Universidade Federal do Maranhão
Centro de Ciências Exatas e Tecnologia
Curso de Ciência da Computação

Moizanilton Pestana Soares

Simulador de Terminais Portuários Graneleiros Sujeitos a Restrições de Maré e Nível de Estoque

São Luís
2013

Moizanilton Pestana Soares

Simulador de Terminais Portuários Graneleiros Sujeitos a Restrições de Maré e Nível de Estoque

Monografia apresentada ao Curso de Ciência da Computação da UFMA, como requisito parcial para a obtenção do grau de BACHAREL em Ciência da Computação.

Orientador: Alexandre César Muniz de Oliveira

Prof. Dr. em Computação Aplicada pelo Instituto Nacional de Pesquisas Espaciais - INPE

São Luís

2013

Soares, Moizanilton Pestana.

Simulador de terminais portuários graneleiros sujeitos a restrição de maré e nível de estoque/ Moizanilton Pestana Soares. – São Luís, 2013.

54 f.

Impresso por computador (fotocópia).

Orientador: Alexandre César Muniz de Oliveira.

. Monografia (Graduação) – Universidade Federal do Maranhão, Curso de Ciência da Computação, 2013.

1. Simulação computacional. 2. Pesquisa operacional. 3. Problema de alocação de berços. I. Título.

CDU 004.891

Moizanilton Pestana Soares

Simulador de Terminais Portuários Graneleiros Sujeitos a Restrições de Maré e Nível de Estoque

Monografia apresentada ao Curso de Ciência da Computação da UFMA, como requisito parcial para a obtenção do grau de BACHAREL em Ciência da Computação.

Aprovado em 18 de Dezembro de 2013

BANCA EXAMINADORA

Alexandre César Muniz de Oliveira

Prof. Dr. em Computação Aplicada pelo Instituto Nacional de Pesquisas Espaciais - INPE

Maria Auxiliadora Freire

Prof.ª Msc. em Ciência de Engenharia - PUC-RIO

Geraldo Braz Junior

Prof. Msc. em Engenharia de Eletricidade - UFMA

Para minha mãe.

Agradecimentos

Agradeço à minha família, por dar sentido a este esforço monográfico. Ao meu orientador, o Prof. Alexandre César Muniz de Oliveira, pela paciência e disponibilidade. E aos meus colegas de trabalho da UFMA e do Hospital Universitário, que contribuíram com dicas e incentivo.

Resumo

Berços são áreas do cais onde os navios fazem a atracação para carregamento e descarregamento. Eles estão entre os principais recursos de um terminal portuário, e a forma de gerenciá-los reflete-se no tempo de estada dos navios e no nível de utilização dos demais recursos do terminal. A busca por uma maior eficiência nas operacões portuárias deu origem a estudos que visam encontrar melhores formas de se manejar esses recursos. O Problema de Alocação de Berços (BAP, do inglês *Berth Allocation Problem*) consiste em atribuir navios a posições de berço de forma ótima; em definir quando e onde os navios esperados irão atracar de maneira a maximizar os indicadores de eficiência portuária. A literatura sobre o tema se concentra majoritariamente em terminais de contêiner, mas neste trabalho referenciou-se um artigo que aborda esse problema de otimização em terminais com as características encontradas no Complexo Portuário de São Luís. Nesse estudo, é apresentada uma formulação do BAP para terminais graneleiros que sofrem influência das marés e do controle de estoque em sua operação (chamada BAPTGS, do inglês *Berth Allocation Problem in Tidal Grain Ports with Stock Level Constraint).* Ademais, é construído um modelo de Programação Linear (PL) do problema e apresentado um formato de instância compatível com o CPLEX Optimizer, um *solver* comercial para problemas de PL. Este trabalho introduz uma contribuição a tal estudo. É apresentado aqui um simulador de terminal portuário capaz de gerar instâncias do BAPTGS no formato especificado. O objetivo é que a aplicação funcione como um gerador de instâncias prático e flexível, com múltiplas configurações, e que contribua para as pesquisas sobre soluções de TI que auxiliem na tomada de decisões em relação à alocação de berços nos terminais portuários de São Luís.

Palavras-chave: Pesquisa Operacional, Simulação Computacional, Problema de Alocação de Berços.

Abstract

Berths are places alongside a quay where a vessel is moored for loading and unloading of cargo. They are among the key resources of a port terminal, and how to manage them is reflected in the length of stay of vessels and the level of utilization of the other resources of the terminal. The search for greater efficiency in port operations has led to studies aimed at finding better ways to manage these resources. The Berth Allocation Problem (BAP) consists of assigning ships to berthing positions in an optimal way. It consists of determining where and when ships will moor in order to maximize the port performance indicators. The literature on the subject focuses mainly on container terminals, but in this work we reference a paper that addresses this optimization problem in terminal with features found in Port Complex of São Luís. In that study, the Berth Allocation Problem is formulated to bulk terminals that are influenced by tides and stock control in its operation (BAPTGS). Moreover, it is built a Linear Programming model of the problem and presented an instance format compatible with CPLEX Optimizer, a commercial LP solver. This paper introduces a contribution to that study. We present a port terminal simulator capable of generating instances of BAPTGS in that specified format. The objective is that the application works as a convenient and flexible instance generator, with plenty of configuration options, and that it contributes to research on IT solutions that assist with decision making regarding the allocation of berths in port terminals of São Luís.

Keywords: Operational Research, Computer Simulation, Berth Allocation Problem.

"Não importa ao tempo o minuto que passa, mas o minuto que vem."

(Machado de Assis)

Sumário

Lista de Figuras

1 Introdução

O transporte marítimo de cargas é utilizado desde a antiguidade e, no mundo moderno e globalizado, só vem ganhando importância. Estima-se que 70% de todas as mercadorias que circulam no planeta são transportadas por mar[6]. Os massivos investimentos em infraestrutura portuária e na construção de navios cada vez maiores ampliam as vantagens competitivas deste meio de transporte em relação a outros. Navios de contêiner modernos podem transportar carga por uma fração do custo que seria despendido por terra em carga equivalente[7].

No Brasil, o transporte marítimo tem valor estratégico. O país é um dos maiores produtores de *commodities* do mundo, além de possuir 7.500 km de litoral, com condições favoráveis de navegação[8]. É embarcada em grandes navios cargueiros que o país exporta a maior parte de sua produção. E esse movimento vem aumentando ao longo dos anos, como demonstram dados da Agência Nacional de Transportes (ANTAQ) publicados em forma de gráfico em 2012[1] (Figura 1.1).

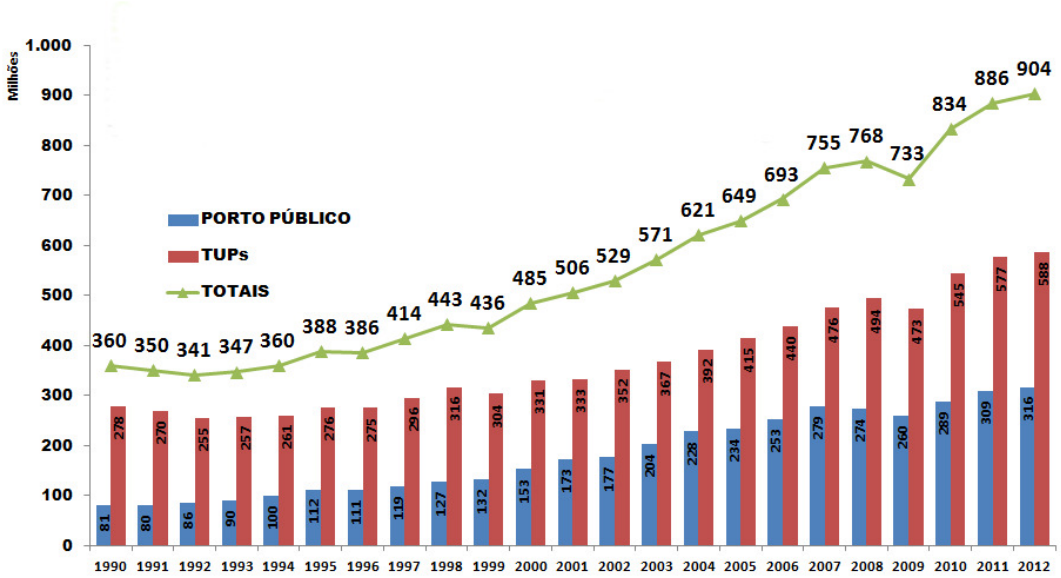

Figura 1.1: Movimentação de cargas nos portos brasileiros (em milhões de toneladas). Adaptado de ANTAQ[1]

Tendo uma importância econômica tão grande, não surpreende o ambiente de

[1]No gráfico, TUP se refere a Terminais de Uso Privado

intensa competição entre os portos ao redor do mundo. Os administradores de porto buscam atrair clientes, oferecendo preços competitivos e, para isso, é imprescindível utilizar os recursos do porto com eficiência. E entre os recursos mais importantes estão os berços. Berços são áreas do cais equipadas com guindastes onde os navios fazem a atracação para carregamento e/ou descarregamento. Determinar, de forma ótima, em que berço e em que momento os navios esperados irão atracar é um problema conhecido em Pesquisa Operacional como Problema de Alocação de Berços.

O objetivo deste trabalho é apresentar um programa simulador de terminal portuário que possa gerar instâncias para o Problema de Alocação de Berços em terminais graneleiros sujeitos a restrições de maré e nível de estoque. Entenda-se por instância um arquivo com um formato padrão, que possa ser lido e submetido a um programa otimizador que o solucione. O simulador foi modelado com base no Complexo Marítimo Industrial de São Luís, um dos mais importantes do Brasil, e que possui como característica peculiar a interferência das marés em sua operação.

O restante do texto é organizado da seguinte forma: no Capítulo 2, o Problema de Alocação de Berços é definido detalhadamente, incluindo uma breve revisão bibliográfica sobre o assunto. O Capítulo 3 contém os conceitos mais relevantes sobre Simulação e Teoria de Filas utilizados na construção do simulador. O Capítulo 4 apresenta a aplicação desenvolvida, expondo detalhes de projeto e implementação. E, finalmente, o quinto capítulo relata as conclusões tiradas no decorrer do projeto, além de abordar algumas perspectivas futuras.

2 Problema de Alocação de Berços

Ao chegar em um porto, um navio aguarda autorização para atracar no cais. O cais é uma plataforma que avança sobre a água para a facilitar as operacões de carregamento e descarregamento. Os locais onde os navios podem atracar são chamados de berços. Estes são equipados com guindastes capazes de carregar e descarregar contêineres, que são transferidos por caminhões para a área de armazenamento do porto[9].

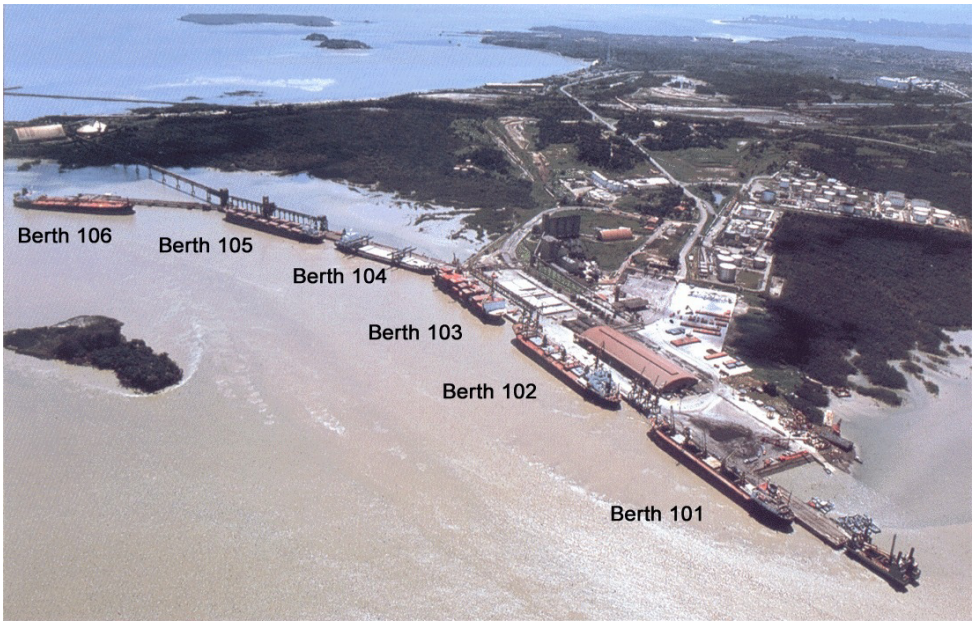

Figura 2.1: Vista aérea do Porto de Itaqui. Fonte: Ponto a Porto[2].

Um terminal portuário possui vários berços de atracação, que podem comportar um ou mais navios, dependendo do tamanho de cada um. Cada berço possui características estruturais próprias, como extensão e profundidade, que impõem restricões aos tipos de carga com que podem lidar. Por exemplo, no Porto de Itaqui (Figura 2.1), um dos mais importantes portos brasileiros, localizado em São Luís do Maranhão, os berços são denominados por um número entre 101 e 106. O berço 103 apresenta profundidade de 13 metros em toda extensão e destina-se à movimentação de derivados de petróleo, soda cáustica, sebo bovino e carga geral; já o berço 106 trata-se de um píer petroleiro, destinado exclusivamente a derivados de petróleo, e permite a operação de navios de até

200 mil toneladas de DWT[1].

O Problema de Alocação de Berços (BAP, do inglês *Berth Allocation Problem*) consiste em atribuir navios a posições de berço ou, mais precisamente, em determinar *onde* e *quando* os navios esperados irão atracar[11]. Para isso, o operador do porto dispõe, com algumas semanas de antecedência, de informações sobre a chegada dos navios, tais como data e hora esperada de chegada (ETA, em inglês *Expected Time of Arrival*), tipo e tamanho do navio, natureza da carga e profundidade de água requerida. E deve resolver esse problema, com dois objetivos básicos em vista:

- Satisfazer os clientes, buscando minimizar o tempo de estada (a soma do tempo de espera e do tempo de atendimento);
- Atingir os objetivos do porto, que são minimizar custos e aumentar lucros, utilizando os recursos disponíveis — que incluem máquinas de alto custo de aquisição e manutenção — da maneira mais eficiente possível.

