UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ADMINISTRAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM ADMINISTRAÇÃO

Éfrem de Aguiar Maranhão Filho

ALOCAÇÃO E MOVIMENTAÇÃO DINÂMICA DE CONTÊINERES: Um Modelo Integrado de Escalonamento

Porto Alegre
2009

Éfrem de Aguiar Maranhão Filho

ALOCAÇÃO E MOVIMENTAÇÃO DINÂMICA DE CONTÊINERES: Um Modelo Integrado de Escalonamento

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Administração da Universidade Federal do Rio Grande do Sul, como requisito parcial para a obtenção do título de Mestre em Administração.

Orientador: Prof. Dr. João Luiz Becker

Porto Alegre
2009

Dados Internacionais de Catalogação na Publicação (CIP)

M311a Maranhão Filho, Éfrem de Aguiar
Alocação e movimentação dinâmica de contêineres: um modelo integrado de escalonamento / Éfrem de Aguiar Maranhão Filho. – 2009.
85 f. : il.

Dissertação (Mestrado) – Universidade Federal do Rio Grande do Sul, Escola de Administração, Programa de Pós-Graduação em Administração, 2009.

Orientador: Prof. Dr. João Luiz Becker

1. Logística. 2. Alocação dinâmica de contêineres. 3. Movimentação dinâmica de contêineres. 4. Programação matemática. I. Título

CDU 658.7

Ficha elaborada pela equipe da Biblioteca da Escola de Administração

Éfrem de Aguiar Maranhão Filho

ALOCAÇÃO E MOVIMENTAÇÃO DINÂMICA DE CONTÊINERES: Um Modelo Integrado de Escalonamento

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Administração da Universidade Federal do Rio Grande do Sul, como requisito parcial para a obtenção do título de Mestre em Administração.

Conceito final:

Aprovado em ..... de ..... de .....

BANCA EXAMINADORA

_____

Prof. Dr. Denis Borenstein – PPGA – UFRGS

_____

Prof. Dr. Eduardo Ribas Santos – PPGA – UFRGS

_____

Prof. Dr. Flávio Sanson Fogliatto – PPGEP– UFRGS

_____

Orientador – Prof. Dr. João Luiz Becker – PPGA – UFRGS

Para meu avô, Carlos Humberto Bártholo,
descanse em paz.

AGRADECIMENTOS

A minha felicidade é saber que pude criar parcerias duradouras e amizades que não se restringirão a esse agradecimento. É impossível agradecer a todos que fizeram parte desse processo. Aqui nomearei alguns e não há dúvida que esquecerei outros, a esses minhas desculpas. Memória nunca foi o meu forte.

Aos meus pais, **Éfrem** e **Grácia**, que sempre me apóiam na realização dessa trajetória longa e que eu tenho prazer em realizar. A eles devo a minha vida e eterna gratidão.

Ao meu orientador **João Luiz Becker**, por quem tenho uma admiração explícita e por ter me orientado nessa trajetória. Ao professor **Denis Borenstein**, pelas conversas esclarecedoras, que foram de extrema importância em me nortear e pela oportunidade de visitar uma escola no exterior. Ao professor **Eduardo Ribas Santos**, quem eu admiro pelo jeito espontâneo de se expressar e por ter despertado o meu interesse pela subjetividade da pesquisa operacional. Ao professor **Leo Lopes**, pelos ensinamentos, correções da formulação, a oportunidade de permanecer na Universidade do Arizona e pela amizade extraordinária. À professora **Ângela Freitag Brodbeck**, por ter me fornecido o apoio necessário para me manter no mestrado, pela oportunidade de participar de projetos paralelos à dissertação e pela grande amizade propiciada. Ao professor **Flávio Sanson Fogliatto**, por contribuir com diversas correções e atentar para diversos pontos nebulosos. À professora **Denise Lindstrom Bandeira**, por ser a "madrinha" desse trabalho, sendo a precursora da área na Escola de Administração.

Às minhas famílias, "*Greyskull*" (**Bruno**, **Keila** e **Mário**) e "Gaúcha" (**Abilio**, **Clarissa, Marcus, Paulo** e **André**) pela agradabilíssima convivência e por terem me acolhido como um membro de suas famílias. A todos os amigos, colegas e funcionários (da Escola de Administração, Café Moenda e Santo Antônio), os quais demorariam outra dissertação para enumerá-los.

Muito obrigado!

“All models are wrong…
but some models are useful”.
(George Box)

RESUMO

A logística de contêiner vem aumentando sua participação em volume de cargas transportadas, tornando-se a parcela mais significativa do tráfego de mercadorias. Com isso, o gerenciamento dos altos custos envolvidos com a aquisição, manutenção, manipulação e transporte desses contêineres tornam-se um problema relevante para as organizações. As alocações dos contêineres cheios e vazios são comumente vistos como dois sistemas distintos e estáticos e não de forma intregada e dinâmica. Há um número restrito de trabalhos na literatura desenvolvendo heurísticas integrando os sistemas, porém não foi encontrada uma formulação ótima para o problema. Logo, a questão para a dissertação é quão próximo estão os resultados das heurísticas encontradas na literatura, para o problema da alocação de contêineres, dos resultados ótimos. O presente trabalho apresenta uma formulação matemática para o problema de alocação dinâmica, e integrada, para contêineres cheios e vazios. A formulação foi testada com diversos cenários, objetivando saber o limite computacional das instâncias para a formulação. Como o problema é um problema *NP-Hard*, heurísticas são comumente apresentadas na literatura. Demonstra-se como podem ser realizadas comparações entre os resultados das heurísticas e os resultados ótimos e visam a constatação da importância de uma formulação ótima para comparações.

Palavras-chave: Alocação Dinâmica de Contêineres; Problema de Escalonamento; Problema de *Job-Shop*; Programação Matemática

ABSTRACT

Containers' Logistics has increased their importance in the goods transportion and nowadays, has the most important share of them. With that in mind, the management of high costs of acquisition, maintenance, manipulation and transportation of them became a significant problem to organizations. The problem of empty container allocation and load container allocation are commonly treated as two distinct, and static, systems, which means without integration and not dynamically. Just a couple of examples could be found of the two systems dynamically integrated, and no optimal model was found. So, the question here is how close heuristics' results are from the optimal results. A mathematical formulation is presented to the problem concerned with the integration and the dynamics associated to it. The formulation was tested with several scenarios to determine the maximum size that could be tested with optimal results, in an acceptable computacional time. Since the problem is a *NP-Hard* problem, heuristics approach are commonly used. Here is demonstrated how could be compare optimal solutions of the formulation and solutions from heuristics, and aim to demonstrate the significance of the optimal formulation.

Key-words: Dynamic Container Allocation; Scheduling Problem; Job-Shop problem; Mathematical Programming

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A organização, movimentação e armazenamento de materiais e pessoas são atividades de logística que visam a fornecer aos clientes do sistema o produto certo, no lugar certo e no tempo certo (GHIANI, LAPORTE e MUSMANNO, 2004). A logística de contêineres pertence a esse domínio e é uma atividade que se diferencia pela eficiência do seu gerenciamento, possibilitando economias de escala.

Desde os anos 60, quando foi introduzido, o contêiner – embalagem secundária para transporte em navio – tem aumentado a sua participação no volume de cargas transportadas ao ponto de se tornar a parcela mais significativa do tráfego de mercadorias (TALEB-IBRAHIMI, CASTILHO e DAGANZO, 1999). Dejax e Crainic (1987) afirmam que o gerenciamento dos altos custos envolvidos com a aquisição, manutenção, manipulação e transporte desses contêineres se tornam uma questão relevante para as organizações. Corroborando com Dejax e Crainic (1987), Kroon e Vrijens (1995) apresentam quatro pontos os quais devem ser observados no contexto do gerenciamento dos contêineres:

- Quantos contêineres devem estar disponíveis no sistema;
- Quantos depósitos de contêineres devem existir e quais são as suas localizações;
- Como a distribuição, a coleta e a realocação dos contêineres devem ser organizada; e
- Quais são as taxas apropriadas de serviços, distribuição e coleta.

Dejax e Crainic (1987) apontam a escassez de literatura sobre o gerenciamento de fluxos vazios em geral. Especificamente para contêineres vazios, a alocação dinâmica de contêineres (DCA, do original em inglês, *Dynamic Container Allocation*) começou a ser observada por Crainic, Gendreau e Dejax (1993). Os autores afirmam ser um problema significante para as companhias de navegação. Mais recentemente, Shitani *et al.* (2007) destacam que já há uma quantidade considerável de pesquisa na área, mostrando a importância da logistica destes contêineres.

Após este trabalho seminal em 1993, o problema de DCA passou a ser comumente visto separadamente da alocação dos contêineres cheios, sendo esses dados como parâmetros para a alocação dos vazios. Os padrões de uso ocasionam o desbalanceamento entre a oferta e demanda, gerando custos para as organizações realocarem esses contêineres (SHEN E KHOONG, 1995). Bandeira (2005) reforça esta afirmação, ressaltando ainda que isto decore das diferentes necessidades econômicas de cada região. Ou seja, na medida em que há regiões mais importadoras e outras mais exportadoras, o desbalanceamento é inevitável.

A pesquisa operacional pode contribuir, e de fato contribui, com diversas abordagens para os problemas logísticos. Eles são considerados em sua maioria como problemas intratáveis (*NP-Hard*) devido à natureza matemática discreta dos mesmos. Este fato é comprovado pelo uso constante de heurísticas como métodos de solução para o gerencimento de contêineres nos trabalhos encontrados na literatura (LAI, LAM e CHAN, 1995; SHEN e KHOONG, 1995; CHEUNG e CHEN, 1998; BANDEIRA, 2005; LI *et al*., 2007).

O foco da presente dissertação é a alocação de contêineres cheios e vazios dinamicamente integrados. Para simplificação, alocação de contêineres é doravante tratada como sinônimo à alocação dinâmica e integrada de contêineres cheios e vazios.

A primeira abordagem integrando os dois sistemas de contêineres é apresentada por Bandeira (2005), quem, entretanto não explicitou uma formulação matemática exata para o problema. Com isso, não se pôde avaliar a qualidade das soluções encontradas em relação aos resultados ótimos. Em recente artigo, aautora, em colaboração com outros co-autores, apresenta uma formulação matemática baseada no problema de alocação de veículos de múltiplos depósitos (MDVSP, do original em inglês, *Multiple Depot Vehicle Scheduling Problem*) (BANDEIRA, BECKER e BORENSTEIN, 2009). Os autores adaptam a formulação de MDVSP com foco na redução de custos. Entretanto, não apresentam comparações entre os resultados da heurística desenvolvida com os resultados de sua formulação.

Como integrante do mesmo grupo de investigação, apresenta-se aqui uma formulação baseada em outro importante problema clássico, o escalonamento para

tarefas em máquinas (do original em inglês, *m/n Job-Shop problem*) (CONWAY, 1967), com foco na redução de atrasos na entrega (do original em inglês, *tardiness*).

Da revisão e discussão acima emerge uma proeminente questão de pesquisa: **quão próximo estão os resultados das heurísticas encontradas na literatura, para o problema da alocação de contêineres, dos resultados ótimos?**

1.1 OBJETIVOS

O objetivo geral da dissertação pode ser descrito como: estabelecer um instrumental básico de comparação entre resultados oferecidos por heurísticas encontradas na literatura e resultados ótimos para o problema da alocação dinâmica e integrada de contêineres cheios e vazios.

O objetivo geral, acima declarado, decompõe-se nos seguintes objetivos específicos:

i. Definir e descrever a situação problemática de alocação dinâmica e integrada de contêineres cheios e vazios;

ii. Desenvolver uma formulação matemática para o problema definido;

iii. Implementar computacionalmente a formulação desenvolvida;

iv. Testar a implementação computacional com diferentes cenários; e

v. Demonstrar como podem ser realizadas as comparações entre as soluções ótimas encontradas pela formulação desenvolvida e as encontradas por heurísticas.

1.2 ORGANIZAÇÃO DA DISSERTAÇÃO

Até este ponto, neste capítulo introdutório, o contexto e a motivação para realizar a investigação proposta nesta dissertação foram justificados. A seguir, no capítulo 2, aprofunda-se a contextualização do problema de pesquisa, com uma revisão da literatura pertinente, incluindo uma apresentação dos modelos de escalonamento para tarefas em máquinas. No capítulo 3 é apresentado o procedimento metodológico utilizado na dissertação. Em seguida, no capítulo 4, são apresentadas a descrição do sistema e a formulação desenvolvida. Na seção 5 é apresentada a heurística elaborada visando demonstrar como podem ser realizadas as comparações entre as soluções ótimas encontradas pela formulação desenvolvida e as encontradas por heurísticas. No capítulo 6, são apresentadas as comparações entre as soluções encontradas pela heurística desenvolvida e as soluções ótimas da formulação matemática. Por último, no capítulo 7, apresentam-se as considerações finais do trabalho.

2 REVISÃO DA LITERATURA

No presente capítulo é aprofundado o contexto no qual o presente trabalho se insere. Além da contextualização são apresentados os trabalhos encontrados na literatura sobre o problema da alocação e movimentação de contêineres. Por fim, o problema de alocação de tarefas em máquinas é apresentado, pois tal problema é usado como inspiração para a formulação do modelo para a alocação de contêineres.

2.1 CONTÊINER

O contêiner (do original em inglês, *shipping container*) é definido seguindo a classificação de Kroon e Vrijens (1995), como embalagem secundária reusável. Alshamrania, Mathur e Ballou (2007) complementam a definição como um tipo de reuso direto. Assim, o contêiner é uma embalagem utilizada para o envio de materiais que pode ser reutilizada e, normalmente, não precisa ser reprocessada para sua reutilização[1].

Mais especificamente, contêiner é uma caixa que normalmente segue as especificações – padrões – da Organização Internacional de Padrões, (ISO, em inglês, *International Standards Organization*), e é medido em unidades equivalente a vinte pés (TEUs, em inglês, *Twenty-foot Equivalent Units*). A tabela 2.1, apresenta os especificações dos contêineres mais comuns, de 20 TEUs e de 40 TEUs.

[1] Aqui se entende reprocessamento como um processo mais complexo do que uma simples manutenção. Em raros casos o contêiner precisa ser reprocessado, porém isso é decorrente de alguma anormalidade.

Tabela 2.1– Tamanhos padrões de contêineres

Tipo	Tamanho (m^3)	Tara (kg)	Capacidade (kg)	Capacidade (m^3)
ISO 20	5,899 × 2,352 × 2,388	2.300	21.700	33,13
ISO 40	12,069 × 2,373 × 2,405	3.850	26.630	67,8

Fonte: Adaptada de Ghiani, Laporte e Musmanno (2004, p. 11)

Há outras formas de diferenciação. Ele pode ser refrigerado, ventilado, fechado, aberto no topo, como também, diversas outras formas menos convencionais. (GHIANI, LAPORTE e MUSMANNO, 2004). Vale ressaltar que ainda pode ser diferenciado quanto à sua propriedade, podendo pertencer à companhia de navegação, ser objeto de arrendamento a curto prazo ou arrendamento a longo prazo.

Quase cinco décadas depois de entrar no mercado internacional para fretes marítimos (STEENKEN, VOß e STAHLBOCK, 2004), o contêiner é hoje parte indissociável do transporte marítimo. Yun e Choi (1999) ressaltam que 90% do transporte marítimo internacional de cargas se movem através de portos e, desses, 80% move-se por meio de contêineres.

2.2 TRANSPORTE CONTEINERIZADO

O transporte conteinerizado trata do envio de mercadorias usando a embalagem secundária, como definida na seção 2.1. Esse transporte pode ser atendido por cinco modalidades básicas água, trem, caminhão, ar e dutos ou combinações dessas modalidades (transporte intermodal) (BALLOU, 1998). Para Ghiani, Laporte e Musmanno (2004), na prática, o transporte intermodal não leva em consideração todas as possíveis combinações e apenas algumas se tornam convenientes, como ar-caminhão, trem-caminhão e água-caminhão, sendo as outras formas de combinações de baixa relevância.

Os principais atores (do original em inglês, *players*) envolvidos no transporte contenerizado são: (i) embarcadores (do original em inglês, *Shippers*) – são tanto os

produtores quanto os distribuidores; (ii) transportadores (do original em inglês, *Carriers*) – são os atores envolvidos com o transporte rodoviário, marítimo e ferroviário, assim como, o operador de transporte (do original em inglês, *supply transportation service*); e (iii) governo – responsável pelo fornecimento da infra-estrutura e da regulamentação (GHIANI, LAPORTE e MUSMANNO, 2004).

Além das duas características acima, classifica-se o tipo de transporte conteinerezado em relação ao seu curso. O curso longo (do original em inglês, *long-haul*) envolve os transportes entre instalações (do original em inglês, *facilities*) e o curso curto (do original em inglês, *short-haul*) consiste em tarefas de curta duração feita dentro da uma mesma área, em geral dentro de uma mesma instalação (GHIANI, LAPORTE e MUSMANNO, 2004).

Dada a complexidade envolvida, o gerenciamento deste transporte apresenta problemas. Ballou (1998) menciona vários apsectos a serem levados em conta na busca de solução, com: o preço, o tempo médio de trânsito (do original em inglês, *average transit time*), a variabilidade do tempo de passagem, e as perdas e danos. Em outro contexto logístico Lampert e Harrington *(apud* BALLOU, 1998) apresentam como aspecto mais revelante a habilidade das partes envolvidas cumprirem os prazos de entrega, assim aperfeiçoando o nível de serviço.

