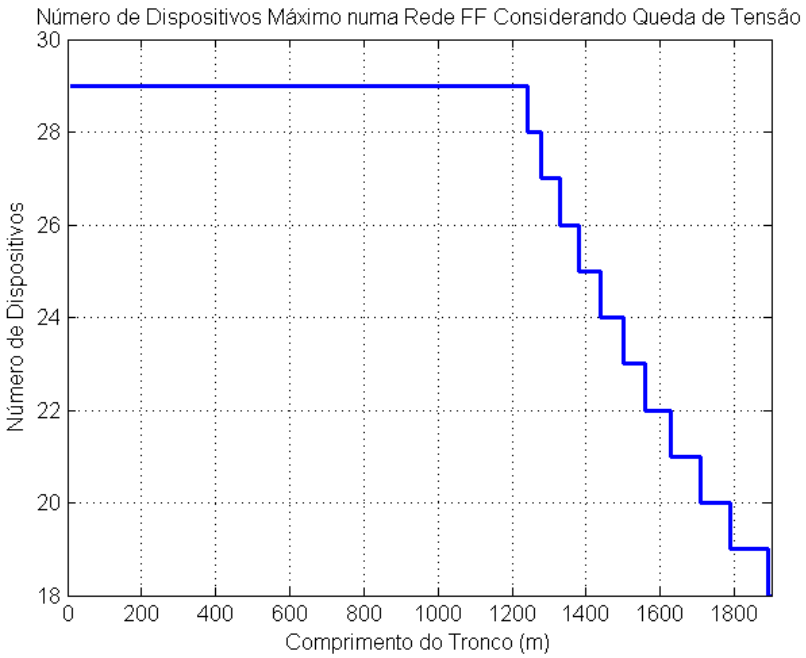

Figura A.3: Gráfico comparando o número máximo de dispositivos com o tamanho do tronco, considerando a análise de queda de tensão desta seção.

A.2.2 Macrociclo

A última restrição, a restrição 4, refere-se aos limites da banda. Segundo [15], a rede deve acomodar todas as execuções das funções e comunicações dentro do seu Macrociclo. Ou seja, deve haver tempo para todas as tarefas periódicas de todos os dispositivos da rede e todas as comunicações. Além disso, deve haver 40% a 50% de tempo disponível para comunicações acíclicas em projetos novos (até 30% em redes operacionais) para garantir que funcionalidades sob demanda (mudanças de configuração, requisição de parâmetros, etc.) serão executadas.

Uma regra dada como recomendação em [15] é a seguinte: no máximo 4 malhas de controle para uma rede com macrociclo de 1000 ms, 2 malhas para 500 ms e 1 malha

para 250 ms. Esses são os tempos mais comuns de ciclo para redes com controle. Observe que isso limita pesadamente a rede. Desconsiderando instrumentos para monitoração, e levando em conta pares transmissor/posicionador de válvula por malha, as respectivas redes teriam 8, 4 e 2 dispositivos.

Para entender melhor essas recomendações, é necessária uma breve introdução ao tema Configuração no Ambiente FF. É o que será explicado a seguir.

A.3 Configuração no ambiente FF

O protocolo FOUNDATION™ Fieldbus H1 define blocos de função que são usados para monitoramento e controle de processos. Os blocos de função podem ser executados nos dispositivos de campo, isto é, transmissores, posicionadores de válvula, etc.. Cada dispositivo de campo tem seus blocos de função disponíveis, cada qual possuindo tempos de execução fixos e conhecidos, variando de 10 a 500 ms. Blocos de função incluem AI, Saída Analógica (AO, do inglês *Analog Output*), PID, ISEL, Divisor de Saída (SPLTR, do inglês *Output Splitter*), Integrador (IT) e outras funções usadas em controle de processos. A configuração dos dispositivos numa rede e das malhas de controle pode ser feita *offline*, antes de colocar a rede em operação. Toda a informação necessária dos dispositivos pode ser retirada de arquivos chamados Capability Files (CFs), que descrevem suas funcionalidades.

A configuração de malhas de controle, portanto, é feita na interface do SDCD, através de ligações entre blocos de função. É possível usar blocos de função do controlador para comunicação com os blocos de função FF. Na janela de configuração, o usuário determina os blocos usados, a alocação deles, ou seja, que dispositivo os irá executar, e as ligações lógicas entre os blocos. Numa simples malha PID, por exemplo, conecta-se o valor de OUT do bloco AI ao IN do PID, o OUT do PID ao IN do AO e o BCKAL_OUT do AO ao BCKAL_IN do PID. A Figura A.4 ilustra essa configuração no ambiente DeltaV 8.3. Nesse caso, o bloco AI foi alocado ao transmissor de nome AI1, enquanto os blocos PID e AO foram alocados ao posicionador de nome AO1.

Após essa ligação entre blocos, o usuário deve parametrizá-los. Isso inclui determinações de valores mínimo e máximo de cada variável, unidades de engenharia, níveis de alarme, ganhos padrão do PID e o que mais estiver disponível em cada bloco. Por fim, o usuário determina o ciclo de execução da malha de controle. Para colocar as malhas em operação, além de os dispositivos estarem fisicamente conectados ao SDCD, é necessária uma operação de download.

Por trás das ligações lógicas dos blocos de função existe a necessidade de comunicação entre os dispositivos que serão encarregados de executar tais blocos. Além disso, existe uma ordenação natural proveniente das malhas de controle. No caso

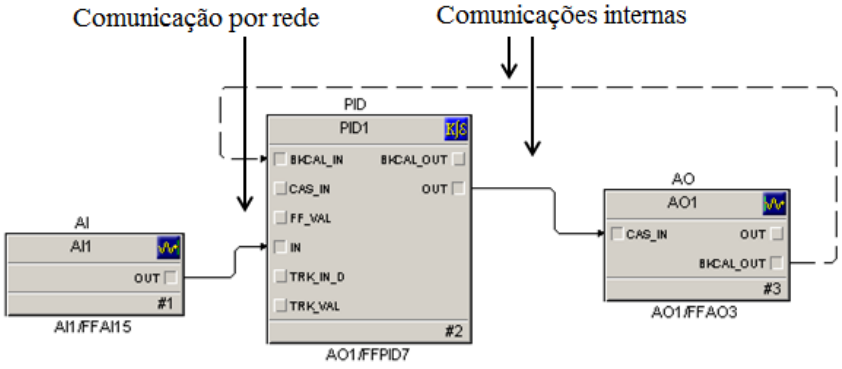

Figura A.4: Uma malha PID configurada no ambiente DeltaV 8.3.

do PID, a cada ciclo o transmissor AI1 executa seu bloco AI, envia, pela rede FF, sua saída ao transmissor AO1. Este por sua vez executa o bloco PID, usa internamente sua saída ao bloco AO, executa-o e usa internamente o *readback* do AO para o PID do ciclo seguinte, onde todas essa operações são refeitas com a periodicidade determinada. Existe um elemento da rede que precisa conhecer essa sequência de operações, desta e de outras malhas, de forma a gerenciar as execuções e determinar qual dispositivo pode usar a rede e quando.

Por isso tudo, torna-se necessário o escalonamento das tarefas offline. Isso é feito no sistema de controle automaticamente através de algoritmos proprietários, e em geral o usuário pode ajustá-lo manualmente. No momento em que é feito o download, a informação do escalonamento é passada para o LAS, que é o elemento responsável pelo gerenciamento das execuções e comunicações. No início da operação da rede, o LAS passa aos dispositivos o instante e a periodicidade em que eles devem iniciar a execução de cada bloco de função e realizar cada comunicação.

Essa comunicação entre dispositivos, pré-configurada em malhas de controle, é feita através de um CD. Enquanto a comunicação é feita, nenhuma outra comunicação pode ocorrer na rede, naturalmente. Para tornar a comunicação robusta garantindo determinismo, sistemas predeterminam um valor fixo e conservador para a duração do CD, que normalmente varia entre 20 e 30 ms.

Na malha da Figura A.4, com o bloco PID alocado ao posicionador de válvula AO1, pode-se ver que a cada execução ocorre apenas uma comunicação via rede FF. Supondo tempos de execução para AI, PID e AO de 25, 40 e 40 ms respectivamente, com CD de 30ms, uma possível agenda para cada ciclo pode ser vista no gráfico de Gantt da Figura A.5. Supondo agora que o mesmo bloco PID seja alocado no transmissor AI1, e que esse bloco PID leve os mesmos 40ms no transmissor, duas comunicações via rede acontecem: o OUT do PID, no transmissor, vai até o AO do posicionador, enquanto o *readback* do AO vai pela rede até o transmissor, para seu

PID. Um escalonamento possível nesse caso é visto no gráfico de Gantt da Figura A.6. Observe, portanto, como a alocação de blocos de função a dispositivos interfere drasticamente no escalonamento das tarefas e na ocupação da rede.