A tarefa de definir a sequência de atracação dos navios esperados se complica porque diversos fatores devem ser considerados. Na maioria dos portos comerciais, por exemplo, o atendimento deve obedecer a ordem de chegada — o primeiro a chegar tem direito a ser atendido primeiro. Essa regra só pode ser desobedecida, caso haja um entendimento entre as partes[3]. Outra regra que é também bastante difundida é a que determina que não pode haver trocas de berço, o que significa que, uma vez atracado, um navio não pode ser deslocado para outro berço. Há também questões de segurança, como distância mínima obrigatória entre navios atracados. Fatores naturais, como marés, que impedem tanto a atracação quanto a saída de navios em determinados horários. Fatores arquiteturais, como distância entre o berço e a área de armazenamento, que tem efeito direto no tempo de carregamento e descarregamento. Há os contratos com os clientes, que prevêem multas, em caso de atraso. Enfim, não há a pretensão neste trabalho de elencar exaustivamente todas as váriáveis envolvidas, mas apenas ressaltar a complexidade do problema e demonstrar a necessidade de um sistema automatizado que auxilie na tomada de decisões.

[1]Porte bruto (*deadweight*), isto é, peso máximo, incluindo o próprio navio, a carga e o lastro[10]

2.1 Revisão da literatura sobre o BAP

Devido à grande importância econômica do transporte marítimo, a necessidade de encontrar soluções que importem em uma maior eficiência operacional nos terminais portuários é cada vez maior. Consequentemente, já existe uma extensa literatura que trata do Problema de Alocação de Berços. Nesta seção, faremos uma breve revisão dos estudos sobre o tema que serviram de suporte a este trabalho. Nos interessa principalmente o modelo de porto utilizado por cada um, já que não faz parte do escopo deste trabalho encontrar uma solução própria para o BAP (ver "Formulação do Problema" na Seção 2.3).

O BAP pode ser modelado de maneira discreta (BAPD) ou contínua (BAPC). No primeiro caso, os berços podem ser vistos como segmentos de tamanho fixo ou, simplesmente, pontos no cais[9]. No segundo caso (BAPC), o cais é modelado como um espaço contínuo, dentro do qual a atracação pode ser realizada em qualquer lugar. Os resultados obtidos através do BAPC podem ser mais práticos e eficientes que o do BAPD, já que o modelo contínuo se aproxima mais da realidade, no entanto as dificuldades quanto à solução e à definição do problema são maiores[12].

No caso discreto, o BAP pode ser modelado como um problema de programação paralela em máquinas não-relacionadas[13], onde o navio é tratado como um processo e os berços, como máquinas. O tempo de chegada dos navios seria o tempo de liberação dos processos. No caso contínuo, é um problema análogo ao problema de corte de estoque, no qual deseja-se empacotar uma série de produtos em uma caixa de maneira eficiente, sem desperdiçar espaço[14]. Os navios, estejam eles esperando ou em atendimento, são representados como retângulos em um diagrama espaço-tempo (ver Figura 2.2), de forma que o problema se resume em acomodar os navios/retângulos nos espaços-tempo disponíveis em uma caixa com algum esquema restrito, em que rotações em navios/retângulos não são permitidas.

Na literatura, o BAP divide-se entre modelos estáticos e dinâmicos. Nos modelos estáticos ou SBAP (*Static Berth Allocation Problem*), todos os navios já estão no porto antes que a alocação dos berços seja definida[15]. Nos modelos dinâmicos ou DBAP (*Dynamic Berth Allocation Problem*), os navios podem chegar após a alocação ser definida, desde que as chegadas aconteçam dentro do horizonte de planejamento. Neste caso, a modelagem do BAP deve incluir o tempo esperado de chegada (ETA) dos navios.

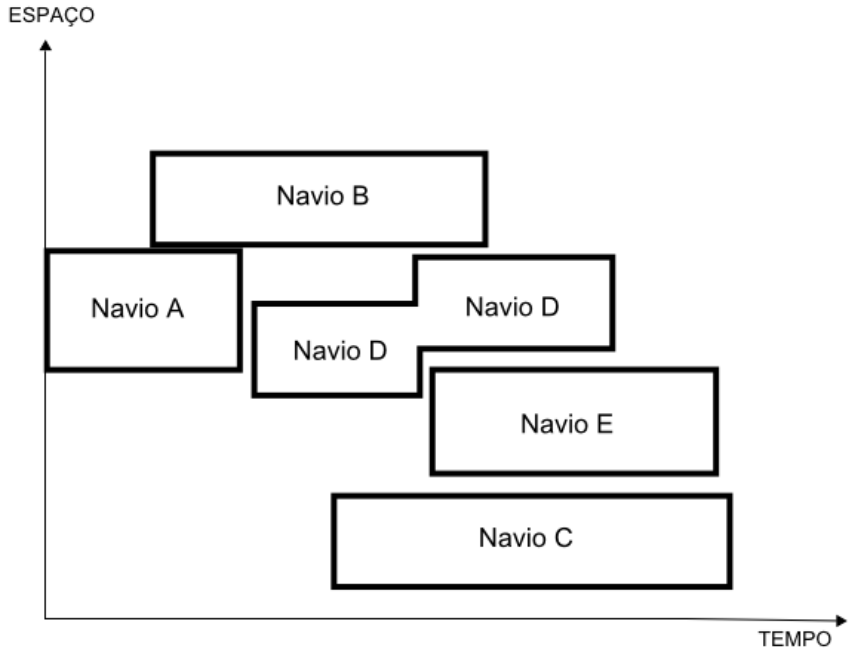

Figura 2.2: Ilustração de um diagrama espaço-tempo. Adaptado de Lee e Chen[3].

Em Imai et al. [16], os autores utilizam um modelo discreto e estático de BAP, onde os berços são um número finito de pontos no cais e as dimensões dos navios e dos berços são desconsideradas. Os autores assumem que um berço pode atender apenas um navio por vez e que não há restrições técnicas relativas à profundidade da água. Ademais, o modelo estabelece que o tempo de atendimento é dependente do berço escolhido, o que vai de encontro ao fato de que muitas vezes a distância até o local onde estão os contêineres varia de acordo com o berço. O SBAP reduz-se, nestas circunstâncias, a um Problema de Atribuição e, como tal, pode ser resolvido em tempo polinomial[13]. Eles concluem que, para o porto atingir alta produtividade, o conjunto de atribuições navio-berço deveria ser encontrado sem o emprego da regra FCFS para o atendimento. Mais tarde, esse modelo é estendido para uma versão dinâmica em Imai et al.[15] e Nishimura et al[17]. Neste último, os autores enriquecem o modelo, acrescentando a possibilidade de múltiplas configurações de profundidade de água. Imai et al.[18] retornam novamente a esse modelo BAPD, associando níveis de prioridade aos navios esperados.

Um bom exemplo de BAP contínuo e dinâmico encontra-se em Lee e Chen[3]. Como mencionado anteriormente, nestes tipos de modelo, a atracação pode ocorrer em qualquer local disponível do cais. No entanto, devido a razões práticas, adotou-se aqui uma abordagem ligeiramente diferente: cada navio possui várias seções preferenciais, dependendo da natureza da carga e do equipamento exigido para manejá-la. A atracação

pode ocorrer apenas nessas seções, cujo tamanho varia de acordo com o navio e o local. Para forçar a atracação em locais onde o atendimento é mais eficiente, cada seção recebe um nível de prioridade. Há ainda um tempo máximo de espera para cada seção preferencial. Nas seções de baixa prioridade, esse tempo é definido com um número elevado, para assegurar que os navios atraquem apropriadamente. Lee e Chen[3] apresentam uma inovação neste trabalho — a atracação deve respeitar um espaçamento mínimo de seguranca entre navios. Outro aspecto que os autores consideram é a possibilidade de trocas de berço. Li et al.[19] fazem o mesmo, mas pressupõem que a troca não causa nenhum efeito. Brown et al.[20] aborda a troca de berços, mas em portos navais.

2.2 Referência para esse trabalho

Este trabalho é motivado por um problema operacional no Complexo Marítimo Industrial de São Luís. Para realizar a modelagem da aplicação aqui proposta, utilizou-se como referência o artigo de Barros et al.[21]. Nesse artigo discute-se uma solução para o BAP em portos graneleiros sujeitos a restrições de maré e condições de estoque, denominado BAPTGS (*Berth Allocation Problem in Tidal Grain ports with Stock level conditions*) . As informações a seguir foram extraídas desse trabalho.

O complexo portuário objeto de estudo é formado pelo Porto do Itaqui e pelos terminais privados de Ponta da Madeira e Aluminum. Ambos os terminais privados estão associados a grandes empresas transnacionais — o primeiro, à mineradora Vale S.A., e o último, ao Consórcio de Alumínio do Maranhão (ALUMAR). Juntos, os três terminais movimentam a segunda maior quantidade de cargas a granel do Brasil.

Em Barros et al.[21], vários fatores foram levados em consideração na modelagem do BAP. Uma das características peculiares neste complexo portuário é a sujeição às condições de maré. Tanto a atracação quanto a saída dos navios atracados somente podem acontecer em condições de maré alta, o que ocorre em ciclos regulares de 12 horas, aproximadamente.

Outra variável importante considerada é o nível de prioridade de cada navio, que é estipulado com base em dois fatores principais: o estoque e os contratos. A prioridade no atendimento é dada segundo as necessidades do estoque. Por exemplo, caso

determinada matéria-prima atinja um nível mínimo, os navios que as contêm recebem uma prioridade elevada. A importância relativa entre os navios também é afetada pelos contratos, na medida que eles estipulam multas em caso de atraso no atendimento (conhecido como *demurrage* ou sobrestadia) e emitem créditos (*dispatch*) ao contratante, se o atendimento terminar antes do planejado.

Baseado nisso, esse artigo propõe um modelo de Programação Linear Inteira em que o objetivo é minimizar o *demurrage* total, dada as condições de maré e as restrições de estoque. O BAPTGS é modelado de forma discreta como um Problema de Transporte, onde N navios são vistos como fornecedores e M janelas de tempo, como consumidores. As janelas de tempo acontecem em função das marés favoráveis, em intervalos de 12 horas, que é o ciclo aproximado das marés. Os berços, neste modelo, podem atender apenas um navio por vez. Instâncias do problema são resolvidas através de um *solver* comercial chamado CPLEX Optimizer e por um algoritmo baseado em *Simulated Annealing*[22].

2.3 Formulação do Problema

Este trabalho objetiva propor um Simulador de Porto capaz de gerar instâncias para o Problema de Alocação de Berços em Terminais Graneleiros sujeitos a restrições de maré e nível de estoque. Uma instância corresponde a um arquivo-texto padronizado do qual é possível extrair todas as informações referentes às condições do terminal em dado instante. Informações completas, que possam ser processadas posteriormente por um otimizador (Figura 2.3), que retorne uma solução para o problema. Uma instância deverá informar o conjunto de navios esperados e suas característica (tipo de navio, hora esperada de chegada, carga), quantidade de berços e a especificação de cada um, bem como as condições de estoque e de maré.

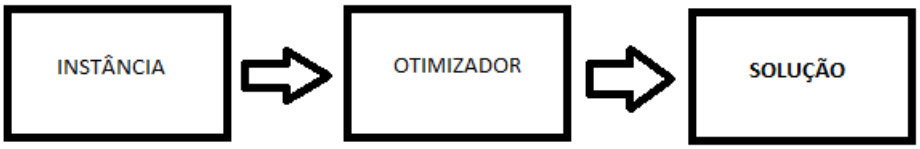

Figura 2.3: Processamento de uma instância de BAP

A proposta é realizar uma aplicação visual, que simule a dinâmica de entrada e

saída dos navios, carregamento e descarregamento, e a mudança nos níveis de estoque para cada matéria-prima, além das oscilações das marés ao longo do tempo. A aplicação deve ser amplamente configurável, de modo a permitir a experimentação de variadas condições de tráfego, estoque e capacidade de atendimento.

Por fim, salienta-se que o otimizador **não** faz parte do escopo do projeto. A aplicação apenas gera um arquivo-texto com as informações necessárias. No entanto, é um dos objetivos desse trabalho fornecer uma interface simples o suficiente para que em trabalhos futuros seja integrado a ela um programa otimizador. O cenário ideal é que, futuramente, o usuário execute a aplicação, escolhendo um algoritmo otimizador no próprio programa e possa observar na tela a sequência otimizada de atracação sendo realizada. E que, ao final da execução, a aplicação mostre dados estatísticos de desempenho (ver Subseção 3.2.4).

3 Embasamento Teórico

Neste capítulo, são abordados os principais conceitos e a terminologia relacionada a Simulação Computacional e Teoria de Filas, necessários à produção e ao entendimento deste trabalho.

3.1 Simulação Computacional

Simulação Computacional é o processo de criar e fazer experimentações com um modelo matemático computadorizado de um sistema físico[23]. Um programa simulador imita o comportamento de um sistema real, permitindo sua manipulação, e se constitui em uma valiosa técnica de estudo do sistema simulado.

Segundo Chung[23], o conceito de simulação abrange tanto a simulação tradicional quanto os simuladores. Os primeiros são programas de computador utilizados para analisar sistemas, visando uma melhor tomada de decisões operacionais e utilização de recursos. É o caso dos simuladores de filas em terminais de atendimento bancário, que reproduzem o ambiente aonde os clientes chegam para ser atendidos e ocasionalmente formam filas, devido à capacidade limitada de atendimento. Através da simulação, é possível experimentar diversos cenários, manipulando variáveis tais como quantidade de terminais, taxa de chegada de clientes no sistema e tempo de serviço, e, desse modo, equacionar a melhor configuração para atender o fluxo de clientes na agência.