A seguir são apresentadas formulações matemáticas desenvolvidas para apoiarem o gerenciamento do transporte conteinerezido. Como o enfoque da dissertação está no transporte de curso longo, não serão apresentados trabalhos que lidem com o gerenciamento de curso curto.

2.3 MODELOS DE ALOCAÇÃO DE CONTÊINERES VAZIOS

Na literatura, o gerenciamento do transporte conteinerizado é abordado por diferentes métodos. Nessa seção são apresentadas duas formulações matemáticas para a alocação de contêineres vazios.

Fluxo de redes é a formulação mais comum para a alocação de contêineres vazios (CHEUNG e CHEN, 1998). Para Bandeira (2005), o problema de alocação desses contêineres é visto como um problema de transbordo. O problema de transbordo é uma extensão do modelo de fluxo de redes, com pontos intermediários por onde os itens passam depois de embarcarem na origem e antes de chegarem ao destino. A formulação apresentação por Goldbarg (2000) é adaptada ao contexto dos contêineres e reproduzida a seguir.

Conjunto de índices

M	conjunto de fornecedores {1,2,...,*m*}
S	conjunto de depósitos {1,2,...,*s*}
N	conjunto de demandantes {1,2,...,*n*}

Parâmetros

o_i	quantidade de contêineres disponíveis em *i*; $i \in M$
d_j	quantidade de contêineres solicitada em *j*; $j \in N$
c_{ik}	custo unitário de percorrer $\overline{ik}$; $i \in M$; $k \in S$
w_{kj}	custo unitário de percorrer $\overline{kj}$; $k \in S$; $j \in N$
f_k	custo unitário de ativar *k*; $k \in S$
a_k	capacidade de armazenamento de *k*; $k \in S$

Variáveis de decisão

x_{ik}	fluxo que pecorre $\overline{ik}$; $i \in M$; $k \in S$; $x_{ik} \in \Re_+^*$
y_{kj}	fluxo que pecorre $\overline{kj}$; $k \in S$; $j \in N$; $y_{ik} \in \Re_+^*$
v_k	variável binária que indica a ativação de um depósito; $v_k = 1$ se o depósito é ativado; $v_k = 0$ caso contrário

Função objetivo

$$\min z = \sum_{i \in M} \sum_{k \in S} c_{ik} x_{ik} + \sum_{k \in S} \sum_{j \in N} w_{kj} y_{kj} + \sum_{k \in S} f_k v_k \qquad (01)$$

Sujeito a

$$\sum_{i \in M} x_{ik} \leq a_k v_k \qquad \forall k \in S \qquad (02)$$

$$\sum_{k \in S} y_{kj} = d_j \qquad \forall j \in N \qquad (03)$$

$$\sum_{i \in M} x_{ik} = \sum_{j \in N} y_{kj} \qquad \forall k \in S \qquad (04)$$

$$\sum_{k \in S} x_{ik} \leq o_i \qquad \forall i \in M \qquad (05)$$

O foco é na redução dos custos. A minizimação (01) é em relação à redução dos custos de fluxos entre fornecedores e depósitos e entre depósitos e demandantes, respectivamente. Por último é considerado o custo de ativação dos depósitos.

A restrição (02) assegura que só ocorra fluxo entre fornecedores e cada depósito se o mesmo for ativado. Já a (03) garante que o atendimento da demanda solicitada de contêineres seja igual ao fluxo entre os depósitos e o respectivo demandante. A cotinuidade do fluxo é garantida pela (04), a qual exige que o fluxo de entrada seja igual ao fluxo de saída para cada depósito. Já a inequação (05) garante que os fornecedores não tenham fluxos maiores do que a sua quantidade disponível.

Dejax e Crainic (1987) apontam que os estudos na área de gerenciamento de fluxos vazios usam dos seguintes métodos: formulação matemática para otimização (linear, não-linear, inteira, fluxo de redes e outras); métodos estocásticos; e simulação.

Aqui é apresentada a formulação do problema de DCA proposta no trabalho seminal da área por Crainic, Gendreau e Dejax (1993). Esse trabalho inspira o trabalho de Bandeira (2005) que também o reproduz. Trata-se de uma formulação de fluxo de rede, dinâmica e determinística, e que possui um único tipo de contêiner. A preocupação é com a minimização dos custos, e esses são funções lineares. O sistema foi delineado para tratar da movimentação terrestre dos contêineres vazios, sendo a movimentação marítima dos contêineres considerada como externa ao sistema.

Apresentado graficamente na figura 2.1, o foco é dado para algumas possíveis movimentações: entre depósitos em terra e clientes demandantes (por exemplo, $\vec{ki}$); entre portos e depósitos em terra (por exemplo, $\overrightarrow{hk}$) e entre depósitos em terra e portos[2] (por exemplo, $\overrightarrow{kh}$); e entre os depósitos em terra (por exemplo, $\overleftrightarrow{jk}$).

Além dessas movimentações, há a possibilidade dos contêineres serem objetos de contratos de *leasing* de contêineres, empréstimos por outra companhia e aquisição. Essas possibilidades são inseridas na formulação através de um agrupamento de contêineres (do original em inglês, *pool*) que pode "enviar" para todas as instalações do sistema[3], sendo o custo de transporte o valor da inserção.

Um pressuposto importante é que o transporte da carga cheia é visto como parâmetro, ou seja, já foi realizado o planejamento antecipadamente. Assim o transporte do contêiner vazio está subordinado a essa atividade.

[2] Esse tipo de alocação é chamado de balaceamento de contêineres (do inglês, *balancing flows*)
[3] O agrupamento de contêineres está ligado apenas aos depósitos *j* e *k*, por simplificação no desenho. Na realidade ele é ligado a todas as instalações.

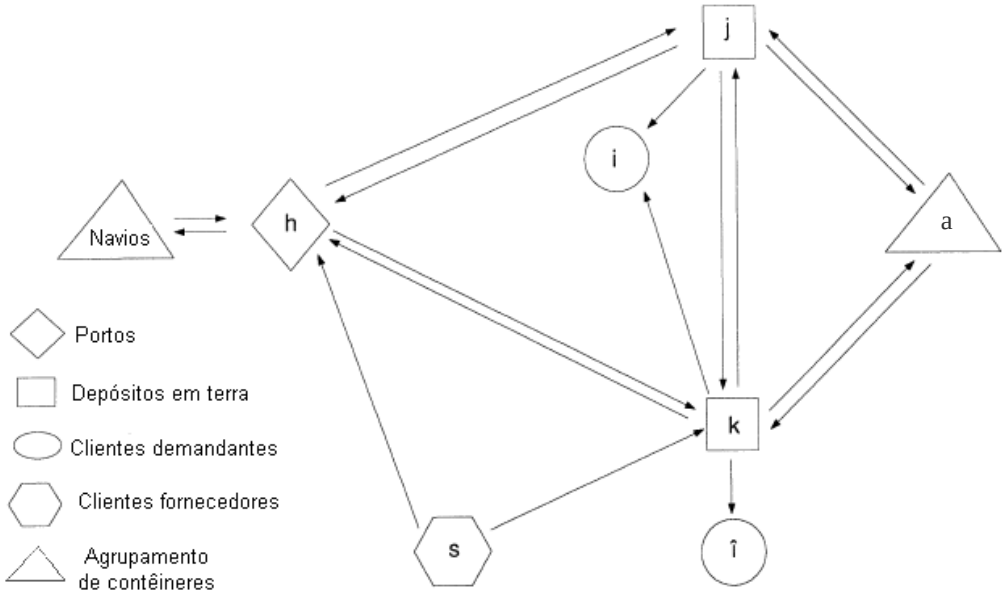

Figura 2.1 – Representação do problema de DCA
Fonte – Adaptada de Crainic, Gendreau e Dejax (1993, p. 106)

A preocupação da formulação é saber o número de contêineres que devem ser alocados de cada porto ou depósito. Para tal, foi desenvolvida a seguinte formulação:

Conjunto de índices

T horizonte de planejamento { 1,2,...,*r*}

D conjunto de depósitos em terra

H conjunto de portos

I^t conjunto de requisições dos clientes demandantes com prazo máximo no instante *t*; $t \in T$

S^t conjunto de requisições dos clientes fornecedores que devem ser recolhidos no instante *t*; $t \in T$

I_j^t conjunto de todas as requisições do cliente demandante *i* que podem ser satisfeitas pelo depósito *j* que iniciam no instante *t* e termina em $t + \tau_{ji}$, logo o cliente *i* pertence ao conjunto se $t' - \Delta_i \leq t + \tau_{ji} \leq t'$; $i \in I^{t'}$; $j \in D \cup H$; $t \in T$

$J_i^{t'}$ conjunto de todos os depósitos *j* que podem ser usados para satisfazer as requisições do cliente demandante *i* e o carregamento começa em *t'*; $t - \Delta_i \leq t' + \tau_{ji} \leq t$; $i \in I^t$; $J_i^{t'} \subseteq D \cup H$; $t \in T$

J_s^t conjunto de todos os depósitos *j* que podem receber os contêineres vazios do cliente fornecedor *s* no instante *t*; $J_s^t \subseteq D \cup H$; $s \in S^t$; $t \in T$

S_j^t conjunto de todos os clientes fornecedores *s* de onde o depósito *j* pode receber contêiner no instante *t*; $s \in S^{t'}$; $t' = t - \tau_{sj}$; $j \in D \cup H$; $t \in T$

Parâmetros

Δ_i a janela de tempo de entrega do cliente *i*; $\Delta_i \geq 0$

K_i^t quantidade de contêineres enviados para o cliente demandante *i* antes de *t*, ou seja, ajusta a demanda identificada para o horizonte de tempo *T;* $i \in I^t$; $t \in T$

K_j^t quantidade de contêineres enviados para do depósito *j* que devem chegar no instante *t'*, ou seja, ajusta a demanda identificada para o horizonte de tempo *T*; $j \in D \cup H$; $t \in T$

X_i^t demanda do cliente demandante *i* por contêineres que devem chegar no máximo no instante *t*; $i \in I^t$; $t \in T$

X_j^t demanda de exportação do porto *j* no instante *t*, que representa as demandas por contêineres vazios fora do escopo do sistema; $j \in H$; $t \in T$

Y_s^t suprimento de contêineres do cliente fornecedor *s* que estão disponíveis no instante *t*; $s \in S^t$; $t \in T$

Y_j^t suprimento de contêineires do porto *j* no instante *t* que representa os suprimentos por contêineres vazios fora do escopo do sistema; $j \in H$; $t \in T$

$\tau_{\eta\mu}$ tempo de traslado da origem *η* para o destino *μ*, onde a origem e o destino representam depósitos em terra, clientes e portos. Esses tempos são independentes do intervalo e os tempos de manuseio nas instalações são considerados como incluso nesse tempo

b_j^t quantidade de contêineres emprestados, arrendados ou adquiridos no depósito *j* no instante *t*, $j \in D \cup H$; $t \in T$

e_j^t demanda por contêineres vazios insatisfeita no porto *j* no instante *t*; $e_j^t \geq 0$; $j \in H$; $t \in T$

c_{ji}^t custo unitário do transporte do depósito *j* para o cliente demandante *i* no instante *t*; $j \in D \cup H$; $i \in I^t$; $t \in T$

c_{sj}^t custo unitário do transporte do cliente fornecedor *s* para o depósito *j* no instante *t*; $s \in S^t$; $j \in D \cup H$; $t \in T$

c_{jk}^t custo unitário de transporte do depósito *j* para o depósito *k* no perído *t*; $j,k \in D \cup H$; $t \in T$

c_j^t custo de permanência de um contêiner de fora do sistema no depósito *j* no período *t*; $j \in D \cup H$; $t \in T$

$\bar{c}_j^t$ custo unitário de ingresso do contêiner no sistema sendo o ingresso no depósito *j*; $j \in D \cup H$; $t \in T$

$\tilde{c}_j^t$ custo unitário de penalidade por não atender a demanda externa ao sistema no porto *j* no instante *t*; $j \in H$; $t \in T$

l_{jk}^t limite inferior de balanceamento de contêineres iniciado do depósito *j* no instante *t* para o depósito *k*; $j,k \in D \cup H$; $t \in T$

u_{jk}^t limite superior de balanceamento de contêineres iniciado do depósito *j* no instante *t* para o depósito *k*; $j,k \in D \cup H$; $t \in T$

Variáveis de decisão

v_{ji}^t quantidade de contêineres alocados no instante *t* do depósito *j* para o cliente *i*; $v_{ji}^t \in \Re_+^*$; $j \in D \cup H$; $i \in I_j^t$; $t \in T$

v_{sj}^t quantidade de contêineres vazios do cliente fornecedor *s* para o depósito *j* que chegam no instante $t - \tau_{sj}$; $v_{sj}^t \in \Re_+^*$; $s \in S^t$; $j \in J_s^t$; $t \in T$

w_{jj}^t quantidade em estoque do depósito *j* no final do instante *t*, ou seja, o fluxo de *j* no período *t* para *j* no instante *t*+1; $w_{jj}^t \in \Re_+^*$; $j \in D \cup H$; $t \in T$

w_{jk}^t quantidade de contêineres vazios do depósito *j* para o depósito *k* no instante *t*; $w_{jk}^t \in \Re_+^*$; $j,k \in D \cup H$; $t \in T$

Função objetivo

$$\min \sum_{t=1,2,...,T} \left\{ \sum_{j \in D \cup H} \left(\sum_{i \in I_j^t} c_{ji}^t v_{ji}^t + \sum_{k \in D \cup H} c_{jk}^t w_{jk}^t + c_j^t w_{jj}^t + \bar{c}_j^t b_j^t \right) + \sum_{j \in H} \tilde{c}_j^t e_j^t + \sum_{s \in S^t} \sum_{j \in J^t} c_{sj}^t v_{sj}^t \right\} \quad (06)$$

Sujeito a

$$\sum_{t" \le t} \sum_{j \in J^{t'}} v_{ji}^{t'} = X_i^t - K_i^t \qquad \forall i \in I^t, t = 1,2,...,T \quad (07)$$

$$\sum_{t" \le t} v_{sj}^t = Y_s^t \qquad \forall s \in S^t, t = 1,2,...,T \quad (08)$$

$$w_{jj}^t = w_{jj}^{(t-1)} + \sum_{t' \le t} \left(\sum_{k \in D \cup H | \tau_{kj} = t - t'} w_{kj}^{t'} + \sum_{s \in S_j^t} v_{sj}^t \right) + \qquad \forall j \in D, t = 1,2,...,T \quad (09)$$

$$b_j^t + K_j^t - \sum_{k \in D \cup H} w_{jk}^t - \sum_{i \in I_j^t} v_{ji}^t$$

$$w_{jj}^t = w_{jj}^{(t-1)} + \sum_{t' \le t} \left(\sum_{k \in D \cup H | \tau_{kj} = t - t'} w_{kj}^{t'} + \sum_{s \in S_j^t} v_{sj}^t \right) + \qquad \forall j \in H, t = 1,2,...,T \quad (10)$$

$$b_j^t + K_j^t - \sum_{k \in D \cup H} w_{jk}^t - \sum_{i \in I_j^t} v_{ji}^t - X_j^t + Y_j^t + e_j^t$$

$$l_{jk}^t \le w_{jk}^t \le u_{jk}^t \qquad j,k \in D \cup H, t = 1,2,...,T \quad (11)$$

A minimização (06) é o somatório de todos os custos (depósito-cliente demandante, depósito-depósito, permanência no depósito, ingresso de um contêiner, penalidade por não atender exportação e cliente fornecedor-depósito, respectivamente) considerados no sistema.

Essa minimização está sujeita as restrições: (07) onde a demanda identificada do cliente é tratada, sendo satisfeita por todos os depósitos ligados ao cliente demandante que respeitem a janela de tempo. O parâmetro K_i^t é para as demandas já satisfeitas antes de iniciar o horizonte de planejamento; (08) trata de se respeitar o suprimento dos clientes fornecedores que podem ser movidos para os portos ou os depósitos em terra; (09) trata dos depósitos em terra, sendo possível calcular os estoques de contêineres vazios no final do período, calculando as quantidades de contêineres anteriores, entre depósitos, para os clientes, contêineres inseridos no sistema, contêineres enviados antes do horizonte de tempo planejado subtraídos dos contêineres enviados no instante *t* para os outros depósitos e os clientes, respectivamente; (10) faz o mesmo que a (09), porém para os portos; e (11) que trata de que os balanceamentos de contêineres respeitem os seus limites inferiores e superiores.

A seguir é apresentada uma revisão dos trabalhos encontrados na literatura para o problema de DCA, enfatizando os métodos de soluções abordados. Além destes, é revisado o único método de solução intregada, apresentado por Bandeira (2005) e publicado em Bandeira, Becker e Borenstein (2009).

2.4 TRABALHOS RELACIONADOS À ALOCAÇÃO DE CONTÊINERES

O primeiro trabalho encontrado é White (1972), quem apresenta um algoritmo *out-of-kilter* para o problema de alocação de contêineres vazios. A preocupação é com o dinamismo, que é tratado como um problema de transbordo. Quando e onde estão os contêineres vazios, e consequentemente a quantidade de contêineres, são

pressupostos importantes do sistema. Antecedência e atraso são tratados através de custos, e não como restrições que devem ser atendidas.

