UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

METAHEURÍSTICAS GRASP E ILS APLICADAS AO PROBLEMA DA VARIABILIDADE NO TEMPO DE DOWNLOAD EM AMBIENTES DE TV DIGITAL

DANIEL GONÇALVES RAMOS

JOÃO PESSOA-PB
Setembro-2012

UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

METAEHURÍSTICAS GRASP E ILS APLICADAS AO PROBLEMA DA VARIABILIDADE NO TEMPO DE DOWNLOAD EM AMBIENTES DE TV DIGITAL

DANIEL GONÇALVES RAMOS

JOÃO PESSOA-PB
Setembro-2012

DANIEL GONÇALVES RAMOS

METAHEURÍSTICAS GRASP E ILS APLICADAS AO PROBLEMA DA VARIABILIDADE NO TEMPO DE DOWNLOAD EM AMBIENTES DE TV DIGITAL

DISSERTAÇÃO APRESENTADA AO CENTRO DE INFORMÁTICA DA UNIVERSIDADE FEDERAL DA PARAÍBA, COMO REQUISITO PARCIAL PARA OBTENÇÃO DO TÍTULO DE MESTRE EM INFORMÁTICA (SISTEMAS DE COMPUTAÇÃO).

Orientador: Prof. Dr. Lucídio dos Anjos Formiga Cabral

JOÃO PESSOA-PB
Setembro-2012

R175m Ramos, Daniel Gonçalves.
Metaheurísticas GRASP e ILS aplicadas ao problema da variabilidade no tempo de download em ambientes de TV digital / Daniel Gonçalves Ramos.-- João Pessoa, 2012.
68f. : il.
Orientador: Lucídio dos Anjos Formiga Cabral
Dissertação (Mestrado) – UFPB/CCEN
1. Informática. 2. Sistemas de computação. 3. Programas interativos. 4. Variabilidade do Tempo de Download (PVTD). 5. Metaherísticas GRASP e ILS.

UFPB/BC CDU: 004(043)

Ata da Sessão Pública de Defesa de Dissertação de Mestrado de **DANIEL GONÇALVES RAMOS**, candidato ao Título de Mestre em Informática na Área de Sistemas de Computação, realizada em 31 de agosto de 2012.

Ao trigésimo primeiro dia do mês de agosto do ano dois mil e doze, às quatorze horas, no auditório do CCEN - da Universidade Federal da Paraíba, reuniram-se os membros da Banca Examinadora constituída para examinar o candidato ao grau de Mestre em Informática, na área de "*Sistemas de Computação*", na linha de pesquisa "*Computação Distribuída*", o Sr. **DANIEL GONÇALVES RAMOS**. A comissão examinadora foi composta pelos professores doutores: LUCÍDIO DOS ANJOS FORMIGA CABRAL (PPGI-UFPB), Orientador e Presidente, VALÉRIA GONÇALVES SOARES (PPGI-UFPB) como examinadora interna, e DANIEL ALOISE (UFRN) como examinador externo. Dando início aos trabalhos, o professor LUCÍDIO DOS ANJOS FORMIGA CABRAL, cumprimentou os presentes, comunicou aos mesmos a finalidade da reunião e passou a palavra ao candidato para que o mesmo fizesse, oralmente, a exposição do trabalho de dissertação intitulado "*Metaheurísticas Grasp e ILS Aplicadas ao Problema da Variabilidade do Tempo de Download em Ambientes de TV Digital*". Concluída a exposição, o candidato foi arguido pela Banca Examinadora que emitiu o seguinte parecer: "*aprovado*". Assim sendo, deve a Universidade Federal da Paraíba expedir o respectivo diploma de Mestre em Informática na forma da lei e, para constar, eu, professor Alisson Vasconcelos de Brito, coordenador deste programa, servindo de secretário, lavrei a presente ata que vai assinada por mim mesmo e pelos membros da Banca Examinadora. João Pessoa, 31 de agosto de 2012.

Alisson Vasconcelos de Brito

Prof. Dr. LUCÍDIO DOS ANJOS FORMIGA CABRAL
Orientador e Presidente (PPGI-UFPB)

Profa. Dra. VALÉRIA GONÇALVES SOARES
Examinadora Interna (PPGI-UFPB)

Prof. Dr. DANIEL ALOISE
Examinador Externo (UFRN)

Dedico este trabalho a todos os meus familiares e amigos, que sempre estiveram presentes em minha vida

Agradecimentos

Agradeço primeiramente a Deus, grande responsável pelas conquistas dos homens na terra. Agradeço também a minha família, que me deu forças para perseverar na minha vida acadêmica e aos meus amigos, que me deram momentos de tranquilidade quando passei por dificuldades. Um agradecimento especial ao professor Lucídio Cabral, que me passou ensinamentos essenciais para a realização deste trabalho e ao professor Bruno Pessoa, que contribuiu ativamente no desenvolvimento do mesmo.

Resumo

A chegada da TV Digital trouxe consigo a possibilidade de criação de progra- mas interativos por parte das emissoras. Para isso, aplicativos para TV devem ser enviados pela emissora através do padrão Carrossel DSM-CC. Esse padrão permite que os dados sejam enviados de forma cíclica, a fim de que a qualquer momento que o usuário ligue a TV, o mesmo possa receber todos os dados transmitidos. Porém, a forma com que cada aplicativo interativo vai estar disponível no carrossel tem um impacto no tempo de espera do usuário. O carrossel pode ser modificado de forma a priorizar algumas aplicações, e assim dar maior satisfação a maioria dos usuários e também aumentar o lucro da emissora. Ainda não existe um modelo definido de como tratar a prioridade das aplicações. O trabalho corrente sugere um modelo de negócio inovador, buscando satisfazer os usuários, a emissora e a empresa contratante. Com as prioridades das aplicações, surge um novo problema, denominado neste trabalho como o Problema da Variabilidade do Tempo de Download (PVTD). Ele trata da forma com que o carrossel deve ser gerado para minimizar o atraso no download das aplicações. Isso é um problema difícil, o que inviabiliza a utilização de técnicas exatas para grandes instâncias. O trabalho propõe a utilização das metaheurísticas GRASP e ILS para solucionar o problema.

Abstract

The arrival of Digital TV has brought the possibility to broadcasters create interactive programs. For this, applications should be sent to the TV station via the standard DSM-CC carousel. This standard enables data to be sent cyclically, so that any time you turn on the TV, it can receive all data transmitted. However, the way each interactive application will be available on the carousel has an impact on the users' waiting time. The carousel can be modified to prioritize some applications, and so give more satisfaction to most users and also increase the profits of the station. It is not been defined yet a model of how to handle the priority of applications. Current work suggests an innovative business model, seeking to satisfy users, the broadcaster and the contractor.With the priorities of the applications, a new problem arises, termed here as the Download Time Variability Problem (DTVP). It defines the way that the carousel should be created to minimize the users' waiting times. This is a difficult problem, which makes the use of exact techniques unapplicable for large instances. The paper proposes the use of GRASP and ILS metaheuristics to solve the problem.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

O forte avanço na TV Digital proporcionou um aumento significativo nas pesquisas na área. A possibilidade de interação com a TV criou uma enorme expectativa acerca dessa nova tecnologia. Dessa forma, as pessoas além de assistirem seus programas prediletos em alta definição, podem usar a TV como meio para fazer compras, alugar filmes, opinar em enquetes, entre outras infinidades de formas de interação.

