RODRIGO NOGUEIRA CARDOSO

CONTRIBUIÇÃO AO ESTUDO DE PLANEJAMENTO AUTOMÁTICO APLICADO A SISTEMA DE MOVIMENTAÇÃO DE MATERIAIS

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

FACULDADE DE ENGENHARIA MECÂNICA

2014

RODRIGO NOGUEIRA CARDOSO

CONTRIBUIÇÃO AO ESTUDO DE PLANEJAMENTO AUTOMÁTICO APLICADO A SISTEMA DE MOVIMENTAÇÃO DE MATERIAIS

Dissertação apresentada ao Programa de Pós-graduação em Engenharia Mecânica da Universidade Federal de Uberlândia, como parte dos requisitos para obtenção do título de **MESTRE EM ENGENHARIA MECÂNICA.**

Área de concentração: Mecânica dos Sólidos e Vibrações

Orientador: Prof. Dr. José Jean-Paul Zanlucchi de Souza Tavares

UBERLÂNDIA - MG
2014

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU , MG, Brasil

C268c
2014

Cardoso, Rodrigo Nogueira, 1989-
Contribuição ao estudo de planejamento automático aplicado a sistema de movimentação de materiais / Rodrigo Nogueira Cardoso. - 2014.
244 f. : il.

Orientador: José Jean-Paul Zanlucchi de Souza Tavares.
Dissertação (mestrado) – Universidade Federal de Uberlândia, Programa de Pós-Graduação em Engenharia Mecânica.
Inclui bibliografia.

1. Engenharia mecânica - Teses. 2. Containers - Teses. 3. Sistemas de unitização de cargas - Teses. 4. Automação industrial - Teses. I. Tavares, José Jean-Paul Zanlucchi de Souza. II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Engenharia Mecânica. III. Título.

CDU: 621

DEDICATÓRIA

Eu dedico esta dissertação a todos que me auxiliaram nessa jornada e me apoiaram neste empreendimento.

AGRADECIMENTOS

Eu agradeço primeiramente a Deus, pela saúde, capacidade e a oportunidade de estar desenvolvendo estudos que possam vir a contribuir no avanço da ciência e qualidade de vida da humanidade.

Eu agradeço a minha família, que apesar das desavenças, sempre me apoiou e me estimulou a alcançar os meus sonhos e dar o melhor de mim.

A minha namorada que durante todo esse período do mestrado foi invariavelmente atenciosa e confortadora apesar de todas as frustrações decorrentes do trabalho de pesquisa e também sempre me apoiou nas minhas decisões.

A todos os meus colegas/companheiros/amigos do MAPL cuja boa vontade de dividir os seus conhecimentos e auxiliar um amigo nos momentos de maiores dificuldades, foi de fundamental importância para que eu chegasse aonde cheguei.

Ao meu orientador José Jean-Paul que sempre dividiu as dificuldades no desenvolvimento deste trabalho e me orientou das mais variadas possibilidades de solução.

A Universidade Federal de Uberlândia e á Faculdade de Engenharia Mecânica pelo espaço físico e os recursos necessários.

Ao CNPQ pelo auxílio financeiro.

Cardoso, R. N. Sistema Automático para Movimentação de Materiais com Planejamento Automático e Algoritmo Simplex.

Resumo

No transporte de cargas, o problema de carregamento de contêineres tem tido muita repercussão nos estudos logísticos. Pode se provar que se utilizando esse dispositivo existe uma redução de perdas, roubos, danos aos produtos e custos gerais. Os problemas de carregamento são tipicamente problemas de otimização cujo objetivo é a maximização do volume utilizado ou minimização de parâmetros de custo. Por outro lado, o planejamento automático, uma subárea de estudo da inteligência artificial, pretende auxiliar nos processos industriais através de uma especificação sistemática de um conjunto de ações que permitem se atingir um objetivo predefinido. Neste paradigma a planta da fábrica pode ser modelada em se utilizando uma série de diagramas descritos através do padrão UML, para a obtenção do domínio do problema e os estados finais e iniciais. Vale ressaltar que no processo de especificação das ações podem ser inseridas métricas para propósitos de minimização, porém de acordo com a arquitetura de cada planejador automático a inserção de métrica se torna um processo extremamente restrito. Em uma tentativa de se unir ambas as áreas do conhecimento, este trabalho faz uso de um das técnicas mais utilizadas na literatura de carregamento de contêineres, que trata-se da programação linear para a modelagem e o método simplex para resolução do modelo, de tal forma a especificar o estado final desejado a ser utilizado no processo de resolução via planejadores automáticos. A proposta principal desta integração consiste da criação de um sistema totalmente automatizado capaz de realizar todo o processo de carregamento através das informações referentes aos produtos, à planta industrial e aos dispositivos de transporte.

Palavras Chave: Problema de Carregamento de Contêineres, Programação Linear, Algoritmo Simplex, Planejamento Automático.

Cardoso, R. N. Automatic System for the Transporting of Material with Automated Planning and Simplex Algorithm.

Abstract

In the transporting of loads, the container loading problems have had much repercussion among logistical studies. It can be proved that through using such device there is a reduction on losses, theft, damaging of goods and general costs. The loading problems are typical optimization problems whose objective is maximizing the used volume or minimize the costs parameters. On the other hand, automated planning research, a sub area of artificial intelligence, intends to assist the industrial processes for systematically specifying a set of actions that allows achieving a predefined goal. In this paradigm the industrial plant is modeled with a series of diagrams respecting the UML standard, for obtaining the problem domain and the initial and final states. It is noteworthy that in the actions specifying process a metric can be added for minimizing purposes, but according to each automated planners characteristics the insertion of such metrics is relatively bounded. In an attempt to accrete both areas of knowledge, it is intended to use one of the widely studied techniques available in the loading problem literature, in particular the linear programming for modeling and the Simplex method for solving, in a way to obtain a specific final state to be utilized in the automated planning solving process. The main proposal of this integration is to create a fully automated system, capable of carrying out the entire loading process through the information referred to the products, to the industrial plant and the transporting devices.

Keywords: Container Loading Problem, Linear Programming, Simplex Algorithm, Automated Planning.

LISTA DE FIGURAS

LISTA DE TABELAS

Sumário

CAPÍTULO I

INTRODUÇÃO

Atualmente a automação tem se tornado imprescindível para manter a competitividade dos processos produtivos. Já está consolidada a automação tanto em atividades diretas da produção, como é o caso de máquinas ferramentas automatizadas, por exemplo, tornos e fresas CNC (Controle Numérico Computadorizado), robôs; bem como atividades indiretas de produção, como a movimentação automática de produtos por meio de AGVs (*Automated Guided Vehicle* ou Veículo Guiado Automaticamente). Há ainda outros equipamentos que não passaram por esse processo de automação, sendo esse o caso da movimentação de cargas e processos logísticos para equipamentos como pontes rolantes, empilhadeiras, caminhões, pórticos e outros.

No ambiente empresarial a logística tem um importante papel, que é o da criação de valor - valor para os clientes e fornecedores da empresa, e valor para todos aqueles que têm nela interesses diretos. O valor da logística é manifestado primariamente em termos de tempo e lugar. Produtos e serviços não tem valor a menos que estejam em poder dos clientes quando(tempo) e onde(lugar) eles pretendem consumi-los (BALLOU, 2006). A logística existe para satisfazer às necessidades do cliente, facilitando as operações relevantes de produção e marketing. Do ponto de vista estratégico, os executivos de logística procuram atingir uma qualidade predefinida de serviço ao cliente por meio de uma competência operacional que represente o estado da arte (BOWERSOX; CLOSS, 2007).

Há de se considerar que nos processos e atividades existentes no mecanismo que liga os clientes aos seus fornecedores existam custos associados, que são denominados de custos logísticos. De acordo com informações do Fundo Monetário Internacional, os custos logísticos representam em média 12% do produto interno bruto (PIB) mundial. Existem várias atividades que se classificam nesta categoria logística e cada uma delas possui certo custo de implementação e manutenção por parte das empresas. Dentre esses custos podem-se destacar os principais: custos relativos ao transporte de cargas e produtos; custos relativos à armazenagem, custos relativos aos serviços prestados aos

clientes e processamento de pedidos; custo de administração dos processos; e por último custos de manutenção de estoques.

Nesse ínterim o transporte normalmente representa a atividade que mais requer dispêndio dos recursos destinados aos processos logísticos. Segundo Ballou (2006) a movimentação de cargas absorve de um a dois terços dos custos logísticos totais.

Em resposta a redução dos custos inerentes a movimentação de cargas foi introduzido um conceito no meio logístico, que é o da unitização de cargas. A unitização de carga consiste no carregamento de volumes unitários da carga sobre paletes ou dentro de contêineres de maneira que a configuração final se conforme exatamente com dimensões especificadas de comprimento, largura e altura. Estas dimensões são geralmente selecionadas para maximizar o espaço utilizado nos veículos selecionados para o transporte dos produtos. Segundo Junqueira (2009) a unitização de cargas proporciona as seguintes vantagens: redução de perdas, roubos e avarias de carga; redução de despesas com rotulagem e marcação das cargas; redução de despesas com utilização de mão de obra para movimentação da carga; aumento da rapidez nas operações de carregamento e descarregamento de veículos e embarcações, etc.

Para implementação da unitização existem alguns dispositivos que são atualmente utilizados, dentre estes pode-se citar as cintas marinhas, *big bags*, paletes, contêineres e ULDs (*Unit Load Device* ou Dispositivo de Carregamento Unitário). Nesse contexto, para que se tenha o melhor aproveitamento do dispositivo de unitização utilizado, normalmente são elaboradas estratégias para a alocação dos produtos nesses dispositivos. Em particular o Problema de Carregamento de Contêineres (PCC) tem tido grande repercussão e várias técnicas têm sido formuladas para a resolução deste problema. Dentre estas técnicas pode-se citar: a busca em árvore, busca tabu, algoritmos para construção de paredes e camadas, programação linear, dentre outras. A programação linear em particular tem tido sua relevância no fato de que através de uma modelagem matemática pode-se expressar de maneira concisa as diferentes restrições que devem ser consideradas no PCC.

Por sua vez a automação industrial tem o mesmo propósito na viabilização dos processos no seguimento de produção. As empresas realizam projetos de automação da produção e de manufatura integrada por computador por diferentes motivos. Algumas das razões mais comuns para justificar a automação são as seguintes: aumentar a produtividade; reduzir custos do trabalho; minimizar o efeito da falta de trabalhadores; reduzir ou eliminar as rotinas manuais e das tarefas administrativas; aumentar a

segurança do trabalhador; melhorar a qualidade do produto; diminuir o tempo de produção; realizar processos que não podem ser executados manualmente; evitar o alto custo da não automação. (GROOVER, 2011)

Dentre as atividades no processo de automatização, está o planejamento de processos. O planejamento de processos é a base para otimização de todo o cenário e suas alternativas, e não somente para simples operações (HALEVI; WEILL, 1995). A abordagem para planejamento de processos está fundamentada na utilização dos CAPPs (*Computer Aided Process Planning* ou Planejamento de Processo Auxiliado por Computador), que se tratam de softwares nos quais são inseridas informações sobre a planta e os processos que devem ser realizados, além dos algoritmos para a realização de tais processos. Os CAPPs podem ser divididos em quatro categorias dependendo da forma de interação com o usuário. São eles os CAPPs de planejamento variante,CAPPs de planejamento generativo interativo, CAPPs de planejamento generativo automático e os CAPPs híbridos. Vale ressaltar que o estudo e desenvolvimento dos CAPPs estão voltados em particular para processos de manufatura (em geral de usinagem) na fabricação e montagem de peças. O planejamento automático faz um paralelo a esta tecnologia, visando de maneira generalista à obtenção de planos para a execução de quaisquer tipos de tarefas.

1.1. Objetivo

Este trabalho tem por objetivo estudar a automação do processo de carregamento de mercadorias, utilizando-se técnicas de planejamento automático juntamente com uma modelagem em programação linear. Através desse conjunto, pretende-se dotar o sistema de carregamento de certa autonomia para que o mesmo seja capaz de desempenhar o processo de carregamento com um mínimo de interferência humana.

A proposta deste trabalho compreende a integração de técnicas de planejamento generativo automático(daqui em diante será chamado somente de planejamento automático) com as técnicas de resolução para o problema de carregamento de contêiner, os quais serão detalhados no decorrer deste relatório.

Os objetivos secundários deste trabalho correspondem ao estudo de diversos modelos de programação linear, na tentativa de representar de forma satisfatória as

características desejadas quando da montagem de um contêiner no despacho de mercadorias. Objetiva-se também, estudar a linguagem utilizada no modelo de planejamento automático para obtenção da sequência de ações do processo de empacotamento de mercadorias.

1.2. Justificativa

A automação industrial tem por premissa básica melhorar as condições de trabalho e a produtividade no seu meio de implementação. Dessa forma, pretende-se utilizar esta tecnologia na tentativa de aprimorar uma das tarefas que é ainda realizada de forma manual ou semi-automatizadas por muitas empresas do seguimento industrial, que é o carregamento de cargas. A automação, conforme mencionado anteriormente, tem potencial para o melhoramento de diversas características do processo de carregamento de produtos. Assim pretende-se utilizar um novo segmento dessa tecnologia, que se trata do planejamento automático, para que se obtenha uma solução sistemática e flexível para este processo.

Esse trabalho se justifica por utilizar ferramentas já desenvolvidas para problemas específicos de logística, no caso PCC acrescentando a possibilidade de automaticamente executar o carregamento por meio de equipamentos integrados ao plano solução, principalmente quando lidamos com domínios de planejamento.

Os capítulos deste trabalho estão divididos da seguinte forma: o capítulo 2 mostra toda a fundamentação teórica que foi utilizada no projeto; o capítulo 3 expõe a proposta do trabalho que foi desenvolvido; o capítulo 4 apresenta todos os resultados obtidos e os respectivos comentários; o capítulo 5 mostra as conclusões a respeito do desenvolvimento do trabalho.

CAPÍTULO II

FUNDAMENTAÇÃO TEÓRICA E REVISÃO DA LITERATURA

2.1. Logística Empresarial

Desde as épocas mais antigas, é sabido que as mercadorias necessárias não eram fabricadas perto dos seus locais de consumo. A fabricação das mercadorias era feitas em regiões esparsas, sendo acessíveis apenas em determinadas condições. Normalmente os produtos ou eram consumidos nos locais de fabricação ou eram estocados para o consumo posterior. Devido a essa limitada capacidade de transporte as pessoas eram obrigadas a viveram perto das regiões de produção.

Ainda hoje existem regiões que o acesso às mercadorias ou *comodities* é dificultado. Mas no geral, em razão do desenvolvimento das capacidades de transporte, estocagem e troca de informações, o conceito de logística logo surgiu para superar essas diferenças de tempo e espaço de produção e consumo. Dessa forma, o excedente de produção teve vazão para outras regiões com a possibilidade da obtenção de lucro. Assim, as vantagens geográficas de cada uma das regiões do globo puderam ser melhor e mais eficazmente utilizadas.

De acordo com o *Council of Logistics and Management* (CLM) a logística empresarial pode ser definida como "O processo de planejamento, implantação e controle do fluxo eficiente e eficaz de mercadorias, serviços e das informações relativas desde o ponto de origem até o ponto de consumo com o propósito de atender as exigências dos clientes".

No contexto empresarial o conceito da logística muitas das vezes vem atrelado ao conceito do gerenciamento da cadeia de suprimento, portanto é relevante diferencia-los para um melhor entendimento do assunto. Segundo algumas abordagens, a logística empresarial pode ser enxergada como um processo, sendo que este faz parte de um âmbito mais global que é a cadeia de suprimento. Mas ainda assim não existe uma definição precisa de quais são as atividades logísticas, uma vez que esta é variável em função dos ramos e nichos industriais e dos encarregados do gerenciamento industrial.

A cadeia de suprimento normalmente trata do processo como um todo, agregando as informações desde a obtenção da matéria prima até a venda do produto final, uso, consumo, remanufatura, reparo e descarte dos itens, sendo que este processo pode ser executado por uma ou várias empresas. Desse conceito decorre que o gerenciamento da cadeia de suprimentos implica na troca de serviços, produtos e informações entre todas as etapas envolvidas no processo. Porém, na prática o que ocorre é que o gerenciamento da cadeia de suprimentos limita-se apenas as partes "controláveis" do processo por parte da empresa e é onde os conceitos de logística e cadeia de suprimentos se confundem.

Na tentativa de se obter uma clara descrição das atividades logísticas, normalmente adota-se um conjunto básico de procedimentos. Novamente de acordo com o CLM: "Os componentes de um sistema logístico típico são: serviços ao cliente, previsão de demanda, comunicações de distribuição, controle de estoque, manuseio de materiais, processamento de pedidos, peças de reposição e serviços de suporte, escolha de locais para fábrica e armazenagem (análise de localização), embalagem, manuseio de produtos devolvidos, reciclagem de sucata, tráfego e transporte, e armazenagem e estocagem.

As atividades descritas no conceito anterior podem ser visualizadas na Fig. 2.1

Figura 2.1 – Atividades da logística (adaptado de Ballou (2006)).

As atividades listadas anteriormente podem ser divididas em duas categorias: as atividades principais e as atividades de suporte. As atividades principais são: serviços de marketing, transporte de cargas, gerenciamento de estoques e fluxo de informações e

processamento de dados. As atividades de suporte normalmente são aquelas vinculadas à parte inicial de implantação dos sistemas de transporte, de estocagem, canal de compras, embalagens, sistema de coleta de informações, dentre outras. Portanto no dia a dia das atividades logísticas costumeiramente se opera com as atividades principais uma vez que estas são responsáveis por manter o fluxo de produtos, serviços e informações.

Nesse contexto o transporte e a manutenção dos estoques são atividades logísticas primárias na absorção de custos. A experiência demonstra que cada um deles representará entre metade e dois terços dos custos logísticos totais (BALLOU, 2006).

2.1.1. Custos operacionais

De acordo com o fundo monetário internacional os custos logísticos representam em média 12% do produto interno bruto mundial. Para as empresas, esses custos podem absorver um montante de 4 a 30% do valor das vendas. Ainda assim, apesar do dispêndio de recursos às atividades logísticas, a globalização do comercio surgiu como resposta natural à redução dos custos e expansão dos mercados. Uma prova desses benefícios são justamente os acordos comerciais existentes entre as nações (NAFTA, União Europeia, MERCOSUL, dentre outros) para redução de custos de importação e exportação.

Apesar da porcentagem considerável de recursos destinada aos procedimentos logísticos, essas atividades são essenciais, sendo que são responsáveis pela criação de dois tipos de valores aos clientes: valores de espaço e tempo, que são associados respectivamente, ao transporte e estocagem. Dessa forma a logística não deve ser apenas considerada como uma técnica para redução de custos, mas também como uma estratégia a ser elaborada para contribuição no crescimento e expansão do mercado empresarial.

2.1.2. Estratégia logística

Para definição de uma forma de atuação é necessário inicialmente que a empresa defina sua visão de trabalho, ou seja, suas metas para com o mercado. A logística de certa forma é responsável por tornar as metas empresariais factíveis, uma vez que

trabalha em cima de quatro fatores primordiais que são: clientes, fornecedores, concorrentes e a empresa propriamente dita.

É opinião mais ou menos unânime que uma estratégia logística inclui três objetivos principais: redução de custos, redução de capital e melhoria dos serviços (BALLOU, 2006). Para redução de custos normalmente trabalha-se na redução das despesas relativas ao transporte e armazenamento de mercadorias. Nesse sentido devem-se estudar as possibilidades existentes quanto a modais de transporte e locais de armazenagem. A redução de capital implica na redução de investimentos em processos logísticos. Desta premissa entende-se obter um melhor aproveitamento dos recursos já destinados a essa atividade. A melhoria dos serviços implica que os lucros dependem do nível dos serviços logísticos proporcionados. Logicamente que a melhoria nos serviços implica em aumento dos custos, porém resulta que os lucros podem ser maximizados devidos a essa qualidade aumentada.

Nesse sentido, conforme já mencionado, o transporte de mercadorias é um importante fator a ser considerado na elaboração de um plano estratégico. Vale ressaltar que invariavelmente os planos ou abordagens empresariais dependem de alguma forma de um meio de transporte seja para o deslocamento de mercadorias ou de suprimentos para a prestação de serviços. Daí decorre que para uma boa especificação e funcionamento do canal de transporte, um estudo extremamente meticuloso deve ser realizado em torno das possibilidades existentes quanto aos modais e suas características.

2.1.3. Transporte de Cargas

O transporte de cargas pode ser conceituado como a atividade de circulação de mercadorias, de um ponto a outro de um determinado território, sendo este, nacional ou internacional (LUIZ, 2007). Segundo Bowersox e Closs (2007), a funcionalidade do transporte corresponde a dois papeis principais: a movimentação e armazenagem de produtos. O transporte é necessário para movimentar produtos até a fase seguinte do processo de fabricação ou até um local fisicamente mais próximo ao cliente final, estejam os produtos na forma de materiais, componente, subconjuntos, produtos semi acabados ou produtos acabados. Uma função menos comum do transporte é a estocagem temporária.

Segundo Bowersox e Closs (2007) há dois princípios fundamentais que norteiam as operações e o gerenciamento do transporte: a economia de escala e a economia de distância. A economia de escala é a economia obtida com a diminuição do custo de transporte por unidade de peso com cargas maiores. Cargas fechadas (CF) (isto é, cargas que utilizam toda a capacidade do veículo), por exemplo, têm um custo menor por unidade de peso que cargas fracionadas (CFr) (isto é, cargas que utilizam parte da capacidade do veículo). Um pormenor importante para um bom gerenciamento é o cuidado de consolidar pequenas cargas em cargas maiores, a fim de obter vantagem da economia de escala. A economia de distância tem como característica a diminuição do custo de transporte por unidade de distância à medida que a distância aumenta.

Na tentativa de se obter um melhor aproveitamento da capacidade de transporte no que tange ao embarque de mercadorias, o conceito de unitização de cargas foi introduzido no meio logístico. Unitizar uma carga refere-se ao conceito de agrupar vários volumes pequenos ou grandes em um maior, ou mesmo em um único volume. O referido agrupamento tem por objetivo facilitar o manuseio, armazenagem e transporte da carga. Com a aplicação desse conceito, o total de volumes envolvidos em cada unitização passa a ser um volume unificado (LUIZ, 2007).

Segundo Luiz (2007) a unitização de cargas resulta nas seguintes vantagens:

a) Minimização do custo hora/homem:

- menor utilização de mão de obra;

- redução de acidentes pessoais.

b) Ganhos significativos em estocagem e armazenagem:

- racionalização do espaço de armazenagem, com melhor aproveitamento vertical da área de estocagem;

- agilidade na estocagem;

- diminuição das operações de movimentação;

- redução da quantidade de volumes a manipular e menor número de manuseios;

- economia de até 50% no custo da movimentação;

- diminuição de avarias e roubos de mercadorias;

- melhor aproveitamento dos equipamentos de movimentação;

- possibilidade do uso de mecanização;

- redução de custo com embalagens.

c) Ganhos nos fatores de exigências dos clientes, como:

- melhoria no tempo de operação de embarque e desembarque;

- diminuição de danos aos produtos;

- redução do tempo de rotulagem;

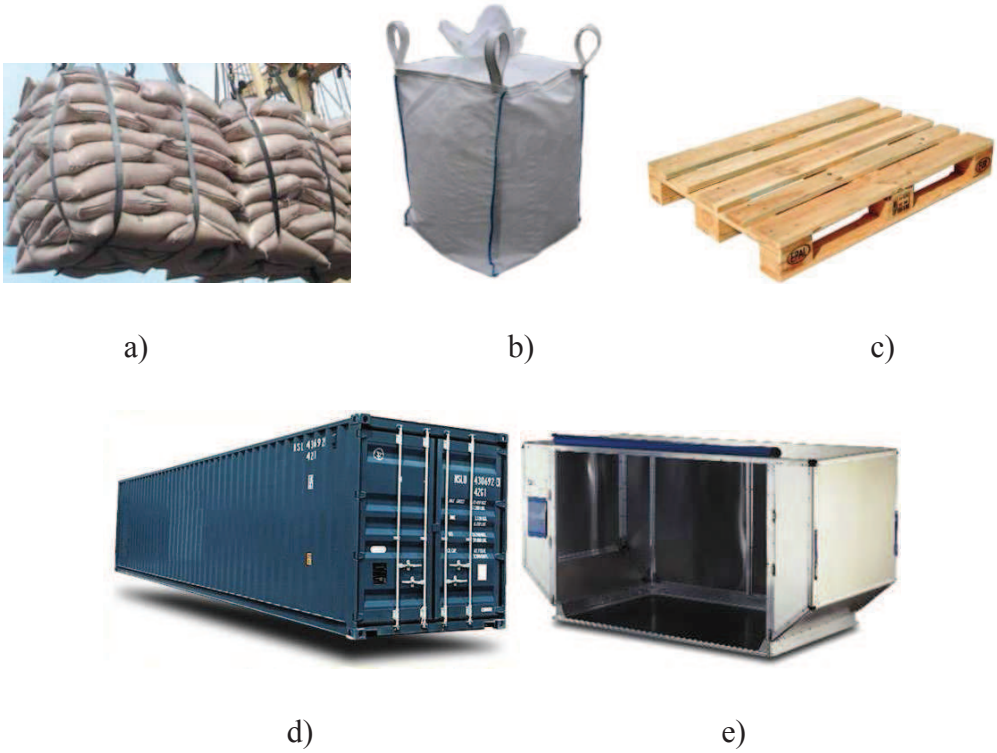
- redução dos custos de seguro de mercadoria;

- padronização internacional dos recipientes de unitização;

- redução do *lead-time* (tempo de percurso da origem até o cliente final).

Os principais dispositivos de unitização que são atualmente utilizados são: cinta marinha (*marino slings*), *big bags*, paletes, contêineres e ULDs; conforme mostrado na Fig. 2.2.

Neste trabalho em particular o foco está na utilização de contêineres para o agrupamento dos produtos. Segundo Bowersox e Closs (2007) os contêineres são equipamentos em que são colocadas embalagens secundárias ou produtos soltos, durante a armazenagem e o transporte. A utilização de contêineres no transporte de mercadorias, além das vantagens previamente citadas, em particular reduz o desperdício e a necessidade de descarte do dispositivo de unitização e uma maior proteção dos produtos contra fatores ambientais (BOWERSOX; CLOSS, 2007).



a) b) c)

d) e)

Figura 2.2 – Dispositivos de unitização: a) *marino slings*[1] b) *big bag*[2] c) palete[3] d) contêiner[4] e) ULD[5];

Existem diversos tipos de contêineres cujas dimensões e características variam de acordo com o tipo de carga que os mesmo transportam. Alguns exemplos de contêineres são mostrados na Fig. 2.3.

[1] Disponível em: <:http://www.logismarket.ind.br/fitacabo/cinta-para-elevacao/1211320573-1179618464-p.html> Acesso em Novembro de 2013.

[2]Disponível em: <http://www.sbpallet.com.br/big-bag/saco-big-bag.php> Acesso em Novembro de 2013.

[3]Disponível em: <http://www.logismarket.pt/palser/palete-de-madeira-eur-epal/148856561-1584345-p.html> Acesso em Novembro de 2013.

[4]Disponível em: <http://www.frontrangecontainers.com/> Acesso em Novembro de 2013

[5]Disponível em: <http://horizoninbusiness.com/id36.html> Acesso em Novembro de 2013.

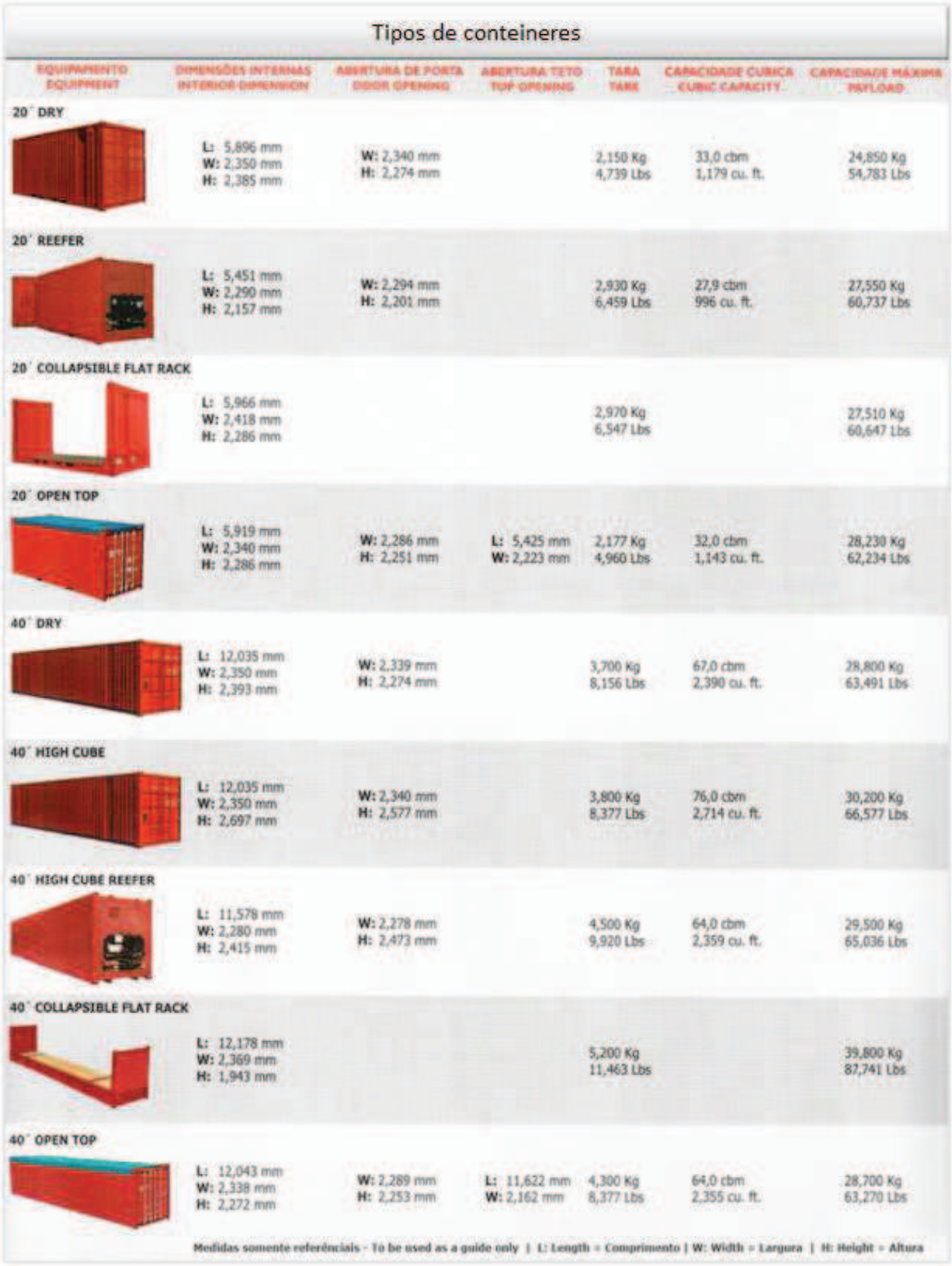


Figura 2.3 – Tipos de conteineres[6].

[6]Adaptado de <http://www.tanicomex.com.br/includes/tipos_de_containers.html> Acesso em Novembro de 2013.

2.2. Problemas de Corte e Empacotamento

Problema de Corte e Empacotamento (PCE) é o nome geral dado a uma classe de problemas de otimização combinatória que consiste na combinação de unidades menores (itens) dentro de unidades maiores (objetos) (TEMPONI, 2007).

Os PCEs em geral representam uma série de problemas que se classificam nesta categoria de alocação de produtos e recursos. A bibliografia deste assunto aborda os seus problemas com uma série de nomenclaturas são elas: *cutting and packing problems* (problemas de corte e empacotamento), *cutting stock problems* (problemas de corte de estoque), *trim loss problems* (problemas de cortes de perdas), *bin packing problems* (problema de carregamento de caixas), *strip packing problems* (problema de carregamento de faixas), *knapsack packing problems* (problema de carregamento de mochilas), *vehicle loading problems* (problema de carregamento de veículos), *pallet loading problems* (problema de carregamento de paletes), *container loading problems* (problema de carregamento de contêineres), dentre outros.

Segundo Dyckhoff (1990) os PCEs vêm sendo estudados há aproximadamente 60 anos. Um fato que se observa no início da década de 90 é o aparecimento de diversos trabalhos no assunto. Em resposta a imersão de vários artigos no que concernem a esta área de estudo, Dyckhoff (1990) e Wäscher; Haußner; Schumann (2006) em seus trabalhos estabelecem uma sistemática para o estudo e a classificação destes problemas. De acordo com Wäscher; Haußner; Schumann (2006), o estudo de uma tipologia ajuda a unificar as definições e notação auxiliando desta forma a comunicação entre os pesquisadores desta área.

A tipologia apresentada por Dyckhoff (1990) é pioneira no assunto de corte e empacotamento e posteriormente é complementada pelas considerações de Wäscher; Haußner; Schumann (2006) que contextualiza a tipologia inicial para uma abordagem mais moderna do assunto.

Inicialmente, Dyckhoff (1990) e Wäscher; Haußner; Schumann (2006) em seus trabalhos levantam as características em comuns para a lógica dos PCEs. A primeira delas diz que nesses problemas existem basicamente dois grupos de informação básica cujos elementos definem corpos geométricos ou tamanhos fixos e uma ou mais dimensões de espaço de números reais. O primeiro grupo consiste dos chamados “objetos grandes” e o segundo de uma lista de “itens pequenos”. A segunda

característica vem para complementar a primeira e diz que os problemas de corte e empacotamento consistem na geração de padrões de combinação dos "itens pequenos" associado aos "objetos grandes" e as partes residuais que não corresponde à lista dos itens pequenos tratam-se de perdas. Para exemplificar esta conceituação Dyckhoff (1990), apresenta dois problemas, o primeiro deles consiste no corte de tubos para fabricação de radiadores (Fig. 2.4) e o segundo consiste de um problema de carregamento de contêineres (Fig. 2.5).

Na Fig. 2.4 os tubos utilizados como matéria prima correspondem aos objetos grandes e por sua vez os produtos que são fabricados a partir destes são os itens pequenos. No problema de carregamento de contêiner, o contêiner propriamente corresponde ao objeto grande que será preenchido com as caixas que são os itens pequenos. Em ambos os problemas, a seção de tubo que não foi utilizada, assim como o volume do contêiner que não foi preenchido correspondem às perdas a serem minimizadas.

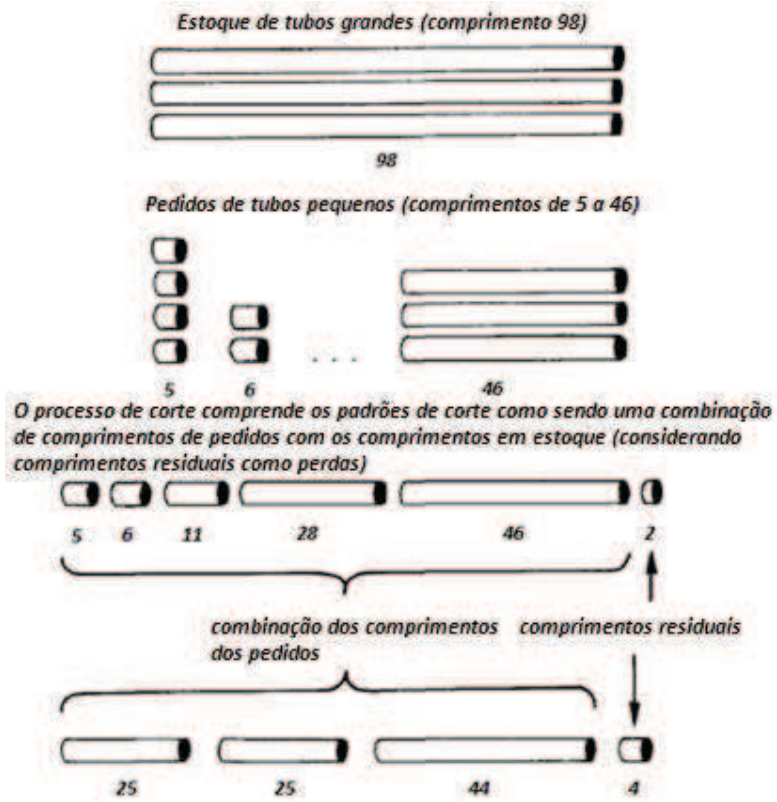


Figura 2.4 – Problema de corte de tubos. (adaptado de Dyckhoff (1990))

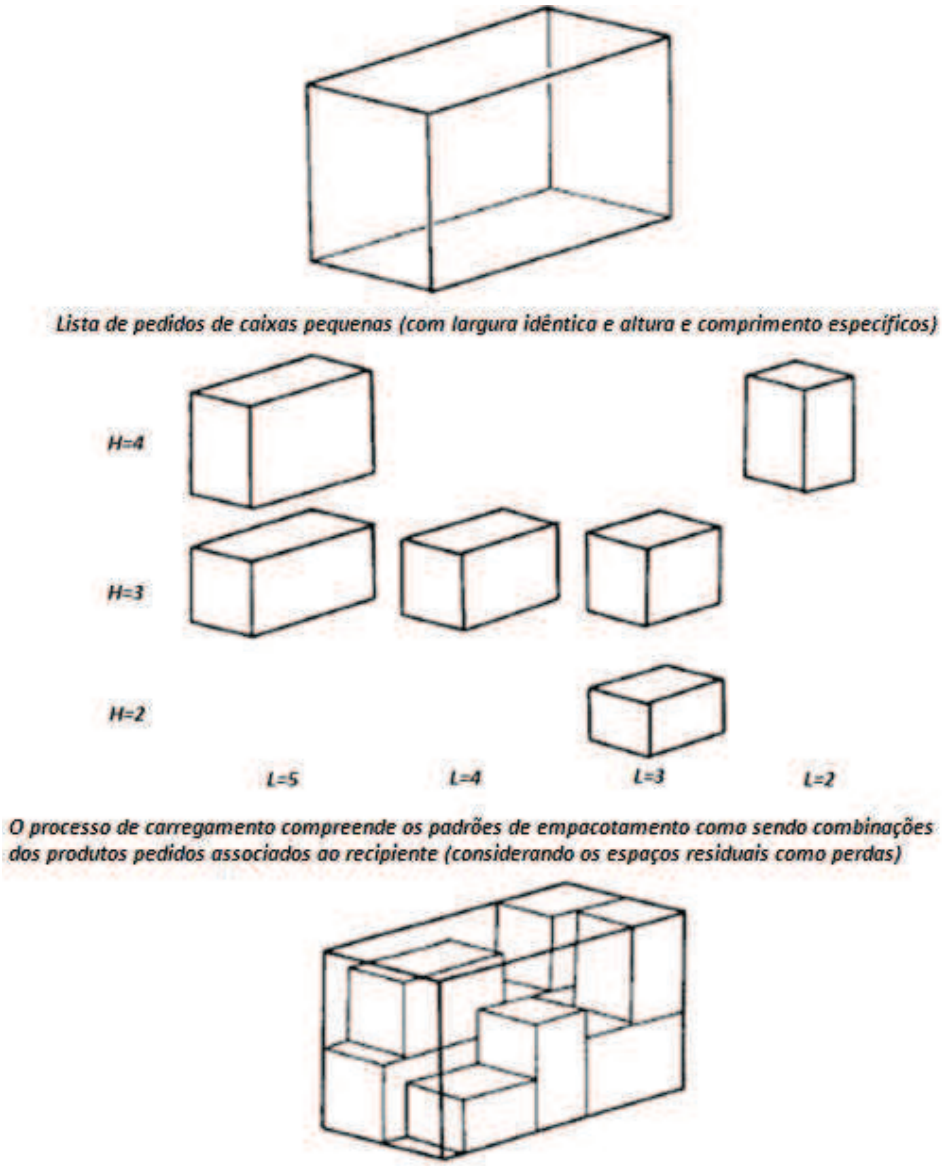


Figura 2.5 – Problema de carregamento de contêiner (adaptado de Dyckhoff (1990)).

Uma definição complementar é apresentada em Wäscher; Haußner; Schumann (2006), e diz que todos os itens pequenos selecionados devem caber inteiramente dentro das dimensões dos objetos grandes e, além disso, não pode haver uma sobreposição dos itens pequenos. Desta definição pode-se inferir que uma solução pode conter alguns ou todos os objetos pequenos e/ou alguns ou todos os objetos grandes.

Feito o embasamento lógico para os PCEs, Wäscher; Haußner; Schumann (2006) define cinco subproblemas a serem endereçados por esta área de estudo, são eles: problemas de seleção com relação aos objetos grandes; problemas de seleção com relação aos itens pequenos, problema de agrupamento com relação aos itens pequenos;

problemas de alocação dos subconjuntos de itens pequenos nos objetos grandes; e por último, problema de layout do arranjo dos itens pequenos em cada um dos objetos grandes selecionados.

Feito o embasamento lógico, Dyckhoff (1990) apresenta a primeira tipologia para os PCEs conforme mostrado na figura 2.6. A tipologia de Dyckhoff apresenta uma primeira tentativa para categorização dos PCEs, e serviu de base para o trabalho desenvolvido por Wäscher; Haußner; Schumann (2006), que vem para realizar críticas e complementar a tipologia pioneira. A ideia apresentada por Dyckhoff (1990) era baseada em quatro características principais: a dimensionalidade, que diz a respeito de quantas dimensões eram analisadas no problema; o tipo de atribuição, que diz a respeito de quais dos grupos (objetos ou itens) devem ser invariavelmente considerados na íntegra; a variabilidade dos objetos, que considerada as variações que podem ocorrer aos objetos; e por último a variabilidade dos itens, que considera a variação dos itens.

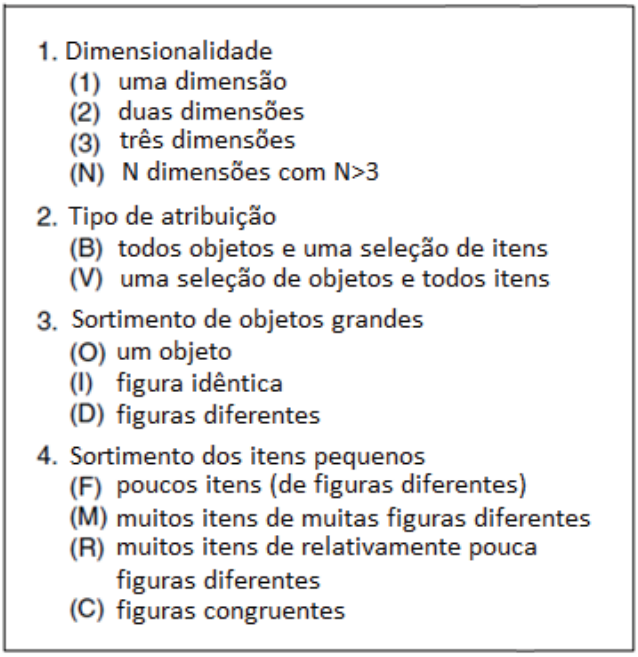


Figura 2.6– Tipologia apresentada por Dyckhoff (1990).

Vale ressaltar que a tipologia apresentada por Dyckhoff sofreu várias críticas uma vez que não contempla de maneira clara e semântica as variações existem nos PCEs. Dessa forma Wäscher; Haußner; Schumann (2006) modifica o paradigma anterior através das definições a seguir.

A dimensionalidade é considerada basicamente da mesma maneira que o modelo de Dyckhoff. Existem variantes de PCEs que podem considerar uma, duas ou três

dimensões espaciais e alguns casos existe a possibilidade de mais outras dimensões além das geométricas.

O tipo de atribuição é modificado de forma a contemplar dois tipos de situações: a maximização de saída e a minimização de entrada. A maximização de saída consiste nos problemas em que os objetos grandes não conseguem conter todos os itens a ele atribuído e o objetivo trata-se de se escolher um subconjunto de itens de forma que o aproveitamento do objeto grande seja máximo. Já a minimização de entrada consiste dos problemas em que ao contrário do anterior os objetos grandes conseguem conter todos os itens e o objetivo é minimizar o conjunto de objetos grandes utilizados. Vale ressaltar que na avaliação de um determinado objetivo seja para qualquer um dos dois tipos de atribuição, o valor atribuído a cada item pode ter um significado específico dependendo do problema considerado. Esse valor pode ser um valor de custo, um valor proporcional às dimensões, um valor de lucro, dentre outros.

A premissa da variabilidade dos itens é modificada de forma a contemplar uma definição mais clara. Wäscher; Haußner; Schumann (2006) faz uso de três situações para descrever essa característica. A primeira delas está relacionada a itens pequenos idênticos, nesta situação como o próprio nome já diz todos os itens relacionados ao problema são idênticos em forma e tamanho sendo a orientação deixada a parte. Esta situação faz um paralelo à situação C apresentada por Dyckhoff (1990). A segunda situação corresponde à variabilidade fracamente heterogênea. Nesta situação existem algumas classes de itens que são idênticos em forma e tamanho. Neste caso em particular a rotação é considerada como um diferencial entre classes. Esta situação faz um paralelo à classe R apresentada por Dyckhoff (1990). Por último, a terceira situação corresponde à variabilidade fortemente heterogênea. Neste caso, muito poucos itens são idênticos em forma e tamanho, assim sendo cada item pode ser tratado praticamente como uma classe a parte. Esta última situação corresponde às classes M e F apresentadas na tipologia de Dyckhoff (1990).

A variabilidade dos objetos foi reformulada para contemplar duas situações gerais: problemas com um objeto grande e problemas com alguns objetos grandes. Para a primeira situação conforme o próprio nome já diz tem-se apenas um objeto grande, e este pode estar submetido a duas outras condições. A primeira delas considera que o objeto grande possui as dimensões relevantes ao problema com valores fixos, na segunda condição uma ou mais dimensões podem ter o seu valor variável. Já a segunda situação contempla todos aqueles problemas em que existe um conjunto de objetos

grandes diferente do unitário. Nesta classe de problemas faz-se uma analogia ao paradigma da variabilidade dos objetos pequenos e resulta nas três seguintes classes: objetos grandes idênticos, variabilidade fracamente heterogênea, e variabilidade fortemente heterogênea.

Além das modificações propostas por Wäscher; Haußner; Schumann (2006), em seu trabalho ele introduz uma nova definição a tipologia, que é o formato dos itens. Nesta definição existem duas divisões: as formas regulares (retângulos, círculos, caixas, cilindros, bolas, etc.) e as formas irregulares.

Com esta nova tipologia em mãos Wäscher; Haußner; Schumann (2006) propõe uma série de problemas básicos, que são realizados através da combinação das características de atribuição e a variabilidade dos itens, conforme mostrado na Fig. 2.7.

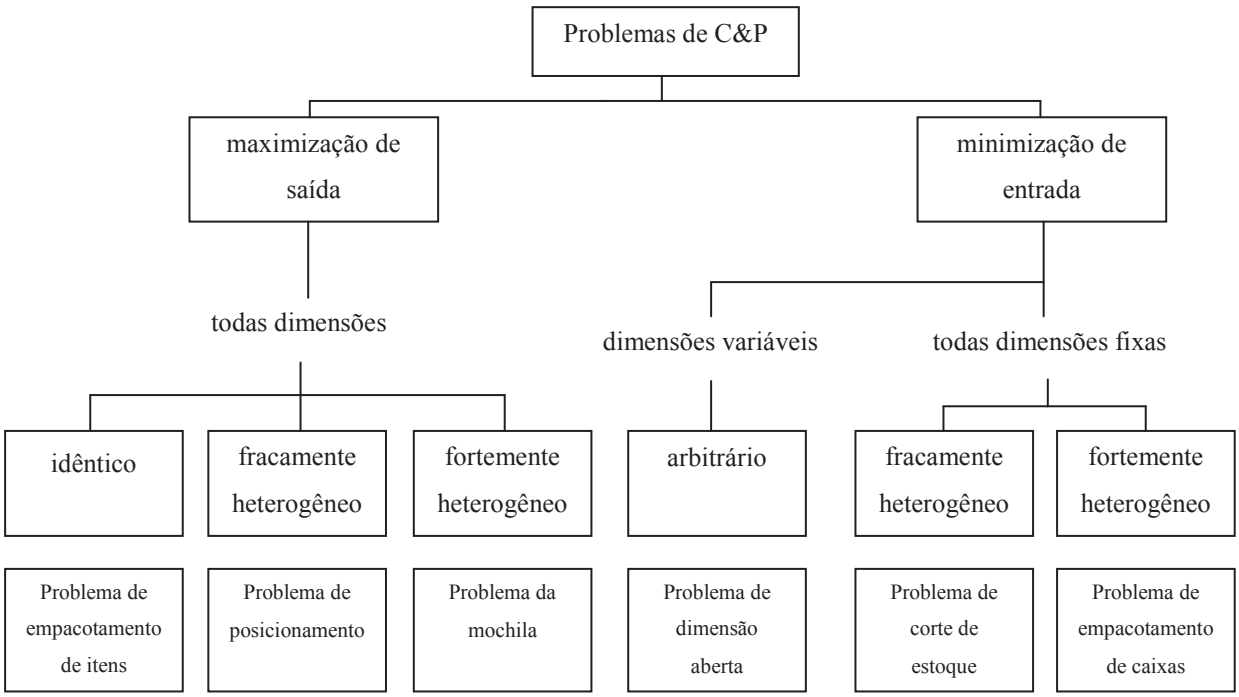


Figura 2.7 - Problemas básicos de corte e empacotamento (adaptado de Wäscher; Haußner; Schumann (2006)).

Este trabalho em particular possui interesse na maximização da saída de problemas fracamente heterogêneos resultando em uma classe de problemas denominadas problemas de posicionamento.

Ainda em seu trabalho Wäscher; Haußner; Schumann (2006) detalha a respeito de cada um dos problemas básicos apresentados na Fig. 2.8, o que o autor denomina de problemas intermediários, e os problemas de posicionamento, por sua vez, possuem

uma seção que detalha algumas das especificidades que se pode obter uma vez que se consideram as características dos objetos grandes, conforme mostra a figura abaixo.

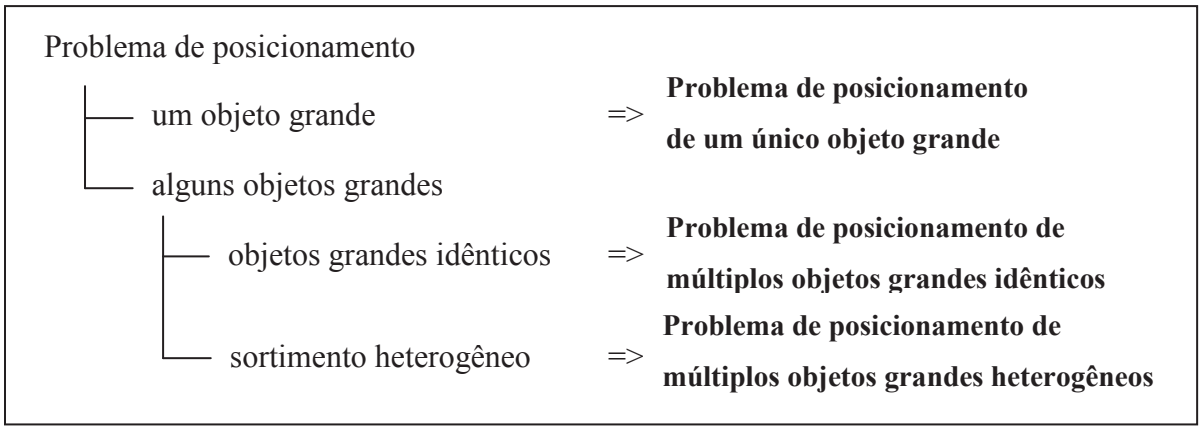


Figura 2.8 – Tipos de problemas de posicionamento (adaptado de Wäscher; Haußner; Schumann (2006)).

E por último é realizado um refinamento dos problemas em se utilizando as características de dimensionalidade e forma dos itens da tipologia apresentada.

Através da tipologia de Wäscher; Haußner; Schumann (2006) pode-se contextualizar o estudo deste trabalho como sendo um problema de posicionamento tridimensional e retangular de um único objeto grande (*3D Rectangular Single Large Object Placement Problem* – 3D-R-SLOPP). O problema de carregamento de contêiner de maneira geral corresponde a um problema particular desta classe em que o objeto grande trata-se de um contêiner e deseja-se alocar um conjunto de produtos dentro do mesmo.

2.2.1. Problema de Carregamento de Contêineres

O objetivo deste trabalho é justamente dotar o Problema de Carregamento de Contêineres (PCC) de certa "inteligência", de maneira que um sistema automatizado possa então realizar todas as etapas do posicionamento do produto. Para tanto o estudo das sistemáticas para determinação de padrão de empacotamento de caixas ótimo é necessário para que se obtenha um melhor aproveitamento deste modal de transporte.

A forma como será abordada o assunto neste trabalho, conforme já dito, se encaixa na categoria de problemas de posicionamento tridimensional e retangular de um

único objeto grande através da tipologia de Wäscher; Haußner; Schumann (2006), mas vale ressaltar que na literatura este problema é abordado sobre diversos de pontos de vista que não o restringe à apenas esta categoria de problemas. Na literatura existem varias designações para este assunto são elas: *container loading problem* (problema de carregamento de contêiner), *container packing problem* (problema de empacotamento de contêiner), *three-dimensional cargo-loading problem* (problema de carregamento de carga tridimensional), *three-dimensional packing problem* (problema de empacotamento tridimensional), *three-dimensional knapsack packing problem* (problema de empacotamento de mochila tridimensional), *three-dimensional bin packing problem* (problema de carregamento de caixas tridimensional), *three-dimensional strip packing problem* (problema de empacotamento de faixas tridimensional), *single container loading problem* (problema de carregamento de um único contêiner), *multiple container loading problem* (problema de carregamento de múltiplos contêineres), *three-dimensional pallet loading problem* (problema de carregamento de paletes tridimensional), *multi-pallet loading problem* (problema de carregamento de múltiplos paletes), *multi-pallet packing problem* (problema de empacotamento de múltiplos paletes), *three-dimensional palletization problem* (problema de paletização tridimensional), *three-dimensional cutting problem* (problema de corte tridimensional), dentre outros.

Junqueira (2009), em seu trabalho faz uma relação de todos os subtipos de problemas de acordo com a tipologia de Wäscher; Haußner; Schumann (2006), que podem endereçar o estudo de carregamento de contêineres.

Com relação à maximização de saída:

3D-R-IIPP (*Identical Item Packing Problem*): Um subconjunto de caixas iguais deve ser selecionado para ser carregado em um único contêiner, de modo a maximizar o volume (ou valor) total de caixas empacotadas dentro do contêiner.

3D-R-SLOPP (*Single Large Object Placement Problem*): Um subconjunto de caixas pouco diferentes deve ser selecionado para ser carregado em um único contêiner, de modo a maximizar o volume (ou valor) total de caixas empacotadas dentro do contêiner.

3D-R-MILOPP (*Multiple Identical Large Object Placement Problem*): Um subconjunto de caixas pouco diferentes deve ser selecionado para ser carregado em

vários contêineres iguais, de modo a maximizar o volume (ou valor) global de caixas empacotadas dentro dos contêineres.

3D-R-MHLOPP (*Multiple Heterogeneous Large Object Placement Problem*): Um subconjunto de caixas pouco diferentes deve ser selecionado para ser carregado em vários contêineres diferentes, de modo a maximizar o volume (ou valor) global de caixas empacotadas dentro dos contêineres.

3D-R-SKP (*Single Knapsack Problem*): Um subconjunto de caixas muito diferentes deve ser selecionado para ser carregado em um único contêiner, de modo a maximizar o volume (ou valor) total de caixas empacotadas dentro do contêiner.

3D-R-MIKP (*Multiple Identical Knapsack Problem*): Um subconjunto de caixas muito diferentes deve ser selecionado para ser carregado em vários contêineres iguais, de modo a maximizar o volume (ou valor) global de caixas empacotadas dentro dos contêineres.

3D-R-MHKP (*Multiple Heterogeneous Knapsack Problem*): Um subconjunto de caixas muito diferentes deve ser selecionado para ser carregado em vários contêineres diferentes, de modo a maximizar o volume (ou valor) global de caixas empacotadas dentro dos contêineres.

Com relação à minimização de entrada:

3D-R-SSSCSP (*Single Stock Size Cutting Stock Problem*): Um subconjunto de contêineres iguais deve ser selecionado para ser carregado com caixas pouco diferentes, de modo a minimizar o número (ou custo) global de contêineres necessários para empacotar todas as caixas.

3D-R-MSSCSP (*Multiple Stock Size Cutting Stock Problem*): Um subconjunto de contêineres pouco diferentes deve ser selecionado para ser carregado com caixas pouco diferentes, de modo a minimizar o número (ou custo) global de contêineres necessários para empacotar todas as caixas.

3D-R-RCSP (*Residual Cutting Stock Problem*): Um subconjunto de contêineres muito diferentes deve ser selecionado para ser carregado com caixas pouco diferentes, de modo a minimizar o número (ou custo) global de contêineres necessários para empacotar todas as caixas.

3D-R-SBSBPP (*Single Bin Size Bin Packing Problem*): Um subconjunto de contêineres iguais deve ser selecionado para ser carregado com caixas muito diferentes, de modo a minimizar o número (ou custo) global de contêineres necessários para empacotar todas as caixas.

3D-R-MBSBPP (*Multiple Bin Size Bin Packing Problem*): Um subconjunto de contêineres pouco diferentes deve ser selecionado para ser carregado com caixas muito diferentes, de modo a minimizar o número (ou custo) global de contêineres necessários para empacotar todas as caixas.

2.2.2. Restrições práticas

Para o estudo sistematizado do problema de carregamento de contêineres invariavelmente é necessário à criação de um modelo para a minimização ou maximização de uma determinada variável do problema. Os modelos, desta forma, são responsáveis pela tradução do comportamento e das características do sistema físico real. Vale ressaltar que de acordo com o problema analisado o modelo deve incorporar mais ou menos características ou restrições para torná-lo representativo do problema em questão. Nesse sentido, Bortfeldt e Wäscher (2012) apresentam as restrições que normalmente são consideradas no transporte de mercadorias em se utilizando a contêineres, paletes, caminhões, dentre outros. Neste trabalho as restrições são classificadas em cinco categorias: as restrições relacionadas aos contêineres, que diz respeito aos objetos grandes; as restrições relacionadas a itens individuais e a um subgrupo de itens, ambas relacionadas aos itens pequenos; e as restrições que estabelecem relações entre os objetos grandes e os itens pequenos, denominadas de restrições de posicionamento; e por último as restrições relacionadas ao resultado do processo de carregamento, denominadas de retrições de carregamento.

2.2.2.1. Restrições relacionadas ao contêiner

Restrição de limite de peso do contêiner: Esta restrição levanta o questionamento a respeito do limite máximo de peso das mercadorias a serem alocadas em um contêiner. Está restrição nem sempre é aparente, mas pode se tornar de extrema importância caso sejam carregados produtos pesados. Nestes casos o peso do contêiner pode vira a ser mais crítica do que o espaço disponível.