Apesar de não apresentarem White (1972), Lam, Lee e Tang (2007) sintetizam os trabalhos relacionados ao contexto da alocação dinâmica de contêineres vazios. A classificação é ilutrada na figura 2.2. Essa classificação se presenta em dois eixos: o tipo de modelo (determinístico, estocástico e simulação) e o nível de aplicação (estratégico, tático e operacional).

Para a presente dissertação, só os trabalhos que se enquadram como operacionais, com seus diferentes tipos de modelos, são analisados. Os demais trabalhos tratam de localização de depósitos (GENDRON *et al*., 1995; BOURBEAU *et al.*, 2000) e gerenciamento de frotas (BEAUJON e TURNQUIST, 1991; GAO,1994; GENDRON *et al*., 1995; BOURBEAU *et al.*, 2000) não se enquadrando no contexto da dissertação. Alem desses, Powel e Carvalho (1998) não são considerados nessa revisão por tratarem de alocação de rebocadores e *flatcars*[4], os quais transportam contêineres.

Logo após White (1972), Ermol'ev, Krivets e Petukhov (1976) propoem abordar balanceamento de contêineres vazios por uma formulação de fluxo de rede. A rede considera a quantidade de contêineres vazios que podem ser alocados nos navios. Com isso, reduz-se o problema à busca de fluxo ótimo de contêineres vazios.

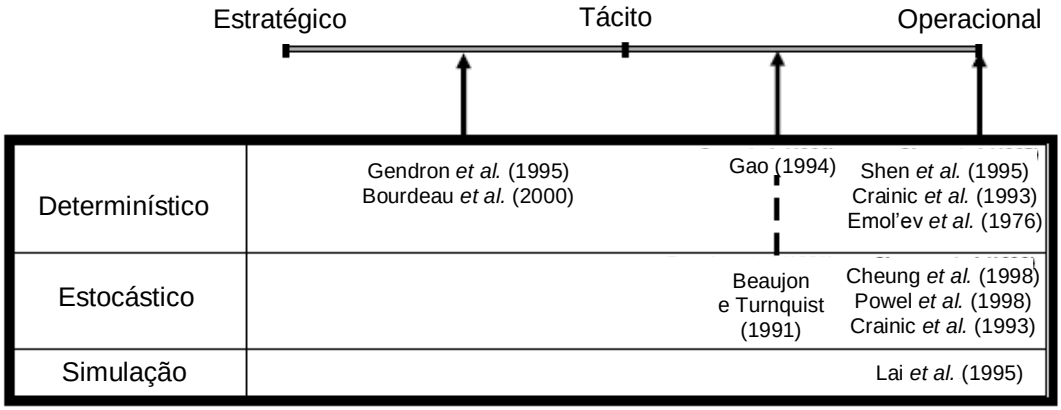

Figura 2.2 – Classificação das pesquisas em alocação dinâmica de contêiner
Fonte: Adaptada de Lam, Lee e Tang (2007, p. 266)

[4] *Flatcars* são vagões que não possuem laterais e teto. Servem para o transporte de contêineres em trens.

Na década de 1980, Dejax e Crainic (1987), revisando a literatura de fluxos vazios (do original em inglês, *empty flows*), salientam a importância da alocação dos contêineres vazios, principalmente pelo fato de poderem usar diferentes tipos de transportes. Os autores também constatam a persistência do problema na indústria conteinerizada (do original em inglês, *containerized shipping industry*), o que é reafirmado por Crainic, Gendreau e Dejax (1993), Bandeira (2005) e Lam, Lee e Tang (2007).

Uma reorganização, estendida, dos trabalhos para o problema de DCA é apresentada na figura 2.3. Ela representa a evolução temporal do problema de DCA, o qual foi formalizado no trabalho de Crainic, Gendreau e Dejax (1993). Estes autores apresentam formulações dinâmicas, sendo duas determinísticas e uma estocástica, para o planejamento da parte terrestre do comércio de contêineres. A formulação apresentada na seção 2.3 é a primeira apresentada no trabalho. Trata-se de uma formulação determinística para um único tipo de contêiner. Em seguida é apresentada uma formulação, também determinística, em que os contêineres de 20 TEUs e de 40 TEUs são considerados intercambiáveis. Por fim, apresenta-se um modelo estocástico, para um único tipo de contêiner, tendo como incerteza a demanda e o fornecimento dos mesmos.

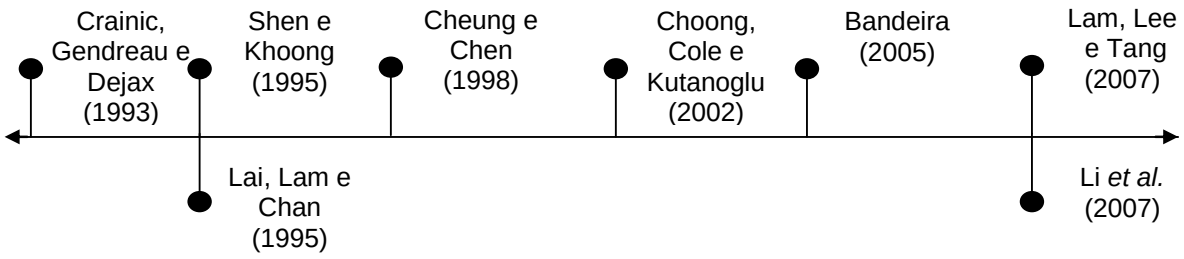

Figura 2.3 – Evolução das pesquisas em alocação dinâmica de contêiner
Fonte: Elaborada pelo autor

Em seguida, se preocupando com a questão prática, Lai, Lam e Chan (1995) desenvolveram um modelo de simulação, que considera as atividades operacionais de uma companhia de navegação na tentativa de melhor alocar os contêineres vazios. O problema, para os autores, está na previsão de demanda futura dos contêineres vazios e se usa heurísticas para identificação de políticas de custo.

No mesmo ano, Shen e Khoong (1995) apresentam um Sistema de Apoio à Decisão (SAD) para o gerenciamento da distribuição dos contêineres vazios e ressaltando a utilidade prática de um sistema como esse. Entretanto, eles não apresentam uma formulação matemática e o SAD funciona através do relaxamento de restrições. A questão é vista de maneira dinâmica, onde utiliza modelos de fluxo de rede com restrições relaxadas. Sua preocupação é com a escalabilidade e não com a solução ótima, como Lai, Lam e Chan (1995) e diferentemente do Crainic, Gendreau e Dejax (1993).

Já Cheung e Chen (1998) foca na incerteza associada à alocação. Os autores propõem um modelo de rede dinâmico e estocástico em dois estágios para auxiliar as transportadoras navais. Atenção especial é dada ao segundo estágio deste modelo, pois nele a minimização é em relação ao custo do excesso e da falta de contêineres. Todos os parâmetros aleatórios, por simplificação, são vistos ao mesmo tempo. Como métodos de solução são utilizados dois métodos de aproximação: quasi-gradiente e aproximação híbrida.

Crainic, Gendreau e Dejax (1993) já afirmavam que o tempo do planejamento deve ser determinado cuidadosamente. Em seu estudo, Choong, Cole e Kutanoglu (2002) tiveram como objetivo estudar o efeito do tamanho do período de planejamento para o gerenciamento dos contêineres vazios considerando transportes intermodais que usam da estrutura de rede do problema. Eles demonstram, empiricamente, que um horizonte longo para um sistema com uma grande quantidade de contêineres não necessariamente é melhor do que um mais curto.

Pela primeira vez, Bandeira (2005) aborda o problema da alocação e movimentação de contêineres vazios integradamente com os cheios. Primeiramente é visto como um modelo de rede e transbordos, sem dinamismo. O dinamismo é satisfeito através do uso de uma heurística que opera em estágios a alocação dos contêineres

cheios e vazios. No entanto, não é apresentada uma comparação dos resultados obtidos pela heurística com resultados ótimos de uma formulação matemática.

Posteriormente ao trabalho de Bandeira (2005), dois trabalhos envolvendo o problema de DCA podem ser encontrados. Lam, Lee e Tang (2007) retomam a preocupação com a otimização do sistema. Estes autores desenvolvem uma abordagem por aproximação dinâmica, e estocástica, para a alocação dos contêineres vazios. Apesar de realizarem a otimização em uma escala reduzida, alegam que a abordagem pode ser escalável. Eles também chamam a atenção para a melhoria, na formulação, que se pode obter adotando heurísticas comumente encontradas na literatura.

O outro trabalho é apresentado por Li *et al*. (2007). A preocupação é com a alocação de contêineres vazios entre multi-portos, não abordando o problema de forma integrada ao fluxo de entrega. O modelo tem como pressupostos: a quantidade de contêineres serem constante; uma quantidade de tempo discreta e finita; não há múltiplas rotas; e preocupando-se com a política de estoque dos contêineres vazios. Usa-se a estrutura de rede do problema, porém se tem estoques de contêineres que possuem um limite superior e inferior (*U,D*) e há uma política para quando a quantidade de contêineres estiver menos que *U*, importar até a quantidade *U* e quando estiver mais do que *D*, exportar até a quantidade *D*. Heurísticas são usadas para classificar o problema em dois casos e só se preocupam em resolver onde ocorre redução de custos nos dois casos.

No quadro 2.1, apresenta-se um resumo dos métodos usados nos trabalhos encontrados na literatura e se apresentam soluções ótimas para o problema. Como pode ser constatado mais de 70% dos trabalhos não apresentam resultados ótimos. Além disso, o único trabalho que trata da decisão dos contêineres cheios e vazios, silmutaneamente é Bandeira (2005).

Autores	Ano	Contêineres vazios (V) ou cheios (C)	Métodos de solução	Solução ótima
Crainic, Grendreau e Dejax	1993	V	Formulação matemática dinâmica e determinística	Não
		V	Formulação matemática dinâmica e determinística	Não
		V	Formulação matemática dinâmica e estocástica	Não
Lai, Lam e Chan	1995	V	Simulação	Não
Shen e Khoong	1995	V	Formulação matemática dinâmica e determinística	Sim
Cheung e Chen	1998	V	Formulação matemática dinâmica e estocástica	Não
Choong, Cole e Kutanoglu	2002	V	Formulação matemática dinâmica e determinística	Sim
Bandeira	2005	V e C	Heurístico	Não
Lam, Lee e Tang	2007	V	Formulação matemática dinâmica e estocástica	Não
Li *et al.*	2007	V	Formulação matemática dinâmica e determinística	Sim

Quadro 2.1 – Resumo das pesquisas em DCA
Fonte: Elaborado pelo autor

Apesar de Bandeira, Becker e Borenstein (2009) apresentarem uma formulação baseada no MDVSP para o problema, não são demonstradas comparações entre os resultados da heurística e os resultados da formulação apresentada. Fica clara uma lacuna com relação à demonstração de resultados ótimos para o problema de alocação de contêineres.

2.5 PROBLEMAS DE ESCALONAMENTO E SUAS HEURÍSTICAS

Uma importante função gerencial é a coordenação e o controle de atividades complexas. Há uma categoria de problemas que está preocupada tanto com a alocação de recursos, como a seqüência em que as operações são realizadas (JONHSON, 1974). Essa categoria é abordada na pesquisa operacional e problemas classificados

assim, são denominados de problemas de escalonamento (do original em inglês, *scheduling*).

As subseções 2.5.1, 2.5.2 e 2.5.3 são baseadas em Conway (1967) e apresentam o clássico problema de escalonamento para tarefas em máquinas. As mesmas são apresentadas devido a sua utilização como inspiração para a formulação proposta para alocação e movimentação integrada de contêineres cheios e vazios, em especial aproximando o problema de alocação de contêineres do caso encontrado na subseção 2.5.2.

2.5.1 O Problema Generalizado n/m de *Job-Shop*[5]

O problema generalizado *n/m* de *Job-Shop* (do original em inglês, *The General n/m Job-Shop Problem*) é de fácil visualização e compreensão, porém sua solução é extremamente difícil de ser obtida. A fascinação por um problema tão simples de estruturar, mas tão complexo de solucionar, explica o porquê de pesquisas continuarem a ser elaboradasnos dias atuais.

2.5.1.1 Descrição do Problema Generalizado n/m de *Job-Shop*

Considera-se um sistema que possui máquinas e tarefas que precisam ser alocadas nas máquinas. Essas tarefas são divididas em operações que são independentes uma das outras. Tais operações precisam ser realizadas na sequência da tarefa, ou seja, não é permitido o paralelismo de operações de uma mesma tarefa.

[5] Por conveniência o nome *job-shop* é mantido ao invés de escalonamento para tarefas em máquinas. Mesmo em português, o nome *job-shop* é bastante utilizado na literatura. A tradução é apresentada para um melhor entendimento do problema.

Entretanto as tarefas são independentes, ou seja, a sequência em que as tarefas são realizadas não altera seu tempo de processamento.

A figura 2.4 apresenta esquematicamente um problema com cinco tarefas e três máquinas. Cada tarefa *i* é representada como um conjunto de g_i blocos. Cada bloco representa uma operação e seu tamanho é proporcional ao seu tempo de processamento. Cada operação tem três identificadores, *i*, *o* e *k*: *i* – o número da tarefa a qual a operação pertence; *o* – o número da seqüência da operação; e *k* – o número da máquina em qual a operação deve ser processada.

Por exemplo, a Tarefa 1 exige o uso das três máquinas. A primeira operação deve ser processada na Máquina 2, a segunda operação deve ser processada na Máquina 3, e por fim a Operação 3 deve ser executada na Máquina 1. A mesma lógica pode ser usada para as outras tarefas.

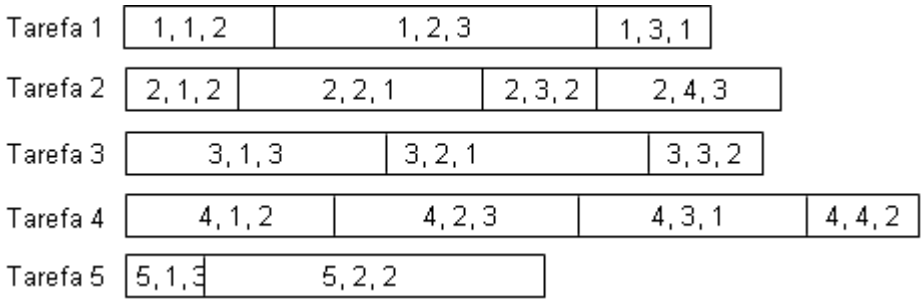

Figura 2.4 – Um problema generalizado 5/3 de *Job-Shop*
Fonte: Adaptada de Conway (1967, p. 104)

Soluções para esse problema se constituem em escalonar essas operações, e consequentemente as tarefas, nas máquinas. Um possível modo de apresentar a solução é por meio de um gráfico de *Gantt*, organizando as operações por máquinas, respeitando as restrições dos tempos de execução de cada operação.

Objetivando minimizar o tempo total de uso das máquinas, a solução ótima é apresentada na figura 2.5. A lógica é que a Máquina 1 inicia ociosa e espera a execução da Operação 1 da Tarefa 2 na máquina 2 (2, 1, 2). Com o término desta, inicia-se a Operação 2 da Tarefa 2 na Máquina 1 (2, 2, 1). Em seguida, a Operação 2

da Tarefa 3 (3, 2, 1), pois a Operação 1 (3, 1, 3) da mesma tarefa já foi concluída. Por fim, a Operação 3 da Tarefa 4 (4, 3, 1) e a Operação 3 da Tarefa 1 (1, 3, 1) são executadas. A mesma lógica pode ser usada para as Máquinas 2 e 3.

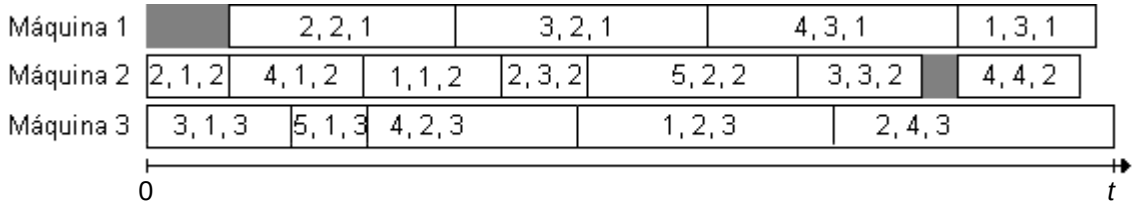

Figura 2.5 – Gráfico de *Gantt* do problema 5/3.[6]
Fonte: Adaptada de Conway (1967, p. 104)

É importante ressaltar que cada operação ocupa uma máquina por todo o seu tempo de processamento, não sendo permitida a sobreposição de operações nas máquinas. Também, não é permitida a preempção da operação, ou seja, parar de processá-la antes do seu término e continuar a sua execução em outro instante futuro.

2.5.1.2 Formulação do Problema de *Job-Shop*

A seguir, é apresentada a formulação de Manne (1960 *apud* CONWAY, 1967) para o problema de *Job-Shop*. Diferentemente da instância apresentada na subseção anterior, essa formulação exige que cada tarefa seja processada uma única vez por cada máquina, ou seja, as tarefas só utilizam cada máquina uma única vez. Assim, apenas é necessário explicitar qual tarefa está usando a máquina, sem necessidade de explicitar qual operação exatamente.