Tudo isso só foi possível graças ao uso de poderosos algoritmos de compressão (MPEG). O vídeo que antes tinha baixa qualidade e ocupava uma largura de banda de 6MHz, agora é transmitido em alta definição e ainda há sobras nessa mesma banda de 6MHz (ABNT, 2007). Com isso, é possível transmitir, além do vídeo, variedades de informações, que vão de aplicações interativas até outros vídeos em menor qualidade.

A possibilidade de transmissão de aplicações também acarreta em exigências por parte do receptor de TV Digital. O mesmo deve ser capaz de armazenar essas aplicações, e executá-las de forma adequada. Assim é exigido um espaço de memória reservado para guardar essas informações.

O caráter de transmissão em broadcast, ou seja, o mesmo conteúdo sendo enviado para todos os receptores, faz com que a transmissão de dados seja feita em forma cíclica. Isso para que todos os receptores, independente da hora que se conectarem a um canal, sejam capazes de capturar os dados que estão sendo enviados naquele momento. Esse modelo de transmissão cíclica lembra um carrossel (Figura 1.1) e o padrão utilizado na transmissão é o DSM-CC (Digital Storage Media Command and Control).

Porém a forma com que o carrossel vai estar organizado pode acarretar em grandes atrasos por parte dos receptores, que precisarão esperar um tempo elevado para carregar o aplicativo desejado. O ideal seria que os set-top boxes tivessem memória ilimitada, de forma que toda aplicação transmitida pela emissora pudesse ficar armazenada nos mesmos. Mas essa situação está longe da real no Brasil, dado que, por questões de inclusão digital, os set-top boxes produzidos aqui devem ter seus

custos reduzidos para atingir o maior número de telespectadores possível [Pessoa, 2008].

Sendo assim, é possível favorecer apenas algumas aplicações dentro do carrossel, melhorando a experiência da maioria dos usuários e satisfazendo as empresas contratantes e a emissora. Para isso, é necessário se definir um modelo de negócio justo. O presente trabalho propõe um modelo inovador e apresenta uma formulação matemática para o mesmo.

Definidas as prioridades das aplicações interativas dentro do carrossel, o problema passa a ser como organizar essas aplicações para refletir o modelo definido. Esse problema pode ser entendido como uma variação do *Response Time Variability Problem* (Corominas, 2009), o qual pertence à classe de problemas chamada sequências justas. O objetivo deste último é a construção de sequências justas utilizando n símbolos, onde o símbolo $i (i = 1, \ldots, n)$ deve ocorrer d_i vezes numa sequência e cada símbolo de qualquer subsequência está alocado em posições justas (Kubiak, 2004).

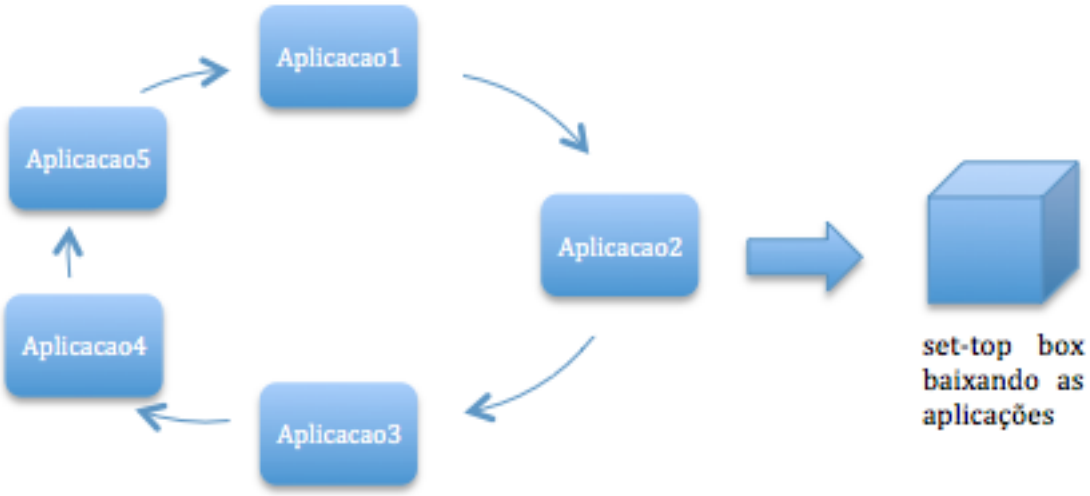

Figura 1.1: O conceito de Carrossel

1.1 Objetivos do trabalho

O objetivo geral desse trabalho é resolver o problema de organização das aplicações de um carrossel, a fim de otimizar o tempo de espera dos telespectadores, e viabilizar um modelo lucrativo para todas as partes envolvidas na transmissão de conteúdo televisivo: o telespectador, o anunciante e a emissora. Os objetivos específicos desse trabalho são:

- Propor regras de negócio para que as emissoras possam vender o espaço da banda restante para que terceiros enviem suas aplicações. Esse modelo de

negócio deve explicitar quais aplicações terão maior prioridade no carrossel.

- Implementar um algoritmo para gerar um carrossel de boa qualidade, respeitando o modelo de negócio proposto, além de gerar um banco com esses dados, para que trabalhos futuros possam utilizar essa fonte para comparar os resultados.

- Comparar os resultados obtidos com um trabalho semelhante, a fim de verificar a eficácia do algoritmo proposto, bem como comparar os modelos de negócios, que divergem em alguns aspectos.

1.2 Organização da proposta

O texto desta dissertação está dividido em capítulos, na seguinte forma:

- Capitulo 1: O primeiro capitulo descreve superficialmente o problema abordado, trazendo o contexto em que o mesmo está inserido. Também são apresentados os objetivos gerais e específicos do trabalho desenvolvido.

- Capitulo 2: Neste capitulo os conceitos acerca da TV Digital e da Pesquisa operacional são apresentados. As duas técnicas a serem utilizadas (GRASP e ILS) para resolver o problema são descritas.

- Capitulo 3: Neste capítulo, o problema abordado é mais detalhado, mostrando o funcionamento do carrossel, e a dificuldade de se montar um carrossel de boa qualidade. São mostrados trabalhos relacionados na literatura, além de formalizar o problema, comparando com problemas já conhecidos na literatura.