Restrição de distribuição de peso: Para esta restrição leva-se em consideração a distribuição dos produtos dentro do container de forma que o centro de massa resultante do conjunto (contêiner + produtos) seja o desejado (normalmente deseja-se que o centro de massa corresponda à posição centro geométrico do contêiner). Cargas balanceadas reduzem o risco de movimentação das produtos durante o processo de transporte. Além disso, deve-se considerar a distribuição dos produtos no caso do transporte por caminhões onde o peso por eixo é regulamentado.

2.2.2.2. Restrições relacionadas aos itens individuais

Restrição de prioridade: Esta restrição trabalha com o fato de que alguns podem apresentar uma maior prioridade com relação aos outros. Normalmente essa prioridade esta relacionada a prazos de entrega ou até mesmo prazos de validade no caso de produtos perecíveis.

Restrição de orientação: Para esta restrição existem normalmente aquelas caixas que devem manter uma orientação fixa devido às características dos produtos que nelas estão contidas, ou devido as características de acesso para o carregamento. Normalemente, neste caso, pode-se ter até duas dimensões que devem obedecer à restrição.

Restrição de empilhamento: Esta restrição trabalha com a ideia da pressão exercida no topo de uma caixa devido ao empilhamento de produtos. Devido às próprias características estruturais da embalagem, muitas vezes deve-se especificar um máximo de peso permissível na superfície da caixa. Vale ressaltar que muita das vezes o peso por área da caixa pode não ser o parâmetro ideal nesta restrição, portanto considera-se também o peso necessário para danificar os cantos da caixa.

2.2.2.3. Restrições relacionadas a um subgrupo de itens

Restrição de carregamento completo de itens: Para esta restrição considera-se o caso em que a um conjunto de item que formam uma máquina, equipamento ou conjunto funcional, nessas situações o conjunto de itens deve ser considerado como apenas um para efeitos de carregamento.

Restrição de alocação: Esta restrição surge apenas no carregamento de múltiplos contêineres, em que um subgrupo de itens deve ser alocado em um mesmo contêiner para o caso de mesma destinação, ou também quando o cliente final deseja que seus produtos sejam entregues em um mesmo carregamento e não em carregamentos fracionários. Existem situações em que alguns tipos de produtos não podem ser transportados em conjunto a outros (por exemplo, perfumaria e alimentícios).

2.2.2.4. Restrições relacionadas ao posicionamento dos itens

Restrição de posicionamento: Nesta restrição é de interesse que o posicionamento de itens, seja ele absoluto (está ou não em determinado container) ou relativo (está proximo a quais itens), seja trabalhado. Para o caso de posicionamento absoluto considera-se, por exemplo, a situação em que objetos mássicos (de grande volume) devem ser alocados no sentido de facilitar o processo de carregamento/descarregamento. No caso de posicionamento relativo tem-se o agrupamento de itens, em situações que envolvem a separação de produtos relativos a diferentes clientes. Por último existem também as situações de múltiplos destinos das mercadorias, em que ambos os casos de posicionamento absoluto e relativo devem ser combinados.

2.2.2.5. Restrições relacionadas ao carregamento

Restrição de estabilidade: Esta restrição considera a estabilidade do padrão de empacotamento das caixas, para que haja pouco movimento relativo entre as mesmas e por consequência menos a quantidade de danos aos produtos. Pode ser dividida em estabilidade vertical, em que os produtos não devem cair e portanto o empacotamento deve lidar diretamente com a força gravitacional, e em estabilidadade horizontal, que considera o movimento relativo de caixas durante o processo de transporte.

Restrição de complexidade de empacotamento: Esta restrição considera que quanto mais complexo o padrão de empacotamento das caixas maior o esforço para a manipulação das mesmas. Uma vez que normalmente não se pode substituir a tecnologia utilizada para a alocação das caixas, o padrão de empacotamento deve se conformar à tecnologia disponível.

2.2.3. Metodologias para resolução do PCC

Para a obtenção de um empacotamento ótimo no que diz respeito às restrições apresentadas anteriormente, deve-se propor um algoritmo de otimização que seja capaz de investigar as possibilidades de alocação dos produtos. O tema de carregamento de contêineres vem sendo estudado exaustivamente, e através deste estudo inúmeras técnicas e abordagens foram criadas para o assunto. A riqueza das técnicas de resolução do PCC traz todo um leque de possibilidades para abordagem do assunto em questão. Sendo assim uma breve revisão da literatura foi realizada neste trabalho a fim de que uma das técnicas fosse então escolhida no intuito da otimização do processo de carregamento.

Na tentativa de se obter um padrão ótimo de carregamento, várias técnicas tentam, através de diferentes abordagens, propor otimizações numéricas, evolutivas, simulações do pensamento humano, dentre outras. Entre as principais classes encontradas pode-se citar: os algoritmos baseados em colônias de formiga (YAP et al., 2012), os algoritmos baseados em colônias de abelhas (DERELI; DAS, 2010), algoritmos baseados na construção de blocos (ELEY, 2001), algoritmos baseado na técnica de *Branch and Bound* (BORTFELDT; MACK, 2006), algoritmos baseados em cortes guilhotinados (HIFI, 2001), algoritmos genéticos (GEHRING; BORTFELDT, 1997, NETO, 2006, SOAK; LEE, 2010), algoritmos híbridos (MACK; BORTFELDT;

GEHRING 2004, EGEBLAD; PISINGER, 2007, LIU et al. 2010, LIM et al., 2011), algoritmos baseados em programação linear (MATHUR, 1998, SCHEITHAUER, 1999, CARVALHO, 2002, MORABITO; ARENALES, 2007, CHIEN et al., 2008, JUNQUEIRA; MORABITO; YAMASHITA, 2011a, JUNQUEIRA; MORABITO; YAMASHITA, 2011b, CHE et al., 2011, JUNQUEIRA; MORABITO; YAMASHITA, 2012), algoritmos baseados em *tabu search* (LODI; MARTELLO; VIGO 2002, BORTFELDT; GEHRING; MACK, 2003, POLI; PUREZA, 2012), algoritmos de construção de torres (BISCHOFF; JANETZ; RATCLIFF, 1995) e por último algoritmos baseados em busca em árvore (WANG; LI; LEVY, 2007, REN; TIAN; SAWARAGI, 2011, LIM et al., 2012, ZHANG; PENG; LEUNG, 2011).

Para este trabalho em específico, após uma avaliação das técnicas encontradas na literatura, optou-se por uma abordagem em se utilizando a programação linear, uma vez que os trabalhos que utilizam esta abordagem fornecem um maior embasamento para construção de um algoritmo. Vale ressaltar que o objetivo inicial desta dissertação não constituiu na elaboração de uma nova técnica de para resolução do PCC e sim na construção de um sistema automático para o carregamento de produtos em container.

Sendo assim a próxima seção fará uma breve contextualização a respeito da teoria de programação linear e as possibilidades para resolução de problemas desta natureza.

2.3. Programação Linear

A programação linear lida com problemas de minimização ou maximização de uma função linear na presença de restrições lineares de igualdade e desigualdade (BAZARRA; JARVIS; SHERALI, 2010). A programação linear foi concebida pelo matemático George B. Dantzig em meados de 1947, quando o mesmo desempenhava o seu papel como conselheiro matemático para os Estados Unidos. Nesta época Dantzig estava trabalhando em uma ferramenta para o planejamento do transporte, treinamento e das atividades logística no que concernem as forças aéreas norte americanas. Curiosamente o matemático soviético L. V. Kantorovich resolvia esta mesma classe de problemas em um período anterior a Dantzig, porém seus feitos vieram ao público anos

depois da exposição do americano, motivo pela qual o crédito da programação linear é atribuído a Dantzig.

2.3.1. Formulação geral

Um problema de programação linear (PL) normalmente constitui-se de uma função objetivo da forma $Z = c_1x_1 + c_2x_2 + \cdots + c_nx_n$ (equação 2.1) a ser minimizada ou maximizada em que os coeficientes $\{c_1, c_2, \dots, c_n\}$ são denominados de coeficientes de custo e $\{x_1, x_2, \dots, x_n\}$ correspondem as variáveis de decisão ou de projeto. Associada a função objetivo existe uma série de expressões lineares de igualdade e desigualdade que são denominadas de restrições do problema. Um problema de programação linear pode então ser formulado conforme segue:

$$minimizar\ Z = c_1x_1 + c_2x_2 + \cdots + c_nx_n \tag{2.1}$$

$$sujeito\ a$$

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \geq b_1 \tag{2.2}$$

$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \geq b_2$$

$$\vdots$$

$$a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \geq b_m$$

$$x_i \geq 0 \tag{2.3}$$

Do sistema linear anterior os coeficientes $\{a_{ij}, |i = 1 \dots m \wedge j = 1 \dots n\}$ correspondem aos coeficientes tecnológicos do problema e o conjunto $\{b_1, b_2, \dots, b_m\}$ são denominadas constantes do lado direito. Na forma matricial pode-se representar o modelo geral da seguinte forma:

$$minimizar\ Z = cx \tag{2.4}$$

$$sujeito\ a$$

$$Ax \geq b \tag{2.5}$$

$$x \geq 0 \tag{2.6}$$

Em que x corresponde ao vetor das variáveis de projeto ou decisão, c corresponde ao vetor dos coeficientes de custo, A corresponde a matriz dos coeficientes tecnológicos e b corresponde ao vetor dos coeficientes do lado direito. A última

desigualdade, apresentada na formulação geral, traz consigo uma importante restrição que é inerente a todos os problemas de programação, que estabelece que todas as variáveis de projeto do problema devem ser estritamente não negativas.

As expressões de restrição de modo geral delimitam a região de soluções viáveis para um determinado problema. Para ilustrar o equacionamento apresentado, Bazarra; Jarvis; Sherali, (2010) apresenta um problema básico:

$$minimizar\ Z = 2x_1 + 5x_2 \tag{2.7}$$

$$sujeito\ a$$

$$x_1 + x_2 \geq 6 \tag{2.8}$$

$$-x_1 - 2x_2 \geq -18 \tag{2.9}$$

$$x_1, x_2 \geq 0 \tag{2.10}$$

A partir das restrições anteriores pode-se obter o seguinte gráfico.

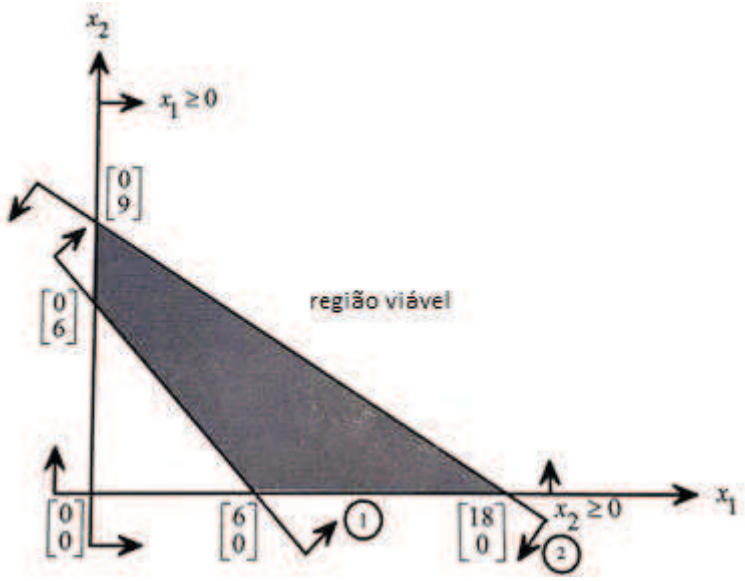
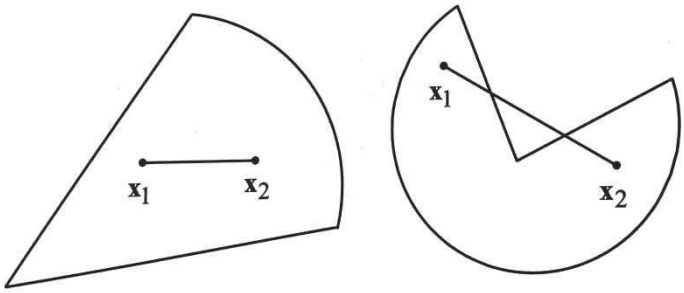


Figura 2.9 – Região viável para o problema (BAZARRA; JARVIS; SHERALI, 2010).

O problema trata-se então da busca de uma combinação de x_1 e x_2 tal qual a função objetivo seja minimizada. Vale ressaltar que em problemas de programação linear a região viável do problema, de acordo com a álgebra linear corresponde a uma região convexa (Fig. 2.10), ou seja, dados dois pontos x_1 e x_2 pertencentes à região viável, todos os pontos pertencentes à linha que liga os dois pontos iniciais, representados pela expressão $\lambda x_1 + (1 - \lambda)x_2$ também pertencem à região viável.



(a) (b)

Figura 2.10 – (a) região convexa (b) região não convexa (adaptado de Bazarra; Jarvis; Sherali (2010)).

O modelo geral apresentado anteriormente pode vir a sofrer algumas modificações dependendo do problema a ser abordado. O objetivo do problema pode vir a ser a maximização de alguma função ou pode haver restrições envolvendo inequações do tipo $\leq$ ou até mesmo equações de igualdade e, além disso, existem casos em que as variáveis de projeto podem ser irrestritas em sinal. É importante lembrar que para cada uma dessas variações existe uma forma correta para se endereçar e resolver o problema.

Para que se possa caracterizar um problema de programação linear existem algumas hipóteses que são implícitas e, portanto devem ser respeitadas.

Proporcionalidade: Esta hipótese considera que dada uma variável x_i, sua contribuição para a função objetivo e as equações de restrição é sempre proporcional ao seu valor.

Aditividade: Esta hipótese implica no fato de que a contribuição total das variáveis corresponde à soma da contribuição individual de cada uma das variáveis, dessa forma, não existe nenhum efeito de substituição de variáveis ou interação entre as mesmas.

Divisibilidade: Esta hipótese diz que o valor das variáveis de projeto pode ser dividido de maneira irrestrita por se tratar de uma resolução matemática, o que às vezes não é possível considerando-se o mundo real. Daí conclui-se que valores não inteiros são permitidos às variáveis de projeto.

Determinismo: Esta hipótese implica dizer que todas as constantes do problema (constantes de custo, tecnológicas e do lado direito) são todas conhecidas de forma

determinística, ou seja, não pode haver nenhuma incerteza no valor dessas constantes. Caso haja incerteza nos valores não se pode garantir que a programação linear levará a resultados corretos.

Uma vez definido o problema de programação linear existem basicamente dois métodos para a resolução do problema: o método gráfico e o método Simplex. O método gráfico é uma ferramenta bastante intuitiva e útil para a resolução de problemas de no máximo a 3 variáveis, porque esta metodologia basea-se numa inspeção visual da região viável das variáveis de projetos. Para abordagem de quaisquer tipos de problema, sem que haja a restrição do número de variável, utiliza-se o método Simplex.

2.3.2. Método Simplex

Dois anos após a apresentação de sua teoria de programação linear, Dantzig propôs um método para a resolução desta classe de problemas denominado de método Simplex. No decorrer dos anos não somente Dantzig quanto outros matemáticos trabalharam em cima deste método a fim de aperfeiçoa-lo, ainda assim apesar destas modificações, a ideia básica do método permaneceu e neste capítulo serão descritas todas as etapas para utilização desta técnica.

Inicialmente para avaliação de um problema de programação linear devem-se fazer algumas considerações quanto à optimalidade da solução. Para um problema de programação linear existem basicamente duas possibilidades para uma solução ótima: existe apenas uma solução, ou existem infinitas soluções ótimas. Para ilustrar o conceito apresentado, Bazarra; Jarvis; Sherali, (2010), utiliza-se de dois exemplos considerando-se cada uma das possibilidades de optimalidade. Sendo assim sejam os seguintes problemas:

$$minimizar\ \ z_1 = -3x_1 + x_2 (2.11) \qquad e \qquad minimizar\ z_2 = -2x_1 - 4x_2 \quad (2.12)$$

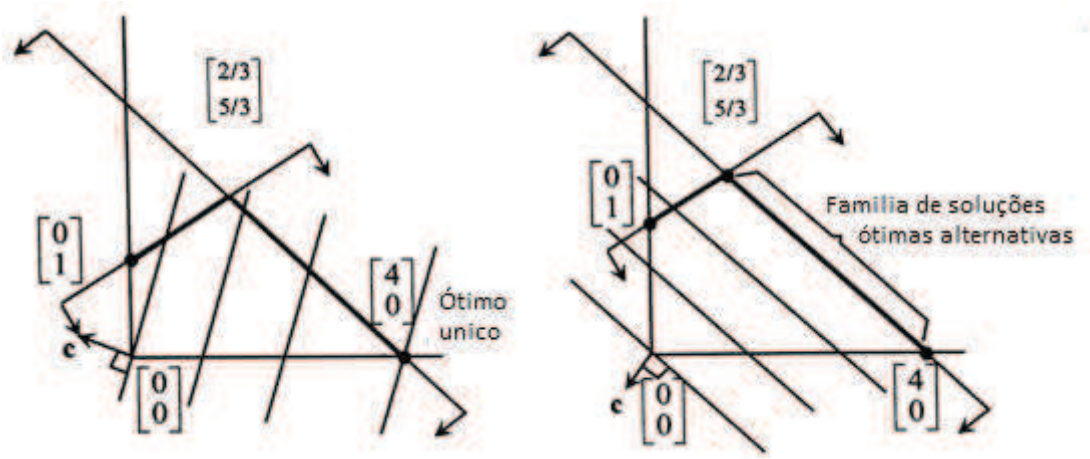
$ambas\ submetidas\ a$

$$x_1 + 2x_2 + x_3 = 4 \tag{2.13}$$

$$-x1 + x2 + x_4 = 1 \tag{2.14}$$

$$x_1, x_2, x_3, x_4 \geq 0 \tag{2.15}$$

Pela análise gráfica podem-se obter as seguintes figuras:



(a) (b)

Figura 2.11 – (a) Comportamento da função z_1 na região viável (b) Comportamento da função z_2 na região viável (adaptado de Bazarra; Jarvis; Sherali, (2010)).

Conforme se pode observar através da figura anterior, para o caso de uma única solução, a função objetivo no seu valor ótimo tangencia a região viável em um único ponto. Para esta situação diz-se que a solução faz parte de um conjunto de pontos chamados de pontos extremos. Estes pontos consistem de todos os vértices que se formam da delimitação da região viável do problema, da definição da álgebra linear os pontos extremos são pontos da região viável para qual a expressão $\lambda x_1 + (1 - \lambda)x_2$, conforme apresentada anteriormente, resulta que x_1 deve necessariamente ser igual a x_2. Para a segunda situação o tangenciamento da função objetivo no ótimo define uma semi-reta em que todos os pontos nela contida correspondem a uma solução ótima. De maneira geral para determinação do ponto ótimo deve-se considerar a direção definida pelo gradiente da função objetivo (que corresponde ao vetor dos coeficientes de custo) ou seu correspondente inverso, de forma que o hiperplano definido pela solução para um determinado valor de Z permaneça dentro da região viável do problema.

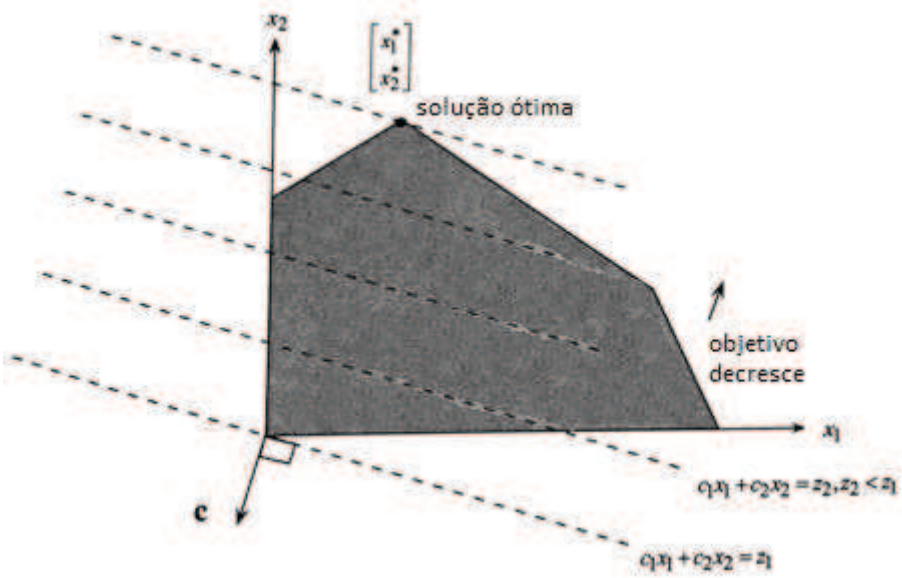


Figura 2.12 – Exemplo de direção de busca para $z = c_1x_1 + c_2x_2$ (adaptado de Bazarra; Jarvis; Sherali, (2010)).

Uma terceira situação pode ser levantada, que corresponde ao caso em que região viável e valor ótimo da função objetivo são irrestritos (para o caso do valor da função objetivo tem-se o valor∞ ou $-\infty$), para este caso se diz que a solução ótima é inexistente.

Feita as considerações a respeito da optimalidade de um problema de programação linear, faz-se necessário à introdução de alguns conceitos fundamentais para o entendimento do método simplex:

Solução: uma solução corresponde a uma atribuição qualquer de valores às variáveis de projeto ou de decisão de um problema.

Solução praticável: é uma solução em que nenhuma das restrições do problema são violadas.

Solução impraticável: é uma solução em que há violação de pelo menos uma das equações de restrição do problema.

Solução básica: seja um problema com m equações linearmente independentes com n incógnitas. Uma solução básica corresponde a uma solução em que $(n - m)$ são igualadas a zero e as outras são obtidas através do sistema de equações. O número de soluções básicas de um problema de PL pode ser calculado através da seguinte expressão: $n^{\underline{o}}\ sol\ básicas = \binom{n}{m} = \frac{n!}{m!(n-m)!}$

Variáveis básicas (VB): da definição as variáveis básicas são aquelas que não são igualadas à zero.

Variáveis não básicas (VNB): as variáveis não básicas, por sua vez, são aquelas igualadas á zero na solução básica.

Solução básica degenerada: é uma solução básica em que pelo menos uma das variáveis básicas tem seu valor calculado igual à zero.

Variáveis de folga: as variáveis de folga são um artifício matemático utilizada na conversão de inequações em equações.

Para ilustrar as definições apresentadas, segue um problema utilizado por Santos (2000) em seu trabalho.

$$maximizar\ \ Z = 20x_1 + 60x_2 \tag{2.16}$$

$sujeito\ a$

$$70x_1 + 70x_2 \leq 4900 \tag{2.17}$$

$$90x_1 + 50x_2 \leq 4500 \tag{2.18}$$

$$2x_1 \leq 80 \tag{2.19}$$

$$3x_2 \leq 180 \tag{2.20}$$

$$x_1, x_2 \geq 0 \tag{2.21}$$

O método simplex trabalha com a investigação do conjunto das soluções básicas do sistema, porém, para que esta metodologia possa ser utilizada o sistema deve estar obrigatoriamente na forma padrão, em que todas as restrições estejam representadas por meio de equações. Para tanto, conforme apresentado anteriormente, deve-se utilizar as variáveis de folga. Assim o sistema fica como segue:

$$maximizar\ \ Z = 20x_1 + 60x_2 \tag{2.16}$$

$sujeito\ a$

$$70x_1 + 70x_2 + F_1 = 4900 \tag{2.22}$$

$$90x_1 + 50x_2 + F_2 = 4500 \tag{2.23}$$

$$2x_1 + F_3 = 80 \tag{2.24}$$

$$3x_2 + F_4 = 180 \tag{2.25}$$

$$x_1, x_2 \geq 0 \tag{2.21}$$

Deste sistema em específico pode-se então avaliar o número de soluções básicas como sendo igual 15. Vale ressaltar que em problemas com centenas de variáveis e

restrições esse número pode atingir valores gigantesco o que inviabiliza a avaliação de todas as soluções básicas. Complementando o que foi dito no parágrafo anterior a metodologia do Simplex propõe uma busca orientada no domínio das soluções básicas.

Uma solução básica do problema anterior pode ser encontrada da seguinte forma. Das informações fornecidas o número de variáveis não básicas ou iguais à zero corresponde a$(n - m) = (6 - 4) = 2$. Sejam entãox_1 e x_2 as variáveis não básicas e, portanto, F_1, F_2, F_3 e F_4 as variáveis básicas. Substituindo no sistema de equações

$$Z = 20.0 + 60.0 = 0 \tag{2.26}$$

$$70.0 + 70.0 + F_1 = 4900 \rightarrow F_1 = 4900 \tag{2.27}$$

$$90.0 + 50.0 + F_2 = 4500 \rightarrow F_2 = 4500 \tag{2.28}$$

$$2.0 + F_3 = 80 \rightarrow F_3 = 80 \tag{2.29}$$

$$3.0 + F_4 = 180 \rightarrow F_4 = 180 \tag{2.30}$$

Uma ressalva que deve ser feita ao cálculo das soluções básicas é o fato de que da mesma forma que as variáveis projeto devem ser estritamente não negativas, as variáveis de folga devem possuir o mesmo comportamento. É fácil provar por meio da análise gráfica que quando uma das variáveis de folga é negativa caracteriza-se uma solução impraticável em que uma ou mais restrições são violadas.

Por fim faz-se necessário uma última definição que é fundamental a metodologia do Simplex:

Solução básica praticável adjacente: Duas soluções básicas praticáveis são adjacentes se elas diferem entre si de apenas uma variável não básica.

Então, de maneira geral a ideia do método Simplex consiste da utilização de operações elementares de matrizes no intuito da avaliação das soluções básicas de forma orientada. Do conceito de solução básica adjacente resultam duas situações: se uma solução básica praticável é melhor que suas adjacentes, ela é a solução ótima; se uma solução básica for igual a algumas soluções e melhor que todas as outras adjacentes, tem-se a situação de infinitas soluções.

O algoritmo para implementação do método simplex básico é apresentado por Vanderplaats (2005), conforme mostra a tabela a seguir.

Tabela 2.1 - Metodologia do Simplex (adaptado de Vanderplaats (2005)).

1. Encontre o mínimo $c_i, i = 1, n.$ seja k o índice desta variável com coeficiente de custo c_k. Sec_ké não negativo, vá para o passo 5.
2. c_ké negativo. Agora encontre o mínimo $b_j/a_{jk} j = 1, m$ para todos os a_{jk} positivos. Se todos os a_{jk}são não positivos, a solução é irrestrita. Então vá para o passo 5. Caso contrário, seja r o índice da coluna do referido mínimo. a_{rk}é agora o element pivô.
3. Pivô em a_{rk}. Primeiro divide-se a linha r por a_{rk}. Depois subtraia a_{jk}vezes $r$das colunas $j, j = 1, m + 1$e$j \neq r$.
4. Vá para o passo 1 e repita.
5. A solução esta completa.
6. Se todos os $c_i$associados com variáveis não básicas são positivos, a solução é única.
7. Se algum c_i associado a variável não básica é zero, a solução não é única.
8. Se algum c_ié negativo, a solução é irrestrita.

Para demonstrar o procedimento apresentado por Vanderplaats (2005), considera-se o problema na forma padrão apresentado anteriormente.

$$Z - 20x_1 - 60x_2 = 0 \tag{2.31}$$

$$70x_1 + 70x_2 + F_1 = 4900 \tag{2.32}$$

$$90x_1 + 50x_2 + F_2 = 4500 \tag{2.33}$$

$$2x_1 + F_3 = 80 \tag{2.34}$$

$$3x_2 + F_4 = 180 \tag{2.35}$$

$$x_1, x_2 \geq 0 \tag{2.21}$$

Para dar inicio a procedimento, primeiramente deve-se encontrar uma solução inicial viável. Em problemas de maximização, uma solução inicial pode ser encontrada fazendo as variáveis de projeto iguais à zero. Assim a solução básica inicial é da forma:

$$VB = \begin{Bmatrix} F_1 = 4900 \\ F_2 = 4500 \\ F_3 = 80 \\ F_4 = 180 \end{Bmatrix}, \qquad VNB = \begin{Bmatrix} x_1 = 0 \\ x_2 = 0 \end{Bmatrix}, \qquad Z = 0$$

Dando inicio ao método apresentado analisa-se a função objetivo a fim de se encontrar o menor valor de c_i. Este procedimento é realizado no sentido de se encontrar a variável que mais contribui para o crescimento da função objetivo caso ela seja transformada em uma VB. O mínimo valor de c_i é -60, que está associado a variável x_2. Como o coeficiente de custo da variável x_2 é negativa, esta variável será selecionada

para entrar no grupo das VBs, e uma vez quex_2 entrará para o grupo das VBs, uma das variáveis básicas atuais deverá se tornar uma VNB.

A escolha da variável que se tornará uma VNB é uma consequência do fato de que x_2 não poderá crescer de forma irrestrita. Assim analisando as restrições tem-se que:

$$F_1 = 4900 - 70x_1 - 70x_2 \quad (2.36)$$
$$F_2 = 4500 - 90x_1 - 50x_2 \quad (2.37)$$
$$F_3 = 80 - 2x_1 \quad (2.38)$$
$$F_4 = 180 - 3x_2 \quad (2.39)$$

Mas como x_1 faz parte das VNBs, ou seja, $x_1 = 0$:

$$F_1 = 4900 - 70x_2 \quad (2.40)$$
$$F_2 = 4500 - 50x_2 \quad (2.41)$$
$$F_3 = 80 \quad (2.42)$$
$$F_4 = 180 - 3x_2 \quad (2.43)$$

Na equação (2.40) observa-se que o crescimento de x_2 está limitado ao valor de 70, pois para valores maiores do que esse resulta em F_1 negativo, o que não é permitida a uma solução praticável. Na equação (2.41) x_2 está limitado ao valor 90. Na equação (2.42) não existe nenhuma restrição ao crescimento de x_2. Na equação (2.43) x_2 está limitado ao valor 60. Desta forma, para que nenhuma restrição seja violada x_2 pode atingir um valor de no máximo 60 e variável que entrará para as VNBs é F_4. Esta etapa corresponde ao cálculo da relação b_j/a_{jk}conforme o procedimento, e o valor 3 que está associado a x_2 na restrição de maior limitação é agora o pivô.

O grupo das variáveis básicas e não básicas está definido e para continuação do processo é necessário o reescalonamento do sistema. Observando a iteração realizada, no inicio tem-se todas as VNB contidas na função objetivo e as VBs em contrapartida aparecem apenas uma vez nos sistema de equações. Esta notação é extremamente favorável visto que facilita o intercambio de variáveis entre as VBs e VNBs, além de permitir uma análise direta das restrições. Portanto faz-se o reescalonamento do sistema utilizando o pivô.

Fazendo as operações mostradas no procedimento resulta no seguinte sistema de equações:

$$z - 20x_1 + 20F_4 = 3600 \quad (2.44)$$

$$70x_1 + F_1 - \frac{70}{3}F_4 = 700 \quad (2.45)$$

$$90x_1 + F_2 - \frac{50}{3}F_4 = 1500 \quad (2.46)$$

$$2x_1 + F_3 = 80 \quad (2.47)$$

$$x_2 + \frac{1}{3}F_4 = 60 \quad (2.48)$$

Sendo que:

$$VB = \begin{Bmatrix} F_1 = 700 \\ F_2 = 1200 \\ F_3 = 80 \\ x_2 = 60 \end{Bmatrix}, \qquad VNB = \begin{Bmatrix} x_1 = 0 \\ F_4 = 0 \end{Bmatrix}, \qquad Z = 3600$$

Realizando o procedimento mais uma vez, tem-se que a variável x_1 é a escolhida para fazer parte das VBs. Analisando as restrições resulta que a equação contendo F_1e x_1 é a mais limintante, portanto F_1 fara parte das VNBs e o pivô é o coeficiente 70. Reescalonando o sistema:

$$Z + \frac{20}{70}F_1 + \frac{40}{3}F_4 = 3800 \quad (2.49)$$

$$x_1 + \frac{1}{70}F_1 - \frac{1}{3}F_4 = 700 \quad (2.50)$$

$$-\frac{90}{70}F_1 + F_2 + \frac{40}{3}F_4 = 600 \quad (2.51)$$

$$-\frac{2}{70}F_1 + F_3 + \frac{2}{3}F_4 = 60 \quad (2.52)$$

$$x_2 + \frac{1}{3}F_4 = 60 \quad (2.53)$$

Sendo que:

$$VB = \begin{Bmatrix} x_1 = 10 \\ F_2 = 600 \\ F_3 = 60 \\ x_2 = 60 \end{Bmatrix}, \qquad VNB = \begin{Bmatrix} F_1 = 0 \\ F_4 = 0 \end{Bmatrix}, \qquad Z = 3600$$

Para a situação em que o sistema se encontra não existe nenhum c_i negativo, então de acordo com o procedimento deve-se seguir para o passo 5, que diz que a solução está completa. Os passos 6, 7 e 8 determinam o tipo de solução encontrada, para este caso como todos os coeficientes associados às VNBs na função objetivo são positivos a solução é única, se por acaso um dos coeficientes associados a alguma das

VNB fosse zero é caracterizado o caso de infinitas soluções, por último se o passo 5 for atingido através da condição do passo 2 tem-se uma solução irrestrita.

Nos problemas de carregamento de contêiner muitas vezes a modelagem contêm expressões de restrição que não se encaixam no formato apresentado no exemplo, conforme já dito existe uma forma de endereçar cada uma dessas diferenças.

Primeiramente pode-se desejar a minimização de um determinado custo. Na abordagem de problemas de minimização o procedimento descrito pode ser utilizado através de um artifício matemático, que consiste em se multiplicar a expressão a ser minimizada por -1 e trabalhar com esta nova expressão no procedimento de maximização. Ao fim do processo de maximização basta multiplicar o valor de Z por -1 e esse é o valor da função no mínimo. Existe também a situação em que algumas variáveis são irrestritas em sinal, um outro artifício matemático é então utilizado e consiste na transformação de uma variável irrestrita em duas restrição, por exemplo, seja x_4 irrestrita em sinal. Com o artifício citado pode se então dizer que $x_4 = x_5 - x_6$, com $x_5, x_6 \geq 0$. Para valores não negativos de x_4 tem-se que $x_5 \geq x_6$, para valores negativos de x_4 tem-se que $x_6 > x5$.

Por último em problemas PL em que há restrições do tipo $\geq$ e $=$, é necessário uma abordagem mais elaborada. Nesta classe de problemas existem algumas que podem ser citada no intento de sua resolução, são elas: substituição da igualdade por duas desigualdades, processo do “M grande”, método da função objetivo artificial e o método das duas fases. Neste trabalho em particular foi estudado o método de duas fases, o qual será explicado na sequência.

2.3.3. Método Simplex de Duas Fases

O método do Simplex de duas fases tem esse nome porque para a resolução dos problemas com as restrições citadas anteriormente são necessárias duas fases: a primeira delas consiste de se encontrar uma solução básica praticável inicial, pois nessa classe de problemas essa solução muitas vezes não é trivial uma vez que deve atender a restrições de igualdade, feito isso a segunda fazenda trabalha com um modelo equivalente ao original no sentido da maximização ou minimização da função objetivo em se utilizando o procedimento do simplex tradicional.

Novamente utilizando um exemplo de Santos (2000) será demonstrada a metodologia do Simplex de duas fases. Seja o seguinte problema:

$$maximizar\ Z = 4x_1 + 3x_2 \tag{2.54}$$

$sujeito\ a$

$$x_1 + 2x_2 = 10 \tag{2.56}$$

$$6x_1 + 6x_2 \leq 40 \tag{2.57}$$

$$x_1 \geq 2 \tag{2.58}$$

$$x_2 \geq 0 \tag{2.59}$$

Note que o problema em questão possui uma expressão de igualdade (equação 2.56) e uma desigualdade do tipo ≥ (equação 2.58), vale ressaltar que a expressão 2.59 apenas representa a condição inerente de todo problema de PL. Para lidar com esses tipos de restrições deve se utilizar o conceito de **variáveis artificiais**. As variáveis artificiais são adicionadas a fim de se acrescentar certa flexibilidade para o início do processo e normalmente são utilizadas como variáveis básicas iniciais, porém não possuem nenhum significado físico para o problema. Dessa forma o sistema na forma padrão fica como segue:

$$Z - 4x_1 - 3x_2 = 0 \tag{2.60}$$

$$x_1 + 2x_2 + A_1 = 10 \tag{2.61}$$

$$6x_1 + 6x_2 + F_2 = 40 \tag{2.62}$$

$$x_1 - F_3 + A_3 = 2 \tag{2.63}$$

Do sistema anterior observa-se que na segunda equação, pelo fato de ela ser originalmente uma expressão de igualdade, a mesma não possui folga, portanto acrescenta-se apenas uma variável artificial. Já a equação (2.63) consiste de uma desigualdade do tipo ≥, esta equação não possui folga e sim uma sobra, razão pela qual é introduzida a variável F_3. Nos problemas apresentados na seção anterior a solução básica inicial com variáveis de projeto nulas era uma opção, se esse procedimento for adotado para a desigualdade do tipo ≥ resulta em um valor inicial de F_3 negativo ($x_1 - F_3 = 2 \rightarrow 0 - F_3 = 2 \rightarrow F_3 = -2$). Por esta razão é introduzida uma segunda variável artificial.

Nota-se que para uma solução atender de fato todas as restrições do sistema original, o valor das variáveis artificiais devem ser iguais a zero, motivo pela qual a primeira etapa do Simplex de duas fases consiste na minimização das variáveis artificiais. Assim nesta primeira fase a função objetivo inicial será substituída pela soma das variáveis artificiais do problema, de tal forma que:

$$W = A_1 + A_3 \tag{2.64}$$
$$x_1 + 2x_2 + A_1 = 10 \tag{2.61}$$
$$6x_1 + 6x_2 + F_2 = 40 \tag{2.62}$$
$$x_1 - F_3 + A_3 = 2 \tag{2.63}$$

Mas como as variáveis artificiais serão utilizadas como VBs iniciais deve-se então retira-las da função objetivo. Dessa forma o sistema de equação fica conforme segue:

$$W - 2x_1 - 2x_2 + F_3 = -12 \tag{2.65}$$
$$x_1 + 2x_2 + A_1 = 10 \tag{2.61}$$
$$6x_1 + 6x_2 + F_2 = 40 \tag{2.62}$$
$$x_1 - F_3 + A_3 = 2 \tag{2.63}$$

Utilizando a metodologia apresentada na seção anterior o sistema no ótimo fica da seguinte forma:

$$W + A_1 + A_3 = 0 \tag{2.66}$$
$$x_2 + \frac{1}{2}F_3 + \frac{1}{2}A_1 - \frac{1}{2}A_3 = 4 \tag{2.67}$$
$$F_2 + 3F_3 - 3A_1 - 3A_3 = 4 \tag{2.68}$$
$$x_1 - F_3 + A_3 = 2 \tag{2.69}$$

Sendo que:

$$VB = \begin{Bmatrix} x_2 = 4 \\ F_2 = 4 \\ x_1 = 2 \end{Bmatrix}, \qquad VNB = \begin{Bmatrix} A_1 = 0 \\ A_3 = 0 \\ F_3 = 0 \end{Bmatrix}, \qquad W = 0$$

Desse estado pode-se observar que o valor ótimo do sistema é zero, o que significa dizer que as variáveis artificiais podem então ser retiradas umas vez que sua contribuição é nula. Caso o resultado desta primeira fase fosse diferente zero dizer que o modelo não possui nenhuma solução básica praticável, ou seja, não existe nenhuma

solução para o problema. Para a segunda fase resgata-se a função objetivo inicial considerando o sistema de equações obtido com a primeira fase. Assim:

$$Z - 4x_1 - 3x_2 = 0 \tag{2.60}$$

$$x_2 + \frac{1}{2}F_3 = 4 \tag{2.70}$$

$$F_2 + 3F_3 = 4 \tag{2.71}$$

$$x_1 - F_3 = 2 \tag{2.72}$$

Como as variáveis x_1 e x_2 são variáveis básicas de acordo com a primeira fase, então elas devem ser retiradas da função objetivo. Dessa forma:

$$Z - \frac{5}{2}F_3 = 20 \tag{2.73}$$

$$x_2 + \frac{1}{2}F_3 = 4 \tag{2.70}$$

$$F_2 + 3F_3 = 4 \tag{2.71}$$

$$x_1 - F_3 = 2 \tag{2.72}$$

Utilizando a metodologia tradicional do Simplex pode se obter a seguinte resposta:

$$Z + \frac{5}{6}F_2 = \frac{70}{3} \tag{2.73}$$

$$x_2 - \frac{1}{6}F_2 = \frac{10}{3} \tag{2.74}$$

$$\frac{1}{3}F_2 + F_3 = \frac{4}{3} \tag{2.75}$$

$$x_1 + \frac{1}{3}F_2 = \frac{10}{3} \tag{2.76}$$

E:

$$VB = \begin{Bmatrix} x_2 = \dfrac{10}{3} \\ F_3 = \dfrac{4}{3} \\ x_1 = \dfrac{10}{3} \end{Bmatrix}, \qquad VNB = \{F_2 = 0\}, \qquad Z = \frac{70}{3}$$

Dessa forma, tem-se em mãos uma ferramenta para a resolução de problemas de programação linear de qualquer categoria, que é de grande ajuda uma vez que as proposta de modelos muitas das vezes possuem características das mais variadas. Na

próxima seção são avaliadas as possibilidades para construção de um sistema totalmente automatizado

2.4. Automação Industrial

Na sociedade atual, com o desenvolvimento dos parâmetros de mercado e a acessibilidade a uma grande variedade de produtos, as empresas trabalham de forma a considerar algumas dessas características da realidade econômica. Em geral, devido a globalização no sentido comercial, deseja-se cada vez mais competitividade e redução nos custos de manufatura, ambas considerando um certo nível de qualidade a ser mantido ou atingido.

Para modificação das características de produção, normalmente o que se faz é o estudo sistemático do sistema de produção e seus processos. A manufatura , ou produção, pode ser definida com a aplicação de processos fisicos e químicos na alteração da geometria, de propriedades e/ou da aparência de determinado material inicial com vistas a produzir peças ou produtos.(...) Os processos envolvidos na produção englobam um combinação de máquinas, ferramentas, força e trabalho manual (...) e quase sempre acontecem como uma sequencia de operações (GROOVER, 2011). De forma geral, as operações contidas em processo produtivo podem ser de três categorias: as operações manuais, nas quais um ou mais trabalhadores executam uma tarefa sem a assistência de ferramentas motorizadas;as operações em que há uma combinação homem-máquina, em que um trabalhor opera um equipamento tal qual uma máquina ferramenta ou alguma máquina de produção; e por último as operações automatizas, nas quais o processo é desempenhado de forma integral por uma máquina sem que haja a participação de um operador.

Na modificação dos sistemas de produção, a exceção de alguns tipos de négocios, existe uma tendência de migração dos processos que vai das operações manuais até as operações totalmente automatizadas. Conforme já dito anteriormente as razões para automação são as seguintes: aumentar a produtividade; reduzir custos do trabalho (mão de obra); minimizar o efeito da falta de trabalhadores; reduzir ou eliminar as rotinas manuais e das tarefas administrativas; aumentar a segurança do trabalhador;

melhorar a qualidade do produto; diminuir o tempo de produção; realizar processos que não podem ser executados manualmente; evitar o alto custo da não automação.

Dessa forma, automação industrial pode ser definida como a tecnologia por meio da qual um processo ou procedimento é alcançado sem assistencia humana (GROOVER, 2011).

Para consolidação da automação de determinado ambiente industrial deve-se ter em mente três etapas para condução deste processo de acordo com Groover (2011). A primeira etapa consiste na compreensão do processo. Nesta etapa, todos processos são analisados até os mínimos detalhes. Deve-se investigar a respeitos dos insumos, dos produtos, as etapas de processamento, a sequencia das etapas, a possibilidade de combinação de processos, dentre outros. Normalmente para um melhor entendimento do processo produtivo são elaborados diagramas denominado de diagrama de fluxo de processo, o qual especifica todos componentes da planta assim como a sequência do processo. Pode-se támbem utilizar modelos matématicos na descrição de um processo. Feita a investigação a respeito do processo, a segunda etapa consiste da simplificação do mesmo. Esta etapa tem por intenção eliminar todos os processos desnecessários e determinar quais processos podem ser unidos, quais equipamentos são mais adequados, considerar o processamento simultaneo, dentre outras razões. A última etapa, consiste da automação propriamente dita.

Vale ressaltar que dependendo das características do sistema produtivo, como vazão de produtos, variedade, tempo de processamento, pode-se optar por diferentes formas de automação, são essas: automação rígida, automação programável e automação flexível. A automação rígida é aquela na qual a sequência das operações é definida pela configuração dos equipamentos. Nesse tipo de automação normalmente cada equipamento desempenham funções relativamente simples e são extramente especializados, porém porcionam altas taxas de produção e um certa inflexibilidade quanto a variações no produto. A automação programável consiste de um sistema em que os equipamentos de produção podem modificar a sequencia das operações a fim de que produtos com diferentes características possam ser produzidos. Normalmente nesse tipo de automação a sequencia das operações é controlada através de um programa ou um conjunto de instruções. Devido as alterações que devem ser realizadas nas configurações dos equipamento, esse sistema de produção é capaz de produzir com taixas médias e baixas. Já a automação flexível consiste em um sistema de produção que

é previamente expecificado para produção de uma variedade de produtos. Ela trabalha com fato de que os produtos não possuem grandes diferenças entre si e é capaz de produzir em taxas intermediarias aos dois tipos citados anteriormente.

De forma geral pode-se ter a relação do tipo de automação que é necessário para cada tipo de demanda do processo produtivo, conforme mostrado na Fig. 2.13.

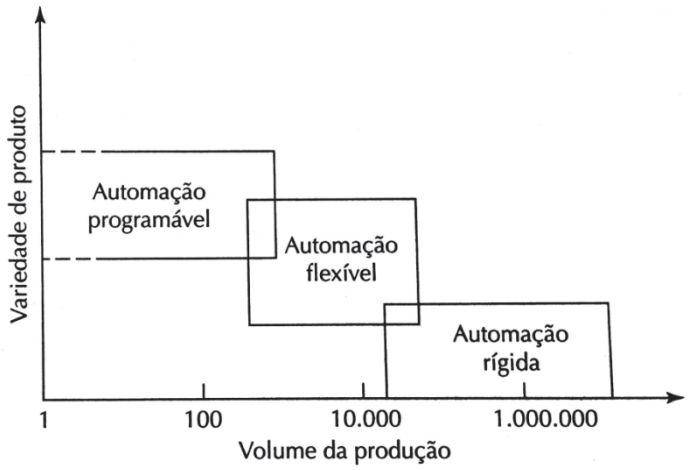


Figura 2.13 – Tipo de automação relativo a variedade e volume de produção (GROOVER, 2011).

Para este trabalho em particular o enfoque está na automação programável, uma vez que deseja-se atuar sobre o programa ou as instruções do sistema. Neste contexto o enfoque está justamente na rapidez com que são geradas essas rotinas no sentido de viabilizar o tempo de produção.

Na automação programável uma das tarefas mais importantes a ser desempenhada é o planejamento de processo. É essa função que determina os processos e as etapas envolvidas em um determinado objetivo. Assim a rotina anteriormente mencionado é resultado direto do planejamento de processos. O planejamento de processo, na maioria dos casos trata diretamente com os processos de manufatura, razão pela qual dentro da automação programável estão inclusas as máquinas de comando numérico computadorizado (CNC), mas também pode-se considerar a programação de robôs para manipulação e montagem de produtos e também a programação de controladores lógicos programaveis (CLP).

2.5. Planejamento de Processos

O planejamento de processos trata-se de uma atividade extremamente importante no meio industrial, visto que, é a através dele que são definidas como serão realizadas as etapas para fabricação de um determinado produto ou consolidação de um determinado processo. Dessa forma para se atingir certo objetivo deve-se dividir um processo de manufatura ou logístico em várias etapas, de forma que em cada uma seja realizada uma tarefa que contribua no avanço da matéria prima crua para o produto finalizado(para o caso de manufatura) ou para consolidação de atividades logísticas(processos de estocagem, montagem, desmontagem e transporte de produtos).

De forma mais específica, na manufatura de peças o trabalho do planejador de processos, consiste na interpretação das especificações contidas no projeto de determinada peça (dimensões e tolerâncias de forma e geometria), na seleção dos processos e as ferramentas possíveis para o processamento da peça, respeitando as restrições do projeto, na determinação de tolerâncias e características das máquinas-ferramenta, na seleção de superfícies de referência para execução das tarefas com precisão, além dos elementos de fixação, no sequenciamento das operações, levando em conta a prioridade dos processos e restrições tecnológicas e de precisão, dentre outras funções(HALEVI, 1995).

A abordagem atual para o planejamento de processos consiste na utilização de ferramentas que auxiliam no momento da escolha das etapas e características do processo. Para tanto são utilizados os CAPP's (*Computer Aided Process Planning*) que são basicamente programas que servem os planejadores de processos com informações e algoritmos para manufatura de peças. Deve-se ressaltar que um processo de planejamento totalmente automatizado ainda está longe de ser alcançado à exceção de processos que estejam minuciosamente definidos.

2.5.1. Planejamento de processos sob o ponto de vista econômico

No gerenciamento de uma empresa são necessárias incessantes decisões a respeito dos processos de produção, que são primariamente realizadas considerando os fatores econômicos e a obtenção de lucro. Sabe-se que para fabricação, montagem e distribuição de produtos são necessárias estratégias, algumas das quais feitas a nível tático pelo planejamento de processos. Se deseja-se implementar um novo produto deve-se então obter informações a respeito do mesmo para que o planejador de

processos possa, então, determinar as etapas para sua produção considerando os recursos disponíveis na empresa (mão de obra, maquinário, matéria-prima), as restrições intrínsecas do projeto, o tamanho do lote a ser fabricado, dentre outros. Em uma outra situação, se deseja-se determinar o investimento necessário para fabricação de determinado produto, recorre-se também ao planejador de processo, uma vez que este fica responsável por determinar as instalações básicas para fabricação do produto, minimizando custos e otimizando a implementação.

Dessa forma pode-se observar que o planejamento de processo é uma parte inerente do ambiente empresarial, devendo ser considerando em termos de tempo gasto em trabalho indireto. De acordo com Halevi e Weill (1995) o custo total de produção de uma peça pode ser modelado através da seguinte equação, para um processo de manufatura:

$$C = Q.C_d.T_d + C_t.T_p + C_s.T_s \tag{2.77}$$

Onde:

C = Custo total;

Q = Quantidade do lote;

C_d= Taxa horária de trabalho direto ($ por hora);

T_d= Tempo direto de usinagem (horas);

C_t= Taxa horária de trabalho indireto ($ por hora);

T_p = Tempo de trabalho indireto (horas);

C_s = Taxa horária de tempo de setup ($ por hora);

T_s= Tempo de setup;

Da equação anterior temos que T_p pode ser divido em duas partes:

T_{pf} – tempo fixo para lidar com um pedido e para chegar até um plano de processo inicial;

T_{pv} – tempo para criação de planos de processo alternativos e avaliação dos mesmos visando otimização.

T_{pv} corresponde a variável que contabiliza o tempo de raciocínio, sendo que quanto maior esse tempo, melhor será o plano gerando, porém maiores serão os custos associados. Desta forma devem-se levantar as vantagens e desvantagens do tempo de raciocínio considerando-se a intenção de se trabalhar com grandes ou pequenos lotes.

Considerando-se que o tempo de usinagem T_d é função de T_{pv}, pode-se enunciar a seguinte equação:

$$T_d = K_1 + \frac{K_2}{T_p} = K_1 + \frac{K_2}{T_{pf}+T_{pv}} \tag{2.78}$$

Onde K_1e K_2 são constantes que podem ser avaliadas por meio das seguintes condições de contorno:

Para $T_{pv} = \infty$

$$T_d = K_1 = T_{dmin} \tag{2.79}$$

Para $T_{pv} = 0$

$$T_d = K_1 + \frac{K_2}{T_{pf}} = T_{dmax} \tag{2.80}$$

Então:

$$K_2 = (T_{dmax} - T_{dmin}).T_{pf} \tag{2.81}$$

Dessa forma:

$$T_d = T_{dmin} + (T_{dmax} - T_{dmin}).T_{pf}/T_p \tag{2.82}$$

Substituindo na equação do custo:

$$C = Q.C_d.\left\{T_{dmin} + (T_{dmax} - T_{dmin}).\frac{T_{pf}}{T_p}\right\} + C_t.T_p + C_s.T_s \tag{2.83}$$

Derivando em relação a T_p e igualando a zero para encontrar o ponto de custo mínimo com relação ao tempo de raciocínio tem-se que:

$$\frac{dC}{dT_p} = -Q.C_d.\left\{(T_{dmax} - T_{dmin}).\frac{T_{pf}}{T_p^2}\right\} + C_t = 0 \tag{2.84}$$

Assim:

$$T_p^2 = Q.\frac{C_d}{C_t}.T_{pf}.(T_{dmax} - T_{dmin}) \quad (2.85)$$

Os valores das variáveis na equação anteriores são conhecidos e com exceção das variáveis de custo e o tamanho do lote, elas dependem da experiência do planejador de processo. Assim sendo, por meio do exemplo da manufatura de peças, pode-se observar a importância do planejamento de processo no resultado final dos custos de fabricação.

Na tentativa de contornar a limitação dos CAPPs com relação à possibilidade de planejamento para outras áreas que não sejam a manufatura, foi realizado um estudo a respeito das tecnologias de inteligência artificial que se dedicam a buscas orientadas para determinação sistemática de planos. Esta área corresponde ao planejamento automático que será descrito na seção a seguir.

2.6. Planejamento Automático

Planejar consiste na parte racional da ação. É um processo abstrato e explícito de deliberação que escolhe e organiza ações pela antecipação dos seus resultados esperados (GHALLAB; NAU; TRAVERSO, 2004). Nesse contexto o planejamento automático corresponde a area da inteligencia artificial que estuda esse processo de deliberação através da utilização de computadores.

A utilização do planejamento automático como tecnologia de inteligência artificial possui basicamente duas motivações. A primeira delas consiste do desenvolvimento de ferramentas que auxiliem no processo de planejamento, uma vez que no domínio industrial muitas vezes os planejadores de processos enfretam situações em que são necessarias mudanças nas tarefas de forma a atingir novos objetivos, tendo em vista uma determinada demanda. Dessa forma o planejamento automático tem por objetivo acelerar esse processo de análise e escolha de alternativas.

A outra motivação compreende uma abordagem mais teórica, referente ao estudo da inteligencia em si. Então em se desenvolvendo tais tecnologias, é possível se aproximar do que se trata da essência da inteligencia, permitindo dessa forma que se desenvolvam tecnologias que consigam abstrair de maneira mais completa esse conceito de inteligência.

Por fim pode-se dizer que desta tecnologia pressupõe-se a criação de equipamento e máquinas inteligentes que possam ter de alguma forma um comportamtento mais autônomo, daí a pertinência deste estudo com o tema do trabalho em questão.

Em meio aos vários tipos de situações e contextos que podem vir a se apresentar, seja em um ambiente industrial ou em até mesmo fatos cotidianos, existem variados tipos de ações e por consequencia varios tipos de planejamento. Ghallab; Nau; Traverso (2004) em seu livro, cita algumas dessas formas de planejamento, que são: Planejamento de trajetória e movimento, que está interessado nas formas geométricas de um determinado caminho que liga um ponto inicial a um ponto final; Planejamento de percepção, que está interessado na aquisição de informações através de sensoriamento para posterior planejamento; Planejamento de navegação que consiste da combinação dos dois tipos anteriores, no sentido de se obter informações com relação ao ambiente para obtenção de uma melhor trajetoria para exploração de determinada área; Planejamento de comunicação, que consiste da troca de informações entre os agentes de um determinado sistema; e por último o Planejamento de manipulação que está preocupado com a manipulação de objetos, cujas ações involvem força, torque, visão e alcance em ações como pegar um objeto e depositá-lo em locais predeterminados.

O enfoque deste trabalho está justamente no planejamento da manipulação, uma vez que a proposta consiste de um sistema de carregamento de produtos que não necessite de nenhuma interferência humana. Nesta abordagem utilizando o planejamento automático normalmente existem duas formas de planejamento: o planejamento de domínio específico e o planejamento de domínio independente. Em se utilizando o planejamento de domínio específico normalmente o que se faz é uma exploração das características específicas dos sistema no sentido de otimizar o processo de obtenção de resultados ou planos. Normalmente este tipo de planejamento possui excelentes resultados, porém esta tecnica é extremamente limitada, uma vez que, é realizada com o propósito de endereçar um problema em específico.

Na área do planejamento automático o enfoque maior está no estudo do planejamento em domínio independente. Nesta classe de planejamento são exploradas as características comuns a todo e qualquer processo de tomada de decisões, resultando em técnicas que podem ser aplicadas em quaisquer tipos de problemas. O planejamento em domínio independente, para o seu funcionamento, toma informações a respeito do domínio através de modelos que traduzem as especificidades de um determinado

problema. O seu objetivo geral consiste da síntese de planos, e apesar de ainda estar no seus estágios iniciais com relação ao embasamento teórico já está desenvolvido o bastante para ser utilizado em algumas aplicações.Um exemplo desta aplicações é o robô espacial Deep Space 1 (GHALLAB; NAU; TRAVERSO, 2004).

2.6.1. Modelo Geral

Conforme dito na seção anterior para a análise de um problema em específico em se utilizando o planejamento automático, deve-se obter informações do sistema que são normalmente representadas através de um modelo. De maneira geral um modelo de planejamento automático ou sistema de transição de estados por ser considerado um quarteto $\sum = (S, A, E, \gamma)$segundo Ghallab; Nau; Traverso(2004). Onde

- $S = \{s_1, s_2, \dots\}$ é um conjunto finito de estados;
- $A = \{a_1, a_2, \dots\}$ é um conjunto finito de ações;
- $E = \{e_1, e_2, \dots\}$ é um conjunto finito de eventos;
- $\gamma: S \times A \times E$ é a função de transição de estados.

O conjunto S representam todos os estados possiveis de acontecer em um determinado sistema. Os conjuntos A e E são responsáveis pela evolução do sistema, a diferenção entre eles está no fato de que o executor do plano tem controle sobre a ações possiveis do sistema e não sobre os eventos que podem ocorrer.γ define a interação entre os conjuntos mencionados anteriormente.

Assim o planejamento automático tem por objetivo determinar as ações assim como a ordem das mesmas para que um sistema parta de um determinado estado inicial $S_i$para atingir um determinado objetivoS_f ou um conjunto de objetivos$Sf = \{s_4, s_5 \dots\}$. Para esse conjunto de ações normamente dá-se o nome de plano. Para um melhor entendimento da função do planejamento automático a Fig. 2.14 mostra a interação entre os componentes do sistema e o planejador.