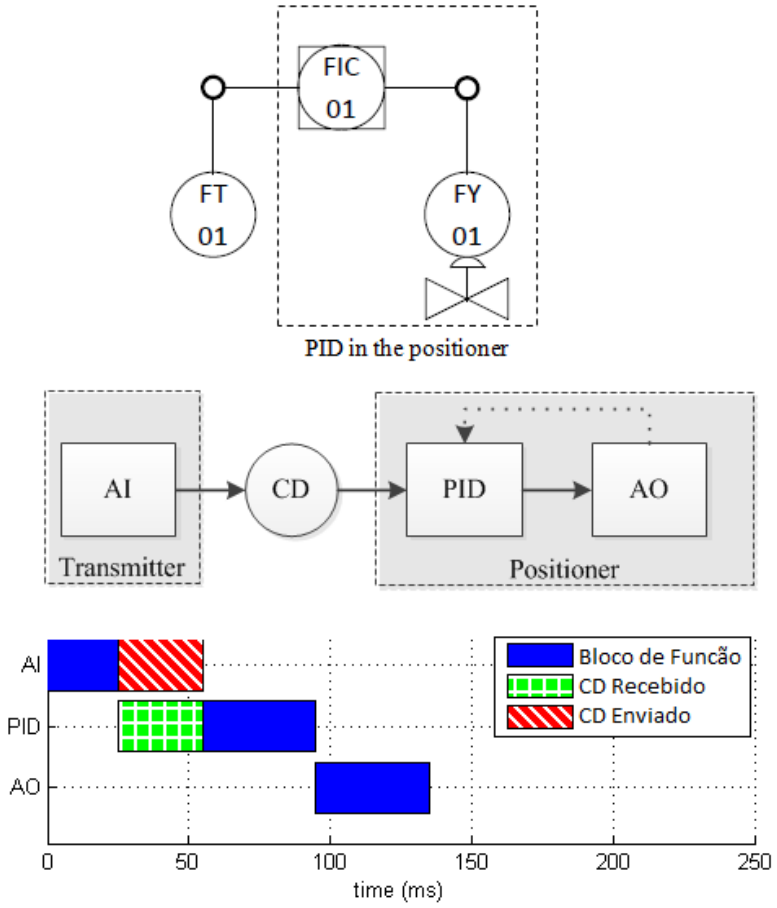

Figura A.5: Um diagrama P&I simplificado representando um PID executado no posicionador (topo) e a representação via gráfico de Gantt de uma possível agenda associada (abaixo).

Gráficos de Gantt como os das figuras A.6 e A.5 são comumente usados para representar agendas de macrociclo. Neste caso, cada linha do gráfico representa uma tarefa, precedida/seguida ou não por um CD recebido/enviado. Dois blocos de CD no mesmo intervalo de tempo representam a mesma comunicação. É importante observar que alguns CDs não possuem receptor ou transmissor no gráfico. Isso acontece porque o gráfico apenas representa recepções ou transmissões quando as tarefas são precedidas ou seguidas por um CD. Quando não há tarefa que precise de uma dada transmissão no atual macrociclo, como o *readback* da figura A.6. Em outros casos, o sistema *host* pode enviar um valor para um dispositivo para uma tarefa específica. Com isso, apenas a recepção do CD por essa tarefa aparece, sem que nenhuma transmissão seja representada no gráfico. Note que uma malha de controle típica, em função dos *readbacks*, representa um grafo direcionado cíclico. É

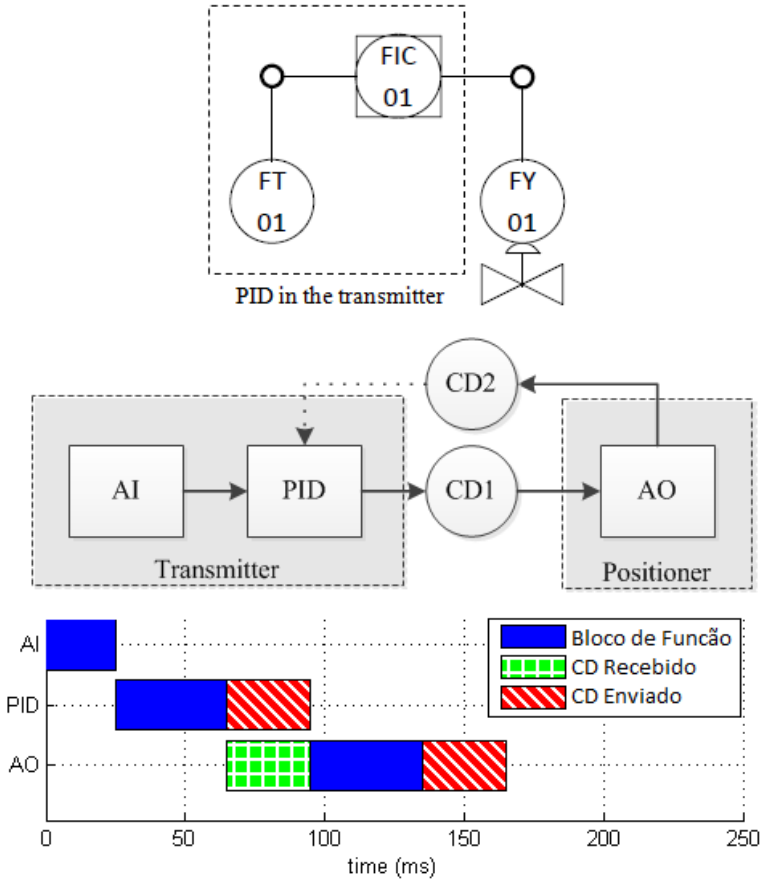

Figura A.6: Um diagrama P&I simplificado representando um PID executado no transmissor (topo) e a representação via gráfico de Gantt de uma possível agenda associada (abaixo).

necessário abrir os ciclos para ser possível escalonar as tarefas. Isso será explicado no Capítulo 3.

Apesar de a escolha do dispositivo que vai executar qual bloco seja crítica para o tamanho da agenda e a ocupação de banda com comunicações P/A, esse trabalho não lida com este critério. Obviamente, em termos de otimização de macrociclo, as tarefas deveriam ser alocadas de tal maneira que o mínimo de comunicações seja requerido. No exemplo acima, alocar o PID no posicionador é indubitavelmente a melhor escolha sob esse ponto de vista. No entanto, para alocação de blocos funcionais, segurança e disponibilidade operacional do sistema são critérios mais importantes. Dito isto, esse trabalho apenas trata da ordem das tarefas e da determinação do instante de tempo de execução de cada uma delas. A alocação dos blocos funcionais a dispositivos é considerada predeterminada.

A.3.1 Limitações da Rede pelo Macrociclo

Uma vez explicados como funcionam a configuração de malhas e o escalonamento de tarefas, é possível entender as recomendações limitantes em [15].

Para ilustrar, analisa-se a alocação das tarefas. Para uma malha PID simples, já foram explicados os casos em que o bloco PID é alocado no posicionador (um CD por malha por ciclo) e no transmissor (dois CDs por malha por ciclo). Um terceiro caso é o controle PID ser feito no controlador. Neste caso, o transmissor envia o OUT do AI para o controlador, o controlador calcula a saída e a passa para o transmissor e este ainda passa ao controlador o valor do *readback*. No total, três comunicações são realizadas apenas para uma malha. Se for considerado um tempo de 30ms por CD, como é o caso padrão do DeltaV 8.3, isso significa 120ms de ocupação de banda por malha por ciclo. Isso é 40% de um ciclo de 250ms. Duas malhas nesse ciclo chegariam a 80% de ocupação, o que é impeditivo pela regra 4 de projetos [15]. Raciocínio análogo é feito para ciclos de 500ms, onde três malhas chegariam a 60%, e para um ciclo de 1000ms, onde 5 malhas, chegariam a 50% de ocupação, que é o limite. Vale lembrar que nem toda malha é um PID simples, o que pode acarretar em comunicações extras e maior ocupação da banda. A conclusão é que, nada se sabendo sobre a alocação das funções de controle, a regra geral enunciada na Seção A.2.2 é uma regra simples mas conservadora o suficiente para garantir que as redes projetadas por ela funcionem bem. Outra conclusão é que, nos casos em que se sabe onde são executadas as mesmas, pode valer a pena investigar mais a fundo e conseguir redes com mais dispositivos, mais bem aproveitadas.

O segundo ponto de discussão se refere ao escalonamento. Imagine uma regra de escalonamento ingênua que simplesmente sequencia os blocos de função e as comunicações que a rede terá de comportar. Para o escalonamento ilustrado na Figura A.5, chega-se a um tempo total entre o início e o fim de execuções de uma malha de 25+30+40+40 = 135ms. Colocando duas malhas em sequência chega-se ao tempo total de 270ms, que é maior que 250ms. Portanto, sob essa ótica também é impossível carregar uma rede com duas malhas de ciclo igual a 250ms cada. Raciocínios análogos podem ser aplicados aos ciclos de 500 e 1000 ms. Conclui-se que a regra 4 de projetos de redes FF [15] consegue abranger de forma conservadora sistemas com algoritmos de escalonamento ineficientes.

No entanto, sob o ponto de vista do problema de escalonamento, por mais que comunicações não possam ocorrer em paralelo, execuções de blocos de função de diferentes dispositivos podem fazê-lo. E um bom algoritmo de escalonamento deve tirar proveito disso. Por mais que pareça simples imaginar uma rotina de escalonamento eficiente que aproveite esses paralelismos, é importante notar que muitas malhas de controle vão além de um simples PID. O usuário tem a flexibilidade de

compor suas malhas e ligar seus blocos de função da maneira que for mais apropriada para o processo a ser controlado. Isso aumenta a complexidade do escalonamento e traz a necessidade de algoritmos que considerem casos gerais, sem supor estruturas prévias para malhas de controle. É esse caso geral que é abordado neste trabalho.