Esta versão simplificada atende as propostas da presente dissertação, pois como se verá no capítulo 4, cada carga ocupa uma única vez algum contêiner. Isso

[6] Em Conway (1967) a figura 6-2 e a figura 6-4, da página 104, estão na ordem inversa. Logo a figura reproduzida aqui é a figura 6-2.

equivalerá a cada tarefa (carga) ser processada uma única vez por alguma máquina (contêiner).

Assim, temos como formulação para o problema de *Job-Shop*:

Conjunto de índices

I conjunto das tarefas {1,2,..,*n*}

J^I conjunto das operações *j* da tarefa *i*; $i \in I$

K conjunto das máquinas {1,2,...,*m*}

Parâmetros

p_{ik} tempo de processamento da tarefa *i* na máquina *k*; $i \in I$; $k \in K$

r_{ijk} constante binária que determina se a operação necessita da máquina; $r_{ijk} = 1$ se a *j-ésima* operação da tarefa *i* requer a máquina *k;* $r_{ijk} = 0$ caso contrário; $i \in I$; $j \in J^i$; $k \in K$

M constante suficientemente grande

Variáveis de decisão

t_{ik} instante de início da tarefa *i* na máquina *k*; $t_{ik} \geq 0$

y_{ijk} variável binária auxiliar indicativa de precedência entre tarefas; $y_{ijk} = 1$ se a tarefa *i* precede a tarefa *j* na máquina *k*; $y_{ijk} = 0$ caso contrário

Função objetivo

$$\min \sum_i \sum_k r_{imk} t_{ik} \tag{12}$$

Sujeito a

$$\sum_k r_{ijk}(t_{ik} + p_{ik}) \leq \sum_k r_{i(j+1)k} t_{ik} \qquad \forall j \in J^i; j \neq m \forall i \in I; \forall k \in K \tag{13}$$

$$(M+p_{jk})y_{ijk} + t_{ik} - t_{jk} \geq p_{jk} \qquad \forall j \in J^i; \forall i \in I; \forall k \in K \tag{14}$$

$$(M + p_{ik})\,(1 - y_{ijk}) + t_{jk} - t_{ik} \geq p_{ik} \qquad \forall j \in J^i; \forall i \in I; \forall k \in K \qquad (15)$$

A minimização (12) é o somatório dos instantes de início das últimas operações (*m*) de todas as tarefas. Conway (1967) argumenta que esta minização euqivale a minimizar o fluxo das tarefas. A restrição (13) garante a execução das operações de cada tarefa, respeitando a ordem sequêncial. As restrições duas (14) e (15) garantem a ordenação das tarefas nas máquinas conectando logicamente a ordenação das tarefas com as variáveis binárias y_{ijk} (0 ou 1) e o artifício da constante suficientemente grande.

2.5.2 Sequenciamento em Máquinas Paralelas Idênticas com Tempo de *Setup* Dependente da Seqüência

Conforme já tangenciado na subseção 2.5.1.2, utiliza-se nessa dissertação de uma correspondência entre contêineres e máquinas e entre cargas e tarefas. Assim, tarefas ocupam máquinas para serem processadas e cargas ocupam contêineres para serem trasladadas.

A situação de transporte envolvendo contêineres é tal que os contêineres são, e princípio, indistinguíveis quando vazios. Isto é as “máquinas” (contêineres) têm capacidade de processar qualquer “tarefa” (carga) e há uma multiplicidade de “máquinas” semelhantes (vários contêineres idênticos). Desta forma, verifica-se que os modelos de sequenciamento mais próximos a esta correspondência são modelos de sequenciamento em máquinas paralelas idênticas com tempo de *setup* dependente da sequência (MPITSDS), pois um contêiner (máquina) após trasladar (processar) uma carga (tarefa) será realocado para trasladar (processar) outra carga (tarefa) podendo haver necessidade de traslado do contêiner vazio (tempo de *setup*).

Nesta subseção se apresenta uma revisão dos trabalhos de escalonamento, com foco no escalonamento com o tempo de *setup* dependente da sequência. A importância

da área pode ser constatada na revisão da literatura de Allahverdi *et al.* (2008). Nela foram encontrados, e analisados, trezentos artigos sobre o tema.

Conway (1967) apresenta uma formulação para uma única máquina, inspirada no problema do caixeiro viajante (TSP, do original em inglês, *traveling-salesman problem*). O autor destaca como diferença para o problema independente de sequência o fato de normalmente as tarefas não serem mais divididas em operações. Também são citados alguns exemplos de casos reais que dependem de sequência. Um dos exemplos é o caso de uma manufatura de pinturas onde é necessária a limpeza da máquina quando se altera de uma cor para outra.

Diferentemente da formulação generalizada apresentada na subseção 2.5.1, as tarefas não têm como atributo qual máquina necessitam, já que todas são idênticas. Além disso, é preciso atentar para a sequência, pois os tempos de *setup* da máquina entre duas tarefas distintas variam de acordo com cada tarefa. O exemplo apresentado por Conway para a manufatura de tintas pode ser generalizado para o caso de máquinas paralelas. Pode-se imaginar que no lugar de uma máquina há um conjunto de máquinas idênticas que podem ser usadas para manufaturar o produto.

Além do exemplo acima, outros problemas podem se inspirar no problema de MPITSDS. Um exemplo recente é apresentado por Liu e Kozan (2009) que se inspiram no problema para resolver o escalonamento de trens. Eles propõem um método de solução que é testado e validado, em escala real, com os dados da Queensland Rail.

Uma ilustração do problema pode ser vista na figura 2.6, onde as tarefas 1 e 2 compartilham a Máquina 1, porém sua utilização não pode ser simultânea. Para esse caso é necessário saber o tempo de *setup* da Máquina 1 entre a tarefa 1 e 2 ou da tarefa 2 e 1.

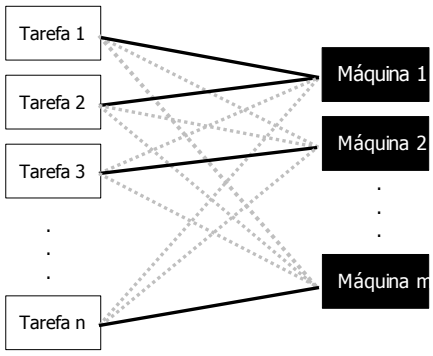

Figura 2.6 – Problemas de máquinas paralelas idênticas e com tempo de setup dependente da seqüência
Fonte: Elaborada pelo autor

Os trabalhos tratando especificamente do problema de MPITSDS apresentam-se, em ordem cronológica, no quadro 2.2. Como pode ser visto, todos os trabalhos usam heurísticas para solucionar o problema.

Jeong, Kim e Lee (2001) tratam da construção de *displays* de cristal líquido, desenvolvendo uma abordagem heurística para o problema divindido-se em duas heurísticas distintas para cada estágio de produção. Uma formulação matemática é apresentada discretizando-se os intervalos de tempo e os usando como índices, sendo conveniente apenas para pequenas janelas de tempo.

Autores	Ano	Heurística
Jeong, Kim e Lee	2001	sim
Eom *et al*.	2002	sim
Kim *et al*.	2002	sim
Kim, Na e Chen	2003	sim
Yi e Wang	2003	sim

Quadro 2.2 – Publicações sobre máquinas paralelas idênticas e com tempo de setup dependente da seqüência e não loteado
Fonte: Elaborado pelo autor

Eom *et al.* (2002) apresentam uma heurística eficiente em três fases para o problema: a primeira fase, agrupam as tarefas; a segunda, melhora o agrupamento através de uma busca tabu; e a terceira, usa um limiar (do original em inglês, *threshold*)

para alocar as tarefas. Tal limiar é determinado por uma previsão futura (do original em inglês, *look-ahead*) de alocação. Não é apresentada formulação matemática.

Uma meta-heurística usando *simulated annealing* é apresentada por Kim *et al.* (2002) serve de inspiração para Kim, Na e Chen (2003) que estedem a pesquisa anterior e sugerem outras três abordagens heurísticas para o problema.

Yi e Wang (2003) apresentam uma heurística para o problema usando algoritmos genéticos com lógica nebulosa. Os autores apresentam resultados que demonstram a usabilidade em escala industrial. É apresentada uma formulação matemática *NP-Hard* para o problema, levando em conta as antecipações e os atrasos das tarefas.

Os trabalhos de Jeong, Kim e Lee (2001) e Yi e Wang (2003) são os únicos a apresentarem formulações matemáticas. O contexto dos mesmos foi limitado a um pequeno número de máquinas, sendo possível discretizar os tempos em conjuntos de pequena cardinalidade.

3 PROCEDIMENTO METODOLÓGICO DO TRABALHO

Neste capítulo é apresentado como foi desenvolvida a dissertação. Os passos do procedimento metodológico são apresentados no Quadro 3.1 e se dividem em três grandes fases, apresentadas a seguir.

Fase		Preocupações Metodológicas
i	Formulação do sistema integrado	Seguir o processo clássico da pesquisa operacional.
ii	Desenvolvimento da heurística	Apresentar uma heurística que use da estrutura da formulação do problema de *Job-Shop*.
iii	Comparações entre a heurística e a formulação	Comparar os resultados, e os tempos de execução, da heurística com os resultados, e os tempos de execução, da formulação (otimalidade).

Quadro 3.1 – Fases do Procedimento Metodológico e suas preocupações
Fonte: Elaborado pelo autor

3.1 FORMULAÇÃO DO SISTEMA INTEGRADO

O método abordado pelo trabalho é o processo clássico da pesquisa operacional para formulação de problemas adaptado de Wagner (1972) e Jensen (2003):

Estágio 1: Descrição e suposições da formulação

Interpretativamente é feita a descrição da situação e de seu processo (percepção da situação), definição das fronteiras da abordagem do problema e suposições assumidas. São apresentados as entidades envolvidas, os parâmetros, as variáveis de decisão, o objetivo da minimização e as restrições do problema de uma maneira dissertativa (seção 4.1).

Estágio 2: Construção do modelo

O modelo, quando passa do verbal para termos lógicos e quantitativos, como toda abstração, terá aspectos intangíveis que não poderão ser tratados na forma analítica. Logo, para ser útil, tem que considerar dois objetivos: ser tratável (capaz de ser resolvido) e válido (representar a situação descrita). Como Jensen (2003) afirma, esses dois objetivos regularmente são contraditórios e nem sempre são contemplados. A formulação analítica é apresentada (seção 4.2).

Estágio 3: Execução das análises

Esse estágio visa à validade, tanto do modelo como da solução e é apresentado na seção 4.4. Testa se a computação feita está correta, se é adequado ao problema original e se as suposições não são demasiadamente restritivas.

Como em Bandeira (2005), houve dificuldade de acesso a dados reais e foi necessário a geração de dados aleatórios. Esses dados são usados apenas para testes de desempenho computacional, visando saber a limitação computacional para se encontrar a solucionar ótima.

3.2 DESENVOLVIMENTO DA HEURÍSTICA

A busca por heurísticas para o problema de alocação dinâmica e integrada de contêineres cheios e vazios revelou-se infrutífera. Como salientado na seção 2.4, o único trabalho que aborda o problema é Bandeira (2005). A heurística lá desenvolvida, é de razoável complexidade e sua implementação (programação) foge ao escopo dessa dissertação. Optou-se, portanto, por desenvolver uma heurística simples que servisse aos propósitos de entendimento do objetivo específico *v*.

A heurística elaborada, e apresentada no capítulo 5, baseia-se em uma abordagem computacionalmente simples e que exije pouco tempo computacional, ou seja, em tempo polinomial. A possibilidade de se encontrar facilmente a solução ótima

para um problema de *Job-Shop*, onde *m* é igual a *n*, ou seja, a quantidade de máquinas é igual a quantidade de tarefas, é a inspiração para a heurística elaborada.

3.3 COMPARAÇÕES ENTRE A HEURÍSTICA E A FORMULAÇÃO

A comparação entre os resultados obtidos pelas soluções da heurística apresentada e os resultados da solução ótima são apresentados no capítulo 6. Testes são realizados em relação ao desempenho computacional e à proximidade da solução ótima, ou seja, qualidade da solução.

4 FORMULAÇÃO DO PROBLEMA DE ALOCAÇÃO E MOVIMENTAÇÃO DE CONTÊINERES

A formulação é apresentada como descrita na seção 3.1. Primeiramente é apresentada a descrição do sistema. Em seguida, a formulação matemática para o problema de alocação de contêineres. Por fim, a execução das análises e possíveis atualizações da formulação sugerida, demonstrando a extensibilidade da mesma.

4.1 DESCRIÇÃO E SUPOSIÇÕES DAS FORMULAÇÕES

Seguindo a classificação de Lam, Lee e Tang (2007) para problemas de alocação dinâmica de contêineres, o tipo de decisão é para transporte de curso longo e operacional.

Considera-se que já existe, e não é alterado, um *design* da rede logística, ou seja, instalações e serviço de transporte são fixos e ativos. Instalações são onde os materiais devem ser processados e podem ser fábricas, centros de montagem, armazéns, centros de distribuição, pontos de transbordos (do original em inglês, *transhipment points*), terminais de transporte, lojas de varejo, entre outros (GHIANI, LA PORTE e MUSMANNO, 2004). Para fins da modelagem, não há distinção entre modalidades de transportes, apenas pressupõe-se que existe um único serviço de transporte, que usa contêineres, conectando todas as instalações. Não há limites (do original em inglês, *bounds*) de contêineres transportados. Há uma quantidade finita, e constante, de contêineres, que podem ser reutilizados no sistema. Há um único tipo de contêiner no sistema, como usualmente tratado na literatura, e rotulado por TEUs. Não há distinção de quem são os seus proprietários dos contêineres, apenas que pertencem ao sistema, e todas as instalações têm a liberdade de usar qualquer dos contêineres.

Há uma quantidade fixa de cargas demandando transporte entre quaisquer duas instalações no sistema. Essas cargas são demandadas em instantes de tempo e é delimitado um horizonte de tempo onde estas cargas são conhecidas a priori.

As variáveis de decisões do modelo referem-se a alocação e escalonamento, isto é, a designição de qual contêiner é utilizado para transportar qual carga e em que instante de tempo este transporte será iniciado. Todos os tempos de transporte são considerados constantes.

Uma ilustração do sistema pode ser encontrada na figura 4.1. Nela estão representadas as instalações (origem e destino), as cargas que serão enviadas apartir dessas instalações, e os contêineres que são utilizados para esse envio. Por exemplo, a carga 1 da Instalação A (origem) é enviada no contêiner 1, em um instante anterior, para a Instalação B (destino). Em um instante posterior, o mesmo contêiner *k* é usado para enviar a carga 5 da Instalação B (origem) para a Instalação C (destino). A mesma lógica serve para as outras cargas e contêineres.

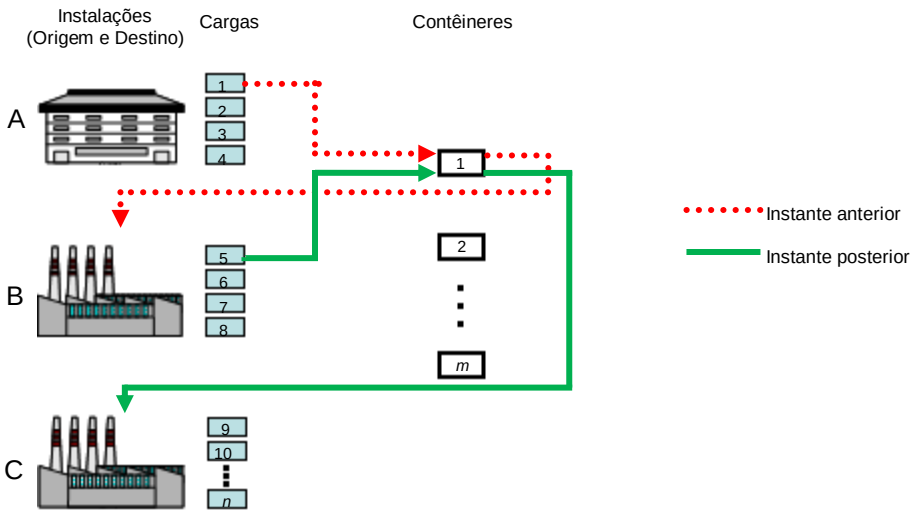

Figura 4.1 – Representação gráfica da utilização dos contêineres pelas cargas
Fonte: Elaborada pelo autor

Não há distinção entre as instalações, todas podem enviar e receber contêineres. O planejamento das demandas por transporte (cargas) pode ser representada através de uma matriz quadrada cuja ordem é igual ao número de instalações do sistema. As

linhas representam as origens das cargas e as colunas seus destinos. Os elementos representam a quantidade de cargas que necessitam ser enviadas entre as instalações. O planejamento é realizado a cada instante de tempo, isto é, haverá *t* matrizes de origem x destino das cargas. A figura 4.2 ilustra um exemplo desta situação, onde o horizonte temporal (*t*) é igual a cinco e há três instalações no sistema.

1	1	2	3
1	-	0	2
2	3	-	1
3	0	0	-

2	1	2	3
1	-	4	2
2	1	-	0
3	3	0	-

3	1	2	3
1	-	2	5
2	2	-	1
3	3	0	-

4	1	2	3
1	-	4	0
2	5	-	0
3	2	1	-

5	1	2	3
1	-	3	0
2	2	-	0
3	0	1	-

Figura 4.2 – Planejamento do envio das cargas
Fonte: Elaborada pelo autor

Sem perda de generalidade, assume-se que cada carga ocupa exatamente um contêiner, não sendo permitida preempção, ou seja, um transporte não pode ser interrompido durante seu curso.