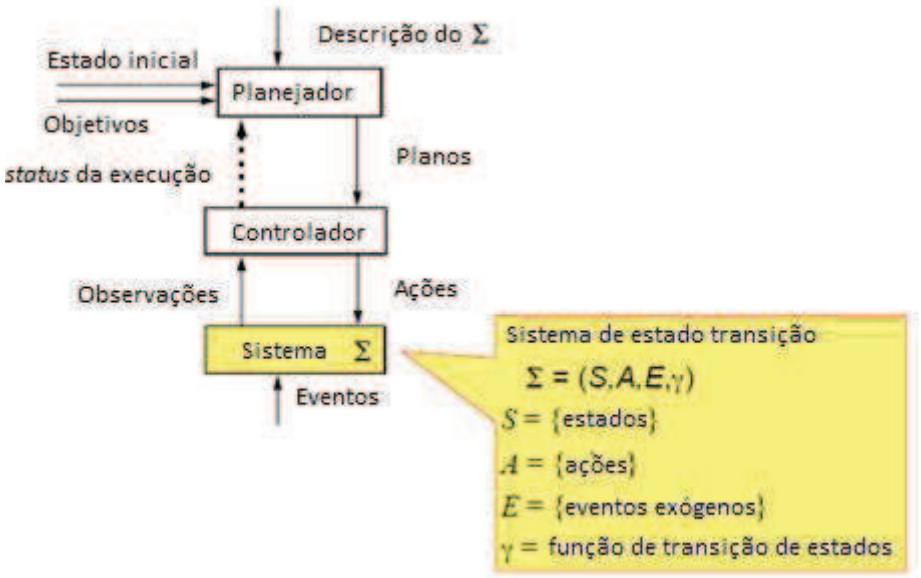


Figura 2.14 – Modelo conceitual de planejamento.(adaptado de Ghallab; Nau; Traverso (2004)).

Na figura anterior observa-se que o planejador obtém as informações a respeito do sistema juntamente com um estado inicial e os objetivos. Tendo esta informação o planejador elabora um plano que é então enviado ao controlador do sistema, e este por sua vez é responsável por conduzir a execucação das ações e verificar se as mesmas foram executadas. Por último tem-se um retorno do controlador ao planejador que fornece informações a respeito do andamento do plano para o caso em que seja necessário reformular a estratégia em execução, vale ressaltar que esta última interação é valida apenas quando o planejamento em feito em tempo de execução ou *online.*

Para utilização do paradigma de planejamento automático mostrado anteriormente são necessárias algumas restrições para que o sistema seja funcional, são estas: sistema finito, ou seja, deve possuir um conjunto de estados finitos; sistema totalmente observável, em que um observador tenha informação de tudo ocorre no domínio de $\sum$; sistema determinístico, ou seja, para cada estado s e ação a existe apenas um resultado possível para a interação $\gamma(s,a)$; sistema estático, ou seja, o sistema permanece no estado em que o controlador o colocou sem a possibilidade de haver alguma dinâmica interna de $\sum$; objetivos restritos, o problema deve apenas tratar de objetivos restritos que são descritos nos estados finais; planos sequenciais, a resposta do planejador é um lista de finita e linear de ações; tempo implícito, para esta hipótese ações e eventos não possuem duração de tempo; e por último planejamento *offline*, o

planejador não considera nenhuma alteração no sistema enquanto está no processo de planejamento.

2.6.2. Planejamento Clássico

O planejamento clássico é aquele que endereça problemas que respeitam todas as restrições citadas anteriormente. Um problema de planejamento clássico é definido por um trio $P = (\sum, s_0, g)$, em que $\sum$ representa o sistema, conforme notação apresentada anteriormente, s_0 representa o estado inicial do sistema e por último g corresponde a um conjunto de objetivos.

O estudo do planejamento clássico tem uma de suas justificativas na sua forma de abordagem. Usualmente em problemas de engenharia, em se tratando de problemas muito complexos é necessário um abstração simplificada dos mesmos, de tal forma que um modelo reduzido normalmente é utilizado na obtenção da solução. O planejamento clássico, por respeitar as restrições apresentadas na seção anterior, possui uma base consolidada para investigação dos mais diversos domínios e problemas não somente da área de engenharia, mas também com relação a procedimentos logísticos, dentre outros. Além disso, o planejamento clássico abre um leque de oportunidades para o desenvolvimento de novas técnicas que venham a contribuir para o conhecimento da área de planejamento, sempre buscando novas maneiras de tornar os domínios de planejamento mais realistas e completos.

Um desafio para utilização do paradigma do planejamento clássico está na forma de se traduzir as informações do sistema sem que sejam descritos quaisquer procedimentos que caracterizem um problema em expecifico, além disso, como investigar e quais algoritimos utilizar são outras das dificuldades desta metodologia. Assim ainda no intuito de se representar um problema real para um algoritmo de planejamento deve-se utilizar um representação através da qual pode-se então inferir o comportamento do sistema em questão. Segundo Ghallab; Nau; Traverso (2004), existem três forma de se representar os problemas de planejamento clássico: a representação de conjuntos teóricos, na qual o estado do ambiente em abordagem é representado através de um conjunto de proposições e as ações são expressões sintáticas que estabelecem condições dos estados, para que as ações sejam executadas, assim

como as proposições que serão removidas e adicionadas devido execução da própria ação; a representação de estado variável, na qual cada estado é representado por um conjunto de valores relacionados referentes as variáveis de estado e cada ação é representada por um função que mapeia esse conjunto de valores relacionados com outros conjuntos possíveis; por último a representação clássica, em que as ações e estados são mapeados de forma semelhante a representação de conjuntos teóricos exceto pelo fato de que as conexões são representadas por meio de literais de primeira ordem e conexões lógicas ao invés de proposições. Este tipo de modelagem é a mais utilizada para a respresentação de sistemas de transição de estado e por sua vez é a de interesse deste trabalho.

2.6.3. Representação Clássica

Conforme dito anteriormente a representação clássica estende a representação de conjuntos teóricos no sentido de utilizar uma notação que é proveniente da lógica de primeira ordem.

Na lógica de primeira ordem o ambiente é considerado como sendo um conjunto de objetos que possuem relações válidas e não válidas entre si. Na representação clássica para a formulação dos estados são necessários apenas predicados e constantes. Dessa forma um estado é definido com sendo um conjunto de sentenças atômicas, que são formadas por um predicado seguido de por um conjunto de constantes entre parênteses. Para ilustrar o conceito anterior Ghallab; Nau; Traverso (2004) mostram através de um domínio de robôs de doca como são formulados os estados através da representação clássica.

Seja um domínio com dois locais (*loc1, loc2*), um robô (*r1*), um guindaste (*crane1*), duas pilhas de produtos (*p1*, *p2*) e três contêineres (*c1, c2, c3*) e um palete (*pallet*, que é inserido debaixo de cada pilha). Seja um estado deste domínio mostrado na Fig. 2.15.

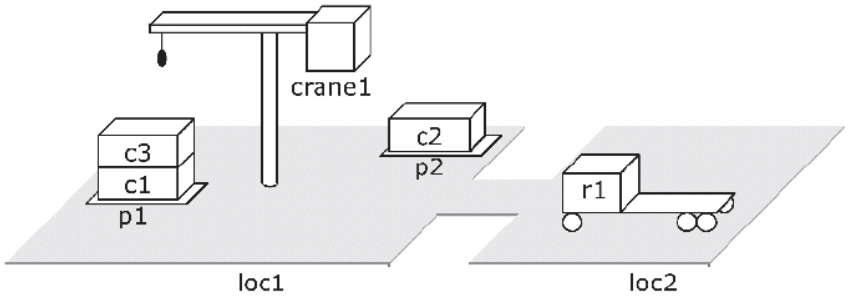


Figura 2.15 – Estado qualquer do domínio de robôs de doca (GHALLAB; NAU; TRAVERSO, 2004).

Utilizando a representação clássica o estado anterior pode ser traduzido através da seguinte expressão: *s1*={*attached*(p1,loc1), *attached*(p2,loc1), *in*(c1,p1), *in*(c3,p1), *top*(c3,p1), *on*(c3,c1), *on*(c1,pallet), *in*(c2,p2), *top*(c2,p2), *on*(c2,pallet), *belong*(crane1,loc1), *empty*(crane1), *adjacent*(loc1,loc2), *adjacent*(loc2,loc1), *at*(r1,loc2), *occupied*(loc2), *unloaded*(r1)}

Vale ressaltar que para cada um dos estados do sistema, as sentenças atômicas podem ou não serem verdadeiras, o que implica que as mesmas variam conforme a modificação do sistema.

Para a tradução das ações a representação clássica faz uso de operadores de planejamento. No planejamento clássico, um operador de planejamento consiste de um trio o= (*name* (o), *precond* (o), *effects* (o)), em que *name* (o) corresponde ao nome do operador, *precond* (o) corresponde às precondições para execução da ação e *effects* (o) corresponde aos efeitos causados pela execução da ação.

Ainda no exemplo mostrado anteriormente, são mostrados alguns exemplos de ações:

Tabela 2.2 – Exemplos de ações.

move(r, l, m) ,, robô r move de um local l para um local adjacente m precond: adjacent(l, m), at(r, l), ¬ occupied(m) effects: at(r, m), occupied(m), ¬ occupied(l), ¬ at(r, l)
load(k, l, c, r) ;; gindaste k no local l carrega o conteiner c no robô r precond: belong(k, l), holding(k, c), at(r, l), unloaded(r) effects: empty(k),¬ holding(k, c), loaded(r, c), ¬ unloaded(r)
unload(k, l, c, r) ,, guindaste k no local lpega o conteiner c do robô r precond: belong(k, l), at(r, l), loaded(r, c), empty(k)

effects: ¬ empty(k), holding(k, c), unloaded(r), ¬ loaded(r, c)

¬ corresponde ao operador de negação;

Conforme o exposto, esta corresponde à modelagem utilizando a representação clássica. Vale ressaltar que nesta técnica são permitidos apenas operadores que utilizam a lógica de primeira ordem, mas muitas vezes são necessárias algumas extensões a essa lógica, o que acaba por torná-la uma técnica extremamente limitada.

Uma das formas de extensão ao modelo de restrições clássica consiste da utilização de variáveis e restrições tipificadas. Assim o problema não se trata apenas da descrição do sistema, do estado e inicial e dos objetivos, mas também de um conjunto de constantes (robôs, locais, *pallets*, dentre outros.). Uma das linguagens que implementa esta forma de representação é a PDDL (*Planning Domain Definition Language*).

2.6.4. Planning Domain Definition Language

A linguagem PDDL foi apresentada pela primeira vez em 1998, o objetivo do desenvolvimento desta linguagem era servir como base para especificação dos problemas de planejamento da competição AIPS-98. Segundo, *Yale Center for Computational Vision and Control* (1998), a linguagem PDDL destina-se a expressar a física de um domínio, ou seja, quais predicados existem, quais ações são possíveis, qual a estrutura das ações compostas, quais são os efeitos das ações. Vale ressaltar que a mesma já passou por várias modificações desde a sua apresentação em 1998, há esse tempo foi apresentada a PDDL 1.2 e nos dias atuas já se tem a versão 3.0.

A linguagem PDDL, devido a suas características teve grande repercussão no momento por incorporar uma série de possibilidades de sintaxes para representação dos domínios. São estas:

- Ações básicas definidas ao estilo do STRIPS (FIKES, R. E.; NILSSON, N. J., 1971);
- Efeitos condicionais;
- Quantificação universal em universos dinâmicos (por exemplo, criação e destruição de objetos);

- Axiomas de domínio sobre teorias estratificadas;
- Especificação de restrições de segurança;
- Especificação de ações hierárquicas compostas de sub-ações e sub-objetivos;
- Gerenciamento de múltiplos problemas em múltiplos domínios utilizando deferentes subconjuntos de recursos da linguagem (para suportar o compartilhamento de ações para diferentes planejadores que lidam com níveis variáveis de expressividade).

Em se utilizando deste formalismo para representação dos problemas de planejamento clássico tem-se a prática da divisão da informação em dois arquivos distintos: o domínio e o problema. No primeiro são descritas todas as informações a respeito do sistema, as classes, os atributos dessas classes, as ações, enfim a dinâmica do modelo representado. No segundo são descritas as condições para aplicação do arquivo anterior, ou seja, são especificadas ambas as condições iniciais e estados desejáveis do sistema.

É importante destacar que este trabalho não tem por objetivo descrever toda a linguagem PDDL mais ainda serão descritos algumas das características dessa linguagem.

Inicialmente a linguagem PDDL, para o correto funcionamento dos algoritmos de planejamento, introduz o conceito de requerimentos ou *requirements*. Os *requirements* traduzem os requisitos que são necessários para avaliação de um determinado domínio de planejamento. Desta forma, caso algum planejador não seja capaz de lidar com certo tipo de requerimento, o mesmo já desiste do processo de investigação.

Para declaração de um novo domínio faz-se utilização da seguinte notação:

Tabela 2.3 – Notação para definição do domínio.

```
<domain> ::= (define (domain <name>)
                [<extension-def>]
                [<require-def>]
                [<types-def>]:typing
                [<constants-def>]
                [<domain-vars-def>]:expression-evaluation
                [<predicates-def>]
                [<timeless-def>]
                [<safety-def>]:safety-constraints
                <structure-def>_)
```

Segue que da notação anterior à cláusula *extention* implementa a herança de outros domínios. A cláusula *require* implementa os requisito citados anteriormente. O *types* diz respeito às entidades ou classes de objetos que farão parte do domínio de planejamento, já a cláusula *constants* representam as instâncias das classes enunciadas em *types.* A cláusula *domain vars* é utilizada apenas quando o *requirement* do tipo *expression-evaluation* é utilizado (os *requirements* serão apresentados a fim da abordagem sobre PDDL). A cláusula *predicates*, enuncia os predicados do domínio, que são expressões a respeito do comportamento do sistema que tem um comportamento booleano (*true/false*). A cláusula *timeless* representa a enumeração de todos os literais que devem permanecer verdadeiros por todo o domínio. As outras duas cláusulas definem as funções e o comportamento do sistema que são descritas com suas notações.

Yale Center for Computational Vision and Control (1998) faz uso do exemplo do transporte de objetos de casa para o trabalho para exemplificar a notação apresentada.

Tabela 2.4 – Exemplo do transporte de objetos.

```
(define (domain briefcase-world)
(:requirements :strips :equality :typing :conditional-effects)
(:types location physob)
(:constants (B - physob))
(:predicates (at ?x - physob ?l - location)
(in ?x ?y - physob)   ...
```

É importante ressaltar que uma das importantes cláusulas da representação do domínio foi introduzida apenas na versão 2.1 da linguagem PDDL, que é a cláusula *function*. A cláusula *function* introduz valores numéricos para contabilização de quantidade de recursos ou características variáveis do problema. Fox e Long (2003) mostra um exemplo da distribuição de água em garrafas.

Tabela 2.5 – Exemplo de distribuição de água em garrafas.

```
(define (domain jug-pouring)
(:requirements :typing :fluents)
(:types jug)
(:functions
   (amount ?j - jug)
(capacity ?j - jug))
```

Seguindo com a representação são apresentadas as ações. De acordo com esta metodologia as ações são representadas da seguinte forma:

Tabela 2.6 – Notação para definição de ações.

```
<action-def>          ::= (:action <action functor>
                          :parameters ( <typed list (variable)> )
          <action-def body>)

<action functor>      ::= <name>
<action-def body>     ::= [:vars (<typed list(variable)>)]
                                                  :existential-preconditions
                                                  :conditional-effects
                          [:precondition <GD>]
                          [:expansion
                                  <action spec>]:action-expansions
                          [:expansion :methods]:action□expansions
                          [:maintain <GD>]:action□expansions
                          [ :effect <effect>]
                          [:only-in-expansions <boolean>]:action□expansions
```

Desta representação tem-se que a cláusula *action functor* representa o nome da ação, que corresponde a uma abreviação responsável por abstrair todas as condições representadas nas outras cláusulas da ação. A cláusula *parameters* lista todas as variáveis que serão utilizadas pela ação. A cláusula *action def body* contém a descrição da ação e pode ser fracionada em algumas partes: a cláusula *vars* representa variáveis que possuem alguma limitação existencial no domínio de tal forma que somente podem ser alteradas por meios de funções. A cláusula *precondition* lista todas as condições que devem ser satisfeitas para que uma determinada ação seja de fato efetuada. A cláusula *expansion* está condicionada ao *requirement* do tipo *action-expansions* e define a maneira pela qual uma ação de deve se comportar em termos de expressões simplificadas com relação aos efeitos sobre o sistema. Daí decorre que uma ação pode possuir apenas a cláusula *expansion* ou a cláusula *effects* sendo que as duas simultaneamente não é possível. Desta forma a cláusula *effects* lista todas as modificações que devem ocorrer no sistema em virtude da ação.

Novamente *Yale Center For Computational Vision And Control* (1998) faz utilização de um exemplo para exemplificação de sua notação:

Tabela 2.7 – Exemplo da ação pintura.

```
(:action spray-paint
   :parameters (?c - color)
   :vars (?x - location)
```

```
    :precondition (at robot ?x)
    :effect (forall (?y - physob)
(when (at ?y ?x)
(color ?y ?c))))
```

Neste exemplo, o autor modela a ação pintura, de tal forma que no contexto da função existe o parâmetro cor e a variável lugar. As precondições para esta ação ser executada corresponde ao robô estar no lugar determinado. Já os efeitos dessa ação dizem que todos os objetos situados no local determinado tem a sua cor substituída pela cor do contexto da função.

Para representação do problema a ser abordado através do domínio já apresentado, a notação da PDDL faz uso dos conceitos de estado inicial e objetivos. Na versão 1.2 da PDDL a notação para a definição de um problema de planejamento é da seguinte forma:

Tabela 2.8 – Notação para definição do problema segundo a PDDL 1.2.

```
<problem>              ::= (define (problem <name>)
                               (:domain <name>)
                               [<require-def>]
                               [<situation> ]
                               [<object declaration> ]
                               [<init>]
                               <goal>+
                               [<length-spec> ])
<situation>            ::= (:situation <initsit name>)
<object declaration>   ::= (:objects <typed list (name)>)
<init>                 ::= (:init <literal(name)>+)
<initsit name>         ::= <name>
<goal>                 ::= (:goal <GD>)
<goal>                 ::=:action-expansions
(:expansion <action spec(action-term)>)
<length-spec>          ::= (:length [(:serial <integer>)] [(:parallel <integer>)])
<initsit def>          ::= (define (situation <initsit name>)
                               (:domain <name>)
                               [ <object declaration> ]
                               [ <init> ])
```

Porém já na versão 2.1 a ideia de uma declaração a parte da situação inicial foi eliminada e, além disso, foi adicionada a ideia de métrica para guiar o processo de investigação resultando na arquitetura mostrada abaixo que é basicamente a utilizada atualmente.

Tabela 2.9 – Notação para definição do problema segundo a PDDL 2.1

```
<problem>                ::= (define (problem <name>)
                                 (:domain <name>)
                                 [<require-def>]
                                 [<object declaration> ]
                                 <init>
                                 <goal>
                                 [<metric-spec>]
                                 [<length-spec> ])
<object declaration>     ::= (:objects <typed list (name)>)
<init>                   ::= (:init <init-el>*)
<init-el>                ::= <literal(name)>
<init-el>                ::=:fluents (= <f-head><number>)
<goal>           ::= (:goal <GD>)
<metric-spec>    ::= (:metric <optimization><ground-f-exp>)
<optimization> ::= minimize
<optimization> ::= maximize
<ground-f-exp>::= (<binary-op><ground-f-exp><ground-f-exp>)
<ground-f-exp>::= (- <ground-f-exp>)
<ground-f-exp>::= <number>
<ground-f-exp>::= (<function-symbol><name>* )
<ground-f-exp>::= total-time
<ground-f-exp>::= <function-symbol>
<length-spec>    ::= (:length [(:serial <integer>)]
                          [(:parallel <integer>)])
                          The length-spec is deprecated.
```

Para ilustrar a sintaxe apresentada Fox e Long (2003) utilizam-se de alguns exemplos como o mostrado a seguir:

Tabela 2.10 – Exemplo de um problema de planejamento.

```
(define (problem metricVehicle-example)
  (:domain metricVehicle)
  (:objects
       truck car - vehicle
       Paris Berlin Rome Madrid - location)
  (:init
       (at truck Rome)
       (at car Paris)
       (= (fuel-level truck) 100)
       (= (fuel-level car) 100)
       (accessible car Paris Berlin)
       (accessible car Berlin Rome)
       (accessible car Rome Madrid)
       (accessible truck Rome Paris)
       (accessible truck Rome Berlin)
       (accessible truck Berlin Paris)
       (= (fuel-required Paris Berlin) 40)
       (= (fuel-required Berlin Rome) 30)
       (= (fuel-required Rome Madrid) 50)
       (= (fuel-required Rome Paris) 35)
       (= (fuel-required Rome Berlin) 40)
```

```
      (= (fuel-required Berlin Paris) 40)
      (= (total-fuel-used) 0)
      (= (fuel-used car) 0)
      (= (fuel-used truck) 0)
 )
 (:goal (and (at truck Paris)
             (at car Rome))
)
 (:metric minimize (total-fuel-used))
)
```

O exemplo apresentado por Fox e Long (2003), define os veículos do tipo carro e caminhão, e as cidades, Paris, Berlim, Roma e Madrid. A situação inicial do problema diz que o caminhão está em Roma, o carro está em Paris e ambos os veículos estão com o nível de combustível em 100%. Para o problema em questão as viagens possíveis são definidas através do operador *accessible*, que informa o veiculo, o lugar de início e destino, respectivamente. Além disso, define a quantidade necessária de combustível para cada viagem e o valor do combustível consumido no estado inicial. O estado final dita que o caminhão deve estar em Paris e o carro em Roma, e este processo deve ser realizado com a menor quantidade de combustível possível.

No paradigma da PDDL existem ainda existem vários outros componentes, como a expansão de ações, os axiomas, as restrições de segurança que por sua vez não serão aqui descritos, pois não cabe a este trabalho detalhar a fundo sobre o assunto. Mas por último deve-se apresentar os *requirements* cuja importância serve para adequação dos planejadores aos domínios. Na divulgação original da PDDL 1.2 a lista dos *requirements* consistia dos seguintes itens.

Tabela 2.11 – Lista de *requirements* segundo a PDDL 1.2.

:strips	Basic STRIPS-style adds and deletes
:typing	Allow type names in declarations of variables
:disjunctive-preconditions	Allow or in goal descriptions
:equality	Support = as built-in predicate
:existential-preconditions	Allow exists in goal descriptions
:universal-preconditions	Allow forall in goal descriptions
:quantified-preconditions	= :existential-preconditions + :universal-preconditions
:conditional-effects	Allow when in action e_ects
:action-expansions	Allow actions to have :expansions
:foreach-expansions	Allow actions expansions to use foreach (implies :action-expansions)
:dag-expansions	Allow labeled subactions (implies :action-expansions)
:domain-axioms	Allow domains to have :axioms

:subgoal-through-axioms	Given axioms p⊃q and goal q, generate subgoal p
:safety-constraints	Allow :safety conditions for a domain
:expression-evaluation	Support eval predicate in axioms (implies :domain-axioms)
:fluents	Support type (fluent t). Implies :expression-evaluation
:open-world	Don't make the “closed-world assumption" for all predicates - i.e., if an atomic formula is not known to be true, it is not necessarily assumed false
:true-negation	Don't handle not using negation as failure, but treat it as in _rst-order logic (implies :open-world)
:adl	= :strips + :typing + :disjunctive-preconditions + :equality + :quantified-preconditions + :conditional-effects
:ucpop	= :adl + :domain-axioms + :safety-constraints

Com a modificação realização na versão 2.1 foram adicionados alguns *requirements* a este rol. São estes:

Tabela 2.12 – *Requirements* adicionados na versão 2.1.

:negative-preconditions	Allow not in goal descriptions
:durative-actions	Allows durative actions. Note that this does not imply :fluents.
:duration-inequalities	Allows duration constraints in durative actions using inequalities.
:continuous-effects	Allows durative actions to affect fluents continuously over the duration of the actions.

Para maiores detalhes a respeito de cada um dos requirements favor consultar os documentos originais da apresentação da linguagem PDDL (YALE CENTER FOR COMPUTATIONAL VISION AND CONTROL, 1998; FOX; LONG, 2003; EDELKAMP; HOFFMAN, 2004; DEPARTMENT OF ELECTRONICS FOR AUTOMATION, 2005).

De maneira geral, cada uma das versões posteriores da PDDL 1.2 (GHALLAB et al. 1998) tiveram como objetivo o acréscimo de algumas características que foram julgadas necessárias com o passar do tempo e utilização da mesma. Assim a PDDL 2.1 (FOX; LONG, 2003) introduziu os conceitos de métrica e ações durativas no tempo, já a PDDL 2.2 (EDELKAMP; HOFFMAN, 2004) introduziu o conceito de predicados derivados, que substituiu os axiomas apresentados na sintaxe inicial, e também a ideia de literais iniciais temporizados. A PDDL 3.0 (DEPARTMENT OF ELECTRONICS

FOR AUTOMATION, 2005), por sua vez, apresenta os conceitos de restrições fortes e fracos e objetivos fortes e fracos e por último uma versão da PDDL 3.1 está sendo atualmente formulada.

Para a resolução dos problemas gerados através desta linguagem existem uma série de planejadores especificamente desenvolvidos com este propósito. O principal congresso na área de planejamento automático, que trata-se do ICAPS (*International Conference on Automated Planning and Scheduling*), realiza tradicionalmente uma competição denominada IPC (*International Planning Competion*) cujo objetivo é avaliar o desempenho de diversos planejadores em algumas categorias, são elas: a sequência determinística (*deterministic track*), a sequência de aprendizado (*learning track*) e a sequência incerta (*uncertaintly track*) (Informações baseadas no ICAPS 2011). Das categorias citadas vale ressaltar a sequência determinística que considera um domínio totalmente observável e determinístico, que destaca o planejamento clássico abordado neste trabalho. Nesta classe de problemas a avaliação do desempenho pode ser feita através do tempo de processamento (*temporal*), o valor de custo mínimo obtido (*sequencial*) e a combinação dos dois anteriores (*preferences*). Ao longo dos anos esta competição teve vários vencedores: Fast downward ss1 e Selective Max (IPC 2011); Gamer e Lama (IPC 2008); Maxplan, Satplan, SGPlan e MIPS XXL (IPC 2006) e outros mais.

CAPÍTULO III

PROPOSTA DO PROJETO

A proposta desse projeto consiste do desenvolvimento de um sistema que integra as técnicas de solução do problema de carregamento de contêineres juntamente com a tecnologia de planejamento automático. De acordo com o apresentado, a ideia básica consiste em se utilizar as técnicas da programação linear, para elaboração de um padrão de empacotamento, do qual pode se retirar as informações com relação aos estados iniciais e finais a serem considerados no ambiente de planejamento automático. Através do Planejamento Automático pode-se então determinar as ações para que um sistema automatizado seja capaz de montar o padrão de empacotamento conforme especificado pela programação linear. A proposta inicial do trabalho envolvia o desenvolvimento de um conversor de planos e de uma bancada didática cujo controlador seja capaz de seguir as informações providas pelo conversor, que por sua vez recebe as informações do planejador. Para uma melhor visualização do sistema proposto foi elaborada a seguinte arquitetura.

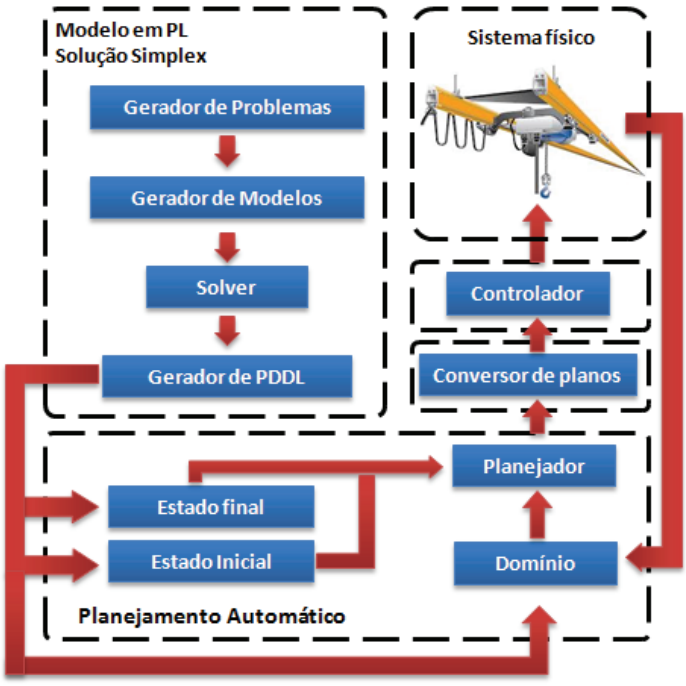


Figura 3.1 – Arquitetura do sistema automático.

Nas seções seguintes serão detalhadas cada uma das partes abordadas na referida arquitetura.

3.1. Gerador de Problemas

O Gerador de Problemas consiste de uma função capaz de adquirir todas as informações a respeito do problema. Em específico deseja-se obter as seguintes características do problema:

- Largura do contêiner
- Comprimento do contêiner
- Altura do contêiner
- Número de tipos de caixa
- Quantidade de caixas de cada tipo
- Largura de cada tipo de caixa
- Comprimento de cada tipo de Caixa
- Altura de cada tipo de caixa

Esta função deve ser capaz de determinar todas as rotações possíveis das caixas, além de condicionar os dados de forma estruturada para fácil manipulação das funções posteriores.

3.2. Gerador de Modelos

Esta função é responsável por transformar toda informação obtida com a função anterior em um modelo matemático, que será otimizado em se utilizando a metodologia do Simplex. Para a construção dessa função foi avaliado uma série de artigos (SCHEITHAUER, 1999; PISINGER; SIGURD, 2005; LIM ET AL. 2013; HADJICONSTANTINOU; CHRISTOFIDES,1995; JUNQUEIRA, 2009), cujos

modelos foram estudados no sentido da determinação do adequado ao problema. A seguir é mostrada uma breve revisão dos modelos estudados:

3.2.1. Scheithauer (1999)

Em seu trabalho Scheithauer (1999) descreve uma metodologia para a solução do problema de carregamento de contêineres em se utilizando o conceito de barras unitárias. Neste problema o formalismo do problema de carregamento de contêineres é descrito da seguinte maneira:

Seja L, W e H as variáveis correspondentes a comprimento, largura e altura do contêiner, respectivamente. Nessa abordagem considera-se um número m de tipos de caixas denotados por uma variável t e seja $T = \{1, \dots, m\}$. Cada caixa possui um volume denotado por v_t, um valor associado g_t (sendo que normalmente $g_t = v_t$) e uma quantidade máxima disponível u_t. O modelo de Scheithauer (1999) utiliza um vetorI_t para representar as rotações possíveis das caixas. Seja l_i, w_i e h_i o comprimento, a altura e largura de um caixa com uma determinada orientação i para $i \in I_t$. Para este problema considera-se o padrão de empacotamento possa ser definido através de um conjunto de barras de unidimensionais com determinado comprimento e seção transversal 1×1 sendo estas na direção L (comprimento), W (Largura) e H (altura). Uma barra unidimensional L é representada através de um vetor não negativo inteiro $a^p = (a_{1p}, \dots, a_{\tilde{m}p})$, onde a_{ip} representa o número de vezes que uma parte $1/w_i h_i$ da caixa i é contida pela barra p. Da mesma forma as notações b_{iq} e c_{ir} representam as barras W e H, respectivamente. A quantidade de cada barra que é utilizada no padrão de empacotamento é denotada pelas variáveis inteiras e não negativas x_p, y_q e z_r. A formulação do problema através da denominada relaxação de barra é mostrada a seguir:

$$\eta = \sum_{p \in P} \sum_{t \in T} \sum_{i \in I_t} \frac{g_t}{v_t} l_i a_{ip} x_p \rightarrow max \tag{3.1}$$

$sujeito\ a$

$$\sum_{p \in P} l_i a_{ip} x_p - \sum_{q \in Q} w_i b_{iq} y_q = 0, i \in \bar{T} \tag{3.2}$$

$$\sum_{p \in P} l_i a_{ip} x_p - \sum_{q \in Q} h_i c_{ir} z_r = 0, i \in \bar{T} \tag{3.3}$$

$$\sum_{p\in P}\sum_{i\in I_t} l_i a_{ip} x_p = u_t v_t, t \in T_1 \quad (3.4)$$

$$\sum_{p\in P}\sum_{i\in I_t} l_i a_{ip} x_p \geq u_t v_t, t \in T_2 \quad (3.5)$$

$$\sum_{p\in P}\sum_{i\in I_t} l_i a_{ip} x_p \leq u_t v_t, t \in T_3 \quad (3.6)$$

$$\sum_{p\in P} x_p \leq WH, \sum_{q\in Q} y_q \leq LH, \sum_{r\in R} z_r \leq LW \quad (3.7)$$

$$x_p \geq 0, p \in P, y_q \geq 0, q \in Q, z_r \geq 0, r \in R \quad (3.8)$$

Neste modelo a função objetivo visa à contagem do valor das caixas empacotadas, sendo que este valor pode ser o volume propriamente. A primeira restrição diz respeito à utilização do volume que de acordo com a modelagem proposta deve-se a mesma tanto nas direções de comprimento e largura. De maneira similar a segunda equação verifica a utilização do volume das caixas para as direções de comprimento e altura. Da terceira equação a quinta, verifica-se uma condição particular da modelagem proposta que se trata da quantidade mínima e máxima de cada tipo de caixa que deve ser empacotada. As variáveis x_p, y_q e z_r correspondem a número de vez que os padrões nas direções L, W e H foram utilizados, respectivamente.

Ainda em seu trabalho Scheithauer (1999), faz uso de uma modelagem relaxada com base em faixas cujo resultado é semelhante ao apresentado anteriormente.

3.2.2. Pisinger e Sigurd (2005)

No trabalho de Pisinger e Sigurd (2005) é abordado o problema de carregamento de caixas. Sua abordagem é um pouco diferenciada uma vez que pressupõe a alocação de produtos retangulares dentro de caixas. Este problema é visto sob a ótica de duas dimensões, porém, sua modelagem e notações podem ser facilmente estendidas para um problema tridimensional.

Neste trabalho o problema é formulado como sendo um conjunto de n tipos de objetos retangulares e m tipos de caixas, sendo os objetos de largura w_i e altura h_i e as caixas de largura W_k e altura H_k. Cada caixa possui um custo c_k associado. Além disso, são introduzidos alguns operadores binários que indicam a posição relativa dos itens, são eles: l_{ij} que é 1 se o item i está localizado a esquerda de j; b_{ij} que é 1 se o

item $i$está localizado abaixo do item j; f_{ik} que é 1 se o item i está alocado na caixa k; z_k que é 1 se a caixa k estiver sendo utilizada. A coordenadas dos itens por sua vez são definidas pelas variáveis x_i e y_i e corresponde ao canto inferior esquerdo de um item.

No sentido de minimizar o custo (c_k) das caixas utilizadas é formulado o seguinte problema de PL.

$$minimizar \sum_{k=1}^{m'} c_k z_k \tag{3.9}$$

$sujeito\ a$

$$l_{ij} + l_{ji} + b_{ij} + b_{ji} + (1 - f_{ik}) + \left(1 - f_{jk}\right) \geq 1, \quad i,j \in R, i < j, k \in \{1, \dots, m'\} \tag{3.10}$$

$$x_i - x_j + W l_{ij} \leq W - w_i, \quad i,j \in R \tag{3.11}$$

$$y_i - y_j + H b_{ij} \leq H - h_i, \quad i,j \in R \tag{3.12}$$

$$x_i \leq W_k - w_i + (1 - f_{ik})W, \quad i \in R, k \in \{1, \dots, m'\} \tag{3.13}$$

$$y_i \leq H_k - h_i + (1 - f_{ik})H, \quad i \in R, k \in \{1, \dots, m'\} \tag{3.14}$$

$$\sum_{k=1}^{m'} f_{ik} \geq 1, \quad i \in R \tag{3.15}$$

$$f_{ik} \leq z_k, \quad i \in R, k \in \{1, \dots, m'\} \tag{3.16}$$

$$l_{ij}, b_{ij} \in \{0,1\}, \quad i,j \in R \tag{3.17}$$

$$f_{ik} \in \{0,1\}, \quad i \in R, k \in \{1, \dots, m'\} \tag{3.18}$$

$$x_i, y_i \geq 0, \quad i \in R \tag{3.19}$$

$$z_k \in \{0,1\}, \quad k \in \{1, \dots, m'\} \tag{3.20}$$

Onde W e H correspondem aos maiores valores de largura e altura das caixas, respectivamente, e m' corresponde ao número de tipos de caixas.

Nesse problema a inequação (3.10) garante que dois itens não se sobreponham no espaço. As inequações (3.11) a (3.14) garantem as relações dimensionais dos itens para com as caixas. A inequação (3.15) diz que um item deve estar em 1 caixa e a inequação (3.16) diz respeito à alocação de produtos em caixas que estão sendo utilizadas. As demais expressões apresentam o domínio das variáveis utilizadas no problema.

3.2.3. Lim et al.(2013)

Neste trabalho é a apresentada uma proposta para resolução do problema de carregamento de contêiner em se considerando restrições de peso por eixo no caso de transporte por caminhões. Sua abordagem trata-se de um híbrido do algoritmo para construção de paredes juntamente com a modelagem em programação linear. A programação linear atua no sentido de adequar as soluções obtidas através do procedimento de construção de paredes. Para os casos em que o limite de peso por eixo do caminhão é excedido o algoritmo usa da programação linear para o rearranje das paredes ou a retirada de caixas.

A modelagem para o rearranjo das paredes dentro do container considerando que o peso total do contêiner está dentro do limite é mostrada a seguir:

$$Minimizar\ \Delta d = \left| \frac{\sum_{i=1}^{p} w_i(z_i + d_i)}{\sum_{i=1}^{p} w_i} - \frac{L\overline{N_2}}{\overline{N_1} + \overline{N_2}} \right| \tag{3.21}$$

$sujeito\ a$

$$l_{ik} + l_{ki} = 1 \quad (1 \leq i < k \leq p) \tag{3.22}$$

$$z_i + L_i \leq z_k + (1 - l_{ik})M \quad (1 \leq i < k \leq p) \tag{3.23}$$

$$0 \leq z_i \leq \sum_{j=1}^{p} L_j - L_i \quad (1 \leq i \leq p) \tag{3.24}$$

Nessa modelagem as características das paredes são traduzidas através das seguintes variáveis: w_i corresponde ao peso da parede; L_i corresponde à largura da parede; d_i corresponde à distância do centroide da parede à sua posição inferior esquerda. A variável p corresponde ao número de paredes alocadas, a variável L por sua vez indica o comprimento interno do contêiner e as variáveis $\overline{N_1}$ e $\overline{N_2}$ corresponde aos limites máximos de peso por eixo. Nesse modelo o operador l_{ik} é semelhante àquele utilizado por Pisinger e Sigurd (2005), e indica se uma caixa i esta a esquerda de uma determinada caixa k. A variável z_i indica a posição do ponto inferior esquerda de uma parede ao longo do comprimento do contêiner. Neste modelo a função objetivo deseja minimizar a distância entre o centro de massa real e o centro de massa desejado. A primeira parcela da função objetivo é responsável por calcular o centro de massa real e a segunda calcula o centro de massa desejado. A restrição (3.22) verifica a posição relativa das paredes. A restrição (3.23) diz respeito à sobreposição das paredes e por último a restrição (3.24) garante que todas as paredes estão contidas dentro do contêiner.

Para a remoção de caixas do sistema existem duas situações que são consideradas neste trabalho: a remoção de caixas para atender o limite máximo de peso do container e a remoção de caixas para a atender a carga por eixo. A primeira situação resulta no seguinte modelo:

$$minimizar\ V = \sum_{i=1}^{n} v_i f_i x_i \tag{3.25}$$

$sujeito\ a$

$$\sum_{i=1}^{m} w_{bi} - \sum_{i=1}^{m} w_{bi} f_i x_i \leq W_c \tag{3.26}$$

$$x_i \leq f_i \tag{3.27}$$

$$x \in \{0,1\} \tag{3.28}$$

Do modelo anterior as informações das caixas são fornecidas pelas variáveis w_{bi} e v_i, e corresponde ao peso e ao volume da caixa, respectivamente. O operador f_i é um operador binário responsável por indicar todas as caixas que podem ser removidas, normalmente são aquelas que estão desimpedidas para retirada. A variável x_i é também um operador binário que diz se a caixa i foi retirada $(x_i = 1)$ ou não $(x_i = 0)$. Por último a variável W_c corresponde ao peso máximo suportado pelo contêiner.

Nessa modelagem a função objetivo (3.25) visa minimizar a quantidade total de volume retirado do contêiner. A primeira restrição (3.26) diz respeito à condição de peso máximo do contêiner que deve ser atendida e a restrição (3.27) garante que apenas as caixas desimpedidas possam ser retiradas.

A segunda situação promove a retirada de caixas para atender a restrição de peso por eixo. Nesse caso a modelagem de Lim et al.(2013) é da seguinte forma:

$$minimizar\ V = \sum_{i=1}^{n} v_i f_i x_i \tag{3.29}$$

$sujeito\ a$

$$\sum_{i=1}^{p} n_i^1 - \sum_{i=1}^{p} n_i^1 f_i x_i \leq \overline{N_1} \tag{3.30}$$

$$\sum_{i=1}^{p} n_i^2 - \sum_{i=1}^{p} n_i^2 f_i x_i \leq \overline{N_2} \tag{3.31}$$

$$x_i \leq f_i \tag{3.32}$$

$$x \in \{0,1\} \tag{3.33}$$

Para esse modelo as variáveis n_i^1 e n_i^2 representam as forças resultantes nos eixos 1 e 2, respectivamente, e as variáveis $\overline{N_1}$ e $\overline{N_2}$ são os limites de peso nos referidos eixos.

A função objetivo, da mesma forma que o modelo anterior, visa minimizar a quantidade de caixas retiradas. As restrições (3.30) e (3.31) estão interessadas na contabilização do peso por eixo do caminhão. A última restrição (3.32) é análoga a do modelo de retirada caixas para atender o limite máximo de peso contêiner.

3.2.4. Hadjiconstantinou e Christofides (1995)

Em seu trabalho Hadjiconstantinou e Christofides (1995) abordam o problema de corte de peças retangulares bidimensionais. Sua proposta consiste de um modelo totalmente em programação linear cujo objetivo é maximizar a somatória dos valores associados a um determinado retângulo ou minimizar o desperdício de material. Em sua modelagem Hadjiconstantinou e Christofides (1995) utilizam um modelo em programação linear inteira 0-1, em que as variáveis de decisão têm um comportamento binário.

Para a definição dos problemas os autores estabelecem as seguintes notações: Seja $A_0 = \{\alpha_0,\ \beta_0\}$ o retângulo a ser cortado com comprimento α_0 e largura β_0 e seja R o conjunto de retângulos menores $R_1, R_2, \dots, R_m$ de dimensões $(\alpha_1, \beta_1), (\alpha_2, \beta_2), \dots, (\alpha_m, \beta_m)$(Fig. 3.2)e um valor associado a cada retângulo $v_1, v_2, \dots, v_m$. Além disso, esta modelagem considera que para retângulo existe um valor mínimo de peças (P_i) a serem cortadas e um valor máximo a ser cortado (Q_i) e que os retângulos possuem orientação fixa.

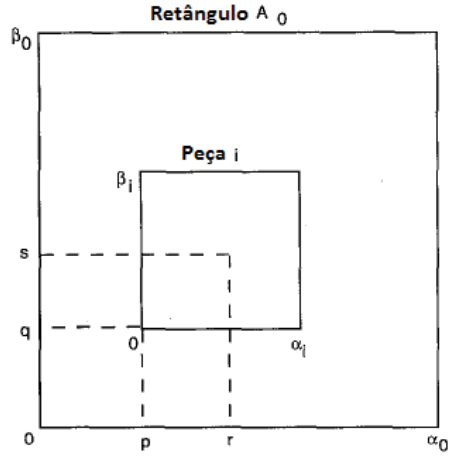


Figura 3.2 – Definição do problema (adaptado de Hadjiconstantinou; Christofides (1995)).

A modelagem toma por base duas variáveis de decisão, que são os operadores x_{ijp} e y_{ijq}. Ambos os operadores podem assumir apenas os valores de 0 e 1, sendo que x_{ijp} vale 1 se a $j-ésima$ réplica da peça i está alocada com sua posição inferior esquerda na posição x igual a p e vale 0 caso o contrário. Da mesma forma, y_{ijq} vale 1 se a $j-ésima$ réplica da peça i está alocada com sua posição inferior esquerda na posição y igual a q e vale 0 caso o contrário. Nesse contexto é definido operador auxiliar z_{rs} que é 1 se o ponto (r,s) foi utilizado por algum retângulo e 0 caso contrário.

O modelo é definido como segue:

$$maximizar\ Z = \sum_{i=1}^{m} v_i \sum_{j=1}^{Q} \sum_{p \in L} x_{ijp} \tag{3.34}$$

$sujeito\ a$

$$\sum_{s=q}^{q+\beta_1-1} \sum_{r=p}^{p+\alpha_1-1} z_{rs} \leq (2 - x_{ijp} - y_{ijq})\alpha_i\beta_i, i = 1, \dots, m\ , j = 1, \dots, Q_i\ , p \in L_i\ , q \epsilon W_i \tag{3.35}$$

$$\sum_{p \in L_i} x_{ijp} \leq 1, \qquad i = 1, \dots, m, \qquad j = 1, \dots, Q_i \tag{3.36}$$

$$\sum_{p \in L_i} x_{ijp} = \sum_{q \in W_i} y_{ijq}\ , \qquad i = 1, \dots, m, \qquad j = 1, \dots, Q_i \tag{3.37}$$

$$\sum_{i=1}^{m} \beta_i \sum_{j=1}^{Q_i} \sum_{p=r-\alpha_1+1, p \in L_i}^{r} x_{ijp} + \sum_{s=0}^{\beta_0-1} z_{rs} = \beta_0, \qquad r = 0, \dots, \alpha_0 - 1 \tag{3.38}$$

$$\sum_{i=1}^{m} \alpha_i \sum_{j=1}^{Q_i} \sum_{q=s-\beta_1+1, q\in W_i}^{s} y_{ijq} + \sum_{r=0}^{\alpha_0-1} z_{rs} = \alpha_0, \qquad s = 0, \dots, \beta_0 - 1 \tag{3.39}$$

$$x_{ijp}, y_{ijq} \in \{0,1\}, \qquad i = 1, \dots, m, \qquad j = 1, \dots, Q_i, \qquad p \in L_i, \qquad q \in W_i \tag{3.40}$$

$$z_{rs} \in \{0,1\}, \qquad r = 0, \dots, \alpha_0 - 1, \qquad s = 0, \dots, \beta_0 - 1 \tag{3.41}$$

$$L_i = \{x | 0 \leq x \leq \alpha_0 - \alpha_i, x\ inteiro\} \tag{3.42}$$

$$W_i = \{y | 0 \leq y \leq \beta_0 - \beta_i, y\ inteiro\} \tag{3.43}$$

Neste modelo a função objetivo visa à maximização do valor dos retângulos cortados, sendo que na restrição (3.35) é garantido que um determinado ponto interno de A_0 é cortado por apenas um retângulo. Já as restrições (3.36) e (3.37) garantem que um retângulo é cortado apenas uma vez de A_0. E por último as restrições (3.38) e (3.39) garantem que o comprimento e largura de A_0 não sejam extrapolados devido ao corte dos retângulos.

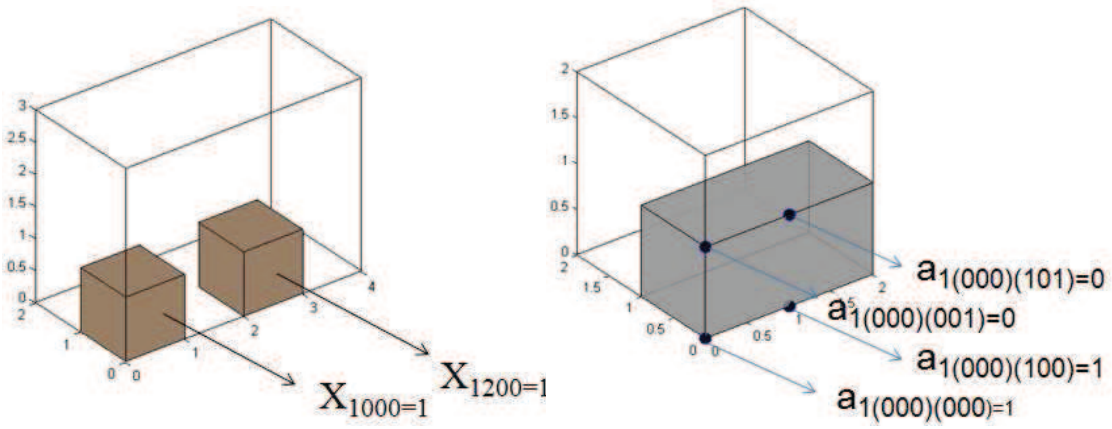
3.2.5. Junqueira (2009)

Em seu trabalho, Junqueira (2009) faz uma revisão da literatura de uma série de modelagens que utilizam da programação linear para a resolução do problema de carregamento de contêiner. Através deste estudo, Junqueira (2009) propõe a sua própria modelagem com base nos trabalhos revisados. Seu modelo consiste de uma adaptação da metodologia apresentada em Beasley (1985) para inserção de restrições de estabilidade do carregamento, limite de peso do contêiner, orientação de caixas, resistência ao empilhamento e múltiplos destinos.

Para introdução da sua modelagem Junqueira (2009) estabelece um conjunto de notações que serão mostradas a seguir: Seja um contêiner de dimensões L, W e H, e caixas com dimensões l_i, w_i e h_i, em que o índice i é utilizado para denotar o tipo de caixas, e seja v_i um valor associado às caixas (custo, volume, dentre outros) e b_i a quantidade de caixas disponíveis do tipo i.

Na definição de seu modelo Junqueira (2009) faz uso de alguns operadores que são úteis na definição das equações de restrição. O operador x_{ipqr}, que é também a variável de decisão do modelo, indica se uma caixa do tipo i foi alocada com o vértice de sua posição frontal inferior esquerda na posição $(p, q, r)(x_{ipqr} = 1)$ ou não $(x_{ipqr} =$

0) (Fig. 3.3a). Para contabilização da sobreposição é introduzido o operador $a_{ipqrstu}$, que é 1 caso a caixa do tipo i alocada com seu vértice frontal inferior esquerdo na posição (p, q, r) ocupa um determinado ponto do espaço, definido pelas coordenadas (s, t, u) e é zero caso contrário (Fig. 3.3b).



a) b)

Figura 3.3 – Operadores no modelo de Junqueira (2009).

Assim o modelo básico apresentado por Junqueira (2009) é da seguinte forma:

$$maximizar\ z = \sum_{i=1}^{m} \sum_{p \in X_i} \sum_{q \in Y_i} \sum_{r \in Z_i} v_i x_{ipqr} \tag{3.44}$$

$sujeito\ a$

$$\sum_{i=1}^{m} \sum_{p \in X_i} \sum_{q \in Y_i} \sum_{r \in Z_i} a_{ipqrstu} x_{ipqr} \leq 1, \qquad s \in X, \qquad t \in Y, \qquad u \in Z \tag{3.45}$$

$$\sum_{p \in X_i} \sum_{q \in Y_i} \sum_{r \in Z_i} x_{ipqr} \leq b_i, \qquad i = 1, \ldots, m \tag{3.46}$$

$$x_{ipqr} \in \{0,1\} \tag{3.47}$$

A função objetivo (3.44) do modelo anterior visa à maximização do somatório do valor associado às caixas individuais. A restrição (3.45) corresponde à restrição de sobreposição de caixas e a restrição (3.46) corresponde à restrição de quantidade de itens.

3.3. Solver

O *solver* corresponde à função capaz de resolver o problema de programação linear obtido através do modelo utilizado. De acordo com o estudado pretende-se avaliar o modelo em se utilizando as técnicas do simplex simples e de duas fases, e ao final obter uma estrutura de dados que indique qual o padrão de empacotamento a ser implementado.

3.4. Gerador de PDDL

Esta função é responsável por traduzir as informações obtidas com solver de forma a ser resolvida por um planejador automático. O modelo implementado toma por base as notações apresentadas na PDDL 3.0, 2.2 e 2.1. Assim, para a obtenção do problema e domínio em PDDL, foi realizado um estudo em cima do sistema a ser modelado, para obtenção das características consideradas fundamentais na obtenção da resposta para o controlador.

Neste estudo foram gerados domínios para dois tipos de caixas diferentes com dimensões (1,1,1) e (2,1,1) unidades. Vale ressaltar que semelhante ao constatado nos modelos de programação linear todas as variáveis do problema são inteiras.

A seguir é mostrada a modelagem para a movimentação de caixas com dimensões (1,1,1) através de uma ponte rolante em se utilizando a notação da UML. A modelagem obtida para a caixa (2,1,1) é extremamente semelhante e portanto não será apresentada. No estudo de caso considerado foram elaborados dois modelos distintos, um modelo completo e outro simplificado, para verificar a capacidade dos planejadores na resolução de cada tipo de modelo. A Fig. 3.4 mostra os diagramas de caso de uso para o sistema considerado.

Observa-se que no modelo completo basicamente três ações: a ação de mover, que envolve apenas a ponte rolante e trata da ação de posicionamento da mesma; a ação pegar, que consiste do procedimento em que a ponte rolante se utiliza de seu elemento terminal para agarrar um produto; e por último a ação soltar consiste do procedimento inverso em que o produto é solto pelo elemento terminal.

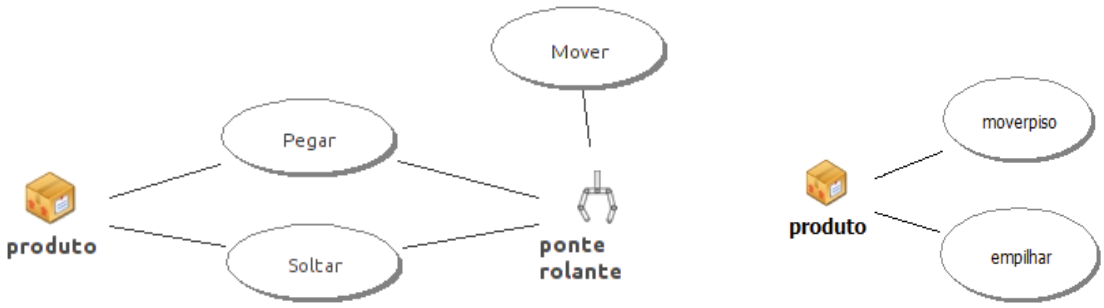


Figura 3.4 – Diagramas de caso de uso do modelo completo à esquerda e do modelo simplificado à direita para a caixa de dimensões (1,1,1).

Já o modelo simplificado contém apenas duas ações: a ação moverpiso, que move um produto do local de recebimento para o chão do contêiner; e a ação empilhar, que empilha um produto do local de recebimento em cima de outras caixa. É importante ressaltar que no procedimento de carregamento todas as operações são realidas pelo topo do contêiner, então considera-se que o mesmo seja aberto na parte superior. Na figura 3.5 são mostrados os diagrama de classes referentes ao modelo completo e simplificado.

Os diagramas de classes foram elaborados no sentido de avaliar diferentes aspectos do processo de resolução através de planejadores automáticos. Conforme pode-se observar na modelo completo, este possui três classes principais. A classe ponteRolante contém as informações a respeito do posicionamento da mesma, através dos atributos $coordx$ e $coordy$, ambas variáveis inteiras. Além disso, informa se a ponte está carregando algum produto ou não, através do atributo booleano $ocupada$. A classe produto contém informações complementares a respeito da posição do produto, indicadas pelas variáveis inteiras $crdx, crdy$ e $crdz$, que também podem ser obtidas pelas relações $estaEM$ e $estaNa$. E por último a classe lugar informa a respeito das coordenadas do mesmo cx, cy e cz, e também se o lugar está ocupado ou não, através do atributo $ocupado$. O atributo $disponível$ desta classe indica se um dado local qualquer no espaço interno do contêiner, o local abaixo está ocupado e por sua vez fornece suporte a alocação de caixas no lugar considerado, evitando assim a presença de caixas flutuando no espaço do contêiner.

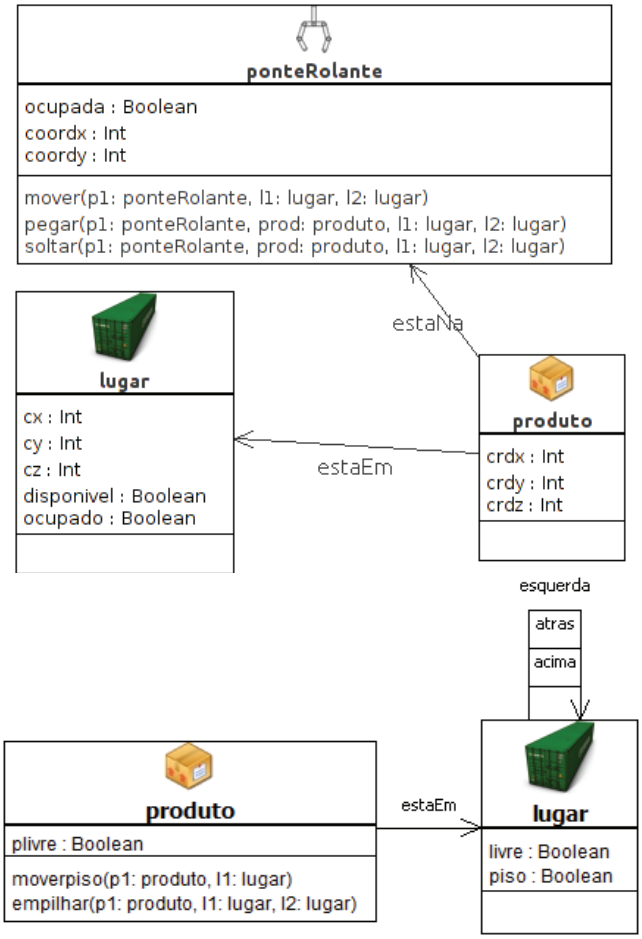


Figura 3.5 – Diagrama de classes do modelo completo (superior) e do modelo simplificado (inferior).

Já o modelo simplificado tem por objetivo eliminar todas as aquelas variáveis inteiras presentes no modelo completo, de tal forma que um determinado estado do sistema possa ser representado através de uma quantidade menor de informações. Este modelo possui apenas duas classes: a classe produto, e a classe lugar. A classe produto possui apenas um atributo *plivre* que indica se o produto está na área de recebimento do processo de carregamento e possui um relação *estaEm* que indica a posição em que o mesmo foi alocado dentro contêiner após o processo de carregamento. Já a classe lugar possui dois atributos. O atributo *livre* indica se o lugar está sendo ocupado por alguma caixa e o atributo *piso* indica se este lugar esta localizado no piso do contêiner. Vale ressaltar que neste modelo não existem coordenadas inteiras associadas aos lugares, dessa forma, para o arranjo organizado dos lugares internos do contêineres, todos eles são conectados através das relações *esquerda*, *acima* e *atrás,* ou seja, cada local tem conhecimento dos locais que estão à esquerda, acima e atrás de sua posição. Através

destas relações pode-se obter sem dificuldades as condições para alocação de caixas de variadas dimensões. Na sequência é mostrado o diagrama de estados para o problema em questão.