Já os simuladores são programas de treinamento de usuários, onde estes aprendem a operar o sistema real, ou aprimoram o seu desempenho nas atividades exigidas por ele. Bons exemplos desse tipo de aplicação são os simuladores de vôo, que reproduzem com fidelidade o interior de um *cockpit* de avião e oferece aos futuros pilotos a oportunidade de conduzir um aeroplano de maneira virtual.

3.1.1 Classificação

Há diversas formas de se classificar um modelo de simulação. A seguir são apresentados os termos mais comuns encontrados na literatura, com uma breve explicação de cada um.

3.1.1.1 Discretos e Contínuos

Modelos discretos são aqueles em que as variáveis de estado são discretas, ou seja, podem assumir apenas um número finito ou infinito contável de valores, normalmente números inteiros. Por exemplo, em uma simulação de fila, uma variável discreta poderia ser o número de clientes aguardando atendimento.

Modelos contínuos são aqueles em que as variáveis de estado podem assumir infinitos valores em um intervalo de tempo. Um exemplo extraído de (Kelton, 2002) [4] é de um reservatório, em que o nível de água (variável de estado) oscila segundo o fluxo sai ou entra, e devido à evaporação e precipitação de água no reservatório.

Na maioria dos casos, no entanto, os modelos contêm tanto elementos contínuos quanto discretos e são classificados como sistemas mistos ou discreto-contínuos.

3.1.1.2 Estáticos e Dinâmicos

Um modelo estático é aquele em que o tempo não é uma variável considerada. O método de Monte Carlo (um tipo de simulação a ser discutido posteriormente) utiliza-se de modelos estáticos. Um exemplo desse tipo de modelo é a fórmula de transformação da matéria em energia $E = mc^2$[5].

Em um modelo dinâmico, o estado do sistema muda ao longo do tempo da simulação. Existe uma linha de tempo ao longo da qual acontecem os eventos que transformam o estado do sistema. Um exemplo de sistema cujo comportamento é dinâmico é de uma linha de montagem de uma fábrica de automóveis.

3.1.1.3 Determinísticos e Probabilísticos

Um modelo é determinístico quando não possui um componente aleatório que influa no seu comportamento, os mesmos parâmetros sempre conduzem ao mesmo

resultado.

Em um modelo probabilístico ou estocástico, por outro lado, há componentes randômicos no modelo. O sistema, nesse caso, pode apresentar diferentes resultados, se executado sob as mesmas condições. Um exemplo comum de variável randômica é a chegada aleatória de clientes em uma fila.

3.1.2 Tipos de Simulação

Os três tipos de simulação mais comumente encontrados na literatura são a Simulação de Monte Carlo, a Simulação Dirigida por *Traces* e a Simulação de Eventos Discretos. A seguir uma breve explanação de cada uma delas.

3.1.2.1 Simulação de Monte Carlo

Método de Monte Carlo é uma designação genérica para uma variedade de métodos alternativos utilizados para avaliar expressões matemáticas complexas, tal como integrais, cuja resolução pelos meios usuais é custosa ou impraticável. Esse método aplica conceitos estatísticos e probabilísticos, com a utilização de variáveis aleatórias e técnicas de amostragem, no estudo de problemas de natureza não-probabilística.

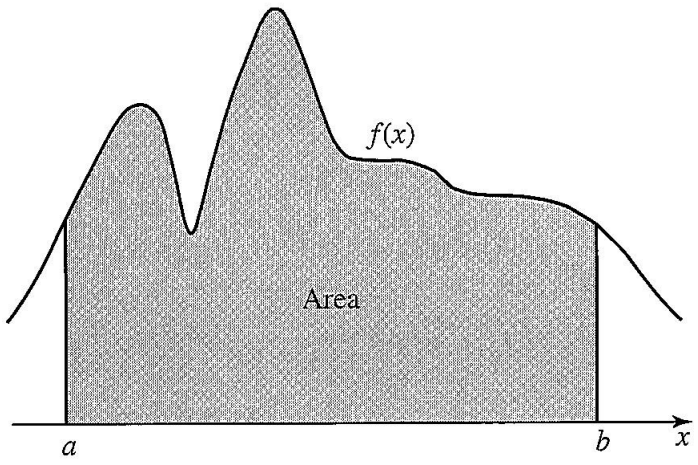

Figura 3.1: Integral definida no intervalo entre a e b. Extraído de Kelton[4].

Um exemplo de método de Monte Carlo, fornecido por Kelton[4], para estimar integrais baseia-se no fato de que a área sob o gráfico da função $f(x)$ no plano cartesiano limitada ao intervalo entre a e b equivale à integral definida $\int_a^b f(x)dx$ (Figura 3.1). O

método consiste em construir um retângulo de altura h e base igual a $b-a$, de modo que sua área $A = h(b-a)$ contenha todo o gráfico da função $f(x)$ (ver Figura 3.2)

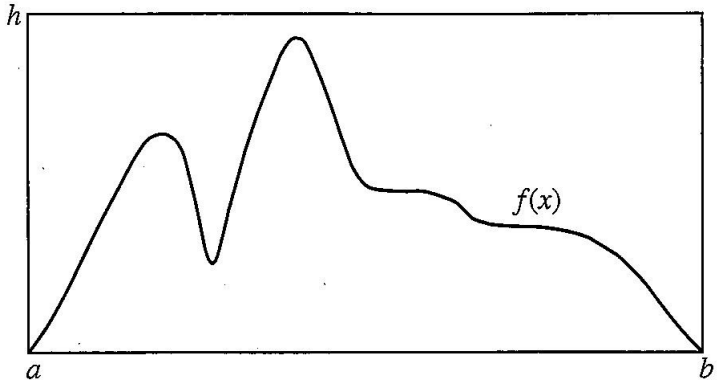

Figura 3.2: Método de Monte Carlo para cálculo de integrais. Extraído de Kelton[4].

Computa-se n pares de números aleatórios x_i e y_i, com $a \leq x_i \leq b$ e $0 \leq b \leq h$, contabiliza-se o número n_s de pontos (x_i, y_i) que estão sob a curva, e calcula-se a razão entre n_s e total de pontos (x_i, y_i) gerados. A razão obtida é uma estimativa da razão entre a área A e a integral de interesse. Logo, o valor estimado da integral definida $\int_a^b f(x)dx$ é dado por $F_n = A\frac{n_s}{n}$. onde n_s é número de pontos sob a curva e n é o número total de pontos gerados.

Métodos que seguem esses mesmos princípios são utilizados em diversas áreas da ciência e, no que tange à simulação, servem para modelar fenômenos probabilísticos que não mudam de estado ao longo do tempo[5].

3.1.2.2 Simulação Dirigida por *Traces*

Traces são registros de eventos de um sistema real, ordenados de acordo com o tempo[5]. Neste tipo de simulação — diferentemente do que acontece em outros casos, onde são utilizados geradores de números aleatórios para simular eventos do mundo real —, registros verdadeiros capturados durante a execução de um sistema são utilizados como entrada do processo de simulação.

3.1.2.3 Simulação de Eventos Discretos

Uma simulação de eventos discretos é um tipo de simulação em que se utiliza um modelo discreto para descrever o estado de um sistema[5]. Neste modelo, as variáveis

de estado que o representam mudam de valor instantaneamente em pontos separados no tempo. Nesses pontos, é que ocorrem os eventos, que, por sua vez, são definidos como ocorrências instantâneas que podem alterar o estado do sistema[24].

3.1.3 Componentes de uma simulação

3.1.3.1 Estado do sistema

Trata-se dos valores correspondentes às variáveis que descrevem o estado do sistema em um determinado momento. Por exemplo, no simulador de porto objeto desta monografia, o estado poderia ser descrito pelo número de navios no sistema, a altura da maré, a quantidade de cada produto em estoque e o número de navios sendo atendidos.

3.1.3.2 Entidades

A maior parte dos trabalhos envolvendo simulação envolve objetos denominados entidades, sejam clientes em uma fila, ou navios chegando ao porto, ou veículos em uma simulação de tráfego urbano. Entidades são os objetos dinâmicos de uma simulação, elas alteram o estado do sistema, afetam e são afetadas por outras entidades, possuem atributos que podem mudar de valor ao longo da simulação[4]. As entidades podem representar seres reais, como nos exemplos citados anteriormente, mas também podem ser objetos imaginários, criados para representar algo que afete o estado do sistema.

3.1.3.3 Atributos

As entidades possuem atributos que as distinguem umas das outras. Todas as entidades de um mesmo tipo possuem o mesmo conjunto de atributos, o que as individualizam são os valores de cada atributo. Por exemplo, no nosso simulador de porto, existe a entidade Navio, que possui, entre outros, os atributos nome e carga. À medida que são criadas, cada entidade Navio leva um nome único. Cada navio possui uma carga, criada através de um mecanismo randômico que seleciona os produtos e define a quantidade respectiva, de modo que um navio pode chegar ao porto repleto de óleo vegetal, enquanto outro leva soja e milho, e assim por diante.

3.1.3.4 Variáveis Globais

O conceito de variável global aqui é idêntico ao encontrado em linguagens de programação. Uma variável global possui escopo que "atravessa" todo o código. É uma variável que pode ser acessada em qualquer lugar e deve ser utilizada para conter informação de interesse geral. Em simulações, uma variável global típica é o tempo da simulação.

3.1.3.5 Recursos

Grande parte dos modelos de simulação dispõe de recursos que são utilizados pelas entidades. Tais recursos podem ser caixas automáticos de um banco, berços onde os navios atracam no porto (como é o caso da simulação apresentada neste trabalho), ou um microprocessador disputado por vários processos. Enfim, os recursos são os meios (escassos) pelos quais os serviços são oferecidos.

3.1.3.6 Acumuladores Estatísticos

Elementos deste gênero são variáveis que servem para manter o registro de determinadas dados estatísticos ao longo da simulação, os quais serão utilizados para gerar o relatório com as medidas de desempenho (fazer referência). No modelo de simulador de porto, alguns acumuladores estatísticos guardam as seguintes informações:

- Número de navios presentes no sistema até o momento
- Somatório do tempo de espera dos navios
- Somatório do tempo de serviço de cada navio
- Horas trabalhadas de cada berço

3.1.3.7 Eventos

Evento é algo que acontece em um instante de tempo (simulado) que pode mudar atributos, variáveis ou acumuladores estatísticos[4]. No simulador de porto, os principais eventos estão relacionados aos navios e são:

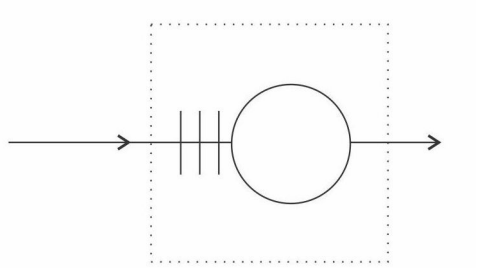

Figura 3.3: Modelo de sistema de filas simples

- NAVIO_CHEGOU – Esse evento ocorre quando o navio chega à área de espera do porto. O navio pode atracar de imediato, se houver disponibilidade de berço e condições de maré, ou esperar atendimento;
- NAVIO_ESPERANDO – O navio fica aguardando atendimento até que haja a possibilidade de atracar;
- NAVIO_ATRACOU – Quando o navio terminou de fazer sua manobra de atracação ao berço também é disparado um evento para que a operação de carga (ou descarga) se inicie;
- NAVIO_VAZIO – O navio já foi atendido e apenas aguarda a oportunidade para deixar o porto;
- NAVIO_FORA – Todo o ciclo de operação já foi concluído. O navio já foi atendido e deixou o porto.

3.1.3.8 Relógio (clock)

Este é um componente que existe nas simulações dinâmicas. O Relógio é uma variável que contém o valor atual do tempo de simulação.

Diferentemente de outros trabalhos, onde o tempo da simulação não flui continuamente, e é apenas atualizado à medida que ocorrem os eventos, no simulador aqui apresentado optou-se por utilizar um relógio contínuo, mais próximo da realidade.

O tempo de simulação é uma variável fundamental para o cálculo de medidas de desempenho (ver Subseção 3.2.4).

3.1.3.9 Filas

Filas são onipresentes em nosso dia-a-dia. Pessoas formam filas quando vão pagar as compras no supermercado; no cinema, ao comprar uma entrada; nos caixas dos bancos, nos hospitais, e em qualquer lugar onde haja a necessidade de distribuição de serviços ou produtos de maneira igualitária e ordenada. Em uma fila, a ordem de chegada define a ordem de atendimento, normalmente o primeiro a chegar será o primeiro a sair. Isso evita os conflitos que porventura possam surgir da competição entre os interessados.

As filas são necessárias ao convívio em sociedade, mas não são formadas apenas por pessoas. O mesmo conceito, com as devidas adaptações, serve para diversas áreas. Em computação, elas estão por toda parte. Pacotes de informação que trafegam pela internet formam filas ao serem enviados de roteador em roteador até chegar ao destino. Em um sistema operacional, os processos são armazenados em filas, enquanto aguardam processamento.

Como não poderia deixar de ser, filas são comuns também em sistemas simulados. Em geral, nos modelos de simulação, entidades esperam em fila até que sejam servidas ou processadas[23]. Normalmente, se utiliza um esquema FIFO (*First In First Out*, sou seja, o primeiro a chegar é o primeiro a sair) para estabelecer a prioridade entre as entidades.

Em razão de sua importância e da estreita relação com o tema desta monografia, na seção a seguir serão incluídos os principais conceitos da Teoria de Filas, um ramo da probabilidade encarregado do estudo das mesmas.