Apêndice B

Códigos Fonte para GLPK

B.1 Caso Monociclo

B.1.1 Modelos

Modelo por Pontos de Evento

```
###############################################
###                 SETS                    ###
###############################################

set I;

set J 'devices'; # 0 é a rede

param net = 0; # rede

set M; # cada malha tem suas tarefas independentemente

set Ij {j in J} within I; # tarefas do device j

set N {j in J} := 1..card(Ij[j]);

set L {m in M} within {I,I}; # Ligações entre tarefas

set Lj {j in J, m in M} within {Ij[j],Ij[j]} default {{},{}}; # sequências
entre tarefas num mesmo device
```

```
set E within {I,I} default {{},{}}; # Pares de tarefas equivalentes

set Ir within Ij[net] default {}; # Readback communications

set Ce {i in Ir} within {I,I}; # De onde vem e para onde vai o readback

##############################################
###              PARAMETERS                ###
##############################################

param H 'time horizon'; # Macrociclo requisitado

param Te 'execution time' {i in I};

param alpha 'tradeoff' >= 0, <= 1;

param beta 'tradeoff2' >= 0, <= 1;
#check alpha + beta <= 1; depois!

param R 'comm length' >=0, <=H;

param a 'loop weight' {m in M} >=0;

##############################################
###              VARIABLES                 ###
##############################################

var wv 'beginning of task' {j in J, i in Ij[j], n in N[j]} binary;

var y 'before or after' {i in Ir} binary;

var Ts 'start time' {i in I} >=0, <=H;

var Tf 'final time' {i in I} >=0, <=H;

var TsCDs 'first CD' >=0, <=H;

var TfCDs 'last CD' >=0, <=H;
```

```
var TF 'final time of all' >=0, <= H;

var D {m in M};

###############################################
###              CONSTRAINTS                ###
###############################################

### ALLOCATION CONSTRAINTS

subject to alloc 'allocation' {j in J, n in N[j]}:
sum{i in Ij[j]}(wv[j,i,n]) = 1; # em cada ponto de evento n, um
device só pode executar uma tarefa

### EXECUTION CONSTRAINTS - Minha modificação

subject to onexec 'one execution per macrocycle' {j in J, i in Ij[j]}:
sum{n in N[j]}(wv[j,i,n]) = 1; # cada tarefa é executada uma e
somente uma vez no período

### SEQUENCE CONSTRAINTS

subject to samedev 'different tasks same device'
{j in J, i in Ij[j], i2 in Ij[j], n in N[j]: n < card(N[j]) and
i2 <> i}:
Ts[i2] >= Tf[i] - (H)*(2 - wv[j,i,n] - wv[j,i2,n+1]); # A próxima
tarefa só pode ser executada ao final da atual

subject to seqtime 'task times in sequence' {m in M, (i,i2) in L[m]}:
Ts[i2] >= Tf[i];

subject to seqlink 'sequence of task events on same device' {j in J,
m in M,(i,i2) in Lj[j,m], n in N[j]}:
sum{n2 in N[j]: n2 > n}(wv[j,i2,n2]) >= wv[j,i,n]; # A próxima
tarefa só pode ser executada depois da anterior na sequência

### FINAL TIME CONSTRAINTS

subject to endtimes {i in I}:
```

```
Tf[i] = Ts[i] + Te[i];

subject to makespan {i in I}:
TF >= Tf[i];

### COMPEL DATA BOUNDS CONSTRAITS

subject to firstCD {i in Ij[net]}:
TsCDs <= Ts[i];

subject to lastCD {i in Ij[net]}:
TfCDs >= Tf[i];

### PROJECT CONSTRAINTS

subject to project:
TfCDs - TsCDs <= R;

### EQUIVALENCE CONSTRAINTS

subject to equivtime {(i,i2) in E}:
Ts[i2] >= Ts[i];

subject to equivevent {(i,i2) in E, j in J, n in N[j]: (i in Ij[j])
and (i2 in Ij[j])}:
sum{n2 in N[j]: n2 > n}(wv[j,i2,n2]) >= wv[j,i,n];

### RECYCLE CONSTRAINTS

subject to beforetask {i in Ir, (iA,iB) in Ce[i]}:
Ts[iA] >= Tf[i] - H*y[i];

subject to aftertask {i in Ir, (iA,iB) in Ce[i]}:
Ts[i] >= Tf[iB] - H*(1-y[i]);

### DELAYS

subject to delays {m in M}:
D[m] = sum{(i,i2) in L[m]}(Ts[i2] - Ts[i]);
```

```
##############################################
###              OBJECTIVES                ###
##############################################

minimize tradeoff 'trade-off between objectives':
alpha*(TfCDs - TsCDs) + beta*sum{m in M,(i,i2) in L[m]}
(a[m]*(Ts[i2] - Ts[i])) + (1-alpha-beta)*TF;

end;
```

Modelo por Sobreposição

```
##############################################
###                SETS                    ###
##############################################

set I;

set J 'devices'; # 0 é a rede

param net = 0; # rede

set M; # cada malha tem suas tarefas independentemente

set Ij {j in J} within I; # tarefas do device j

set L {m in M} within {I,I}; # Ligações entre tarefas

set E within {I,I} default {{},{}}; # Pares de tarefas equivalentes

set Ir within Ij[net] default {}; # Readback communications

set Ce {i in Ir} within {I,I}; # De onde vem e para onde vai o readback

##############################################
###              PARAMETERS                ###
##############################################
```

```
param H 'time horizon'; # Macrociclo requisitado

param Te 'execution time' {i in I};

param alpha 'tradeoff' >= 0, <= 1;

param beta 'tradeoff2' >= 0, <= 1;
#check alpha + beta <= 1; depois!

param R 'comm length' >=0, <=H;

param a 'loop weight' {m in M} >=0;

###############################################
###              VARIABLES                  ###
###############################################

var o 'overlap' {j in J, i in Ij[j], i2 in Ij[j]} binary;

var y 'rback before or after' {i in Ir} binary;

var Ts 'start time' {i in I} >=0, <=H;

var Tf 'final time' {i in I} >=0, <=H;

var TsCDs 'first CD' >=0, <=H;

var TfCDs 'last CD' >=0, <=H;

var TF 'final time of all' >=0, <= H;

###############################################
###              CONSTRAINTS                ###
###############################################

### SEQUENCE CONSTRAINTS

subject to samedev 'different tasks same device' {j in J, i in Ij[j],
```

```
i2 in Ij[j]: i2 <> i}:
Tf[i2] - Ts[i] <= H*o[j,i,i2];

subject to seqtime 'task times in sequence' {m in M, (i,i2) in L[m]}:
Ts[i2] >= Tf[i];

### OVERLAP CONSTRAINTS

subject to overlap 'overlap of tasks in a device' {j in J, i in Ij[j],
i2 in Ij[j]: i <> i2}:
o[j,i,i2] + o[j,i2,i] <= 1;

### FINAL TIME CONSTRAINTS

subject to endtimes {i in I}:
Tf[i] = Ts[i] + Te[i];

subject to makespan {i in I}:
TF >= Tf[i];

### COMPEL DATA BOUNDS CONSTRAITS

subject to firstCD {i in Ij[net]}:
TsCDs <= Ts[i];

subject to lastCD {i in Ij[net]}:
TfCDs >= Tf[i];

### PROJECT CONSTRAINTS

subject to project:
TfCDs - TsCDs <= R;

### EQUIVALENCE CONSTRAINTS

subject to equivtime {(i,i2) in E}:
Ts[i2] >= Ts[i];

### RECYCLE CONSTRAINTS
```

```
subject to beforetask {i in Ir, (iA,iB) in Ce[i]}:
Ts[iA] >= Tf[i] - H*y[i];

subject to aftertask {i in Ir, (iA,iB) in Ce[i]}:
Ts[i] >= Tf[iB] - H*(1-y[i]);

###############################################
###              OBJECTIVES                 ###
###############################################

minimize tradeoff 'trade-off between objectives':
alpha*(TfCDs - TsCDs) + beta*sum{m in M,(i,i2) in L[m]}
(a[m]*(Ts[i2] - Ts[i])) + (1-alpha-beta)*TF;

end;
```

B.1.2 Dados para os Estudos de Caso

Caso I

```
###############################################
###                 SETS                    ###
###############################################

set I := AI1 AI2 AI3 AI41 AI42 AI5 AO1 AO2 AO3 AO4 AO5 CD1 CD2 CD3 CD4 CD5
 CD6 CD7 CD8;

set J := 0 AI1 AI2 AI3 AI4 AI5 AO1 AO2 AO3 AO4 AO5;

set Ij[AI1] := AI1;
set Ij[AI2] := AI2;
set Ij[AI3] := AI3;
set Ij[AI4] := AI41 AI42;
set Ij[AI5] := AI5;
set Ij[AO1] := AO1;
set Ij[AO2] := AO2;
set Ij[AO3] := AO3;
set Ij[AO4] := AO4;
```

```
set Ij[AO5] := AO5;
set Ij[0] := CD1 CD2 CD3 CD4 CD5 CD6 CD7 CD8;

set M := 1 2 3 4;

set L[1] := (AI1,CD1) (CD1,AO1);
set L[2] := (AI2,CD2) (AI3,CD3) (CD2,AO2) (CD3,AO2);
set L[3] := (AI41,AI42) (AI42,CD4) (AI42,CD5) (CD4,AO3) (CD5,AO4);
set L[4] := (AI5,CD8) (CD8,AO5);

set E := (AI2,AI3) (CD2,CD3) (CD4,CD5) (AO3,AO4);

set Ir := CD6 CD7;

set Ce[CD6] := (AI42,AO3);
set Ce[CD7] := (AI42,AO4);

##############################################
###              PARAMETERS                ###
##############################################

param H := 555;

param net := 0;

param R := 400;

param Te :=  AI1 25
AI2 30
AI3 30
AI41 45
AI42 20
AI5 30
AO1 90
AO2 80
AO3 40
AO4 40
AO5 55
CD1 30
```

```
CD2  30
CD3  30
CD4 30
CD5 30
CD6 30
CD7 30
CD8 30;

param a :=  1 1.0
2 1.0
3 1.0
4 1.0;

param alpha := 0.9; # weight grouping
param beta := 0.099; # weight latency
```