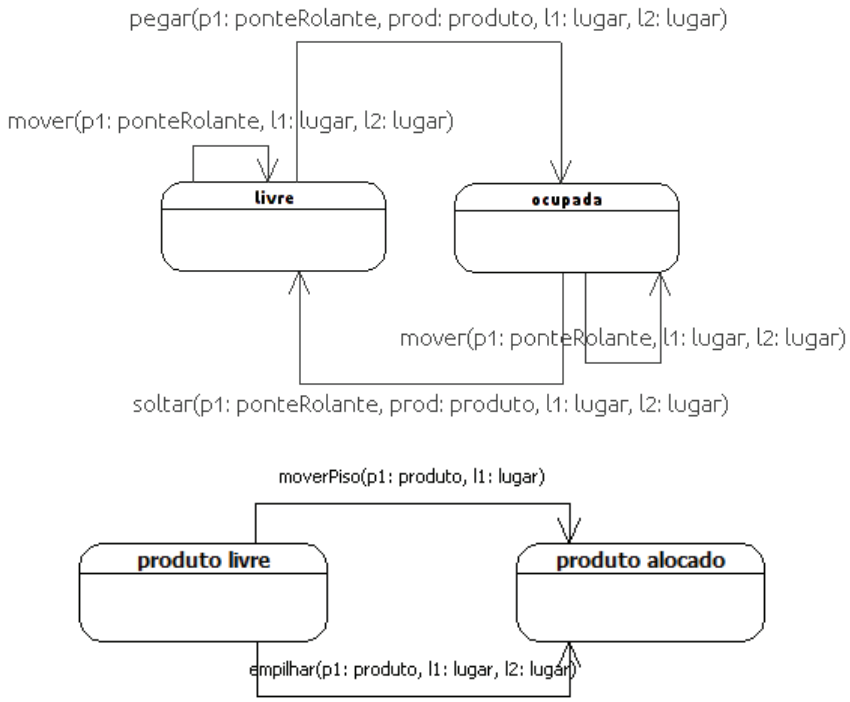


Figura 3.6 – Diagramas de estado do modelo completo (superior) e do modelo simplificado (inferior).

O diagrama de estados superior mostrado na Fig. 3.6 corresponde ao diagrama para a ponte rolante no modelo completo. Nesse problema podem-se diferenciar basicamente dois estados: o estado livre, em que a ponte não está carregando nenhum produto; e o estado ocupada, em que a ponte está carregando algum produto. As funções que ligam os estados são literais e em qualquer estado a ponte rolante pode se mover. No diagrama de estados inferior da Fig. 3.6, representa-se apenas o comportamento dinâmico do produto. Neste modelo simplificado existe dois estados possíveis: o estado produto livre representa a situação em que o produto não alocado dentro do contêiner; e o estado produto alocado, representa a situação contrária. Para transição entre os dois estados deve-se utilizar das moverPiso e empilhar. Vale ressaltar que neste modelo as ações do ponte rolante, que por sua vez não representada, podem ser inferidas a partir dos locais de inicio e fim do produto e também pelo tipo do produto, no procedimento de automatização do processo de carregamento.

Com base nos diagramas expostos foram elaborados os domínios completo e simplificado em se utilizando a notação da PDDL para as caixas (1,1,1) e (2,1,1), conforme é mostrado a seguir. A impossibilidade de se criar ações com argumentos variáveis aumenta a complexidade da modelagem que requer ações específicas para cada tipo de caixa, nesse sentido os produtos citados anteriormente foram estudados em domínios separados.

Tabela 3.1 – Domínios para caixa (1,1,1).

Domínio completo caixa (1,1,1)

```
(define (domain New_Project_1)
    (:requirements :typing :fluents :negative-preconditions
:equality)
(:types
      ponteRolante - object
      produto - object
      lugar - object
    )
    (:predicates
      (estaEm ?pro - produto ?lug - lugar)
      (estaNa ?pro - produto ?pon - ponteRolante)
      (ocupada ?pon - ponteRolante)
      (disponivel ?lug - lugar)
      (ocupado ?lug - lugar)
    )
    (:functions
      (coordx ?pon - ponteRolante)
      (coordy ?pon - ponteRolante)
      (crdx ?pro - produto)
      (crdy ?pro - produto)
      (crdz ?pro - produto)
      (cx ?lug - lugar)
      (cy ?lug - lugar)
      (cz ?lug - lugar)
    )
    (:action mover
     :parameters (?p1 - ponteRolante ?l1 - lugar ?l2 - lugar)
:precondition
       (and
         (= (coordx ?p1) (cx ?l1))
         (= (coordy ?p1) (cy ?l1))
         (not (= ?l1 ?l2))
       )
     :effect
       (and
         (assign (coordx ?p1) (cx ?l2))
         (assign (coordy ?p1) (cy ?l2))
)
    )
    (:action pegar
     :parameters (?p1 - ponteRolante ?prod - produto ?l1 -
lugar ?l2 - lugar)
```

```
:precondition
       (and
         (= (coordx ?p1) (cx ?l1))
         (= (coordy ?p1) (cy ?l1))
         (= (cx ?l1) (cx ?l2))
         (= (cy ?l1) (cy ?l2))
         (= (cz ?l1) (- (cz ?l2) 1))
(ocupado ?l1)
         (not (ocupado ?l2))
         (estaEm ?prod ?l1)
         (not (ocupada ?p1))
)
     :effect
       (and
         (estaNa ?prod ?p1)
         (not (ocupado ?l1))
         (not (disponivel ?l2))
         (assign (crdx ?prod) (- 0 1))
         (assign (crdy ?prod) (- 0 1))
(assign (crdz ?prod) (- 0 1))
         (ocupada ?p1)
(not (estaEm ?prod ?l1))
       )
    )
    (:action soltar
:parameters (?p1 - ponteRolante ?prod - produto ?l1 - lugar ?l2
- lugar)
:precondition
       (and
         (= (coordx ?p1) (cx ?l1))
         (= (coordy ?p1) (cy ?l1))
         (= (cx ?l1) (cx ?l2))
         (= (cy ?l1) (cy ?l2))
         (= (cz ?l1) (- (cz ?l2) 1))
(not (ocupado ?l1))
         (not (ocupado ?l2))
         (disponivel ?l1)
         (estaNa ?prod ?p1)
         (ocupada ?p1)
)
     :effect
       (and
         (estaEm ?prod ?l1)
(ocupado ?l1)
(disponivel ?l2)
         (assign (crdx ?prod) (cx ?l1))
         (assign (crdy ?prod) (cy ?l1))
         (assign (crdz ?prod) (cz ?l1))
(not (ocupada ?p1))
         (not (estaNa ?prod ?p1))
       )
    )
)
```

Domínio simplificado caixa (1,1,1)

```
(define (domain New_Project_1)
    (:requirements :strips)
```

```
    (:types
      lugar - object
      produto1 - object
    )
    (:predicates
      (estaEm1 ?pro - produto1 ?lug - lugar)
      (plivre1 ?prod - produto1)
      (atras ?lug - lugar ?lug - lugar)
      (esquerda ?lug - lugar ?lug - lugar)
      (acima ?lug - lugar ?lug - lugar)
      (livre ?lug - lugar)
      (piso ?lug - lugar)
    )
    (:action moverPiso1
     :parameters (?prod - produto1 ?l1 - lugar)
     :precondition
       (and
         (plivre1 ?prod)
         (livre ?l1)
         (piso ?l1)
       )
     :effect
       (and
         (estaEm1 ?prod ?l1)
         (not (plivre1 ?prod))
         (not (livre ?l1))
       )
    )
    (:action empilhar1
     :parameters (?prod - produto1 ?l1 - lugar ?l2 - lugar)
     :precondition
       (and
         (plivre1 ?prod)
         (not (livre ?l1))
         (acima ?l2 ?l1)
       )
     :effect
       (and
         (estaEm1 ?prod ?l2)
         (not (plivre1 ?prod))
         (not (livre ?l2))
       )
    )
)
```

Tabela 3.2 – Domínios para caixa (2,1,1).

Dominio completo caixa (2,1,1)

(define (domain New_Project_1)
    (:requirements :typing :fluents :negative-preconditions
:equality)
(:types
      lugar - object
      produto - object

```
      ponteRolante - object
    )
    (:predicates
      (estaEm ?pro - produto ?lug - lugar)
      (estaNa ?pro - produto ?pon - ponteRolante)
      (ocupado ?lug - lugar)
      (disponivel ?lug - lugar)
      (ocupada ?pon - ponteRolante)
    )
    (:functions
      (cx ?lug - lugar)
      (cy ?lug - lugar)
      (cz ?lug - lugar)
      (crdx ?pro - produto)
      (crdy ?pro - produto)
      (crdz ?pro - produto)
      (coordX ?pon - ponteRolante)
      (coordY ?pon - ponteRolante)
    )
    (:action mover
     :parameters (?p1 - ponteRolante ?l1 - lugar ?l2 - lugar)
:precondition
       (and
         (= (coordX ?p1) (cx ?l1))
         (= (coordY ?p1) (cy ?l1))
         (not (= ?l1 ?l2))
       )
     :effect
       (and
         (assign (coordX ?p1) (cx ?l2))
         (assign (coordY ?p1) (cy ?l2))
)
    )
    (:action pegar
     :parameters (?p1 - ponteRolante ?prod - produto ?l1 -
lugar ?l2 - lugar ?l3 - lugar ?l4 - lugar)
:precondition
       (and
         (not (ocupada ?p1))
(estaEm ?prod ?l1)
         (ocupado ?l1)
         (ocupado ?l2)
         (not (ocupado ?l3))
         (not (ocupado ?l4))
         (= (coordX ?p1) (cx ?l2))
(= (coordY ?p1) (cy ?l2))
         (= (cy ?l1) (cy ?l2))
(= (cz ?l1) (cz ?l2))
         (= (cx ?l1) (cx ?l3))
(= (cy ?l1) (cy ?l3))
         (= (cx ?l2) (cx ?l4))
         (= (cy ?l2) (cy ?l4))
         (= (cx ?l1) (- (cx ?l2) 1))
(= (cz ?l1) (- (cz ?l3) 1))
         (= (cz ?l2) (- (cz ?l4) 1))
)
```

```
     :effect
       (and
         (estaNa ?prod ?p1)
(ocupada ?p1)
         (not (ocupado ?l1))
(not (ocupado ?l2))
         (not (disponivel ?l3))
         (not (disponivel ?l4))
         (assign (crdx ?prod) (- 0 1))
         (assign (crdy ?prod) (- 0 1))
         (assign (crdz ?prod) (- 0 1))
         (not (estaEm ?prod ?l1))
)
    )
    (:action soltar
     :parameters (?p1 - ponteRolante ?prod - produto ?l1 -
lugar ?l2 - lugar ?l3 - lugar ?l4 - lugar)
     :precondition
       (and
         (ocupada ?p1)
         (estaNa ?prod ?p1)
         (not (ocupado ?l1))
         (not (ocupado ?l2))
         (not (ocupado ?l3))
         (not (ocupado ?l4))
         (disponivel ?l1)
         (disponivel ?l2)
         (= (coordX ?p1) (cx ?l2))
(= (coordY ?p1) (cy ?l2))
         (= (cx ?l1) (- (cx ?l2) 1))
         (= (cy ?l1) (cy ?l2))
         (= (cz ?l1) (cz ?l2))
         (= (cx ?l1) (cx ?l3))
         (= (cy ?l1) (cy ?l3))
(= (cz ?l1) (- (cz ?l3) 1))
         (= (cx ?l2) (cx ?l4))
(= (cy ?l2) (cy ?l4))
         (= (cz ?l2) (- (cz ?l4) 1))
       )
     :effect
       (and
         (estaEm ?prod ?l1)
(not (ocupada ?p1))
         (ocupado ?l1)
         (ocupado ?l2)
         (disponivel ?l3)
(disponivel ?l4)
(assign (crdx ?prod) (cx ?l1))
         (assign (crdy ?prod) (cy ?l1))
         (assign (crdz ?prod) (cz ?l1))
(not (estaNa ?prod ?p1))
       )
    )
)
```

Dominio simplificado caixa (2,1,1)

```
(define (domain New_Project_1)
```

```
    (:requirements :strips)
    (:types
      lugar - object
      produto - object
    )
    (:predicates
      (estaEm ?pro - produto ?lug - lugar)
      (plivre ?prod - produto)
      (atras ?lug - lugar ?lug - lugar)
      (esquerda ?lug - lugar ?lug - lugar)
      (acima ?lug - lugar ?lug - lugar)
      (livre ?lug - lugar)
      (piso ?lug - lugar)
    )
    (:action moverPiso
     :parameters (?prod - produto ?l1 - lugar ?l2 - lugar)
     :precondition
       (and
         (plivre ?prod)
         (livre ?l1)
         (piso ?l1)
         (livre ?l2)
         (piso ?l2)
         (atras ?l2 ?l1)
       )
     :effect
       (and
         (estaEm ?prod ?l1)
         (not (plivre ?prod))
         (not (livre ?l1))
         (not (livre ?l2))
       )
    )
    (:action empilhar
     :parameters (?prod - produto ?l1 - lugar ?l2 - lugar ?l3 -
lugar ?l4 - lugar)
     :precondition
       (and
         (plivre ?prod)
         (not (livre ?l1))
         (not (livre ?l2))
         (atras ?l4 ?l3)
         (acima ?l3 ?l1)
         (acima ?l4 ?l2)
       )
     :effect
       (and
         (estaEm ?prod ?l3)
         (not (plivre ?prod))
         (not (livre ?l3))
         (not (livre ?l4))
       )
    )
)
```

Para a validação dos domínios apresentados anteriormente foram elaborados os seguintes problemas, cujos resultados puderam ser encontrados por meio dos planejadores automáticos.

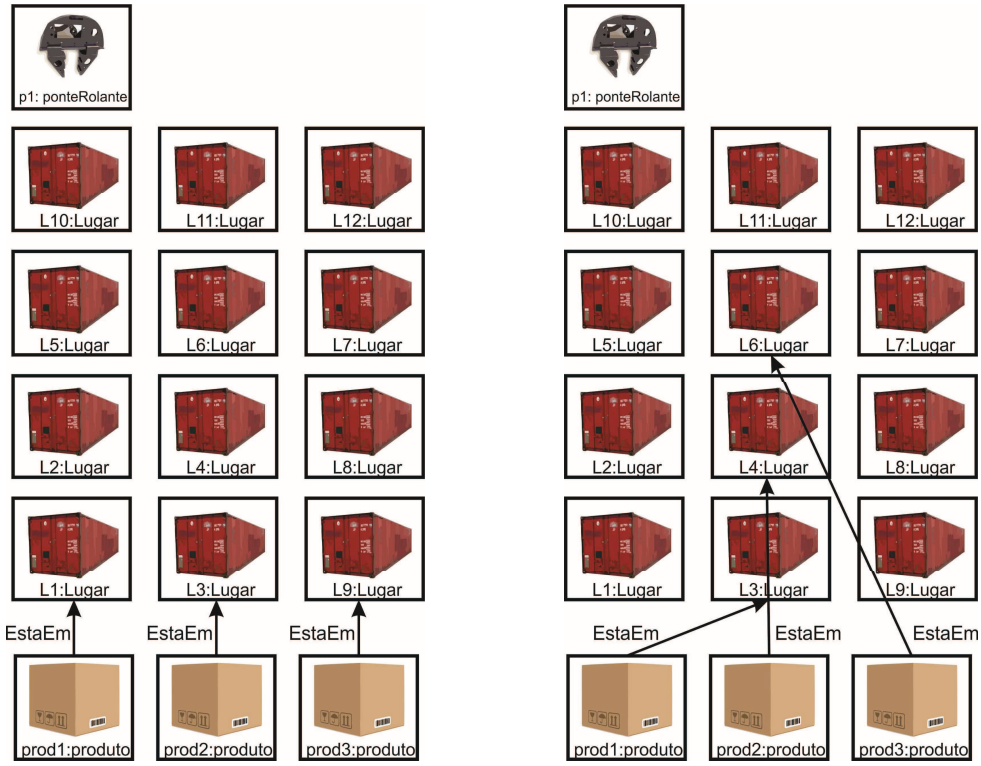


Figura 3.7 – Problema proposto para caixa 111 utilizando o modelo completo. *Snapshot* inicial à esquerda e *snapshot* à direita.

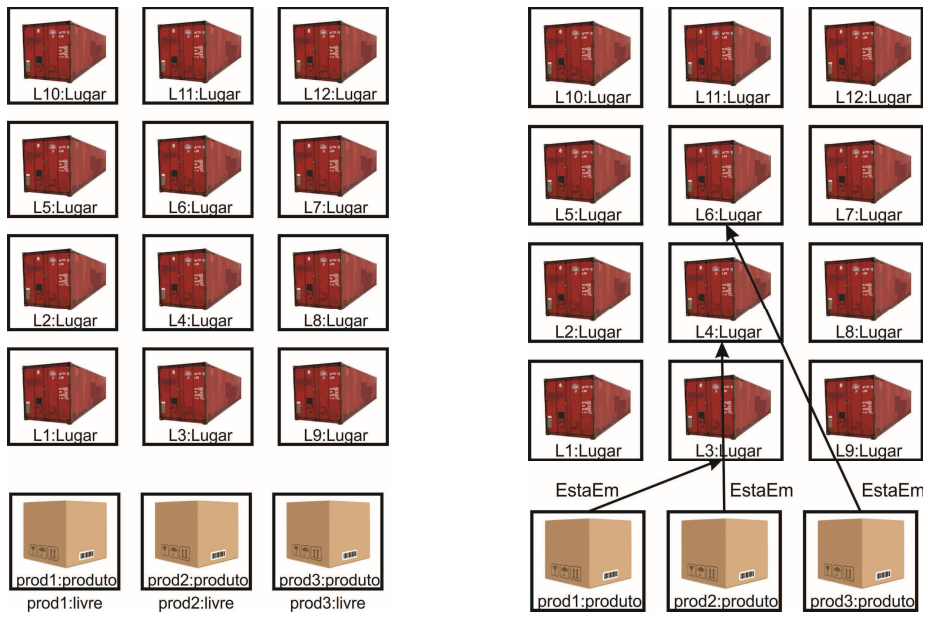


Figura 3.8 – Problema proposto para caixa 111 utilizando o modelo simplificado. *Snapshot* inicial à esquerda e *snapshot* à direita.

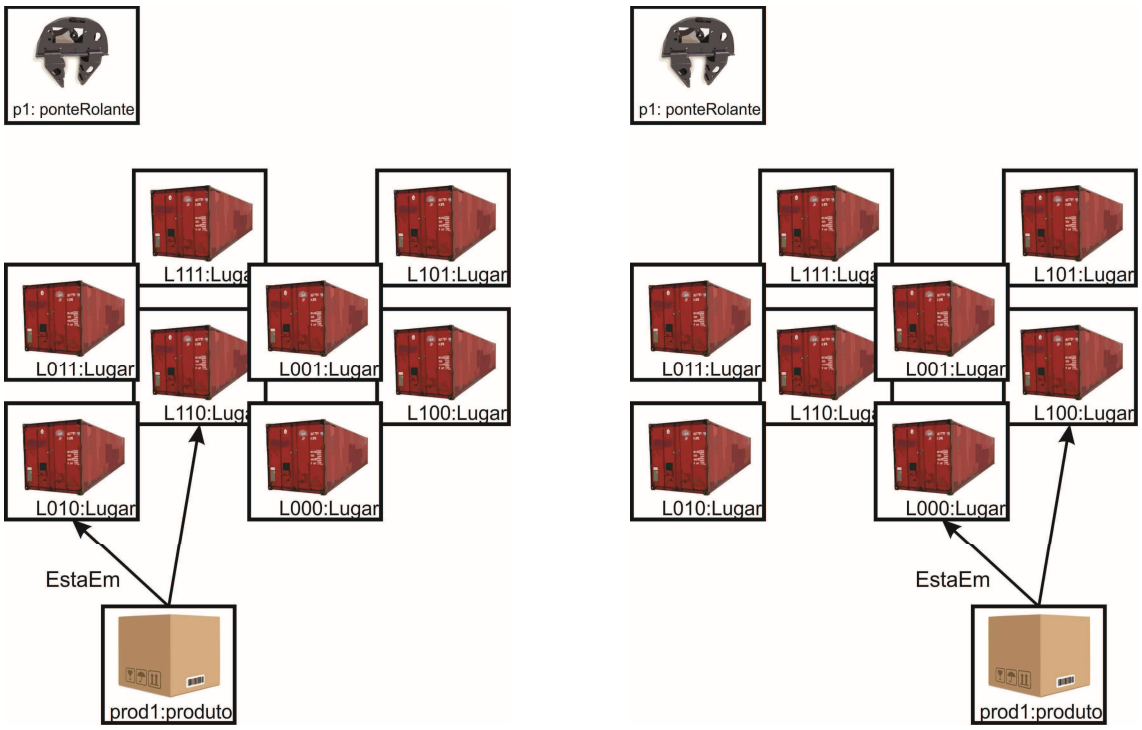


Figura 3.9 – Problema proposto para caixa 211 utilizando o modelo completo. *Snapshot* inicial à esquerda e *snapshot* à direita.

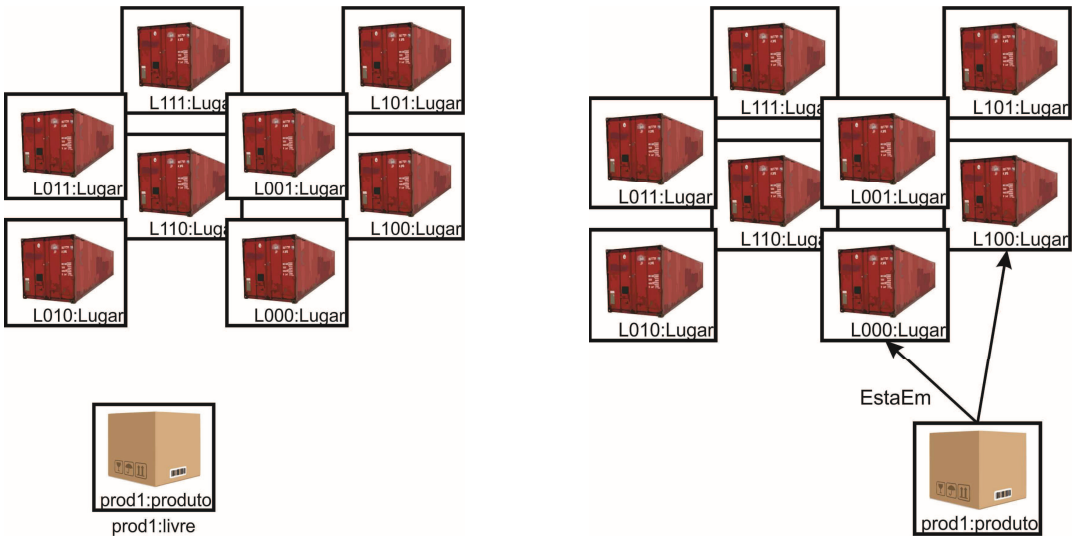


Figura 3.10 – Problema proposto para caixa 211 utilizando o modelo simplificado. *Snapshot* inicial à esquerda e *snapshot* à direita.

Tabela 3.3 – Problema para o domínio da caixa (1,1,1).

Problema caixa (1,1,1) modelo completo

```
(define (problem Planning_Problem)
    (:domain New_Project_1)
(:objects
      prod1 - produto
      l1 - lugar
      l2 - lugar
      p1 - ponteRolante
      l3 - lugar
      l4 - lugar
      l5 - lugar
      l6 - lugar
      prod2 - produto
      l7 - lugar
      l8 - lugar
      l9 - lugar
      prod3 - produto
      l10 - lugar
      l11 - lugar
      l12 - lugar
    )
    (:init
      (= (coordx p1) 1)
      (= (coordy p1) 0)
      (estaEm prod1 l1)
      (estaEm prod2 l3)
      (estaEm prod3 l9)
      (= (cx l1) 0)
      (= (cy l1) 0)
      (= (cz l1) 0)
      (disponivel l1)
      (ocupado l1)
      (= (crdx prod1) 0)
      (= (crdy prod1) 0)
      (= (crdz prod1) 0)
      (= (cx l2) 0)
      (= (cy l2) 0)
      (= (cz l2) 1)
      (disponivel l2)
      (= (cx l3) 1)
      (= (cy l3) 0)
      (= (cz l3) 0)
      (disponivel l3)
      (ocupado l3)
      (= (crdx prod2) 1)
      (= (crdy prod2) 0)
      (= (crdz prod2) 0)
      (= (cx l4) 1)
      (= (cy l4) 0)
      (= (cz l4) 1)
      (disponivel l4)
      (= (cx l5) 0)
      (= (cy l5) 0)
      (= (cz l5) 2)
```

```
      (= (cx l6) 1)
      (= (cy l6) 0)
      (= (cz l6) 2)
      (= (cx l7) 2)
      (= (cy l7) 0)
      (= (cz l7) 2)
      (= (cx l8) 2)
      (= (cy l8) 0)
      (= (cz l8) 1)
      (disponivel l8)
      (= (cx l9) 2)
      (= (cy l9) 0)
      (= (cz l9) 0)
      (disponivel l9)
      (ocupado l9)
      (= (crdx prod3) 2)
      (= (crdy prod3) 0)
      (= (crdz prod3) 0)
      (= (cx l10) 0)
      (= (cy l10) 0)
      (= (cz l10) 3)
      (= (cx l11) 1)
      (= (cy l11) 0)
      (= (cz l11) 3)
      (= (cx l12) 2)
      (= (cy l12) 0)
      (= (cz l12) 3)
    )
    (:goal
(and
        (estaEm prod1 l3)
        (estaEm prod2 l4)
        (estaEm prod3 l6)
        (not (ocupada p1))
        (disponivel l3)
        (ocupado l3)
      )
    )
)
```

Problema caixa (1,1,1) modelo simplificado

```
(define (problem Planning_Problem)
    (:domain New_Project_1)
    (:objects
      l1 - lugar
      l2 - lugar
      l3 - lugar
      l4 - lugar
      l5 - lugar
      l6 - lugar
      l7 - lugar
      l8 - lugar
      l9 - lugar
      l10 - lugar
      l11 - lugar
      l12 - lugar
      prod1 - produto
```

```
      prod2 - produto
      prod3 - produto
    )
    (:init
      (livre l1)
      (piso l1)
      (livre l2)
      (piso l2)
      (livre l3)
      (piso l3)
      (livre l4)
      (livre l5)
      (livre l6)
      (livre l7)
      (livre l8)
      (livre l9)
      (livre l10)
      (livre l11)
      (livre l12)
      (atras l3 l1)
      (atras l9 l3)
      (atras l4 l2)
      (atras l8 l4)
      (atras l6 l5)
      (atras l7 l6)
      (atras l11 l10)
      (atras l11 l6)
      (acima l2 l1)
      (acima l5 l2)
      (acima l4 l3)
      (acima l6 l4)
      (acima l10 l5)
      (acima l9 l6)
      (acima l12 l7)
      (acima l7 l8)
      (acima l8 l9)
      (plivre prod1)
      (plivre prod2)
      (plivre prod3)
    )
    (:goal
      (and
        (estaEm prod1 l3)
        (estaEm prod2 l4)
        (estaEm prod3 l6)
      )
    )
)
```

Tabela 3.4 – Problema para o domínio da caixa (2,1,1).

Problema caixa (2,1,1) modelo completo
(define (problem Planning_Problem)   (:domain New_Project_1)

```
(:objects
      l111 - lugar
      l101 - lugar
      l110 - lugar
      l100 - lugar
      l011 - lugar
      l010 - lugar
      l001 - lugar
      l000 - lugar
      prod1 - produto
      p1 - ponteRolante
    )
    (:init
      (estaEm prod1 l010)
      (= (cx l111) 1)
      (= (cy l111) 1)
      (= (cz l111) 1)
      (disponivel l111)
      (= (cx l101) 1)
      (= (cy l101) 0)
      (= (cz l101) 1)
      (= (cx l110) 1)
      (= (cy l110) 1)
      (= (cz l110) 0)
      (ocupado l110)
      (disponivel l110)
      (= (cx l100) 1)
      (= (cy l100) 0)
      (= (cz l100) 0)
      (disponivel l100)
      (= (cx l011) 0)
      (= (cy l011) 1)
      (= (cz l011) 1)
      (disponivel l011)
      (= (cx l010) 0)
      (= (cy l010) 1)
      (= (cz l010) 0)
      (ocupado l010)
      (disponivel l010)
      (= (cx l001) 0)
      (= (cy l001) 0)
      (= (cz l001) 1)
      (= (cx l000) 0)
      (= (cy l000) 0)
      (= (cz l000) 0)
      (disponivel l000)
      (= (len prod1) 2)
      (= (wid prod1) 1)
      (= (hei prod1) 1)
      (= (crdx prod1) 0)
      (= (crdy prod1) 1)
      (= (crdz prod1) 0)
      (= (coordX p1) 1)
      (= (coordY p1) 1)
    )
    (:goal
```

```
      (and
        (estaEm prod1 l000)
        (not (ocupado l111))
        (not (disponivel l111))
        (not (ocupado l011))
        (not (disponivel l011))
        (= (len prod1) 2)
        (= (wid prod1) 1)
        (= (hei prod1) 1)
        (not (ocupada p1))
        (= (coordX p1) 1)
        (= (coordY p1) 1)
      )
    )
)
```

Problema caixa (2,1,1) modelo simplificado

```
(define (problem Planning_Problem)
    (:domain New_Project_1)
    (:objects
      l000 - lugar
      l100 - lugar
      l010 - lugar
      l110 - lugar
      l001 - lugar
      l101 - lugar
      l011 - lugar
      l111 - lugar
      prod1 - produto
    )
    (:init
      (livre l000)
      (piso l000)
      (livre l100)
      (piso l100)
      (livre l010)
      (piso l010)
      (livre l110)
      (piso l110)
      (livre l001)
      (livre l101)
      (livre l011)
      (livre l111)
      (atras l100 l000)
      (atras l110 l010)
      (atras l101 l001)
      (atras l111 l011)
      (esquerda l010 l000)
      (esquerda l110 l100)
      (esquerda l010 l001)
      (esquerda l111 l101)
      (acima l001 l000)
      (acima l101 l100)
      (acima l011 l010)
      (acima l111 l110)
      (plivre prod1)
    )
```

```
    (:goal
      (and
        (estaEm prod1 l000)
      )
    )
)
```

Com base nos domínios e problemas apresentados pretende-se então elaborar uma função que os produza de forma sistemática sabendo das informações providas pelo solver e as funções anteriores.

3.5. Planejador Automático

O planejador automático normalmente trata-se de um software capaz de avaliar as diferentes possibilidades para execução de uma determinada tarefa. Este assunto já foi previamente abordado na seção referente aos fundamentos teóricos e neste trabalho pretende-se apenas utilizar o que já está consolidado nesta área do conhecimento, que corresponde aos planejadores encontrados na literatura.

3.6. Conversor de Planos

O conversor de planos é uma função que deve ser capaz de traduzir a informação fornecida pelo planejador automático em algo que seja interpretado diretamente pelo controlador. Vale ressaltar que a saída do planejador corresponde a uma lista de ações consecutivas cuja nomenclatura é determinada pelo domínio de planejamento.

3.7. Controlador

O controlador corresponde à parte de controle do sistema físico. Para esta tarefa pode-se utilizar um microcontrolador do tipo PIC, Arduino, Arm, dentre outros.

CAPITULO IV

RESULTADOS

A proposta inicial deste projeto consistia da integração de técnicas de planejamento automático com uma aplicação real. Com o desenvolvimento do trabalho foram encontrados alguns percalços. A abordagem inicial para na construção do modelo para organização das caixas tomava por base um dos problemas clássicos da literatura de planejamento automático, que se trata do mundo de blocos. Com este modelos foram realizadas algumas tentativas para a introdução de novas restrições, conforme constado na literatura do problema de carregamento de contêineres. Os planejadores utilizados nesta tentativa de resolução do problema tiveram dificuldades em lidar com as restrições impostas, pois se tratavam de variáveis de custo não triviais as quais os mesmos não foram projetados para lidar e invariavelmente não foram capazes de fornecer nenhum plano. Constatada esta dificuldade a abordagem prosseguiu conforme a arquitetura mostrada na proposta de trabalho. O objetivo era que cada uma das técnicas utilizadas pudessem de certa forma, compensar a limitação umas das outras. A arquitetura atual parte do pressuposto de que a modelagem com a programação linear seja capaz de definir o padrão de empacotamento ótimo, feito pode-se então obter ambos os estados iniciais e finais necessários ao processo de planejamento automático.

Nesta seção serão mostrados todos os resultados referentes à proposta de trabalho apresentado na seção anterior. Para uma maior facilidade de leitura serão abordados cada um dos itens expostos no diagrama da Fig. 3.1. Devido à própria limitação do autor com técnicas de programação, toda a parte referente à programação linear, algoritmo simplex e geração de arquivos PDDL, foi consolidada em se utilizando do Matlab 2012a (7.14.0.739).

Dando início a exposição dos resultados, o primeiro item trata do gerador de problemas. O gerador de problemas foi concebido para que pudessem ser introduzidas as informações a respeito do problema a ser resolvido. Conforme dito na seção da proposta do trabalho as informações a serem coletadas são as seguintes:

- Largura do contêiner
- Comprimento do contêiner
- Altura do contêiner
- Número de tipos de caixa
- Quantidade de caixas de cada tipo
- Largura da caixa
- Comprimento da Caixa
- Altura da caixa

Para obtenção dessas informações foi construída uma função no Matlab denominada de *problem_generator*. O código referente a esta função pode ser encontrado na seção de apêndices deste trabalho. A Fig. 4.1 mostra uma instância da função sendo utilizada no Matlab.

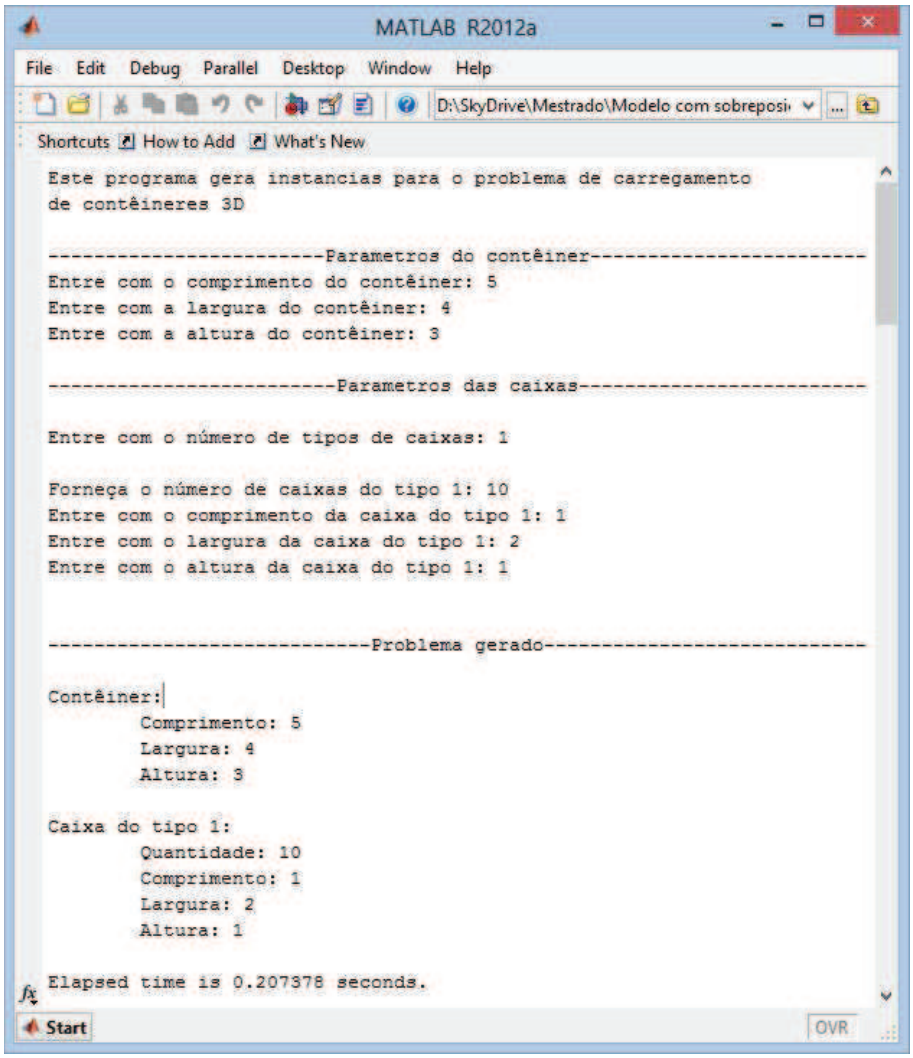


Figura 4.1 – Execução da função do *problem_generator*.

Vale ressaltar que nesta função são verificadas as dimensões das caixas com relação ao contêiner, verificação de valores de negativos e, além disso, são consideradas todas as rotações ortogonais possíveis (R_i) para uma caixa (Fig. 4.2) e se essas rotações resultam em violação às dimensões do contêiner.

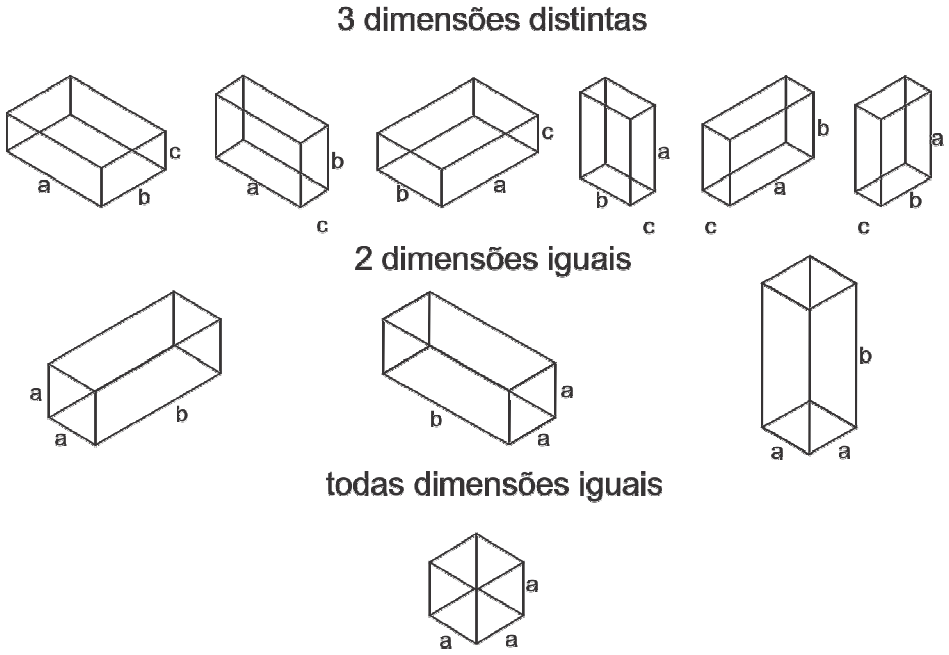


Figura 4.2 – Rotações ortogonais possíveis para caixas.

Na sequência gera-se o modelo com as informações obtidas na função anterior através do *model_generator*. Esta função implementa um modelo matemático que é semelhante àquele utilizado por Junqueira (2009) em seu trabalho.

Seja um contêiner de comprimento$L$, largura W e altura H, e caixas com comprimento l_{ig}, largura w_{ig} e comprimento h_{ig}, onde $i|\ i = (1 \dots m)$ representa o tipo da caixa e $g|\ g \in R_i$ corresponde às rotações ortogonais, e seja também b_i as quantidades disponíveis para cada tipo de caixa e v_i um valor genérico associado a cada caixa. Para o contêiner em questão adota-se o seguinte sistema de coordenadas.

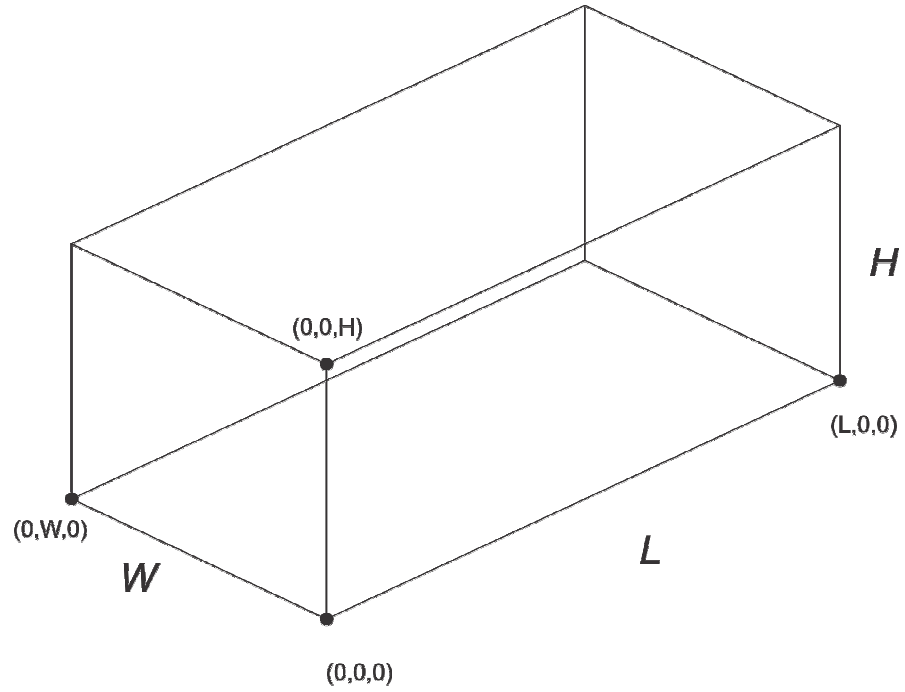


Figura 4.3 – Sistema de coordenadas.

O problema de carregamento com rotação de caixas é também descrito no trabalho de Junqueira (2009). Tomando por base o referido trabalho é proposto o seguinte modelo, correspondente um modelo básico de Junqueira (2009) adicionado de algumas modificações.

$$maximizar \sum_{i=1}^{m} \sum_{g \epsilon R_i} \sum_{p \epsilon X_{ig}} \sum_{q \epsilon Y_{ig}} \sum_{r \epsilon Z_{ig}} v_i x_{igpqr} \tag{4.1}$$

$sujeito\ a$

$$\sum_{i=1}^{m} \sum_{g \epsilon R_i} \sum_{p \epsilon X_{ig}} \sum_{q \epsilon Y_{ig}} \sum_{r \epsilon Z_{ig}} a_{igpqrstu} x_{igpqr} \leq 1, \quad s \in X, \quad t \epsilon Y, \quad u \epsilon Z \tag{4.2}$$

$$\sum_{g \epsilon R_i} \sum_{p \epsilon X_{ig}} \sum_{q \epsilon Y_{ig}} \sum_{r \epsilon Z_{ig}} x_{igpqr} \leq b_i \tag{4.3}$$

$$X = \{p \in X | 0 \leq p \leq L - 1 \wedge inteiro\} \tag{4.4}$$

$$Y = \{q \in Y | 0 \leq q \leq W - 1 \wedge inteiro\} \tag{4.5}$$

$$Z = \{r \in Z | 0 \leq r \leq H - 1 \wedge inteiro\} \tag{4.6}$$

$$X_{ig} = \{p \in X | 0 \leq p \leq L - l_{ig} \wedge inteiro\} \tag{4.7}$$

$$Y_{ig} = \{q \in Y | 0 \leq q \leq W - w_{ig} \wedge inteiro\} \tag{4.8}$$

$$Z_{ig} = \{r \in Z | 0 \leq r \leq H - h_{ig} \wedge inteiro\} \tag{4.9}$$

No modelo em questão o operador x_{igpqr} corresponde à variável de decisão. Seu comportamento é estritamente binário e corresponde a 1 quando uma caixa do tipo i e rotação ortogonal g está alocada na posição (p, q, r) (Fig. 4.4) definida através do sistema de coordenadas, e é 0 caso contrário.

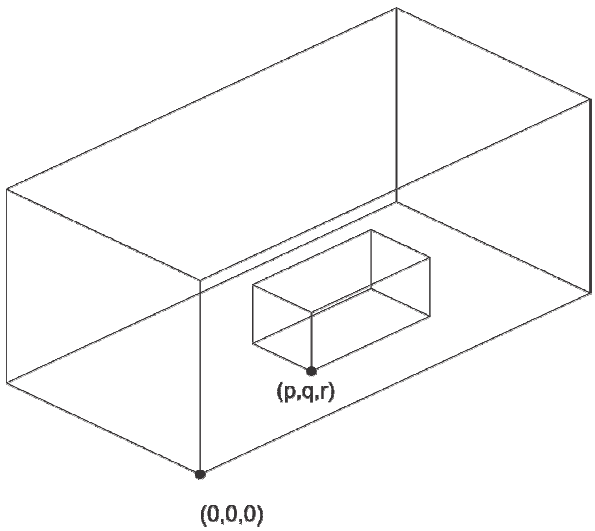


Figura 4.4 – Caixa alocada na posição (p,q,r).

O operador $a_{igpqrstu}$ diz se uma caixa do tipo i e rotação g quando alocada na posição (p, q, r) ocupa o lugar no espaço definido pela coordenada (s, t, u).

O problema reformulado conforme mostrado anteriormente, considera basicamente as restrições de sobreposição das caixas e a quantidade de itens. Para o trabalho em questão é interessante considerar a estabilidade vertical das caixas, para que durante o processo de resolução do sistema não ocorra a flutuação das mesmas. Esta restrição também é contemplada no trabalho de Junqueira (2009). Nesta restrição conforme o modelo apresentado por Junqueira (2009) em seu trabalho é necessário a presença das variáveis de projeto em ambos os lados da equação de restrição, o que não suportado pelo solver desenvolvido neste trabaho. Assim, teve-se que formular uma restrição totalmente nova cujo modelo completo é mostrado na sequência.

$$maximizar \sum_{i=1}^{m} \sum_{g \epsilon R_i} \sum_{p \epsilon X_{ig}} \sum_{q \epsilon Y_{ig}} \sum_{r \epsilon Z_{ig}} v_i x_{igpqr} \tag{4.10}$$

$sujeito\ a$

$$\sum_{i=1}^{m}\sum_{g\epsilon R_i}\sum_{p\epsilon X_{ig}}\sum_{q\epsilon Y_{ig}}\sum_{r\epsilon Z_{ig}} a_{igpqrstu}x_{igpqr} \leq 1, \quad s \in X, \quad t\epsilon Y, \quad u\epsilon Z \tag{4.11}$$

$$\sum_{g\epsilon R_i}\sum_{p\epsilon X_{ig}}\sum_{q\epsilon Y_{ig}}\sum_{r\epsilon Z_{ig}} x_{igpqr} \leq b_i \tag{4.12}$$

$$\begin{aligned} -x_{igp'q'r'} + \sum_{i=1}^{m}\sum_{g\epsilon R_i}\sum_{p\epsilon X_{ig}}\sum_{q\epsilon Y_{ig}}\sum_{r\epsilon Z_{ig}} c_{igpqrstr'}x_{igpqr} \geq 0, \quad p' \in X_{ig}, \\ q' \in Y_{ig}, \quad r' \in Z_{ig} p' \leq s \leq p' + l_{ig} - 1, \\ q' \leq t \leq q' + w_{ig} - 1 \end{aligned} \tag{4.13}$$

$$X = \{p \in X | 0 \leq p \leq L - 1 \wedge inteiro\} \tag{4.14}$$

$$Y = \{q \in Y | 0 \leq q \leq W - 1 \wedge inteiro\} \tag{4.15}$$

$$Z = \{r \in Z | 0 \leq r \leq H - 1 \wedge inteiro\} \tag{4.16}$$

$$X_{ig} = \{p \in X | 0 \leq p \leq L - l_{ig} \wedge inteiro\} \tag{4.17}$$

$$Y_{ig} = \{q \in Y | 0 \leq q \leq W - w_{ig} \wedge inteiro\} \tag{4.18}$$

$$Z_{ig} = \{r \in Z | 0 \leq r \leq H - h_{ig} \wedge inteiro\} \tag{4.19}$$

Onde o operador $c_{igpqrstr'}$ diz se um caixa do tipo i e rotação g alocada em (p, q, r) contém o ponto (s, t, r') na sua superfície superior. Neste modelo o valor v_i associado às variáveis de decisão corresponde ao volume da caixa, portanto este problema consiste da maximização do volume de caixas dentro do contêiner. O modelo apresentado anteriormente é implementado através da função *model_generator* que se encontra na seção de apêndices deste relatório.

Prosseguindo chega-se então a função referente ao solver. O solver é responsável por encontrar a solução ótima para o problema de programação linear obtido com o *model_generator*. Vale ressaltar que na etapa que engloba tanto a modelagem quanto a resolução do problema, as restrições foram introduzidas individualmente para que o solver pudesse então evoluir para contemplar as novas características. No modelo inicial foi considerado apenas as restrições de sobreposição e quantidades de itens. Observando-se a modelagem resultante dessas restrições, verificou-se um problema de PL com restrições apenas do tipo ≤. Para esta classe de problemas pode-se então utilizar o simplex tradicional, que por sua vez foi implementado na função *spxPadrao* disponível na seção de apêndices deste trabalho juntamente com a função *model_generator* desta etapa.

Na sequência foi introduzida a rotação de caixas. É importante relevar que a rotação de caixas é introduzida na função *problem_generator*, porém esta modificação possui profunda relação com a modelagem, sendo que alterações tiveram de ser feitas na função *model_generator* para lidar com a quantidade extra de combinações possíveis para a alocação. Nesta etapa o solver permaneceu o mesmo da etapa anterior.

Por último foi introduzida à restrição de estabilidade vertical através da equação (4.13). Esta restrição foi implementada no âmbito da função *model_generator*, porém deve-se ressaltar que a natureza dessa restrição ($\geq$) não pode mais ser resolvida em se utilizando o simplex tradicional. Para tanto, foi estudado e implementado o simplex de duas fases na função *simplexDuasFases* que se encontra na seção de apêndices. Mas com um estudo meticuloso sobre as características do problema, chegou-se a conclusão de que a metodologia do simplex de duas fases não deveria ser levada na integra.

A primeira fase deste algoritmo consiste na determinação de uma solução básica inicial e a segunda corresponde à mudança da função objetivo e a resolução do problema resultante. Para o problema em questão uma das soluções básicas iniciais sempre será o contêiner vazio, e então o algoritmo do simplex de duas fases deve ser ajustado apenas para lidar com as folgas e sobras nas inequações do problema de maneira adequada. Pelo fato de que este solver consegue resolver restrições do tipo $\geq$ sem utilizar a metodologia do simplex duas fases na integra, foi criada uma nova função denominada *simplex1FaseEMeia,* cujo código se encontra na seção de apêndices desse relatório.

Até então o resultado do solver não poderia ser visualizado de maneira gráfica, portanto foi desenvolvida uma função denominada *drawResult* para este fim. A função encontra-se na seção de apêndices. A Fig. 4.5 mostra um exemplo de saída da função *drawResult*.

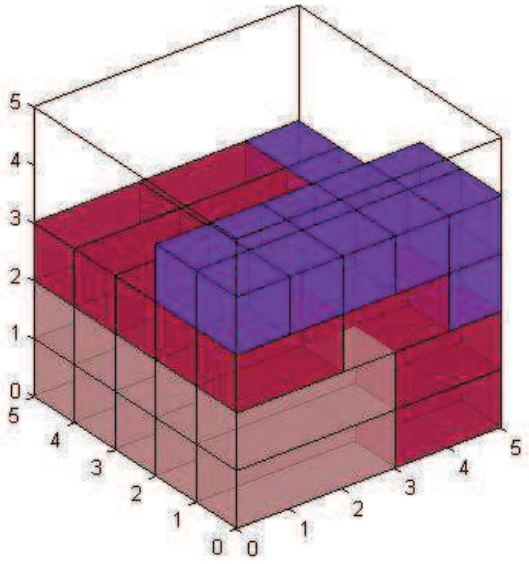


Figura 4.5 – Exemplo da saída da função *drawResult*.

Para visualização do funcionamento do algoritmo foram realizados alguns testes que serão mostrados na sequência. Para estes testes considerou-se sempre um número de caixas superior ao necessário para o enchimento do contêiner, para que desta forma, o algoritmo pudesse trabalhar as combinação na obtenção de um padrão ótimo. O computador utilizado trata-se de um Core i5 2.4 GHz e 4Gb de RAM.

Tabela 4.1 – Problema 1.

Características do Problema 1	
Contêiner	
Comprimento	5
Largura	4
Altura	3
Caixas	
Tipo	1
Comprimento	1
Largura	1
Altura	1
Quantidade	100

Tabela 4.2 – Resposta para o problema 1.

Características da resposta	
Tempo de construção do modelo [seg]	0.069166
Tempo de resolução [seg]	0.13272
Possibilidades de combinação das caixas	60
Variáveis no problema de PL	161
Porcentagem de utilização do volume [%]	100

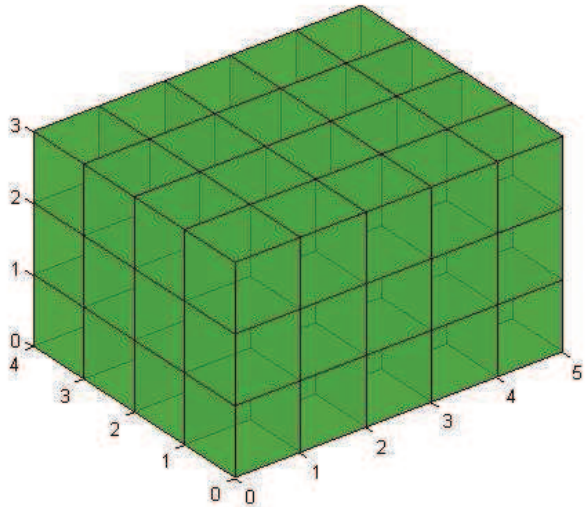


Figura 4.6 – Padrão de empacotamento do problema 1.

Tabela 4.3 – Problema 2.

Características do Problema 2		
Contêiner		
Comprimento	5	
Largura	4	
Altura	3	
Caixas		
Tipo	1	2
Comprimento	1	2
Largura	1	1
Altura	1	1
Quantidade	40	40

Tabela 4.4 – Resposta para o problema 2.

Características da resposta	
Tempo de construção do modelo [seg]	0.29615
Tempo de resolução [seg]	0.321446
Possibilidades de combinação das caixas	193
Variáveis no problema de PL	439
Porcentagem de utilização do volume [%]	100

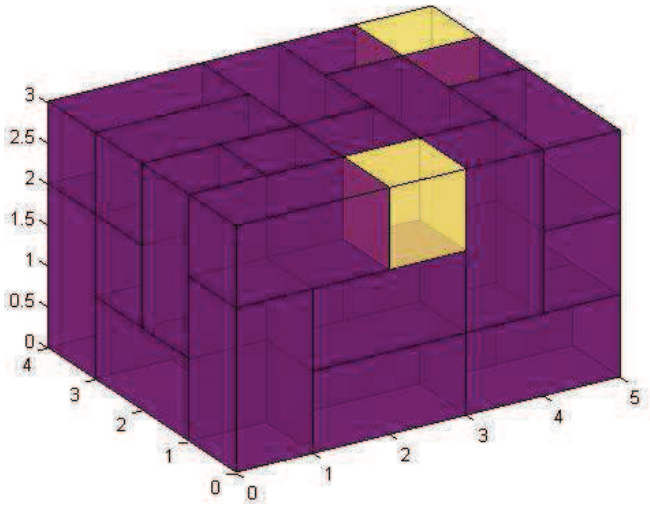


Figura 4.7 – Padrão de empacotamento do problema 2.

Tabela 4.5 – Problema 3.

Características do Problema 3		
Contêiner		
Comprimento	5	
Largura	5	
Altura	5	
Caixas		
Tipo	1	2
Comprimento	1	2
Largura	1	1
Altura	1	1
Quantidade	100	40

Tabela 4.6 – Resposta para o problema 3.

Características da resposta	
Tempo de construção do modelo [seg]	1.498444
Tempo de resolução [seg]	0.321446
Possibilidades de combinação das caixas	425
Variáveis no problema de PL	1047
Porcentagem de utilização do volume [%]	100

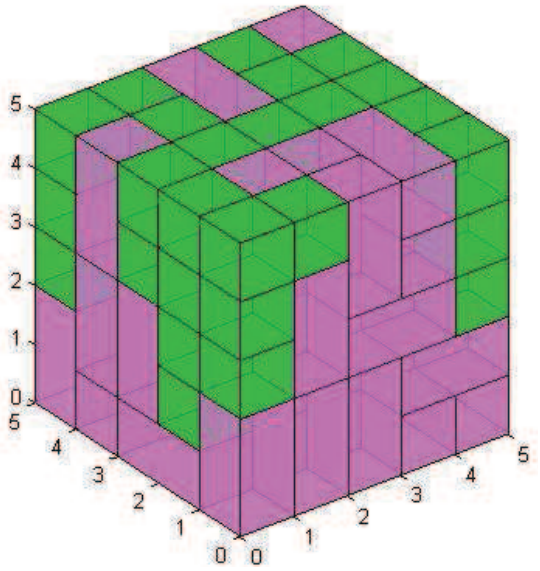


Figura 4.8 – Padrão de empacotamento do problema 3.

Tabela 4.7 – Problema 4.

Características do Problema 4			
Contêiner			
Comprimento	5		
Largura	5		
Altura	5		
Caixas			
Tipo	1	2	3
Comprimento	1	2	3
Largura	1	1	1
Altura	1	1	1
Quantidade	100	100	100

Tabela 4.8 – Resposta para o problema 4.

Características da resposta	
Tempo de construção do modelo [seg]	4.387734
Tempo de resolução [seg]	8.583115
Possibilidades de combinação das caixas	650
Variáveis no problema de PL	1683
Porcentagem de utilização do volume [%]	100

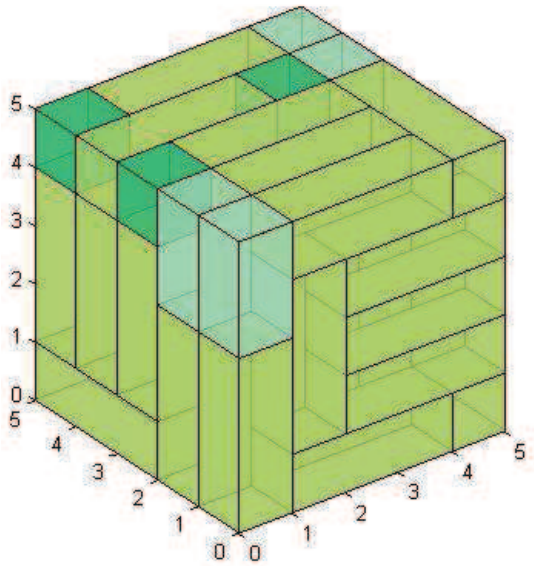


Figura 4.9 – Padrão de empacotamento do problema 4.

Tabela 4.9 – Problema 5.

Características do Problema5			
Contêiner			
Comprimento	10		
Largura	5		
Altura	5		
Caixas			
Tipo	1	2	3
Comprimento	1	2	2
Largura	1	1	2
Altura	1	1	1
Quantidade	300	300	300

Tabela 4.10 – Resposta para o problema 5.