3.2 Teoria de Filas

3.2.1 Componentes de um Sistema de Filas

A Teoria de Filas estabelece uma série de atributos ou componentes que caracterizam um sistema de filas. Esses atributos devem ser conhecidos para que se possa realizar a análise do sistema. A seguir, uma breve descrição de cada um deles, utilizando como referência o trabalho de Jain[5]. Para ilustrar os conceitos, o autor usa como exemplo uma sala de computadores para estudantes. Quando não há computadores disponíveis, os estudantes (clientes, na terminologia utilizada em Teoria de Filas) esperam

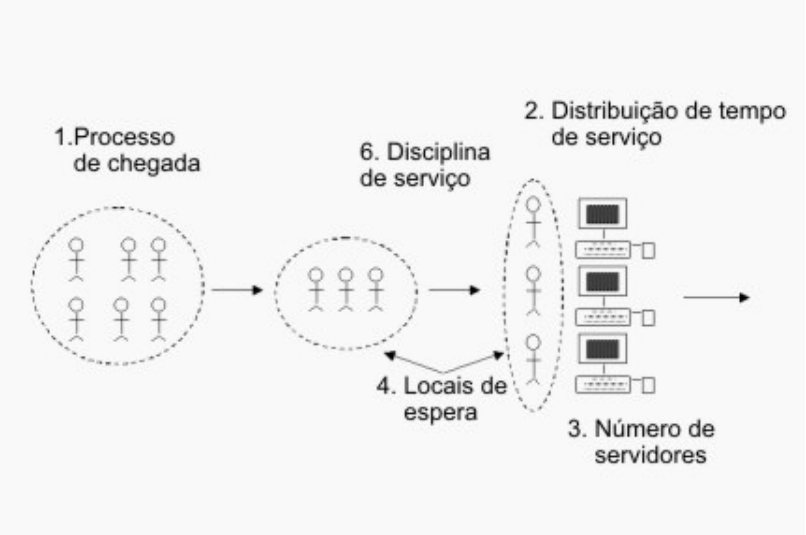

Figura 3.4: Componentes básicos de uma fila. Adaptado de Jain[5].

em fila até que uma das máquinas fique desocupada (Figura 3.4).

3.2.1.1 Processo de chegada

Para caracterizar um sistema de filas é necessário conhecer o modo como os clientes chegam ao sistema. Assume-se que os clientes chegam a intervalos de tempo aleatórios entre as chegadas. Ilustrando: os clientes chegam nos tempos $t_1, t_2, ..., t_n$ e o intervalo entre chegadas é dado por $X_1 = t_2 - t_1, ..., X_n = t_n - t_{n-1}$. Assim, as seqüências de valores $X_1, X_2...X_n$ são consideradas variáveis aleatórias independentes e identicamente distribuídas (IID). A distribuição de probabilidade mais comumente utilizada em processos de chegada é a de Poisson, o que implica que os tempos entre chegadas são distribuídos exponencialmente.

3.2.1.2 Distribuição de Tempo de Serviço

O tempo que cada entidade passa em um servidor é chamado de tempo de serviço. Comumente, esse tempo é considerado uma variável aleatória, tal qual o processo de chegada. Também neste caso são utilizadas distribuições independentes e identicamente distribuídas, sendo a mais comum a distribuição exponencial.

3.2.1.3 Número de Servidores

Um sistema de filas pode possuir um ou mais servidores. Servidor, para a Teoria de Filas, é onde o cliente é atendido, onde o serviço é prestado. No exemplo citado anteriormente, servidores são os computadores utilizados pelos estudantes.

3.2.1.4 Capacidade do Sistema

O número máximo de clientes que podem estar, ao mesmo tempo, no sistema define sua capacidade. Incluem-se, para o cálculo da capacidade, tanto os clientes em fila quanto aqueles em atendimento.

3.2.1.5 Tamanho da População

Tamanho da população é o número total de clientes em potencial que podem entrar no sistema.

3.2.1.6 Disciplina de Serviço

A ordem em que os clientes são atendidos chama-se disciplina de serviço. A mais comum delas é a *First Come, First Served* (FCFS), na qual o primeiro cliente a chegar será o primeiro a ser atendido. Outra disciplina comum é a *Last Come, First Served* (LCFS), ou seja, o último a chegar será atendido primeiro. Existem ainda várias outras possibilidades, mas para os objetivos desta monografia, estas duas bastam.

3.2.2 Notação de Kendall

A notação de Kendall é uma maneira padrão de se especificar um sistema de filas. Utiliza-se a forma A/S/m/B/K/SD, em que cada letra corresponde a um dos atributos elencados anteriormente, conforme quadro abaixo:

A/S/m/B/K/SD

- A: Processo de chegada;
- S: Distribuição de tempo de serviço;
- M: Número de servidores;
- B: Capacidade do sistema;
- K: Tamanho da população;
- SD: Disciplina de serviço.

As distribuições de probabilidade utilizadas no processo de chegada (A) e na distribuição de tempo de serviço (S) são representadas por letras de acordo com o seguinte quadro:

M: Exponencial
Ek: Erlang com parâmetro k
Hk: Hyperexponential com parâmetro k
D: Determinística
G: Geral

3.2.3 Lei de Little

Existe uma fórmula, conhecida como Lei de Little, que relaciona o número médio de clientes em fila e o tempo médio de espera no sistema, e é muito utilizada em análise de sistemas de filas. A Lei de Little enuncia que, em condições estáveis, o número médio de clientes em fila (L) é igual ao tempo médio de espera no sistema (W) multiplicado pela taxa de chegada média de clientes (γ), ou seja,

$$L = \gamma W \tag{3.1}$$

3.2.4 Medidas de Desempenho

Durante a análise de um sistema simulado, é importante avaliar o desempenho do mesmo sob variadas condições. Para isso existem métricas, que, uma vez calculadas, fornecem essa informação de maneira precisa. De acordo com Kelton et al [4], as medidas

de desempenho mais comuns são:

- Tempo de sistema
- Tempo em fila
- Número médio de clientes em fila
- Utilização

3.2.4.1 Tempo de sistema

Tempo de sistema é o tempo total que uma entidade passa dentro do sistema. Se inicia quando a entidade chega no sistema e entra na fila, e termina quando a entidade sai do sistema, após completar seu tempo de serviço[23].

Esta medida de desempenho é calculada através da seguinte equação:

$$Tempo\,Médio\,de\,Sistema = \frac{\sum_{i=1}^{n} T_i}{n}, \tag{3.2}$$

onde T_i é o tempo de sistema para cada entidade, e n é o número total de entidades processadas.

3.2.4.2 Tempo em fila

Esta é uma medida semelhante ao tempo de sistema, com a diferença que, neste caso, apenas o tempo em fila é considerado. Observou-se que, pelo menos nos casos de entidades humanas, o tempo gasto aguardando em fila é mais custoso, do ponto de vista do cliente, do que o tempo de serviço, daí a importância de se otimizar o tempo em fila. A equação para o Tempo Médio em Fila é:

$$Tempo\,Médio\,em\,Fila = \frac{\sum_{i=1}^{n} D_i}{n}, \tag{3.3}$$

em que D_i é o tempo em fila para cada entidade individualmente, e n é o número de entidades processadas.

3.2.4.3 Número médio de clientes em fila

De acordo com Chung[23], essa é uma medida que depende do tempo (mas não da quantidade de entidades processadas), e indica o número médio esperado de clientes na fila, durante um período de tempo estipulado. A fórmula para o cálculo do número médio de clientes em fila é

$$Número\,Médio\,de\,Clientes\,em\,Fila = \frac{\int_0^T Q dt}{T}, \tag{3.4}$$

onde

Q = número de clients em fila durante um período de tempo dado.

dt = interval de tempo em que Q é observado

T = tempo total da simulação.

3.2.4.4 Utilização

A Utilização é um dado estatístico que, assim como o do item anterior, é dependente do tempo. Trata-se de calcular o quão ociosos ou não estão os recursos durante um dado período de tempo. Matematicamente, um recurso ocioso possui nível de utilização igual a 0, enquanto um recurso ocupado possui nível igual 1. A fórmula que calcula a utilização média durante um período de tempo específico é

$$Utilização\,Média = \frac{\int_0^T B dt}{T}, \tag{3.5}$$

onde

$B = 0$ para ocioso, 1 para ocupado,

dt = duração de tempo em que B é observado

T = tempo total da simulação.

4 Simulador de Terminal Portuário

Neste capítulo, serão apresentados os detalhes mais relevantes da construção da aplicação. Começando com a Análise de Requisitos, passando pelo Projeto, Implementação e, finalmente, mostrando a aplicação construída.

4.1 Análise e Especificação de Requisitos

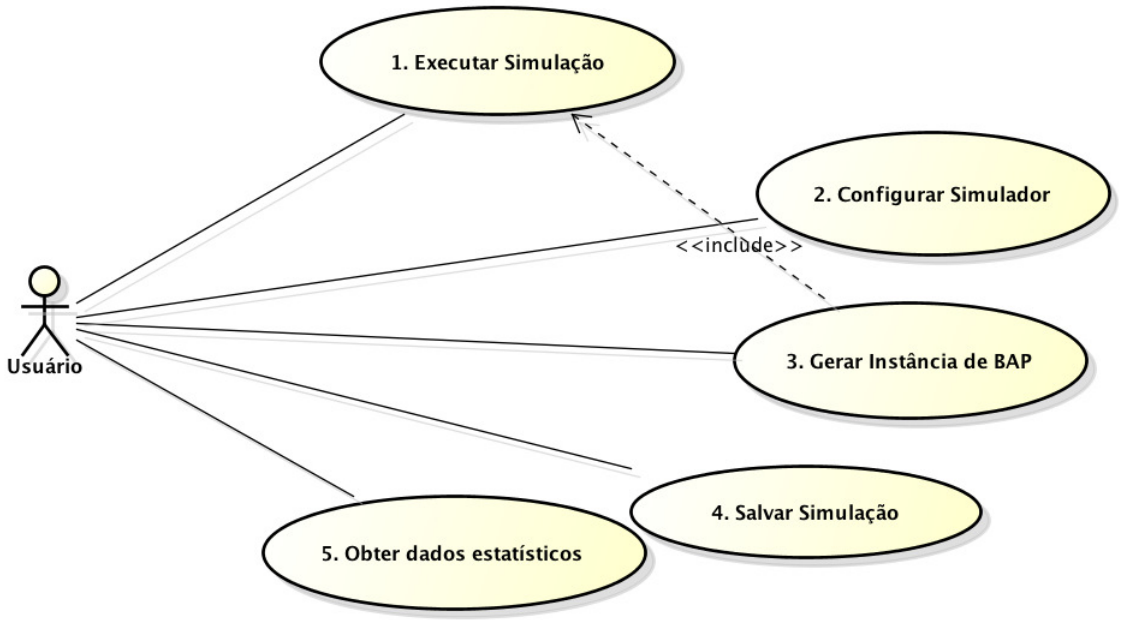

Figura 4.1: Diagrama de Casos de Uso

4.1.1 Requisitos

No Capítulo anterior já se tratou superficialmente dos requisitos da aplicação. Nesta seção serão definidos os requisitos em mais detalhes.

A aplicação desenvolvida é um Simulador de Porto, cujo objetivo principal é gerar instâncias para o Problema de Alocação de Berços. Como ilustrado na Figura 4.1, tem-se cinco casos de uso que orientaram todo o processo de desenvolvimento:

1. Executar Simulação - O usuário dá início à ação no simulador. A aplicação mostra

um modelo de terminal portuário em movimento, com os navios chegando ao porto, e aguardando condições propícias para serem atendidos nos berços. A aplicação mostra as variações do estoque, refletindo as operações de carregamento e de descarregamento dos navios. A aplicação ainda mostra as mudanças do nível de maré ao longo do tempo. Sendo que este (o tempo) é marcado em um relógio;

2. Configurar Simulador - O usuário tem condições de configurar vários aspectos do simulador: processo de chegada dos navios, número de berços, peso e natureza da carga, etc;

3. Gerar Instância de BAP - Após executar o passo 1, o usuário pode pausar a simulação a qualquer momento e gerar a instância. Esta se trata de um documento de texto em algum formato padronizado, que poderá ser processado depois em algum otimizador;

4. Salvar simulação - O usuário pode interromper a simulação, salvar o estado atual em um arquivo binário e, posteriormente, pode retomar o processo de onde estava. O arquivo referido possui toda informação sobre o estado da simulação no momento da interrupção;

5. Obter dados estatísticos - Após finalizar a simulação, o usuário pode obter os dados sobre o desempenho da simulação, tais como throughput, utilização (dos berços), média total do *demurrage*, etc.

4.1.2 Modelo

O modelo de simulação empregado neste trabalho é de eventos discretos, com tempo contínuo. Como é típico neste tipo de simulação, tem-se entidades (navios) que buscam atendimento nos servidores (berços) e, eventualmente, formam filas, devido a indisponibilidade destes. Neste caso específico, os navios formam uma única fila e, além da indisponibilidade de berços, o nível de maré condiciona o atendimento, já que os navios precisam de uma altura de água suficiente para atracar e sair dos berços.

Trata-se de um modelo multiservidor e, pela notação de Kendall, pode ser definido como: G/U/1-5/∞/∞/FCFS, o que significa que: a)a distribuição de probalidade para o processo de chegada é genérica (na realidade, trabalhou-se com as distribuições Normal, Exponencial e Poisson); b) a distribuição do tempo de serviço

é Uniforme, já que depende da capacidade dos berços e da quantidade de carga em navio, e esta é atribuída aleatoriamente seguindo a distribuição Uniforme; c) o número de servidores varia de 1 a 5; d) a capacidade do sistema e o tamanho da população são infinitos e f)a disciplina de serviço é a FCFS.