- Capitulo 4: O quarto capítulo é referente a metodologia utilizada na implementação das metaheurísticas propostas para resolver o problema abordado. Também é mostrado como foram escritos os arquivos de entrada e saída, bem como a forma de gerar as instâncias utilizadas nos testes.

- Capitulo 5: Este capítulo traz os resultados alcançados com o trabalho. Inicialmente se descreve os valores adotados para teste, além de descrever o ambiente em que foram feitas as execuções. Posteriormente, se apresenta os resultados obtidos com comparações de diferentes implementações e abordagens.

- Capitulo 6: O último capítulo apresenta as conclusões obtidas com o trabalho e possíveis abordagens para trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Neste capítulo são mostrados os principais conceitos acerca da Televisão Digital, cenário ao qual o problema descrito está inserido. Também é mostrado como o carrossel de objetos foi concebido segundo o padrão DSM-CC. Por fim, alguns conceitos relacionados a pesquisa operacional são apresentados, já que técnicas dessa área serão utilizadas para resolver o problema.

2.1 História da TV Digital

Nos anos 70, movidos por um forte avanço tecnológico, o Japão foi o primeiro país a propor o desenvolvimento de uma TV de alta definição, que seria chamada de HDTV (High Definition Television). Os cientistas responsáveis do NHK Science & Technical Research Laboratories logo perceberam que não seria possível trazer grandes melhorias caso continuassem com o padrão analógico (Pereira, 2008).

Pouco depois, motivados por essa idéia inovadora, os europeus desenvolveram a base que viabilizou toda compressão da imagem e áudio, o MPEG-1 e posteriormente o MPEG-2. Logo surgiram os padrões de transmissão e recepção que são utilizados nos dias de hoje, como o ATSC (Advanced Television System Committee), o ISDB (Integrated Services Digital Broadcasting) e o DVB (Digital Video Broadcasting), que foram implantados nos EUA, no Japão e na Europa respectivamente (Pereira, 2008).

No Brasil, o formato digital começou a ser transmitido pelas TVs por assinatura, porém não utilizavam alta definição e tinham interação limitada. Em 1998 a Anatel iniciou seus estudos sobre TV Digital, comparando os padrões existentes no mercado. Observou-se que o padrão americano não atendia as necessidades brasileiras, já que seu desempenho era insatisfatório em recepções domésticas (Becker, 2009).

Descartado o ATSC, a dúvida passou a ser entre os padrões europeu e japonês. O segundo acabou vencendo essa disputa, pois se comportava melhor em ambientes fechados. Em 2000 a Anatel encerou as discussões sobre qual padrão seria adotado.

Com um novo governo, o país passou a enxergar a necessidade de se desenvolver um padrão próprio, ao invés de simplesmente copiar o japonês. Isso para permitir a inclusão digital e formas mais atrativas de interação. Com isso surgiu o SBTVD, conhecido comercialmente no mundo como ISDB-Tb. Esse padrão é reconhecido como o mais avançado, e há estudos de como incorporar a tecnologia brasileira aos padrões existentes (Becker, 2009).

O único componente do ISDB -Tb desenvolvido totalmente no Brasil é o middleware Ginga, que conta com a participação das universidades PUC-Rio e UFPB. Na UFPB o laboratório responsável por fazer esse desenvolvimento é o LAVID, ambiente em que o trabalho corrente foi produzido.

2.2 Estrutura de um Fluxo de Vídeo TS

Para se transmitir um fluxo de áudio e vídeo para TV digital, é necessário utilizar o padrão MPEG-2 (Moving Picture Experts Group-2). Ele define toda sintaxe dos dados transmitidos. Para utilizar este padrão, é necessário que esses dados sejam empacotados e multiplexados, de forma que mais de um fluxo seja transmitido simultaneamente. Isso é possível graças a um robusto padrão de controle, que faz com que cada fluxo elementar tenha apenas um tipo de informação, como áudio, vídeo, ou aplicações.

Cada pacote que forma um fluxo elementar é formado por um cabeçalho e um payload, que é a carga útil desse pacote. O cabeçalho possui informações que identificam o mesmo unicamente no fluxo, e a qual fluxo ele pertence. A informação mais importante, que define o fluxo do pacote, é o PID (Packet ID).

Os pacotes de um fluxo podem ser utilizados para formar um pacote PES, que é uma estrutura criada para transmitir áudio e vídeo, ou podem formar estruturas mais complexas, que são as sessões. Atualmente existem centenas de sessões distintas descritas no padrão brasileiro, e em uma delas são transportadas as sessões DSM-CC, que podem conter carrosséis.

Para gerenciar os pacotes que fazem parte de fluxos distintos, existem tabelas de controle chamadas de PSI (Program Specific Information), e as principais são a PAT (Program Association Table) e a PMT (Program Map Table). A PAT descreve os serviços presentes em um fluxo, ou seja, quantas PMTs existem, e o PID das mesmas. Já cada PMT descreve um serviço único, apontando o PID de cada fluxo diferente presente no serviço, como o áudio, vídeo, ou dados. O esquema pode ser visto na Figura 2.1 (Morris, 2002).

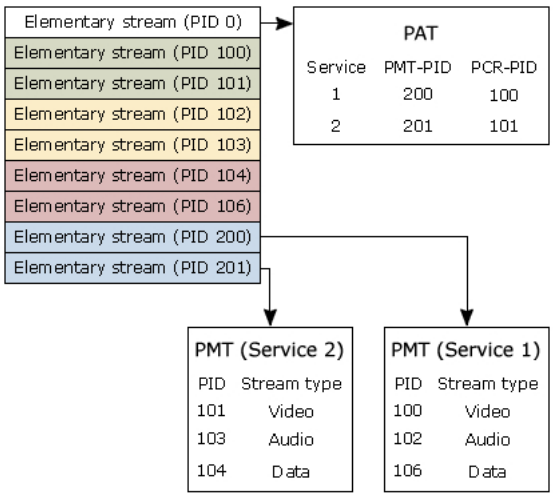

Figura 2.1: Tabelas PAT e PMT (Morris, 2002)

2.3 Carrossel DSM-CC

O padrão DSM-CC foi desenvolvido para dar suporte a transmissão de serviços multimídia. Um protocolo aberto é essencial para a entrega em larga escala desses serviços. A interoperabilidade total entre o provedor de serviço e os consumidores requer que muitos aspectos do uso da banda sejam definidos. A possibilidade de envio de sistemas de arquivos completos para um receptor é um dos principais motivos do sucesso desse padrão.

No modelo DSM-CC, um fluxo é provido por um servidor e entregue a um cliente. Tanto o cliente como o servidor são considerados usuários da rede DSM-CC. Ambos são sub-sistemas lógicos e não implicam em um único dispositivo na implementação real (Balabanian, 1996).

Os dados carregados no Carrossel provém normalmente de um único PID, havendo instâncias mais complexas que o carregam em mais de um PID. Para o primeiro caso, identificar os pacotes do carrossel se torna fácil, bastando apenas criar um filtro para um determinado PID.