Caso II

```
#############################################
###                SETS                   ###
#############################################

set I := AI1 AI2 AI3 AI4 AI51 AI52 AO11 AO12 AO21 AO22 AO31 AO32 AO41 AO42
 AO43 AO5 CD1 CD2 CD3 CD4 CD5 CD6 CD8 CD9 CD10 CD11;

set J := 0 AI1 AI2 AI3 AI4 AI5 AO1 AO2 AO3 AO4 AO5;

set Ij[AI1] := AI1;
set Ij[AI2] := AI2;
set Ij[AI3] := AI3;
set Ij[AI4] := AI4;
set Ij[AI5] := AI51 AI52;
set Ij[AO1] := AO11 AO12;
set Ij[AO2] := AO21 AO22;
set Ij[AO3] := AO31 AO32;
set Ij[AO4] := AO41 AO42 AO43;
set Ij[AO5] := AO5;
set Ij[0] := CD1 CD2 CD3 CD4 CD5 CD6 CD8 CD9 CD10 CD11;
```

```
set M := 1 2 3;

set L[1] := (AI1,CD1) (CD1,AO11) (AO11,AO12) (CD2,AO12);
set L[2] := (AI2,CD3) (CD3,AO21) (AO21,AO22) (CD4,AO22);
set L[3] := (AI3,CD5) (AI4,CD6) (CD5,AO31) (CD6,AO31) (AO31,AO32)
 (AO31,CD8) (AI51,AI52) (CD8,AI52) (AI52,CD9) (CD9,AO41) (AO41,AO42)
 (AO42,AO43) (AO42,CD10) (CD10,AO5);

set E := (AI1,AI2) (CD1,CD3) (AO11,AO21) (AO12,AO22) (CD2,CD4) (AI3,AI4)
 (CD5,CD6);

set Ir := CD11;

set Ce[CD11] := (AO42,AO5);

###############################################
###              PARAMETERS                 ###
###############################################

param H := 480;

param net := 0;

param R := 400;

param Te :=  AI1  30
AI2 30
AI3 30
AI4 30
AI51 30
AI52 30
AO11 30
AO12 50
AO21 30
AO22 50
AO31 25
AO32 55
AO41 30
AO42 25
```

```
AO43 25
AO5 25
CD1 30
CD2  30
CD3  30
CD4 30
CD5 30
CD6 30
CD8 30
CD9 30
CD10 30
CD11 30;

param a :=  1  1.0
2 1.0
3 1.0;

param alpha := 0.9; # weight grouping
param beta := 0.099; # weight latency

### FOR EVENT POINTS MODEL ONLY
set Lj[AI5,3] := (AI51,AI52);
set Lj[AO1,1] := (AO11,AO12);
set Lj[AO2,2] := (AO21,AO22);
set Lj[AO3,3] := (AO31,AO32);
set Lj[AO4,3] := (AO41,AO42) (AO42,AO43);
set Lj[0,3] := (CD5,CD8) (CD6,CD8) (CD8,CD9) (CD9,CD10);

end;
```

B.2 Caso Multiciclo

B.2.1 Modelos

Modelo por Pontos de Evento

```
##############################################
###                SETS                    ###
```

```
##############################################

set I;

set J 'devices'; # 0 é a rede

param net = 0; # rede

set C {i in I}; # Ciclos num macrociclo para uma dada tarefa

set M; # cada malha tem suas tarefas independentemente

set Ij {j in J} within I; # tarefas do device j

set Im {m in M} within I; # tarefas da malha m

set N {j in J} := 1..(sum{i in Ij[j]}(card(C[i])));

set L {m in M} within {Im[m],Im[m]}; # Ligações entre tarefas

set Lj {j in J, m in M} within {Ij[j],Ij[j]} default {{},{}}; # sequências
 entre tarefas num mesmo device

set E within {I,I} default {{},{}}; # Pares de tarefas equivalentes

set Ir within Ij[net] default {}; # Readback communications

set Re {i in Ir} within {I,I}; # De onde vem e para onde vai o readback

set Rb {i in Ir} within Ij[net]; # Comunicações no 'meio'

##############################################
###              PARAMETERS                ###
##############################################

param H 'time horizon'; # Macrociclo requisitado

param Hi {i in I}; # Cicle time for task i
```

```
param Te 'execution time' {i in I};

param alpha 'tradeoff' >= 0, <= 1;

param beta 'tradeoff2' >= 0, <= 1;
#check alpha + beta <= 1; depois!

param r 'comm ratio' >=0, <=1;

param a 'loop weight' {m in M} >=0;

param gamma 'gap weight' >=0;

param mono 'monocycle loop' {m in M} binary;

###############################################
###              VARIABLES                  ###
###############################################

var wv 'beginning of task' {j in J, i in Ij[j], c in C[i], n in N[j]}
binary;

var y 'rback before or after' {i in Ir} binary;

var Tsb 'start time' {i in I} >=0, <=Hi[i];

var Tfb 'final time' {i in I} >=0, <=Hi[i];

var Ts {i in I, c in C[i]} >=0, <=H;

var Tf {i in I, c in C[i]} >=0, <=H;

var TF 'final time of all' >=0, <= max{i in I}(Hi[i]);

# New grouping optimization technique

var g 'gap' {n in N[net]: n < card(N[net])} binary;

# Cycle basis
```

```
var cb {i in I} >=1, <= (max{i2 in I, m in M: i in Im[m] and i2 in Im[m]}
(card(C[i2])))/(min{i2 in I, m in M: i in Im[m] and i2 in Im[m]}
(card(C[i2]))) integer;

var G >= 0, <= card(N[net])-1;

var D {m in M} >=0;

##############################################
###              CONSTRAINTS               ###
##############################################

### SINGLE USE CONSTRAINTS

subject to alloc 'allocation' {j in J, n in N[j]}:
sum{i in Ij[j], c in C[i]}(wv[j,i,c,n]) = 1; # em cada ponto de
evento n, um device só pode executar uma tarefa

### EXECUTION CONSTRAINTS - Minha modificação

subject to onexec 'one execution per macrocycle' {j in J, i in Ij[j],
c in C[i]}:
sum{n in N[j]}(wv[j,i,c,n]) = 1; # cada tarefa é executada uma
e somente uma vez no período

### SEQUENCE CONSTRAINTS

subject to samedevcut 'precedent cycles' {j in J, i in Ij[j], c in C[i],
i2 in Ij[j], c2 in C[i2], n in N[j]: (i2 <> i or c <> c2) and
(Hi[i2]*c2 <= Hi[i]*(c-1))}:
sum{n2 in N[j]: n2 > n}(wv[j,i,c,n2]) >= wv[j,i2,c2,n];
# i2 ends before i starts

subject to samedev 'different tasks same device' {j in J, i in Ij[j],
c in C[i], i2 in Ij[j], c2 in C[i2], n in N[j]: (i2 <> i or c <> c2)
and n < card(N[j])}: # and (Hi[i2]*c2 > Hi[i]*(c-1) + 1)}:
Ts[i2,c2] >= Tf[i,c] - H*(2 - wv[j,i,c,n] - wv[j,i2,c2,n+1]);
```

```
subject to seqtime 'task times in sequence' {m in M, (i,i2) in L[m]}:
Tsb[i2] >= Tfb[i];

subject to seqlink 'sequence of task events on same device' {j in J,
m in M,(i,i2) in Lj[j,m], c in C[i], n in N[j]: card(C[i]) = card(C[i2])}:
sum{n2 in N[j]: n2 > n}(wv[j,i2,c,n2]) >= wv[j,i,c,n];
# A próxima tarefa só pode ser executada depois da anterior na sequência

### NEXT CYCLE CONSTRAINTS

subject to cyclenext {j in J, i in Ij[j], c in C[i], n in N[j]:
 c < card(C[i])}:
sum{n2 in N[j]: n2 > n}(wv[j,i,c+1,n2]) >= wv[j,i,c,n];
# A de c+1 vem depois da de c

### FINAL TIME CONSTRAINTS

subject to endtimes {i in I, c in C[i]}:
Tf[i,c] = Ts[i,c] + Te[i];

subject to endtimebasis {i in I}:
Tfb[i] = Tsb[i] + Te[i];

subject to makespan {i in I}:
TF >= Tf[i,1];

### EQUIVALENCE CONSTRAINTS

subject to equivtime {(i,i2) in E}:
Tsb[i2] >= Tsb[i];

subject to equivevent {(i,i2) in E, j in J, n in N[j], c in C[i]:
(i in Ij[j]) and (i2 in Ij[j]) and card(C[i]) = card(C[i2])}:
sum{n2 in N[j]: n2 > n}(wv[j,i2,c,n2]) >= wv[j,i,c,n];

### RECYCLE CONSTRAINTS

subject to beforetask {i in Ir, (iA,iB) in Re[i]}:
Tsb[iB] >= Tfb[i] - Hi[i]*y[i];
```

```
subject to aftertask {i in Ir, (iA,iB) in Re[i]}:
Tsb[i] >= Tfb[iA] - Hi[i]*(1-y[i]);

### COMMUNICATION GAP CONSTRAINTS

subject to gapexist {i in Ij[net], c in C[i], i2 in Ij[net], c2 in C[i2],
n in N[net]: n < card(N[net]) and (i <> i2 or c <> c2)}:
Tf[i,c] >= Ts[i2,c2] - H*(2 - wv[net,i,c,n] - wv[net,i2,c2,n+1] +
 g[n]);

### BASIC CYCLE CONSTRAINTS

subject to basiccycle {i in I}:
0 >= Tsb[i] - Hi[i]*cb[i] >= -Hi[i];

subject to samesubloop {m in M, i in Im[m], i2 in Im[m]: i <> i2 and
card(C[i]) = card(C[i2])}:
cb[i] = cb[i2];

### PERIODICITY CONSTRAINTS

subject to periodic {i in I, c in C[i]: c < card(C[i])}:
Ts[i,c+1] = Ts[i,c] + Hi[i];

subject to basic {i in I, c in C[i]}:
Ts[i,c] = Tsb[i] + Hi[i]*(c-cb[i]);

### CYCLE LIMIT CONSTRAINTS

subject to limitstart {i in I, c in C[i]}:
Hi[i]*(c-1) <= Ts[i,c] <= Hi[i]*c;

subject to limitend {i in I, c in C[i]}:
Hi[i]*(c-1) <= Tf[i,c] <= Hi[i]*c;

### MONOCYCLE CUT

subject to monocycleloop {m in M, i in Im[m]: mono[m] = 1}:
```

```
cb[i] = 1;

### GROUPING CONSTRAINTS

subject to grouping:
G = sum{n in N[net]: n < card(N[net])}(g[n]);


### DELAY CONSTRAINTS

subject to delays {m in M}:
D[m] = sum{(i,i2) in L[m]}(Tsb[i2] - Tfb[i]);

### SAME LOOP CUTS

subject to nextloopcycles {j in J, m in M, i in Im[m], c in C[i],
i2 in Im[m], c2 in C[i], n in N[j]: i in Ij[j] and i2 in Ij[j] and
i <> i2 and card(C[i]) = card(C[i2]) and c < card(C[i])}:
sum{n2 in N[j]: n2 > n}(wv[j,i2,c+1,n2]) >= wv[j,i,c,n];


##############################################
###              OBJECTIVES                ###
##############################################

minimize tradeoff 'trade-off between objectives':
alpha*gamma*G + beta*sum{m in M}(a[m]*D[m]) + (1-alpha-beta)*TF;

end;
```