Foram inúmeras as simplificações feitas no modelo para que ele fosse compatível àquele empregado por Barros et al.[21] e, ao mesmo tempo, viável em termos de programação. As suposições feitas estão listadas a seguir:

- Todos os navios são idênticos, no tocante à velocidade, tamanho, capacidade de carga e quantidade de produtos possíveis em sua carga (embora a quantidade de cada produto varie aleatoriamente).
- A capacidade dos berços pode ser configurada, podendo ser distintas (berços heterogêneos) ou não (homogêneos);
- O número n de berços é uma variável que pode ser configurada em tempo de execução, sendo que, por restrições de espaço na tela, n possui o valor máximo de cinco.
- A maré possui ciclos exatos de 12 horas.
- O atendimento segue a disciplina de serviço FCFS.

Graficamente, enfatizou-se o caráter meramente ilustrativo do modelo. As entidades são representadas por figuras básicas, como retângulos e círculos. Os movimentos na tela também foram simplificados ao máximo. Os navios, que são as únicas entidades que se locomovem, possuem apenas três movimentos: o primeiro — retilíneo — em direção à área de espera, o segundo segue uma curva em direção ao berço, e o terceiro — o movimento de saída —, também segue curva semelhante (Figura 4.2). Ao chegar na área de espera, os navios ficam alinhados lado a lado ao longo de uma linha transversal que demarca essa área. Não foi implementado algoritmo detector de colisões, o que implica que os navios podem se sobrepor uns aos outros. Neste ponto, a situação se assemelha à de veículos aguardando em um semáforo, com a diferença que o comando *siga* é válido para apenas aqueles indivíduos com permissão para atracar.

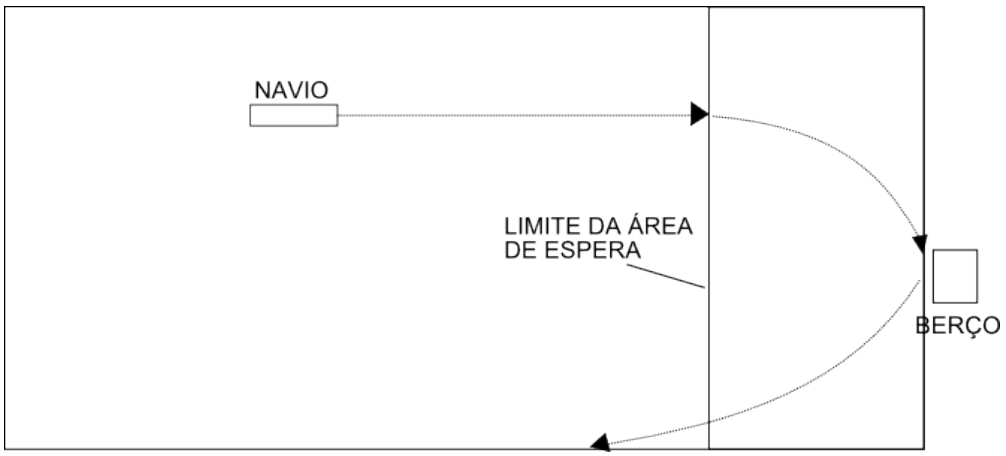

Figura 4.2: Movimentos dos navios na tela

4.2 Projeto

Um simulador, como o proposto neste trabalho, é uma aplicação semelhante a um jogo de computador. Em vista disso, foram utilizadas técnicas de programação comuns a esse tipo de aplicação. Em Davison[25], o autor apresenta um *framework* de animação utilizado como base para jogos. Esse algoritmo foi adaptado e aplicado neste projeto.

O diagrama de atividades da Figura 4.3 mostra o algoritmo adaptado. No primeiro passo (Iniciar), a interface é criada e desenhada na tela. É essa tela que o usuário verá antes de iniciar a simulação. O usuário, ao clicar um botão, envia um sinal que altera variáveis de estado. Se esse sinal for para iniciar/retomar a simulação, o algoritmo entra em um laço, que atualiza os objetos (navios, berços, mar, etc) continuamente, e os desenha na tela. Dentro do laço, há um intervalo de tempo em que aplicação fica em suspenso, que neste caso é 20 ms. Isso é necessário porque evita o consumo excessivo de recursos da CPU. Outra necessidade, é realizar o desenho dos objetos em dois passos: primeiro a imagem é "renderizada", o que significa desenhar os objetos em um *buffer*, e só depois essa mesma imagem é passada para a tela. Segundo Davison[25], isso evita o *flicker*, um efeito indesejado de tremulação na imagem. O laço é executado até que haja um sinal para interromper a simulação.

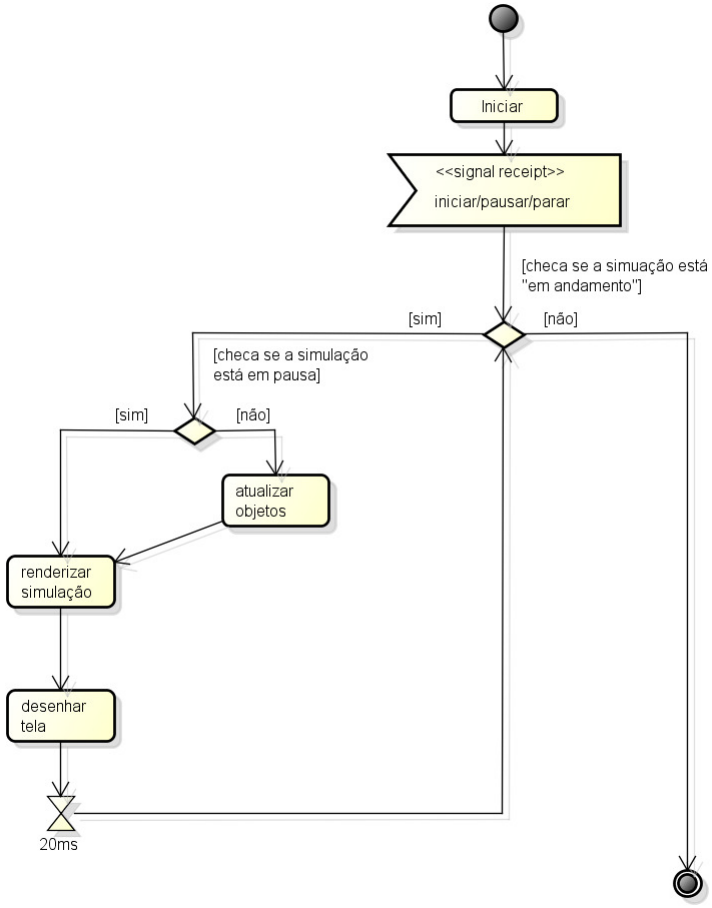

Figura 4.3: Framework de animação

A atualização dos objetos, referida no parágrafo anterior, é o que proporciona o efeito de animação. Nos objetos móveis, como os Navios, essa atualização normalmente significa mudança de posição. Nos objetos fixos, essa atualização afeta outros aspectos, dependendo do objeto. A Figura 4.4 ilustra a hierarquia de classes dos objetos visíveis da simulação. Os objetos da classe ObjetoMovel são aqueles que atravessam a tela com determinada velocidade (atributo dx) e direção (atributo aTan). Os objetos da classe ObjetoFixo, como o nome sugere, permanecem na mesma posição, embora possam ter partes móveis incluídas. Cumpre observar que cada objeto possui o seu método “desenhar” e “atualizar”, e que este último toma como argumento o intervalo de tempo decorrido desde a última atualização.

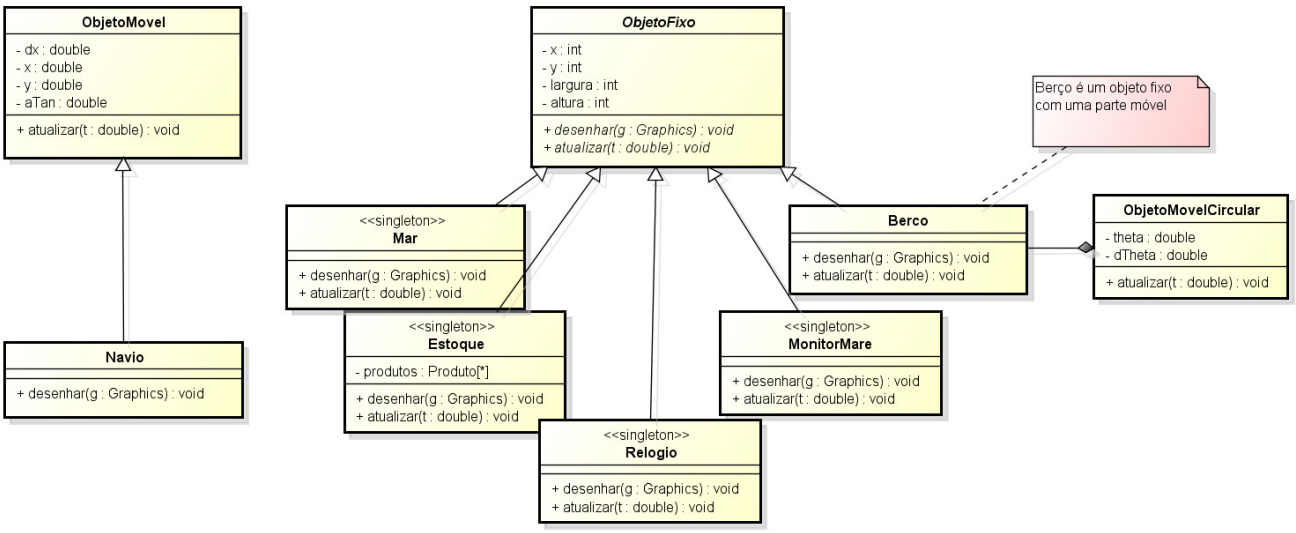

Figura 4.4: Objetos Móveis e Fixos

A classe Navio é uma das mais complexas e importantes do projeto. Por isso, aqui vão mais alguns detalhes dessa classe. Os objetos Navio são criados a cada laço do framework explicado anteriormente (ver Figura 4.3). A cada execução desse laço, a sequência de ações ilustrada abaixo (Figura 4.5) acontece: o objeto pnlAnimation (um objeto da classe Java JPanel que implementa o framework de animação ilustrado na Figura 4.3) solicita a GeradorNavio uma lista de novos objetos Navio; este recorre a um objeto RandomNumberGenerator para determinar aleatoriamente a quantidade a ser gerada; de posse do número, GeradorNavio solicita que ConstrutorNavio construa objetos Navio; após construídos, essa lista é repassada e os novos objetos Navio são incluídos na lista total de navios da simulação. As classes GeradorNavio e ConstrutorNavio possuem uma relação de agregação, como mostra o diagrama de classes da Figura 4.6.

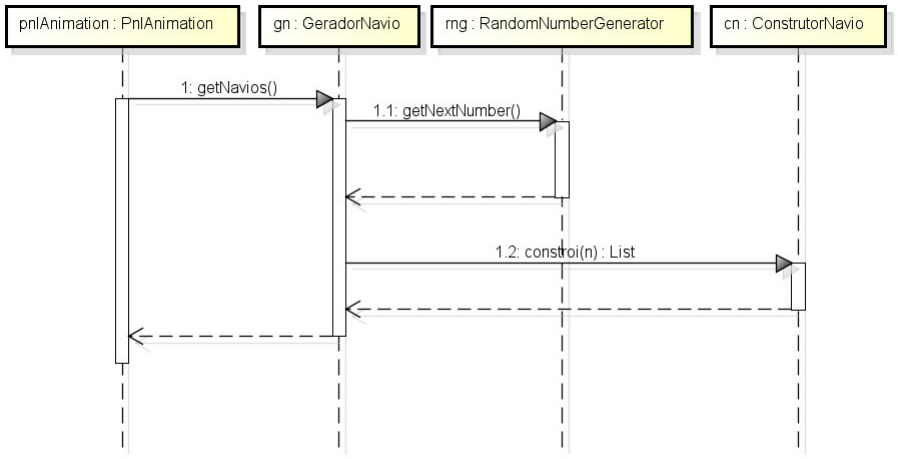

Figura 4.5: Diagrama de sequência ilustrando como os navios são criados

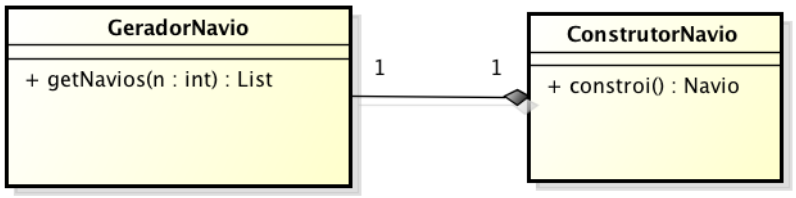

Figura 4.6: Relação de agregação entre as classes GeradorNavio e ConstrutorNavio

Após serem criados, os objetos Navio seguem seu ciclo de vida dentro da aplicação. Outro objeto, do tipo AdministradorPorto, comanda cada Navio durante esse ciclo, e por isso é necessário haver uma comunicação contínua entre ambos. Essa é razão pela qual utilizou-se o padrão de projeto *Observer* na arquitetura dessas classes (Figura 4.7). A cada mudança de estado relevante, o objeto Navio comunica AdministradorPorto através de eventos. Com base no evento lançado, no estado da maré e da disponibilidade dos berços, AdministradorPorto decide que ação deve ser tomada e envia uma mensagem de volta ao Navio (atracar, sair, esperar, etc.). Note-se no diagrama abaixo que AdministradorPorto é uma classe abstrata, logo é necessário haver uma subclasse que implemente o método "administrar". Quem faz este papel é a classe AdministradorPortoFCFS, que implementa a disciplina de serviço FCFS. Esse detalhe de arquitetura permite que o código possa ser extendido através de polimorfismo, incorporando diferentes versões do método "administrar".