Foram definidos dois diferentes padrões de carrossel: O carrossel de dados e o carrossel de objetos.

2.3.1 Carrossel de Dados

O carrossel de dados oferece mecanismos para transmissão periódica de dados brutos para um conjunto de clientes. Esses dados são encapsulados em blocos DDB (Download Data Block), que são transmitidos juntamente com mensagens de controle, chamadas DII (Download Info Indication) e DSI (Download Service Indication). Informações de tamanho e identificação dos blocos são trazidas nessas mensagens de controle.

Após capturar todos os blocos e mensagens de controle de um carrossel de dados, basta verificar a ordem e a forma com que esses blocos vão se juntar para formar os módulos. A DII identifica exatamente isso, descrevendo a quantidade de blocos de cada módulo do carrossel, e o tamanho dos mesmos. A DSI é utilizada somente quando se faz necessário mais de 1 DII, já que esta só pode referenciar no máximo 150 módulos (Morris, 2002). O conjunto de módulos referenciado por uma DII forma um grupo, e o conjunto de grupos referenciados por uma DSI forma um super grupo. O esquema pode ser visto na figura 2.2.

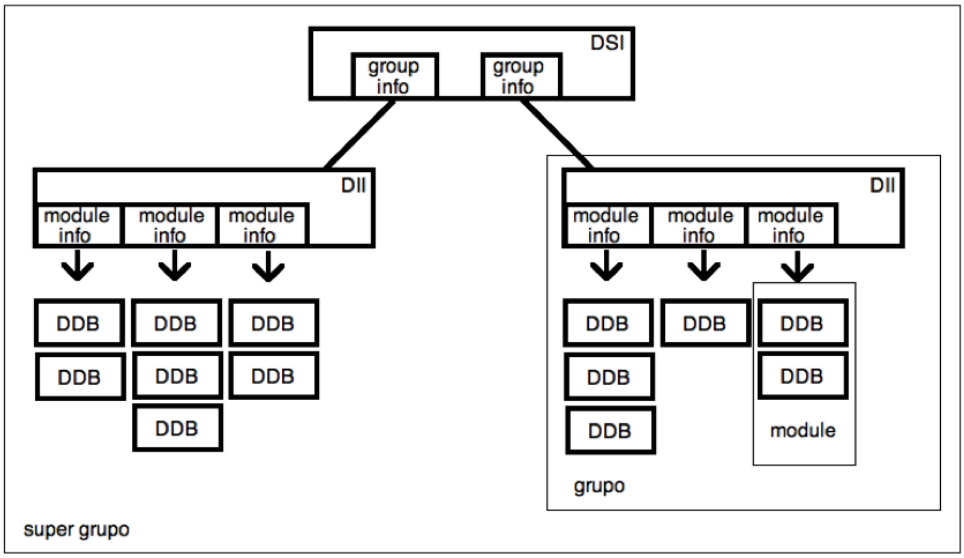

Figura 2.2: Carrossel de Dados

O conjunto de módulos é o produto final de um carrossel de dados, e a interpretação desses dados cabe ao transmissor e receptor. Cada módulo pode carregar um ou mais arquivos, porém não se pode carregar um único arquivo em mais de um módulo. O tamanho máximo para agrupar arquivos em um módulo é 64KB, portanto caso um arquivo seja maior ou igual a esse tamanho, ele deve ser transmitido sozinho em um módulo.

2.3.2 Carrossel de Objetos

Enquanto que um carrossel de dados é suficiente para aplicações simples, que não demandem grande semântica nos dados transmitidos, como um tele texto, o carrossel de objetos foi criado para dar suporte a transmissões mais avançadas, pois oferece um avançado protocolo que encapsula um sistema de arquivos completo (Morris, 2002).

Carrosséis de objetos são feitos em cima de carrosséis de dados, mas estendem seu conceito. O conjunto de dados brutos, transmitidos no carrossel de dados, passa a ser um conjunto de objetos BIOP (Broadcast Inter-ORB Protocol Messages).

Esses objetos foram definidos pela CORBA (Vinoski, 1997). A figura 2.3 ilustra o encapsulamento do carrossel de dados para formar o carrossel de objetos.

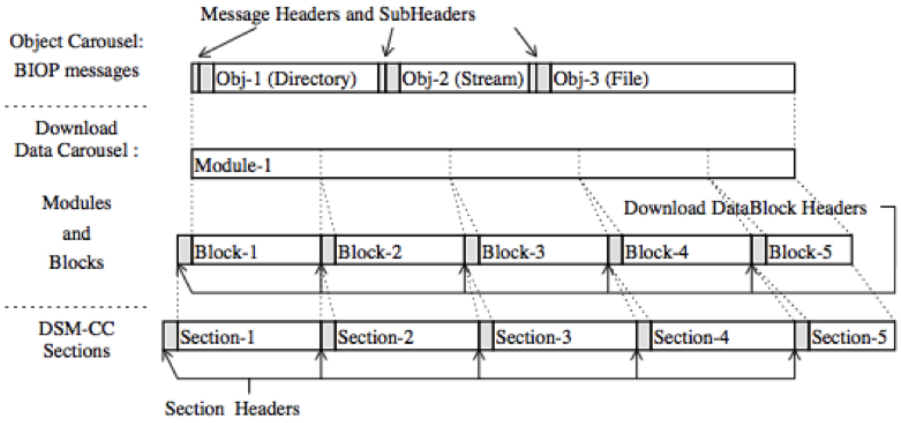

Figura 2.3: Encapsulamento do Carrossel de Dados

Existem diversos tipos de objetos BIOP, cada um tendo um significado diferente para o receptor. O primeiro tipo de objeto é o File BIOP, que é um objeto que armazena um arquivo real, incluindo todo seu conteúdo. O segundo é o Directory BIOP, que contém um diretório, que pode conter arquivos e até outros diretórios. Um segundo tipo de diretório, chamado de Service Gateway BIOP, é utilizado para guardar o diretório principal do carrossel de objetos transmitido. Só existe um Service Gateway por carrossel.

Existem outros 2 tipos de objetos BIOPs, o Stream BIOP e o Stream Event BIOP. O primeiro é utilizado para referenciar outros fluxos naquela transmissão, seja de vídeo ou áudio. O segundo é utilizado para descrever um evento, que pode ser transmitido através de um Stream Event Descriptor. Uma visão alto nível do Carrossel de Objetos DSM-CC pode ser visto na figura 2.4.