Modelo por Sobreposição Básico

```
##############################################
###                 SETS                   ###
##############################################

set I;
```

```
set J 'devices'; # 0 é a rede

param net = 0; # rede

set M; # cada malha tem suas tarefas independentemente

set Ij {j in J} within I; # tarefas do device j

set Im {m in M} within I; # tarefas da malha m

set L {m in M} within {Im[m],Im[m]}; # Ligações entre tarefas

set Lj {j in J, m in M} within {Ij[j],Ij[j]} default {{},{}}; # sequências
entre tarefas num mesmo device

set E within {I,I} default {{},{}}; # Pares de tarefas equivalentes

set Ir within Ij[net] default {}; # Readback communications

set Re {i in Ir} within {I,I}; # De onde vem e para onde vai o readback

### Multicicle Sets

set C {i in I}; # Ciclos num macrociclo para uma dada tarefa

##############################################
###              PARAMETERS              ###
##############################################

param H 'time horizon'; # Macrociclo requisitado

param Hi {i in I}; # Cicle time for task i

param Te 'execution time' {i in I};

param alpha 'tradeoff' >= 0, <= 1;

param beta 'tradeoff2' >= 0, <= 1;
#check alpha + beta <= 1;
```

```
param r 'comm ratio' >=0, <=1;

param a 'loop weight' {m in M} >=0;

param gamma 'gap weight' >=0;

param mono 'monocycle loop' {m in M} binary;

###############################################
###              VARIABLES                  ###
###############################################

var o 'overlap' {j in J, i in Ij[j], c in C[i], i2 in Ij[j], c2 in C[i2]}
binary;

var y 'rback before or after' {i in Ir} binary;

var Tsb 'start time' {i in I} >=0, <=min(2*(max{m in M, i2 in Im[m]:
i in Im[m]}(Hi[i2])),H);

var Tfb 'final time' {i in I} >=0, <=min(2*(max{m in M, i2 in Im[m]:
i in Im[m]}(Hi[i2])),H);

var Ts {i in I, c in C[i]} >=0, <=H;

var Tf {i in I, c in C[i]} >=0, <=H;

var TF 'final time of all' >=0, <= max{i in I}(Hi[i]);

# New grouping optimization technique

var g 'gap exists' {i in Ij[net], c in C[i], i2 in Ij[net], c2 in C[i2]}
binary;

# Cycle basis

var cb {i in I} >=1, <= min(2*(max{i2 in I, m in M: i in Im[m] and
i2 in Im[m]}(card(C[i2])))/(min{i2 in I, m in M: i in Im[m] and
```

```
i2 in Im[m]}(card(C[i2]))),card(C[i])) integer;

var G >= 0, <= (sum{i in Ij[net]}(card(C[i])) - 1); # in case all
consecutive comms have gaps

var D {m in M} >=0;

###############################################
###              CONSTRAINTS                ###
###############################################

### SEQUENCE CONSTRAINTS


subject to samedev 'different tasks same device' {j in J, i in Ij[j],
c in C[i], i2 in Ij[j], c2 in C[i2]: i2 <> i or c <> c2}:
Tf[i2,c2] - Ts[i,c] <= H*o[j,i,c,i2,c2];

subject to seqtime 'task times in sequence' {m in M, (i,i2) in L[m]}:
Tsb[i2] >= Tfb[i];

subject to seqlink 'sequence of task events on same device'
{j in J, m in M,(i,i2) in Lj[j,m], c in C[i]: card(C[i]) = card(C[i2])}:
o[j,i,c,i2,c] = 1; # A próxima tarefa só pode ser executada
depois da anterior na sequência


### OVERLAP CONSTRAINTS

subject to overlap 'overlap of tasks in a device' {j in J, i in Ij[j],
c in C[i], i2 in Ij[j], c2 in C[i2]: i <> i2 or c <> c2}:
o[j,i,c,i2,c2] + o[j,i2,c2,i,c] = 1;

### FINAL TIME CONSTRAINTS

subject to endtimes {i in I, c in C[i]}:
Tf[i,c] = Ts[i,c] + Te[i];

subject to endtimebasis {i in I}:
```

```
Tfb[i] = Tsb[i] + Te[i];

subject to makespan {i in I}:
TF >= Tf[i,1];

### EQUIVALENCE CONSTRAINTS

subject to equivtime {(i,i2) in E}:
Tsb[i2] >= Tsb[i];

### RECYCLE CONSTRAINTS

subject to beforetask {i in Ir, (iA,iB) in Re[i]}:
Tsb[iB] >= Tfb[i] - H*y[i];

subject to aftertask {i in Ir, (iA,iB) in Re[i]}:
Tsb[i] >= Tfb[iA] - H*(1-y[i]);

subject to afterlast {i in Ir, (iA,iB) in Re[i]}:
Tsb[i] >= Tfb[iA] - Hi[iA] - H*y[i];

subject to beforenext {i in Ir, (iA,iB) in Re[i]}:
Tsb[iB] + Hi[iB] >= Tfb[i] - H*(1-y[i]);

### COMMUNICATION GAP CONSTRAINTS

subject to gapexist {i in Ij[net], c in C[i], i2 in Ij[net], c2 in C[i2]:
i <> i2 or c <> c2}:
H*g[i,c,i2,c2] >= Ts[i2,c2] - Tf[i,c];

subject to cyclegapsame {i in Ij[net], c in C[i], c2 in C[i]: c2 > c}:
g[i,c,i,c2] = 1;

### BASIC CYCLE CONSTRAINTS

subject to basiccycle {i in I}:
0 >= Tsb[i] - Hi[i]*cb[i] >= -Hi[i];

### PERIODICITY CONSTRAINTS
```

```
subject to periodic {i in I, c in C[i]: c < card(C[i])}:
Ts[i,c+1] = Ts[i,c] + Hi[i];

subject to basic {i in I, c in C[i]}:
Ts[i,c] = Tsb[i] + Hi[i]*(c-cb[i]);

### CYCLE LIMIT CONSTRAINTS

subject to limitstart {i in I, c in C[i]}:
Hi[i]*(c-1) <= Ts[i,c] <= Hi[i]*c;

subject to limitend {i in I, c in C[i]}:
Hi[i]*(c-1) <= Tf[i,c] <= Hi[i]*c;

### GROUPING CONSTRAINTS

subject to grouping:
G = sum{i in Ij[net], c in C[i], i2 in Ij[net], c2 in C[i2]:
i <> i2}g[i,c,i2,c2] - 0.5*( (sum{i in Ij[net]}(card(C[i])) - 1)*
(sum{i in Ij[net]}(card(C[i])) - 2) - (sum{i in Ij[net]}(card(C[i])*
(card(C[i]) - 1))) );
# accounting tasks with some task in between
# accounting different cycles for same task (which is removed in the
original summation)


### DELAY CONSTRAINTS

subject to delays {m in M}:
D[m] = sum{(i,i2) in L[m]}(Tsb[i2] - Tsb[i]) - sum{(i,i2) in L[m]}
(Te[i]);

##############################################
###              OBJECTIVES                ###
##############################################

minimize tradeoff 'trade-off between objectives':
alpha*gamma*G + beta*sum{m in M}(a[m]*D[m]) + (1-alpha-beta)*TF;
```

```
end;
```