Características da resposta	
Tempo de construção do modelo [seg]	29.316604
Tempo de resolução [seg]	86.36035
Possibilidades de combinação das caixas	1395
Variáveis no problema de PL	3764
Porcentagem de utilização do volume [%]	100

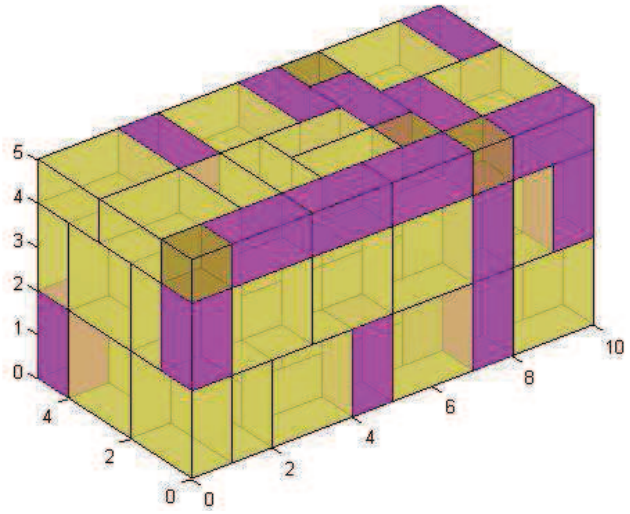


Figura 4.10 – Padrão de empacotamento do problema 5.

Os resultados obtidos revelaram que o modelo tem a capacidade de representar de forma satisfatória as características do PCC. Os tempos para elaboração do modelo e resolução do problema foram relativamente baixos considerando-se a variabilidade e as dimensões utilizadas. Para efeitos de produção de padrões de empacotamento a ser implementados em ambientes reais, o algoritmo atingiu o seu objetivo. De maneira mais técnica observa-se que a complexidade do modelo em termos de variáveis e possibilidades de alocação, variam diretamente com as dimensões do contêiner aliado as características dimensionais das caixas. Vale ressaltar que, quanto mais assimétrica a caixa, maiores são as possibilidades de rotação e, portanto, maior a complexidade do problema. Outra característica relevante trata-se da tendência de alocação de caixas maiores. Este comportamento é resultado do valor associado a cada caixa na função objetivo e, portanto, caixas com volumes maiores tendem a contribuir mais para o crescimento em volume ocupado da função objetivo. Caso se deseje que não haja uma prioridade entre as caixas basta então atribuir um valor idêntico para todas elas.

Na sequência foi então construído a função que elabora o código PDDL para integração com o planejamento automático. Os detalhes dos modelos estudados já foram apresentados na seção da proposta de trabalho. Através destes modelos, pode-se então avaliar a construção de uma função que seja capaz de criar modelos em PDDL de maneira sistemática. A metodologia consistiu da observação das diferenças obtidas nos domínio de caixas de diferentes dimensões, e a posterior construção das funções denominadas *createPDDL* e *createPDDL2*. Estas funções, através dos detalhes obtidos com a solução em simplex, são capazes de gerar os domínios e problemas referente à situação em trabalho. Para exemplificar o funcionamento dessas funções foi criado o problema abaixo, cujos arquivos PDDL são mostrados logo em sequência.

Tabela 4.11 – Problema exemplo da função *createPDDL* .

Características do Problema	
Contêiner	
Comprimento	2
Largura	2
Altura	2
Caixas	
Tipo	1
Comprimento	1
Largura	1
Altura	1
Quantidade	40

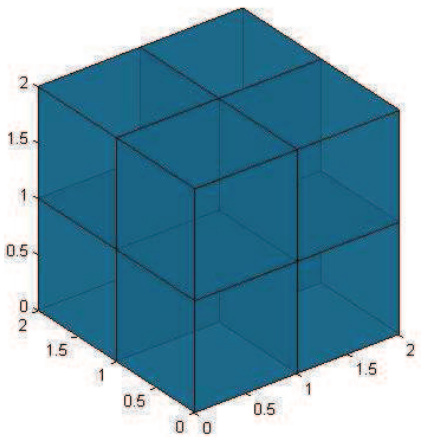


Figura 4.11 – Exemplo para a função *createPDDL*.

Tabela 4.12 – Saída da função *createPDDL* (domínio.pddl).

Domínio completo

```
(define (domain New_Project_1)
    (:requirements :typing :fluents :negative-preconditions :equality)
    (:types
      ponteRolante - object
      lugar - object
      produto1 - object
    )
    (:predicates
      (estaEm1 ?pro - produto1 ?lug - lugar)
      (estaNa1 ?pro - produto1 ?pon - ponteRolante)
      (ocupada ?pon - ponteRolante)
      (disponivel ?lug - lugar)
      (ocupado ?lug - lugar)
    )
    (:functions
      (coordx ?pon - ponteRolante)
      (coordy ?pon - ponteRolante)
      (crdx1 ?pro - produto1)
      (crdy1 ?pro - produto1)
      (crdz1 ?pro - produto1)
      (cx ?lug - lugar)
      (cy ?lug - lugar)
      (cz ?lug - lugar)
    )
    (:action mover
     :parameters (?p1 - ponteRolante ?l1 - lugar ?l2 - lugar)
     :precondition
       (and
         (= (coordx ?p1) (cx ?l1))
         (= (coordy ?p1) (cy ?l1))
         (not (= ?l1 ?l2))
         (= (cz ?l1) 0)
         (= (cz ?l2) 0)
       )
     :effect
       (and
         (assign (coordx ?p1) (cx ?l2))
         (assign (coordy ?p1) (cy ?l2))
       )
    )
    (:action pegar1
     :parameters (?p1 - ponteRolante ?prod - produto1 ?l1 - lugar ?l2 - lugar)
     :precondition
       (and
         (not (ocupada ?p1))
         (estaEm1 ?prod ?l1)
         (ocupado ?l1)
         (not (ocupado ?l2))
         (= (coordx ?p1) (cx ?l1))
         (= (coordy ?p1) (cy ?l1))
         (= (cx ?l1) (- (cx ?l2) 0))
         (= (cy ?l1) (- (cy ?l2) 0))
         (= (cz ?l1) (- (cz ?l2) 1))
       )
     :effect
       (and
         (estaNa1 ?prod ?p1)
         (not (estaEm1 ?prod ?l1))
         (ocupada ?p1)
         (not (ocupado ?l1))
         (not (disponivel ?l2))
         (assign (crdx1 ?prod) (- 0 1))
         (assign (crdy1 ?prod) (- 0 1))
         (assign (crdz1 ?prod) (- 0 1))
       )
```

```
    )
    (:action soltar1
     :parameters (?p1 - ponteRolante ?prod - produto1 ?l1 - lugar ?l2 - lugar)
     :precondition
       (and
         (ocupada ?p1)
         (estaNa1 ?prod ?p1)
         (not (ocupado ?l1))
         (not (ocupado ?l2))
         (disponivel ?l1)
         (= (coordx ?p1) (cx ?l1))
         (= (coordy ?p1) (cy ?l1))
         (= (cx ?l1) (- (cx ?l2) 0))
         (= (cy ?l1) (- (cy ?l2) 0))
         (= (cz ?l1) (- (cz ?l2) 1))
       )
     :effect
       (and
         (not (estaNa1 ?prod ?p1))
         (estaEm1 ?prod ?l1)
         (not (ocupada ?p1))
         (ocupado ?l1)
         (disponivel ?l2)
         (assign (crdx1 ?prod) (cx ?l1))
         (assign (crdy1 ?prod) (cy ?l1))
         (assign (crdz1 ?prod) (cz ?l1))
       )
    )
)
```

Domínio simplificado

```
(define (domain New_Project_1)
    (:requirements :strips)
    (:types
      lugar - object
      produto1 - object
    )
    (:predicates
      (estaEm1 ?pro - produto1 ?lug - lugar)
      (plivre1 ?prod - produto1)
      (atras ?lug - lugar ?lug - lugar)
      (esquerda ?lug - lugar ?lug - lugar)
      (acima ?lug - lugar ?lug - lugar)
      (livre ?lug - lugar)
      (piso ?lug - lugar)
    )
    (:action moverPiso1
     :parameters (?prod - produto1 ?l1 - lugar)
     :precondition
       (and
         (plivre1 ?prod)
         (livre ?l1)
         (piso ?l1)
       )
     :effect
       (and
         (estaEm1 ?prod ?l1)
         (not (plivre1 ?prod))
         (not (livre ?l1))
       )
    )
    (:action empilhar1
     :parameters (?prod - produto1 ?l1 - lugar ?l2 - lugar)
     :precondition
       (and
         (plivre1 ?prod)
         (not (livre ?l1))
         (acima ?l2 ?l1)
       )
```

```
      :effect
        (and
          (estaEm1 ?prod ?l2)
          (not (plivre1 ?prod))
          (not (livre ?l2))
        )
    )
)
```

Tabela 4.13 – Saída da função *createPDDL* (problema.pddl).

Problema para modelagem completa

```
(define (problem Planning_Problem)
    (:domain New_Project_1)
    (:objects
      p1 - ponteRolante
      l1 - lugar
      l2 - lugar
      l3 - lugar
      l4 - lugar
      l5 - lugar
      l6 - lugar
      l7 - lugar
      l8 - lugar
      l9 - lugar
      l10 - lugar
      l11 - lugar
      l12 - lugar
      prod1 - produto1
      l13 - lugar
      l14 - lugar
      prod2 - produto1
      l15 - lugar
      l16 - lugar
      prod3 - produto1
      l17 - lugar
      l18 - lugar
      prod4 - produto1
      l19 - lugar
      l20 - lugar
      prod5 - produto1
      l21 - lugar
      l22 - lugar
      prod6 - produto1
      l23 - lugar
      l24 - lugar
      prod7 - produto1
      l25 - lugar
      l26 - lugar
      prod8 - produto1
      l27 - lugar
      l28 - lugar
    )
    (:init
      (= (coordx p1) -10)
      (= (coordy p1) -10)
      (disponivel l1)
      (= (cx l1) 0)
      (= (cy l1) 0)
      (= (cz l1) 0)
      (disponivel l2)
      (= (cx l2) 1)
      (= (cy l2) 0)
      (= (cz l2) 0)
      (disponivel l3)
      (= (cx l3) 0)
```

```
        (= (cy l3) 1)
        (= (cz l3) 0)
        (disponivel l4)
        (= (cx l4) 1)
        (= (cy l4) 1)
        (= (cz l4) 0)
        (= (cx l5) 0)
        (= (cy l5) 0)
        (= (cz l5) 1)
        (= (cx l6) 1)
        (= (cy l6) 0)
        (= (cz l6) 1)
        (= (cx l7) 0)
        (= (cy l7) 1)
        (= (cz l7) 1)
        (= (cx l8) 1)
        (= (cy l8) 1)
        (= (cz l8) 1)
        (= (cx l9) 0)
        (= (cy l9) 0)
        (= (cz l9) 2)
        (= (cx l10) 1)
        (= (cy l10) 0)
        (= (cz l10) 2)
        (= (cx l11) 0)
        (= (cy l11) 1)
        (= (cz l11) 2)
        (= (cx l12) 1)
        (= (cy l12) 1)
        (= (cz l12) 2)
        (estaEm1 prod1 l13)
        (= (crdx1 prod1) -10)
        (= (crdy1 prod1) -10)
        (= (crdz1 prod1) 0)
        (ocupado l13)
        (disponivel l13)
        (= (cx l13) -10)
        (= (cy l13) -10)
        (= (cz l13) 0)
        (disponivel l14)
        (= (cx l14) -10)
        (= (cy l14) -10)
        (= (cz l14) 1)
        (estaEm1 prod2 l15)
        (= (crdx1 prod2) -10)
        (= (crdy1 prod2) -10)
        (= (crdz1 prod2) 0)
        (ocupado l15)
        (disponivel l15)
        (= (cx l15) -10)
        (= (cy l15) -10)
        (= (cz l15) 0)
        (disponivel l16)
        (= (cx l16) -10)
        (= (cy l16) -10)
        (= (cz l16) 1)
        (estaEm1 prod3 l17)
        (= (crdx1 prod3) -10)
        (= (crdy1 prod3) -10)
        (= (crdz1 prod3) 0)
        (ocupado l17)
        (disponivel l17)
        (= (cx l17) -10)
        (= (cy l17) -10)
        (= (cz l17) 0)
        (disponivel l18)
        (= (cx l18) -10)
        (= (cy l18) -10)
```

```
      (= (cz l18) 1)
      (estaEm1 prod4 l19)
      (= (crdx1 prod4) -10)
      (= (crdy1 prod4) -10)
      (= (crdz1 prod4) 0)
      (ocupado l19)
      (disponivel l19)
      (= (cx l19) -10)
      (= (cy l19) -10)
      (= (cz l19) 0)
      (disponivel l20)
      (= (cx l20) -10)
      (= (cy l20) -10)
      (= (cz l20) 1)
      (estaEm1 prod5 l21)
      (= (crdx1 prod5) -10)
      (= (crdy1 prod5) -10)
      (= (crdz1 prod5) 0)
      (ocupado l21)
      (disponivel l21)
      (= (cx l21) -10)
      (= (cy l21) -10)
      (= (cz l21) 0)
      (disponivel l22)
      (= (cx l22) -10)
      (= (cy l22) -10)
      (= (cz l22) 1)
      (estaEm1 prod6 l23)
      (= (crdx1 prod6) -10)
      (= (crdy1 prod6) -10)
      (= (crdz1 prod6) 0)
      (ocupado l23)
      (disponivel l23)
      (= (cx l23) -10)
      (= (cy l23) -10)
      (= (cz l23) 0)
      (disponivel l24)
      (= (cx l24) -10)
      (= (cy l24) -10)
      (= (cz l24) 1)
      (estaEm1 prod7 l25)
      (= (crdx1 prod7) -10)
      (= (crdy1 prod7) -10)
      (= (crdz1 prod7) 0)
      (ocupado l25)
      (disponivel l25)
      (= (cx l25) -10)
      (= (cy l25) -10)
      (= (cz l25) 0)
      (disponivel l26)
      (= (cx l26) -10)
      (= (cy l26) -10)
      (= (cz l26) 1)
      (estaEm1 prod8 l27)
      (= (crdx1 prod8) -10)
      (= (crdy1 prod8) -10)
      (= (crdz1 prod8) 0)
      (ocupado l27)
      (disponivel l27)
      (= (cx l27) -10)
      (= (cy l27) -10)
      (= (cz l27) 0)
      (disponivel l28)
      (= (cx l28) -10)
      (= (cy l28) -10)
      (= (cz l28) 1)
    )
    (:goal
```

```
      (and
        (not (ocupada p1))
        (= (crdx1 prod1) 0)
        (= (crdy1 prod1) 0)
        (= (crdz1 prod1) 0)
        (= (crdx1 prod2) 1)
        (= (crdy1 prod2) 0)
        (= (crdz1 prod2) 0)
        (= (crdx1 prod3) 0)
        (= (crdy1 prod3) 1)
        (= (crdz1 prod3) 0)
        (= (crdx1 prod4) 1)
        (= (crdy1 prod4) 1)
        (= (crdz1 prod4) 0)
        (= (crdx1 prod5) 0)
        (= (crdy1 prod5) 0)
        (= (crdz1 prod5) 1)
        (= (crdx1 prod6) 1)
        (= (crdy1 prod6) 0)
        (= (crdz1 prod6) 1)
        (= (crdx1 prod7) 0)
        (= (crdy1 prod7) 1)
        (= (crdz1 prod7) 1)
        (= (crdx1 prod8) 1)
        (= (crdy1 prod8) 1)
        (= (crdz1 prod8) 1)
      )
    )
)
```

Problema para modelagem simplificada

```
(define (problem Planning_Problem)
    (:domain New_Project_1)
    (:objects
      l1 - lugar
      l2 - lugar
      l3 - lugar
      l4 - lugar
      l5 - lugar
      l6 - lugar
      l7 - lugar
      l8 - lugar
      prod1 - produto1
      prod2 - produto1
      prod3 - produto1
      prod4 - produto1
      prod5 - produto1
      prod6 - produto1
      prod7 - produto1
      prod8 - produto1
    )
    (:init
      (livre l1)
      (piso l1)
      (livre l2)
      (piso l2)
      (livre l3)
      (piso l3)
      (livre l4)
      (piso l4)
      (livre l5)
      (livre l6)
      (livre l7)
      (livre l8)
      (atras l2 l1)
      (atras l4 l3)
      (atras l6 l5)
      (atras l8 l7)
      (esquerda l3 l1)
```

```
      (esquerda l4 l2)
      (esquerda l7 l5)
      (esquerda l8 l6)
      (acima l5 l1)
      (acima l6 l2)
      (acima l7 l3)
      (acima l8 l4)
      (plivre1 prod1)
      (plivre1 prod2)
      (plivre1 prod3)
      (plivre1 prod4)
      (plivre1 prod5)
      (plivre1 prod6)
      (plivre1 prod7)
      (plivre1 prod8)
    )
    (:goal
      (and
        (estaEm1 prod1 l1)
        (estaEm1 prod2 l2)
        (estaEm1 prod3 l3)
        (estaEm1 prod4 l4)
        (estaEm1 prod5 l5)
        (estaEm1 prod6 l6)
        (estaEm1 prod7 l7)
        (estaEm1 prod8 l8)
      )
    )
)
```

Da mesma forma que as funções anteriores, o código encontra-se na seção de apêndices deste trabalho.

Em seguida as funções *createPDDL,* conforme exposto anteriormente, foi utilizada para criação de alguns problemas a serem resolvidos em se utilizando o planejamento automático. Vale ressaltar que nesta fase do trabalho os planejadores automáticos inicialmente foram pesquisados através dos *requirements* necessários para a resolução do problema. Para a modelagem completa são necessários os *requirements*: *typing,* que permite a declaração de nomes de tipos para as variáveis; *fluents,* que trabalha com avaliação de expressões; *negative-preconditions,* que permite a utilização do operador de negação; e por fim *equality,* que permite relações de igualdade. Para a modelagem simplificada é necessário apenas o *requirement* do tipo *strips*, que são basicamento operadores de lógica booleana.

Para a modelagem completa, através desta avaliação inicial, foi encontrado um conjunto muito restrito de planejadores e que ainda assim não garantiam a capacidade de resolução do problema. Então, por meio de alguns testes aleatórios, descobriu-se que apesar do declarado no conjunto dos *requirements,* alguns planejadores eram de fato capazes de lidar com condições extras. Por esta razão,decidiu-se pela realização de

testes com um conjunto de todos os planejadores mais atuais, apresentados na sétima e sexta IPC (*International Planning Competition*), realizadas respectivamente nos anos de 2011 e 2008 e mais alguns outros. Com relação ao modelo simplificado, devido a natureza reduzida de sua modelagem, em teoria pode ser abordado pela grande maioria dos planejadores. Conforme os planejadores do IPC 2011 e 2008 foram utilizados na modelagem completa, estes foram reutilizados na modelagem simplificada à titulo de comparação de desempenho.

Para avaliar a capacidade de resolução dos planejadores foram elaborados quatro problemas os quais são mostrados pelas figuras a seguir:

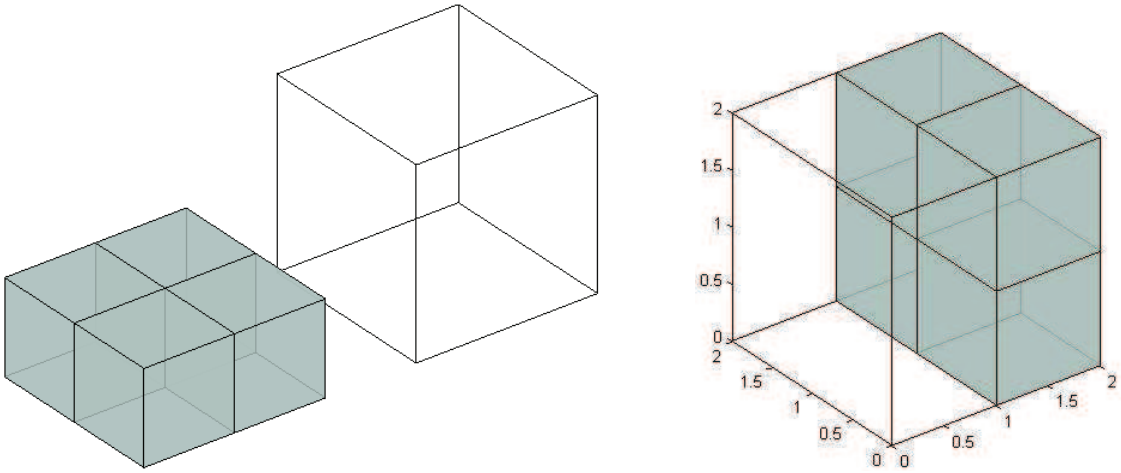


Figura 4.12 – Problema 1, snapshot inicial (esquerda) e snapshot final (direita).

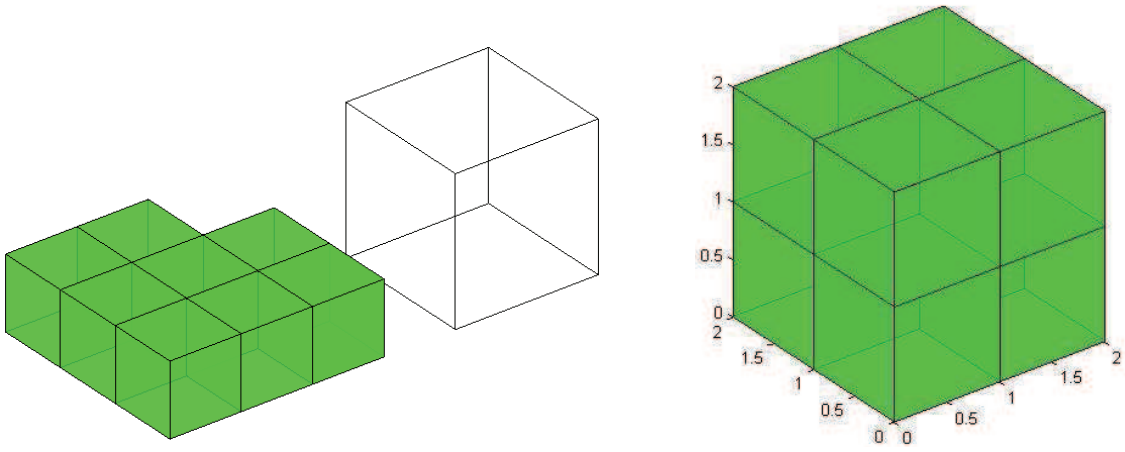


Figura 4.13 – Problema 2, snapshot inicial (esquerda) e snapshot final (direita).

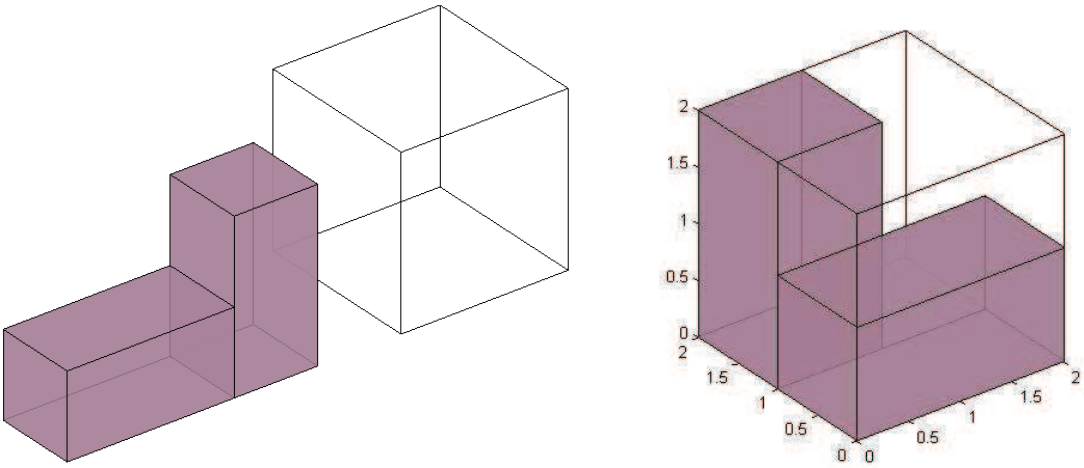


Figura 4.14 – Problema 3, snapshot inicial (esquerda) e snapshot final (direita).

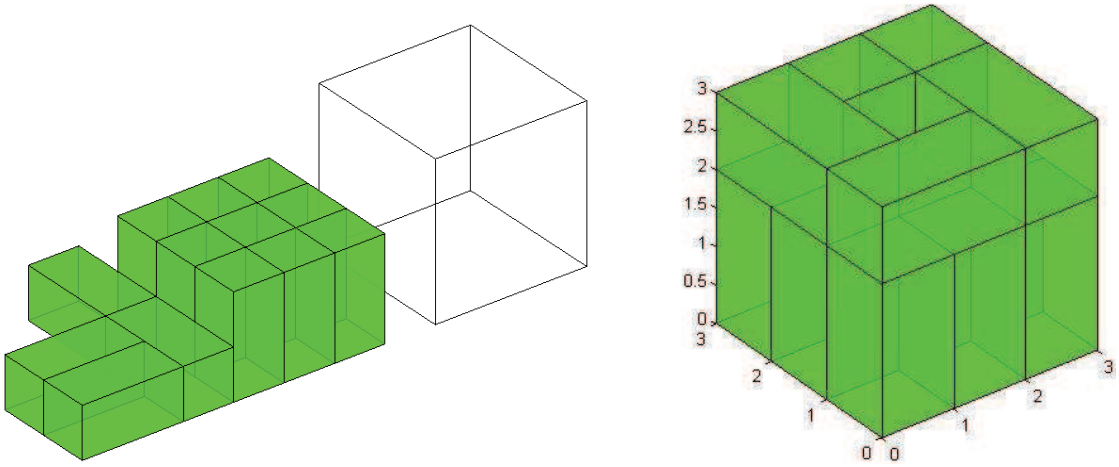


Figura 4.15 – Problema 4, snapshot inicial (esquerda) e snapshot final (direita).

Para maiores informações a respeito dos problemas apresentados, os arquivos em pddl do domínio e do problema encontram-se na seção de apêndices desta dissertação. Na tentativa de se resolver os problemas apresentados anteriormente foi obtida a seguinte lista de planejadores.

- *BJOLP: The Big Joint Optimal Landmarks planner* (DOMSHLAK, C. et al., 2011.)
- *CPT4: Constraint Programming Temporal planner* (VIDAL, V., 2011a.)
- *FD-Autotune: Fast downward Autotune planner* (FAWCETT, C. et al., 2011)
- *FD-Autotune 1: Fast downward Autotune 1 planner* (FAWCETT, C. et al., 2011)

- *FD-Autotune 2: Fast downward Autotune 2 planner* (FAWCETT, C. et al., 2011)
- *FDSS1: Fast Downward Stone Soup 1 planner* (HELMERT, M. et al.; 2011)
- *FDSS2: Fast Downward Stone Soup 2 planner* (HELMERT, M. et al.; 2011)
- *ForkInit planner* (KATZ, M.; DOMSHLAK, C., 2011)
- *Gamer planner* (KISSMANN, P.; EDELKAMP, S., 2011)
- *Iforkinit planner* (KATZ, M.; DOMSHLAK, C., 2011)
- *LM-Cut: Landmark-Cut Heuristic planner* (HELMERT. M.; DOMSHLAK, C., 2011)
- *LMfork: Landmark Fork planner* (KATZ, M.; DOMSHLAK, C., 2011)
- *Merge-and-Shrink planner* (NISSIM, R.; HOFFMANN. J.; HELMERT, M., 2011)
- *SelMax: Selective Max* (DOMSHLAK, C. et al., 2011)
- *ACOPlan: Ant Colony Optimization planner* (BAIOLETTI, M. et al., 2011)
- *ACOPlan 2: Ant Colony Optimization 2 planner* (BAIOLETTI, M. et al., 2011)
- *Arvand planner* (NAKHOST, H. et al., 2011)
- *BRT: Biased Rapidly-exploring Tree planner* (ALCÁZAR, V.; VELOSO, M., 2011)
- *CBP: Cost-Based Planner* (FUENTETAJA, R., 2011)
- *CBP2: Cost-Based Planner 2* (FUENTETAJA, R., 2011)
- *DAE-YAHSPplanner* (DRÉO, J. et al., 2011)
- *Forkuniform planner* (KATZ, M.; DOMSHLAK, C., 2011)
- *Lama 2008 planner* (RICHTER, S.; WESTPHAL, M.; HELMERT, M., 2011)
- *Lama 2011 planner* (RICHTER, S.; WESTPHAL, M.; HELMERT, M., 2011)
- *Lamar planner* (OLSEN, A.; BRYCE, D., 2011)
- *LPRPG planner* (COLES, A. et al., 2011a)
- *Madagascar planner* (RINTANEN, J., 2011)

- *Madagascar –p planner* (RINTANEN, J., 2011)
- *POPF2: Forward-Chaining Partial Order Planner* (COLES, A. et al., 2011b)
- *Probe planner* (LIPOVETZKY, N.; GEFFNER, H., 2011)
- *Randward planner* (OLSEN, A.; BRYCE, D., 2011)
- *Roamer planner* (LU, Q., et al., 2011)
- *SatPlanLM-c planner* (CAI, D., 2011)
- *Sharaabi planner* (KAVULURI, B. R., 2011)
- *YAHSP2 planner* (VIDAL, V., 2011b)
- *YAHSP2-MT planner* (VIDAL, V., 2011b)
- *C3: Consistent Causal Chains planner* (LIPOVETZKY, N.; RAMÍREZ, M.; GEFFNER, H, 2008)
- *DAE1: Divide-and-Evolve 1 planner* (BIBAÏ, J., 2008)
- *DAE2: Divide-and-Evolve 2 planner* (BIBAÏ, J., 2008)
- *DTG-Plan: Domain Transition Graphs planner* (HUANG, R.; CHEN, Y.; ZHANG, W., 2008)
- *FF(ha) planner* (KEYDER, E.; GEFFNER, H., 2008)
- *FF(sa) planner* (KEYDER, E.; GEFFNER, H., 2008)
- *Lama planner* (RICHTER, S.; WESTPHAL, M., 2008)
- *Plan-A planner* (CHEN, Y.; LV, Q.; HUANG, R, 2008)
- *SGPlan6 planner* (HSU, C. W.; WAH, B. W., 2008)
- *Upwards planner* (COLES, A.; SMITH, A., 2008)
- *CFDP planner* (GRANDCOLAS, S.; BARRE, C. P., 2008)
- *CO-PLAN planner (*ROBINSON, N.; GRETTON, C.; PHAM, D. N., 2008*)*
- *Gamer planner* (EDELKAMP, S.; KISSMANN, P., 2008)
- *Hsps0 planner* (HASLUM, P., 2008)
- *Hspsf planner* (HASLUM, P., 2008)
- *Hspsp planner* (HASLUM, P., 2008)
- *Mips XXL planner* (EDELKAMP, S.; JABBAR, S., 2008)
- *Temporal Fast Downward planner* (RÖGER, G.; EYERICH, P.; MATTMÜLLER, R., 2008)

- *TLP-GP planner* (MARIS, F.; RÉGNIER, P., 2008)
- *Metric FF planner* (HOFFMANN, J., 2003)
- *SGPlan 5.2.2 planner* (HSU, C. W. et al., 2006)

Para cada um dos planejadores listados acima foi obtido o código fonte, disponível para download no site das respectivas competições ou pesquisadores. Portanto para utilização dos mesmos foi necessário à realização do processo de compilação. Vale ressaltar que o ambiente de desenvolvimento do mesmo trata-se do sistema Linux, dessa forma os planejadores foram compilados utilizando-se do Ubuntu 12.04 LTS. O planejador SGPlan 5.2.2 foi encontrado na forma de executável, portanto não foi necessário o processo de compilação.

Nesta etapa houve alguns problemas de compilação, o que excluiu alguns dos planejadores da etapa seguinte, que trata-se dos testes com problemas criados. A lista abaixo mostra todos os planejadores que apresentaram algum problema na compilação. É importante lembrar que devido às particularidades de cada um dos planejadores no momento da compilação, não pode ser realizado um diagnóstico mais aprofundado tendo em vista o objetivo deste trabalho.

Planejadores que apresentaram problema na compilação.

- *Gamer planner (IPC 2011)*
- *CPT4: Constraint Programming Temporal planner (IPC 2011)*
- *DAE-YAHSP planner (IPC 2011)*
- *POPF2: Forward-Chaining Partial Order Planner (IPC 2011)*
- *Probe planner (IPC 2011)*
- *Sharaabi planner (IPC 2011)*
- *Hspsppplanner (IPC 2008)*
- *Hspsf planner (IPC 2008)*
- *Hsps0planner (IPC 2008)*
- *CO-PLAN planner (IPC 2008)*
- *CPT3 planner (IPC 2008)*
- *C3: Consistent Causal Chains planner (IPC 2008)*
- *DAE1: Divide-and-Evolve 1 planner (IPC 2008)*
- *DAE2: Divide-and-Evolve 2 planner (IPC 2008)*
- *DTG-Plan: Domain Transition Graphs planner (IPC 2008)*
- *TLP-GPplanner (IPC 2008)*

Com os planejadores restantes foram realizados testes caracterizados pelos problemas expostos anteriormentes.

Inicialmente foram feitos os testes em se considerando a modelagem completa. Para esta modelagem a grande maioria dos planejadores não conseguiu lidar com o *requirement* de *fluents* sendo que apenas um conjunto de cinco planejadores forneceram planos para a execução do processo de carregamento. Os testes foram realizados em uma máquina virtual do Ubuntu 12.04 LTS 32 bits. O computador trata-se de um Core i5 de 2.4 GHz e 4Gb de RAM, sendo que para a máquina virtual foram disponibilizados 2 *cores* ou núcleos dos 4 disponíveis do processador e 2 Gb de RAM. Os resultados dos planejadores são mostrados a seguir:

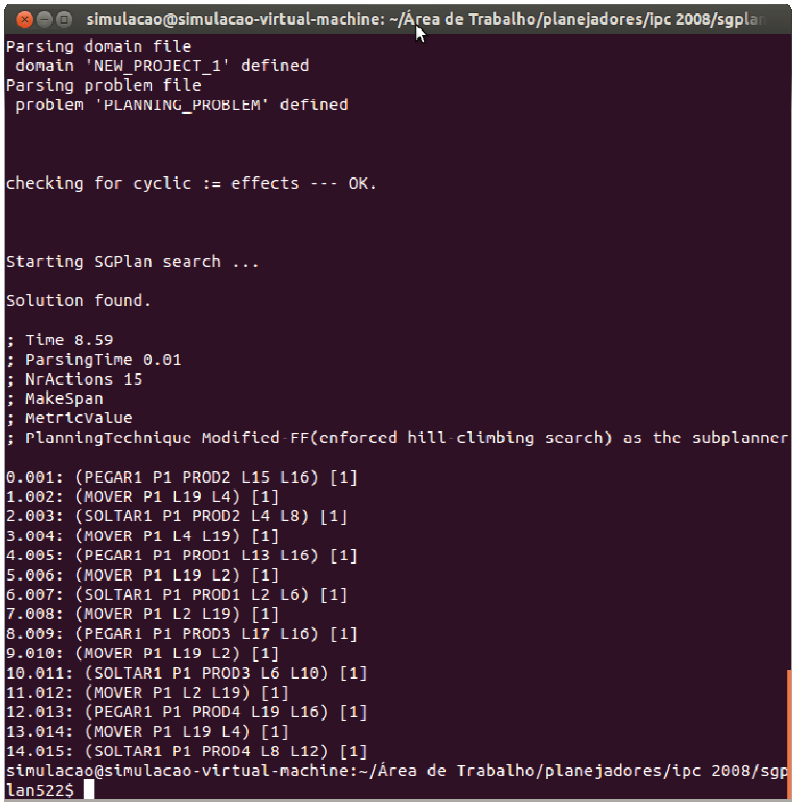


Figura 4.16 – Resposta do planejador SGPlan 5.2.2 para o problema 1 em linha comando.

Tabela 4.14 - Resultado do CBP para o problema 1.

CBP – Cost Based Planner
0: (PEGAR1 P1 PROD2 L15 L20)
1: (MOVER P1 L19 L4)
2: (SOLTAR1 P1 PROD2 L4 L8)
3: (MOVER P1 L4 L19)

4: (PEGAR1 P1 PROD1 L13 L20)
5: (MOVER P1 L19 L2)
6: (SOLTAR1 P1 PROD1 L2 L6)
7: (MOVER P1 L2 L19)
8: (PEGAR1 P1 PROD3 L17 L20)
9: (MOVER P1 L19 L2)
10: (SOLTAR1 P1 PROD3 L6 L10)
11: (MOVER P1 L2 L19)
12: (PEGAR1 P1 PROD4 L19 L20)
13: (MOVER P1 L19 L4)
14: (SOLTAR1 P1 PROD4 L8 L12)
tempo total: 247,16 segundos
estados avaliados: 215017

Tabela 4.15 - Resultado do CBP para o problema 3.

CBP – Cost Based Planner
0: (PEGAR3 P1 PROD2 L17 L18 L19)
1: (MOVER P1 L17 L3)
2: (SOLTAR3 P1 PROD2 L3 L7 L11)
3: (MOVER P1 L3 L17)
4: (PEGAR2 P1 PROD1 L13 L14 L18 L16)
5: (MOVER P1 L17 L1)
6: (SOLTAR2 P1 PROD1 L1 L2 L5 L6)
tempo total: 0,56 segundos
estados avaliados: 597

Tabela 4.16 - Resultado do Metric FF para o problema 1.

Metric FF
0: PEGAR1 P1 PROD1 L13 L20
1: MOVER P1 L19 L4
2: SOLTAR1 P1 PROD1 L4 L8
3: MOVER P1 L4 L19
4: PEGAR1 P1 PROD2 L15 L20
5: MOVER P1 L19 L4
6: SOLTAR1 P1 PROD2 L8 L12
7: MOVER P1 L4 L19

8: PEGAR1 P1 PROD4 L19 L20
9: MOVER P1 L19 L2
10: SOLTAR1 P1 PROD4 L2 L6
11: MOVER P1 L2 L19
12: PEGAR1 P1 PROD3 L17 L20
13: MOVER P1 L19 L4
14: MOVER P1 L4 L2
15: SOLTAR1 P1 PROD3 L6 L10
16: MOVER P1 L2 L4
17: PEGAR1 P1 PROD2 L8 L12
18: MOVER P1 L4 L19
19: SOLTAR1 P1 PROD2 L19 L20
20: MOVER P1 L19 L4
21: PEGAR1 P1 PROD1 L4 L8
22: MOVER P1 L4 L19
23: SOLTAR1 P1 PROD1 L17 L20
24: PEGAR1 P1 PROD2 L19 L20
25: MOVER P1 L19 L4
26: SOLTAR1 P1 PROD2 L4 L8
27: MOVER P1 L4 L19
28: PEGAR1 P1 PROD1 L17 L20
29: SOLTAR1 P1 PROD1 L19 L20
30: MOVER P1 L19 L2
31: PEGAR1 P1 PROD3 L6 L10
32: MOVER P1 L2 L4
33: SOLTAR1 P1 PROD3 L8 L12
34: MOVER P1 L4 L19
35: MOVER P1 L19 L2
36: PEGAR1 P1 PROD4 L2 L6
37: MOVER P1 L2 L19
38: SOLTAR1 P1 PROD4 L17 L20
39: PEGAR1 P1 PROD1 L19 L20
40: MOVER P1 L19 L2
41: SOLTAR1 P1 PROD1 L2 L6
42: MOVER P1 L2 L4
43: PEGAR1 P1 PROD3 L8 L12

44: MOVER P1 L4 L19
45: MOVER P1 L19 L2
46: SOLTAR1 P1 PROD3 L6 L10
47: MOVER P1 L2 L19
48: PEGAR1 P1 PROD4 L17 L20
49: MOVER P1 L19 L4
50: SOLTAR1 P1 PROD4 L8 L12
tempo total: 14,46 segundos
estados avaliados: 117105

Tabela 4.17 - Resultado do Metric FF para o problema 3.

Metric FF
0: PEGAR3 P1 PROD2 L17 L18 L19
1: MOVER P1 L17 L3
2: SOLTAR3 P1 PROD2 L3 L7 L11
3: MOVER P1 L3 L17
4: PEGAR2 P1 PROD1 L13 L14 L18 L16
5: MOVER P1 L17 L1
6: SOLTAR2 P1 PROD1 L1 L2 L5 L6
tempo total: 0,59 segundos
estados avaliados: 187

Tabela 4.18 - Resultado do SGPlan6 para o problema 1.

SGPlan6
0.001: (PEGAR1 P1 PROD4 L19 L20) [1]
1.002: (MOVER P1 L19 L4) [1]
2.003: (SOLTAR1 P1 PROD4 L4 L8) [1]
3.004: (MOVER P1 L4 L19) [1]
4.005: (PEGAR1 P1 PROD3 L17 L20) [1]
5.006: (MOVER P1 L19 L3) [1]
6.007: (SOLTAR1 P1 PROD3 L3 L7) [1]
7.008: (MOVER P1 L3 L4) [1]
8.009: (PEGAR1 P1 PROD4 L4 L8) [1]
9.010: (MOVER P1 L4 L3) [1]
10.011: (SOLTAR1 P1 PROD4 L7 L11) [1]

11.012: (MOVER P1 L3 L19) [1]
12.013: (PEGAR1 P1 PROD1 L13 L14) [1]
13.014: (MOVER P1 L19 L2) [1]
14.015: (SOLTAR1 P1 PROD1 L2 L6) [1]
15.016: (MOVER P1 L2 L19) [1]
16.017: (PEGAR1 P1 PROD2 L15 L20) [1]
17.018: (MOVER P1 L19 L4) [1]
18.019: (SOLTAR1 P1 PROD2 L4 L8) [1]
19.020: (MOVER P1 L4 L3) [1]
20.021: (PEGAR1 P1 PROD4 L7 L11) [1]
21.022: (MOVER P1 L3 L4) [1]
22.023: (SOLTAR1 P1 PROD4 L8 L12) [1]
23.024: (MOVER P1 L4 L3) [1]
24.025: (PEGAR1 P1 PROD3 L3 L7) [1]
25.026: (MOVER P1 L3 L2) [1]
26.027: (SOLTAR1 P1 PROD3 L6 L10) [1]
tempo total: 15,03 segundos

Tabela 4.19 - Resultado do SGPlan6 para o problema 3.

SGPlan6
0.001: (PEGAR3 P1 PROD2 L17 L18 L19) [1]
1.002: (MOVER P1 L17 L4) [1]
2.003: (SOLTAR3 P1 PROD2 L4 L8 L12) [1]
3.004: (PEGAR3 P1 PROD2 L4 L8 L12) [1]
4.005: (MOVER P1 L4 L3) [1]
5.006: (SOLTAR3 P1 PROD2 L3 L7 L11) [1]
6.007: (MOVER P1 L3 L4) [1]
7.008: (MOVER P1 L4 L17) [1]
8.009: (PEGAR2 P1 PROD1 L13 L14 L18 L16) [1]
9.010: (MOVER P1 L17 L1) [1]
10.011: (SOLTAR2 P1 PROD1 L1 L2 L5 L6) [1]
tempo total: 0,53 segundos

Tabela 4.20 - Resultado do SGPlan5.2.2 para o problema 1.

SGPlan5.2.2
0.001: (PEGAR1 P1 PROD2 L15 L16) [1]

1.002: (MOVER P1 L19 L4) [1]
2.003: (SOLTAR1 P1 PROD2 L4 L8) [1]
3.004: (MOVER P1 L4 L19) [1]
4.005: (PEGAR1 P1 PROD1 L13 L16) [1]
5.006: (MOVER P1 L19 L2) [1]
6.007: (SOLTAR1 P1 PROD1 L2 L6) [1]
7.008: (MOVER P1 L2 L19) [1]
8.009: (PEGAR1 P1 PROD3 L17 L16) [1]
9.010: (MOVER P1 L19 L2) [1]
10.011: (SOLTAR1 P1 PROD3 L6 L10) [1]
11.012: (MOVER P1 L2 L19) [1]
12.013: (PEGAR1 P1 PROD4 L19 L16) [1]
13.014: (MOVER P1 L19 L4) [1]
14.015: (SOLTAR1 P1 PROD4 L8 L12) [1]
tempo total: 8,59 segundos

Tabela 4.21 - Resultado do SGPlan5.2.2 para o problema 3.

SGPlan5.2.2
0.001: (PEGAR3 P1 PROD2 L17 L18 L19) [1]
1.002: (MOVER P1 L17 L3) [1]
2.003: (SOLTAR3 P1 PROD2 L3 L7 L11) [1]
3.004: (MOVER P1 L3 L17) [1]
4.005: (PEGAR2 P1 PROD1 L13 L14 L18 L16) [1]
5.006: (MOVER P1 L17 L1) [1]
6.007: (SOLTAR2 P1 PROD1 L1 L2 L5 L6) [1]
tempo total: 2,62 segundos

Tabela 4.22 - Resultado do Mips-XXL para o problema 1.

Mips-XXL
0: (PEGAR1 P1 PROD1 L13 L20)
1: (MOVER P1 L19 L4)
2: (SOLTAR1 P1 PROD1 L4 L8)
3: (MOVER P1 L4 L19)
4: (PEGAR1 P1 PROD2 L15 L20)
5: (MOVER P1 L19 L4)
6: (SOLTAR1 P1 PROD2 L8 L12)

7: (MOVER P1 L4 L19)
8: (PEGAR1 P1 PROD4 L19 L20)
9: (MOVER P1 L19 L2)
10: (SOLTAR1 P1 PROD4 L2 L6)
11: (MOVER P1 L2 L19)
12: (PEGAR1 P1 PROD3 L17 L20)
13: (MOVER P1 L19 L4)
14: (MOVER P1 L4 L2)
15: (SOLTAR1 P1 PROD3 L6 L10)
16: (MOVER P1 L2 L4)
17: (PEGAR1 P1 PROD2 L8 L12)
18: (MOVER P1 L4 L19)
19: (SOLTAR1 P1 PROD2 L19 L20)
20: (MOVER P1 L19 L4)
21: (PEGAR1 P1 PROD1 L4 L8)
22: (MOVER P1 L4 L19)
23: (SOLTAR1 P1 PROD1 L17 L20)
24: (PEGAR1 P1 PROD2 L19 L20)
25: (MOVER P1 L19 L4)
26: (SOLTAR1 P1 PROD2 L4 L8)
27: (MOVER P1 L4 L19)
28: (PEGAR1 P1 PROD1 L17 L20)
29: (SOLTAR1 P1 PROD1 L19 L20)
30: (MOVER P1 L19 L2)
31: (PEGAR1 P1 PROD3 L6 L10)
32: (MOVER P1 L2 L4)
33: (SOLTAR1 P1 PROD3 L8 L12)
34: (MOVER P1 L4 L19)
35: (MOVER P1 L19 L2)
36: (PEGAR1 P1 PROD4 L2 L6)
37: (MOVER P1 L2 L19)
38: (SOLTAR1 P1 PROD4 L17 L20)
39: (PEGAR1 P1 PROD1 L19 L20)
40: (MOVER P1 L19 L2)
41: (SOLTAR1 P1 PROD1 L2 L6)
42: (MOVER P1 L2 L4)

43: (PEGAR1 P1 PROD3 L8 L12)
44: (MOVER P1 L4 L19)
45: (MOVER P1 L19 L2)
46: (SOLTAR1 P1 PROD3 L6 L10)
47: (MOVER P1 L2 L19)
48: (PEGAR1 P1 PROD4 L17 L20)
49: (MOVER P1 L19 L4)
50: (SOLTAR1 P1 PROD4 L8 L12)

Tabela 4.23 - Resultado do Mips-XXL para o problema 3.

Mips-XXL
0: (PEGAR3 P1 PROD2 L17 L18 L19)
1: (MOVER P1 L17 L3)
2: (SOLTAR3 P1 PROD2 L3 L7 L11)
3: (MOVER P1 L3 L17)
4: (PEGAR2 P1 PROD1 L13 L14 L18 L16)
5: (MOVER P1 L17 L1)
6: (SOLTAR2 P1 PROD1 L1 L2 L5 L6)

Os resultados apresentados anteriormente mostram os planos obtidos do conjunto de todos os planejadores que foram compilados na etapa anterior. Apenas os compiladores: CBP, Metric FF, SGPlan6, SGPlan5.2.2 e o Mips-XXL; conseguiram resolver o problema, já o restante dos planejadores não eram capazes de avaliar expressões, o que se traduz através do *requirement* do tipo *fluents*. Dos resultados obtidos observa-se que apenas os problemas 1 e 3 obtiveram resultados. Para a definição de um plano ótimo neste domínio de planejamento, tem-se que a ponte deve sempre descrever as seguintes ações: mover; pegar um produto; mover; soltar o produto. Com esta tipologia de resposta pode-se afirmar que para o problema 1 apenas os planejadores CBP e SGPlan 5.2.2 apresentaram uma solução ótima, e para o problema 3 os planejadores CBP, Metric FF, SGPlan 5.2.2 e Mips-XXL apresentaram uma solução ótima.

Para os problemas 2 e 4 foi realizado um monitoramento do sistema enquanto da tentativa de resolução destes problemas. Com relação ao planejador CBP ambos os problemas 2 e 4 resultaram em um estouro de memória RAM do computador, o que implica que a investigação do plano envolve a manipulação de uma quantidade de

informações na arvore de busca que é superior a capacidade do computador utilizado. Para o caso do Metric FF ocorreu a mesma situação que a do planejador anterior, havendo então um estouro de memória devido á explosão de informação. Com relação ao planejador SGPlan 6, com o monitoramento do sistema, é possível afirmar que o mesmo possui internamente uma lógica para o gerenciamento do processo de alocação de memória sendo que durante o processo de buscas várias vezes ocorreu a liberação de memória, apesar disso constatou-se que o planejador entrou em um processo de execução cíclico, jamais fornecendo quais quer resultados. Para o problema 4, o SGPlan 6 apresentou um estouro da memória do computador logo na fase de inicio da busca. Em se tratando do Planejador SGPlan 5.2.2 verificou-se que para o problema 2 o processo de busca foi semelhante ao SGPlan 6 com relação ao gerenciamento de memória e a busca cíclica, porém em um dado momento da busca o planejador simplesmente desistiu, dessa forma encerrando o processo. Com relação ao planejador Mips-XXL houve o estouro de memória RAM para os dois problemas.

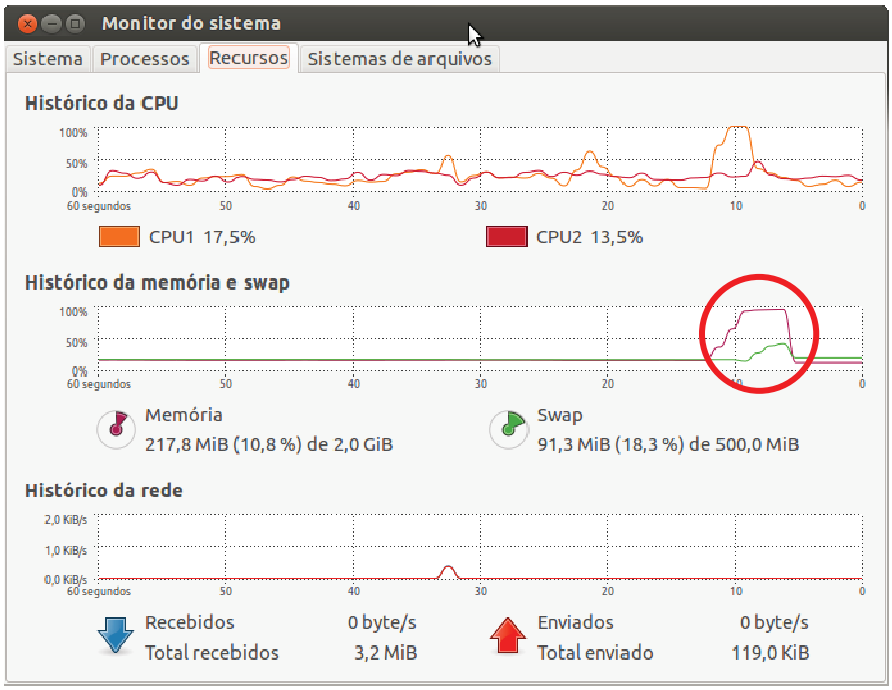


Figura 4.17 – Execução do CBP para o problema 4.

Na tentativa de se contornar as dificuldades encontradas em se executando no computador mencionado anteriormente, buscou-se então um novo computador cujas especificações são as seguintes: Core i7 de 3,4 GHz e 16 Gb de memória RAM. Para este computador, devido a máquina virtual se tratar de um sistema de 32 bits o

endereçamento de memória de memória RAM em condições normais trata-se apenas de 4Gb, porém devido a funcionalidade chamada de PAE (*Physical Adress Extention*- Extensão de Endereço Físico), é possível endereçar 8Gb. Dessa forma, para a máquina virtual foram destinados 8Gb de RAM e 4 *cores* do processador (os testes preliminares revelaram através do monitoramento do sistema que os todos planejadores utilizados trabalham em regime *single core*, ou seja, utilizando apenas um núcleo, portanto a quantidade de núcleos ou *cores* não exerce tanta influência na velocidade de resposta).

Feitos os testes com o novo computador ocorreu que, devido ao sistema operacional, os planejadores novamente resultaram em erro quando os processos envolvidos na execução do planejador atingiam tamanhos superiores à capacidade de manipulação do sistema operacional de 32 bits. Normalmente nos *kernel* padrão 32-bit x86, os processos podem atingir no máximo 3Gb sendo um 1Gb utilizado pelo *kernel*.

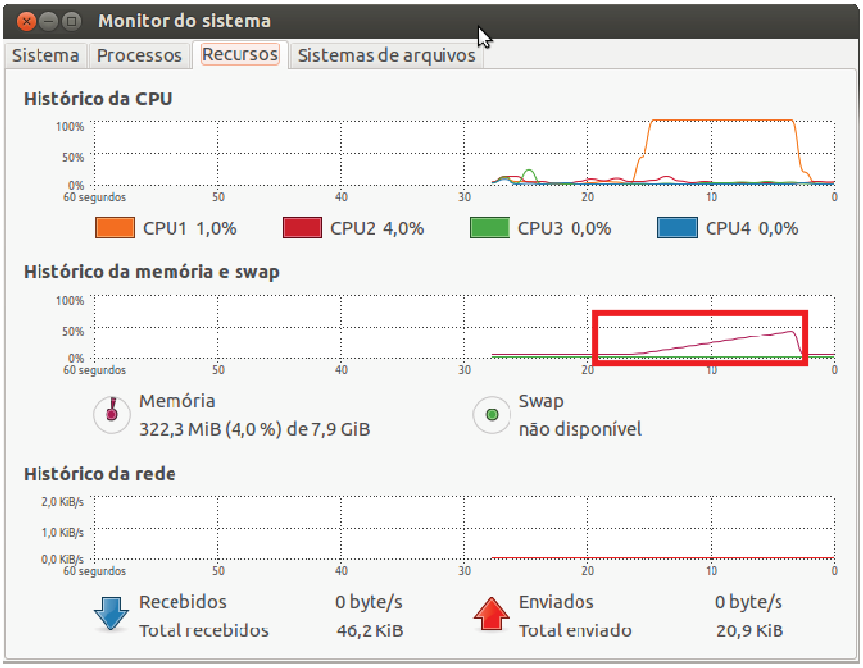


Figura 4.18 – Monitoramento do sistema para CBP com o problema 2.

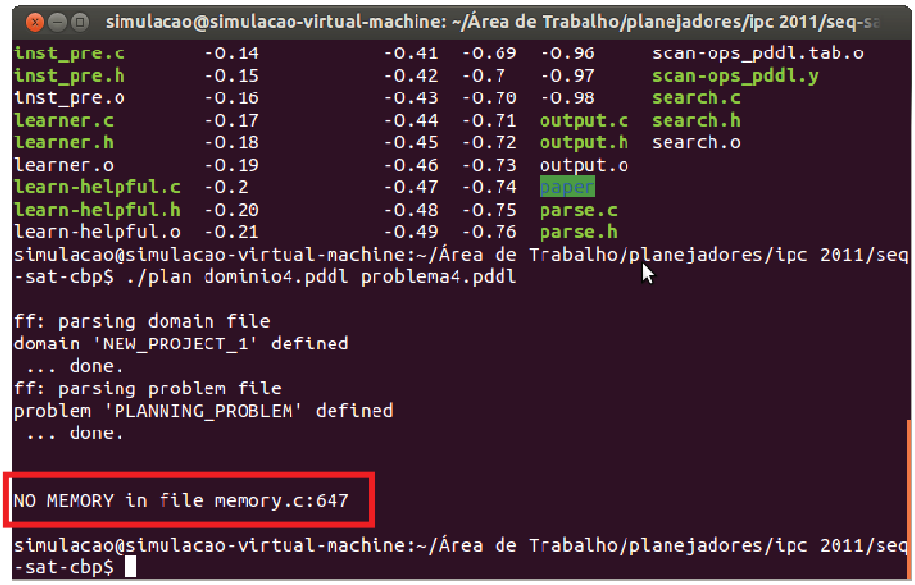


Figura 4.19 – Erro por estouro de tamanho do processo.

Para uma nova abordagem foi criada um nova máquina virtual, sendo esta com o sistema operacional do Ubuntu 12.04 LTS 64bits. Para este caso os planejadores tiveram de ser recompilados para adaptação ao sistema de 64 bits. Vale ressaltar que nesta etapa o planejador SGPlan6 não pôde ser utilizado pelo fato de que o mesmo já possui arquivos precompilados em um sistema de 32 bits (Fig.4.20)

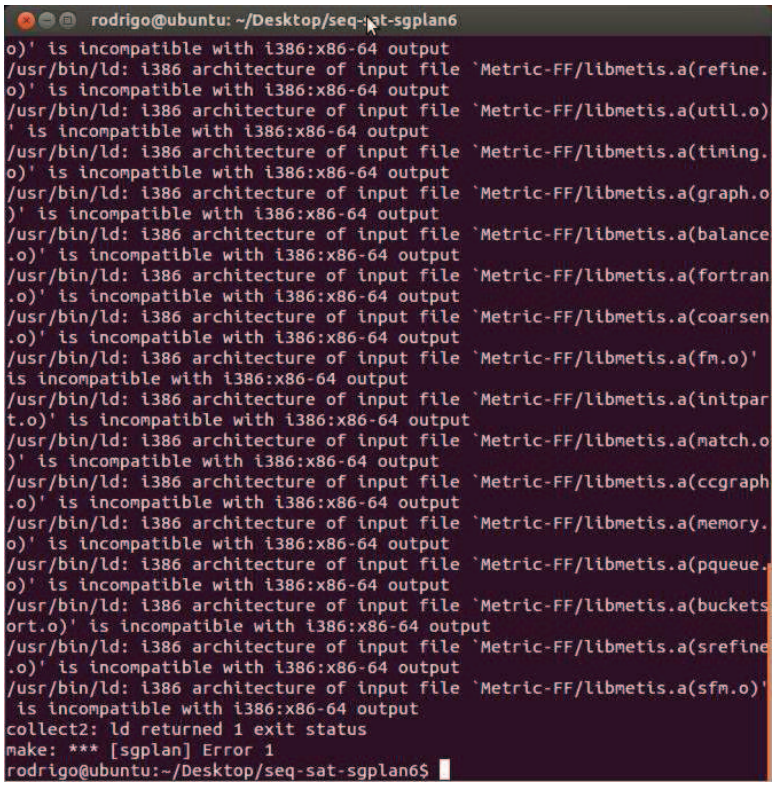


Figura 4.20 – Problemas na compilação do planejador SGPlan 6.