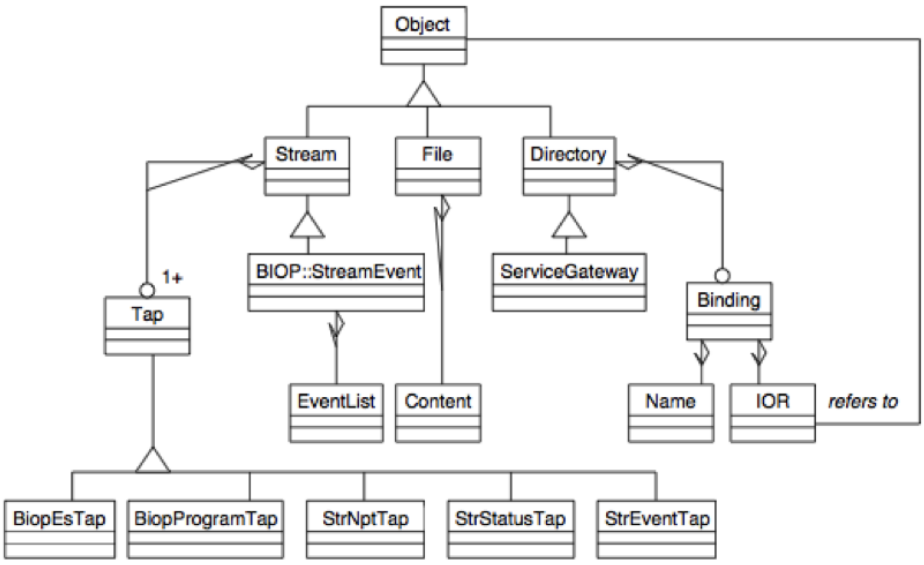

Figura 2.4: Hierarquia do Carrossel

2.4 Programação Linear

Problemas de Programação Linear são aqueles que podem ser descritos matematicamente com restrições lineares, isto é, as restrições e relações entre varíaveis são lineares. Para eles, existem técnicas da pesquisa operacional que permitem a busca pela solução exata sem a necessidade de percorrer um número exponencial de elementos. O Simplex é um exemplo clássico de algoritmo que se mostrou bastante eficaz na solução de problemas dessa classe (Schrijver, 1998).

O simplex se baseia no fato de que soluções boas para problemas lineares estão sempre nos vértices dos contornos das regiões de soluções viáveis. Então, para se buscar a melhor solução, em vez de varrer todo o campo de possíveis soluções, basta se buscar nos vértices.

Existe uma sub-classe da Programação Linear, chamada de Programação Linear Inteira ou simplesmente Programação Inteira (PI). Nela, as variáveis só podem assumir valores inteiros. Muitos desses problemas são considerados NP-difícil, isto é, não há um algoritmo capaz de resolvê-lo em tempo polinomial.

Foi criada então uma técnica, chamada de branch-and-bound, que aliada ao Simplex, diminui bastante o tempo necessário para se buscar a solução exata para os problemas de PI. O branch-and-bound utiliza o Simplex para buscar a melhor solução (provavelmente inviável, pois viola a restrição do caráter Inteiro das varíaveis) e depois aplica uma técnica que busca as soluções inteiras mais próximas da solução encontrada.

O Cplex é uma biblioteca que permite a execução do Simplex + branch-and-bound através da linguagem C++. Ela implementa os 2 algoritmos, e resolve uma grande variedade de problemas de PI, bastando apenas inserir as restrições e a função

objetivo corretamente no algoritmo implementado em C++. Porém, para instâncias muito grandes, o tempo para se calcular uma solução inviabiliza sua utilização.

2.5 Heurísticas

Calcular a solução exata para problemas de natureza NP-difícil é uma tarefa na maioria das vezes inviável, devido ao tempo necessário para alcançar tal objetivo. Por outro lado, na prática, em geral, é suficiente encontrar uma "boa"solução para o problema, ao invés do ótimo global. Para isso, pesquisadores contam com a utilização de heurísticas para solucionar problemas deste nível de complexidade. Heurística é definida como uma técnica inspirada em processos intuitivos que procura uma boa solução a um custo computacional aceitável, sem, no entanto, estar capacitada a garantir sua otimalidade, bem como garantir o quão próximo está da solução ótima (Souza, 2011).

Existem dois tipos de heurísticas: as construtivas e as de refinamento. Uma heurística construtiva tem por objetivo construir uma solução, elemento por elemento. A forma de escolha de cada elemento a ser inserido a cada passo varia de acordo com a função de avaliação adotada, a qual, por sua vez, depende do problema abordado. Um exemplo de heurística construtiva é a gulosa, que estima o benefício da inserção de cada elemento, e somente o "melhor"elemento é inserido a cada passo. Porém isso não garante que a solução gerada será a melhor. Já as heurísticas de refinamento buscam melhorar a solução através de técnicas de busca local. Essas técnicas utilizam a noção de vizinhança, soluções que estão próximas por um dado critério da solução atual, para tentar se chegar a um ótimo local (Souza, 2011).

2.6 Metaheurísticas

As metaheurísticas são métodos heurísticos utilizados para resolver problemas de otimização de alta complexidade. Elas são empregadas no caso em que um algoritmo exato se torna impraticável e o uso de uma única heurística se torna ineficiente. Apesar de não resolver o problema na forma exata, elas são uma ótima ferramenta capaz de prover resultados bastante satisfatórios dependendo do ambiente em que é aplicada (Glover, 3003).

Uma metaheurística utiliza mais de uma heurística para construir e melhorar muitas soluções em uma execução. Na maioria das vezes existe alguma heurística gulosa e também uma heurística de refinamento. Suas soluções também possuem normalmente um certo grau de aleatoriedade, a fim de fugir dos ótimos locais e quem sabe se chegar ao ótimo global.

2.6.1 GRASP

O GRASP (Greedy Randomized Adaptive Search Procedure) é uma metaheurística que utiliza uma heurística gulosa e aleatória para tentar alcançar um resultado satisfatório. Ela constrói algumas soluções iniciais, e através de uma busca local, procura encontrar uma solução quase-ótima (Resende, 1998).

As soluções iniciais do GRASP são feitas a partir da geração de soluções gulosas com um certo grau de aleatoriedade. O custo atribuído aos elementos que constituem uma solução válida são dados pelo usuário, o que torna o algoritmo adaptativo. Após a geração de uma solução inicial, o algoritmo faz uma busca local com trocas simples para tentar encontrar uma solução melhor. O processo é repetido "GRASP-Max"vezes. Um pseudo-código que efetua o algoritmo GRASP pode ser visto no algoritmo 1.

Algoritmo $GRASP(f(.), g(.), N(.), GRASPMax, s)$
1 f* $\leftarrow\infty$;
2 **para** $1, 2, ..., GRASPMax$ **faça**
3 Construção$(g(.), \alpha, s)$;
4 BuscaLocal$(f(.), N(.), s)$;
5 **se** $f(s) < f*$ **então**
6 $s* \leftarrow s$;
7 $f* \leftarrow f(s)$;
8 **fim se**
9 **fim para**
10 **devolva** $s*$;
fim

Algoritmo 1: Algoritmo GRASP

Para se construir uma solução de forma aleatória e gulosa, é utilizada uma lista, chamada de LCR (Lista restrita de candidatos). A construção se dá por iterações, onde em cada uma delas, a lista é recomposta. A construção da lista é feita se utilizando um fator (alfa), que é o grau de aleatoriedade. Quanto maior o alfa, mais elementos terá a lista, e mais aleatória será a solução. Selecionam-se os (alfa) melhores elementos que otimizem de forma gulosa a solução atual (ou seja, para aquela nova iteração, os elementos que deixem a melhor solução possível para o próximo passo). Daí se escolhe aleatoriamente um dos elementos da lista para se compor a solução.