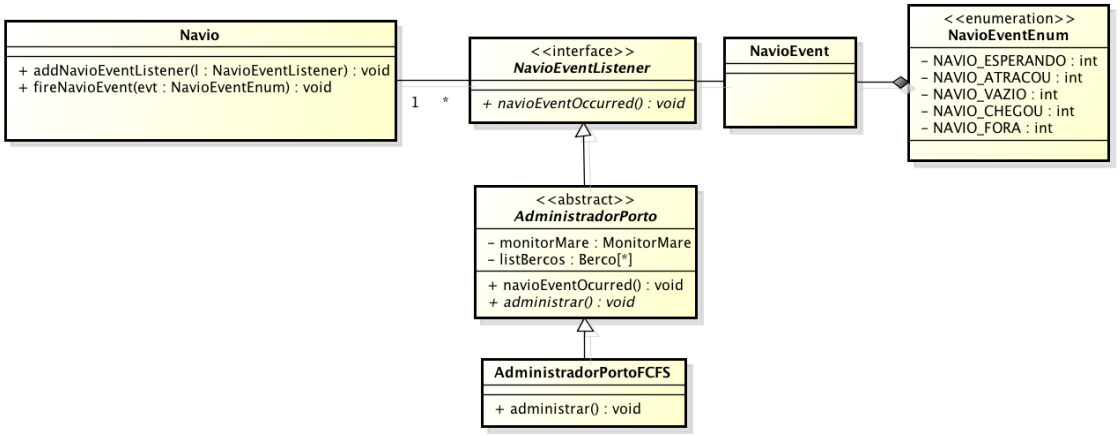

Figura 4.7: Relações entre a classe Navio e AdministradorPorto

Por fim, é necessária uma explicação sobre o processo que gera a instância de BAP, o objetivo principal deste projeto. Há uma classe chamada Porto que contém nos seus atributos todos os objetos visíveis e não-visíveis que fazem parte da simulação

(Figura 4.8). Essa classse é convertida em arquivo texto através do processo ilustrado no diagrama de sequência abaixo (Figura 4.9). Fazem parte do processo três classes de objeto: Porto, PortoConverter e InstanciaSerializer. O primeiro contém a instância, o segundo a converte em uma *string* de texto e o último faz a persistência do arquivo no diretório do usuário.

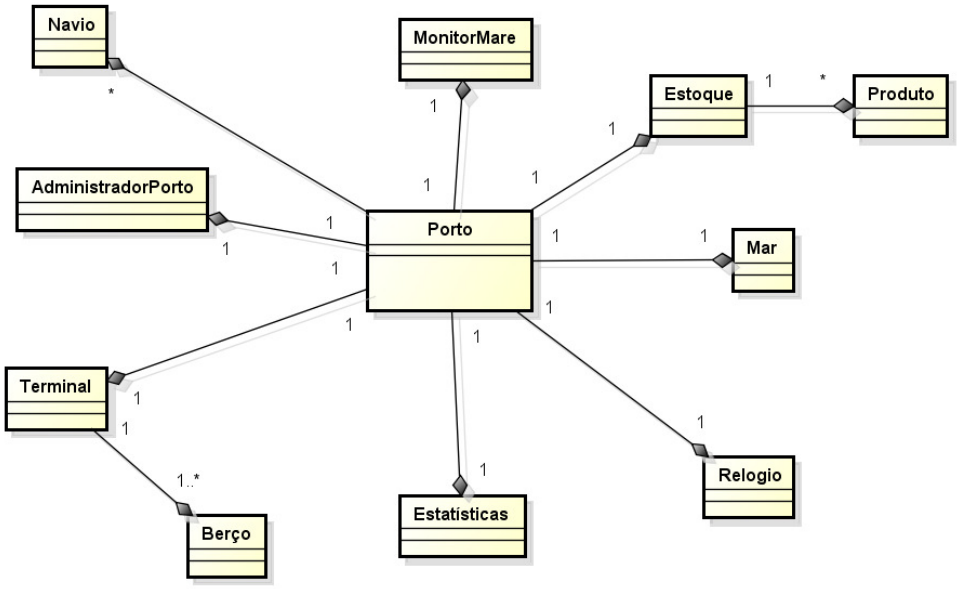

Figura 4.8: A classe Porto

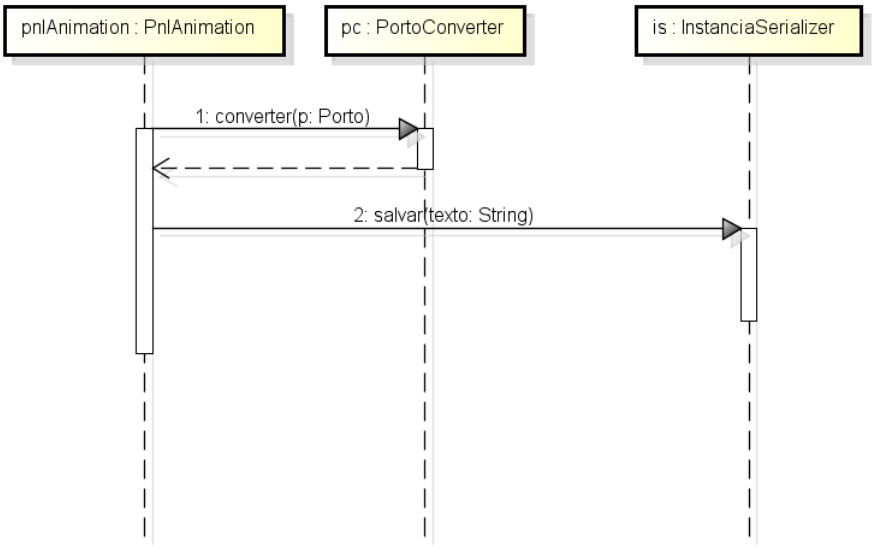

Figura 4.9: Gerando uma instância de BAP

4.3 Implementação

O simulador foi implementado usando a linguagem Java. As seguintes características fizeram dessa linguagem a escolha ideal:

- é uma linguagem orientada a objetos, o que permite projetar a aplicação com foco no reúso e extensão;
- disponibilidade de ambientes de desenvolvimento gratuitos e de qualidade, como o Netbeans;
- permite a criação de aplicações *web*, como os *applets*, que podem ser executados no contexto de um navegador de *internet*;
- farta documentação disponível na *internet*;
- disponibilidade de bibliotecas gráficas sofisticadas, como a Java2D, que supriu as necessidades da animação.

Além das bibliotecas distribuídas juntamente com o *Java Development Kit* (JDK), o conjunto de ferramentas padrão fornecido para o desenvolvimento de aplicações, foram utilizadas bibliotecas de terceiros: a *Jfreechart* (`http://www.jfree.org/jfreechart/`) e a *Apache Commons Math* (`http://commons.apache.org/proper/commons-math/`).

Jfreechart é uma biblioteca 100% Java que provê recursos para apresentação de gráficos em aplicações nessa linguagem. Esta biblioteca foi utilizada unicamente para gerar um gráfico ilustrando as mudanças no estoque ao longo da simulação.

Apache Commons Math é uma biblioteca que contém funções matemáticas e estatísticas e foi utilizada para gerar os números aleatórios necessários à simulação.

4.4 A aplicação

A aplicação final trata-se de um *applet* java que pode ser embutido em qualquer página html. A tela inicial do simulador está na Figura 4.10. A interface possui quatro abas: Simulação, Configurar, Estatísticas e Arquivos. A primeira aba, mostrada na Figura 4.10, contém a simulação propriamente dita, com os objetos que a compõem: o relógio,

no canto superior esquerdo; o mar (o retângulo preenchido em azul); o Monitor de Maré, que indica o nível da mesma e sinaliza quando as condições permitem atracação; os berços e o painel de estoque. Mais abaixo há três botões de controle, que servem para pausar, iniciar/retomar e parar a simulação. No canto inferior direito, mais outro botão, que serve para gerar a instância do BAP.

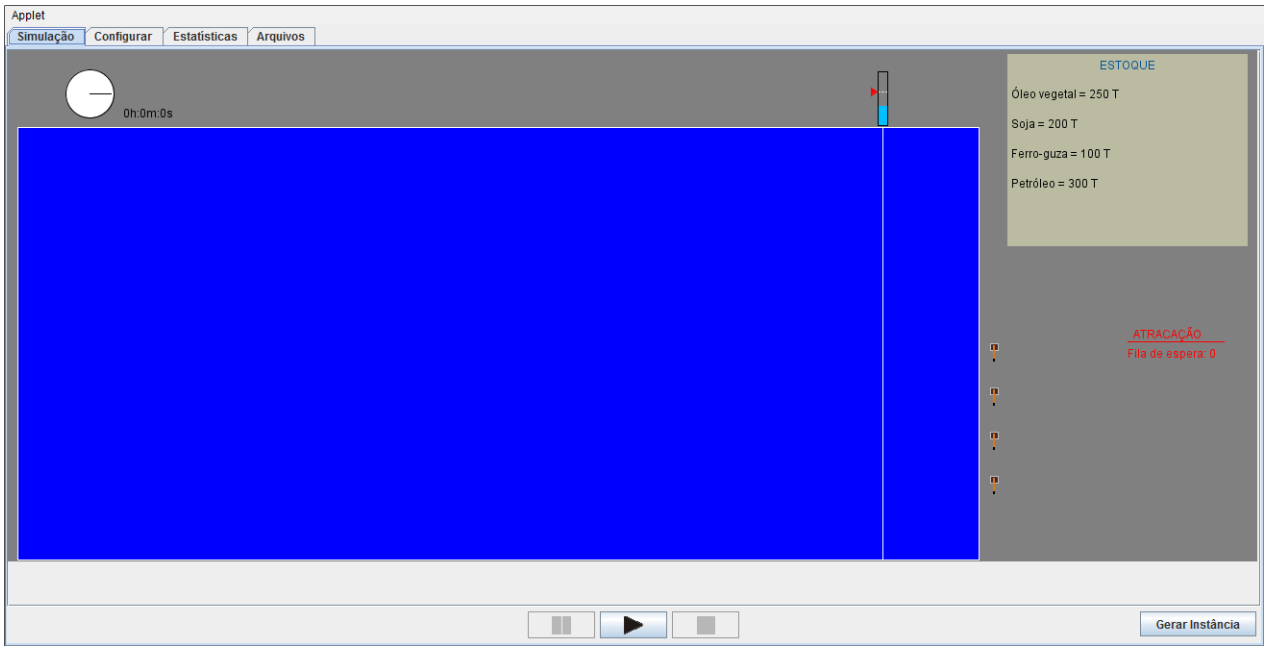

Figura 4.10: Tela inicial

4.4.1 Executando a Simulação

O usuário dá início à simulação clicando no botão  . A partir daí, surgem navios no horizonte se aproximando a determinada velocidade. Esse navios chegam até a área de espera, demarcada com uma linha branca vertical sob o Monitor de Maré (Figura 4.11), e permanecem lá até que o nível de maré seja adequado (o Monitor de Maré indica isso) e que haja um berço disponível. Caso essas duas condições sejam satisfeitas, o navio atraca em um berço e começa o atendimento, obedecendo a disciplina de serviço FCFS. À medida que os navios são carregados e/ou descarregados, as mudanças no estoque são refletidas no Painel de Estoque.

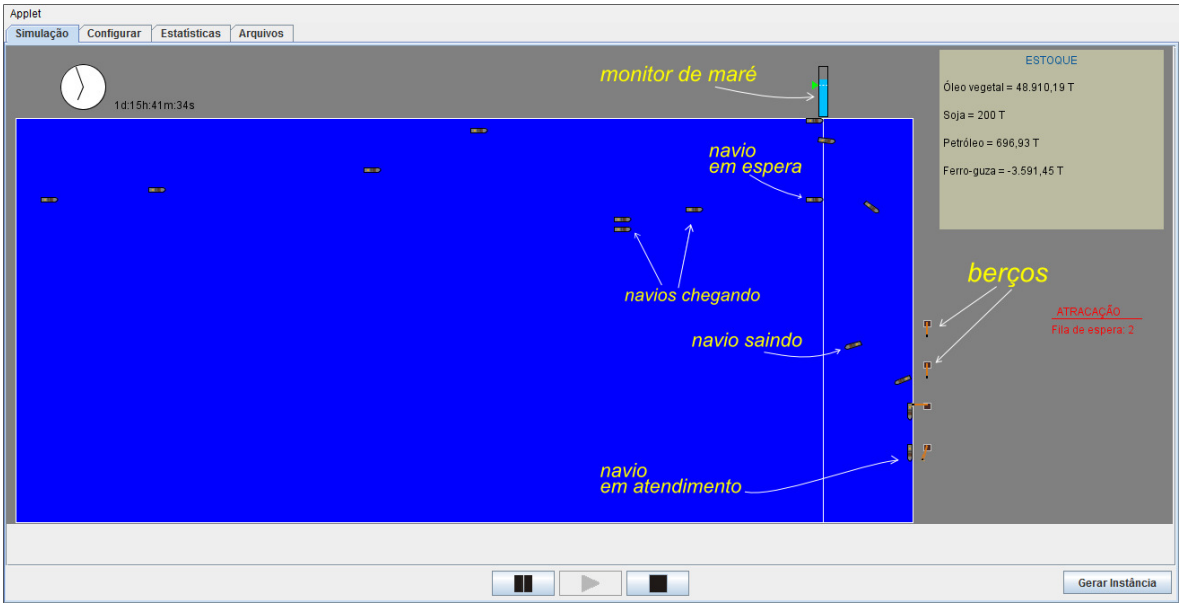

Figura 4.11: Simulação em andamento

4.4.2 Configurando a Simulação

O simulador foi construído para ser amplamente configurável. A aba "Configurar" dá acesso a uma série de opções, organizadas em abas internas.

Na aba "Geral" (Figura 4.12), o usuário tem a possibilidade de estabelecer a relação tempo real/tempo simulado, inserindo um valor equivalente para 1 hora real (na Figura, 1 hora real equivale a 1000 ms no simulador). O usuário pode fazer algo semelhante em relação ao espaço, estabelecendo a relação entre quilômetro real/pixels na tela. Por último, o usuário pode configurar a extensão do percurso, ou seja, a distância que os navios devem percorrer até o berço (em km).