Modelo por Sobreposição Completo

```
##############################################
###                 SETS                   ###
##############################################

set I;

set J 'devices'; # 0 é a rede

param net = 0; # rede

set M; # cada malha tem suas tarefas independentemente

set Ij {j in J} within I; # tarefas do device j

set Im {m in M} within I; # tarefas da malha m

set L {m in M} within {Im[m],Im[m]}; # Ligações entre tarefas

set Lj {j in J, m in M} within {Ij[j],Ij[j]} default {{},{}}; # sequências
 entre tarefas num mesmo device

set E within {I,I} default {{},{}}; # Pares de tarefas equivalentes

set Ir within Ij[net] default {}; # Readback communications

set Re {i in Ir} within {I,I}; # De onde vem e para onde vai o readback

### Multicicle Sets

set C {i in I}; # Ciclos num macrociclo para uma dada tarefa

##############################################
###              PARAMETERS                ###
```

```
##############################################

param H 'time horizon'; # Macrociclo requisitado

param Hi {i in I}; # Cicle time for task i

param Te 'execution time' {i in I};

param alpha 'tradeoff' >= 0, <= 1;

param beta 'tradeoff2' >= 0, <= 1;
#check alpha + beta <= 1; depois!

param r 'comm ratio' >=0, <=1;

param a 'loop weight' {m in M} >=0;

param gamma 'gap weight' >=0;

param mono 'monocycle loop' {m in M} binary;

##############################################
###       UTILITY SETS & PARAMETERS      ###
##############################################

param d {j in J, i in Ij[j], i2 in Ij[j]} := card(C[i])/
(min{c in C[i], c2 in C[i2]: c*Hi[i] = c2*Hi[i2]}(c));

param dm {m in M};

param Z := sum{i in Ij[net]}(card(C[i]));

###

set P {j in J} := {i in Ij[j], c in C[i], i2 in Ij[j], c2 in C[i2]:
 i2 <> i and (Hi[i2]*c2 > Hi[i]*(c-1) + 1) and (Hi[i]*c > Hi[i2]*(c2-1)
+ 1) and c <= card(C[i])/d[j,i,i2] and c2 <= card(C[i2])/d[j,i,i2]};

set Q := {i in Ij[net], c in C[i], i2 in Ij[net], c2 in C[i2]: i2 <> i
```

```
and (Hi[i2]*c2 >= Hi[i]*(c-1)) and (Hi[i]*c >= Hi[i2]*(c2-1)) and
c <= card(C[i])/d[net,i,i2] and c2 <= card(C[i2])/d[net,i,i2]};

set Ql := {i in Ij[net], c in C[i], i2 in Ij[net], c2 in C[i2]: i2 <> i
 and (((Hi[i2]*c2 = Hi[i]*(c-1)) and c = card(C[i])/d[net,i,i2]+1 and
c2 = card(C[i2])/d[net,i,i2]) or ((Hi[i]*c = Hi[i2]*(c2-1)) and
c = card(C[i])/d[net,i,i2] and c2 = card(C[i2])/d[net,i,i2]+1))};

set Qbar := {i in Ij[net], c in C[i], i2 in Ij[net], c2 in C[i2]:
(i2 = i and c < c2) or Hi[i]*c + 1 < Hi[i2]*(c2-1)}; # Obvious gaps
from (i,c) to (i2,c2)

##############################################
###              VARIABLES                 ###
##############################################

var o 'overlap' {j in J, (i,c,i2,c2) in P[j]} binary;

var y 'rback before or after' {i in Ir} binary;

var Tsb 'start time' {i in I} >=0, <=min(2*(max{m in M, i2 in Im[m]:
i in Im[m]}(Hi[i2])),H);

var Tfb 'final time' {i in I} >=0, <=min(2*(max{m in M, i2 in Im[m]:
i in Im[m]}(Hi[i2])),H);

var Ts {i in I, c in C[i]} >=0, <=H;

var Tf {i in I, c in C[i]} >=0, <=H;

var TF 'final time of all' >=0, <= max{i in I}(Hi[i]);

# New grouping optimization technique

var g 'gap exists' {(i,c,i2,c2) in (Q union Ql)} binary;

# Cycle basis

var cb {i in I} >=1, <= min(2*card(C[i])/(sum{m in M: i in Im[m]}(dm[m])),
```

```
card(C[i])) integer;

var G >= 0, <= (sum{i in Ij[net]}(card(C[i])) - 1); # in case all
consecutive comms have gaps

var D {m in M} >=0;

###############################################
###              CONSTRAINTS                ###
###############################################

### SEQUENCE CONSTRAINTS

subject to samedev 'different tasks same device' {j in J, (i,c,i2,c2) in
 P[j]}:
Tf[i2,c2] - Ts[i,c] <= (Hi[i2]*c2 - Hi[i]*(c-1))*o[j,i,c,i2,c2];

subject to seqtime 'task times in sequence' {m in M, (i,i2) in L[m]}:
Tsb[i2] >= Tfb[i];

subject to seqlink 'sequence of task events on same device' {j in J,
m in M,(i,i2) in Lj[j,m], c in C[i]: card(C[i]) = card(C[i2]) and
(i,c,i2,c) in P[j]}:
o[j,i,c,i2,c] = 1; # A próxima tarefa só pode ser executada
depois da anterior na sequência

subject to gaplink 'sequence of task events on same device' {m in M,
(i,i2) in Lj[net,m], c in C[i]: card(C[i]) = card(C[i2]) and (i,c,i2,c)
in (Q union Ql)}:
g[i,c,i2,c] = 1; # Há um gap porque entre duas comunicações
da mesma malha há uma tarefa

### OVERLAP CONSTRAINTS

subject to overlap 'overlap of tasks in a device' {j in J, (i,c,i2,c2)
in P[j]}:
o[j,i,c,i2,c2] + o[j,i2,c2,i,c] = 1;

### FINAL TIME CONSTRAINTS
```

```
subject to endtimes {i in I, c in C[i]}:
Tf[i,c] = Ts[i,c] + Te[i];

subject to endtimebasis {i in I}:
Tfb[i] = Tsb[i] + Te[i];

subject to makespan {i in I}:
TF >= Tf[i,1];

### EQUIVALENCE CONSTRAINTS

subject to equivtime {(i,i2) in E}:
Tsb[i2] >= Tsb[i];

### RECYCLE CONSTRAINTS

subject to beforetask {m in M, i in Ir, (iA,iB) in Re[i]}:
Tsb[iB] >= Tfb[i] - H*y[i];

subject to aftertask {i in Ir, (iA,iB) in Re[i]}:
Tsb[i] >= Tfb[iA] - H*(1-y[i]);

subject to afterlast {i in Ir, (iA,iB) in Re[i]}:
Tsb[i] >= Tfb[iA] - Hi[iA] - H*y[i];

subject to beforenext {i in Ir, (iA,iB) in Re[i]}:
Tsb[iB] + Hi[iB] >= Tfb[i] - H*(1-y[i]);

### COMMUNICATION GAP CONSTRAINTS

subject to gapexist {(i,c,i2,c2) in (Q union Q1)}:
(Hi[i2]*c2 - Hi[i]*(c-1))*g[i,c,i2,c2] >= Ts[i2,c2] - Tf[i,c];

### BASIC CYCLE CONSTRAINTS

subject to basiccycle {i in I}:
0 >= Tsb[i] - Hi[i]*cb[i] >= -Hi[i];
```

```
### PERIODICITY CONSTRAINTS

subject to periodic {i in I, c in C[i]: c < card(C[i])}:
Ts[i,c+1] = Ts[i,c] + Hi[i];

subject to basic {i in I, c in C[i]}:
Ts[i,c] = Tsb[i] + Hi[i]*(c-cb[i]);

### CYCLE LIMIT CONSTRAINTS

subject to limitstart {i in I, c in C[i]}:
Hi[i]*(c-1) <= Ts[i,c] <= Hi[i]*c;

subject to limitend {i in I, c in C[i]}:
Hi[i]*(c-1) <= Tf[i,c] <= Hi[i]*c;

### GROUPING CONSTRAINTS



subject to grouping:
G = sum{(i,c,i2,c2) in Q}(d[net,i,i2]*g[i,c,i2,c2]) +
sum{(i,c,i2,c2) in Ql}((d[net,i,i2]-1)*g[i,c,i2,c2]) + card(Qbar) -
(Z-1)*(Z-2)/2;


### DELAY CONSTRAINTS

subject to delays {m in M}:
D[m] = sum{(i,i2) in L[m]}(Tsb[i2] - Tfb[i]);

### GAP CUTS (!)

subject to gapsymmetry {(i,c,i2,c2) in (Q union Ql): (i2,c2,i,c) in
(Q union Ql)}:
g[i,c,i2,c2] + g[i2,c2,i,c] <= 1; # can be zero if there is
effectively no gap between them.

subject to gapsforward {(i,c,i2,c2) in ((Q union Ql) inter P[net])}:
```

```
g[i,c,i2,c2] <= o[net,i,c,i2,c2]; # only gap from a to b if
b > a

subject to gapsbetween {(i,c,i2,c2) in P[net], i3 in Ij[net], c3 in C[i3]:
 (i,c,i3,c3) in ((Q union Ql) inter P[net]) and (i2,c2,i3,c3) in P[net]}:
g[i,c,i3,c3] + 1 >= o[net,i,c,i2,c2] + o[net,i2,c2,i3,c3]; # if
there is a comm between, there will be a gap

### OVERLAP CUTS

subject to nextoverlap {j in J, (i,c,i2,c2) in P[j]: c2 < card(C[i2]) and
(i,c,i2,c2+1) in P[j]}:
o[j,i,c,i2,c2] <= o[j,i,c,i2,c2+1];

subject to nextgap {(i,c,i2,c2) in (Q union Ql): c2 < card(C[i2]) and
(i,c,i2,c2+1) in (Q union Ql)}:
g[i,c,i2,c2] <= g[i,c,i2,c2+1];

###############################################
###              OBJECTIVES                 ###
###############################################

minimize tradeoff 'trade-off between objectives':
alpha*gamma*G + beta*sum{m in M}(a[m]*D[m]) + (1-alpha-beta)*TF;

end;
```

B.2.2 Dados para os Estudos de Caso

Caso III

```
###############################################
###                 SETS                    ###
###############################################

set I := AI1 AI2 AI3 AI41 AI42 AI5 AO1 AO2 AO3 AO4 AO5 CD1 CD2 CD3 CD4 CD5 CD6 CD7

set J := 0 AI1 AI2 AI3 AI4 AI5 AO1 AO2 AO3 AO4 AO5;
```

```
set Ij[AI1] := AI1;
set Ij[AI2] := AI2;
set Ij[AI3] := AI3;
set Ij[AI4] := AI41 AI42;
set Ij[AI5] := AI5;
set Ij[AO1] := AO1;
set Ij[AO2] := AO2;
set Ij[AO3] := AO3;
set Ij[AO4] := AO4;
set Ij[AO5] := AO5;
set Ij[0] := CD1 CD2 CD3 CD4 CD5 CD6 CD7 CD8;

set M := 1 2 3 4;

set Im[1] := AI1 CD1 AO1;
set Im[2] := AI2 CD2 AI3 CD3 AO2;
set Im[3] := AI41 AI42 CD4 CD5 AO3 AO4 CD6 CD7;
set Im[4] := AI5 CD8 AO5;

set L[1] := (AI1,CD1) (CD1,AO1);
set L[2] := (AI2,CD2) (AI3,CD3) (CD2,AO2) (CD3,AO2);
set L[3] := (AI41,AI42) (AI42,CD4) (AI42,CD5) (CD4,AO3) (CD5,AO4);
set L[4] := (AI5,CD8) (CD8,AO5);

set E := (AI2,AI3) (CD2,CD3) (CD4,CD5) (AO3,AO4);

set Ir := CD6 CD7;

set Re[CD6] := (AO3,AI42);
set Re[CD7] := (AO4,AI42);

set C[AI1] := 1 2;
set C[AI2] := 1;
set C[AI3] := 1;
set C[AI41] := 1;
set C[AI42] := 1;
set C[AI5] := 1 2 3 4;
set C[AO1] := 1 2;
```

```
set C[AO2] := 1;
set C[AO3] := 1;
set C[AO4] := 1;
set C[AO5] := 1 2 3 4;
set C[CD1] := 1 2;
set C[CD2] := 1;
set C[CD3] := 1;
set C[CD4] := 1;
set C[CD5] := 1;
set C[CD6] := 1;
set C[CD7] := 1;
set C[CD8] := 1 2 3 4;

##############################################
###              PARAMETERS                ###
##############################################

param H := 1000;

param net := 0;

param r := 0.4;

param Te :=  AI1 25
AI2 30
AI3 30
AI41 45
AI42 20
AI5 30
AO1 90
AO2 80
AO3 40
AO4 40
AO5 55
CD1 30
CD2  30
CD3  30
CD4 30
CD5 30
```

```
CD6 30
CD7 30
CD8 30;

param Hi :=  AI1 500
AI2 1000
AI3 1000
AI41 1000
AI42 1000
AI5 250
AO1 500
AO2 1000
AO3 1000
AO4 1000
AO5 250
CD1 500
CD2  1000
CD3  1000
CD4 1000
CD5 1000
CD6 1000
CD7 1000
CD8 250;

param a :=  1 1.0
2 1.0
3 1.0
4 1.0;

param alpha := 0.49; # weight grouping
param beta := 0.49; # weight latency

param gamma := 50;
```