Na fase de busca local, o algoritmo busca soluções próximas através de uma ou mais vizinhanças. Sempre que achar uma solução melhor, o algoritmo atualiza a solução. Após não encontrar nenhuma possível melhora com as vizinhanças conhecidas, o algoritmo para, grava a solução, e inicia novamente, até que um critério de parada seja obedecido.

2.6.2 ILS

O algoritmo ILS (Iterated Local Search) se baseia no fato de que as soluções ótimas globais nem sempre são encontradas por uma busca local. Isso porque o espaço de soluções pode ser tão grande, que para uma determinada vizinhança a solução ótima não seja nem próxima da melhor solução global.

A aplicação do ILS é feita a partir de uma perturbação da solução encontrada, a fim de encontrar uma vizinhança distante, que possua um melhor ótimo local. A aplicação de varias perturbações, percorrendo o maior numero de vizinhanças possíveis, nos dá uma maior probabilidade de encontrar o ótimo global. Um pseudo-código que efetua o algoritmo ILS pode ser visto no algoritmo 2.

Algoritmo *ILS*
1 $s_0 \leftarrow$GeraSoluçãoInicial;
2 $s \leftarrow$BuscaLocal(s_0);
3 **enquanto** os critérios de parada não estiverem satisfeito **faça**
4 $s' \leftarrow$Pertubação(s);
5 $s'' \leftarrow$BuscaLocal(s');
6 $s \leftarrow$CritérioAceitação(s, s'');
7 **fim enquanto**
8 **devolva** s;
fim

Algoritmo 2: Algoritmo ILS.

Para se entender o ILS, basta analisar a figura 2.5. Após um algoritmo construtivo gerar uma solução, o mesmo só é capaz de otimizar com sua busca local até s. Portanto, é necessário fugir daquele ótimo local. Com a perturbação, a solução cai em s' e com mais uma busca local, se chega no ótimo global s".

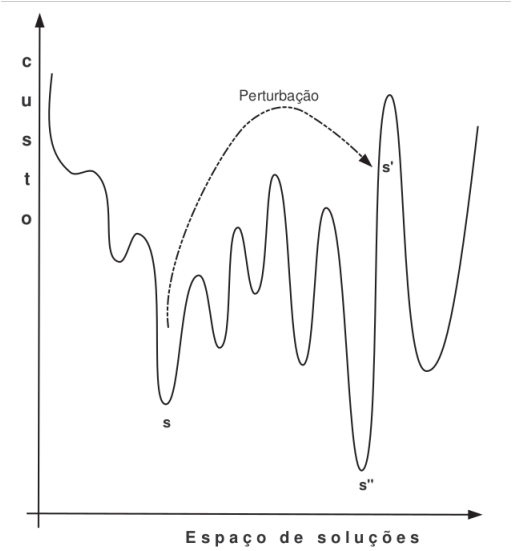

Figura 2.5: Representação esquemática do funcionamento do ILS

Capítulo 3

Apresentação e Descrição do Problema

Durante uma transmissão digital, a emissora envia o vídeo juntamente com uma enorme variedade de dados. Esses dados podem ser desde a grade de programação do canal até aplicativos personalizados. Os dados são enviados na banda restante do canal de 6MHz disponível para cada emissora no padrão brasileiro. Pode-se atingir com isso uma taxa total de 20 Mbps, sendo 14 Mbps reservados para o vídeo no padrão H.264. Assim, sobra uma banda de até 6 Mbps para se transmitir dados.

Os receptores de TV equipados com o Midleware Ginga são capazes de executar qualquer aplicativo enviado pelas emissoras. Esses aplicativos são específicos para TV Digital, e, para o padrão brasileiro, são permitidas as linguagens Java e NCL (ABNT NBR 15601).

Porém, uma característica dos receptores nacionais é a pouca capacidade de armazenamento. Isso para que esses produtos tivessem um preço reduzido e assim um maior número de brasileiros pudessem adquirí-los. O impacto disso é que toda vez que um receptor comum precisar dos dados de um aplicativo, o mesmo deve esperar o envio por parte da emissora.

A transmissão dos aplicativos é feita através do Carrossel de Objetos DSM-CC, e portanto, para que o usuário execute uma aplicação, seu receptor deve ser capaz de receber e interpretar esses dados. As informações de uma aplicação interativa são enviadas numa tabela a parte, chamada AIT. Nela, todos os dados descritivos das aplicações são enviados. Com essa tabela, o usuário pode escolher uma aplicação de seu interesse, e começar a baixar do Carrossel que está sendo enviado.

Com a tendência de que muitas aplicações vão para o ar num futuro próximo, se faz necessário um estudo para melhorar a qualidade do serviço oferecido aos telespectadores interessados em rodar aplicações em suas TVs. Para isso, deve-se garantir um tempo de carregamento máximo para cada aplicação. Logo percebe-se que não se pode enviar muitas aplicações de uma vez, tampouco aplicações com

tamanho exagerado, pois assim o usuário teria que esperar um tempo exorbitante para poder utilizá-las.

A forma com que as aplicações vão estar dispostas no Carrossel também tem um impacto importante na espera dos usuários. Favorecer algumas aplicações pode melhorar a experiência da maioria dos usuários, além de satisfazer as empresas que pagam mais pela veiculação de suas aplicações. Dentro de um carrossel, uma aplicação pode ter prioridade sobre as outras. Para isso, basta replicar essa aplicação dentro do carrossel. A figura 3.1 mostra um exemplo onde a aplicação vermelha tem maior prioridade sobre todas as outras. Assim, o tempo máximo que o usuário espera para carregar a aplicação com prioridade pode cair consideravelmente.

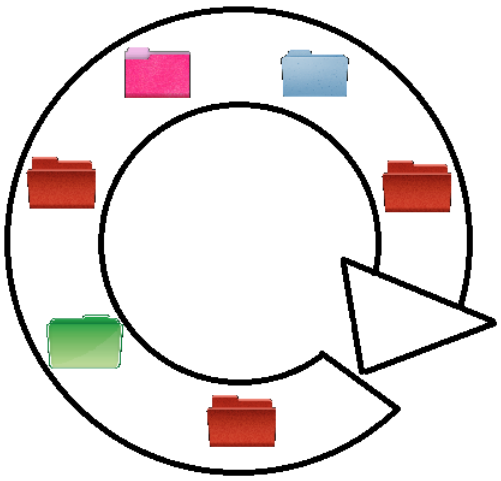

Figura 3.1: Carrossel com Prioridade

Definir quais aplicações devem ter prioridade no carrossel é uma tarefa complicada, já que isso deve respeitar o contrato das emissoras, e as aplicações mais utilizadas pelos usuários devem ter um tempo de espera menor. Além disso, a prioridade pode ser alterada instantaneamente, caso a aplicação esteja recebendo muitos acessos.