Até esse ponto, sabemos que as cargas devem ser enviadas de uma instalação para outra instalação e que esse envio é planejado para um certo horizonte de tempo. O planejamento de envio ilustrado na figura 4.2, pode ser reorganizado como na figura 4.3. A figura 4.2 explora a dimensão temporal do horizonte de tempo planejado. A figura 4.3, em forma de lista, é mais adequada para o uso pelos sistemas computacionais.

cargas	instalação de origem	instalação de destino	instante de demanda
1	1	3	1
2	1	3	1
3	2	1	1
⋮	⋮	⋮	⋮
47	3	2	5

Figura 4.3 – Lista de cargas e seus atributos
Fonte: Elaborada pelo autor

Observa-se que a quantidade total de cargas demandantes da figura 4.2 (quarenta e sete cargas) é equivalente a quantidade de linhas geradas na figura 4.3. Por exemplo, a quantidade de cargas demandantes entre a Instalação 1 e a Instalação 3, no instante 1 (igual a dois) equivale as linhas 1 e 2 da figura 4.3, que possuem como atributos: a instalação de origem (Instalação 1); a instalação de destino (Instalação 3); e o instante de demanda (*t* igual a um). A mesma lógica se aplica as outras cargas com os seus atributos.

4.2 FORMULAÇÃO DO PROBLEMA DE ALOCAÇÃO DE CONTÊINERES

Definido o sistema, pode-se elaborar uma formulação matemática. Como o foco são os instantes de envio das cargas, e consequentemente os tempos decorrentes desse fato, não é necessário considerar quais são as origens e os destinos das cargas, apenas que existe o transporte e que se tem um tempo de duração para esse processamento. Ressalta-se que há a possibilidade de transporte vazio do contêiner.

Apresentada a descrição do sistema, podem-se resumir as características do sistema apresentada na seção 4.1 como:

i. Há um número finito de instalações, contêineres e cargas;
ii. O planejamento é feito para *t* janelas de tempo*;*
iii. Todas as instalações podem enviar e receber cargas; e
iv. Há uma única rota de envio de cargas entre cada par de instalações (rota ótima) com capacidade ilimitada para enviar contêineres;

Tendo tais características, e se inspirando no problema de *Job-Shop*, os conjuntos e parâmetros do modelo podem ser definidos como:

Conjunto dos índices

I conjunto das cargas

K conjunto dos contêineres

Parâmetros

p_i tempo de transporte da carga *i* entre sua origem e seu destino

o_i instante de demanda da carga *i*

tt_{ij} tempo de transporte do contêiner vazio entre destino da carga *i* e origem da carga *j*

M constante suficientemente grande

Sem perda de generalidade, o parâmetro tt_{ij} pode incorporar, além do tempo de transporte do contêiner vazio propriamente dito: o tempo de descarregamento (desova) da carga *i* no seu destino; o tempo de manutenção necessária para o contêiner ser colocado em condições de uso no transporte da carga *j*; e o tempo de carregamento (estufamento) da carga *j* na sua origem. Tal simplificação é equivalente ao do parâmetro $\tau_{\eta\mu}$ de Crainic, Gendreau e Dejax (1993), apresentado na seção 2.3. A simplificação é possível, pois os tempos incorporados dependem da sequência das cargas. Representados os conjuntos e parâmetros do modelo, configuram-se as variáveis de decisão:

Variáveis de Decisão

t_i instante de início de embarque da carga *i* ; $t_i \geq o_i$; $t_i \in \Re_+^*$

r_{ik} variável binária de alocação da carga *i ao* contêiner *k*; $r_{ik} = 1$ se a carga *i* é transportada no contêiner *k*; $r_{ik} = 0$ caso contrário

y_{ij} variável binária auxiliar de precedência entre cargas; $y_{ij} = 1$ se a carga *i* precede a carga *j*; $y_{ij} = 0$ caso contrário

Definadas as variáveis de decisão, resta formular um objetivo a ser alcançado. No contexto apresentado, nível de serviço está ligado ao atendimento tão cedo quanto possível das demandas por transporte. Consequentemente o objetivo é:

Função objetivo

$$\min \sum_i t_i \tag{16}$$

A qual está sujeita a:

Restrições

$$\sum_k r_{ik} = 1 \qquad \forall i \in I, \forall k \in K \tag{17}$$

$$M(3 - y_{ij} - r_{ik} - r_{jk}) + t_j - t_i - p_i - tt_{ij} \geq 0 \qquad \forall i \neq j \in I, \forall k \in K \tag{18}$$

$$M(2 + y_{ij} - r_{ik} - r_{jk}) + t_i - t_j - p_j - tt_{ji} \geq 0 \qquad \forall i \neq j \in I, \forall k \in K \tag{19}$$

A minimização (16) busca a redução dos atrasos (do original em inglês, *tardiness*) dos instantes de início de embarque das cargas. A equação (17) assegura que todas as demandas sejam atendidas, cada carga sendo enviada por apenas um contêiner. Já as inequações duais (18) e (19) servem para garantir que a sequência temporal seja respeitada para as cargas que utilizam o mesmo contêiner. Por exemplo, se a carga *i* preceder a carga *j* ($y_{ij} = 1$) e essas cargas usam o mesmo contêiner ($r_{ik} = 1$ e $r_{jk} = 1$), então o parâmetro *M* se torna inócuo e a inequação (18) assegura que o instante de embarque da carga *j* (t_j) só ocorra quando o contêiner estiver na sua origem ($t_i + p_i + tt_{ij}$). Já a inequação (19) se torna inócuo, pois o valor de *M* já a torna maior que zero, permitindo a viabilidade (do original em inglês, *feasibility*) desses valores.

4.3 FORMULAÇÃO DO PROBLEMA DE ALOCAÇÃO DE CONTÊINERES EXTENDIDA

A formulação apresentada na seção anterior fornece como solução a alocação de cargas a contêineres, assim como o escalonamento de transporte dessas cargas. Entretanto, há uma suposição implícita um tanto irreal. Os contêineres surgem, sem qualquer custo, nas instalações de origem das primeiras cargas onde são alocados. A presente seção apresenta uma extensão da formulação considerando a localização inicial de cada contêiner no sistema.

Usa-se o artifício de incluir um conjunto de cargas fictícias correspondendo ao número de contêineres existente no sistema, isto é, |*K*| cargas fictícias, onde |*K*| representa a cardinalidade do conjunto de contêineres. Define-se o novo conjunto de cargas como I', que é a união do conjunto I com as cargas fictícias. Estas novas cargas servem para determinar onde os contêineres devem iniciar no sistema. Para isso, o tempo de transporte deve ser igual a zero ($p_i = 0, \forall i \in I'-I$) e seus atributos definidos como:

i. Instalação de origem – qualquer valor;

ii. Instalação de destino – localização inicial do contêiner; e

iii. Instante de demanda – zero ($o_i = 0, \forall i \in I'-I$);

Além desses parâmetros, o tempo de transporte do contêiner vazio (tt_{ij}) precisa ser definido. Há dois possíveis casos para cargas fictícias:

- A carga *i* é uma carga fictícia e *j* é uma carga real ($i \in I'-I$, $j \in I$) – o valor do parâmetro é igual ao tempo necessário para o deslocamento da instalação onde o contêiner inicia até a origem da carga *j*. Por exemplo, se o contêiner iniciar na Instalação 1 e a primeira carga alocada (*j*) possuir a Instalação 3 como origem, tt_{ij} precisa ser igual ao tempo necessário para o deslocamento entre as Instalações 1 e 3.
- A carga *i* é qualquer carga (fictícia ou não) e a carga *j* é fictícia ($i \in I'$, $j \in I'-I$) – o valor do parâmetro tt_{ij} não é relevante, pois as

inequações duais (18) e (19) não permitem, por conta das alterações abaixo, o uso desses parâmetros. Entretanto, é conviniente mantê-los iguais a zero, para facilidade de cálculo.

As restrições também precisam ser estendidas. Para tal é necessária a criação dos conjuntos $K^i, i \in I'-I$, onde se determina qual carga fictícia representa cada contêiner. Assim, além da equação (17) e as inequações (18) e (19), devem ser incluídas as seguintes equações:

$$r_{ik} = 1 \qquad \forall i \in I'-I, k \in K^i \qquad (20)$$

$$r_{ik} = 0 \qquad \forall i \in I'-I, k \in K, k \notin K^i \qquad (21)$$

$$t_i = 0 \qquad \forall i \in I'-I \qquad (22)$$

$$y_{ij} = 1 \qquad \forall i \in I'-I, j \in I \qquad (23)$$

$$y_{ij} = 0 \qquad \forall i \in I'-I, j \in I'-I \qquad (24)$$

$$y_{ij} = 0 \qquad \forall i \in I, j \in I' \qquad (25)$$

A equação (20) assegura que a carga fictícia seja transportada pelo contêiner correto. Por outro lado, a equação (21) assegura que todos os casos em que o contêiner não é representado pela carga fictícia sejam iguais a zero. A equação (22) serve para garantir que todos os transportes comecem no instante zero e a equação (23) para que as cargas fictícias precedam todas as cargas reais. Nesta última, não é necessário identificar exatamente para qual contêiner, pois as equações (20) e (21) asseguram o correto funcionamento das inequações (18) e (19). Por fim, as equações (24) e (25) garantem que nenhuma carga fictícia, ou real, preceda uma fictícia.

É importante observar que as equações (21), (22), (24) e (25) não são necessárias, porém são inseridas para que todas as novas variáveis sejam tratadas como paramêtro na formulação.

4.4 EXECUÇÃO DAS ANÁLISES

A formulação foi implementada no ILOG OPL Studio©, versão 5.5.1, utilizando a linguagem OPL (do original em inglês, *Optimization Programming Language*), e é resolvida pelo *engine* ILOG Cplex©, versão 11.0. A implementação está dividida em dois tipos de arquivos: arquivo de modelo – onde se encontra o modelo em si; e arquivo de dados – onde se localizam os dados. Nos apêndices A, B e C são encontrados exemplos do arquivo de modelo para a formulação apresentada na seção 4.2, um exemplo do arquivo de dados da mesma formulação e a alteração necessária para a versão extendida, respectivamente.

Todas as implementações foram realizadas no modo *default* da *engine*. A ferramenta usa o algoritmo *Branch & Bound* para solucionar problemas inteiros mistos (do original em inglês, *mixed integer problem*). Vários métodos de geração de planos de corte e eliminação de colunas e linhas são executados no seu pré-processamento, porém a *engine* (ILOG Cplex) não disponibiliza a maneira como estas são implementadas. Além desses métodos, a ferramenta utiliza heurísticas para percorrer a árvore de solução, eficientemente.

Os parâmetros apresentados na seção 4.1, como também o vetor de contêineres, ilustrado na figura 4.4, são inseridos a partir de uma planilha do Microsoft Excel 2007© através do arquivo de dados. Um novo atributo (coluna) é inserido na lista de cargas correspondente ao tempo de traslado (p_i) para simplificação de implementação.

Contêineres	1	2	3	4	5

Figura 4.4 – Vetor de Contêineres
Fonte: Elaborada pelo autor

Para a matriz ilustrada na figura 4.5, temos que as linhas são as cargas precedentes e as colunas as cargas posteriores (parâmetro tt_{ij}). Os elementos dessa matriz equivalem ao tempo de transporte do contêiner vazio entre o destino da carga *i* (linha) e a origem da carga *j* (coluna). Assim, por exemplo, o valor do elemento a_{12}, que

é igual a sete, equivale ao tempo de transporte do contêiner vazio entre o destino da carga 1 e a origem da carga 2.

Na implementação, incluem-se os parâmetros tt_{ii}, que são inseridos com valores iguais a zero. Tais parâmetros são eliminados no pré-processamento da *engine*, porém são mantidos na declaração, por simplicidade de código. O mesmo vale para as variáveis y_{ii}.

Pre \ Pos	1	2	3	4
1	0	7	7	4
2	7	0	7	4
3	0	0	0	3
4	0	0	7	0

Figura 4.5 – Matriz tt_{ij} para um problema com quatro cargas
Fonte: Elaborada pelo autor

O valor do parâmetro *M* é iniciado com o valor de 300, um número razoavelmente grande para os valores usados nos testes. O uso deste número é devido à dificuldade de se calcular um valor ajustado (do original em inglês, *tightened*) para o mesmo.

Após a execução da *engine*, a solução encontrada, o vetor *t**, é inserida como um novo atributo da carga na mesma lista para ser usada na checagem da formulação. A *engine* gera registros das suas execuções, para cada rodada, os quais são armazenados para as análises.

Não foi possível acessar dados reais para os testes da formulação. Tal dificuldade também foi encontrada por Bandeira (2005) que usou dados realistas, porém não reais. Primeiramente foi usada uma instância, adotada anteriormente por Bandeira (2005), com 47 cargas.

Com uma quantidade de contêineres igual a cinco, a execução durou aproximadamente três horas e meia. Entretanto, não foi possível encontrar uma solução ótima e a árvore de soluções atingiu 1,6 *gigabytes* de tamanho, atingindo o limite computacional de armazenamento. Mesmo assim, o *gap* entre os limites superior e o inferior ainda era de 82,77%, não podendo determinar uma solução ótima aceitável

para o problema. Com isso, é necessária a criação de cenários visando a determinar o tamanho máximo para as instâncias que pode ser solucionável em um tempo computacional aceitável[7].

4.4.1 Gerador de Cenários

Os cenários são gerados pelo algoritmo encontrado no apêndice D, o qual é implementado em *Visual Basic for Applications 6.5*© (VBA) no *Microsoft Excel*© 2003.

Knüsel (2005) afirma não encontrar erros no gerador de números aleatórios do *Microsoft Excel*© 2003, porém considera a quantidade de números randômicos, que é de mais de trinta mil, baixa. Já McCullough e Wilson (2005) afirmam que como não houve alteração da função "ALEATÓRIO()" do programa, e como demonstrado em trabalhos anteriores, poderiam gerar números negativos. Nos testes realizados, não foram encontrados números negativos e a quantidade de números randômicos é considerada aceitável para o propósito da presente dissertação.

O foco é, portanto, descobrir o tamanho máximo para as instâncias do problema. Assim, geram-se os seguintes parâmetros:

Geração das Cargas

i. Instalação de origem e instalação de destino – um par no conjunto {(1,2) (1,3) (2,1) (2,3) (3,1) (3,2)}, uniformemente distribuído; e

ii. Instante de Demanda – um valor no conjunto {1, 2, 3, 4, 5, 6, 7, 8, 9}, uniformemente distribuído.

[7] A máquina usada para as análises possui um processador *Core* 2 de 2.00 GHz. com 2 *Gygabytes* de memória RAM e opera com sistema operacional *Microsoft Windows* XP.

Geração dos Tempos de Transporte

iii. Tempo de traslado entre a Instalação 1 e a Instalação 2 – um valor no conjunto {1, 2, 3, 4, 5, 6, 7, 8, 9}, uniformemente distribuído;

iv. Tempo de traslado entre a Instalação 1 e a Instalação 3 – um valor no conjunto {1, 2, 3, 4, 5, 6, 7, 8, 9}, uniformemente distribuído; e

v. Tempo de traslado entre a Instalação 2 e a Instalação 3 – um valor entre 1 e 9, uniformemente distribuído.

A figura 4.6 ilustra um exemplo dos tempos de transporte entre as instalações. Supõe-se que existem três instalações no sistema e os tempos de transportes foram gerados independentemente entre si.

orig \ dest	1	2	3
1	-	4	7
2	4	-	3
3	7	3	-

Figura 4.6 – Tempo de transporte entre as instalações
Fonte: Elaborada pelo autor

A escolha de uma pequena janela de tempo foi feita baseada na suposição de que um horizonte de tempo longo não necessariamente é melhor que um curto (CHOONG, COLE e KUTANOGLU, 2002).

Os valores de *iii*, *iv* e *v* são replicados para os percursos inversos. Por exemplo, o tempo de transporte entre a Instalação 2 e a Instalação 1 é replicado a partir do tempo de transporte entre a Instalação 1 e a Instalação 2. Os mesmo valores são também usados para preencher o atributo tempo de transporte entre a origem e destino de cada carga (parâmetro p_i), assim como a matriz de transporte do contêiner vazio (parametro tt_{ij}), baseado no destino da carga precedente e a origem da carga posterior.

4.4.2 Análises dos Resultados: Otimalidade

Foram solucionados trinta casos (gerados aleatoriamente) para cada instância do problema. No total se realizaram vinte e uma instâncias, expandindo o tamanho do problema de modo incremental. A primeira instância representa problemas com três cargas e dois contêineres (3 e 2). Essa é a única instância possível para três cargas, pois não são testadas as instâncias quadradas (quantidade de contêineres e cargas iguais) e nem aquelas com apenas um contêiner. Na sequência, são tratados os casos com quatro cargas. Há, então, duas instâncias possíveis (4 e 2 e 4 e 3). A mesma lógica é aplicada às outras instâncias. A figura 4.7 ilustra as instâncias analisadas, onde *n* e *m* representam cargas e contêineres, respectivamente.

n \ *m*	2	3	4	5	6	7
3	X					
4	X	X				
5	X	X	X			
6	X	X	X	X		
7	X	X	X	X	X	
8	X	X	X	X	X	X

Figura 4.7 – Instâncias usadas na análise dos resultados
Fonte: Elaborada pelo autor

O limitador temporal adotado foi oito horas. O maior tempo ocorreu na instância 8 e 5 com tempo máximo de execução de aproximadamente sete horas e meia. Testes foram feitos para a instância 8 e 6 tendo como tempo máximo de execução de aproximadamente duas horas e 8 e 7 com tempo máximo de aproximadamente quatro horas e meia. Gerou-se um caso para a instância 9 e 7, porém não se encontrou uma resposta ótima, mesmo depois de 12 horas de execução. Assim, os testes se limitaram até a instância 8 e 7. Os tempos máximos de execução para cada instância são apresentados no gráfico 4.1.