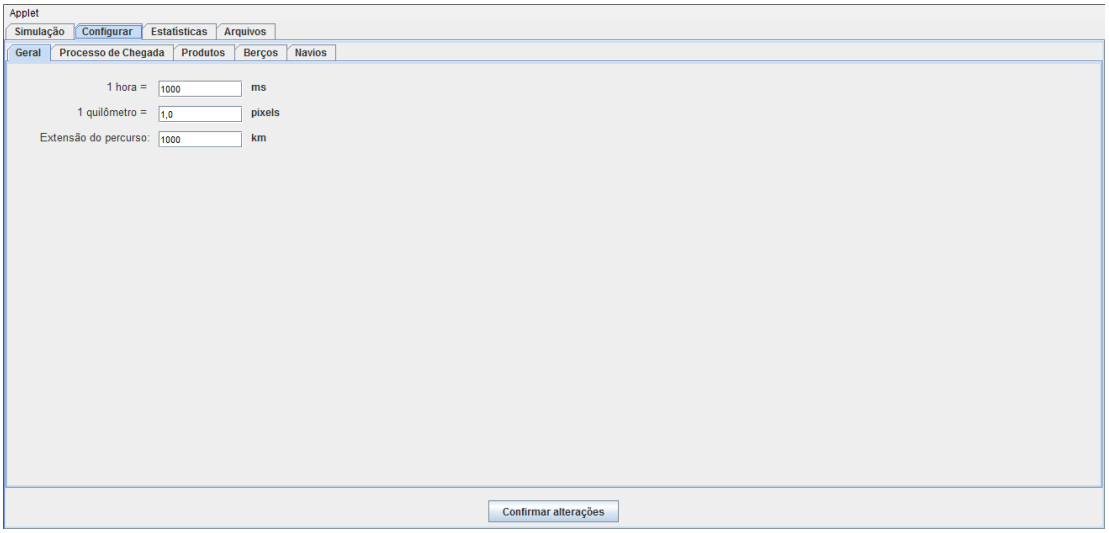

Figura 4.12: Tela de configurações gerais

Na aba "Processo de Chegada" (Figura 4.13), é possivel configurar a distribuição de probabilidade que regula as chegadas dos navios. O simulador dispõe de 3 opções de distribuição: Normal, Exponencial e Poisson. Os parâmetros de cada uma devem ser inseridos também.

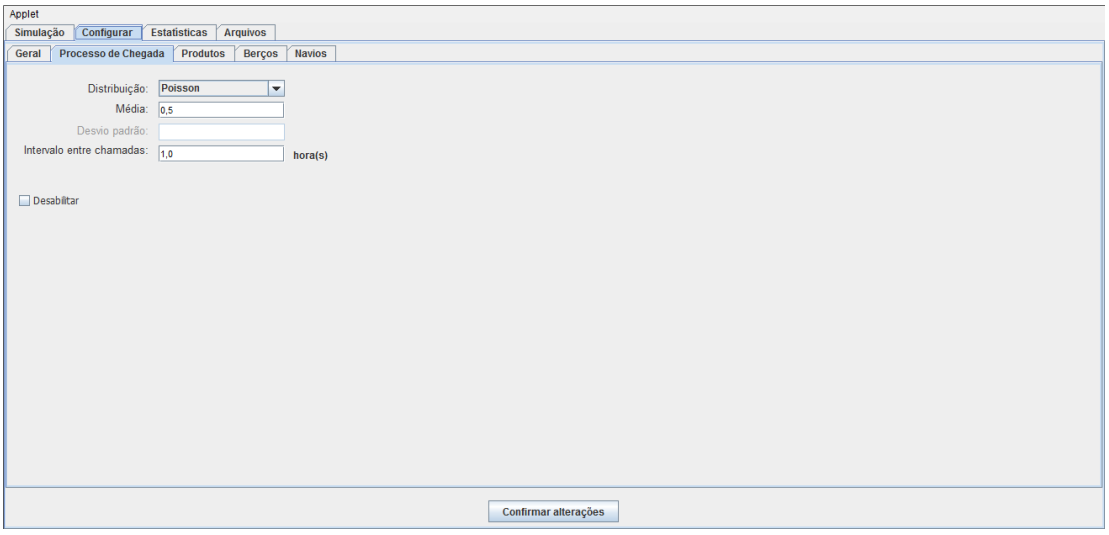

Figura 4.13: Tela de configuração do processo de chegada dos navios

O usuário pode determinar quais os produtos podem compor a carga dos navios, bem como o estado inicial do estoque no porto. Na aba "Produtos" (Figura 4.14) o usuário pode inserir os nomes dos produtos, o estoque inicial, consumo e produção. O estoque não varia apenas conforme o carregamento e descarregamento dos navios, mas também devido à produção e ao consumo, que acontece por terra. O consumo é a taxa em que o produto é retirado do estoque no porto e a produção é o inverso.

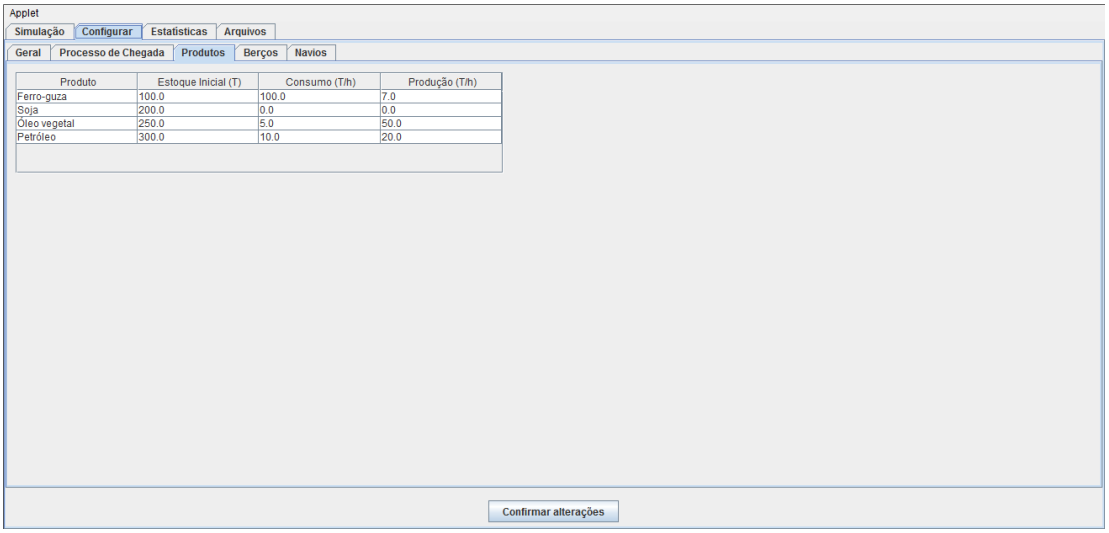

Figura 4.14: Tela de configuração dos produtos

Na aba "Berços" (Figura 4.15), o usuário define a quantidade de berços e a capacidade de cada um.

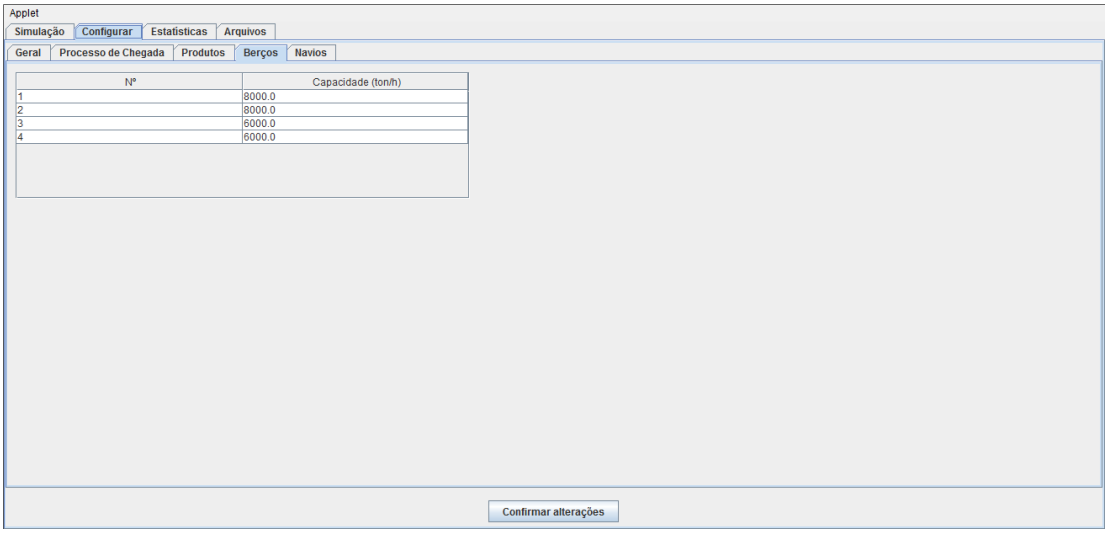

Figura 4.15: Tela de configuração dos berços

Por último, o usuário pode influir na maneira como os navios são construídos na simulação pela aba "Navios" (Figura 4.16). As três primeiras opções valem para todos os navios gerados: velocidade, capacidade de carga e quantidade de produtos na carga. O *demurrage* possui um valor mínimo e máximo (nesse caso, o valor será definido aleatoriamente pelo simulador, dentro desse intervalo). Em "tipo de carga", o usuário define se os navios serão descarregados(importação), carregados(exportação) ou ambos, ou deixa o simulador decidir (aleatório). As opções em "Atracação" indicam se é permitido ou não que um navio faça a atracação enquanto outro está saindo do mesmo berço (entrada

e saída simultânea).

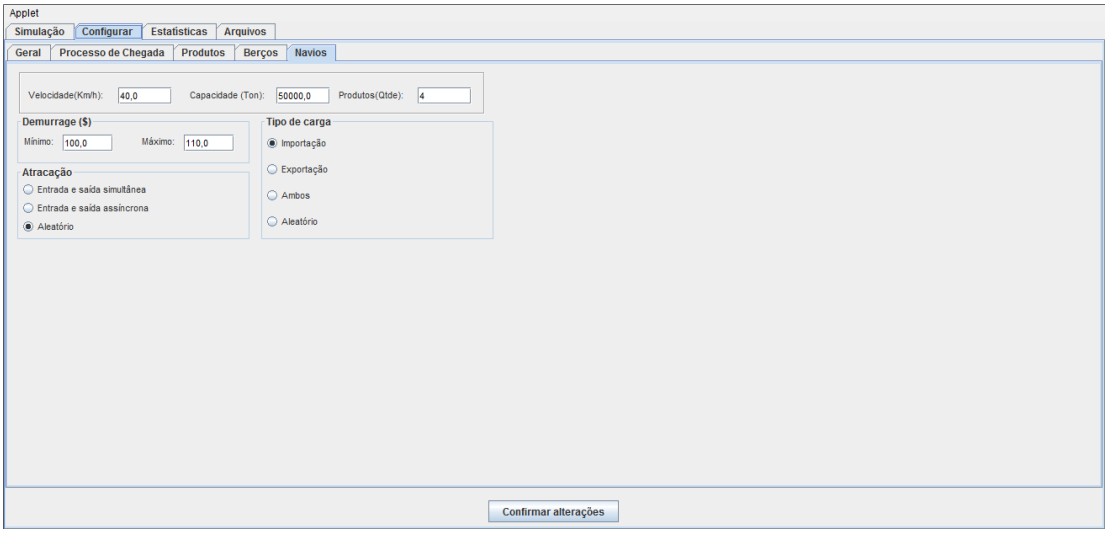

Figura 4.16: Tela de configuração dos navios

4.4.3 Obtendo dados estatísticos

Na aba "Estatísticas" é possivel obter todos os dados estatítiscos de desempenho colhidos durante a simulação. Para isso o usuário necessita apenas clicar no botão "Gerar" na parte inferior da tela.

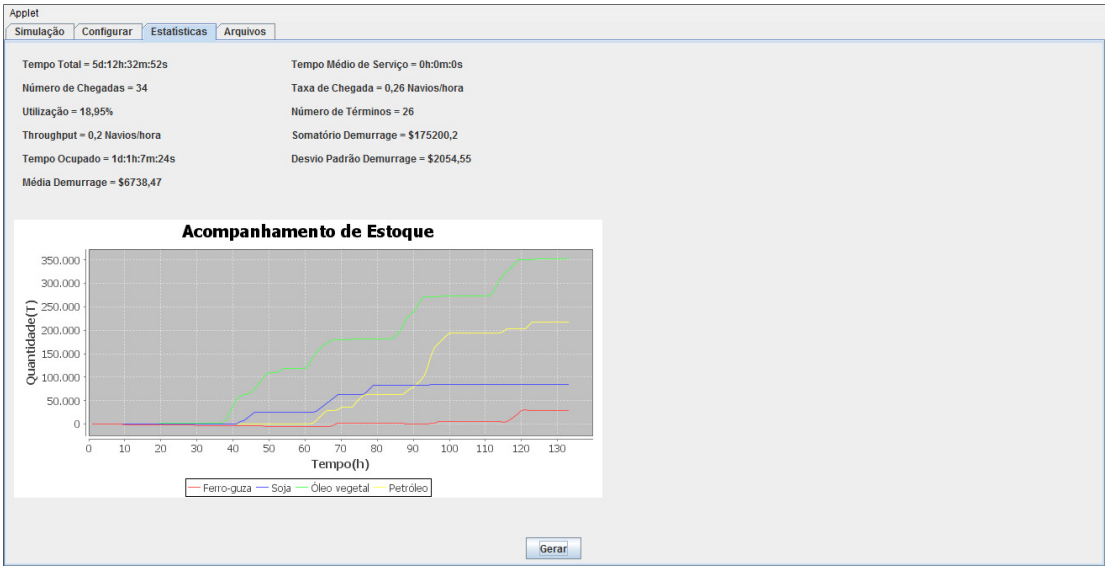

Figura 4.17: Tela de estatísticas

4.4.4 Salvando a simulação

A aplicação possui a função de salvar a simulação em andamento em um arquivo. Isso é feito através da aba “Arquivos” (Figura 4.18). Por padrão, o simulador utiliza o diretório do usuário para estas operações, e os arquivos possuem a extensão “.por”. É possível também abrir todos os arquivos salvos previamente, e retomar a simulação de onde foi interrompida.