Dadas as prioridades das aplicações, o problema passa a ser gerar o carrossel de forma otimizada. Isso é uma tarefa difícil, já que existe um número exponencial de formas para organizar as aplicações, já que as mesmas podem ser replicadas inúmeras vezes. Para que se tenha uma noção da ordem de grandeza desse número, supondo que existem n aplicações, que podem ser colocadas em $q \geq n$ posições no carrossel, o número de combinações possíveis é maior do que $q! \times n!$.

Dessa forma, testar todas as combinações possíveis de se formar um carrossel ótimo, dadas as aplicações e suas prioridades, demoraria um tempo proibitivo, caso se tenha uma quantidade considerável de aplicações. Assim, é necessário se aplicar um algoritmo inteligente, capaz de gerar boas soluções em um tempo aceitável.

3.1 Trabalhos Relacionados

Por ser um tema recente, poucos trabalhos foram desenvolvidos com o propósito de otimizar a geração dos carrosséis. Em sua maioria, os trabalhos buscam otimizar o carrossel para uma única aplicação, aumentando a taxa de envio de certos arquivos, a fim de que o programa possa carregar mais rápido, como o trabalho de Tien-Ying Kuo (2008).

Porém, se formos pensar numa realidade em que várias aplicações são enviadas ao mesmo tempo, apenas a otimização dentro da aplicação não basta. Esse cenário é bastante provável, dado que muito dinheiro é investido em propaganda nas TVs abertas no Brasil, e com a possibilidade de envio de aplicações interativas, certamente os contratantes vão se interessar por tal tecnologia.

Visando otimizar a organização de aplicações interativas, apenas um trabalho foi encontrado. Nele, o autor propõe um modelo de negócio parecido com o utilizado no trabalho corrente, porém tem foco apenas no lucro da emissora (Pessoa, 2008), não levando em consideração a utilização das aplicações por parte dos usuários.

3.1.1 O trabalho de Pessoa (2008)

Na modelagem de Pessoa (2008), o carrossel deveria ser gerado de forma a otimizar o lucro da emissora, propondo um modelo para contratação dos serviços de propaganda por meio de aplicativos interativos. Com esse modelo, são definidas as prioridades dos aplicativos de forma estática, ou seja, não leva em consideração a utilização das aplicações durante a transmissão. Como consequência, um único fluxo de Carrossel é gerado para toda transmissão de um determinado programa.

As prioridades de cada aplicação são determinadas a partir do quanto se ganha pelo envio da mesma. Além disso, uma multa é calculada em cima do valor pago pela empresa contratante, caso o atraso máximo seja superior a um valor fixado. Foram descritos dois cenários: o primeiro leva em consideração o tamanho da aplicação para se calcular o valor pago. Já o segundo, as faixas de preço são indiferentes ao tamanho da aplicação. Porém para ambos, a empresa contratante poderia pagar mais para que sua aplicação tivesse maior prioridade. Dessa forma, foram criadas faixas de valores a serem pagos por MB da aplicação (cenário A) ou por tempo de veiculação (cenário B), onde quanto maior a faixa, maior a prioridade.

Para calcular o carrossel ideal, foram utilizadas as metaheurísticas GRASP + VND (GVND) e GRASP + VNS (GVNS). O calculo do ganho da inserção de uma aplicação no carrossel levou em conta o valor pago pela mesma, o tamanho e o número de vezes que ela já apareceu no carrossel. Uma aplicação poderia nunca ser inserida no carrossel. O tamanho máximo do Carrossel levou em conta o tempo total disponível para transmissão. Porém, nos testes realizados nesse trabalho, verificou-

se que o melhor número de aplicações máxima do carrossel era em torno de duas ou três vezes o número total de aplicações.

Para as buscas locais (VND) e variações nas vizinhanças (VNS), foram utilizados movimentos de troca de posições de aplicações, inserção de novas aplicações e remoção de aplicações do carrossel.

Por ser um trabalho novo na literatura, as comparações dos resultados se deram com um algoritmo padrão para geração do Carrossel: são inseridas todas as aplicações sequencialmente, sem replicação. Os resultados foram bastante relevantes para algumas instâncias, chegando a ter ganhos de mais de 30% no lucro da emissora. Porém, houveram instâncias em que o ganho foi abaixo dos 1%. Todas as instâncias foram geradas a partir de um estudo realizado no Lavid, medindo o tamanho médio das aplicações, e gerando faixas aleatórias para cada aplicação. Esses resultados podem ser observados na tabela 3.1.

Tabela 3.1: Resultados de Pessoa (2008)

Instância	Solução Atual (R$)	GVND (R$)	GVNS (R$)	Diferença Percentual (R$)		
				Ganho 1	Ganho 2	Ganho 3
1	494.213,63	515.989,53	518.979,43	4,41	5,01	0,60
2	847.817,20	856.288,85	862.260,56	1,00	1,70	0,70
3	665.352,83	665.352,83	667.848,09	0,00	0,38	0,38
4	686.040,24	695.735,43	695.735,43	1,41	1,41	0,00
5	564.443,99	594.338,99	596.337,41	5,30	5,65	0,35
6	836.188,44	858.764,58	862.590,53	2,70	3,16	0,46
7	802.245,38	917.090,16	917.090,16	14,32	14,32	0,00
8	852.626,47	936.838,61	936.838,61	9,88	9,88	0,00
9	749.796,63	795.866,14	798.168,60	6,14	6,45	0,31
10	858.005,81	929.091,26	929.091,26	8,28	8,28	0,00
11	974.174,94	1.086.103,89	1.086.103,89	11,49	11,49	0,00
12	907.228,55	1.102.782,58	1.102.782,58	21,56	21,56	0,00
13	840.132,76	1.118.117,48	1.118.117,48	33,09	33,09	0,00
14	842.525,36	1.132.070,52	1.132.070,52	34,37	34,37	0,00
15	778.131,17	1.023.192,76	1.026.302,19	31,49	31,89	0,40

Nela, estão dispostos os resultados, em lucros financeiros, para as instâncias criadas pelo autor. A solução atual é referente a Geração comum de um carrossel, ou seja, todas as aplicações estão presentes em uma ordem sequencial. Já a GVND, mostra o resultado com o algoritmo GRASP + Busca Local. A coluna GVNS traz o resultado obtido com o uso do algoritmo GRASP + VNS. Ganho1 é o ganho obtido com o GVND sobre a solução atual (em porcentagem). Ganho 2 é o ganho obtido com o GVNS sobre a solução atual (em porcentagem). Já o Ganho 3 é o incremento obtido, em termos percentuais, com a solução GVNS em relação ao GVND.