A discrepância no tempo de execução do caso 17 da instância 8 e 5 ocorre porque a escolha da política de busca na árvore de soluções não foi adequada[8]. A solução ótima foi encontrada em um pouco mais de 150 iterações e o restante das iterações, mais de quinze milhões, ocorreu para provar a otimalidade.

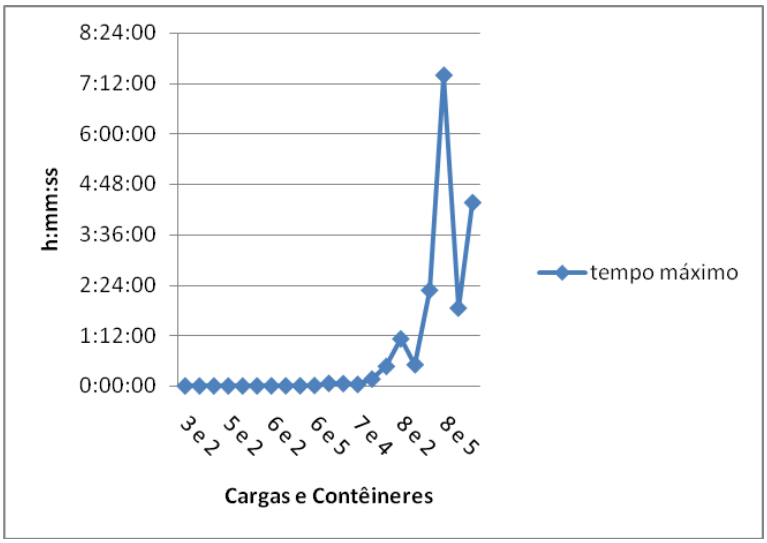

Gráfico 4.1 – Evolução dos tempos máximos
Fonte: Elaborado pelo autor

Como se tem uma grande dispersão nos tempos de execução, analisam-se as medianas, apresentadas no gráfico 4.2. Pode-se notar um grande incremento no valor da mediana para a instância 8 e 6. Tal distorção é uma característica dos problemas de escalonamento, pois o tempo para solucionar está associado à estrutura dos dados de entrada e não à quantidade de variáveis e equações.

Com isso em mente, a análise dos registros de varredura das árvores de solução busca padrões de comportamento. Infelizmente, não foi possível identificar um padrão para a discrepância significativo em casos distintos. O único padrão encontrado foi que a solução ótima é encontrada em praticamente todos os casos antes de se varrer dez mil nós, o que pode ser feito rapidamente. Apenas três casos não ocorreram: o caso 4

[8] Apesar da não adequação da política de busca para o caso específico, no geral a política tem uma qualidade aceitável, como se verá adiante.

da instância 8 e 3, o caso 14 da instância 8 e 3 e o caso 22 da instância 8 e 4. Entretanto, todas já possuíam uma solução próxima da ótima (sendo os valores encontrados 2,67%, 1,03% e 8% acima do resultado ótimo, respectivamente) demonstrando a razoável qualidade da política de busca adotada pela *engine* (Cplex).

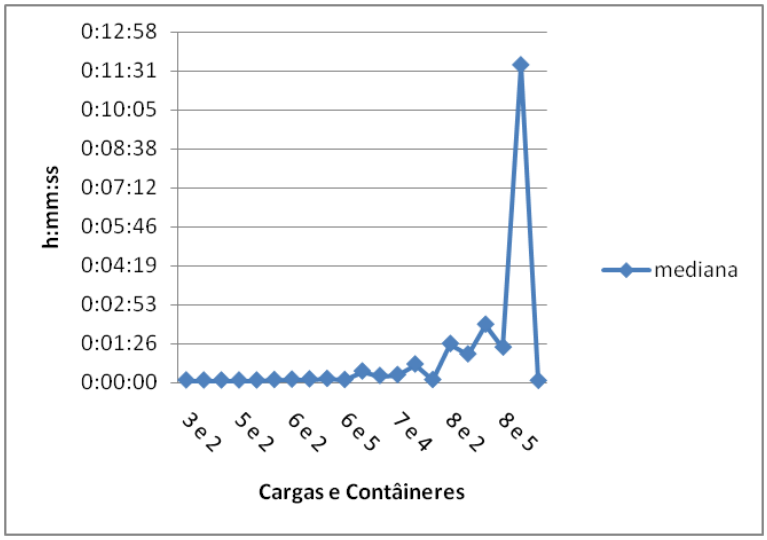

Gráfico 4.2 – Evolução dos tempos das medianas
Fonte: Elaborado pelo autor

4.5 POSSÍVEIS EXTENSÕES DA FORMULAÇÃO

Concluídas as três etapas apresentadas na seção 3.1, apresentam-se duas extensões da formulação encontrada na seção 4.2. As extensões visam a demonstrar a extensibilidade da mesma, sem que sejam necessários novos pressupostos, evidenciando a flexibilidade da formulação elaborada.

4.5.1 Prazo de Entrega

Lampert e Harrington *(apud* BALLOU, 1998) salientam que uma das principais preocupações em um sistema logístico é o cumprimento dos prazos de entrega (do original em inglês, *due date*). Para que isso seja contemplado pela formulação apresentada é necessária a inclusão de um novo parâmetro:

d_i prazo de entrega para a carga *i*

O novo parâmetro visa a determinar um limite máximo para a entrega da carga. Como já temos instante de demanda da carga como um limite inferior para a variável de decisão t_i, podemos incluir o parâmetro d_i como um limite superior para a entrega. Para tal, a declaração da variável t_i deve alterar para:

t_i instante do início de embarque da carga *i* e $d_i - p_i \geq t_i \geq o_i ; t_i \in \Re_+^*$

Com a alteração, garante-se que a carga *i* seja entregue entre a janela de tempo (d_i, o_i). Fica claro que a diferença entre o prazo de entrega e o tempo de processamento $(d_i - p_i)$ deve ser maior ou igual ao instante de demanda da carga (o_i) para que o espaço de solução seja viável.

4.5.2 Minimização do maior atraso

Além da incorporação de novas características, a formulação pode possuir outros objetivos, por exemplo, miniminizar o maior atraso possível. Um artifício usado para

esse objetivo é apresentado por Wagner (1972) e consiste na criação de uma nova variável:

w variável com valor igual ao maior atraso no embarque das cargas, $w \geq 0 \in \Re_+$

Essa variável substitui as variáveis usadas na minimização (16). Logo se precisa definir uma nova função objetivo:

Função objetivo

$$\min w \tag{26}$$

A minimização (26) tem como objetivo criar o menor valor possível para w, consequentemente o maior atraso no envio. Para que w assuma o valor equivalente ao maior atraso é necessária a inserção de uma nova inequação ao conjunto de restrições:

$$w \geq t_i - o_i \qquad \forall i \in I \tag{26}$$

Com (26) não se permite que os valores dos atrasos ($t_i - o_i$) ultrapassem o valor de w.

5 HEURÍSTICA

Como já destacado na seção 3.2, a única heurística encontrada na literatura é apresentada por Bandeira (2005). Por ser de complexa implementação foge ao escopo dessa dissertação a reimplementação da mesma. Uma heurística simples que serve aos propósitos, é desenvolvida e apresentada nesse capítulo. O desenvolvimento da mesma não visa a ter um desempenho eficaz, mas permitir a demonstração das comparações apresentadas no capítulo 6.

5.1 IDÉIA BÁSICA DA HEURÍSTICA

A heurística usa uma abordagem *greedy.* A idéia é organizar as cargas, pelos instantes de demanda, em ordem crescente. Após a ordenação, particionam-se as cargas em lotes de tamanho |*K*|, onde |*K*| representa a cardinalidade do conjunto de contêineres. O último lote pode ser igual ou menor a esse valor, dependendo do número de cargas. A avaliação para alocação é feita por contêiner, avaliando todas as cargas pertencentes ao lote.

Primeiramente, deve-se verificar se há demanda por contêiner na instalação onde se encontra. No caso afirmativo, o contêiner é alocado para a carga demandante. Já no caso de não haver uma carga demandante na instalação, analisa-se a instalação mais próxima, a com menor tempo de transporte para o contêiner vazio, e se verifica a demanda. Se houver carga demandante na instalação mais próxima, o contêiner é designado para o transporte vazio até esta e, em seguida, a carga é transportada para sua instalação destino. Se novamente não houver carga demandada, o contêiner será alocado à primeira carga que ainda não foi alocada, pois como estas estão organizadas por demanda, ela será a carga ideal.

Logo, pode-se definir o procedimento, em pseudo-português, como:

Início
Passo 1 **Ordena** cargas em ordem de demanda
Se |*I*| **igual** |*K*| **ou** |*K*| **igual** 1 **então**
Chama o Cplex
Se mod(|*I*|/|*K*|) = 0 **então**
Separar as cargas em |*I*|/|*K*| lotes
Se não
Separar as cargas em |*I*|/|*K*| + 1 lotes
Passo 2 **Para cada** lote **faça**
Para cada contêiner $\in K$ **faça**
Para cada carga $\in I$ **faça**
Se Instalação Destino da carga **igual** Instalação no qual se encontra o Contêiner **então**
Aloca a carga para o contêiner
Se não
Se Há Demanda na Instalação Mais Próxima **então**
Aloca a carga para o contêiner

No Passo 1, primeiramente, são ordenadas as cargas por seu instante de demanda. Se o problema possuir a quantidade de cargas igual à quantidade de contêineres, chama a *engine* Cplex. Também no caso degenerado em que a quantidade de contêineres é igual a um.

Ainda no Passo 1, se a divisão da cardinalidade do conjunto de cargas pela cardinalidade do conjunto de contêineres for igual a zero, a quantidade de lotes será igual ao valor da divisão. Se não, será o valor da divisão mais um, pois é necessário mais um lote para as cargas finais.

No Passo 2 são realizados três laços (lotes, contêineres e cargas, respectivamente). Nos laços são realizados os testes, apresentados anteriormente, para se alocar as tarefas. Checa-se um lote por vez e para cada lote alocam-se todas as tarefas do lotes, usando todos os contêineres. Em seguida, passa-se para o próximo

lote, seguindo este procedimento até que se execute a última tarefa demandada para o último contêiner.

A implementação do algoritmo pode ser encontrada no Apêndice E. A elaboração é em VBA na ferramenta Microsoft Excel 2007©. Como pode ser constatado, sua complexidade é de $O(n^3)$. Usam-se os dados gerados para a formulação estendida para validar o algoritmo. Realizaram-se trinta testes e suas execuções foram usadas para as comparações encontradas no capítulo 6.

6 POSSÍVEIS COMPARAÇÕES

Com os resultados dos trinta casos da formulação estendida e da heurística, podem-se realizar comparações objetivando verificar o desempenho da heurística em relação aos resultados ótimos.

6.1 COMPARAÇÃO DAS SOLUÇÕES

Analisando o gráfico 6.1, é possível notar grandes discrepâncias entre os valores encontrados pela heurística e os valores ótimos da formulação estendida. A maior discrepância de valores ocorre no caso 18, sendo o valor encontrado pela heurística 82% maior do que o valor encontrado pela formulação ótima. Na média, as heurísticas têm valor 28% mais alto, comparadas aos valores ótimos. O desvio padrão é, aproximadamente, 24%.

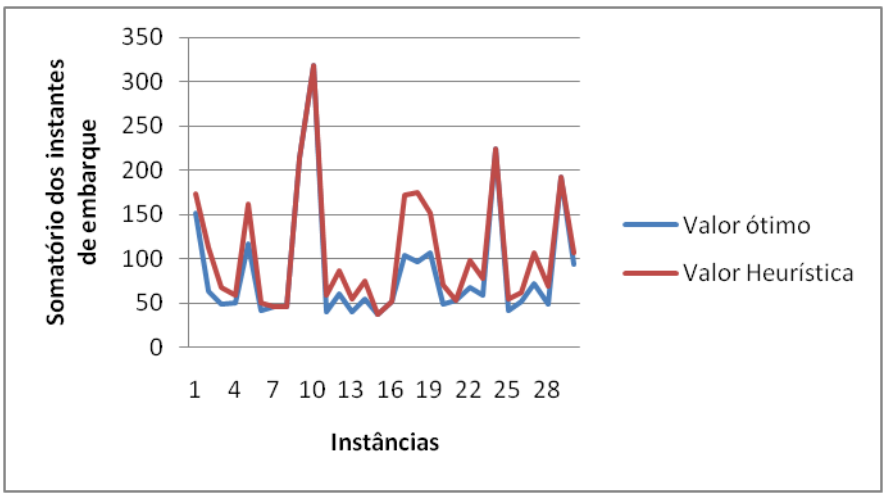

Gráfico 6.1 – Comparação dos resultados dos valores de resposta
Fonte: Elaborado pelo autor

6.2 COMPARAÇÃO DOS TEMPOS DE EXECUÇÃO

Analisando os dados do Gráfico 6.2, pode notar-se a variabilidade da solução ótima decorrente da estrutura poliédrica do problema, como já ressaltado na subseção 4.4.2. Com isso, a média dos tempos da heurística representa 26% do tempo usado para se encontrar os valores da solução ótima. Entretanto, essas médias ficam bastante influenciadas com os tempos discrepantes. Usando a mediana como referência, tem-se que o valor da heurísitca representa 0,16% do tempo de execução da formulação ótima.

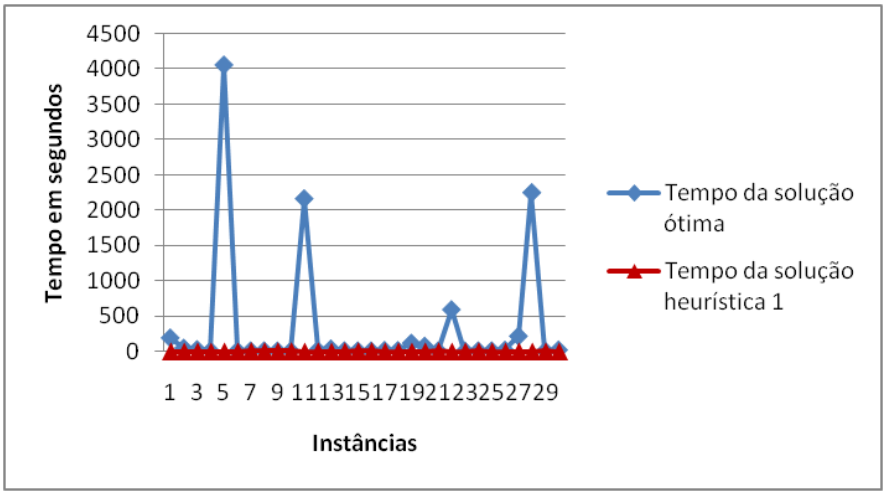

Gráfico 6.2 – Comparação dos tempos de execução
Fonte: Elaborado pelo autor

7 CONSIDERAÇÕES FINAIS

Como fechamento do trabalho é preciso analisar as comparações apresentadas e demonstrar que o objetivo da dissertação foi atingido. Também são propostos trabalhos futuros como continuação ao trabalho apresentado.

7.1 CONCLUSÃO E CONTRIBUIÇÕES

Na presente dissertação foi apresentada uma formulação matemática para alocação integrada de contêineres cheios e vazios. Tal formulação pode ser usada para encontrar resultados ótimos para o problema e assim ser possível realizar comparações das heurísticas encontradas na literatura e os resultados ótimos.

Baseado nas comparações do capítulo 6, conclui-se que, apesar da heurística ter um bom desempenho computacional, polinomial, não são retornados resultados promissores como solução quasi-ótima.

A formulação pode ser generalizada, além do contexto de logística de contêineres e pensada para o escalonamento de qualquer veículo de transporte (do original em inglês, *vehicle scheduling problem*), como também o caso de máquinas paralelas idênticas e com tempo de *setup* dependente da sequência.

A solução da heurística pode ser usada como valor para o parâmetro *M*, tornando o espaço de soluções viáveis um espaço mais ajustado, ou seja, reduzindo o espaço de busca de soluções.

Realizada uma abordagem próxima ao problema de *Job-Shop,* clássico problema da literatura, pode-se utilizar as heurísticas já apresentadas para esse problema, sendo necessárias pequenas adaptações na formulação aqui apresentada.

Como contribuição secundária, o desenvolvimento de um gerador de cenários para o envio de cargas é apresentado. A importância de tal gerador é a possibilidade de

gerar um banco de dados em comum para testes de heurísticas elaboradas para o problema de escalonamento de envio de contêineres cheios e vazios.

7.2 SUGESTÕES PARA TRABALHOS FUTUROS

A contribuição do trabalho é a formulação do modelo, apresentado na seção 4.2 e o aproximar do contexto do problema de *Job-Shop*. Como se trata de um problema *NP-Hard*, deve-se sugerir o uso de heurísticas, encontradas na literatura, para o problema de *Job-Shop*.