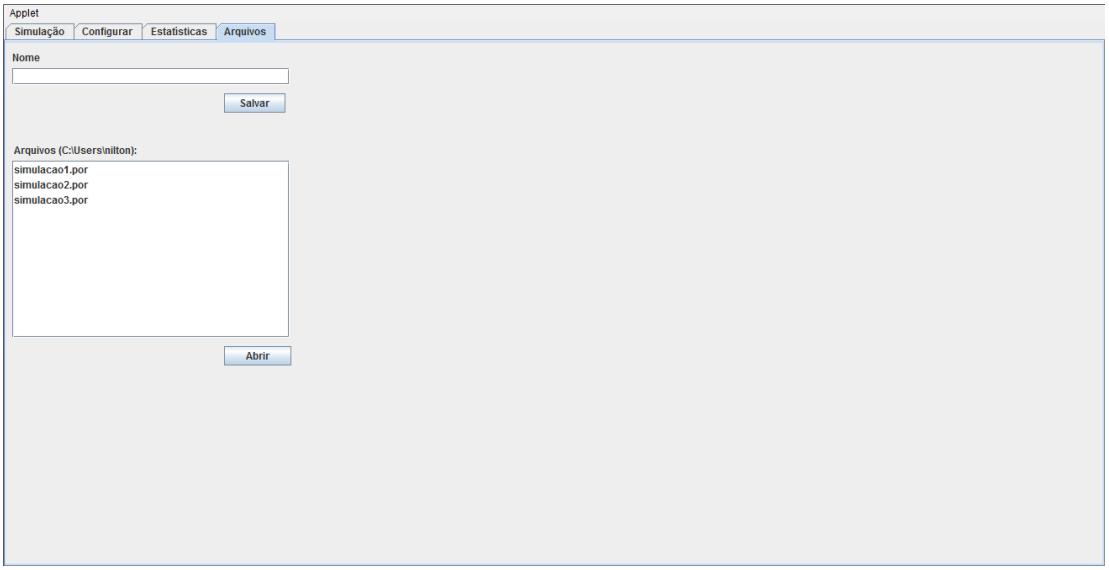

Figura 4.18: Tela de arquivos

4.4.5 Gerando instâncias de BAP

Com a simulação em andamento, o usuário decidir a qualquer momento gerar uma instância de BAP. Para isso é necessário apenas clicar no botão “Gerar instância”. Em alguns segundos, o simulador toma a o estado atual da simulação e converte-a em um arquivo-texto semelhante ao da Figura abaixo (Figura 4.19). Esse arquivo é salvo no diretório do usuário, com a extensão .dat.

```
/*********************************************
* OPL 12.3 Data
* Author: SimulaPorto
* Creation Date: 4/12/2013 at 12:31:23
*********************************************/

Berths = {
 <1,"b0",8.0>,
 <2,"b1",8.0>,
 <3,"b2",6.0>,
 <4,"b3",6.0>,
};
Cargos = {
 <1,"Ferro-guza",100.0,100.0>,
 <2,"Soja",200.0,0.0>,
 <3,"Óleo vegetal",250.0,5.0>,
 <4,"Petróleo",300.0,10.0>,
};
Vessels = {
 <1,"Navio #1",[6,17,4,21],105,6,0,8>,
 <2,"Navio #2",[2,1,29,17],109,10,0,8>,
 <3,"Navio #3",[2,34,6,6],101,12,0,8>,
 <4,"Navio #4",[0,30,7,11],106,20,0,8>,
 <5,"Navio #5",[9,7,24,7],106,22,0,8>,
 <6,"Navio #6",[8,20,14,6],104,24,0,8>,
 <7,"Navio #7",[15,3,23,7],102,24,0,8>,
 <8,"Navio #8",[27,6,5,10],109,24,0,8>,
 <9,"Navio #9",[2,17,29,0],104,24,0,8>,
 <10,"Navio #10",[2,26,11,8],103,24,0,8>,
 <11,"Navio #11",[22,5,12,8],100,26,0,8>,
 <12,"Navio #12",[11,7,19,10],103,30,0,8>,
 <13,"Navio #13",[28,8,1,11],105,48,0,8>,
 <14,"Navio #14",[16,20,12,0],106,50,0,8>,
 <15,"Navio #15",[3,32,9,4],103,50,0,8>,
};
Tides = {
<1,"">,
<2,"">,
<3,"">,
<4,"">,
<5,"">,
<6,"">,
<7,"">,
<8,"">,
<9,"">,
<10,"">,
<11,"">,
<12,"">,
<13,"">,
<14,"">,
<15,"">,
<16,"">,
<17,"">,
<18,"">,
<19,"">,
<20,"">,
<21,"">,
<22,"">,
<23,"">,
<24,"">,
<25,"">,
<26,"">,
<27,"">,
<28,"">,
<29,"">,
<30,"">,
<31,"">,
<32,"">,
<33,"">,
<34,"">,
<35,"">,
<36,"">,
<37,"">,
<38,"">,
<39,"">,
<40,"">,
};
```

Figura 4.19: Instância de BAP

Cabe aqui uma explicação sobre o formato do arquivo utilizado. Como já mencionado, a ideia é que a instância possa ser solucionada através de algum algoritmo otmizador. Dessa forma, foi trabalhado um formato compatível com CPLEX Optimizer, um *solver* comercial para problemas de programação matemática. O arquivo divide-se em 5 partes: o cabeçalho, contendo informações sobre a data de criação e o nome do programa; *Berths*, com os dados sobre os berços; *Cargos*, com os dados sobre as cargas em estoque; *Vessels*, com as informações sobre todos os navios que fazem parte da simulação no momento e *Tides*, com as informações sobre as marés. Não é necessário maiores explicações sobre o cabeçalho. Quanto às outras partes, a Figura 4.20 torna claro o significado de cada linha do arquivo.

```
Berths = {
< ID_DO_BERÇO1, "NOME_DO_BERÇO1", CAPACIDADE1>;
(...)
<ID_DO_BERÇO1, "NOME_DO_BERÇO1", CAPACIDADE1>;
}

Cargos = {
< ID_CARGA1,"NOME DA CARGA1",ESTOQUE_INICIAL1,CONSUMO1>;
(...)
<ID_CARGAn,"NOME DA CARGAn",ESTOQUE_INICIALn,CONSUMOn>;
}

Vessels = {
<ID_NAVIO1,"NOME DO NAVIO1",[qCARGA1,..,qCARGAk],DEMURRAGE1,CHEGADA1,TEMPO_SERVIÇO1,TEMPO_CONTRATO1>
(...)
<ID_NAVIOn,"NOME DO NAVIOn",[qCARGA1,..,qCARGAk],DEMURRAGEn,CHEGADAn,TEMPO_SERVIÇOn,TEMPO_CONTRATOn>
}

Tides = {
<ID_MARE1,"NOME DA MARÉ1">
(...)
<ID_MAREn,"NOME DA MARÉn">
}
```

Figura 4.20: Formato de Instância de BAP

5 Conclusão

Neste trabalho foi apresentado um Simulador de Porto capaz de gerar instâncias para o Problema de Alocação de Berços em Terminais Graneleiros sujeitos a restrições de maré e nível de estoque (BAPTGS).

O simulador foi desenvolvido como um *applet* Java, podendo ser executado diretamente em uma página *web,* independentemente do hardware e do sistema operacional utilizado.

A aplicação implementa uma simulação de eventos discretos, com tempo contínuo, na qual modela-se a dinâmica de operação de um terminal portuário. São disponibilizados ao usuário diferentes opções de configuração, que vão desde a distribuição de probabilidade que regula o processo de chegada dos navios até a quantidade de berços disponíveis. A aplicação realiza também o cálculo das medidas de desempenho e permite visualizá-las de maneira amigável.

A principal função do simulador é gerar instâncias para o BAPTGS, e estas consistem em arquivos-texto compatíveis com o CPLEX Optimizer, um *solver* comercial para problemas de programação matemática. Uma instância pode ser obtida a qualquer momento, enquanto a simulação estiver em andamento, e reflete toda a situação do porto naquele instante.

Em suma, os objetivos propostos foram cumpridos. Fica em aberto, todavia, a possibilidade de várias melhorias para futuros trabalhos. O próximo passo é integrar otimizador e simulador, de modo que este execute uma sequência de atracação otimizada (e não apenas a disciplina de serviço FCFS implementada até aqui). Isso pode ser feito de duas formas: a primeira é extendendo o código da aplicacão, talvez através de polimorfismo, como mencionado no Capítulo 4, Seção 4.2; a segunda é realizando a interface entre o simulador e um programa otimizador independente, como o próprio CPLEX Optimizer.

Com essas melhorias, o simulador aumentaria em muito sua utilidade. Em um cenáro ideal, o usuário poderia executar a simulação e, quando achar conveniente, enviar um comando para que a otimização aconteça. O resultado poderia ser analisado

através das medidas de desempenho já disponíveis. Dessa forma, diferentes algoritmos de otimização poderiam ser testados, comparados e melhorados. E o programa já estaria próximo a um sistema de apoio à decisão, podendo servir até mesmo para treinamento em operação portuária.

Referências Bibliográficas

[1] Agência Nacional de Transportes Aquaviários (ANTAQ). Anuário estatístico aquaviário 2012. `http://www.antaq.gov.br/Portal/Anuarios/Anuario2012/index.htm`, Setembro 2012.

[2] Ponto a Porto. O sistema portuário brasileiro. `http://pontoaporto.blogspot.com.br/2009_02_01_archive.html`, 2013. Acessado em 28 de Outubro de 2013.

[3] LEE, Y.; CHEN, C.-Y. An optimization heuristic for the berth scheduling problem. *European Journal of Operational Research*, Março 2008.

[4] KELTON, W. D.; SADOWSKI, R. P.; SADOWSKI, D. A. *Simulation with arena*. McGraw-Hill, 2002.

[5] JAIN, R. *Art of computer systems performance analysis techniques for experimental design measurements simulation and modeling*. Wiley Computer Publishing, John Wiley & Sons, Inc, Janeiro 1991.

[6] Mundo Educação. Transporte marítimo. `http://www.mundoeducacao.com/geografia/transporte-maritimo.htm`, 2013. Acessado em 12 de Novembro de 2013.

[7] KITE-POWELL, H. L. Marine policy: Shipping and ports. *Encyclopedia of Marine Science*, 2001.

[8] Global Business Reports. A mineração brasileira. *Engineering and Mining Journal*, 2006.

[9] MOCCIA, L. New optimization models and algorithms for the management of maritime container terminals. Março 2005.

[10] AAPA. Glossary of maritime terms. `http://www.aapa-ports.org/Industry/content.cfm?ItemNumber=1077`, 2013. Acesso em 28 de outubro de 2013.

[11] CORDEAU, J.-F.; LAPORTE, G.; LEGATO, P.; MOCCIA, L. Models and tabu search heuristics for the berth allocation problem. Abril 2005.

[12] IMAI, A.; ZHANG, J.-T.; NISHIMURA, E.; PAPADIMITRIOU, S. The berth allocation problem with service time and delay time objectives. *Maritime Economics and Logistics*, Setembro 2007.

[13] PINEDO, M. L. *Scheduling: Theory, algorithms, and systems.* Prentice-Hall, 1995.

[14] IMAI, A.; NISHIMURA, E.; PAPADIMITRIOU, S. Berthing ships at a multi-user container terminal with a limited quay capacity. *European Journal of Operational Research*, Outubro 2005.

[15] IMAI, A.; NISHIMURA, E.; PAPADIMITRIOU, S. The dynamic berth allocation problem for a container port. Agosto 1999.

[16] IMAI, A.; NAGAIWA, K.; TAT, C. W. Efficient planning of berth allocation for container terminals in asia. *Journal of Advanced Transportation*, Agosto 1997.

[17] NISHIMURA, E.; IMAI, A.; PAPADIMITRIOU, S. Berth allocation planning in the public berth system by genetic algorithms. *European Journal of Operational Research*, Junho 2001.

[18] IMAIA, A.; NISHIMURAA, E.; PAPADIMITRIOUC, S. Berth allocation with service priority. *Transportation Research Part B: Methodological*, Junho 2003.

[19] LI, C.-L.; CAI, X.; LEE, C.-Y. Scheduling with multiple-job-on-one-processor. *IEE Transactions Vol. 30, Issue 5, pp 433-445*, Maio 1998.

[20] BROWN, G. G.; CORMICAN, K. J.; LAWPHONGPANICH, S.; WIDDIS, D. B. Optimizing submarine berthing with a persistence incentive. *Naval Research Logistics*, Dezembro 1996.

[21] BARROS, V. H.; COSTA, T. S.; OLIVEIRA, A. C. M.; LORENA, L. A. N. Model and heuristic for berth allocation in tidal grain ports with stock level constraints. Setembro 2009.

[22] KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by simulated annealing. *Science*, Washington, v. 220, p. 671–680, 1983.

[23] CHUNG, C. A. *Simulation modeling handbook - a practical approach.* CRC PRESS, 2004.

[24] LAW, A. M.; KELTON, W. *Simulation modelling and analysis*. McGraw-Hill, Abril 2001.

[25] DAVISON, A. *Killer game programming in java*. O'Reilly, Maio 2005.