Caso IV

```
##############################################
###                SETS                    ###
```

```
##########################################

set I := AI1 AI2 AI3 AI4 PID1 PID2 PID3 AR IN AO CD1 CD2 CD3 CD4 CD5 CD6;

set J := 0 T1 T2 T3 P1;

set Ij[T1] := AI1 PID1 AI4;
set Ij[T2] := AI2 PID2;
set Ij[T3] := AI3;
set Ij[P1] := AR IN PID3 AO;
set Ij[0] := CD1 CD2 CD3 CD4 CD5 CD6;

set M := 1;

set Im[1] := AI1 AI2 AI3 AI4 PID1 PID2 PID3 AR IN AO CD1 CD2 CD3 CD4 CD5 CD6;

set L[1] := (AI1,PID1) (PID1,CD1) (CD1,PID2) (PID2,CD2) (CD2,PID3) (PID3,AO) (AI4,C

set Lj[T1,1] := (AI1,PID1);
set Lj[T2,1] := (AI2,PID2);
set Lj[P1,1] := (AR,IN) (AR,PID3) (PID3,AO);
set Lj[0,1] := (CD1,CD2);

set Ir := CD5 CD6;

set Re[CD5] := (PID2,PID1);
set Re[CD6] := (PID3,PID2);

set C[AI1] := 1;
set C[AI2] := 1 2;
set C[AI3] := 1 2 3 4;
set C[AI4] := 1;
set C[PID1] := 1;
set C[PID2] := 1 2;
set C[PID3] := 1 2 3 4;
set C[AR] := 1 2 3 4;
set C[IN] := 1 2 3 4;
set C[AO] := 1 2 3 4;
set C[CD1] := 1;
```

```
set C[CD2] := 1 2;
set C[CD3] := 1;
set C[CD4] := 1 2 3 4;
set C[CD5] := 1 2;
set C[CD6] := 1 2 3 4;

##############################################
###              PARAMETERS                ###
##############################################

param H := 2000;

param net := 0;

param r := 0.4;

param Te :=  AI1 25
AI2 30
AI3 30
AI4 20
PID1 60
PID2 55
PID3 70
AR 40
IN 35
AO 50
CD1 30
CD2  30
CD3  30
CD4 30
CD5 30
CD6 30;

param Hi :=  AI1 2000
AI2 1000
AI3 500
AI4 2000
PID1 2000
PID2 1000
```

```
PID3 500
AR 500
IN 500
AO 500
CD1 2000
CD2  1000
CD3  2000
CD4 500
CD5 1000
CD6 500;

param a :=  1 1.0;

param alpha := 0.49; # weight grouping
param beta := 0.49; # weight latency

param gamma := 50;
```

Caso V

```
###############################################
###                 SETS                    ###
###############################################

set I := AI1 AI2 AI3 AI4 PID1 PID2 PID3 AR IN AO CD1 CD2 CD3 CD4 CD5 CD6;

set J := 0 T1 T2 T3 P1;

set Ij[T1] := AI1 PID1 AI4;
set Ij[T2] := AI2 PID2;
set Ij[T3] := AI3;
set Ij[P1] := AR IN PID3 AO;
set Ij[0] := CD1 CD2 CD3 CD4 CD5 CD6;

set M := 1;

set Im[1] := AI1 AI2 AI3 AI4 PID1 PID2 PID3 AR IN AO CD1 CD2 CD3 CD4
```

```
CD5 CD6;

set L[1] := (AI1,PID1) (PID1,CD1) (CD1,PID2) (PID2,CD2) (CD2,PID3)
(PID3,AO) (AI4,CD3) (CD3,AR) (AR,IN) (AR,PID3) (AI2,PID2) (AI3,CD4)
(CD4,AR);

set Lj[T1,1] := (AI1,PID1);
set Lj[T2,1] := (AI2,PID2);
set Lj[P1,1] := (AR,IN) (AR,PID3) (PID3,AO);
set Lj[0,1] := (CD1,CD2);

set Ir := CD5 CD6;

set Re[CD5] := (PID2,PID1);
set Re[CD6] := (PID3,PID2);

set C[AI1] := 1;
set C[AI2] := 1 2;
set C[AI3] := 1 2 3 4;
set C[AI4] := 1;
set C[PID1] := 1;
set C[PID2] := 1 2;
set C[PID3] := 1 2 3 4;
set C[AR] := 1 2 3 4;
set C[IN] := 1 2 3 4;
set C[AO] := 1 2 3 4;
set C[CD1] := 1;
set C[CD2] := 1 2;
set C[CD3] := 1;
set C[CD4] := 1 2 3 4;
set C[CD5] := 1 2;
set C[CD6] := 1 2 3 4;

###############################################
###              PARAMETERS                 ###
###############################################

param H := 2000;
```

```
param net := 0;

param r := 0.4;

param Te :=  AI1 25
AI2 30
AI3 30
AI4 20
PID1 60
PID2 55
PID3 70
AR 40
IN 35
AO 50
CD1 30
CD2  30
CD3  30
CD4 30
CD5 30
CD6 30;

param Hi :=  AI1 2000
AI2 1000
AI3 500
AI4 2000
PID1 2000
PID2 1000
PID3 500
AR 500
IN 500
AO 500
CD1 2000
CD2  1000
CD3  2000
CD4 500
CD5 1000
CD6 500;

param a :=  1 1.0;
```

```
param alpha := 0.49; # weight grouping
param beta := 0.49; # weight latency

param gamma := 50;
```

Caso VI

```
###############################################
###                 SETS                    ###
###############################################

set I := AI1 AI2 AI3 AI41 AI42 AI5 AO1 AO2 AO3 AO4 AO5 CD1 CD2 CD3 CD4 CD5
 CD6 CD7 CD8;

set J := 0 AI1 AI2 AI3 AI4 AI5 AO1 AO2 AO3 AO4 AO5;

set Ij[AI1] := AI1;
set Ij[AI2] := AI2;
set Ij[AI3] := AI3;
set Ij[AI4] := AI41 AI42;
set Ij[AI5] := AI5;
set Ij[AO1] := AO1;
set Ij[AO2] := AO2;
set Ij[AO3] := AO3;
set Ij[AO4] := AO4;
set Ij[AO5] := AO5;
set Ij[0] := CD1 CD2 CD3 CD4 CD5 CD6 CD7 CD8;

set M := 1 2 3 4;

set Im[1] := AI1 CD1 AO1;
set Im[2] := AI2 CD2 AI3 CD3 AO2;
set Im[3] := AI41 AI42 CD4 CD5 AO3 AO4 CD6 CD7;
set Im[4] := AI5 CD8 AO5;

set L[1] := (AI1,CD1) (CD1,AO1);
set L[2] := (AI2,CD2) (AI3,CD3) (CD2,AO2) (CD3,AO2);
```

```
set L[3] := (AI41,AI42) (AI42,CD4) (AI42,CD5) (CD4,AO3) (CD5,AO4);
set L[4] := (AI5,CD8) (CD8,AO5);

set E := (AI2,AI3) (CD2,CD3) (CD4,CD5) (AO3,AO4);

set Ir := CD6 CD7;

set Re[CD6] := (AO3,AI42);
set Re[CD7] := (AO4,AI42);

set C[AI1] := 1 2 3 4 5;
set C[AI2] := 1 2;
set C[AI3] := 1 2;
set C[AI41] := 1 2;
set C[AI42] := 1 2;
set C[AI5] := 1 2 3 4 5 6 7 8 9 10;
set C[AO1] := 1 2 3 4 5;
set C[AO2] := 1 2;
set C[AO3] := 1 2;
set C[AO4] := 1 2;
set C[AO5] := 1 2 3 4 5 6 7 8 9 10;
set C[CD1] := 1 2 3 4 5;
set C[CD2] := 1 2;
set C[CD3] := 1 2;
set C[CD4] := 1 2;
set C[CD5] := 1 2;
set C[CD6] := 1 2;
set C[CD7] := 1 2;
set C[CD8] := 1 2 3 4 5 6 7 8 9 10;

###############################################
###              PARAMETERS                 ###
###############################################

param H := 2000;

param net := 0;

param r := 0.4;
```

```
param Te :=  AI1 25
AI2 30
AI3 30
AI41 45
AI42 20
AI5 30
AO1 90
AO2 80
AO3 40
AO4 40
AO5 55
CD1 30
CD2  30
CD3  30
CD4 30
CD5 30
CD6 30
CD7 30
CD8 30;

param Hi :=  AI1 400
AI2 1000
AI3 1000
AI41 1000
AI42 1000
AI5 200
AO1 400
AO2 1000
AO3 1000
AO4 1000
AO5 200
CD1 400
CD2  1000
CD3  1000
CD4 1000
CD5 1000
CD6 1000
CD7 1000
```

```
CD8 200;

param a :=  1 1.0
2 1.0
3 1.0
4 1.0;

param dm :=  1 5
2 2
3 2
4 10;

param alpha := 0.49; # weight grouping
param beta := 0.49; # weight latency

param gamma := 50;
```

Referências Bibliográficas

[1] ABDELZAHER, T., SHIN, K., 1995, "Optimal combined task and message scheduling in distributed real-time systems". In: *Real-Time Systems Symposium, 1995. Proceedings., 16th IEEE*, pp. 162–171. IEEE.