Uma possível heurística que pode ser usada, por já ter sido usada em escala industrial, é a apresentada por Yi e Wang (2003), a qual se utiliza de algoritmos genéticos com lógica nebulosa. Além do uso de heurísticas já encontradas na literatura, podem ser desenvolvidas outras meta-heurísticas. Uma meta-heurística que trabalhe conjuntamente otimização e métodos aleatórios é promissora, como por exemplo, um *GRASP* (do original em inglês, *Greedy Randomized Adaptive Search Procedure*). Essa meta-heurística pode começar usando a solução parcial apresentada pelo método ótimo, uma solução viável intermediária da *engine*, pois como foi demonstrado na análise apresentada na subseção 4.4.2, a *engine* encontra resultados quasi-ótimos em poucas iterações.

Outra forma de abordar é o desenvolvimento de novas heurísticas com políticas de alocação mais robusta e abordagens mais inteligentes do que a heurística apresentada no capítulo 6. Por exemplo, como no trabalho de Eom *et al.* (2002), que usa de métodos de previsão. Como a formulação conhece *a priori* os tempos de transporte e para onde as cargas estão sendo enviadas, pode-se calcular alguns padrões para o modelo que podem auxiliar nessa previsão.

REFERÊNCIAS BIBLIOGRÁFICAS

ALLAHVERDI, A., NG, C. T., CHENG, T. C. E., KOVALYOV, M. Y. A survey of scheduling problems with setup times or costs, *European Journal of Operational Research*, v. 187, n. 3, p. 985-1032, 2008.

ALSHAMRANI, A., MATHUR, K., BALLOU, R. H. Reverse logistics: simultaneous design of delivery routes and returns strategies. *Computers & OR, v.* 34, p. 595-619, 2007.

BALLOU, R. H. *Business logistics management* : planning, organizing and controlling the supply chain. 4.ed. Upper Saddle River: Prentice-Hall, 1999. 681 p.

BANDEIRA, D. L. *Alocação e movimentação de contêineres vazios e cheios* – um modelo integrado e sua aplicação. 2005. 124 f.Tese (Doutorado em Administração) – Programa de Pós-Graduação em Administração, Escola de Administração, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2005.

BANDEIRA, D.L., BECKER, J. L., BORENSTEIN, D. A DSS for integrated distribution of empty and full containers, *Decision Support Systems*, in press, 2009.

BEAUJON, G.J., TURNQUIST, M.A. A model for fleet sizing and vehicle allocation. *Transportation Science,* v. 25, n. 1, p. 19–45, 1991.

BOURBEAU, B., CRAINIC, T.G., GENDRON, B. Branch and bound parallelization strategies applied to a depot location and container fleet management problem. *Parallel Computing*, v. 26, p. 27–46, 2000.

CHEUNG, R. K., CHEN, C. A two-stage stochastic network model and solution methods for the dynamic empty container allocation problem. *Transportation Science* v. 32, n. 2, p. 142-162, 1998.

CHOONG, S. T., COLE, M. H., KUTANOGLU, E. Empty container management for intermodal transportation networks. *Transportation Research Part E, v.* 38, n. 6, p. 423-438, 2002.

CONWAY, R. W. *Theory of scheduling*. Reading: Addison-Wesley, 1967. 294 p.

CRAINIC, T. G., GENDREAU, M., DEJAX, P. Dynamic and stochastic models for the allocation of empty containers. *Operations Research*, v. 41, n. 1, p. 102-126, 1993.

DEJAX, P.J. , CRAINIC T.G. A Review of Empty Flows and Fleet Management Models in Freight Transportation. *Transportation Science*, v. 21, n. 4, 227-248, 1987.

EMOL'EV, Y. M., KRIVETS, T. A., PETUKHOV, V. S. Planning of shipping empty seaborne containers. *Cybernetics and Systems Analysis*, v. 12, n. 4, p. 644-646, 1976.

EOM, D.H., SHIN, H.J., KWUN, I.H., SHIM, J.K., KIM, S.S. Scheduling jobs on parallel machines with sequence-dependent family set-up times. *International Journal of Advanced Manufacturing Technology*, v.19, p. 926–932, 2002.

GAO, Q., An operational approach for container control in liner shipping. *Logistics and Transportation Review*, v. 30, n. 3, p. 267–282, 1994.

GENDRON, B., CRAINIC, T.G. A branch and bound algorithm for depot location and container fleet management. *Location Science*, v. 3, n. 1, p. 39–53, 1995.

GHIANI, G., LAPORTE, G., MUSMANNO R. *Introduction to logistics systems planning and control*. West Sussex: Wiley-Interscience series in systems and optimization, 2004, 360 p.

GOLDBARG, M. C. *Otimização combinatória e programação linear*: modelos e algoritmos 2. ed. Rio de Janeiro: Elsevier, 2005, 518.

JENSEN, P. A. *Operations research* : models and methods. Hoboken: John Wiley, c2003. xii, 675 p.

JEONG, B., KIM, S.W., LEE, Y.J. An assembly scheduler for TFT LCD manufacturing. *Computers and Industrial Engineering* v. 41, p. 37–58, 2001.

KIM, D.W., KIM, K.H., JANG, W., CHEN, F.F. Unrelated parallel machine scheduling with setup times using simulated annealing. *Robotics and Computer-Integrated Manufacturing*, v. 18, p. 223–231, 2002.

KIM, D.W., NA, D.G., CHEN, F.F. Unrelated parallel machine scheduling with setup times and a total weighted tardiness objective. *Robotics and Computer-Integrated Manufacturing*, v. 19, p. 173–181, 2003.

KNÜSEL, L. On the accuracy of statistical distributionsin Microsoft Excel 2003. *Computational Statistics & Data Analysis*, v. 48, n. 3, p. 445-449, 2005.

KROON, L. E VRIJENS, G. Returnable containers: an example of reverse logistics. *International Journal of Physical Distribution & Logistics Management*, v. 25, n. 2, p. 56-68, 1995.

LAI, K. K., LAM, K., CHAN, W. K. Shipping container logistics and allocation. Journal of the *Operational Research Society*, v. 46, n. 6, p. 687-697, 1995.

LAM, S. W., LEE, L. H., TANG, L. C. An approximate dynamic programming approach for the empty container allocation problem. *Transportation Research Part C*, v. 15 n.4, p. 265-277, 2007.

LI, J. A., LEUNG S.C.H., WU Y., LIU K. Allocation of empty containers between multi-ports. *European Journal of Operational Research*, v. 182, n. 1, p. 400-412, 2007.

LIU, S. Q., KOZAN, E. Scheduling trains as a blocking parallel-machine job shop scheduling problem. *Computers & Operations Research*, v. 36, n. 10, p. 2840-2852, 2009.

MCCULLOUGH, B. D., WILSON, B. On the accuracy of statistical procedures in Microsoft Excel 2003. *Computational Statistics & Data Analysis*, v. 49, n. 4, p. 1244-1252, 2005.

POWELL, W. B., CARVALHO, T. A. Dynamic control of logistics queueing networks for large-scale fleet management. *Transportation Science*, v.32, n. 2, p. 90-109, 1998.

SHEN, W. S., KHOONG, C. M. A DSS for empty container distribution planning. *Decision Support Systems*, v. 15, n. 1, p. 75-82, 1995.

SHINTANI, K., IMAI A., NISHIMURA E., PAPADIMITRIOU S. The container shipping network design problem with empty container repositioning. *Transportation Research Part E*, v. 43, n. 1, p. 39-59, 2007.

Steenken, D., Voß, S., Stahlbock, R. Container terminal operation and operations research-a classification and literature review. *OR Spectrum*, v. 26, n. 1, 2004.

TALEB-IBRAHIMI, M., CASTILHO, B., DAGANZO, C. F. Storage space vs handling work in container terminals. *Transportation Research Part B-Methodological*, v. 27, n. 1, p. 13-32, 1993.

WAGNER, H. M. *Principles of operations research*: with applications to managerial decisions. 2 ed. Englewood Cliffs, N.J.: Prentice-Hall, 1975. 1039p.

WHITE, W. W.Dynamic transshipment networks: an algorithm and its application to the distribution of empty containers. *Networks*, v. 2, n. 3, p. 211-236, 1972.

YI, Y., WANG, D.W. Soft computing for scheduling with batch setup times and earliness–tardiness penalties on parallel machines. *Journal of Intelligent Manufacturing*, v. 14,p. 311–322, 2003.

YUN, W. Y., CHOI, Y.S. A simulation model for container-terminal operation analysis using an object-oriented approach. *International Journal of Production Economics*, v. 59, n. 1-3, p. 221-230, 1999.

APÊNDICE A – EXEMPLO DE ARQUIVO COM MODELO PARA O PROBLEMA DE ALOCAÇÃO CONTÊINERES

```
/*********************************************
 * OPL 5.5.1 Model
 * Author: Administrator
 * Creation Date: 11/11/2008 at 10:38 AM
 *********************************************/


//Variables declaration

{string} Loads = ...;
{string} Containers = ...;
int M = ...;

tuple Load {
  string origin;              //Fixed
  string destination;             //Fixed
  int demandTime;        //Fixed
  int transportationTime;    //Fixed
}


Load load[Loads] = ...;
int tt[Loads,Loads] = ...;


//Decision Variables

dvar boolean y[Loads,Loads];
dvar boolean r[Loads,Containers];
dvar float+ t[Loads];


//Objective Fucntion

minimize
   sum(i in Loads)(t[i]);


//Constraints

subject to{

   forall (i in Loads)
      sum(k in Containers) r[i,k] == 1;

   forall (i,j in Loads:i!=j,k in Containers)
```

```
        M*(2 + y[i,j] - r[i,k] - r[j,k]) + (t[i] - t[j]
        - load[j].transportationTime - tt[j,i]) >= 0;

    forall (i,j in Loads:i!=j,k in Containers)
        M*(3 - y[i,j] - r[i,k] - r[j,k]) + (t[j] - t[i]
        - load[i].transportationTime - tt[i,j]) >= 0;

    forall(i in Loads,k in Containers)
        t[i] - load[i].demandTime >= 0;

}
```

APÊNDICE B – EXEMPLO DE ARQUIVO DE DADOS PARA O PROBLEMA DE ALOCAÇÃO DE CONTÊINERES

```
/*********************************************
 * OPL 5.5.1 Data
 * Author: Administrator
 * Creation Date: 11/11/2008 at 10:40 AM
 *********************************************/
/*
SheetConnection sheet("RandomRunsComplete.xls");
Loads from SheetRead(sheet,"example!$A$2:$A$4");
Containers from SheetRead(sheet,"example!$A$33:$B$33");
load from SheetRead(sheet,"case3and2!$B$2:$E$4");
tt from SheetRead(sheet,"case3and2!$O$2:$Q$4");
M = 300;
```

t to SheetWrite(sheet,"case3and2!$F$2:$F$4");

APÊNDICE C - EXEMPLO DE ALTERAÇÃO PARA ESTENDER O ARQUIVO COM MODELO PARA O PROBLEMA DE ALOCAÇÃO DE CONTÊINERES

…

```
  forall(i in Loads,k in Containers)
     t[i] - load[i].demandTime >= 0;

 y["9","1"] == 1;
 y["9","2"] == 1;
 y["9","3"] == 1;
 y["9","4"] == 1;
 y["9","5"] == 1;
 y["9","6"] == 1;
 y["9","8"] == 1;
 y["9","9"] == 0;
 y["9","10"] == 0;
 y["9","11"] == 0;
 y["10","1"] == 1;
 y["10","2"] == 1;
 y["10","3"] == 1;
 y["10","4"] == 1;
 y["10","5"] == 1;
 y["10","6"] == 1;
 y["10","8"] == 1;
 y["10","9"] == 0;
 y["10","10"] == 0;
 y["10","11"] == 0;
 y["11","1"] == 1;
 y["11","2"] == 1;
 y["11","3"] == 1;
 y["11","4"] == 1;
 y["11","5"] == 1;
 y["11","6"] == 1;
 y["11","8"] == 1;
 y["11","9"] == 0;
 y["11","10"] == 0;
 y["11","11"] == 0;
 y["1","9"] == 0;
 y["1","10"] == 0;
 y["1","11"] == 0;
 y["2","9"] == 0;
 y["2","10"] == 0;
 y["2","11"] == 0;
 y["3","9"] == 0;
 y["3","10"] == 0;
 y["3","11"] == 0;
 y["4","9"] == 0;
```

```
  y["4","10"] == 0;
  y["4","11"] == 0;
  y["5","9"] == 0;
  y["5","10"] == 0;
  y["5","11"] == 0;
  y["6","9"] == 0;
  y["6","10"] == 0;
  y["6","11"] == 0;
  y["7","9"] == 0;
  y["7","10"] == 0;
  y["7","11"] == 0;
  y["8","9"] == 0;
  y["8","10"] == 0;
  y["8","11"] == 0;
  t["9"] == 0;
  t["10"] == 0;
  t["11"] == 0;
  r["9","1"] == 1;
  r["9","2"] == 0;
  r["9","3"] == 0;
  r["10","2"] == 1;
  r["10","1"] == 0;
  r["10","3"] == 0;
  r["11","3"] == 1;
  r["11","1"] == 0;
  r["11","2"] == 0;

//Container 1 stars in 1,2 in 2, 3 in 3.

…
```