Para os testes realizados com o problema 2, o comportamento foi semelhante ao observados em todos os testes realizados, a memória lentamente foi sendo preenchida

até o seu estouro. A figura abaixo mostra o teste realizado com o planejador CBP cujo estouro de memória pôde ser observado com maior rapidez.

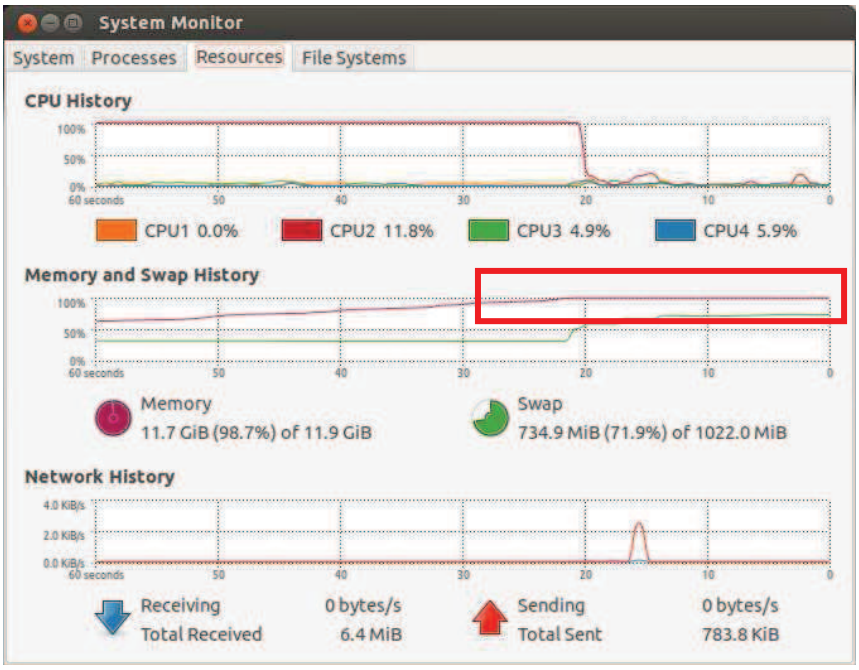


Figura 4.21 – Estouro de memória para o problema 2.

Para o problema 4, observou-se a mesma rampa de subida na memória RAM conforme ocorrido em todos os outros testes, o que sugere que a capacidade encontrada mesmo em um computador de 12 Gb de memória é insuficiente para abertura da árvore de busca neste problema.

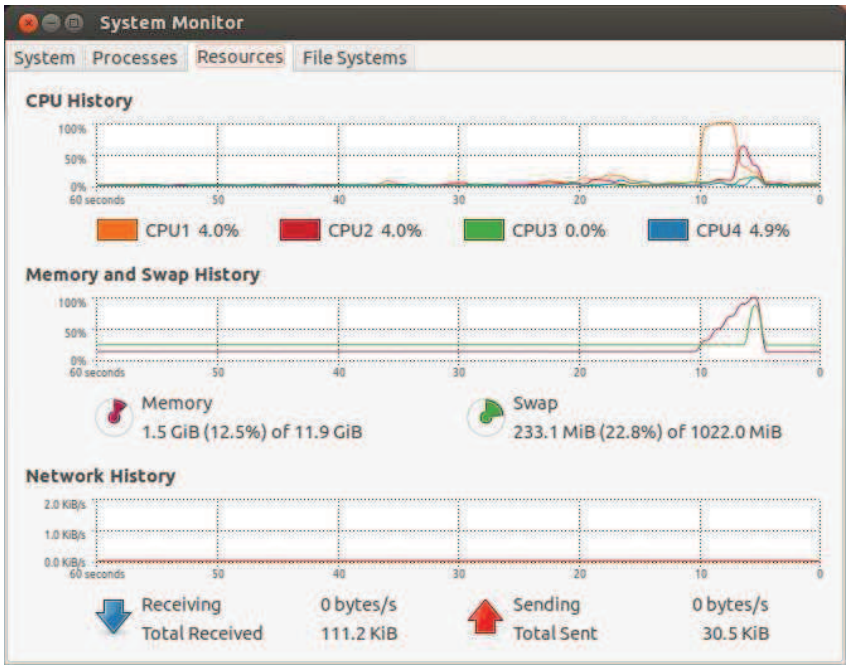


Figura 4.22 – Monitoramento do sistema para o planejador Mips-XXL com o problema 4.

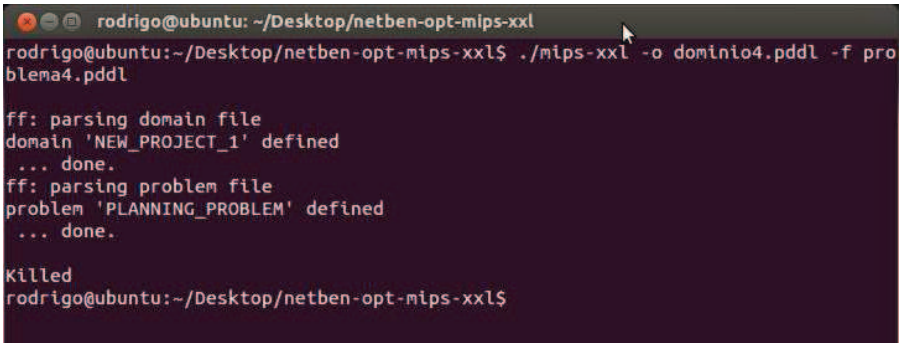


Figura 4.23– Mensagem da linha de comando para o planejador Mips-XXL com o problema 4.

Na sequência foram realizados os testes referentes à modelagem simplificada. Vale ressaltar que esta modelagem foi proposta justamente devido às características da resolução do problema para a modelagem completa, que necessita invariavelmente de uma quantidade de informação maior para representação dos estados e por consequência para a abertura da arvore de busca.

Para esta modelagem ao contrário do observado para a situação anterior uma grande quantidade de planejadores conseguiram fornecer uma solução para o processo de carregamento. Nesta etapa os testes foram realizados em uma máquina virtual do Ubuntu 12.04 LTS 64 bits. O computador trata-se de um Core i5 de 2.4 GHz e 4Gb de RAM, sendo que para a máquina virtual foram disponibilizados 2 *cores* ou núcleos dos 4 disponíveis do processador e 2 Gb de RAM.

Considerando os problemas apresentados anteriormente um total de 25 planejadores, sendo alguns deles diferentes versões de um mesmo planejador, foram capazes de resolver o processo de carregamento. Estes planejadores são listados abaixo:

- *ACOplan*
- *ACOplan 2*
- *BJOLP*
- *CBP*
- *CBP2*
- *FD-Autotune*
- *FD-Autotune 1*
- *FD-Autotune 2*
- *FDSS1*
- *FDSS2*

- *Lama 2008*
- *Lama 2011*
- *Lamar*
- *LM-Cut*
- *Madagascar*
- *Madagascar-p*
- *Merge-and-Shrink*
- *Metric FF*
- *Mips XXL*
- *Randward*
- *Roamer*
- *Selective Max*
- *SGPlan6*
- *SGPlan 5.2.2*
- *Upwards*

A tabela 4 relaciona os resultados obtidos nos problema de 1 a 4. Nesta tabela são mostrados os tesmpo de resolução do problema pelos planejadores, assim como, se os mesmos atingiram o critério de otimalidade.

Tabela 4.24 – Resultado dos problemas 1 a 4 para o modelo simplificado.

Planejador	**Problema 1**		**Problema 2**		**Problema 3**		**Problema 4**	
	Tempo resolução [s]	Sol. ótima	Tempo resolução [s]	Sol. ótima	Tempo resolução [s]	Sol. ótima	Tempo resolução [s]	Sol. ótima
ACOplan	< 0,01	não	< 0,01	não	0,01	sim	7,66	não
ACOplan 2	< 0,01	não	< 0,01	não	0,01	sim	7,88	não
BJOLP	< 0,01	sim	< 0,01	sim	< 0,01	sim	0,02	sim
CBP	< 0,01	sim	< 0,01	sim	< 0,01	sim	< 0,01	sim
CBP2	0,01	sim	< 0,01	sim	< 0,01	sim	< 0,01	sim
FD-Autotune	0,01	sim	< 0,01	sim	< 0,01	sim	0,03	sim
FD-Autotune 1	0,01	sim	0,05	sim	< 0,01	sim	0,03	sim
FD-Autotune 2	0,01	sim	0,04	sim	< 0,01	sim	9,19	sim
FDSS1	< 0,01	sim	< 0,01	sim	< 0,01	sim	0,04	sim

FDSS2	0,01	sim	< 0,01	sim	< 0,01	sim	0,05	sim
Lama 2008	0,01	sim	0,08	sim	< 0,01	sim	0,01	sim
Lama 2011	0,02	sim	0,09	sim	< 0,01	sim	13,56	sim
Lamar	< 0,01	sim	0,02	sim	< 0,01	sim	0,02	sim
LM-Cut	0,01	sim	< 0,01	sim	< 0,01	sim	0,02	sim
Madagascar	< 0,01	sim	< 0,01	sim	< 0,01	sim	0,01	sim
Madagascar-p	< 0,01	sim	< 0,01	sim	< 0,01	sim	0,01	sim
Merge-and-Shrink	0,09	sim	< 0,01	sim	< 0,01	sim	0,02	sim
Metric FF	< 0,01	sim	< 0,01	sim	< 0,01	sim	< 0,01	sim
Mips XXL	-	não	-	não	-	sim	-	não
Randward	< 0,01	sim	0,01	sim	< 0,01	sim	< 0,01	sim
Roamer	< 0,01	sim	0,07	sim	< 0,01	sim	0,01	sim
Selective Max	0,07	sim	0,04	sim	0,01	sim	0,07	sim
SGPlan6	< 0,01	sim	0,04	sim	< 0,01	sim	0,11	sim
SGPlan 5.2.2	< 0,01	sim	0,01	sim	< 0,01	sim	0,05	sim
Upwards	< 0,01	sim	< 0,01	sim	< 0,01	sim	< 0,01	sim

Pela análise da tabela 4.24 observa-se que os resultados obtidos com o modelo simplificado são bem mais promissores do que aqueles para o modelo completo. Nesta fase de testes não foi observado nenhuma mudança significativa na quantidade de memória RAM utilizada pelo processo, o que vale dizer que este modelo possui uma representação bem mais reduzida do que anterior. Em se tratando da quantidade de informações para a representação dos estados, pode–se concluir que este é um importante parâmetro a ser considerado na resolução de problemas com planejadores automáticos. Nos problemas 1 a 4, para a maioria dos planejadores, os resultados foram obtidos em questão de décimos de segundos, a exceção de alguns casos onde aparentemente ocorre o replanejamento em busca de uma solução mais refinada. Além disso, verifica-se que a tecnologia dos planejadores ainda carece na manipulação de informações mais complexas, traduzidas na forma de variáveis inteiras e reais. Vale ressaltar que estes tipos de variáveis consomem um quantidade memória que pode variar de 16 até 64 vezes maior do que uma variável booleana dependendo do

ambiente utilizado. De forma geral, considera-se que os problemas abordados para o modelo simplificado são relativamente simples, sendo necessário uma análise mais ampla a respeito do potencial deste modelo. Em virtude do tempo disponível para realização dos testes com esta modelagem, não foi possível uma análise mais criteriosa, a se dizer estatística, dos resultados.

Vale ressaltar que para o modelo em análise os resultados consistem apenas de um sequenciamento das caixas para alocação dentro do contêiner. Este estudo, conforme apresentado, trata-se apenas de uma tentativa de modelagem mais adequada à manipulação através de planejadores, mas nada impede a inclusão de novas restrições no processo de carregamento, como por exemplo, a ordem das caixas a serem alocados, restrições de manipulação por conta de parâmetros tecnológicos, dentre outros. Através da fase de planejamento automático, pretende-se a flexibilização do processo que transforma um conjunto de caixas armazenadas em um estoque em um padrão de empacotamento ótimo.

A figura 4.24 mostra um exemplo do resultado obtido com o planejador FD-Autotune 1 para o problema 4.

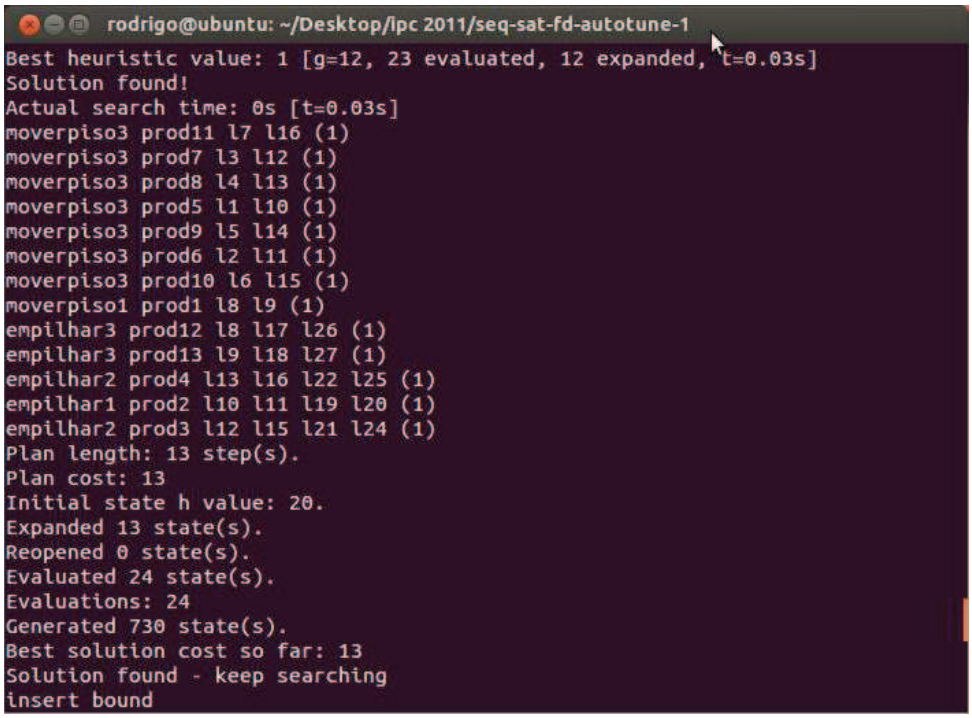


Figura 4.24 – Resultado do problema 4 para o planejador FD-Autotune 1.

Na tabela 4.25 são mostrados alguns dos planos gerados para a situação mais complexa analisada nos testes (problema 4).

Tabela 4.25 – Resultados diversos para o problema 4.

ACOPlan 2	FDSS1	Roamer
(empilhar1 l10 l11 l19 l20 prod2)	(moverpiso1 prod1 l8 l9)	(moverpiso1 prod1 l8 l9)
(empilhar2 l13 l16 l22 l25 prod4)	(moverpiso3 prod11 l7 l16)	(moverpiso3 prod11 l7 l16)
(empilhar3 l8 l17 l26 prod12)	(moverpiso3 prod7 l3 l12)	(moverpiso3 prod6 l2 l11)
(moverpiso3 l2 l11 prod6)	(empilhar3 prod12 l8 l17 l26)	(moverpiso3 prod7 l3 l12)
(moverpiso3 l7 l16 prod11)	(moverpiso3 prod8 l4 l13)	(moverpiso3 prod9 l5 l14)
(empilhar2 l12 l15 l21 l24 prod3)	(empilhar2 prod4 l13 l16 l22 l25)	(empilhar3 prod13 l9 l18 l27)
(moverpiso3 l5 l14 prod9)	(empilhar3 prod13 l9 l18 l27)	(moverpiso3 prod8 l4 l13)
(moverpiso3 l1 l10 prod5)	(moverpiso3 prod5 l1 l10)	(empilhar2 prod4 l13 l16 l22 l25)
(empilhar3 l9 l18 l27 prod13)	(moverpiso3 prod9 l5 l14)	(empilhar3 prod12 l8 l17 l26)
(moverpiso1 l8 l9 prod1)	(moverpiso3 prod6 l2 l11)	(moverpiso3 prod5 l1 l10)
(moverpiso3 l4 l13 prod8)	(empilhar1 prod2 l10 l11 l19 l20)	(empilhar1 prod2 l10 l11 l19 l20)
(moverpiso3 l6 l15 prod10)	(moverpiso3 prod10 l6 l15)	(moverpiso3 prod10 l6 l15)
(moverpiso3 l3 l12 prod7)	(empilhar2 prod3 l12 l15 l21 l24)	(empilhar2 prod3 l12 l15 l21 l24)
tempo total: 7,88 segundos	tempo total: 0,04 segundos	tempo total: 0,01 segundos
Lama 2008	Selective Max	LM-Cut
(moverpiso1 prod1 l8 l9)	(moverpiso1 prod1 l8 l9)	(moverpiso1 prod1 l8 l9)
(moverpiso3 prod7 l3 l12)	(moverpiso3 prod11 l7 l16)	(moverpiso3 prod11 l7 l16)
(moverpiso3 prod8 l4 l13)	(moverpiso3 prod7 l3 l12)	(moverpiso3 prod7 l3 l12)
(moverpiso3 prod11 l7 l16)	(empilhar3 prod12 l8 l17 l26)	(empilhar3 prod12 l8 l17 l26)
(empilhar2 prod4 l13 l16 l22 l25)	(moverpiso3 prod8 l4 l13)	(moverpiso3 prod8 l4 l13)
(empilhar3 prod12 l8 l17 l26)	(empilhar2 prod4 l13 l16 l22 l25)	(empilhar2 prod4 l13 l16 l22 l25)
(empilhar3 prod13 l9 l18 l27)	(empilhar3 prod13 l9 l18 l27)	(empilhar3 prod13 l9 l18 l27)
(moverpiso3 prod10 l6 l15)	(moverpiso3 prod5 l1 l10)	(moverpiso3 prod5 l1 l10)
(moverpiso3 prod5 l1 l10)	(moverpiso3 prod9 l5 l14)	(moverpiso3 prod9 l5 l14)
(moverpiso3 prod9 l5 l14)	(moverpiso3 prod6 l2 l11)	(moverpiso3 prod6 l2 l11)
(moverpiso3 prod6 l2 l11)	(empilhar1 prod2 l10 l11 l19 l20)	(empilhar1 prod2 l10 l11 l19 l20)
(empilhar1 prod2 l10 l11 l19 l20)	(moverpiso3 prod10 l6 l15)	(moverpiso3 prod10 l6 l15)
(empilhar2 prod3 l12 l15 l21 l24)	(empilhar2 prod3 l12 l15 l21 l24)	(empilhar2 prod3 l12 l15 l21 l24)
tempo total: 0,01 segundos	tempo total: 0,07 segundos	tempo total: 0,02 segundos

CAPÍTULO V

CONCLUSÕES

Este trabalho, não se trata mais do que uma tentativa de se empregar a tecnologia de planejamento automático em uma situação real para um processo que ainda pode ser melhorado. Em virtude dos resultados encontrados, pode-se afirmar que o emprego desta tecnologia é possível, e futuramente com os avanços nos algoritmos e na capacidade de processamento e armazenamento de dados, será de grande valia, pois o estudo nesta área do conhecimento proporciona cada vez mais um melhor entendimento nas características da inteligência e do processo de tomada de decisões. É importante citar que na situação atual os mesmos já possuem sua parte de aplicações, sendo esta ainda restrita em se tratatando de planejadores de propósitos gerais. Tendo em vista que o planejamento automático vem sendo estudado há aproximadamente 50 anos, o progresso, na tentativa de se desenvolver novas heurísticas para o tratamento e otimização de problemas, pode ser evidenciado pela existência de diversos tipos de planejadores, até mesmo aqueles utilizados neste trabalho, cada qual com sua própria ideia de tomada de decisão. É fato que os mesmos ainda têm um longo caminho pela frente, até que sejam capazes de contornar as dificuldades na manipulação de variáveis inteiras e reais, e com isso este trabalho pode ser considerado um apelo àqueles que se dedicam a esta área do conhecimento, para um melhor gerenciamento da memória utilizado pelo processo e também um melhor aproveitamento do processador através de aplicações que implementam *multi-thread*. Os planejadores devido ao seu próprio ambiente de desenvolvimento e às características das competições, carecem de certa forma de atributos que os tornam aplicáveis em situações reais, razão pela qual foi elaborado o modelo simplficado que se utiliza apenas de variáveis booleans. Porém nem sempre está simplificação pode ser possível, o que limita a aplicação dos planejadores. Por sua vez este trabalho vem ressaltar a importância da união das áreas da ciência da computação e engenharia na tentativa do desenvolvimento de sistemas cada vez mais inteligentes e autônomos.

Neste trabalho em particular, considera-se como resultado principal, a contribuição na tentativa do desenvolvimento de um sistema de inteligência artificial, a ser utilizado em equipamento destinado ao processo de carregamento de contêineres. Vale ressaltar que o trabalho desenvolvido ainda possui restrições relacionadas ao tamanho do contêiner, as dimensões das caixas, a natureza das variáveis utilizadas no modelo matemático (puramente inteiras) e na capacidade de resolução dos problemas através de planejadores automáticos.

Em geral, considera-se que o passo inicial na tentativa de desenvolvimento de um sistema automático para o processo de carregamento de contêineres foi dado. Além disso, as possibilidades de trabalhos futuros, até mesmo em cima da proposta do projeto apresentada, são inúmeras, sendo que o desenvolvimento de um sistema físico para validação do algoritmo, o refino dos modelos em programação linear e planejamento automático e a realização de testes com problemas mais complexos para a modelagem simplificada são essenciais para a continuação deste trabalho. Vale ressaltar que as possibilidades para o modelo em programação linear são muito amplas, uma vez que a ferramenta para resolução de problemas mais complexos (função *simplexDuasFases*) já está consolidada e permite a adição de novas restrições, além disso, é importante relembrar que a revisão das técnicas de resolução é de fundamental importância, pois permite a simplificação de procedimentos para a uma melhor adequação. Neste caso em particular, a obtenção do *simplex1FaseEMeia* é prova deste fato. Em adição aos trabalhos futuros, pode-se adicionar a revisão de novas técnicas para resolução de problemas de P.L. Ainda assim a experiência obtida com a programação linear e a linguagem PDDL permitiu um entendimento mais amplo a ser utilizado no aprimoramento e criação de novas ferramentas computacionais.

REFERÊNCIAS BIBLIOGRÁFICAS

ALCÁZAR, V.; VELOSO, M. BRT: Biased Rapidly-exploring Tree. In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 7, 2011, Madrid. **Proceedings…**4p. Disponível em: <svn://svn@pleiades.plg.inf.uc3m.es/ipc2011/data/planners/seq-sat/seq-sat-brt/paper>Acesso em 3 Nov. 2013.

BAIOLETTI, M.; MILANI, A.; POGGIONI, V.; ROSSI, F. The ACOPlan Planner.In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 7, 2011, Madrid. **Proceedings…**4p. Disponível em: <svn://svn@pleiades.plg.inf.uc3m.es/ipc2011/data/planners/seq-sat/seq-sat-acoplan/paper> Acesso em 3 Nov. 2013.

BALLOU, R. H. **Gerenciamento da Cadeia de Suprimentos/Logística Empresarial**. 5 ed. Porto Alegre: Bookman, 2006. 616p.

BAZARRA, M. S.; JARVIS, J. J.; SHERALI, H. D. **Linear Programming and Network Flows.** 4 ed. Nova Jersey: John Wiley & Sons. 2010.

BEASLEAY, J. E. An exact two-dimensional non-guillotine cutting tree search procedure. **Operations Research**, v. 33, n. 1, p. 49-64, 1985.

BIBAÏ, J.; SAVÉANT, P.; SCHOENAUER, M.; VIDAL, V. DAE: Planning as Artificial Evolution. In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 6, 2008, Sydney. **Proceedings…**3p. Disponível em: <http://ipc.informatik.uni-freiburg.de/Planners?action=AttachFile&do=view&target=DAE.pdf> Acesso em 3 Nov. 2013.

BISCHOFF, E. E.; JANETZ, F.; RATCLIFF, M. S. W. Loading pallets with non-identical items. **European Journal of Operational Research.** v. 84, n. 1995, p. 681-692. 1995.

BORTFELDT, A.; GEHRING, H; MACK, D. A parallel tabu search algorithm for solving the container loading problem. **Parallel Computing.** v. 29, n. 2003, p. 641-662. 2003.

BORTFELDT, A.; MACK, D. A heuristic for the three-dimensional strip packing problem. **European Journal of Operational Research.** v. 183, n. 2007, p. 1267–1279. Jun. 2006.

BORTFELDT, A.; WÄSCHER, G. Constraints in container loading – A state-of-the-art review. **European Journal of Operational Research.** v. 229, n. 2013, p. 1-20. Dec. 2012.

BOWERSOX, D. J.; CLOSS, D. J. **Logística Empresarial:** O processo de integração da cadeia de suprimentos. 1 ed. São Paulo: Atlas, 2007. 594p.

CAI, D. SatPlanLM and SatPlanLM-c: Using Landmarks and Their Orderings as Constraints. In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 7, 2011, Madrid. **Proceedings…**2p. Disponível em: <svn://svn@pleiades.plg.inf.uc3m.es/ipc2011/data/planners/seq-sat/seq-sat-satplanlm-c/paper> Acesso em 3 Nov. 2013.

CARVALHO, J. M. V. LP models for bin packing and cutting stock problems. **European Journal of Operational Research.** v. 141, n. 2002, p. 253-273. 2002.

CHE, C. H.; HUANG, W.; LIM, A.; ZHU, W. The multiple container loading cost minimization problem. **European Journal of Operational Research.** v. 214, n. 2011, p. 501-511.

CHEN, Y.; LV, Q.; HUANG, R. Plan-A: A Cost Optimal Planner Based on SAT-Constrained Optimization. In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 6, 2008, Sydney. **Proceedings…**4p. Disponível em: <http://ipc.informatik.uni-freiburg.de/Planners?action=AttachFile&do=view&target=Plan-A.pdf> Acesso em 3 Nov. 2013.

CHIEN, C. F.; LEE, C. Y.; HUANG, Y. C.; WU, W. T. An efficient computational procedure for determining the container-loading pattern. **Computers & Industrial Engineering.** v. 56, n. 2009, p. 965-978. Set. 2008.

COLES, A.; COLES, A.; FOX, M.; LONG, D. LPRPG: A Planner for Metric Resources.In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 7, 2011a, Madrid. **Proceedings…**3p. Disponível em: <svn://svn@pleiades.plg.inf.uc3m.es/ipc2011/data/planners/seq-sat/seq-sat-lprpgp/paper/seq-sat-lprpgp.pdf> Acesso em 3 Nov. 2013.

COLES, A.; COLES, A.; FOX, M.; LONG, D. POPF2: a Forward-Chaining Partial Order Planner.In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 7, 2011b, Madrid. **Proceedings…**6p. Disponível em: <svn://svn@pleiades.plg.inf.uc3m.es/ipc2011/data/planners/seq-sat/seq-sat-popf2/paper> Acesso em 3 Nov. 2013.

COLES, A.; SMITH, A. Upwards: The Role of Analysis in Cost-Optimal SAS+ Planning. In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 6, 2008, Sydney. **Proceedings…**4p. Disponível em: <http://ipc.informatik.uni-freiburg.de/Planners?action=AttachFile&do=view&target=upwards.pdf> Acesso em 3 Nov. 2013.

DEPARTMENT OF ELECTRONICS FOR AUTOMATION. **Plan Constraints and Preferences in PDDL3.0.** Universidade da Brescia, Italia, Ago. 2005. 12p. Relatório.

DERELI, T.; DAS, G. S. A hybrid ‘bee(s) algorithm’ for solving container loading problems. **Applied Soft Computing.** v. 11, n. 2011, p. 2854–2862. Dez. 2010.

DOMSHLAK, C.; HELMERT, M.; KARPAS, E.; KEYDER, E.; RICHTER, S.; RÖGER, G.; SEIPP, J.; WESTPHAL, M. BJOLP: The Big Joint Optimal Landmarks Planner. In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 7, 2011, Madrid. **Proceedings…**5p. Disponível em: <svn://svn@pleiades.plg.inf.uc3m.es/ipc2011/data/planners/seq-opt/seq-opt-bjolp/paper> Acesso em 3 Nov. 2013.

DOMSHLAK, C.; HELMERT, M.; KARPAS, E.; MARKOVITCH, S. The SelMax Planner: Online Learning for Speeding up Optimal Planning. In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 7, 2011, Madrid. **Proceedings…**5p. Disponível em:

<svn://svn@pleiades.plg.inf.uc3m.es/ipc2011/data/planners/seq-opt/seq-opt-selmax/paper> Acesso em 3 Nov. 2013.

DRÉO, J.; SAVÉANT, P.; SCHOENAUER, M.; VIDAL, V. Divide-and-Evolve: the Marriage of Descartes and Darwin.In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 7, 2011, Madrid. **Proceedings…**4p. Disponível em: <svn://svn@pleiades.plg.inf.uc3m.es/ipc2011/data/planners/seq-sat/seq-sat-dae_yahsp/paper> Acesso em 3 Nov. 2013.

DYCKHOFF, H. A typology of cutting and packing problems. **European Journal of Operational Research.** North Holland, v. 44, n. 1990, p. 145-159. 1990

EDELKAMP, S.; HOFFMAN, J. PDDL 2.2: The language for the classical part of the 4th International Planning Competition. 2004. Relatorio técnico.

EDELKAMP, S.; JABBAR, S. MIPS-XXL: Featuring External Shortest Path Search for Sequential Optimal Plans and External Branch-And-Bound for Optimal Net Benefit. In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 6, 2008, Sydney. **Proceedings…**4p. Disponível em: <http://ipc.informatik.uni-freiburg.de/Planners?action=AttachFile&do=view&target=mips_xxl.pdf> Acesso em 3 Nov. 2013.

EDELKAMP, S.; KISSMANN, P. GAMER: Bridging Planning and General Game Playing with Symbolic Search. In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 6, 2008, Sydney. **Proceedings…**4p. Disponível em: <http://ipc.informatik.uni-freiburg.de/Planners?action=AttachFile&do=view&target=gamer.pdf> Acesso em 3 Nov. 2013.

EGEBLAD, J.; PISINGER, D. Heuristic approaches for the two- and three-dimensional knapsack packing problem. **Computers & Operations Research.** v. 36, n. 2009, p. 1026-1049. Dez. 2007.

ELEY, M. Solving container loading problems by block arrangement. **European Journal of Operational Research.** v. 141, n. 2002, p. 393–409. Abr. 2001.

FAWCETT, C.; HELMERT, M.; HOOS, H.; KARPAS, E.; RÖGER, G.; SEIPP, J. FD-Autotune: Automated Configuration of Fast Downward. In: INTERNATIONAL

PLANNING COMPETITION BOOKLET, 7, 2011, Madrid. **Proceedings…**7p. Disponível em: <svn://svn@pleiades.plg.inf.uc3m.es/ipc2011/data/planners/seq-opt/seq-opt-fd-autotune/paper> Acesso em 3 Nov. 2013.

FIKES, R. E.; NILSSON, N. J. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. **Artificial Intelligence.** v. 2, n. 1971, p. 189-208. Set. 1971.

FOX, M.; LONG, D. PDDL 2.1: An Extension to pddl for Expressing Temporal Planning Domains. **Journal of Artificial Intelligence Research.** v. 20, n. 2003, p. 61-124. 2003.

FUENTETAJA, R. The CBP planner. In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 7, 2011, Madrid. **Proceedings…**4p. Disponível em: <svn://svn@pleiades.plg.inf.uc3m.es/ipc2011/data/planners/seq-sat/seq-sat-cbp/paper>Acesso em 3 Nov. 2013.

GEHRING. H.; BORTFELDT, A. A genetic algorithm for solving the container loading problem. **International Transactions in Operational Research.** v. 4, n. 5/6, p. 401-418. 1997.

GRANDCOLAS, S.; BARRE, C. P. CFDP: an approach to Cost-Optimal Planning based on FDP. In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 6, 2008, Sydney. **Proceedings…**2p. Disponível em: <http://ipc.informatik.uni-freiburg.de/Planners?action=AttachFile&do=view&target=C-fdp.pdf> Acesso em 3 Nov. 2013.

GROOVER, M. P. **Automação Industrial e Sistemas de Manufatura.** 3 ed. São Paulo: Pearson Prentice Hall, 2011. 581p.

HALEVI, G.; WEILL, R. D. **Principles of Process Planning.** 1 ed. Gloucester: Chapman & Hall, 1995. 352p.

HASLUM, P. Additive and Reversed Relaxed Reachability Heuristics Revisited. In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 6, 2008, Sydney. **Proceedings…**4p. Disponível em: <http://ipc.informatik.uni-freiburg.de/Planners?action=AttachFile&do=view&target=hsps.pdf> Acesso em 3 Nov. 2013.

HELMERT, M.; RÖGER, G.; SEIPP, J.; KARPAS, E.; HOFFMANN, J.; KEYDER, E.; NISSIM, R.; RICHTER, S.; WESTPHAL, M. Fast Downward Stone Soup. In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 7, 2011, Madrid. **Proceedings…**8p. Disponível em: <svn://svn@pleiades.plg.inf.uc3m.es/ipc2011/data/planners/seq-opt/seq-opt-fdss-1/paper> Acesso em 3 Nov. 2013.

HELMERT. M.; DOMSHLAK, C. LM-Cut: Optimal Planning with the Landmark-Cut Heuristic. In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 7, 2011, Madrid. **Proceedings…**3p. Disponível em: <svn://svn@pleiades.plg.inf.uc3m.es/ipc2011/data/planners/seq-opt/seq-opt-lmcut> Acesso em 3 Nov. 2013.

HIFI, M. Approximate algorithms for the container loading problem. **International Transactions in Operational Research.** v. 9, n. 2002 p. 747–774. Dez. 2001.

HOFFMANN, J. The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables. **Journal of Artificial Intelligence Research**. El Segundo, v. 20,n.1, p. 291–341, dez. 2003. Disponível em: <http://www.jair.org/papers/paper1144.html> Acesso em 13 jan. 2012.

HSU, C. W.; WAH, B. W. The SGPlan Planning System in IPC-6. In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 6, 2008, Sydney. **Proceedings…**3p. Disponível em: <http://ipc.informatik.uni-freiburg.de/Planners?action=AttachFile&do=view&target=SGPlan.pdf>Acesso em 3 Nov. 2013.

HSU, C. W.; WAH, B. W.; HUANG, R.; CHEN, Y. X. New features in SGPlan for Handling Soft Constraints and Goal Preferences in PDDL3.0. In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 5, 2006, Cumbria. **Proceedings…**3p. Disponível em: <http://eracle.ing.unibs.it/ipc-5/booklet/deterministic09.pdf> Acesso em 13 jan. 2012.

HUANG, R.; CHEN, Y.; ZHANG, W. DTG-Plan:Fast Planning by Search in Domain Transition Graphs. In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 6, 2008, Sydney. **Proceedings…**4p. Disponível em:

<http://ipc.informatik.uni-freiburg.de/Planners?action=AttachFile&do=view&target=DTG-Plan.pdf> Acesso em 3 Nov. 2013.

JUNQUEIRA, L. **Modelos de programação matemática para problemas de carregamento de caixas dentro de contêineres.** 2009. 134f. Dissertação (Mestrado) - Universidade Federal de SãoCarlos, São Carlos.

JUNQUEIRA, L.; MORABITO, R.; YAMASHITA, D. S. MIP-based approaches for the container loading problem with multi-drop constraints. **Annals of Operations Research.** v. 199, n. 2012, p. 51-75. Ago 2011.

JUNQUEIRA, L.; MORABITO, R.; YAMASHITA, D. S. Three-dimensional container loading models with cargo stability and load bearing constraints. **Computers & Operations Research.** v. 39, n. 2012, p. 74-85.

KATZ, M.; DOMSHLAK, C. Planning with Implicit Abstraction Heuristics. In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 7, 2011, Madrid. **Proceedings…**4p. Disponível em: <svn://svn@pleiades.plg.inf.uc3m.es/ipc2011/data/planners/seq-opt/seq-opt-forkinit/paper> Acesso em 3 Nov. 2013.

KAVULURI, B. R. Extending Temporal Planning for the Interval Class. In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 7, 2011, Madrid. **Proceedings…**4p. Disponível em: <svn://svn@pleiades.plg.inf.uc3m.es/ipc2011/data/planners/seq-sat/seq-sat-sharaabi/paper> Acesso em 3 Nov. 2013.

KEYDER, E.; GEFFNER, H. The FF(ha) Planner for Planning with Action Costs. In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 6, 2008, Sydney. **Proceedings…**3p. Disponível em: <http://ipc.informatik.uni-freiburg.de/Planners?action=AttachFile&do=view&target=ffha.pdf> Acesso em 3 Nov. 2013.

KISSMANN, P.; EDELKAMP, S.; Improving Cost-Optimal Domain-Independent Symbolic Planning. In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 7, 2011, Madrid. **Proceedings…**7p. Disponível em:

<svn://svn@pleiades.plg.inf.uc3m.es/ipc2011/data/planners/seq-opt/seq-opt-gamer/paper> Acesso em 3 Nov. 2013.

LIM, A.; MA, H.; QIU, C.; ZHU, W. The single container loading problem with axle weight constraints. **International Journal of Production Economics.** v. 144, n. 2013, p. 358-369. Mar. 2013.

LIM, A.; MA, H.; XU, J.; ZHANG, X. An iterated construction approach with dynamic prioritization for solving the container loading problems. **Expert Systems with Applications.** v. 39, n. 2012, p. 4292–4305. 2012.

LIPOVETZKY, N.; GEFFNER, H. Searching with Probes: The Classical Planner PROBE.In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 7, 2011, Madrid. **Proceedings…**2p. Disponível em: <svn://svn@pleiades.plg.inf.uc3m.es/ipc2011/data/planners/seq-sat/seq-sat-probe/paper> Acesso em 3 Nov. 2013.

LIPOVETZKY, N.; RAMÍREZ, M.; GEFFNER, H. C3: Planning with Consistent Causal Chains. In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 6, 2008, Sydney. **Proceedings…2**p. Disponível em: <http://ipc.informatik.uni-freiburg.de/Planners?action=AttachFile&do=view&target=C3.pdf> Acesso em 3 Nov. 2013.

LIU, J.; YUE, Y.; DONG, Z.; MAPLE, C.; KEECH, M. A novel hybrid tabu search approach to container loading. **Computers &Operations Research.** v. 38, n. 2011, p. 797–807. Set. 2010.

LODI, A.; MARTELLO, S.; VIGO, D. Heuristic algorithms for the three-dimensional bin packing problem. **European Journal of Operational Research.** v. 141, n. 2002, p. 410-420. 2002.

LU, Q.; XU, Y.; HUANG, R.; CHEN, Y. The Roamer Planner Random-Walk Assisted Best-First Search. In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 7, 2011, Madrid. **Proceedings…**3p. Disponível em: <svn://svn@pleiades.plg.inf.uc3m.es/ipc2011/data/planners/seq-sat/seq-sat-roamer/paper> Acesso em 3 Nov. 2013.

LUIZ, G. C. **Unitização de cargas com ênfase em pallets e containers.** 2007. 80f. Projeto de Pesquisa. Universidade da Região da Campanha, São Borja.

MACK, D.; BORTFELDT, A.; GEHRING, H. A parallel hybrid local search algorithm for the container loading problem. **International Transactions in Operational Research.** v. 11, n. 2004, p. 511-513. Mar. 2004.

MARIS, F.; RÉGNIER, P. TLP-GP: a Planner to Solve Temporally-Expressive Problems. In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 6, 2008, Sydney. **Proceedings…**4p. Disponível em: <http://ipc.informatik.uni-freiburg.de/Planners?action=AttachFile&do=view&target=TLP-GP.pdf> Acesso em 3 Nov. 2013.

MATHUR, K. An integer-programming-based heuristic for the balanced loading problem. **Operations Research Letters.** v. 22, n. 1998, p. 19-25. 1998.

NAKHOST, H.; MÜLLER, M.; VALENZANO, R.; XIE, F. Arvand: the Art of Random Walks.In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 7, 2011, Madrid. **Proceedings…**2p. Disponível em: <svn://svn@pleiades.plg.inf.uc3m.es/ipc2011/data/planners/seq-sat/seq-sat-arvand/paper> Acesso em 3 Nov. 2013.

NETO, L. L. R. **Um algoritmo genético para solução do problema de carregamento de container.**2005. 105f. Dissertação (Mestrado) – UniversidadeFederal do Rio de Janeiro, Rio de Janeiro.

NISSIM, R.; HOFFMANN. J.; HELMERT, M. The Merge-and-Shrink Planner: Bisimulation-based Abstraction for Optimal Planning. In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 7, 2011, Madrid. **Proceedings…**2p. Disponível em: <svn://svn@pleiades.plg.inf.uc3m.es/ipc2011/data/planners/seq-opt/seq-opt-merge-and-shrink/paper> Acesso em 3 Nov. 2013.

OLSEN, A.; BRYCE, D. Randward and Lamar: Randomizing the FF Heuristic.In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 7, 2011, Madrid. **Proceedings…**3p. Disponível em: <svn://svn@pleiades.plg.inf.uc3m.es/ipc2011/data/planners/seq-sat/seq-sat-randward/paper> Acesso em 3 Nov. 2013.

POLI, G. I.; PUREZA, V.Um algoritmo de busca tabu para o carregamentode contêineres com caixas idênticas. **Gestão & Produção.** v. 19, n. 2, p. 323-336. 2012.

REN, J.; TIAN, Y.; SAWARAGI, T. A tree search method for the container loading problem with shipment priority. **European Journal of Operational Research.** v. 214, n. 2011, p. 526-535. Abr. 2011.

RICHTER, S.; WESTPHAL, M. The LAMA Planner Using Landmark Counting in Heuristic Search. In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 6, 2008, Sydney. **Proceedings…**4p. Disponível em: <http://ipc.informatik.uni-freiburg.de/Planners?action=AttachFile&do=view&target=LAMA.pdf> Acesso em 3 Nov. 2013.

RICHTER, S.; WESTPHAL, M.; HELMERT, M. LAMA 2008 and 2011.In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 7, 2011, Madrid. **Proceedings…**5p. Disponível em: <svn://svn@pleiades.plg.inf.uc3m.es/ipc2011/data/planners/seq-sat/seq-sat-lama-2008/paper>Acesso em 3 Nov. 2013.

RINTANEN, J. Madagascar: Efficient Planning with SAT.In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 7, 2011, Madrid. **Proceedings…**4p. Disponível em: <svn://svn@pleiades.plg.inf.uc3m.es/ipc2011/data/planners/seq-sat/seq-sat-madagascar/paper>Acesso em 3 Nov. 2013.

ROBINSON, N.; GRETTON, C.; PHAM, D. N. CO-PLAN: Combining SAT-Based Planning with Forward-Search. In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 6, 2008, Sydney. **Proceedings…**2p. Disponível em: <http://ipc.informatik.uni-freiburg.de/Planners?action=AttachFile&do=view&target=co-plan.pdf> Acesso em 3 Nov. 2013.

RÖGER, G.; EYERICH, P.; MATTMÜLLER, R. TFD: A Numeric Temporal Extension to Fast Downward. In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 6, 2008, Sydney. **Proceedings…**2p. Disponível em: <http://ipc.informatik.uni-

freiburg.de/Planners?action=AttachFile&do=view&target=TFD.pdf> Acesso em 3 Nov. 2013.

SANTOS, M. P. **Programação Linear.** Rio de Janeiro: Universidade do Estado do Rio de Janeiro.146p. Apostila.

SCHEITHAUER, G. LP-based bounds for the container and multi-container loading problem. **International Transactions in Operational Research.** v. 6, n. 1999, p. 199-213. 1999.

SOAK, S. M.; LEE, S. W. A memetic algorithm for the quadratic multiple container packing problem. **Applied Intelligence.** v. 36, n. 2012 p. 119–135. Ago. 2010.

TEMPONI, E. C. C. **Uma proposta de resolução do problema de corte bidimensional via abordagem metaheurística.** 2007. 100f. Dissertação (Mestrado) - Centro Federal de EducaçãoTecnológica de Minas Gerais, Belo Horizonte.

VIDAL, V. YAHSP2: Keep It Simple, Stupid. In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 7, 2011b, Madrid. **Proceedings…**4p. Disponível em: <svn://svn@pleiades.plg.inf.uc3m.es/ipc2011/data/planners/seq-sat/seq-sat-yahsp2/paper> Acesso em 3 Nov. 2013.

VIDAL, V.; CPT4: An Optimal Temporal Planner Lost in a Planning Competition without Optimal Temporal Track. In: INTERNATIONAL PLANNING COMPETITION BOOKLET, 7, 2011a, Madrid. **Proceedings…**4p. Disponível em: <svn://svn@pleiades.plg.inf.uc3m.es/ipc2011/data/planners/seq-opt/seq-opt-cpt4/paper> Acesso em 3 Nov. 2013.

WANG, Z.; LI, K. W.; LEVY, J. K. A heuristic for the container loading problem: A tertiary-tree-based dynamic space decomposition approach. **European Journal of Operational Research.** v. 191, n. 2008, p. 86-99. Ago. 2007.

WÄSCHER, G.; HAUßNER, H.; SCHUMANN H. An improved typology of cutting and packing problems. **European Journal of Operational Research.** v. 183, n. 2007, p. 1109-1130. Jun. 2006.

YALE CENTER FOR COMPUTATIONAL VISION AND CONTROL. **PDDL** - The Planning Domain Definition Language Version 1.2. 1998. 27p. Relatório.

YAP, C. N.; LEE, L. S.; MAJID, Z. A.; SEOW, H. V. Ant Colony Optimization for Container Loading Problem. **Journal of Mathematics and Statistics.** v. 8, n. 2, p. 169-175. 2012.

ZHANG, D.; PENG, Y.; LEUNG, S. C. H. A heuristic block-loading algorithm based on multi-layer search for the container loading problem. **Computers & Operations Research.** v. 39, n. 2012, p. 2267–2276. Out. 2011.

APÊNDICE A

Código fonte da função *problem_generator*.

```
function [ box container qtipo ] = problem_generator()
% Problem Generator
fprintf('Este programa gera instancias para o problema de
carregamento\nde contêineres 3D\n')

% Definição dos parametros do contêiner
fprintf('\n-----------------------Parametros do contêiner-----------
-----------')
L=input('\nEntre com o comprimento do contêiner: ');
W=input('Entre com a largura do contêiner: ');
H=input('Entre com a altura do contêiner: ');

container.dim=[L;W;H];

% Definição dos parametros das caixas
fprintf('\n-------------------------Parametros das caixas-------------
-----------')
n=input('\n\nEntre com o número de tipos de caixas: '); % tipos de
caixa

% verifica se o número de caixas é valido
while n<=0
    fprintf('número incorreto - n > 0\n')
    n=input('Entre com o número de tipos de caixas: ');
end
% obtém as quantidades e dimensões dos tipos de caixas
for i=1:n

    expressao=sprintf('\nForneça o número de caixas do tipo %d: ',i);
    quant(i)=input(expressao);
    expressao=sprintf('Entre com o comprimento da caixa do tipo %d:
',i);
    dim(1,i)=input(expressao);
    expressao=sprintf('Entre com o largura da caixa do tipo %d: ',i);
    dim(2,i)=input(expressao);
    expressao=sprintf('Entre com o altura da caixa do tipo %d: ',i);
    dim(3,i)=input(expressao);

while (quant(i)<1 || dim(1,i) < 1 || dim(1,i) > L || dim(2,i) < 1 ||
dim(2,i) > W || dim(3,i) < 1 || dim(3,i) > H)
        fprintf('Dados incorretos preencha novamente')
        expressao=sprintf('\nForneça o número de caixas do tipo %d:
',i);
        quant(i)=input(expressao);
        expressao=sprintf('Entre com o comprimento da caixa do tipo
%d: ',i);
        dim(1,i)=input(expressao);
        expressao=sprintf('Entre com o largura da caixa do tipo %d:
',i);
        dim(2,i)=input(expressao);
        expressao=sprintf('Entre com o altura da caixa do tipo %d:
',i);
```

```
dim(3,i)=input(expressao);
end
end
c=1;
for i=1:n

    box(c).quant=quant(i);
    box(c).dim=dim(:,i);
    box(c).vol=dim(1,i)*dim(2,i)*dim(3*i);
    c=c+1;

    %% Verifica se é possivel rotacionar e se com a rotação a caixa
cabe no container
if dim(1,i)~=dim(2,i) && dim(1,i)<=W && dim(2,i)<=L
box(c).quant=quant(i);
        box(c).dim=[dim(2,i); dim(1,i); dim(3,i)];
        box(c).vol=dim(1,i)*dim(2,i)*dim(3*i);
c=c+1;
end
if dim(1,i)~=dim(3,i) && dim(1,i)<=H && dim(3,i)<=L
box(c).quant=quant(i);
        box(c).dim=[dim(3,i); dim(2,i); dim(1,i)];
        box(c).vol=dim(1,i)*dim(2,i)*dim(3*i);
c=c+1;
end
if dim(2,i)~=dim(3,i) && dim(2,i)<=H && dim(3,i)<=W
box(c).quant=quant(i);
        box(c).dim=[dim(1,i); dim(3,i); dim(2,i)];
        box(c).vol=dim(1,i)*dim(2,i)*dim(3*i);
        c=c+1;
end
if dim(1,i)~=dim(3,i)&&dim(3,i)~=dim(2,i)&&dim(2,i)~=dim(1,i) &&
dim(1,i)<=H && dim(3,i)<=L && dim(3,i)<=W && dim(2,i)<=H &&
dim(2,i)<=L && dim(1,i)<=W
        box(c).quant=quant(i);
        box(c).dim=[dim(2,i); dim(3,i); dim(1,i)];
        box(c).vol=dim(1,i)*dim(2,i)*dim(3*i);
        c=c+1;
end
if dim(3,i)~=dim(1,i)&&dim(1,i)~=dim(2,i)&&dim(2,i)~=dim(3,i) &&
dim(1,i)<=H && dim(3,i)<=L && dim(3,i)<=W && dim(2,i)<=H &&
dim(2,i)<=L && dim(1,i)<=W
        box(c).quant=quant(i);
        box(c).dim=[dim(3,i); dim(1,i); dim(2,i)];
        box(c).vol=dim(1,i)*dim(2,i)*dim(3*i);
c=c+1;
end
    qtipo(i)=c-1;
end

fprintf('\n\n----------------------------Problema gerado--------------
--------------\n\nContêiner:\n\t\tComprimento: %d\n\t\tLargura:
%d\n\t\tAltura: %d\n\n',L,W,H)

for i=1:n
   fprintf('Caixa do tipo %d:\n\t\tQuantidade: %d\n\t\tComprimento:
%d\n\t\tLargura: %d\n\t\tAltura:
%d\n\n',i,quant(i),dim(1,i),dim(2,i),dim(3,i))
end
end
```

APÊNDICE B

Código fonte da função *model_generator*.

```
function [ fobj fobj_val restr restr_val box container qtipo estV_val
estV] = model_generator()
% Esta função gera o modelo de PL do problema de carregamento de
contêiner

[box container qtipo]=problem_generator();
tic
% container.dim=[2,2,1];
% box(1).quant=2;
% box(2).quant=1;
% box(1).dim=[1;1;1];
% box(2).dim=[2;1;2];
% box(1).vol=1;
% box(2).vol=4;
% container.dim=[10;5;4];
% box(1).quant=10;
% box(2).quant=5;
% box(1).dim=[3;2;1];
% box(2).dim=[2;3;2];
% box(1).vol=6;
% box(2).vol=12;

xmax = container.dim(1);
ymax = container.dim(2);
zmax = container.dim(3);
n=size(box,2);
% fobj(1).pos=[];
% fobj(1).val=0;
w=1;
%% contrução da função objetivo
ctipo=1;
ntip=zeros(1,size(qtipo,2));
for ii=1:n
    X=xmax-box(ii).dim(1);
Y=ymax-box(ii).dim(2);
Z=zmax-box(ii).dim(3);
nt=0;
for k=0:Z
for j=0:Y
for i=0:X
                fobj(w).tipo=ctipo;
                fobj(w).pos=[i;j;k];
                fobj(w).dim=box(ii).dim;
%                 fobj_val(w)=1;
                fobj_val(w)=box(ii).vol;
                w=w+1;
                nt=nt+1;
end
end
end
    ntip(ctipo)=ntip(ctipo)+nt;
if ii==qtipo(ctipo) % mapeia caixas com diferentes orientação mas do
```

```
mesmo tipo
        ctipo=ctipo+1;
end


end

%% Construção das restriçoes

%% Restrição de sobreposição
count=1;
for k=0:zmax-1
for j=0:ymax-1
for i=0:xmax-1
            stu(:,count)=[i; j; k];
            count=count+1;
end
end
end

for ii=1:size(stu,2)
w=1;
for ij=1:n
    X=xmax-box(ij).dim(1);
Y=ymax-box(ij).dim(2);
Z=zmax-box(ij).dim(3);
for k=0:Z
for j=0:Y
for i=0:X
                restr(ii,w).stu=stu(:,ii);
restr(ii,w).tipo=ij;
                restr(ii,w).pos=[i;j;k];
restr_val(ii,w)=ptsvol(stu(:,ii),box(ij).dim,i,j,k);
                w=w+1;
end
end
end
end
restr_val(ii,w)=1;
end

%% Restrição de quantidade de itens
indinic=0;

for i=1:size(qtipo,2)
    quant_val(i,1:size(fobj_val,2))=[zeros(1,indinic) ones(1,ntip(i))
zeros(1,size(fobj_val,2)-indinic-ntip(i))];
    indinic=indinic+ntip(i);
quant_val(i,size(fobj_val,2)+1)=box(qtipo(i)).quant;
end

restr_val=[restr_val;quant_val];

%% Restrição de estabilidade vertical
ct=1;
ctipo=1;
for ii=1:n

    X=xmax-box(ii).dim(1);
Y=ymax-box(ii).dim(2);
```

```
Z=zmax-box(ii).dim(3);

for k=1:Z
for j=0:Y
for i=0:X

for jlinha=j:j+box(ii).dim(2)-1
for ilinha=i:i+box(ii).dim(1)-1

for w=1:size(fobj_val,2)
estV(ct,w).pos=[ilinha jlinha k];

if fobj(w).pos(1)==i && fobj(w).pos(2)==j && fobj(w).pos(3)==k &&
fobj(w).tipo==ctipo && fobj(w).dim(1)==box(ii).dim(1) &&
fobj(w).dim(2)==box(ii).dim(2) && fobj(w).dim(3)==box(ii).dim(3)
                                    estV_val(ct,w)=-1;
else

estV_val(ct,w)=ptspt(fobj(w).pos,fobj(w).dim,[ilinha;jlinha;k]);
end
end
                              estV_val(ct,w+1)=0;
                              ct=ct+1;

end
end

end
end
end
if ii==qtipo(ctipo) % mapeia caixas com diferentes orientação mas do
mesmo tipo
ctipo=ctipo+1;
end
end
toc
```

APÊNDICE C

Código fonte da função *spxPadrao*.

```
function [rp] = spxPadrao( z1,A )
%Simplex Resolve problema pelo método SIMPLEX
%   Função que implementa o método simplex para problemas de
programação
%   linear padrão
%   Problema padrão:
%   Z=c1*x1+c2*x2+...+cn*xn
% sujeita a:
%   Restrições <=
%   a11*x1+a12*x2+...+a1n*xn <= b1
%   a21*x1+a22*x2+...+a2n*xn <= b2
%       :
%   am1*x1+am2*x2+...+amn*xn <= bm

% Onde aij, bi, cj são os coeficientes do sistema
% e bi deve ser maior que zero

    %% introduzindo as variaveis de folga

    z1 = z1(:)';
z1 = [0 -z1];
    iter=0;

if size(A,2)~=size(z1,2)
error('O número de variáveis cujos coeficientes são definidos por z1
não é compatível com o número de variáveis definidos nas
restrições.');
end

if size(A,2) < 3
        error('O sistema de restrições definidos por A precisa de pelo
menos duas variáveis.');
end

    Ac = A(:,1:(end-1));
    B = A(:,end);

if nnz(B<=0)>0
        error('Inequações de restrição precisam ser comparadas a
valores positivos');
end

    tam=size(Ac);
    npvar=tam(2);
z1=[z1 zeros(1,tam(1))]; %
Z+c1*x1+c2*x2+...+cn*xn+b1*F1+b2*F2+...bn*Fn=0
    Ac=[Ac eye(tam(1))];
    %% iniciando as variaveis básicas
for i=1:tam(1)
        VB(i,1)=i+tam(2);
```

```
end
for i=1:tam(2)
        VNB(i,1)=i;
end
flag=1;
    count=1;
    fmax=0; %guarda o valor da função no ponto máximo

while flag == 1
        [S var_e]=min(z1(2:end)); % determina a variavel de entrada
if S>=0
            [var rp]=calcResp(Ac,B,VNB,VB,npvar);
fprintf('\nEste problema possui solução única, com os seguintes
valores \n');
            fprintf('para as variáveis de projeto:\n')
fprintf('x%d = %f\n',[var rp]')
fprintf('função avaliada no ótimo = %f\n',func)
return
end

        ind_s=eqVarSaida( tam,var_e,Ac,B );
        var_deg=0;

if ind_s == inf
            z1aux=z1;
            var_deg=1;
end

while ind_s == inf

z1(var_e+1)=inf;
            [S var_e]=min(z1(2:end)); % determina a variavel de
entrada

if S>=0
                [var rp]=calcResp(Ac,B,VNB,VB,npvar);
fprintf('\nEste problema possui solução única, com os seguintes
valores \n');
                fprintf('para as variáveis de projeto:\n')
fprintf('x%d = %f\n',[var rp]')
fprintf('função avaliada no ótimo = %f\n',func)
return
end

            ind_s=eqVarSaida( tam,var_e,Ac,B );
end

if var_deg == 1
            z1=z1aux;
end
%% loop para determinar qual é a variavel de saída
for i=1:sum(tam)
            loopc=0;
if i==var_e
continue
end
for j=1:tam(2)
if i==VNB(j)
                    loopc=1; % dar continuidade no laço externo
end
```

```
end
if loopc==1
continue
end
if Ac(ind_s,i)~=0 %ERRO
var_s=i;
end
end

        %% divide a equação pelo indice da variavel de entrada
B(ind_s)=B(ind_s)/Ac(ind_s,var_e);
        Ac(ind_s,:)=Ac(ind_s,:)/Ac(ind_s,var_e);
%% faz com que a variavel de entrada apareça apenas uma vez no sistema
for i=1:tam(1)
if i==ind_s
continue
end
            B(i)=B(i)-Ac(i,var_e)*B(ind_s);
            Ac(i,:)=Ac(i,:)-Ac(i,var_e)*Ac(ind_s,:);
end

        z1=[z1(1)-(z1(var_e+1)*B(ind_s)) z1(2:end)-
z1(var_e+1)*Ac(ind_s,:)];
        func=z1(1);

        %% atualiza as variaveis VB e VNB
for i=1:tam(1)
if VB(i)==var_s
                aux=i;
end
end

for i=1:tam(2)
if VNB(i)==var_e
aux2=i;
end
end

        faux=VB(aux);
VB(aux)=VNB(aux2);
        VNB(aux2)=faux;
count=count+1;

if func > fmax
            fmax=func;
end


end
```