[2] BARRETO, R., CAVALCANTE, S., MACIEL, P., 2004, "A time Petri net approach for finding preruntime schedules in embedded hard real-time systems". In: *Distributed Computing Systems Workshops, 2004. Proceedings. 24th International Conference on*, pp. 846–851. IEEE.

[3] BENDER, A., 1996, "MILP based task mapping for heterogeneous multiprocessor systems". In: *Design Automation Conference, 1996, with EURO-VHDL'96 and Exhibition, Proceedings EURO-DAC'96, European*, pp. 190–197. IEEE.

[4] BERGE, J., 2004, *Fieldbuses for Process Control: Engineering, Operation and Maintenance*. ISA.

[5] BRANICKY, M., PHILLIPS, S., ZHANG, W., 2000, "Stability of networked control systems: Explicit analysis of delay". In: *American Control Conference, 2000. Proceedings of the*, v. 4, pp. 2352–2357. IEEE.

[6] CARDEIRA, C., MAMMERI, Z., 1995, "A schedulability analysis of tasks and network traffic in distributed real-time systems", *Measurement*, v. 15, n. 2, pp. 71–83.

[7] CAVALIERI, S., DI STEFANO, A., MIRABELLA, O., 1995, "Pre-run-time scheduling to reduce schedule length in the fieldbus environment", *Software Engineering, IEEE Transactions on*, v. 21, n. 11, pp. 865–880.

[8] CICILLINI, D., 2007. "Desenvolvimento de um Algoritmo de Escalonamento para Rede FOUNDATION Fieldbus". .

[9] COMMISSION, I. E., 2003, *IEC 61158, Digital data communications for measurement and control - Fieldbus for use in industrial control systems*.

[10] DAVARE, A., CHONG, J., ZHU, Q., et al., 2006, "Classification, customization, and characterization: Using milp for task allocation and scheduling", *University of California Berkley, Technical Report No. UCB/EECS-2006-166*.

[11] DAVIS, R., BURNS, A., 2009, "Priority assignment for global fixed priority preemptive scheduling in multiprocessor real-time systems". In: *2009 30th IEEE Real-Time Systems Symposium*, pp. 398–409. IEEE.

[12] DE OLIVEIRA, R., DA SILVA FRAGA, J., 2000, "Fixed priority scheduling of tasks with arbitrary precedence constraints in distributed hard real-time systems", *Journal of systems architecture*, v. 46, n. 11, pp. 991–1004.

[13] FLOUDAS, C., LIN, X., 2004, "Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review", *Computers & chemical engineering*, v. 28, n. 11, pp. 2109–2129.

[14] FLOUDAS, C., LIN, X., 2005, "Mixed integer linear programming in process scheduling: modeling, algorithms, and applications", *Annals of operations Research*, v. 139, n. 1, pp. 131–162.

[15] FOUNDATION, F., 2010, *AG-181 - FOUNDATION Fieldbus System Engineering Guidelines*.

[16] FOUNDATION, F., 2003, *FD-043 - FOUNDATION Fieldbus Technical Overview*.

[17] FRANCO, L., 1996, "Transmission scheduling for fieldbus: a strategy to schedule data and messages on the bus with end-to-end constraints". In: *Intelligence and Systems, 1996., IEEE International Joint Symposia on*, pp. 148–155. IEEE.

[18] GUPTA, R., CHOW, M., 2010, "Networked control system: overview and research trends", *Industrial Electronics, IEEE Transactions on*, v. 57, n. 7, pp. 2527–2535.

[19] HEGYHÁTI, M., FRIEDLER, F., 2010, "Overview of Industrial Batch Process Scheduling", *Chemical Engineering Transactions*, v. 21, pp. 895–900.

[20] HESPANHA, J., NAGHSHTABRIZI, P., XU, Y., 2007, "A survey of recent results in networked control systems", *Proceedings of the IEEE*, v. 95, n. 1, pp. 138–162.

[21] HODSON, R., 2005, "Optimizing Fieldbus Link Schedules Makes a Difference!" *ISA EXPO 2005*.

[22] IERAPETRITOU, M., FLOUDAS, C., 1998, "Effective continuous-time formulation for short-term scheduling. 1. Multipurpose batch processes", *Industrial & engineering chemistry research*, v. 37, n. 11, pp. 4341–4359.

[23] INTERNATIONAL, H., 2010, *Product Information Note - Experion Foundation Fieldbus Integration*, December.

[24] JÜNGER, M., LIEBLING, T., NADDEF, D., et al., 2010, *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*. Springer.

[25] KIM, D., LEE, Y., KWON, W., et al., 2003, "Maximum allowable delay bounds of networked control systems", *Control Engineering Practice*, v. 11, n. 11, pp. 1301–1313.

[26] KIM, H., KIM, J., KIM, D., et al., 2004, "Development of coordinated scheduling strategy with end-to-end response time analysis for CAN-based distributed control systems". In: *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, v. 3, pp. 2099–2104. IEEE.

[27] KUNIS, R., RÜNGER, G., 2011, "Optimizing layer-based scheduling algorithms for parallel tasks with dependencies", *Concurrency and Computation: Practice & Experience*, v. 23, n. 8, pp. 827–849.

[28] LI, Z., WANG, W., JIANG, Y., 2009, "Managing quality-of-control and requirement-of-bandwidth in networked control systems via fuzzy bandwidth scheduling", *International Journal of Control, Automation and Systems*, v. 7, n. 2, pp. 289–296.

[29] LIU, C., LAYLAND, J., 1973, "Scheduling algorithms for multiprogramming in a hard-real-time environment", *Journal of the ACM (JACM)*, v. 20, n. 1, pp. 46–61.

[30] MACULAN, N., PORTO, S., RIBEIRO, C., et al., 1999, "A new formulation for scheduling unrelated processor under precedence constraints", *RAIRO-Operations Research*, v. 33, n. 01, pp. 87–92.

[31] MANAGEMENT, E. P., 2012, *Implementing Fieldbus in DeltaV System - Answers to your Questions about Implementing Fieldbus in DeltaV Systems*, July.

[32] MÉNDEZ, C., CERDÁ, J., GROSSMANN, I., et al., 2006, "State-of-the-art review of optimization methods for short-term scheduling of batch processes", *Computers & chemical engineering*, v. 30, n. 6-7, pp. 913–946.

[33] NAGARAJ SHENOY, U., BANERJEE, P., CHOUDHARY, A., 2000, "A system-level synthesis algorithm with guaranteed solution quality". In: *Design, Automation and Test in Europe Conference and Exhibition 2000. Proceedings*, pp. 417–424. IEEE.

[34] PENG, C., YUE, D., GU, Z., et al., 2009, "Sampling period scheduling of networked control systems with multiple-control loops", *Mathematics and Computers in Simulation*, v. 79, n. 5, pp. 1502–1511.

[35] PENG, D., SHIN, K., ABDELZAHER, T., 1997, "Assignment and scheduling communicating periodic tasks in distributed real-time systems", *Software Engineering, IEEE Transactions on*, v. 23, n. 12, pp. 745–758.

[36] REDELL, O., 1998, *Modelling of Distributed Real-Time Control Systems - An Approach for Design and Early Analysis*. Tese de licenciatura, Royal Institute of Technology (KTH), Stockholm, Sweden.

[37] SCHRIJVER, A., 1998, *Theory of Linear and Integer Programming*. Wiley.

[38] SHAIK, M., FLOUDAS, C., 2009, "Novel unified modeling approach for short-term scheduling", *Industrial & Engineering Chemistry Research*, v. 48, n. 6, pp. 2947–2964.

[39] SONG, J., 2010, *Constraint-based Real-time Scheduling for Process Control*. Tese de Doutorado, University of Texas at Austin.

[40] SONG, J., MOK, A., CHEN, D., et al., 2007, "Optimizing Distributed Foundation Fieldbus Process Control with MSP. RTL Tool". In: *Industrial Informatics, 2007 5th IEEE International Conference on*, v. 2, pp. 867–872. IEEE.

[41] VERHAPPEN, I., PEREIRA, A., 2009, *Foundation Fieldbus*. ISA.

[42] WOLSEY, A., NEMHAUSER, G., 1999, *Integer and Combinatorial Optimization*. Wiley-Interscience.

[43] WU, Q., LI, Y., QIN, Y., 2006, "A scheduling method based on deadline for CAN-based networked control systems". In: *Mechatronics and Automation, Proceedings of the 2006 IEEE International Conference on*, pp. 345–350. IEEE.

[44] XIANG-LI, Z., 2009, "Study on communication scheduling of fieldbus". In: *Control and Decision Conference, 2009. CCDC'09. Chinese*, pp. 565–570. IEEE.

[45] XU, J., 1993, "Multiprocessor scheduling of processes with release times, deadlines, precedence, and exclusion relations", *Software Engineering, IEEE Transactions on*, v. 19, n. 2, pp. 139–154.

[46] XU, J., PARNAS, D., 1990, "Scheduling processes with release times, deadlines, precedence and exclusion relations", *Software Engineering, IEEE Transactions on*, v. 16, n. 3, pp. 360–369.

[47] XU, J., PARNAS, D., 1993, "On satisfying timing constraints in hard-real-time systems", *Software Engineering, IEEE Transactions on*, v. 19, n. 1, pp. 70–84.