APÊNDICE D – GERADOR DE CÉNARIOS

```
Option Base 1
Sub FullFill()


    'Variables declaration

    'Const iStartContainer As Integer = 1
    'Const jStartContainer As Integer = 14
    Const iStartLoad As Integer = 2
    Const jStartLoad As Integer = 2
    Const iStartDistance As Integer = 2
    Const jStartDistance As Integer = 10
    Const iStartTt As Integer = 2
    Const jStartTt As Integer = 15
    Const jEndLoad As Integer = 5
    Const jEndDistance As Integer = 12
    Const icontainerIndexStart As Integer = 33 'Where start the array with names for container
index in OPL Studio
    Const jcontainerIndexStart As Integer = 1 'Where start the array with names for container index
in OPL Studio
    Dim maxValueOfLoads As Integer
    Dim valueOfReplication As Integer
    Dim a As Integer
    Dim b As Integer
    Dim c As Integer
    Dim i As Integer
    Dim j As Integer
    Dim r As Integer
    Dim commandString As String
    'Const iInsertCommandStart As Integer = 1 'Row command line
    'Const jInsertCommandStart As Integer = 30 'Column command line

    'Inicialization of the variables

    maxValueOfLoads = 8
    valueOfReplication = InputBox("Enter the number of Replication:", , 30)
    a = 1 'counter of Replication
    b = 2 'counter of Containers (at least 2)
    c = 3 'counter of Loads (at least 3)
    i = 1
    j = 1

    If (maxValueOfLoads >= 2) And (maxValueOfLoads <= 8) Then

       'Deciding about the rate of change

       Do While c <= maxValueOfLoads
          Select Case c
             Case 2
                r = 6
```

```
    Case 3
       r = 6
    Case 4
       r = 7
    Case 5
       r = 8
    Case 6
       r = 9
    Case 7
       r = 10
    Case 8
       r = 11
End Select

Do While b < c
    'Create the sheet

    Sheets.Add(After:=Sheets(Sheets.Count)).Name = "case" & c & "and" & b
    Sheets("case" & c & "and" & b).Activate

    'Begin insert data

    Do While a <= valueOfReplication

        'Distance table

        Cells(iStartDistance + ((a - 1) * r), jStartDistance) = Int((Rnd * 9) + 1)
        Cells(iStartDistance + ((a - 1) * r), jStartDistance + 1) = Int((Rnd * 9) + 1)
        Cells(iStartDistance + 1 + ((a - 1) * r), jStartDistance + 1) = Int((Rnd * 9) + 1)

        If Cells(iStartDistance + ((a - 1) * r), jStartDistance) > _
        Cells(iStartDistance + ((a - 1) * r), jStartDistance + 1) + _
        Cells(iStartDistance + 1 + ((a - 1) * r), jStartDistance + 1) Then
            Cells(iStartDistance + ((a - 1) * r), jStartDistance) = _
            Cells(iStartDistance + ((a - 1) * r), jStartDistance + 1) + _
            Cells(iStartDistance + 1 + ((a - 1) * r), jStartDistance + 1) - 1
        End If
        If Cells(iStartDistance + ((a - 1) * r), jStartDistance + 1) > _
        Cells(iStartDistance + ((a - 1) * r), jStartDistance) + _
        Cells(iStartDistance + 1 + ((a - 1) * r), jStartDistance + 1) Then
            Cells(iStartDistance + ((a - 1) * r), jStartDistance + 1) = _
            Cells(iStartDistance + ((a - 1) * r), jStartDistance) + _
            Cells(iStartDistance + 1 + ((a - 1) * r), jStartDistance + 1) - 1
        End If
        If Cells(iStartDistance + 1 + ((a - 1) * r), jStartDistance + 1) > _
        Cells(iStartDistance + ((a - 1) * r), jStartDistance) + _
        Cells(iStartDistance + ((a - 1) * r), jStartDistance + 1) Then
            Cells(iStartDistance + 1 + ((a - 1) * r), jStartDistance + 1) = _
            Cells(iStartDistance + ((a - 1) * r), jStartDistance) + _
            Cells(iStartDistance + ((a - 1) * r), jStartDistance + 1) - 1
        End If

        Cells(iStartDistance + ((a - 1) * r) + 1, jStartDistance - 1) = _
        Cells(iStartDistance + ((a - 1) * r), jStartDistance)
        Cells(iStartDistance + ((a - 1) * r) + 2, jStartDistance - 1) = _
        Cells(iStartDistance + ((a - 1) * r), jStartDistance + 1)
```

```
Cells(iStartDistance + 2 + ((a - 1) * r), jStartDistance) = _
Cells(iStartDistance + 1 + ((a - 1) * r), jStartDistance + 1)

'Loads schedule

Do While i <= c

   Cells(iStartLoad + (i - 1) + ((a - 1) * r), jStartLoad) = Int((Rnd * 3) + 1)
   Cells(iStartLoad + (i - 1) + ((a - 1) * r), jStartLoad + 1) = Int((Rnd * 2) + 1)
   If Cells(iStartLoad + (i - 1) + ((a - 1) * r), jStartLoad + 1) >= _
   Cells(iStartLoad + (i - 1) + ((a - 1) * r), jStartLoad) Then

      Cells(iStartLoad + (i - 1) + ((a - 1) * r), jStartLoad + 1) = _
      Cells(iStartLoad + (i - 1) + ((a - 1) * r), jStartLoad + 1) + 1

   End If
   Cells(iStartLoad + (i - 1) + ((a - 1) * r), jStartLoad + 2) = Int((Rnd * 9) + 1)

   'Insert distance in the load schedule

   Select Case Cells(iStartLoad + (i - 1) + ((a - 1) * r), jStartLoad) And _
   Cells(iStartLoad + (i - 1) + ((a - 1) * r), jStartLoad + 1)
      Case (1 And 2) Or (2 And 1)
         Cells(iStartLoad + (i - 1) + ((a - 1) * r), jStartLoad + 3) = _
         Cells(iStartDistance + ((a - 1) * r), jStartDistance)
      Case (1 And 3) Or (3 And 1)
         Cells(iStartLoad + (i - 1) + ((a - 1) * r), jStartLoad + 3) = _
         Cells(iStartDistance + ((a - 1) * r), jStartDistance + 1)
      Case (2 And 3) Or (3 And 2)
         Cells(iStartLoad + (i - 1) + ((a - 1) * r), jStartLoad + 3) = _
         Cells(iStartDistance + 1 + ((a - 1) * r), jStartDistance + 1)
   End Select

   i = i + 1

Loop

'Create the command line

commandString = "SheetConnection sheet(""RandomRunsComplete.xls"");" & _
vbLf & "Loads from SheetRead(sheet,""example!" & _
Range(Cells(iStartLoad, jStartLoad - 1), _
Cells(iStartLoad + (i - 2), jStartLoad - 1)).AddressLocal _
& """);" & vbLf & "Containers from SheetRead(sheet,""example!" & _
Sheets(1).Range(Sheets(1).Cells(icontainerIndexStart, jcontainerIndexStart), _
Sheets(1).Cells(icontainerIndexStart, jcontainerIndexStart - 1 + b)).AddressLocal &
""""");" _
& vbLf & "load from SheetRead(sheet,""case" & c & "and" & b & "!" & _
Range(Cells(iStartLoad + ((a - 1) * r), jStartLoad), _
Cells(iStartLoad + (i - 2) + ((a - 1) * r), jStartLoad + 3)).AddressLocal _
& """);" & vbLf & "tt from SheetRead(sheet,""case" & c & "and" & b & "!" & _
Range(Cells(iStartTt + ((a - 1) * r), jStartTt), _
Cells(iStartTt + (c - 1) + ((a - 1) * r), jStartTt + (c - 1))).AddressLocal _
& """);" & vbLf & "M = 300;" & vbLf & "t to SheetWrite(sheet,""case" & c & "and" & b &
"!" & _
Range(Cells(iStartLoad + ((a - 1) * r), jStartLoad + 4), _
```

```
                    Cells(iStartLoad + (i - 2) + ((a - 1) * r), jStartLoad + 4)).AddressLocal _
                    & """");"

                    'Create the .dat file

                    MyFile = "C:\Documents and Settings\efrem\My
Documents\UFRGS\Dissertacao\OPL\RandomRunsComplete\" _
                    & "RandomRunsComplete" & c & "and" & b & "and" & a & ".dat"
                    fnum = FreeFile()
                    Open MyFile For Output As fnum
                    Print #fnum, commandString
                    Close #fnum


                    'Restart i for the next loop
                    i = 1

                    'Insert distance em tt

                    Do While i <= c

                       Do While j <= c

                          If i = j Then
                             Cells(iStartTt + (i - 1) + ((a - 1) * r), jStartTt + (j - 1)) = 0
                          Else
                             'Cells(iStartTt + (i - 1) + ((a - 1) * r), jStartTt + (j - 1)) = _
                             'Cells(iStartLoad + (i - 1) + ((a - 1) * r), jStartLoad + 1) & "and" & _
                             'Cells(iStartLoad + ((a - 1) * r) + (j - 1), jStartLoad)

                             Select Case (Cells(iStartLoad + (i - 1) + ((a - 1) * r), jStartLoad + 1) And _
                             Cells(iStartLoad + ((a - 1) * r) + (j - 1), jStartLoad))
                                Case (1 And 2) Or (2 And 1)
                                   Cells(iStartTt + (i - 1) + ((a - 1) * r), jStartTt + (j - 1)) = _
                                   Cells(iStartDistance + ((a - 1) * r), jStartDistance)
                                Case (1 And 3) Or (3 And 1)
                                   Cells(iStartTt + (i - 1) + ((a - 1) * r), jStartTt + (j - 1)) = _
                                   Cells(iStartDistance + ((a - 1) * r), jStartDistance + 1)
                                Case (2 And 3) Or (3 And 2)
                                   Cells(iStartTt + (i - 1) + ((a - 1) * r), jStartTt + (j - 1)) = _
                                   Cells(iStartDistance + 1 + ((a - 1) * r), jStartDistance + 1)
                                Case (1 And 1) Or (2 And 2) Or (3 And 3)
                                   Cells(iStartTt + (i - 1) + ((a - 1) * r), jStartTt + (j - 1)) = 0
                             End Select

                          End If

                          j = j + 1

                       Loop

                       j = 1
                       i = i + 1

                    Loop
```

```
            i = 1
            a = a + 1

        Loop
        a = 1
        b = b + 1
    Loop
    b = 2
    c = c + 1
  Loop

 Else
   MsgBox "Value out of Range, type a number >= 2 and <= 8"
 End If

End Sub
```

APÊNDICE E – HEURÍSTICA

```
Option Base 1
Sub Heuristic1()

Const numberOfLoads As Integer = 8 'Testing only for 8 loads, if this isn't constant the vector load should be ReDim
Const iStartLoad As Integer = 2
Const jStartLoad As Integer = 3
Const iStartContainer As Integer = 1
Const jStartContainer As Integer = 14
Const jtimeColumnStart As Integer = 41
Const iStartDistance As Integer = 2
Const jStartDistance As Integer = 10
Const rate As Integer = 11 'The rate for 8 loads
Const origin As Integer = 1
Const destination As Integer = 2
Const demandInstant As Integer = 3
Const transportTime As Integer = 4
Const numberOfContainer As Integer = 7 'j or column
Const numberOfClients As Integer = 3

'variables
Dim numberOfReplication As Integer
Dim numberOfContainers As Integer 'Cell "pool"
Dim NumberOfIterations As Integer 'Defined in fullfillNumberOfIterations()
Dim load(numberOfLoads, 4) As Integer 'Tuple of the load (1 - Origin, 2 - Destination, 3 - Demand Instant, 4 - Transport Time)
Dim containerLocation() As Integer 'This is dynamic because they change each instance
Dim containerClock() As Integer 'Same as above
Dim distance(3, 3) As Integer 'Distance table (for empty transport)
Dim iterationCount As Integer
Dim smallestDistance(numberOfClients) As Integer 'For 3 clients
Dim a As Integer 'Count of Replication
Dim i As Integer
Dim i2 As Integer
Dim i3 As Integer
Dim h As Integer
Dim testIfNoLoad() As Boolean
Dim containerBusy() As Boolean
Dim loadTransported(numberOfLoads) As Boolean
Dim timeCount As Double

'Inicialization of the variables
a = 1
i = 1
i2 = 1
i3 = 1
numberOfReplication = InputBox("Enter the number of Replication:", , 30)
iterationCount = 1
timeCount = 0
```

```
        'Activate the correct sheet
        Sheets(numberOfLoads).Activate 'Activate test8

        'Begin Heuristic
        Do While a <= numberOfReplication
           timeCount = Timer
           'Amount of containers
           numberOfContainers = Cells(iStartContainer + ((a - 1) * rate), jStartContainer)
           If ((numberOfContainers = 1) Or (numberOfContainers = numberOfLoads)) Then
              'test = ThisWorkbook.RunCPLEX(a, numberOfContainers, numberOfLoads)
              'MsgBox "The replication " & a & " should be calculated directly in CPLEX"
           Else
              NumberOfIterations = ThisWorkbook.NumberOfIterations(numberOfLoads,
numberOfContainers)
              ReDim containerLocation(numberOfContainers)
              ReDim containerClock(numberOfContainers)
              ReDim testIfNoLoad(numberOfContainers)
              ReDim containerBusy(numberOfContainers)

              'Distance table
              distance(1, 2) = Cells(iStartDistance + ((a - 1) * rate), jStartDistance)
              distance(2, 1) = Cells(iStartDistance + ((a - 1) * rate), jStartDistance)
              distance(1, 3) = Cells(iStartDistance + ((a - 1) * rate), jStartDistance + 1)
              distance(3, 1) = Cells(iStartDistance + ((a - 1) * rate), jStartDistance + 1)
              distance(2, 3) = Cells(iStartDistance + 1 + ((a - 1) * rate), jStartDistance + 1)
              distance(3, 2) = Cells(iStartDistance + 1 + ((a - 1) * rate), jStartDistance + 1)
              'MsgBox "Distancia 1, 2: " & distance(1, 2) & " Distancia 1, 3: " & distance(1, 3) & " Distancia
3, 2: " & distance(3, 2)
              For h = 1 To 3
                 Select Case h
                    Case 1
                    smallestDistance(h) = 2
                    If distance(1, 3) < distance(1, 2) Then
                       smallestDistance(h) = 3
                    End If

                    Case 2
                    smallestDistance(h) = 1
                    If distance(2, 3) < distance(2, 1) Then
                       smallestDistance(h) = 3
                    End If

                    Case 3
                    smallestDistance(h) = 1
                    If distance(3, 2) < distance(3, 1) Then
                       smallestDistance(h) = 2
                    End If
                 End Select
              Next h
              h = 1

              'load schedule (first sort ...
              Range(Cells(iStartLoad + ((a - 1) * rate), jStartLoad - 1), _
              Cells(iStartLoad + numberOfLoads - 1 + ((a - 1) * rate), jStartLoad + 3)).Sort _
              Key1:=Cells(iStartLoad + ((a - 1) * rate), (jStartLoad + 2))
```

```
            '...and then fullfill and clean loadTransported
            Do While i <= numberOfLoads
               'first fulfill
               load(i, 1) = Cells(iStartLoad + (i - 1) + ((a - 1) * rate), jStartLoad)
               load(i, 2) = Cells(iStartLoad + (i - 1) + ((a - 1) * rate), jStartLoad + 1)
               load(i, 3) = Cells(iStartLoad + (i - 1) + ((a - 1) * rate), jStartLoad + 2)
               load(i, 4) = Cells(iStartLoad + (i - 1) + ((a - 1) * rate), jStartLoad + 3)

               'and Clean loadTransported
               loadTransported(i) = False
               i = i + 1
            Loop
            i = 1

            'Start algorithm
            For h = 1 To numberOfContainers 'Left like this just for test (should be automated)
               containerLocation(h) = 1 'InputBox("Enter where (number of the Client) starts to container "
& h & " : ")
               containerClock(h) = 0 'Clean clock
            Next h
            h = 1

            Do While iterationCount <= NumberOfIterations
      'MsgBox "i = " & i
               For h = 1 To numberOfContainers
                  testIfNoLoad(h) = True
                  containerBusy(h) = False
               Next h
               h = 1
               For h = 1 To numberOfContainers
                  If (i <= numberOfLoads) Or (i2 <= numberOfLoads) Or (i3 <= numberOfLoads) Then
      'MsgBox "container = " & h
                     'i2 = i
                     'i3 = i
                     If (i <= numberOfLoads) Then
                        Do While i <= iterationCount * numberOfContainers
                           If (containerLocation(h) = load(i, origin) And Not containerBusy(h) _
                           And Not loadTransported(i)) Then
                              testIfNoLoad(h) = False
                              containerBusy(h) = True
                              loadTransported(i) = True
                              If containerClock(h) < Cells(iStartLoad + (i - 1) + ((a - 1) * rate), jStartLoad
+ 2) Then
                                 Cells(iStartLoad + (i - 1) + ((a - 1) * rate), jtimeColumnStart) = _
                                 load(i, demandInstant)
                                 containerClock(h) = containerClock(h) + load(i, transportTime) + _
                                 load(i, demandInstant)
                              Else
                                 Cells(iStartLoad + (i - 1) + ((a - 1) * rate), jtimeColumnStart) =
containerClock(h)
                                 containerClock(h) = containerClock(h) + load(i, transportTime)
                              End If
                              Cells(iStartLoad + (i - 1) + ((a - 1) * rate), numberOfContainer) = h
                              containerLocation(h) = load(i, destination)
                              i = i + 1
                              Exit Do
```

```
                End If
                i = i + 1
                If i > numberOfLoads Then
                    Exit Do
                End If
            Loop
        End If
        'i = 1

        If testIfNoLoad(h) And i2 <= numberOfLoads Then 'one time
            Do While i2 <= iterationCount * numberOfContainers
                If ((smallestDistance(containerLocation(h)) = load(i2, origin)) And Not
containerBusy(h) _
                And Not loadTransported(i2)) Then
                    testIfNoLoad(h) = False
                    containerBusy(h) = True
                    loadTransported(i2) = True
                    'Cells(iStartLoad + (i2 - 1) + ((a - 1) * rate), (jStartLoad + 5)) = h
                    If containerClock(h) < Cells(iStartLoad + (i2 - 1) + ((a - 1) * rate), jStartLoad
+ 2) Then
                        Cells(iStartLoad + (i2 - 1) + ((a - 1) * rate), jtimeColumnStart) = _
                        load(i2, demandInstant) + distance(containerLocation(h), load(i2, origin))
                        containerClock(h) = containerClock(h) + load(i2, transportTime) + _
                        load(i2, demandInstant) + distance(containerLocation(h), load(i2, origin))
                    Else
                        Cells(iStartLoad + (i2 - 1) + ((a - 1) * rate), jtimeColumnStart) = _
                        containerClock(h) + distance(containerLocation(h), load(i2, origin))
                        containerClock(h) = containerClock(h) + load(i2, transportTime) + _
                        distance(containerLocation(h), load(i2, origin))
                    End If
                    Cells(iStartLoad + (i2 - 1) + ((a - 1) * rate), numberOfContainer) = h
                    containerLocation(h) = load(i2, destination)
                    i2 = i2 + 1
                    Exit Do
                End If
                i2 = i2 + 1
                If i2 > numberOfLoads Then
                    Exit Do
                End If
            Loop
            'i = 1
        End If
        If testIfNoLoad(h) And i3 <= numberOfLoads Then 'second time
            Do While i3 <= iterationCount * numberOfContainers
                If (Not containerBusy(h) And (Not loadTransported(i3))) Then
                    testIfNoLoad(h) = False 'Could be deleted when work
                    containerBusy(h) = True
                    loadTransported(i3) = True
                    'Cells(iStartLoad + (i3 - 1) + ((a - 1) * rate), (jStartLoad + 5)) = h
                    If containerClock(h) < Cells(iStartLoad + (i3 - 1) + ((a - 1) * rate), jStartLoad
+ 2) Then
                        Cells(iStartLoad + (i3 - 1) + ((a - 1) * rate), jtimeColumnStart) = _
                        load(i3, demandInstant) + distance(containerLocation(h), load(i3, origin))
                        containerClock(h) = containerClock(h) + load(i3, transportTime) + _
                        load(i3, demandInstant) + distance(containerLocation(h), load(i3, origin))
                    Else
```

```
                        Cells(iStartLoad + (i3 - 1) + ((a - 1) * rate), jtimeColumnStart) = _
                        containerClock(h) + distance(containerLocation(h), load(i3, origin))
                        containerClock(h) = containerClock(h) + load(i3, transportTime) + _
                        distance(containerLocation(h), load(i3, origin))
                    End If
                    Cells(iStartLoad + (i3 - 1) + ((a - 1) * rate), numberOfContainer) = h
                    containerLocation(h) = load(i3, destination)
                    i3 = i3 + 1
                    Exit Do
                End If
                i3 = i3 + 1
                If i3 > numberOfLoads Then
                    Exit Do
                End If
            Loop
            'i = iRestore
        End If
        'If testIfNoLoad(h) Then 'Could be deleted when work!
            'MsgBox "Erro!!!!!"
        'End If
    End If
Next h

h = 1
iterationCount = iterationCount + 1
  Loop
  iterationCount = 1
End If
Cells(4 + ((a - 1) * rate), jStartContainer + 1) = Timer - timeCount
a = a + 1
i = 1
i2 = 1
i3 = 1
Loop
a = 1

End Sub
```