APÊNDICE D

Código fonte da função *simplexDuasFases*.

```
function [rp] = simplexDuasFases( z1,A,C,E )
%Simplex Duas Fases
%   Função que implementa o método simplex para problemas de
programação
%   linear
%   Problema:
%   Z=c1*x1+c2*x2+...+cn*xn
% sujeita a:
%   Restrições <=
%   a11*x1+a12*x2+...+a1n*xn <= b1
%   a21*x1+a22*x2+...+a2n*xn <= b2
%       :
%   am1*x1+am2*x2+...+amn*xn <= bm

%   Restrições >=
%   a11*x1+a12*x2+...+a1n*xn >= b1
%   a21*x1+a22*x2+...+a2n*xn >= b2
%       :
%   ap1*x1+ap2*x2+...+apn*xn >= bp

%   Restrições =
%   a11*x1+a12*x2+...+a1n*xn = b1
%   a21*x1+a22*x2+...+a2n*xn = b2
%       :
%   aq1*x1+aq2*x2+...+aqn*xn = bq


% Onde aij, bi, cj são os coeficientes do sistema
% e bi deve ser maior que zero
z1 = z1(:)';

tA=size(A); % A corresponde a matriz de restrições <=
tC=size(C); % C corresponde a matriz de restrições >=
tE=size(E); % E corresponde a matriz de restrições =
Ac = [A(:,1:(end-1));C(:,1:(end-1));E(:,1:(end-1))];
if isempty(E)==1 && isempty(C)==0
    B = [A(:,end);C(:,end)];
end
if isempty(E)==0 && isempty(C)==1
    B = [A(:,end);E(:,end)];
end
if isempty(E)==0 && isempty(C)==0
    B = [A(:,end);C(:,end);E(:,end)];
end

tam=size(Ac);
%% adicionando as variaveis de folga e artificiais

if isempty(A)==0
    k=1;
```

```
for i=1:tA(1)
Ac(:,tam(2)+i)=zeros(tam(1),1);
Ac(k,tam(2)+i)=1;
        k=k+1;
end
end
if isempty(C)==0
    k=1;
for i=1:tC(1)
        Ac(:,tam(2)+tA(1)+i)=zeros(tam(1),1);
        Ac(tA(1)+k,tam(2)+tA(1)+i)=-1;
k=k+1;
end
    k=1;
for i=1:tC(1)
Ac(:,tam(2)+tA(1)+tC(1)+i)=zeros(tam(1),1);
        Ac(tA(1)+k,tam(2)+tA(1)+tC(1)+i)=1;
k=k+1;
end
end
if isempty(E)==0
k=1;
for i=1:tE(1)
        Ac(:,tam(2)+tA(1)+2*tC(1)+i)=zeros(tam(1),1);
        Ac(tA(1)+tC(1)+k,tam(2)+tA(1)+2*tC(1)+i)=1;
k=k+1;
end
end

z1 = [0 -z1];
% if size(A,2)~=size(z1,2)|| size(C,2)~=size(z1,2)||
size(E,2)~=size(z1,2)
%     error('O número de variáveis cujos coeficientes são definidos
por z1 não é compatível com o número de variáveis definidos nas
restrições.');
% end

z1=[z1 zeros(1,(tA(1)+2*tC(1)+tE(1)))];

%% definindo a nova função objetivo

tam2=size(Ac);
w=zeros(1,tam2(2)+1);
w(1,(end-tC(1)-tE(1)+1:end))=1;


%% escalonando o sistema

for i=1:(tC(1)+tE(1))
for j=1:tam2(1)
if Ac(j,tam(2)+tA(1)+tC(1)+i)==1
            aux=[B(j) Ac(j,:)];
            w=w-aux;
end
end
end


%% obtendo as VB e as VNB
```

```
k=1;
l=1;
for i=1:tam2(2)
if w(1+i)==0
        VB(k)=i;
        k=k+1;
else
        VNB(l)=i;
        l=l+1;
end
end

VB=VB';
VNB=VNB';


% Ac
[var rp func VNBaux Ac B]=simplex1fase(w,Ac,B,VNB,VB,tam);
% VNBaux
% (tam2(2)-tE(1)-tC(1))
% (VNBaux(VNBaux>(tam2(2)-tE(1)-tC(1))))
% cond=length(VNBaux(VNBaux>(tam2(2)-tE(1)-tC(1))));

if abs(func)<=1e-12 %%&& cond>=tC(1)+tE(1)
    %% Retirando as variaveis artificiais
    Ac(:,tam2(2)-tC(1)-tE(1)+1:end)=[];
    z1(:,tam2(2)-tC(1)-tE(1)+2:end)=[];
    Ac
    z1
    %% reescalonando o sistema CONFERIR
for i=1:tam(2)
for j=1:tam2(1)
if Ac(j,i)~=0
                aux=z1(i+1)*[B(j) Ac(j,:)];
                z1=z1-aux;
end
end
end

%% obtendo as VB e as VNB
clear VBVNBvar
    k=1;
l=1;
    tam2=size(Ac);
for i=1:tam2(2)
if z1(1+i)==0
            VB(k)=i;
            k=k+1;
else
            VNB(l)=i;
            l=l+1;
end
end
    VB=VB';
    VNB=VNB';

[var rp func VNBaux Ac B]=simplex1fase(z1,Ac,B,VNB,VB,tam);
    fprintf('\nEste problema possui solução única, com os seguintes
valores \n');
    fprintf('para as variáveis de projeto:\n')
fprintf('x%d = %f\n',[var rp]')
```

```
fprintf('função avaliada no ótimo = %f\n',func)
end
end
```

APÊNDICE E

Código fonte da função *simplex1fase*.

```
function [var rp func VNB Ac B] = simplex1fase(z1,Ac,B,VNB,VB,tam)
% Simplex 1 fase
% função que implementa as fases do simplex de duas fases

    npvar=tam(2);
    flag=1;
    count=1;
    fmax=0; %guarda o valor da função no ponto máximo
    func=0;
while flag==1

    %% busca das variaveis de saída e entrada
    ind_s=[];
    var_e=[];
    [S var_e]=min(z1(2:end)); % determina a variavel de entrada


    sc=1;
    conj_var_e=[];
for i=2:length(z1)
if z1(i)==S
% conjunto das possiveis variaveis de entrada
            conj_var_e(sc)=i;
sc=sc+1;
end
end
    indc=randi(length(conj_var_e),1);
var_e=conj_var_e(indc)-1;

if S>=-1e-12
         [var,rp]=calcResp(Ac,B,VNB,VB,npvar);
return
end

ind_s=eqVarSaida( tam,var_e,Ac,B );
var_deg=0;

if ind_s == inf
        z1aux=z1;
        var_deg=1;
end

while ind_s == inf

z1(var_e+1)=inf;
        [S var_e]=min(z1(2:end)); % determina a variavel de entrada
        sc=1;
        conj_var_e=[];
for i=2:length(z1)
if z1(i)==S
% conjunto das possiveis variaveis de entrada
                conj_var_e(sc)=i;
```

```
sc=sc+1;
end
end
       indc=randi(length(conj_var_e),1);
var_e=conj_var_e(indc)-1;
if S>=-1e-12
            [var rp]=calcResp(Ac,B,VNB,VB,npvar);
return
end

ind_s=eqVarSaida( tam,var_e,Ac,B );
end
if var_deg == 1
        z1=z1aux;
end

        %% loop para determinar qual é a variavel de saída
tAc=size(Ac);
for i=1:tAc(2)
        loopc=0;
if i==var_e
continue
end
for j=1:length(VNB)
if i==VNB(j)
loopc=1; % dar continuidade no laço externo
end
end
if loopc==1
continue
end

if Ac(ind_s,i)~=0
            var_s=i;

end
end



        %% divide a equação pelo indice da variavel de entrada

B(ind_s)=B(ind_s)/Ac(ind_s,var_e);
    Ac(ind_s,:)=Ac(ind_s,:)/Ac(ind_s,var_e);

%% faz com que a variavel de entrada apareça apenas uma vez no sistema

for i=1:tam(1)
if i==ind_s
continue

end
        B(i)=B(i)-Ac(i,var_e)*B(ind_s);
        Ac(i,:)=Ac(i,:)-Ac(i,var_e)*Ac(ind_s,:);
end
    z1=[z1(1)-(z1(var_e+1)*B(ind_s)) z1(2:end)-
z1(var_e+1)*Ac(ind_s,:)];
    func=z1(1);

        %% atualiza as variaveis VB e VNB
```

```
for i=1:length(VB)
if VB(i)==var_s
            aux=i;
end
end
for i=1:length(VNB)
if VNB(i)==var_e
            aux2=i;
end
end
    faux=VB(aux);
VB(aux)=VNB(aux2);
    VNB(aux2)=faux;
count=count+1;

if func > fmax
        fmax=func;
end
end
end
```

APÊNDICE F

Código fonte da função *simplex1FaseEMeia.*

```
function [rp func] = simplex1FaseEMeia( z1,A,C )
%Simplex Duas Fases
%   Função que implementa o método simplex para problemas de
programação
%   linear
%   Problema:
%   Z=c1*x1+c2*x2+...+cn*xn
% sujeita a:
%   Restrições <=
%   a11*x1+a12*x2+...+a1n*xn <= b1
%   a21*x1+a22*x2+...+a2n*xn <= b2
%       :
%   am1*x1+am2*x2+...+amn*xn <= bm

%   Restrições >=
%   a11*x1+a12*x2+...+a1n*xn >= b1
%   a21*x1+a22*x2+...+a2n*xn >= b2
%       :
%   ap1*x1+ap2*x2+...+apn*xn >= bp

%   Restrições =
%   a11*x1+a12*x2+...+a1n*xn = b1
%   a21*x1+a22*x2+...+a2n*xn = b2
%       :
%   aq1*x1+aq2*x2+...+aqn*xn = bq


% Onde aij, bi, cj são os coeficientes do sistema
% e bi deve ser maior que zero
z1 = z1(:)';

tA=size(A); % A corresponde a matriz de restrições <=
tC=size(C); % C corresponde a matriz de restrições >=

Ac = [A(:,1:(end-1));C(:,1:(end-1))];
B = [A(:,end);C(:,end)];

tam=size(Ac);
%% adicionando as variaveis de folga e artificiais

if isempty(A)==0
    k=1;
for i=1:tA(1)
Ac(:,tam(2)+i)=zeros(tam(1),1);
Ac(k,tam(2)+i)=1;
        k=k+1;
end
end
if isempty(C)==0
    k=1;
for i=1:tC(1)
        Ac(:,tam(2)+tA(1)+i)=zeros(tam(1),1);
```

```
        Ac(tA(1)+k,tam(2)+tA(1)+i)=-1;
Ac(tA(1)+k,:)=-1*Ac(tA(1)+k,:);
        k=k+1;
end

end

z1 = [0 -z1];
% if size(A,2)~=size(z1,2)|| size(C,2)~=size(z1,2)||
size(E,2)~=size(z1,2)
%     error('O número de variáveis cujos coeficientes são definidos
por z1 não é compatível com o número de variáveis definidos nas
restrições.');
% end

z1=[z1 zeros(1,(tA(1)+tC(1)))];
tam2=size(Ac);
fprintf('\n%d variaveis',tam2(2))
%% obtendo as VB e as VNB

k=1;
l=1;
for i=1:tam2(2)
if z1(1+i)==0
        VB(k)=i;
        k=k+1;
else
        VNB(l)=i;
        l=l+1;
end
end

VB=VB';
VNB=VNB';

[var rp func VNBaux Ac B]=simplex1fase(z1,Ac,B,VNB,VB,tam);
fprintf('\nEste problema possui uma solução, com os seguintes valores
\n');
fprintf('para as variáveis de projeto:\n')
fprintf('x%d = %f\n',[var rp]')
fprintf('função avaliada no ótimo = %f\n',func)


end
```

APÊNDICE G

Código fonte da função *drawResult.*

```
function [] = drawResult( fobj,rp,box,container,qtipo,fmax )
L = container.dim(1);
W = container.dim(2);
H = container.dim(3);
plotcube([L W H],[0 0 0],0,[1 1 1]);
ntipos = size(box,2);
colors = rand(ntipos,3);
k=zeros(1,size(qtipo,2));
for i=1:size(fobj,2)
    n=fobj(i).tipo;
coord=fobj(i).pos';
if rp(i)==1
plotcube(fobj(i).dim',coord,.8,colors(n,:));
        k(n)=k(n)+1;
end

end
axis equal
fprintf('\n\n')
for i=1:size(qtipo,2)
   fprintf('Caixa do tipo %d alocadas:\n\t\tQuantidade:
%d\n\n',i,k(i))
end
fprintf('porcentagem de utilização =
%f\n',100*fmax/(container.dim(1)*container.dim(2)*container.dim(3)))
end
```

APÊNDICE H

Código fonte das funções *createPDDL* e *createPDDL2*.

createPDDL

```
function [ ] = createPDDL( box,rp,fobj,container )

%% domínio
%% cabeçalho

fileID=fopen('D:\SkyDrive\Mestrado\Dissertação\dominio.pddl','w');
fprintf(fileID,'(define (domain New_Project_1)\n');
fprintf(fileID,'    (:requirements :typing :fluents :negative-
preconditions :equality)\n');

%% tipos

fprintf(fileID,'    (:types\n      ponteRolante - object\n      lugar
- object\n');
for i=1:size(box,2);
    produtos=sprintf('      produto%d - object\n',i);
    fprintf(fileID,produtos);
end
fprintf(fileID,'    )\n');

%% predicados

fprintf(fileID,'    (:predicates\n');
for i=1:size(box,2)
    estaEm=sprintf('      (estaEm%d ?pro - produto%d ?lug -
lugar)\n',i,i);
    estaNa=sprintf('      (estaNa%d ?pro - produto%d ?pon -
ponteRolante)\n',i,i);
    fprintf(fileID,estaEm);
    fprintf(fileID,estaNa);
end
fprintf(fileID,'      (ocupada ?pon - ponteRolante)\n');
fprintf(fileID,'      (disponivel ?lug - lugar)\n');
fprintf(fileID,'      (ocupado ?lug - lugar)\n');
fprintf(fileID,'    )\n');

%% funções

fprintf(fileID,'    (:functions\n');
fprintf(fileID,'      (coordx ?pon - ponteRolante)\n');
fprintf(fileID,'      (coordy ?pon - ponteRolante)\n');
for i=1:size(box,2)
    coordenadaX=sprintf('      (crdx%d ?pro - produto%d)\n',i,i);
    coordenadaY=sprintf('      (crdy%d ?pro - produto%d)\n',i,i);
    coordenadaZ=sprintf('      (crdz%d ?pro - produto%d)\n',i,i);
    fprintf(fileID,coordenadaX);
    fprintf(fileID,coordenadaY);
    fprintf(fileID,coordenadaZ);
end
fprintf(fileID,'      (cx ?lug - lugar)\n');
```

```
fprintf(fileID,'       (cy ?lug - lugar)\n');
fprintf(fileID,'       (cz ?lug - lugar)\n');
fprintf(fileID,'    )\n');

%% ação mover

fprintf(fileID,'    (:action mover\n');
fprintf(fileID,'     :parameters (?p1 - ponteRolante ?l1 - lugar ?l2 -
lugar)\n');
fprintf(fileID,'     :precondition\n');
fprintf(fileID,'       (and\n');
fprintf(fileID,'         (= (coordx ?p1) (cx ?l1))\n');
fprintf(fileID,'         (= (coordy ?p1) (cy ?l1))\n');
fprintf(fileID,'         (not (= ?l1 ?l2))\n');
fprintf(fileID,'         (= (cz ?l1) 0)\n');
fprintf(fileID,'         (= (cz ?l2) 0)\n');
fprintf(fileID,'       )\n');
fprintf(fileID,'     :effect\n');
fprintf(fileID,'       (and\n');
fprintf(fileID,'         (assign (coordx ?p1) (cx ?l2))\n');
fprintf(fileID,'         (assign (coordy ?p1) (cy ?l2))\n');
fprintf(fileID,'       )\n');
fprintf(fileID,'    )\n');

%% ação pegar

for i=1:size(box,2)

action=sprintf('    (:action pegar%d\n',i);
fprintf(fileID,action);
    parametros=sprintf('     :parameters (?p1 - ponteRolante ?prod -
produto%d',i);
    fprintf(fileID,parametros);
    lugares=box(i).dim(1)*box(i).dim(2)*(box(i).dim(3)+1);

% lugares necessarios para a função

for j=1:lugares
        lugar=sprintf(' ?l%d - lugar',j);
fprintf(fileID,lugar);
end
    fprintf(fileID,')\n');

% precondições

    fprintf(fileID,'     :precondition\n');
    fprintf(fileID,'       (and\n');
    fprintf(fileID,'         (not (ocupada ?p1))\n');
    estaEm=sprintf('         (estaEm%d ?prod ?l1)\n',i);
    fprintf(fileID,estaEm);

% locais ocupados

    w=1;
for o=1:box(i).dim(3)
for n=1:box(i).dim(2)
for m=1:box(i).dim(1)
                ocupado=sprintf('         (ocupado ?l%d)\n',w);
                fprintf(fileID,ocupado);
                w=w+1;
```

```
end
end
end

% locais não ocupados

for n=1:box(i).dim(2)
for m=1:box(i).dim(1)
                nOcupado=sprintf('          (not (ocupado ?l%d))\n',w);
fprintf(fileID,nOcupado);
                w=w+1;
end
end

% coordenadas da ponte rolante

    fprintf(fileID,'          (= (coordx ?p1) (cx ?l1))\n');
fprintf(fileID,'          (= (coordy ?p1) (cy ?l1))\n');

% restrições de lugares

    w=1;
for o=1:(box(i).dim(3)+1)
for n=1:box(i).dim(2)
for m=1:box(i).dim(1)
if(o==1 && n==1 && m==1)
w=w+1;
continue
end
                cx=m-1;
                cy=n-1;
                cz=o-1;
                scx=sprintf('          (= (cx ?l1) (- (cx ?l%d)
%d))\n',w,cx);
                scy=sprintf('          (= (cy ?l1) (- (cy ?l%d)
%d))\n',w,cy);
                scz=sprintf('          (= (cz ?l1) (- (cz ?l%d)
%d))\n',w,cz);
                fprintf(fileID,scx);
                fprintf(fileID,scy);
                fprintf(fileID,scz);
                w=w+1;
end
end
end
    fprintf(fileID,'       )\n');

% efeitos

    fprintf(fileID,'     :effect\n');
    fprintf(fileID,'       (and\n');
    estaNa=sprintf('         (estaNa%d ?prod ?p1)\n',i);
    fprintf(fileID,estaNa);
    estaEm=sprintf('         (not (estaEm%d ?prod ?l1))\n',i);
fprintf(fileID,estaEm);
    fprintf(fileID,'         (ocupada ?p1)\n');

% locais que deixaram de estar ocupados

    w=1;
```

```
for o=1:box(i).dim(3)
for n=1:box(i).dim(2)
for m=1:box(i).dim(1)
                ocupado=sprintf('          (not (ocupado ?l%d))\n',w);
                fprintf(fileID,ocupado);
                w=w+1;
end
end
end

% locais que deixaram de estar disponíveis

    aux=(box(i).dim(1)*box(i).dim(2))+1; %começa a numeração de caixa
em z=2
for o=1:box(i).dim(3)
for n=1:box(i).dim(2)
for m=1:box(i).dim(1)
                ocupado=sprintf('          (not (disponivel
?l%d))\n',aux);
                fprintf(fileID,ocupado);
aux=aux+1;
end
end
end

    scrdx=sprintf('          (assign (crdx%d ?prod) (- 0 1))\n',i);
    fprintf(fileID,scrdx);
    scrdy=sprintf('          (assign (crdy%d ?prod) (- 0 1))\n',i);
    fprintf(fileID,scrdy);
    scrdz=sprintf('          (assign (crdz%d ?prod) (- 0 1))\n',i);
    fprintf(fileID,scrdz);
    fprintf(fileID,'       )\n');
    fprintf(fileID,'    )\n');

end

%% ação soltar

for i=1:size(box,2)
    action=sprintf('    (:action soltar%d\n',i);
fprintf(fileID,action);
    parametros=sprintf('     :parameters (?p1 - ponteRolante ?prod -
produto%d',i);
    fprintf(fileID,parametros);
    lugares=box(i).dim(1)*box(i).dim(2)*(box(i).dim(3)+1);

% lugares necessarios para a função

for j=1:lugares
        lugar=sprintf(' ?l%d - lugar',j);
fprintf(fileID,lugar);
end
    fprintf(fileID,')\n');

% precondições

    fprintf(fileID,'     :precondition\n');
    fprintf(fileID,'       (and\n');
fprintf(fileID,'          (ocupada ?p1)\n');
    estaNa=sprintf('          (estaNa%d ?prod ?p1)\n',i);
```

```
    fprintf(fileID,estaNa);

% locais não ocupados

    w=1;
for o=1:box(i).dim(3)+1
for n=1:box(i).dim(2)
for m=1:box(i).dim(1)
                nOcupado=sprintf('          (not (ocupado ?l%d))\n',w);
                fprintf(fileID,nOcupado);
w=w+1;
end
end
end

% locais disponíveis

w=1;
for n=1:box(i).dim(2)
for m=1:box(i).dim(1)
            disponivel=sprintf('          (disponivel ?l%d)\n',w);
fprintf(fileID,disponivel);
            w=w+1;
end
end

% coordenadas da ponte

    fprintf(fileID,'          (= (coordx ?p1) (cx ?l1))\n');
fprintf(fileID,'          (= (coordy ?p1) (cy ?l1))\n');

% restrições de lugares

    w=1;
for o=1:(box(i).dim(3)+1)
for n=1:box(i).dim(2)
for m=1:box(i).dim(1)
if(o==1 && n==1 && m==1)
w=w+1;
continue
end
                cx=m-1;
                cy=n-1;
                cz=o-1;
                scx=sprintf('          (= (cx ?l1) (- (cx ?l%d)
%d))\n',w,cx);
                scy=sprintf('          (= (cy ?l1) (- (cy ?l%d)
%d))\n',w,cy);
                scz=sprintf('          (= (cz ?l1) (- (cz ?l%d)
%d))\n',w,cz);
                fprintf(fileID,scx);
                fprintf(fileID,scy);
                fprintf(fileID,scz);
                w=w+1;
end
end
end
    fprintf(fileID,'       )\n');

% efeitos
```

```
    fprintf(fileID,'     :effect\n');
    fprintf(fileID,'       (and\n');
    estaNa=sprintf('         (not (estaNa%d ?prod ?p1))\n',i);
    fprintf(fileID,estaNa);
    estaEm=sprintf('         (estaEm%d ?prod ?l1)\n',i);
    fprintf(fileID,estaEm);
    fprintf(fileID,'         (not (ocupada ?p1))\n');

% locais ocupados

    w=1;
for o=1:box(i).dim(3)
for n=1:box(i).dim(2)
for m=1:box(i).dim(1)
                ocupado=sprintf('         (ocupado ?l%d)\n',w);
                fprintf(fileID,ocupado);
                w=w+1;
end
end
end

% locais disponíveis

    aux=(box(i).dim(1)*box(i).dim(2))+1; %começa a numeração de caixa
em z=box.dim(3)+1
for o=1:box(i).dim(3)
for n=1:box(i).dim(2)
for m=1:box(i).dim(1)
                disponivel=sprintf('         (disponivel
?l%d)\n',aux);
                fprintf(fileID,disponivel);
                aux=aux+1;
end
end
end

    scrdx=sprintf('         (assign (crdx%d ?prod) (cx ?l1))\n',i);
    fprintf(fileID,scrdx);
    scrdy=sprintf('         (assign (crdy%d ?prod) (cy ?l1))\n',i);
    fprintf(fileID,scrdy);
    scrdz=sprintf('         (assign (crdz%d ?prod) (cz ?l1))\n',i);
    fprintf(fileID,scrdz);

    fprintf(fileID,'       )\n');
    fprintf(fileID,'    )\n');
end
fprintf(fileID,')');
fclose(fileID);
%% problema

fileID=fopen('D:\SkyDrive\Mestrado\Dissertação\problema.pddl','w');
fprintf(fileID,'(define (problem Planning_Problem)\n');
fprintf(fileID,'    (:domain New_Project_1)\n');
fprintf(fileID,'    (:objects\n');
fprintf(fileID,'      p1 - ponteRolante\n');

% lugares do container

nLugar=container.dim(1)*container.dim(2)*(container.dim(3)+1);
```

```
for i=1:nLugar
    lugar=sprintf('      l%d - lugar\n',i);
    fprintf(fileID,lugar);
end

% produtos e lugares

nLugar=nLugar+1;
w=1;
for i=1:size(rp,2)
if rp(i)==1
for j=1:size(box,2)
if fobj(i).dim == box(j).dim
                produto=sprintf('      prod%d - produto%d\n',w,j);
                fprintf(fileID,produto);
                w=w+1;
                lugProd=box(j).dim(1)*box(j).dim(2)*(box(j).dim(3)+1);
for k=1:lugProd
                    lugar=sprintf('      l%d - lugar\n',nLugar);
                    fprintf(fileID,lugar);
                    nLugar=nLugar+1;
end
break
end
end
end
end
fprintf(fileID,'    )\n');
%% snapshot inicial

fprintf(fileID,'    (:init\n');

% configuração dos lugares vazios
fprintf(fileID,'      (= (coordx p1) -10)\n');
fprintf(fileID,'      (= (coordy p1) -10)\n');

w=1;
for k=1:container.dim(3)+1
for j=1:container.dim(2)
for i=1:container.dim(1)
if k <= 1
                disponivel=sprintf('      (disponivel l%d)\n',w);
                fprintf(fileID,disponivel);
end
            scx=sprintf('      (= (cx l%d) %d)\n',w,(i-1));
            scy=sprintf('      (= (cy l%d) %d)\n',w,(j-1));
            scz=sprintf('      (= (cz l%d) %d)\n',w,(k-1));
            fprintf(fileID,scx);
            fprintf(fileID,scy);
            fprintf(fileID,scz);
            w=w+1;
end
end
end

% configuração dos lugares iniciais dos produtos

nLugar=(container.dim(1)*container.dim(2)*(container.dim(3)+1))+1;
w=1;
```

```
for i=1:size(rp,2)
if rp(i)==1
for j=1:size(box,2)
if fobj(i).dim == box(j).dim
                xinit=-10;
                yinit=-10;
                zinit=0;
                estaEm=sprintf('      (estaEm%d prod%d
l%d)\n',j,w,nLugar);
                scrdx=sprintf('      (= (crdx%d prod%d)
%d)\n',j,w,xinit);
                scrdy=sprintf('      (= (crdy%d prod%d)
%d)\n',j,w,yinit);
                scrdz=sprintf('      (= (crdz%d prod%d)
%d)\n',j,w,zinit);
                fprintf(fileID,estaEm);
                fprintf(fileID,scrdx);
                fprintf(fileID,scrdy);
                fprintf(fileID,scrdz);
                lugProd=box(j).dim(1)*box(j).dim(2)*(box(j).dim(3)+1);
for m=1:(box(j).dim(3)+1)
                    yinit=-10;
for n=1:box(j).dim(2)
                        xinit=-10;
for o=1:box(j).dim(1)
if m <=box(j).dim(3)
ocupado=sprintf('      (ocupado l%d)\n',nLugar);
                                fprintf(fileID,ocupado);
end
                            disponivel=sprintf('      (disponivel
l%d)\n',nLugar);
scx=sprintf('      (= (cx l%d) %d)\n',nLugar,xinit);
                            scy=sprintf('      (= (cy l%d)
%d)\n',nLugar,yinit);
                            scz=sprintf('      (= (cz l%d)
%d)\n',nLugar,zinit);
                            fprintf(fileID,disponivel);
                            fprintf(fileID,scx);
                            fprintf(fileID,scy);
                            fprintf(fileID,scz);
                            xinit=xinit+1;
                            nLugar=nLugar+1;
end
                        yinit=yinit+1;
end
                    zinit=zinit+1;
end
                w=w+1;

break
end
end
end

end
fprintf(fileID,'    )\n');

%% snapshot final

fprintf(fileID,'    (:goal\n');
```

```
fprintf(fileID,'      (and\n');
fprintf(fileID,'        (not (ocupada p1))\n');
w=1;
for i=1:size(rp,2)
if rp(i) == 1
for j=1:size(box,2)
if fobj(i).dim == box(j).dim
                scrdx=sprintf('        (= (crdx%d prod%d)
%d)\n',j,w,fobj(i).pos(1));
                scrdy=sprintf('        (= (crdy%d prod%d)
%d)\n',j,w,fobj(i).pos(2));
                scrdz=sprintf('        (= (crdz%d prod%d)
%d)\n',j,w,fobj(i).pos(3));
                fprintf(fileID,scrdx);
                fprintf(fileID,scrdy);
                fprintf(fileID,scrdz);
                w=w+1;
end;
end
end
end
fprintf(fileID,'      )\n');
fprintf(fileID,'    )\n');
fprintf(fileID,')\n');
fclose(fileID);
fclose('all');
fprintf('Arquivos gerados com sucesso\n\n');
end
```

createPDDL2

```
function [ ] = createPDDL2( box,rp,fobj,container )

%% domínio
%% cabeçalho

fileID=fopen('D:\OneDrive\Mestrado\Dissertação\dominio.pddl','w');
fprintf(fileID,'(define (domain New_Project_1)\n');
fprintf(fileID,'    (:requirements :strips)\n');

%% tipos

fprintf(fileID,'    (:types\n      lugar - object\n');
for i=1:size(box,2);
    produtos=sprintf('      produto%d - object\n',i);
    fprintf(fileID,produtos);
end
fprintf(fileID,'    )\n');

%% predicados

fprintf(fileID,'    (:predicates\n');
for i=1:size(box,2)
    estaEm=sprintf('      (estaEm%d ?pro - produto%d ?lug -
lugar)\n',i,i);
    fprintf(fileID,estaEm);
    livre=sprintf('      (plivre%d ?prod - produto%d)\n',i,i);
    fprintf(fileID,livre);
```

```
end
fprintf(fileID,'      (atras ?lug - lugar ?lug - lugar)\n');
fprintf(fileID,'      (esquerda ?lug - lugar ?lug - lugar)\n');
fprintf(fileID,'      (acima ?lug - lugar ?lug - lugar)\n');
fprintf(fileID,'      (livre ?lug - lugar)\n');
fprintf(fileID,'      (piso ?lug - lugar)\n');
fprintf(fileID,'    )\n');


%% ação moverPiso

for i=1:size(box,2)

    action=sprintf('    (:action moverPiso%d\n',i);
    fprintf(fileID,action);
    parametros=sprintf('     :parameters (?prod - produto%d',i);
    fprintf(fileID,parametros);
    lugares=box(i).dim(1)*box(i).dim(2)*box(i).dim(3);

    % lugares necessarios para a função

    for j=1:lugares
        lugar=sprintf(' ?l%d - lugar',j);
        fprintf(fileID,lugar);
    end
    fprintf(fileID,')\n');

    % precondições

    fprintf(fileID,'     :precondition\n');
    fprintf(fileID,'       (and\n');
    fprintf(fileID,'         (plivre%d ?prod)\n',i);

    w=1;

    for n=1:box(i).dim(2)
        for m=1:box(i).dim(1)
            livre=sprintf('          (livre ?l%d)\n',w);
            piso=sprintf('          (piso ?l%d)\n',w);
            fprintf(fileID,livre);
            fprintf(fileID,piso);
            w=w+1;
        end
    end


    % restrições de lugares

    w=1;
    for k=1:(box(i).dim(3))
        for j=1:(box(i).dim(2))
            for h=1:(box(i).dim(1)-1)
                atras=sprintf('          (atras ?l%d ?l%d)\n',w+1,w);
                fprintf(fileID,atras);
                w=w+1;
            end
            w=w+1;
        end
    end
    w=1;
```

```
    for k=1:(box(i).dim(3))
        for j=1:(box(i).dim(2)-1)
            esquerda=sprintf('          (esquerda ?l%d
?l%d)\n',w+box(i).dim(1),w);
            fprintf(fileID,esquerda);
            w=w+box(i).dim(1);
        end
        w=w+box(i).dim(1);
    end
    w=1;
    for k=1:(box(i).dim(3)-1)
        abaixo=sprintf('          (acima ?l%d
?l%d)\n',w+(box(i).dim(1)*box(i).dim(2)),w);
        fprintf(fileID,abaixo);
        w=w+(box(i).dim(1)*box(i).dim(2));
    end
    fprintf(fileID,'       )\n');

    % efeitos

    fprintf(fileID,'     :effect\n');
    fprintf(fileID,'       (and\n');
    estaEm=sprintf('         (estaEm%d ?prod ?l1)\n',i);
    fprintf(fileID,estaEm);
    livre=sprintf('          (not (plivre%d ?prod))\n',i);
    fprintf(fileID,livre);

    % locais que ficaram ocupados

    w=1;
    for o=1:box(i).dim(3)
        for n=1:box(i).dim(2)
            for m=1:box(i).dim(1)
                livre=sprintf('          (not (livre ?l%d))\n',w);
                fprintf(fileID,livre);
                w=w+1;
            end
        end
    end

    fprintf(fileID,'       )\n');
    fprintf(fileID,'    )\n');

end

%% ação empilhar

for i=1:size(box,2)
    action=sprintf('    (:action empilhar%d\n',i);
    fprintf(fileID,action);
    parametros=sprintf('     :parameters (?prod - produto%d',i);
    fprintf(fileID,parametros);
    lugares=box(i).dim(1)*box(i).dim(2)*(box(i).dim(3)+1);

    % lugares necessarios para a função

    for j=1:lugares
        lugar=sprintf(' ?l%d - lugar',j);
        fprintf(fileID,lugar);
    end
```

```
    fprintf(fileID,')\n');

    % precondições

    fprintf(fileID,'     :precondition\n');
    fprintf(fileID,'       (and\n');
    fprintf(fileID,'         (plivre%d ?prod)\n',i);

    % locais não ocupados

    w=1;

    for n=1:box(i).dim(2)
        for m=1:box(i).dim(1)
            Ocupado=sprintf('          (not (livre ?l%d))\n',w);
            fprintf(fileID,Ocupado);
            w=w+1;
        end
    end


    % restrições de lugares

    w=box(i).dim(1)*box(i).dim(2)+1;
    for k=1:(box(i).dim(3))
        for j=1:(box(i).dim(2))
            for h=1:(box(i).dim(1)-1)
                atras=sprintf('          (atras ?l%d ?l%d)\n',w+1,w);
                fprintf(fileID,atras);
                w=w+1;
            end
            w=w+1;
        end
    end
    w=box(i).dim(1)*box(i).dim(2)+1;
    for k=1:(box(i).dim(3))
        for j=1:(box(i).dim(2)-1)
            esquerda=sprintf('          (esquerda ?l%d
?l%d)\n',w+box(i).dim(1),w);
            fprintf(fileID,esquerda);
            w=w+box(i).dim(1);
        end
        w=w+box(i).dim(1);
    end
    w=box(i).dim(1)*box(i).dim(2)+1;
    for k=1:(box(i).dim(3)-1)
        abaixo=sprintf('          (acima ?l%d
?l%d)\n',w+(box(i).dim(1)*box(i).dim(2)),w);
        fprintf(fileID,abaixo);
        w=w+(box(i).dim(1)*box(i).dim(2));
    end
    w=1;
    for j=1:(box(i).dim(2))
        for h=1:(box(i).dim(1))
        abaixo=sprintf('          (acima ?l%d
?l%d)\n',w+(box(i).dim(1)*box(i).dim(2)),w);
        fprintf(fileID,abaixo);
        w=w+1;
        end
    end
```

```
    fprintf(fileID,'       )\n');

    % efeitos

    fprintf(fileID,'     :effect\n');
    fprintf(fileID,'       (and\n');
    estaEm=sprintf('         (estaEm%d ?prod
?l%d)\n',i,box(i).dim(1)*box(i).dim(2)+1);
    fprintf(fileID,estaEm);
    livre=sprintf('         (not (plivre%d ?prod))\n',i);
    fprintf(fileID,livre);


    % locais ocupados

    w=box(i).dim(1)*box(i).dim(2)+1;
    for o=1:box(i).dim(3)
        for n=1:box(i).dim(2)
            for m=1:box(i).dim(1)
                ocupado=sprintf('         (not (livre ?l%d))\n',w);
                fprintf(fileID,ocupado);
                w=w+1;
            end
        end
    end

    % locais disponíveis

    fprintf(fileID,'       )\n');
    fprintf(fileID,'    )\n');
end
fprintf(fileID,')');
fclose(fileID);
%% problema

fileID=fopen('D:\OneDrive\Mestrado\Dissertação\problema.pddl','w');
fprintf(fileID,'(define (problem Planning_Problem)\n');
fprintf(fileID,'    (:domain New_Project_1)\n');
fprintf(fileID,'    (:objects\n');

% lugares do container

nLugar=container.dim(1)*container.dim(2)*container.dim(3);

for i=1:nLugar
    lugar=sprintf('      l%d - lugar\n',i);
    fprintf(fileID,lugar);
end

% produtos

w=1;
for i=1:size(rp,2)
    if rp(i)==1
        for j=1:size(box,2)
            if fobj(i).dim == box(j).dim
                produto=sprintf('      prod%d - produto%d\n',w,j);
                fprintf(fileID,produto);
```

```
                w=w+1;
                break
            end
        end
    end
end
fprintf(fileID,'    )\n');
%% snapshot inicial

fprintf(fileID,'    (:init\n');

% configuração dos lugares vazios
w=1;
for k=1:(container.dim(3))
    for j=1:(container.dim(2))
        for h=1:(container.dim(1))
            livre=sprintf('      (livre l%d)\n',w);
            fprintf(fileID,livre);
            if w<=(container.dim(1)*container.dim(2))
                piso=sprintf('      (piso l%d)\n',w);
                fprintf(fileID,piso);
            end
            w=w+1;
        end
    end
end
w=1;
for k=1:(container.dim(3))
    for j=1:(container.dim(2))
        for h=1:(container.dim(1)-1)
            atras=sprintf('      (atras l%d l%d)\n',w+1,w);
            fprintf(fileID,atras);
            w=w+1;
        end
        w=w+1;
    end
end
w=1;
for k=1:(container.dim(3))
    for j=1:(container.dim(2)-1)
        for h=1:(container.dim(1))
            esquerda=sprintf('      (esquerda l%d
l%d)\n',w+container.dim(1),w);
            fprintf(fileID,esquerda);
            w=w+1;
        end
    end
    w=w+container.dim(1);
end
w=1;
for k=1:(container.dim(3)-1)
    for j=1:container.dim(2)
        for h=1:container.dim(1)
            abaixo=sprintf('      (acima l%d
l%d)\n',w+(container.dim(1)*container.dim(2)),w);
            fprintf(fileID,abaixo);
            w=w+1;
        end
    end
end
```

```
% configuração dos lugares iniciais dos produtos

w=1;

for i=1:size(rp,2)
    if rp(i)==1
        for j=1:size(box,2)
            if fobj(i).dim == box(j).dim
                livre=sprintf('      (plivre%d prod%d)\n',j,w);
                fprintf(fileID,livre);
                w=w+1;
                break
            end
        end
    end
end
fprintf(fileID,'    )\n');

%% snapshot final

fprintf(fileID,'    (:goal\n');
fprintf(fileID,'      (and\n');
w=1;

for i=1:size(rp,2)
     if rp(i) == 1
        coordx=fobj(i).pos(1);
        coordy=fobj(i).pos(2);
        coordz=fobj(i).pos(3);
        cz=0;
        count=1;
        for o=1:container.dim(3)
            cy=0;
            for m=1:container.dim(2)
                cx=0;
                for n=1:container.dim(1)
                    if coordx==cx && coordy==cy && coordz==cz
                        for tp=1:size(box,2)
                            if fobj(i).dim(1)==box(tp).dim(1) &&
fobj(i).dim(2)==box(tp).dim(2) && fobj(i).dim(3)==box(tp).dim(3)
                                estaEm=sprintf('        (estaEm%d
prod%d l%d)\n',tp,w,count);
                                fprintf(fileID,estaEm);
                                w=w+1;
                            end
                        end
                    end
                    cx=cx+1;
                    count=count+1;
                end
                cy=cy+1;
            end
            cz=cz+1;
        end
     end
end
fprintf(fileID,'      )\n');
fprintf(fileID,'    )\n');
fprintf(fileID,')\n');
fclose(fileID);
fclose('all');
```

```
fprintf('Arquivos gerados com sucesso\n\n');
end
```

APÊNDICE I

Código fonte das funções complementares do algoritmo em MatLab.

Função *main*

```
clc
clear all
close all

[fobj fobj_val restr restr_val box container qtipo estV_val
estV]=model_generator();
tic
[rp fmax]=simplex1FaseEMeia(fobj_val,restr_val,estV_val);
toc
drawResult(fobj,rp',box,container,qtipo,fmax)
createPDDL( box,rp',fobj,container )
```

Função *calcResp*

```
function [var rp] = calcResp(Ac,B,VNB,VB,npvar)
% função para resolução do sistema de equações para determinar
% o valor das variaveisde projeto

VNBaux=sort(VNB);
Aaux2=Ac;
for i=length(VNB):-1:1
    Aaux2(:,VNBaux(i))=[];
end
resp=Aaux2\B;
VBaux=sort(VB);
V=[VNBaux;VBaux];
resp2=[zeros(length(VNBaux),1);resp];
a = [V(V<=npvar) resp2(V<=npvar)];
[var,idx] = sort(a(:,1));
rp = a(idx,2);

end
```

Função *eqVarSaida*

```
function [ ind_s  ] = eqVarSaida( tam,var_e,A,B )
    Aaux=[];
for i=1:tam(1) % impede que valores negativos afetem a restrição das
variaveis de folga
if A(i,var_e) < 0
            Aaux(i)=0;
else
            Aaux(i)=A(i,var_e);
end
end

    [S ind_s]=min(B./Aaux'); % determina qual equação contém a
variável de saída
```

```
if S == inf
        error('a solução é irrestrita');
end

if S == 0
        ind_s=inf;
end

end
```

Função *ptspt*.

```
function [ val ] = ptspt( pos,dim,pt )
z=pos(3)+dim(3);
val=0;
if z==pt(3)
for j=pos(2):pos(2)+dim(2)-1
for i=pos(1):pos(1)+dim(1)-1
if i==pt(1)&&j==pt(2)
                val=1;
end
end
end

end
```

Função *ptsvol*.

```
function [ flag ] = ptsvol( stu,boxdim,x,y,z)

% função que verifica se um ponto de coordenadas [s t u] está dentro
de
% uma caixa com dimensões boxdim e alocada em [x y z]

xmin=x;
ymin=y;
zmin=z;
xmax=x+boxdim(1)-1;
ymax=y+boxdim(2)-1;
zmax=z+boxdim(3)-1;
flag=0;

for k=z:zmax
for j=y:ymax
for i=x:xmax
if stu(1)==i && stu(2)==j && stu(3)==k
              flag=1;
end
end
end
end
end
```

APENDICE J

Problemas de planejamento automático da seção de resultados.

Problema 1

dominio1.pddl modelo completo

```
(define (domain New_Project_1)
    (:requirements :typing :fluents :negative-preconditions
:equality)
    (:types
      ponteRolante – object
      lugar - object
      produto1 - object
    )
    (:predicates
      (estaEm1 ?pro - produto1 ?lug - lugar)
      (estaNa1 ?pro - produto1 ?pon - ponteRolante)
      (ocupada ?pon - ponteRolante)
      (disponivel ?lug - lugar)
      (ocupado ?lug - lugar)
    )
    (:functions
      (coordx ?pon - ponteRolante)
      (coordy ?pon - ponteRolante)
      (crdx1 ?pro - produto1)
      (crdy1 ?pro - produto1)
      (crdz1 ?pro - produto1)
      (cx ?lug - lugar)
      (cy ?lug - lugar)
      (cz ?lug - lugar)
    )
    (:action mover
     :parameters (?p1 - ponteRolante ?l1 - lugar ?l2 - lugar)
     :precondition
       (and
         (= (coordx ?p1) (cx ?l1))
         (= (coordy ?p1) (cy ?l1))
         (not (= ?l1 ?l2))
         (= (cz ?l1) 0)
         (= (cz ?l2) 0)
       )
     :effect
       (and
         (assign (coordx ?p1) (cx ?l2))
         (assign (coordy ?p1) (cy ?l2))
       )
    )
    (:action pegar1
     :parameters (?p1 - ponteRolante ?prod - produto1 ?l1 -
lugar ?l2 - lugar)
```

```
      :precondition
        (and
          (not (ocupada ?p1))
          (estaEm1 ?prod ?l1)
          (ocupado ?l1)
          (not (ocupado ?l2))
          (= (coordx ?p1) (cx ?l1))
          (= (coordy ?p1) (cy ?l1))
          (= (cx ?l1) (- (cx ?l2) 0))
          (= (cy ?l1) (- (cy ?l2) 0))
          (= (cz ?l1) (- (cz ?l2) 1))
        )
      :effect
        (and
          (estaNa1 ?prod ?p1)
          (not (estaEm1 ?prod ?l1))
          (ocupada ?p1)
          (not (ocupado ?l1))
          (not (disponivel ?l2))
          (assign (crdx1 ?prod) (- 0 1))
          (assign (crdy1 ?prod) (- 0 1))
          (assign (crdz1 ?prod) (- 0 1))
        )
    )
    (:action soltar1
     :parameters (?p1 - ponteRolante ?prod - produto1 ?l1 -
lugar ?l2 - lugar)
      :precondition
        (and
          (ocupada ?p1)
          (estaNa1 ?prod ?p1)
          (not (ocupado ?l1))
          (not (ocupado ?l2))
          (disponivel ?l1)
          (= (coordx ?p1) (cx ?l1))
          (= (coordy ?p1) (cy ?l1))
          (= (cx ?l1) (- (cx ?l2) 0))
          (= (cy ?l1) (- (cy ?l2) 0))
          (= (cz ?l1) (- (cz ?l2) 1))
        )
      :effect
        (and
          (not (estaNa1 ?prod ?p1))
          (estaEm1 ?prod ?l1)
          (not (ocupada ?p1))
          (ocupado ?l1)
          (disponivel ?l2)
          (assign (crdx1 ?prod) (cx ?l1))
          (assign (crdy1 ?prod) (cy ?l1))
          (assign (crdz1 ?prod) (cz ?l1))
        )
    )
)
```

dominio1.pddl modelo simplificado

```
(define (domain New_Project_1)
    (:requirements :strips)
```

```
    (:types
      lugar - object
      produto1 - object
    )
    (:predicates
      (estaEm1 ?pro - produto1 ?lug - lugar)
      (plivre1 ?prod - produto1)
      (atras ?lug - lugar ?lug - lugar)
      (esquerda ?lug - lugar ?lug - lugar)
      (acima ?lug - lugar ?lug - lugar)
      (livre ?lug - lugar)
      (piso ?lug - lugar)
    )
    (:action moverPiso1
     :parameters (?prod - produto1 ?l1 - lugar)
     :precondition
       (and
         (plivre1 ?prod)
         (livre ?l1)
         (piso ?l1)
       )
     :effect
       (and
         (estaEm1 ?prod ?l1)
         (not (plivre1 ?prod))
         (not (livre ?l1))
       )
    )
    (:action empilhar1
     :parameters (?prod - produto1 ?l1 - lugar ?l2 - lugar)
     :precondition
       (and
         (plivre1 ?prod)
         (not (livre ?l1))
         (acima ?l2 ?l1)
       )
     :effect
       (and
         (estaEm1 ?prod ?l2)
         (not (plivre1 ?prod))
         (not (livre ?l2))
       )
    )
)
```

problema1.pddl modelo completo

```
(define (problem Planning_Problem)
    (:domain New_Project_1)
    (:objects
      p1 - ponteRolante
      l1 - lugar
      l2 - lugar
      l3 - lugar
      l4 - lugar
```

```
      l5 - lugar
      l6 - lugar
      l7 - lugar
      l8 - lugar
      l9 - lugar
      l10 - lugar
      l11 - lugar
      l12 - lugar
      prod1 - produto1
      l13 - lugar
      l14 - lugar
      prod2 - produto1
      l15 - lugar
      l16 - lugar
      prod3 - produto1
      l17 - lugar
      l18 - lugar
      prod4 - produto1
      l19 - lugar
      l20 - lugar
    )
    (:init
      (= (coordx p1) -10)
      (= (coordy p1) -10)
      (disponivel l1)
      (= (cx l1) 0)
      (= (cy l1) 0)
      (= (cz l1) 0)
      (disponivel l2)
      (= (cx l2) 1)
      (= (cy l2) 0)
      (= (cz l2) 0)
      (disponivel l3)
      (= (cx l3) 0)
      (= (cy l3) 1)
      (= (cz l3) 0)
      (disponivel l4)
      (= (cx l4) 1)
      (= (cy l4) 1)
      (= (cz l4) 0)
      (= (cx l5) 0)
      (= (cy l5) 0)
      (= (cz l5) 1)
      (= (cx l6) 1)
      (= (cy l6) 0)
      (= (cz l6) 1)
      (= (cx l7) 0)
      (= (cy l7) 1)
      (= (cz l7) 1)
      (= (cx l8) 1)
      (= (cy l8) 1)
      (= (cz l8) 1)
      (= (cx l9) 0)
      (= (cy l9) 0)
      (= (cz l9) 2)
      (= (cx l10) 1)
```

```
        (= (cy l10) 0)
        (= (cz l10) 2)
        (= (cx l11) 0)
        (= (cy l11) 1)
        (= (cz l11) 2)
        (= (cx l12) 1)
        (= (cy l12) 1)
        (= (cz l12) 2)
        (estaEm1 prod1 l13)
        (= (crdx1 prod1) -10)
        (= (crdy1 prod1) -10)
        (= (crdz1 prod1) 0)
        (ocupado l13)
        (disponivel l13)
        (= (cx l13) -10)
        (= (cy l13) -10)
        (= (cz l13) 0)
        (disponivel l14)
        (= (cx l14) -10)
        (= (cy l14) -10)
        (= (cz l14) 1)
        (estaEm1 prod2 l15)
        (= (crdx1 prod2) -10)
        (= (crdy1 prod2) -10)
        (= (crdz1 prod2) 0)
        (ocupado l15)
        (disponivel l15)
        (= (cx l15) -10)
        (= (cy l15) -10)
        (= (cz l15) 0)
        (disponivel l16)
        (= (cx l16) -10)
        (= (cy l16) -10)
        (= (cz l16) 1)
        (estaEm1 prod3 l17)
        (= (crdx1 prod3) -10)
        (= (crdy1 prod3) -10)
        (= (crdz1 prod3) 0)
        (ocupado l17)
        (disponivel l17)
        (= (cx l17) -10)
        (= (cy l17) -10)
        (= (cz l17) 0)
        (disponivel l18)
        (= (cx l18) -10)
        (= (cy l18) -10)
        (= (cz l18) 1)
        (estaEm1 prod4 l19)
        (= (crdx1 prod4) -10)
        (= (crdy1 prod4) -10)
        (= (crdz1 prod4) 0)
        (ocupado l19)
        (disponivel l19)
        (= (cx l19) -10)
        (= (cy l19) -10)
        (= (cz l19) 0)
```

```
      (disponivel l20)
      (= (cx l20) -10)
      (= (cy l20) -10)
      (= (cz l20) 1)
    )
    (:goal
      (and
        (not (ocupada p1))
        (= (crdx1 prod1) 1)
        (= (crdy1 prod1) 0)
        (= (crdz1 prod1) 0)
        (= (crdx1 prod2) 1)
        (= (crdy1 prod2) 1)
        (= (crdz1 prod2) 0)
        (= (crdx1 prod3) 1)
        (= (crdy1 prod3) 0)
        (= (crdz1 prod3) 1)
        (= (crdx1 prod4) 1)
        (= (crdy1 prod4) 1)
        (= (crdz1 prod4) 1)
      )
    )
)
```

problema1.pddl modelo simplificado

```
(define (problem Planning_Problem)
    (:domain New_Project_1)
    (:objects
      l1 - lugar
      l2 - lugar
      l3 - lugar
      l4 - lugar
      l5 - lugar
      l6 - lugar
      l7 - lugar
      l8 - lugar
      prod1 - produto1
      prod2 - produto1
      prod3 - produto1
      prod4 - produto1
    )
    (:init
      (livre l1)
      (piso l1)
      (livre l2)
      (piso l2)
      (livre l3)
      (piso l3)
      (livre l4)
      (piso l4)
      (livre l5)
      (livre l6)
      (livre l7)
      (livre l8)
      (atras l2 l1)
      (atras l4 l3)
      (atras l6 l5)
```

```
      (atras l8 l7)
      (esquerda l3 l1)
      (esquerda l4 l2)
      (esquerda l7 l5)
      (esquerda l8 l6)
      (acima l5 l1)
      (acima l6 l2)
      (acima l7 l3)
      (acima l8 l4)
      (plivre1 prod1)
      (plivre1 prod2)
      (plivre1 prod3)
      (plivre1 prod4)
    )
    (:goal
      (and
        (estaEm1 prod1 l2)
        (estaEm1 prod2 l4)
        (estaEm1 prod3 l6)
        (estaEm1 prod4 l8)
      )
    )
)
```

Problema 2

dominio2.pddl modelo completo

```
(define (domain New_Project_1)
    (:requirements :typing :fluents :negative-preconditions
:equality)
    (:types
      ponteRolante – object
      lugar - object
      produto1 - object
    )
    (:predicates
      (estaEm1 ?pro - produto1 ?lug - lugar)
      (estaNa1 ?pro - produto1 ?pon - ponteRolante)
      (ocupada ?pon - ponteRolante)
      (disponivel ?lug - lugar)
      (ocupado ?lug - lugar)
    )
    (:functions
      (coordx ?pon - ponteRolante)
      (coordy ?pon - ponteRolante)
      (crdx1 ?pro - produto1)
      (crdy1 ?pro - produto1)
      (crdz1 ?pro - produto1)
      (cx ?lug - lugar)
      (cy ?lug - lugar)
      (cz ?lug - lugar)
    )
    (:action mover
     :parameters (?p1 - ponteRolante ?l1 - lugar ?l2 - lugar)
```

```
     :precondition
       (and
         (= (coordx ?p1) (cx ?l1))
         (= (coordy ?p1) (cy ?l1))
         (not (= ?l1 ?l2))
         (= (cz ?l1) 0)
         (= (cz ?l2) 0)
       )
     :effect
       (and
         (assign (coordx ?p1) (cx ?l2))
         (assign (coordy ?p1) (cy ?l2))
       )
    )
    (:action pegar1
     :parameters (?p1 - ponteRolante ?prod - produto1 ?l1 -
lugar ?l2 - lugar)
     :precondition
       (and
         (not (ocupada ?p1))
         (estaEm1 ?prod ?l1)
         (ocupado ?l1)
         (not (ocupado ?l2))
         (= (coordx ?p1) (cx ?l1))
         (= (coordy ?p1) (cy ?l1))
         (= (cx ?l1) (- (cx ?l2) 0))
         (= (cy ?l1) (- (cy ?l2) 0))
         (= (cz ?l1) (- (cz ?l2) 1))
       )
     :effect
       (and
         (estaNa1 ?prod ?p1)
         (not (estaEm1 ?prod ?l1))
         (ocupada ?p1)
         (not (ocupado ?l1))
         (not (disponivel ?l2))
         (assign (crdx1 ?prod) (- 0 1))
         (assign (crdy1 ?prod) (- 0 1))
         (assign (crdz1 ?prod) (- 0 1))
       )
    )
    (:action soltar1
     :parameters (?p1 - ponteRolante ?prod - produto1 ?l1 -
lugar ?l2 - lugar)
     :precondition
       (and
         (ocupada ?p1)
         (estaNa1 ?prod ?p1)
         (not (ocupado ?l1))
         (not (ocupado ?l2))
         (disponivel ?l1)
         (= (coordx ?p1) (cx ?l1))
         (= (coordy ?p1) (cy ?l1))
         (= (cx ?l1) (- (cx ?l2) 0))
         (= (cy ?l1) (- (cy ?l2) 0))
         (= (cz ?l1) (- (cz ?l2) 1))
```

```
        )
      :effect
        (and
          (not (estaNa1 ?prod ?p1))
          (estaEm1 ?prod ?l1)
          (not (ocupada ?p1))
          (ocupado ?l1)
          (disponivel ?l2)
          (assign (crdx1 ?prod) (cx ?l1))
          (assign (crdy1 ?prod) (cy ?l1))
          (assign (crdz1 ?prod) (cz ?l1))
)
    )
)
```

dominio2.pddl modelo simplificado

```
(define (domain New_Project_1)
    (:requirements :strips)
    (:types
      lugar - object
      produto1 - object
    )
    (:predicates
      (estaEm1 ?pro - produto1 ?lug - lugar)
      (plivre1 ?prod - produto1)
      (atras ?lug - lugar ?lug - lugar)
      (esquerda ?lug - lugar ?lug - lugar)
      (acima ?lug - lugar ?lug - lugar)
      (livre ?lug - lugar)
      (piso ?lug - lugar)
    )
    (:action moverPiso1
     :parameters (?prod - produto1 ?l1 - lugar)
     :precondition
       (and
         (plivre1 ?prod)
         (livre ?l1)
         (piso ?l1)
       )
     :effect
       (and
         (estaEm1 ?prod ?l1)
         (not (plivre1 ?prod))
         (not (livre ?l1))
       )
    )
    (:action empilhar1
     :parameters (?prod - produto1 ?l1 - lugar ?l2 - lugar)
     :precondition
       (and
         (plivre1 ?prod)
         (not (livre ?l1))
         (acima ?l2 ?l1)
       )
     :effect
       (and
         (estaEm1 ?prod ?l2)
```

```
          (not (plivre1 ?prod))
          (not (livre ?l2))
        )
    )
)
```

problema2.pddl modelo completo

```
(define (problem Planning_Problem)
    (:domain New_Project_1)
(:objects
      p1 - ponteRolante
      l1 - lugar
      l2 - lugar
      l3 - lugar
      l4 - lugar
      l5 - lugar
      l6 - lugar
      l7 - lugar
      l8 - lugar
      l9 - lugar
      l10 - lugar
      l11 - lugar
      l12 - lugar
      prod1 - produto1
      l13 - lugar
      l14 - lugar
      prod2 - produto1
      l15 - lugar
      l16 - lugar
      prod3 - produto1
      l17 - lugar
      l18 - lugar
      prod4 - produto1
      l19 - lugar
      l20 - lugar
      prod5 - produto1
      l21 - lugar
      l22 - lugar
      prod6 - produto1
      l23 - lugar
      l24 - lugar
      prod7 - produto1
      l25 - lugar
      l26 - lugar
      prod8 - produto1
      l27 - lugar
      l28 - lugar
    )
    (:init
      (= (coordx p1) -10)
      (= (coordy p1) -10)
      (disponivel l1)
      (= (cx l1) 0)
      (= (cy l1) 0)
```

```
(= (cz l1) 0)
(disponivel l2)
(= (cx l2) 1)
(= (cy l2) 0)
(= (cz l2) 0)
(disponivel l3)
(= (cx l3) 0)
(= (cy l3) 1)
(= (cz l3) 0)
(disponivel l4)
(= (cx l4) 1)
(= (cy l4) 1)
(= (cz l4) 0)
(= (cx l5) 0)
(= (cy l5) 0)
(= (cz l5) 1)
(= (cx l6) 1)
(= (cy l6) 0)
(= (cz l6) 1)
(= (cx l7) 0)
(= (cy l7) 1)
(= (cz l7) 1)
(= (cx l8) 1)
(= (cy l8) 1)
(= (cz l8) 1)
(= (cx l9) 0)
(= (cy l9) 0)
(= (cz l9) 2)
(= (cx l10) 1)
(= (cy l10) 0)
(= (cz l10) 2)
(= (cx l11) 0)
(= (cy l11) 1)
(= (cz l11) 2)
(= (cx l12) 1)
(= (cy l12) 1)
(= (cz l12) 2)
(estaEm1 prod1 l13)
(= (crdx1 prod1) -10)
(= (crdy1 prod1) -10)
(= (crdz1 prod1) 0)
(ocupado l13)
(disponivel l13)
(= (cx l13) -10)
(= (cy l13) -10)
(= (cz l13) 0)
(disponivel l14)
(= (cx l14) -10)
(= (cy l14) -10)
(= (cz l14) 1)
(estaEm1 prod2 l15)
(= (crdx1 prod2) -10)
(= (crdy1 prod2) -10)
(= (crdz1 prod2) 0)
(ocupado l15)
(disponivel l15)
```

```
        (= (cx l15) -10)
        (= (cy l15) -10)
        (= (cz l15) 0)
        (disponivel l16)
        (= (cx l16) -10)
        (= (cy l16) -10)
        (= (cz l16) 1)
        (estaEm1 prod3 l17)
        (= (crdx1 prod3) -10)
        (= (crdy1 prod3) -10)
        (= (crdz1 prod3) 0)
        (ocupado l17)
        (disponivel l17)
        (= (cx l17) -10)
        (= (cy l17) -10)
        (= (cz l17) 0)
        (disponivel l18)
        (= (cx l18) -10)
        (= (cy l18) -10)
        (= (cz l18) 1)
        (estaEm1 prod4 l19)
        (= (crdx1 prod4) -10)
        (= (crdy1 prod4) -10)
        (= (crdz1 prod4) 0)
        (ocupado l19)
        (disponivel l19)
        (= (cx l19) -10)
        (= (cy l19) -10)
        (= (cz l19) 0)
        (disponivel l20)
        (= (cx l20) -10)
        (= (cy l20) -10)
        (= (cz l20) 1)
        (estaEm1 prod5 l21)
        (= (crdx1 prod5) -10)
        (= (crdy1 prod5) -10)
        (= (crdz1 prod5) 0)
        (ocupado l21)
        (disponivel l21)
        (= (cx l21) -10)
        (= (cy l21) -10)
        (= (cz l21) 0)
        (disponivel l22)
        (= (cx l22) -10)
        (= (cy l22) -10)
        (= (cz l22) 1)
        (estaEm1 prod6 l23)
        (= (crdx1 prod6) -10)
        (= (crdy1 prod6) -10)
        (= (crdz1 prod6) 0)
        (ocupado l23)
        (disponivel l23)
        (= (cx l23) -10)
        (= (cy l23) -10)
        (= (cz l23) 0)
        (disponivel l24)
```

```
      (= (cx l24) -10)
      (= (cy l24) -10)
      (= (cz l24) 1)
      (estaEm1 prod7 l25)
      (= (crdx1 prod7) -10)
      (= (crdy1 prod7) -10)
      (= (crdz1 prod7) 0)
      (ocupado l25)
      (disponivel l25)
      (= (cx l25) -10)
      (= (cy l25) -10)
      (= (cz l25) 0)
      (disponivel l26)
      (= (cx l26) -10)
      (= (cy l26) -10)
      (= (cz l26) 1)
      (estaEm1 prod8 l27)
      (= (crdx1 prod8) -10)
      (= (crdy1 prod8) -10)
      (= (crdz1 prod8) 0)
      (ocupado l27)
      (disponivel l27)
      (= (cx l27) -10)
      (= (cy l27) -10)
      (= (cz l27) 0)
      (disponivel l28)
      (= (cx l28) -10)
      (= (cy l28) -10)
      (= (cz l28) 1)
    )
    (:goal
      (and
        (not (ocupada p1))
        (= (crdx1 prod1) 0)
        (= (crdy1 prod1) 0)
        (= (crdz1 prod1) 0)
        (= (crdx1 prod2) 1)
        (= (crdy1 prod2) 0)
        (= (crdz1 prod2) 0)
        (= (crdx1 prod3) 0)
        (= (crdy1 prod3) 1)
        (= (crdz1 prod3) 0)
        (= (crdx1 prod4) 1)
        (= (crdy1 prod4) 1)
        (= (crdz1 prod4) 0)
        (= (crdx1 prod5) 0)
        (= (crdy1 prod5) 0)
        (= (crdz1 prod5) 1)
        (= (crdx1 prod6) 1)
        (= (crdy1 prod6) 0)
        (= (crdz1 prod6) 1)
        (= (crdx1 prod7) 0)
        (= (crdy1 prod7) 1)
        (= (crdz1 prod7) 1)
        (= (crdx1 prod8) 1)
        (= (crdy1 prod8) 1)
```

```
        (= (crdz1 prod8) 1)
      )
    )
)
```

problema2.pddl modelo simplificado

```
(define (problem Planning_Problem)
    (:domain New_Project_1)
    (:objects
      l1 - lugar
      l2 - lugar
      l3 - lugar
      l4 - lugar
      l5 - lugar
      l6 - lugar
      l7 - lugar
      l8 - lugar
      prod1 - produto1
      prod2 - produto1
      prod3 - produto1
      prod4 - produto1
      prod5 - produto1
      prod6 - produto1
      prod7 - produto1
      prod8 - produto1
    )
    (:init
      (livre l1)
      (piso l1)
      (livre l2)
      (piso l2)
      (livre l3)
      (piso l3)
      (livre l4)
      (piso l4)
      (livre l5)
      (livre l6)
      (livre l7)
      (livre l8)
      (atras l2 l1)
      (atras l4 l3)
      (atras l6 l5)
      (atras l8 l7)
      (esquerda l3 l1)
      (esquerda l4 l2)
      (esquerda l7 l5)
      (esquerda l8 l6)
      (acima l5 l1)
      (acima l6 l2)
      (acima l7 l3)
      (acima l8 l4)
      (plivre1 prod1)
      (plivre1 prod2)
      (plivre1 prod3)
      (plivre1 prod4)
      (plivre1 prod5)
      (plivre1 prod6)
```

```
      (plivre1 prod7)
      (plivre1 prod8)
    )
    (:goal
      (and
        (estaEm1 prod1 l1)
        (estaEm1 prod2 l2)
        (estaEm1 prod3 l3)
        (estaEm1 prod4 l4)
        (estaEm1 prod5 l5)
        (estaEm1 prod6 l6)
        (estaEm1 prod7 l7)
        (estaEm1 prod8 l8)
      )
    )
)
```

Problema 3

dominio3.pddl modelo completo

```
(define (domain New_Project_1)
    (:requirements :typing :fluents :negative-preconditions
:equality)
    (:types
      ponteRolante – object
      lugar - object
      produto1 - object
      produto2 - object
      produto3 - object
    )
    (:predicates
      (estaEm1 ?pro - produto1 ?lug - lugar)
      (estaNa1 ?pro - produto1 ?pon - ponteRolante)
      (estaEm2 ?pro - produto2 ?lug - lugar)
      (estaNa2 ?pro - produto2 ?pon - ponteRolante)
      (estaEm3 ?pro - produto3 ?lug - lugar)
      (estaNa3 ?pro - produto3 ?pon - ponteRolante)
      (ocupada ?pon - ponteRolante)
      (disponivel ?lug - lugar)
      (ocupado ?lug - lugar)
    )
    (:functions
      (coordx ?pon - ponteRolante)
      (coordy ?pon - ponteRolante)
      (crdx1 ?pro - produto1)
      (crdy1 ?pro - produto1)
      (crdz1 ?pro - produto1)
      (crdx2 ?pro - produto2)
      (crdy2 ?pro - produto2)
      (crdz2 ?pro - produto2)
      (crdx3 ?pro - produto3)
      (crdy3 ?pro - produto3)
      (crdz3 ?pro - produto3)
      (cx ?lug - lugar)
```

```
      (cy ?lug - lugar)
      (cz ?lug - lugar)
    )
    (:action mover
     :parameters (?p1 - ponteRolante ?l1 - lugar ?l2 - lugar)
     :precondition
       (and
         (= (coordx ?p1) (cx ?l1))
         (= (coordy ?p1) (cy ?l1))
         (not (= ?l1 ?l2))
         (= (cz ?l1) 0)
         (= (cz ?l2) 0)
       )
     :effect
       (and
         (assign (coordx ?p1) (cx ?l2))
         (assign (coordy ?p1) (cy ?l2))
)
    )
    (:action pegar1
     :parameters (?p1 - ponteRolante ?prod - produto1 ?l1 -
lugar ?l2 - lugar ?l3 - lugar ?l4 - lugar)
     :precondition
       (and
         (not (ocupada ?p1))
         (estaEm1 ?prod ?l1)
         (ocupado ?l1)
         (ocupado ?l2)
         (not (ocupado ?l3))
         (not (ocupado ?l4))
         (= (coordx ?p1) (cx ?l1))
         (= (coordy ?p1) (cy ?l1))
         (= (cx ?l1) (- (cx ?l2) 0))
         (= (cy ?l1) (- (cy ?l2) 1))
         (= (cz ?l1) (- (cz ?l2) 0))
         (= (cx ?l1) (- (cx ?l3) 0))
         (= (cy ?l1) (- (cy ?l3) 0))
         (= (cz ?l1) (- (cz ?l3) 1))
         (= (cx ?l1) (- (cx ?l4) 0))
         (= (cy ?l1) (- (cy ?l4) 1))
         (= (cz ?l1) (- (cz ?l4) 1))
       )
     :effect
       (and
         (estaNa1 ?prod ?p1)
         (not (estaEm1 ?prod ?l1))
         (ocupada ?p1)
         (not (ocupado ?l1))
         (not (ocupado ?l2))
         (not (disponivel ?l3))
         (not (disponivel ?l4))
         (assign (crdx1 ?prod) (- 0 1))
         (assign (crdy1 ?prod) (- 0 1))
         (assign (crdz1 ?prod) (- 0 1))
       )
    )
```

```
    (:action pegar2
     :parameters (?p1 - ponteRolante ?prod - produto2 ?l1 -
lugar ?l2 - lugar ?l3 - lugar ?l4 - lugar)
     :precondition
       (and
         (not (ocupada ?p1))
         (estaEm2 ?prod ?l1)
         (ocupado ?l1)
         (ocupado ?l2)
         (not (ocupado ?l3))
         (not (ocupado ?l4))
         (= (coordx ?p1) (cx ?l1))
         (= (coordy ?p1) (cy ?l1))
         (= (cx ?l1) (- (cx ?l2) 1))
         (= (cy ?l1) (- (cy ?l2) 0))
         (= (cz ?l1) (- (cz ?l2) 0))
         (= (cx ?l1) (- (cx ?l3) 0))
         (= (cy ?l1) (- (cy ?l3) 0))
         (= (cz ?l1) (- (cz ?l3) 1))
         (= (cx ?l1) (- (cx ?l4) 1))
         (= (cy ?l1) (- (cy ?l4) 0))
         (= (cz ?l1) (- (cz ?l4) 1))
       )
     :effect
       (and
         (estaNa2 ?prod ?p1)
         (not (estaEm2 ?prod ?l1))
         (ocupada ?p1)
         (not (ocupado ?l1))
         (not (ocupado ?l2))
         (not (disponivel ?l3))
         (not (disponivel ?l4))
         (assign (crdx2 ?prod) (- 0 1))
         (assign (crdy2 ?prod) (- 0 1))
         (assign (crdz2 ?prod) (- 0 1))
       )
    )
    (:action pegar3
     :parameters (?p1 - ponteRolante ?prod - produto3 ?l1 -
lugar ?l2 - lugar ?l3 - lugar)
     :precondition
       (and
         (not (ocupada ?p1))
         (estaEm3 ?prod ?l1)
         (ocupado ?l1)
         (ocupado ?l2)
         (not (ocupado ?l3))
         (= (coordx ?p1) (cx ?l1))
         (= (coordy ?p1) (cy ?l1))
         (= (cx ?l1) (- (cx ?l2) 0))
         (= (cy ?l1) (- (cy ?l2) 0))
         (= (cz ?l1) (- (cz ?l2) 1))
         (= (cx ?l1) (- (cx ?l3) 0))
         (= (cy ?l1) (- (cy ?l3) 0))
         (= (cz ?l1) (- (cz ?l3) 2))
       )
```

```
     :effect
       (and
         (estaNa3 ?prod ?p1)
         (not (estaEm3 ?prod ?l1))
         (ocupada ?p1)
         (not (ocupado ?l1))
         (not (ocupado ?l2))
         (not (disponivel ?l2))
         (not (disponivel ?l3))
         (assign (crdx3 ?prod) (- 0 1))
         (assign (crdy3 ?prod) (- 0 1))
         (assign (crdz3 ?prod) (- 0 1))
       )
    )
    (:action soltar1
     :parameters (?p1 - ponteRolante ?prod - produto1 ?l1 -
lugar ?l2 - lugar ?l3 - lugar ?l4 - lugar)
     :precondition
       (and
         (ocupada ?p1)
         (estaNa1 ?prod ?p1)
         (not (ocupado ?l1))
         (not (ocupado ?l2))
         (not (ocupado ?l3))
         (not (ocupado ?l4))
         (disponivel ?l1)
         (disponivel ?l2)
         (= (coordx ?p1) (cx ?l1))
         (= (coordy ?p1) (cy ?l1))
         (= (cx ?l1) (- (cx ?l2) 0))
         (= (cy ?l1) (- (cy ?l2) 1))
         (= (cz ?l1) (- (cz ?l2) 0))
         (= (cx ?l1) (- (cx ?l3) 0))
         (= (cy ?l1) (- (cy ?l3) 0))
         (= (cz ?l1) (- (cz ?l3) 1))
         (= (cx ?l1) (- (cx ?l4) 0))
         (= (cy ?l1) (- (cy ?l4) 1))
         (= (cz ?l1) (- (cz ?l4) 1))
       )
     :effect
       (and
         (not (estaNa1 ?prod ?p1))
         (estaEm1 ?prod ?l1)
         (not (ocupada ?p1))
         (ocupado ?l1)
         (ocupado ?l2)
         (disponivel ?l3)
         (disponivel ?l4)
         (assign (crdx1 ?prod) (cx ?l1))
         (assign (crdy1 ?prod) (cy ?l1))
         (assign (crdz1 ?prod) (cz ?l1))
)
    )
    (:action soltar2
     :parameters (?p1 - ponteRolante ?prod - produto2 ?l1 -
lugar ?l2 - lugar ?l3 - lugar ?l4 - lugar)
```

```
      :precondition
        (and
          (ocupada ?p1)
          (estaNa2 ?prod ?p1)
          (not (ocupado ?l1))
          (not (ocupado ?l2))
          (not (ocupado ?l3))
          (not (ocupado ?l4))
          (disponivel ?l1)
          (disponivel ?l2)
          (= (coordx ?p1) (cx ?l1))
          (= (coordy ?p1) (cy ?l1))
          (= (cx ?l1) (- (cx ?l2) 1))
          (= (cy ?l1) (- (cy ?l2) 0))
          (= (cz ?l1) (- (cz ?l2) 0))
          (= (cx ?l1) (- (cx ?l3) 0))
          (= (cy ?l1) (- (cy ?l3) 0))
          (= (cz ?l1) (- (cz ?l3) 1))
          (= (cx ?l1) (- (cx ?l4) 1))
          (= (cy ?l1) (- (cy ?l4) 0))
          (= (cz ?l1) (- (cz ?l4) 1))
        )
      :effect
        (and
          (not (estaNa2 ?prod ?p1))
          (estaEm2 ?prod ?l1)
          (not (ocupada ?p1))
          (ocupado ?l1)
          (ocupado ?l2)
          (disponivel ?l3)
          (disponivel ?l4)
          (assign (crdx2 ?prod) (cx ?l1))
          (assign (crdy2 ?prod) (cy ?l1))
          (assign (crdz2 ?prod) (cz ?l1))
)
    )
    (:action soltar3
     :parameters (?p1 - ponteRolante ?prod - produto3 ?l1 -
lugar ?l2 - lugar ?l3 - lugar)
      :precondition
        (and
          (ocupada ?p1)
          (estaNa3 ?prod ?p1)
          (not (ocupado ?l1))
          (not (ocupado ?l2))
          (not (ocupado ?l3))
          (disponivel ?l1)
          (= (coordx ?p1) (cx ?l1))
          (= (coordy ?p1) (cy ?l1))
          (= (cx ?l1) (- (cx ?l2) 0))
          (= (cy ?l1) (- (cy ?l2) 0))
          (= (cz ?l1) (- (cz ?l2) 1))
          (= (cx ?l1) (- (cx ?l3) 0))
          (= (cy ?l1) (- (cy ?l3) 0))
          (= (cz ?l1) (- (cz ?l3) 2))
        )
```

```
     :effect
       (and
         (not (estaNa3 ?prod ?p1))
         (estaEm3 ?prod ?l1)
         (not (ocupada ?p1))
         (ocupado ?l1)
         (ocupado ?l2)
         (disponivel ?l2)
         (disponivel ?l3)
         (assign (crdx3 ?prod) (cx ?l1))
         (assign (crdy3 ?prod) (cy ?l1))
         (assign (crdz3 ?prod) (cz ?l1))
       )
    )
)
```

dominio3.pddl modelo simplificado

```
(define (domain New_Project_1)
    (:requirements :strips)
    (:types
      lugar - object
      produto1 - object
      produto2 - object
      produto3 - object
    )
    (:predicates
      (estaEm1 ?pro - produto1 ?lug - lugar)
      (plivre1 ?prod - produto1)
      (estaEm2 ?pro - produto2 ?lug - lugar)
      (plivre2 ?prod - produto2)
      (estaEm3 ?pro - produto3 ?lug - lugar)
      (plivre3 ?prod - produto3)
      (atras ?lug - lugar ?lug - lugar)
      (esquerda ?lug - lugar ?lug - lugar)
      (acima ?lug - lugar ?lug - lugar)
      (livre ?lug - lugar)
      (piso ?lug - lugar)
    )
    (:action moverPiso1
     :parameters (?prod - produto1 ?l1 - lugar ?l2 - lugar)
     :precondition
       (and
         (plivre1 ?prod)
         (livre ?l1)
         (piso ?l1)
         (livre ?l2)
         (piso ?l2)
         (atras ?l2 ?l1)
       )
     :effect
       (and
         (estaEm1 ?prod ?l1)
         (not (plivre1 ?prod))
         (not (livre ?l1))
         (not (livre ?l2))
       )
    )
```

```
    (:action moverPiso2
     :parameters (?prod - produto2 ?l1 - lugar ?l2 - lugar)
     :precondition
       (and
         (plivre2 ?prod)
         (livre ?l1)
         (piso ?l1)
         (livre ?l2)
         (piso ?l2)
         (esquerda ?l2 ?l1)
       )
     :effect
       (and
         (estaEm2 ?prod ?l1)
         (not (plivre2 ?prod))
         (not (livre ?l1))
         (not (livre ?l2))
       )
    )
    (:action moverPiso3
     :parameters (?prod - produto3 ?l1 - lugar ?l2 - lugar)
     :precondition
       (and
         (plivre3 ?prod)
         (livre ?l1)
         (piso ?l1)
         (acima ?l2 ?l1)
       )
     :effect
       (and
         (estaEm3 ?prod ?l1)
         (not (plivre3 ?prod))
         (not (livre ?l1))
         (not (livre ?l2))
       )
    )
    (:action empilhar1
     :parameters (?prod - produto1 ?l1 - lugar ?l2 - lugar ?l3
- lugar ?l4 - lugar)
     :precondition
       (and
         (plivre1 ?prod)
         (not (livre ?l1))
         (not (livre ?l2))
         (atras ?l4 ?l3)
         (acima ?l3 ?l1)
         (acima ?l4 ?l2)
       )
     :effect
       (and
         (estaEm1 ?prod ?l3)
         (not (plivre1 ?prod))
         (not (livre ?l3))
         (not (livre ?l4))
       )
    )
```

```
    (:action empilhar2
     :parameters (?prod - produto2 ?l1 - lugar ?l2 - lugar ?l3
- lugar ?l4 - lugar)
     :precondition
       (and
         (plivre2 ?prod)
         (not (livre ?l1))
         (not (livre ?l2))
         (esquerda ?l4 ?l3)
         (acima ?l3 ?l1)
         (acima ?l4 ?l2)
       )
     :effect
       (and
         (estaEm2 ?prod ?l3)
         (not (plivre2 ?prod))
         (not (livre ?l3))
         (not (livre ?l4))
       )
    )
    (:action empilhar3
     :parameters (?prod - produto3 ?l1 - lugar ?l2 - lugar ?l3
- lugar)
     :precondition
       (and
         (plivre3 ?prod)
         (not (livre ?l1))
         (acima ?l3 ?l2)
         (acima ?l2 ?l1)
       )
     :effect
       (and
         (estaEm3 ?prod ?l2)
         (not (plivre3 ?prod))
         (not (livre ?l2))
         (not (livre ?l3))
       )
    )
)
```

problema3.pddl modelo completo

```
(define (problem Planning_Problem)
    (:domain New_Project_1)
(:objects
      p1 - ponteRolante
      l1 - lugar
      l2 - lugar
      l3 - lugar
      l4 - lugar
      l5 - lugar
      l6 - lugar
      l7 - lugar
      l8 - lugar
      l9 - lugar
```

```
      l10 - lugar
      l11 - lugar
      l12 - lugar
      prod1 - produto2
      l13 - lugar
      l14 - lugar
      l15 - lugar
      l16 - lugar
      prod2 - produto3
      l17 - lugar
      l18 - lugar
      l19 - lugar
    )
    (:init
      (= (coordx p1) -10)
      (= (coordy p1) -10)
      (disponivel l1)
      (= (cx l1) 0)
      (= (cy l1) 0)
      (= (cz l1) 0)
      (disponivel l2)
      (= (cx l2) 1)
      (= (cy l2) 0)
      (= (cz l2) 0)
      (disponivel l3)
      (= (cx l3) 0)
      (= (cy l3) 1)
      (= (cz l3) 0)
      (disponivel l4)
      (= (cx l4) 1)
      (= (cy l4) 1)
      (= (cz l4) 0)
      (= (cx l5) 0)
      (= (cy l5) 0)
      (= (cz l5) 1)
      (= (cx l6) 1)
      (= (cy l6) 0)
      (= (cz l6) 1)
      (= (cx l7) 0)
      (= (cy l7) 1)
      (= (cz l7) 1)
      (= (cx l8) 1)
      (= (cy l8) 1)
      (= (cz l8) 1)
      (= (cx l9) 0)
      (= (cy l9) 0)
      (= (cz l9) 2)
      (= (cx l10) 1)
      (= (cy l10) 0)
      (= (cz l10) 2)
      (= (cx l11) 0)
      (= (cy l11) 1)
      (= (cz l11) 2)
      (= (cx l12) 1)
      (= (cy l12) 1)
      (= (cz l12) 2)
```

```
      (estaEm2 prod1 l13)
      (= (crdx2 prod1) -10)
      (= (crdy2 prod1) -10)
      (= (crdz2 prod1) 0)
      (ocupado l13)
      (disponivel l13)
      (= (cx l13) -10)
      (= (cy l13) -10)
      (= (cz l13) 0)
      (ocupado l14)
      (disponivel l14)
      (= (cx l14) -9)
      (= (cy l14) -10)
      (= (cz l14) 0)
      (disponivel l15)
      (= (cx l15) -10)
      (= (cy l15) -10)
      (= (cz l15) 1)
      (disponivel l16)
      (= (cx l16) -9)
      (= (cy l16) -10)
      (= (cz l16) 1)
      (estaEm3 prod2 l17)
      (= (crdx3 prod2) -10)
      (= (crdy3 prod2) -10)
      (= (crdz3 prod2) 0)
      (ocupado l17)
      (disponivel l17)
      (= (cx l17) -10)
      (= (cy l17) -10)
      (= (cz l17) 0)
      (ocupado l18)
      (disponivel l18)
      (= (cx l18) -10)
      (= (cy l18) -10)
      (= (cz l18) 1)
      (disponivel l19)
      (= (cx l19) -10)
      (= (cy l19) -10)
      (= (cz l19) 2)
    )
    (:goal
      (and
        (not (ocupada p1))
        (= (crdx2 prod1) 0)
        (= (crdy2 prod1) 0)
        (= (crdz2 prod1) 0)
        (= (crdx3 prod2) 0)
        (= (crdy3 prod2) 1)
        (= (crdz3 prod2) 0)
      )
    )
)
```

problema3.pddl modelo simplificado

```
(define (problem Planning_Problem)
    (:domain New_Project_1)
```

```
    (:objects
      l1 - lugar
      l2 - lugar
      l3 - lugar
      l4 - lugar
      l5 - lugar
      l6 - lugar
      l7 - lugar
      l8 - lugar
      prod1 - produto1
      prod2 - produto3
    )
    (:init
      (livre l1)
      (piso l1)
      (livre l2)
      (piso l2)
      (livre l3)
      (piso l3)
      (livre l4)
      (piso l4)
      (livre l5)
      (livre l6)
      (livre l7)
      (livre l8)
      (atras l2 l1)
      (atras l4 l3)
      (atras l6 l5)
      (atras l8 l7)
      (esquerda l3 l1)
      (esquerda l4 l2)
      (esquerda l7 l5)
      (esquerda l8 l6)
      (acima l5 l1)
      (acima l6 l2)
      (acima l7 l3)
      (acima l8 l4)
      (plivre1 prod1)
      (plivre3 prod2)
    )
    (:goal
      (and
        (estaEm1 prod1 l1)
        (estaEm3 prod2 l3)
      )
    )
)
```

Problema 4

dominio4.pddl modelo completo

```
(define (domain New_Project_1)
    (:requirements :typing :fluents :negative-preconditions
:equality)
```

```
(:types
      ponteRolante - object
      lugar - object
      produto1 - object
      produto2 - object
      produto3 - object
    )
    (:predicates
      (estaEm1 ?pro - produto1 ?lug - lugar)
      (estaNa1 ?pro - produto1 ?pon - ponteRolante)
      (estaEm2 ?pro - produto2 ?lug - lugar)
      (estaNa2 ?pro - produto2 ?pon - ponteRolante)
      (estaEm3 ?pro - produto3 ?lug - lugar)
      (estaNa3 ?pro - produto3 ?pon - ponteRolante)
      (ocupada ?pon - ponteRolante)
      (disponivel ?lug - lugar)
      (ocupado ?lug - lugar)
    )
    (:functions
      (coordx ?pon - ponteRolante)
      (coordy ?pon - ponteRolante)
      (crdx1 ?pro - produto1)
      (crdy1 ?pro - produto1)
      (crdz1 ?pro - produto1)
      (crdx2 ?pro - produto2)
      (crdy2 ?pro - produto2)
      (crdz2 ?pro - produto2)
      (crdx3 ?pro - produto3)
      (crdy3 ?pro - produto3)
      (crdz3 ?pro - produto3)
      (cx ?lug - lugar)
      (cy ?lug - lugar)
      (cz ?lug - lugar)
    )
    (:action mover
     :parameters (?p1 - ponteRolante ?l1 - lugar ?l2 - lugar)
     :precondition
       (and
         (= (coordx ?p1) (cx ?l1))
         (= (coordy ?p1) (cy ?l1))
         (not (= ?l1 ?l2))
         (= (cz ?l1) 0)
         (= (cz ?l2) 0)
       )
     :effect
       (and
         (assign (coordx ?p1) (cx ?l2))
         (assign (coordy ?p1) (cy ?l2))
       )
    )
    (:action pegar1
     :parameters (?p1 - ponteRolante ?prod - produto1 ?l1 -
lugar ?l2 - lugar ?l3 - lugar ?l4 - lugar)
     :precondition
       (and
         (not (ocupada ?p1))
```

```
          (estaEm1 ?prod ?l1)
          (ocupado ?l1)
          (ocupado ?l2)
          (not (ocupado ?l3))
          (not (ocupado ?l4))
          (= (coordx ?p1) (cx ?l1))
          (= (coordy ?p1) (cy ?l1))
          (= (cx ?l1) (- (cx ?l2) 1))
          (= (cy ?l1) (- (cy ?l2) 0))
          (= (cz ?l1) (- (cz ?l2) 0))
          (= (cx ?l1) (- (cx ?l3) 0))
          (= (cy ?l1) (- (cy ?l3) 0))
          (= (cz ?l1) (- (cz ?l3) 1))
          (= (cx ?l1) (- (cx ?l4) 1))
          (= (cy ?l1) (- (cy ?l4) 0))
          (= (cz ?l1) (- (cz ?l4) 1))
        )
      :effect
        (and
          (estaNa1 ?prod ?p1)
          (not (estaEm1 ?prod ?l1))
          (ocupada ?p1)
          (not (ocupado ?l1))
          (not (ocupado ?l2))
          (not (disponivel ?l3))
          (not (disponivel ?l4))
          (assign (crdx1 ?prod) (- 0 1))
          (assign (crdy1 ?prod) (- 0 1))
          (assign (crdz1 ?prod) (- 0 1))
        )
    )
    (:action pegar2
     :parameters (?p1 - ponteRolante ?prod - produto2 ?l1 -
lugar ?l2 - lugar ?l3 - lugar ?l4 - lugar)
      :precondition
        (and
          (not (ocupada ?p1))
          (estaEm2 ?prod ?l1)
          (ocupado ?l1)
          (ocupado ?l2)
          (not (ocupado ?l3))
          (not (ocupado ?l4))
          (= (coordx ?p1) (cx ?l1))
          (= (coordy ?p1) (cy ?l1))
          (= (cx ?l1) (- (cx ?l2) 0))
          (= (cy ?l1) (- (cy ?l2) 1))
          (= (cz ?l1) (- (cz ?l2) 0))
          (= (cx ?l1) (- (cx ?l3) 0))
          (= (cy ?l1) (- (cy ?l3) 0))
          (= (cz ?l1) (- (cz ?l3) 1))
          (= (cx ?l1) (- (cx ?l4) 0))
          (= (cy ?l1) (- (cy ?l4) 1))
          (= (cz ?l1) (- (cz ?l4) 1))
        )
      :effect
        (and
```

```
          (estaNa2 ?prod ?p1)
          (not (estaEm2 ?prod ?l1))
          (ocupada ?p1)
          (not (ocupado ?l1))
          (not (ocupado ?l2))
          (not (disponivel ?l3))
          (not (disponivel ?l4))
          (assign (crdx2 ?prod) (- 0 1))
          (assign (crdy2 ?prod) (- 0 1))
          (assign (crdz2 ?prod) (- 0 1))
        )
    )
    (:action pegar3
     :parameters (?p1 - ponteRolante ?prod - produto3 ?l1 -
lugar ?l2 - lugar ?l3 - lugar)
     :precondition
       (and
         (not (ocupada ?p1))
         (estaEm3 ?prod ?l1)
         (ocupado ?l1)
         (ocupado ?l2)
         (not (ocupado ?l3))
         (= (coordx ?p1) (cx ?l1))
         (= (coordy ?p1) (cy ?l1))
         (= (cx ?l1) (- (cx ?l2) 0))
         (= (cy ?l1) (- (cy ?l2) 0))
         (= (cz ?l1) (- (cz ?l2) 1))
         (= (cx ?l1) (- (cx ?l3) 0))
         (= (cy ?l1) (- (cy ?l3) 0))
         (= (cz ?l1) (- (cz ?l3) 2))
       )
     :effect
       (and
         (estaNa3 ?prod ?p1)
         (not (estaEm3 ?prod ?l1))
         (ocupada ?p1)
         (not (ocupado ?l1))
         (not (ocupado ?l2))
         (not (disponivel ?l2))
         (not (disponivel ?l3))
         (assign (crdx3 ?prod) (- 0 1))
         (assign (crdy3 ?prod) (- 0 1))
         (assign (crdz3 ?prod) (- 0 1))
       )
    )
    (:action soltar1
     :parameters (?p1 - ponteRolante ?prod - produto1 ?l1 -
lugar ?l2 - lugar ?l3 - lugar ?l4 - lugar)
     :precondition
       (and
         (ocupada ?p1)
         (estaNa1 ?prod ?p1)
         (not (ocupado ?l1))
         (not (ocupado ?l2))
         (not (ocupado ?l3))
         (not (ocupado ?l4))
```

```
          (disponivel ?l1)
          (disponivel ?l2)
          (= (coordx ?p1) (cx ?l1))
          (= (coordy ?p1) (cy ?l1))
          (= (cx ?l1) (- (cx ?l2) 1))
          (= (cy ?l1) (- (cy ?l2) 0))
          (= (cz ?l1) (- (cz ?l2) 0))
          (= (cx ?l1) (- (cx ?l3) 0))
          (= (cy ?l1) (- (cy ?l3) 0))
          (= (cz ?l1) (- (cz ?l3) 1))
          (= (cx ?l1) (- (cx ?l4) 1))
          (= (cy ?l1) (- (cy ?l4) 0))
          (= (cz ?l1) (- (cz ?l4) 1))
        )
      :effect
        (and
          (not (estaNa1 ?prod ?p1))
          (estaEm1 ?prod ?l1)
          (not (ocupada ?p1))
          (ocupado ?l1)
          (ocupado ?l2)
          (disponivel ?l3)
          (disponivel ?l4)
          (assign (crdx1 ?prod) (cx ?l1))
          (assign (crdy1 ?prod) (cy ?l1))
          (assign (crdz1 ?prod) (cz ?l1))
        )
    )
    (:action soltar2
     :parameters (?p1 - ponteRolante ?prod - produto2 ?l1 -
lugar ?l2 - lugar ?l3 - lugar ?l4 - lugar)
      :precondition
        (and
          (ocupada ?p1)
          (estaNa2 ?prod ?p1)
          (not (ocupado ?l1))
          (not (ocupado ?l2))
          (not (ocupado ?l3))
          (not (ocupado ?l4))
          (disponivel ?l1)
          (disponivel ?l2)
          (= (coordx ?p1) (cx ?l1))
          (= (coordy ?p1) (cy ?l1))
          (= (cx ?l1) (- (cx ?l2) 0))
          (= (cy ?l1) (- (cy ?l2) 1))
          (= (cz ?l1) (- (cz ?l2) 0))
          (= (cx ?l1) (- (cx ?l3) 0))
          (= (cy ?l1) (- (cy ?l3) 0))
          (= (cz ?l1) (- (cz ?l3) 1))
          (= (cx ?l1) (- (cx ?l4) 0))
          (= (cy ?l1) (- (cy ?l4) 1))
          (= (cz ?l1) (- (cz ?l4) 1))
        )
      :effect
        (and
          (not (estaNa2 ?prod ?p1))
```

```
          (estaEm2 ?prod ?l1)
          (not (ocupada ?p1))
          (ocupado ?l1)
          (ocupado ?l2)
          (disponivel ?l3)
          (disponivel ?l4)
          (assign (crdx2 ?prod) (cx ?l1))
          (assign (crdy2 ?prod) (cy ?l1))
          (assign (crdz2 ?prod) (cz ?l1))
)
    )
    (:action soltar3
     :parameters (?p1 - ponteRolante ?prod - produto3 ?l1 -
lugar ?l2 - lugar ?l3 - lugar)
     :precondition
       (and
         (ocupada ?p1)
         (estaNa3 ?prod ?p1)
         (not (ocupado ?l1))
         (not (ocupado ?l2))
         (not (ocupado ?l3))
         (disponivel ?l1)
         (= (coordx ?p1) (cx ?l1))
         (= (coordy ?p1) (cy ?l1))
         (= (cx ?l1) (- (cx ?l2) 0))
         (= (cy ?l1) (- (cy ?l2) 0))
         (= (cz ?l1) (- (cz ?l2) 1))
         (= (cx ?l1) (- (cx ?l3) 0))
         (= (cy ?l1) (- (cy ?l3) 0))
         (= (cz ?l1) (- (cz ?l3) 2))
       )
     :effect
       (and
         (not (estaNa3 ?prod ?p1))
         (estaEm3 ?prod ?l1)
         (not (ocupada ?p1))
         (ocupado ?l1)
         (ocupado ?l2)
         (disponivel ?l2)
         (disponivel ?l3)
         (assign (crdx3 ?prod) (cx ?l1))
         (assign (crdy3 ?prod) (cy ?l1))
         (assign (crdz3 ?prod) (cz ?l1))
       )
    )
)
```

dominio4.pddl modelo simplificado

```
(define (domain New_Project_1)
    (:requirements :strips)
    (:types
      lugar - object
      produto1 - object
      produto2 - object
      produto3 - object
    )
    (:predicates
```

```
      (estaEm1 ?pro - produto1 ?lug - lugar)
      (plivre1 ?prod - produto1)
      (estaEm2 ?pro - produto2 ?lug - lugar)
      (plivre2 ?prod - produto2)
      (estaEm3 ?pro - produto3 ?lug - lugar)
      (plivre3 ?prod - produto3)
      (atras ?lug - lugar ?lug - lugar)
      (esquerda ?lug - lugar ?lug - lugar)
      (acima ?lug - lugar ?lug - lugar)
      (livre ?lug - lugar)
      (piso ?lug - lugar)
    )
    (:action moverPiso1
     :parameters (?prod - produto1 ?l1 - lugar ?l2 - lugar)
     :precondition
       (and
         (plivre1 ?prod)
         (livre ?l1)
         (piso ?l1)
         (livre ?l2)
         (piso ?l2)
         (atras ?l2 ?l1)
       )
     :effect
       (and
         (estaEm1 ?prod ?l1)
         (not (plivre1 ?prod))
         (not (livre ?l1))
         (not (livre ?l2))
       )
    )
    (:action moverPiso2
     :parameters (?prod - produto2 ?l1 - lugar ?l2 - lugar)
     :precondition
       (and
         (plivre2 ?prod)
         (livre ?l1)
         (piso ?l1)
         (livre ?l2)
         (piso ?l2)
         (esquerda ?l2 ?l1)
       )
     :effect
       (and
         (estaEm2 ?prod ?l1)
         (not (plivre2 ?prod))
         (not (livre ?l1))
         (not (livre ?l2))
       )
    )
    (:action moverPiso3
     :parameters (?prod - produto3 ?l1 - lugar ?l2 - lugar)
     :precondition
       (and
         (plivre3 ?prod)
         (livre ?l1)
```

```
          (piso ?l1)
          (acima ?l2 ?l1)
        )
      :effect
        (and
          (estaEm3 ?prod ?l1)
          (not (plivre3 ?prod))
          (not (livre ?l1))
          (not (livre ?l2))
        )
    )
    (:action empilhar1
     :parameters (?prod - produto1 ?l1 - lugar ?l2 - lugar ?l3
- lugar ?l4 - lugar)
     :precondition
       (and
         (plivre1 ?prod)
         (not (livre ?l1))
         (not (livre ?l2))
         (atras ?l4 ?l3)
         (acima ?l3 ?l1)
         (acima ?l4 ?l2)
       )
     :effect
       (and
         (estaEm1 ?prod ?l3)
         (not (plivre1 ?prod))
         (not (livre ?l3))
         (not (livre ?l4))
       )
    )
    (:action empilhar2
     :parameters (?prod - produto2 ?l1 - lugar ?l2 - lugar ?l3
- lugar ?l4 - lugar)
     :precondition
       (and
         (plivre2 ?prod)
         (not (livre ?l1))
         (not (livre ?l2))
         (esquerda ?l4 ?l3)
         (acima ?l3 ?l1)
         (acima ?l4 ?l2)
       )
     :effect
       (and
         (estaEm2 ?prod ?l3)
         (not (plivre2 ?prod))
         (not (livre ?l3))
         (not (livre ?l4))
       )
    )
    (:action empilhar3
     :parameters (?prod - produto3 ?l1 - lugar ?l2 - lugar ?l3
- lugar)
     :precondition
       (and
```

```
          (plivre3 ?prod)
          (not (livre ?l1))
          (acima ?l3 ?l2)
          (acima ?l2 ?l1)
        )
      :effect
        (and
          (estaEm3 ?prod ?l2)
          (not (plivre3 ?prod))
          (not (livre ?l2))
          (not (livre ?l3))
        )
    )
)
```

problema4.pddl domínio completo

```
(define (problem Planning_Problem)
    (:domain New_Project_1)
(:objects
      p1 - ponteRolante
      l1 - lugar
      l2 - lugar
      l3 - lugar
      l4 - lugar
      l5 - lugar
      l6 - lugar
      l7 - lugar
      l8 - lugar
      l9 - lugar
      l10 - lugar
      l11 - lugar
      l12 - lugar
      l13 - lugar
      l14 - lugar
      l15 - lugar
      l16 - lugar
      l17 - lugar
      l18 - lugar
      l19 - lugar
      l20 - lugar
      l21 - lugar
      l22 - lugar
      l23 - lugar
      l24 - lugar
      l25 - lugar
      l26 - lugar
      l27 - lugar
      l28 - lugar
      l29 - lugar
      l30 - lugar
      l31 - lugar
      l32 - lugar
      l33 - lugar
      l34 - lugar
```

```
        l35 - lugar
        l36 - lugar
        prod1 - produto1
        l37 - lugar
        l38 - lugar
        l39 - lugar
        l40 - lugar
        prod2 - produto1
        l41 - lugar
        l42 - lugar
        l43 - lugar
        l44 - lugar
        prod3 - produto2
        l45 - lugar
        l46 - lugar
        l47 - lugar
        l48 - lugar
        prod4 - produto2
        l49 - lugar
        l50 - lugar
        l51 - lugar
        l52 - lugar
        prod5 - produto3
        l53 - lugar
        l54 - lugar
        l55 - lugar
        prod6 - produto3
        l56 - lugar
        l57 - lugar
        l58 - lugar
        prod7 - produto3
        l59 - lugar
        l60 - lugar
        l61 - lugar
        prod8 - produto3
        l62 - lugar
        l63 - lugar
        l64 - lugar
        prod9 - produto3
        l65 - lugar
        l66 - lugar
        l67 - lugar
        prod10 - produto3
        l68 - lugar
        l69 - lugar
        l70 - lugar
        prod11 - produto3
        l71 - lugar
        l72 - lugar
        l73 - lugar
        prod12 - produto3
        l74 - lugar
        l75 - lugar
        l76 - lugar
        prod13 - produto3
        l77 - lugar
```

```
      l78 - lugar
      l79 - lugar
)
    (:init
      (= (coordx p1) -10)
      (= (coordy p1) -10)
      (disponivel l1)
      (= (cx l1) 0)
      (= (cy l1) 0)
      (= (cz l1) 0)
      (disponivel l2)
      (= (cx l2) 1)
      (= (cy l2) 0)
      (= (cz l2) 0)
      (disponivel l3)
      (= (cx l3) 2)
      (= (cy l3) 0)
      (= (cz l3) 0)
      (disponivel l4)
      (= (cx l4) 0)
      (= (cy l4) 1)
      (= (cz l4) 0)
      (disponivel l5)
      (= (cx l5) 1)
      (= (cy l5) 1)
      (= (cz l5) 0)
      (disponivel l6)
      (= (cx l6) 2)
      (= (cy l6) 1)
      (= (cz l6) 0)
      (disponivel l7)
      (= (cx l7) 0)
      (= (cy l7) 2)
      (= (cz l7) 0)
      (disponivel l8)
      (= (cx l8) 1)
      (= (cy l8) 2)
      (= (cz l8) 0)
      (disponivel l9)
      (= (cx l9) 2)
      (= (cy l9) 2)
      (= (cz l9) 0)
      (= (cx l10) 0)
      (= (cy l10) 0)
      (= (cz l10) 1)
      (= (cx l11) 1)
      (= (cy l11) 0)
      (= (cz l11) 1)
      (= (cx l12) 2)
      (= (cy l12) 0)
      (= (cz l12) 1)
      (= (cx l13) 0)
      (= (cy l13) 1)
      (= (cz l13) 1)
      (= (cx l14) 1)
      (= (cy l14) 1)
```

```
        (= (cz l14) 1)
        (= (cx l15) 2)
        (= (cy l15) 1)
        (= (cz l15) 1)
        (= (cx l16) 0)
        (= (cy l16) 2)
        (= (cz l16) 1)
        (= (cx l17) 1)
        (= (cy l17) 2)
        (= (cz l17) 1)
        (= (cx l18) 2)
        (= (cy l18) 2)
        (= (cz l18) 1)
        (= (cx l19) 0)
        (= (cy l19) 0)
        (= (cz l19) 2)
        (= (cx l20) 1)
        (= (cy l20) 0)
        (= (cz l20) 2)
        (= (cx l21) 2)
        (= (cy l21) 0)
        (= (cz l21) 2)
        (= (cx l22) 0)
        (= (cy l22) 1)
        (= (cz l22) 2)
        (= (cx l23) 1)
        (= (cy l23) 1)
        (= (cz l23) 2)
        (= (cx l24) 2)
        (= (cy l24) 1)
        (= (cz l24) 2)
        (= (cx l25) 0)
        (= (cy l25) 2)
        (= (cz l25) 2)
        (= (cx l26) 1)
        (= (cy l26) 2)
        (= (cz l26) 2)
        (= (cx l27) 2)
        (= (cy l27) 2)
        (= (cz l27) 2)
        (= (cx l28) 0)
        (= (cy l28) 0)
        (= (cz l28) 3)
        (= (cx l29) 1)
        (= (cy l29) 0)
        (= (cz l29) 3)
        (= (cx l30) 2)
        (= (cy l30) 0)
        (= (cz l30) 3)
        (= (cx l31) 0)
        (= (cy l31) 1)
        (= (cz l31) 3)
        (= (cx l32) 1)
        (= (cy l32) 1)
        (= (cz l32) 3)
        (= (cx l33) 2)
```

```
(= (cy l33) 1)
(= (cz l33) 3)
(= (cx l34) 0)
(= (cy l34) 2)
(= (cz l34) 3)
(= (cx l35) 1)
(= (cy l35) 2)
(= (cz l35) 3)
(= (cx l36) 2)
(= (cy l36) 2)
(= (cz l36) 3)
(estaEm1 prod1 l37)
(= (crdx1 prod1) -10)
(= (crdy1 prod1) -10)
(= (crdz1 prod1) 0)
(ocupado l37)
(disponivel l37)
(= (cx l37) -10)
(= (cy l37) -10)
(= (cz l37) 0)
(ocupado l38)
(disponivel l38)
(= (cx l38) -9)
(= (cy l38) -10)
(= (cz l38) 0)
(disponivel l39)
(= (cx l39) -10)
(= (cy l39) -10)
(= (cz l39) 1)
(disponivel l40)
(= (cx l40) -9)
(= (cy l40) -10)
(= (cz l40) 1)
(estaEm1 prod2 l41)
(= (crdx1 prod2) -10)
(= (crdy1 prod2) -10)
(= (crdz1 prod2) 0)
(ocupado l41)
(disponivel l41)
(= (cx l41) -10)
(= (cy l41) -10)
(= (cz l41) 0)
(ocupado l42)
(disponivel l42)
(= (cx l42) -9)
(= (cy l42) -10)
(= (cz l42) 0)
(disponivel l43)
(= (cx l43) -10)
(= (cy l43) -10)
(= (cz l43) 1)
(disponivel l44)
(= (cx l44) -9)
(= (cy l44) -10)
(= (cz l44) 1)
(estaEm2 prod3 l45)
```

```
(= (crdx2 prod3) -10)
(= (crdy2 prod3) -10)
(= (crdz2 prod3) 0)
(ocupado l45)
(disponivel l45)
(= (cx l45) -10)
(= (cy l45) -10)
(= (cz l45) 0)
(ocupado l46)
(disponivel l46)
(= (cx l46) -10)
(= (cy l46) -9)
(= (cz l46) 0)
(disponivel l47)
(= (cx l47) -10)
(= (cy l47) -10)
(= (cz l47) 1)
(disponivel l48)
(= (cx l48) -10)
(= (cy l48) -9)
(= (cz l48) 1)
(estaEm2 prod4 l49)
(= (crdx2 prod4) -10)
(= (crdy2 prod4) -10)
(= (crdz2 prod4) 0)
(ocupado l49)
(disponivel l49)
(= (cx l49) -10)
(= (cy l49) -10)
(= (cz l49) 0)
(ocupado l50)
(disponivel l50)
(= (cx l50) -10)
(= (cy l50) -9)
(= (cz l50) 0)
(disponivel l51)
(= (cx l51) -10)
(= (cy l51) -10)
(= (cz l51) 1)
(disponivel l52)
(= (cx l52) -10)
(= (cy l52) -9)
(= (cz l52) 1)
(estaEm3 prod5 l53)
(= (crdx3 prod5) -10)
(= (crdy3 prod5) -10)
(= (crdz3 prod5) 0)
(ocupado l53)
(disponivel l53)
(= (cx l53) -10)
(= (cy l53) -10)
(= (cz l53) 0)
(ocupado l54)
(disponivel l54)
(= (cx l54) -10)
(= (cy l54) -10)
```

```
(= (cz l54) 1)
(disponivel l55)
(= (cx l55) -10)
(= (cy l55) -10)
(= (cz l55) 2)
(estaEm3 prod6 l56)
(= (crdx3 prod6) -10)
(= (crdy3 prod6) -10)
(= (crdz3 prod6) 0)
(ocupado l56)
(disponivel l56)
(= (cx l56) -10)
(= (cy l56) -10)
(= (cz l56) 0)
(ocupado l57)
(disponivel l57)
(= (cx l57) -10)
(= (cy l57) -10)
(= (cz l57) 1)
(disponivel l58)
(= (cx l58) -10)
(= (cy l58) -10)
(= (cz l58) 2)
(estaEm3 prod7 l59)
(= (crdx3 prod7) -10)
(= (crdy3 prod7) -10)
(= (crdz3 prod7) 0)
(ocupado l59)
(disponivel l59)
(= (cx l59) -10)
(= (cy l59) -10)
(= (cz l59) 0)
(ocupado l60)
(disponivel l60)
(= (cx l60) -10)
(= (cy l60) -10)
(= (cz l60) 1)
(disponivel l61)
(= (cx l61) -10)
(= (cy l61) -10)
(= (cz l61) 2)
(estaEm3 prod8 l62)
(= (crdx3 prod8) -10)
(= (crdy3 prod8) -10)
(= (crdz3 prod8) 0)
(ocupado l62)
(disponivel l62)
(= (cx l62) -10)
(= (cy l62) -10)
(= (cz l62) 0)
(ocupado l63)
(disponivel l63)
(= (cx l63) -10)
(= (cy l63) -10)
(= (cz l63) 1)
(disponivel l64)
```

```
        (= (cx l64) -10)
        (= (cy l64) -10)
        (= (cz l64) 2)
        (estaEm3 prod9 l65)
        (= (crdx3 prod9) -10)
        (= (crdy3 prod9) -10)
        (= (crdz3 prod9) 0)
        (ocupado l65)
        (disponivel l65)
        (= (cx l65) -10)
        (= (cy l65) -10)
        (= (cz l65) 0)
        (ocupado l66)
        (disponivel l66)
        (= (cx l66) -10)
        (= (cy l66) -10)
        (= (cz l66) 1)
        (disponivel l67)
        (= (cx l67) -10)
        (= (cy l67) -10)
        (= (cz l67) 2)
        (estaEm3 prod10 l68)
        (= (crdx3 prod10) -10)
        (= (crdy3 prod10) -10)
        (= (crdz3 prod10) 0)
        (ocupado l68)
        (disponivel l68)
        (= (cx l68) -10)
        (= (cy l68) -10)
        (= (cz l68) 0)
        (ocupado l69)
        (disponivel l69)
        (= (cx l69) -10)
        (= (cy l69) -10)
        (= (cz l69) 1)
        (disponivel l70)
        (= (cx l70) -10)
        (= (cy l70) -10)
        (= (cz l70) 2)
        (estaEm3 prod11 l71)
        (= (crdx3 prod11) -10)
        (= (crdy3 prod11) -10)
        (= (crdz3 prod11) 0)
        (ocupado l71)
        (disponivel l71)
        (= (cx l71) -10)
        (= (cy l71) -10)
        (= (cz l71) 0)
        (ocupado l72)
        (disponivel l72)
        (= (cx l72) -10)
        (= (cy l72) -10)
        (= (cz l72) 1)
        (disponivel l73)
        (= (cx l73) -10)
        (= (cy l73) -10)
```

```
      (= (cz l73) 2)
      (estaEm3 prod12 l74)
      (= (crdx3 prod12) -10)
      (= (crdy3 prod12) -10)
      (= (crdz3 prod12) 0)
      (ocupado l74)
      (disponivel l74)
      (= (cx l74) -10)
      (= (cy l74) -10)
      (= (cz l74) 0)
      (ocupado l75)
      (disponivel l75)
      (= (cx l75) -10)
      (= (cy l75) -10)
      (= (cz l75) 1)
      (disponivel l76)
      (= (cx l76) -10)
      (= (cy l76) -10)
      (= (cz l76) 2)
      (estaEm3 prod13 l77)
      (= (crdx3 prod13) -10)
      (= (crdy3 prod13) -10)
      (= (crdz3 prod13) 0)
      (ocupado l77)
      (disponivel l77)
      (= (cx l77) -10)
      (= (cy l77) -10)
      (= (cz l77) 0)
      (ocupado l78)
      (disponivel l78)
      (= (cx l78) -10)
      (= (cy l78) -10)
      (= (cz l78) 1)
      (disponivel l79)
      (= (cx l79) -10)
      (= (cy l79) -10)
      (= (cz l79) 2)
    )
    (:goal
      (and
        (not (ocupada p1))
        (= (crdx1 prod1) 1)
        (= (crdy1 prod1) 2)
        (= (crdz1 prod1) 0)
        (= (crdx1 prod2) 0)
        (= (crdy1 prod2) 0)
        (= (crdz1 prod2) 2)
        (= (crdx2 prod3) 2)
        (= (crdy2 prod3) 0)
        (= (crdz2 prod3) 2)
        (= (crdx2 prod4) 0)
        (= (crdy2 prod4) 1)
        (= (crdz2 prod4) 2)
        (= (crdx3 prod5) 0)
        (= (crdy3 prod5) 0)
        (= (crdz3 prod5) 0)
```

```
        (= (crdx3 prod6) 1)
        (= (crdy3 prod6) 0)
        (= (crdz3 prod6) 0)
        (= (crdx3 prod7) 2)
        (= (crdy3 prod7) 0)
        (= (crdz3 prod7) 0)
        (= (crdx3 prod8) 0)
        (= (crdy3 prod8) 1)
        (= (crdz3 prod8) 0)
        (= (crdx3 prod9) 1)
        (= (crdy3 prod9) 1)
        (= (crdz3 prod9) 0)
        (= (crdx3 prod10) 2)
        (= (crdy3 prod10) 1)
        (= (crdz3 prod10) 0)
        (= (crdx3 prod11) 0)
        (= (crdy3 prod11) 2)
        (= (crdz3 prod11) 0)
        (= (crdx3 prod12) 1)
        (= (crdy3 prod12) 2)
        (= (crdz3 prod12) 1)
        (= (crdx3 prod13) 2)
        (= (crdy3 prod13) 2)
        (= (crdz3 prod13) 1)
      )
    )
)
```

problema4.pddl domínio simplificado

```
(define (problem Planning_Problem)
    (:domain New_Project_1)
    (:objects
      l1 - lugar
      l2 - lugar
      l3 - lugar
      l4 - lugar
      l5 - lugar
      l6 - lugar
      l7 - lugar
      l8 - lugar
      l9 - lugar
      l10 - lugar
      l11 - lugar
      l12 - lugar
      l13 - lugar
      l14 - lugar
      l15 - lugar
      l16 - lugar
      l17 - lugar
      l18 - lugar
      l19 - lugar
      l20 - lugar
      l21 - lugar
      l22 - lugar
      l23 - lugar
      l24 - lugar
      l25 - lugar
```

```
      l26 - lugar
      l27 - lugar
      prod1 - produto1
      prod2 - produto1
      prod3 - produto2
      prod4 - produto2
      prod5 - produto3
      prod6 - produto3
      prod7 - produto3
      prod8 - produto3
      prod9 - produto3
      prod10 - produto3
      prod11 - produto3
      prod12 - produto3
      prod13 - produto3
    )
    (:init
      (livre l1)
      (piso l1)
      (livre l2)
      (piso l2)
      (livre l3)
      (piso l3)
      (livre l4)
      (piso l4)
      (livre l5)
      (piso l5)
      (livre l6)
      (piso l6)
      (livre l7)
      (piso l7)
      (livre l8)
      (piso l8)
      (livre l9)
      (piso l9)
      (livre l10)
      (livre l11)
      (livre l12)
      (livre l13)
      (livre l14)
      (livre l15)
      (livre l16)
      (livre l17)
      (livre l18)
      (livre l19)
      (livre l20)
      (livre l21)
      (livre l22)
      (livre l23)
      (livre l24)
      (livre l25)
      (livre l26)
      (livre l27)
      (atras l2 l1)
      (atras l3 l2)
      (atras l5 l4)
```

```
(atras l6 l5)
(atras l8 l7)
(atras l9 l8)
(atras l11 l10)
(atras l12 l11)
(atras l14 l13)
(atras l15 l14)
(atras l17 l16)
(atras l18 l17)
(atras l20 l19)
(atras l21 l20)
(atras l23 l22)
(atras l24 l23)
(atras l26 l25)
(atras l27 l26)
(esquerda l4 l1)
(esquerda l5 l2)
(esquerda l6 l3)
(esquerda l7 l4)
(esquerda l8 l5)
(esquerda l9 l6)
(esquerda l13 l10)
(esquerda l14 l11)
(esquerda l15 l12)
(esquerda l16 l13)
(esquerda l17 l14)
(esquerda l18 l15)
(esquerda l22 l19)
(esquerda l23 l20)
(esquerda l24 l21)
(esquerda l25 l22)
(esquerda l26 l23)
(esquerda l27 l24)
(acima l10 l1)
(acima l11 l2)
(acima l12 l3)
(acima l13 l4)
(acima l14 l5)
(acima l15 l6)
(acima l16 l7)
(acima l17 l8)
(acima l18 l9)
(acima l19 l10)
(acima l20 l11)
(acima l21 l12)
(acima l22 l13)
(acima l23 l14)
(acima l24 l15)
(acima l25 l16)
(acima l26 l17)
(acima l27 l18)
(plivre1 prod1)
(plivre1 prod2)
(plivre2 prod3)
(plivre2 prod4)
(plivre3 prod5)
```

```
      (plivre3 prod6)
      (plivre3 prod7)
      (plivre3 prod8)
      (plivre3 prod9)
      (plivre3 prod10)
      (plivre3 prod11)
      (plivre3 prod12)
      (plivre3 prod13)
    )
    (:goal
      (and
        (estaEm1 prod1 l8)
        (estaEm1 prod2 l19)
        (estaEm2 prod3 l21)
        (estaEm2 prod4 l22)
        (estaEm3 prod5 l1)
        (estaEm3 prod6 l2)
        (estaEm3 prod7 l3)
        (estaEm3 prod8 l4)
        (estaEm3 prod9 l5)
        (estaEm3 prod10 l6)
        (estaEm3 prod11 l7)
        (estaEm3 prod12 l17)
        (estaEm3 prod13 l18)
      )
    )
)
```