Esses resultados foram utilizados com o cenário A, que é o mais parecido com o

que será adotado pelo trabalho corrente. Já se pode observar que o algoritmo GVNS não deu uma contribuição significante para a solução e por isso o trabalho corrente propõe uma segunda metaheurística para substituí-la, o ILS.

3.2 Modelo de Negócio Proposto

Ainda não há um consenso de como as propagandas via aplicativos interativos serão comercializadas. De qualquer forma, se prevê um aumento considerável na receita das emissoras, já que esse novo modelo tende a se perpetuar (Médola, 2009). Assim, essa seção traz uma possível abordagem para o problema, com um modelo de negócio inovador.

A interatividade na TV Digital trouxe consigo um novo modelo de negócio para as emissoras. Estas, que antes comercializavam espaço para propaganda, diferenciando os preços de acordo com horário e tempo de exibição, agora vão poder explorar a banda extra de seus canais para comercializar aplicativos de terceiros.

Empresas interessadas em divulgar as suas aplicações deverão primeiro encontrar uma emissora para fazer a transmissão. Porém, será possível que mais de uma aplicação esteja no ar ao mesmo tempo, e a forma de comercialização deve levar em conta o tamanho da aplicação, o horário, o tempo que a mesma permanecerá disponível e o número de usuários que vão utilizá-la.

O modelo proposto é focado em propaganda e tem como inspiração o Google Adwords, um modelo de propaganda na Internet que cobra de acordo com o número de clicks nos links patrocinados. Esse é o modelo mais utilizado para publicidade na Internet e é satisfatório para ambas as partes envolvidas, já que o contratante só paga pelos clicks que recebeu, e a Google (ou empresa contratada) só recebe se expor as propagandas adequadamente. O forte avanço da Internet tornou viável esse modelo e tem tido um crescimento bastante elevado, como mostra a Figura 3.2. Ela mostra em percentuais o crescimento da publicidade em diferentes meios, no primeiro semestre de 2011.

De fato o modelo de comercialização da propaganda em função do número de usuários atingidos é bastante atrativo. Para se aplicar esse modelo na veiculação de aplicações interativas para TV Digital, um preço inicial por aplicação deveria ser cobrado em função do tamanho da aplicação, do horário de transmissão e da classe da aplicação. Cada programa do canal teria um preço inicial associado. O contratante poderia escolher em quais programas deseja que a sua aplicação seja exibida, já que determinados horários podem não ser interessantes para o mesmo.

As classes de aplicação são designadas de acordo com o valor pago por elas. Cada classe define um valor fixo a ser pago por KB (1024 bytes – unidade de tamanho) da aplicação. As classes são definidas de 1 a 10, onde na classe 1, o contratante

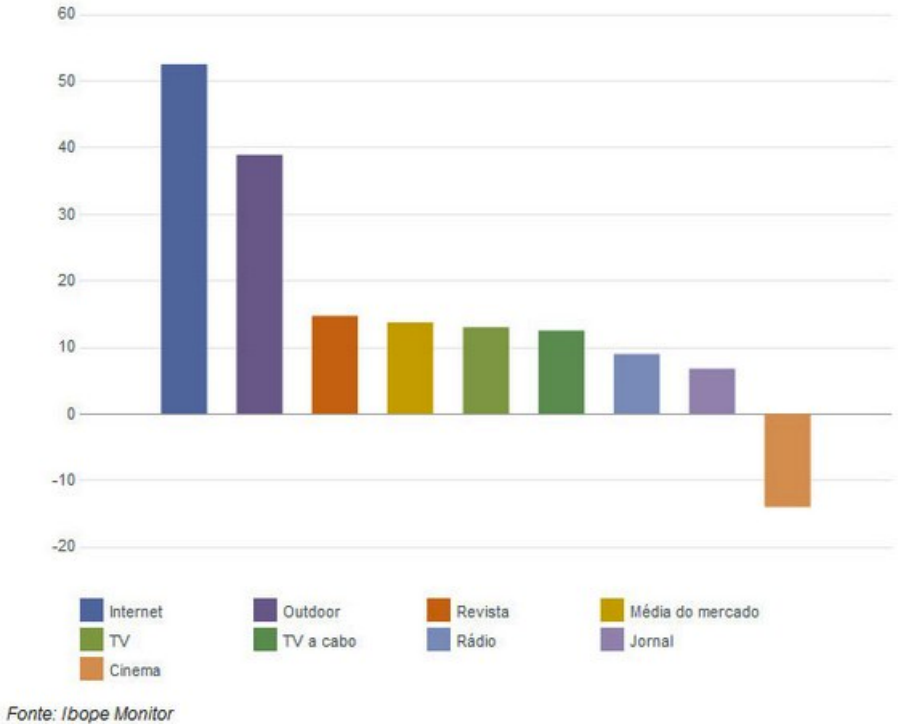

Figura 3.2: Crescimento da Publicidade em Diferentes Meios de Comunicação

paga 1 real por KB de aplicação. Na classe 2, o contratante pagaria 2 reais por KB de aplicação, e assim sucessivamente. Dessa forma, o contratante que optasse pela classe mais alta, teria sua aplicação com maior prioridade inicialmente.

Dado um custo inicial por aplicação, o preço pago passaria a ser em função do número de usuários que vão utilizar aquela aplicação. Dessa forma, as aplicações que tiverem grande repercussão e utilização vão gerar um lucro maior para os contratantes, que deverão pagar um preço maior para a emissora.

Esse modelo força as emissoras a darem maior prioridade às aplicações mais utilizadas pelos usuários e às aplicações de maior classe, já que essas darão maior lucro, diminuindo o tempo de espera da maioria dos usuários. No final, as três partes envolvidas saem beneficiadas.

Esse modelo também abre margem para o controle de uso da aplicação por parte das empresas contratantes, que poderão estipular um limite superior de uso, para não pagarem tão caro. Dessa forma, elas podem controlar o quanto que desejam que suas aplicações sejam utilizadas.

3.3 Formalização do Problema

Definido o modelo de negócio, se faz necessário um aprofundamento matemático para que se possa otimizar o carrossel computacionalmente. Nessa seção estão descritos dois problemas que tem relação direta com o problema abordado, além de

mostrar a formalização do mesmo.

3.3.1 Sequências Justas

O problema de sequências justas primeiramente apareceu nas linhas de montagem da montadora Toyota. Descrito inicialmente como um problema de programação não linear inteira, tem como objetivo minimizar o desvio total do modelo real de produção para a quantidade desejada do modelo de produção determinada pelas demandas $d_1, \ldots, d_n$ para os modelos $1, \ldots, n$ respectivamente (Miltenbug, 1991).

Um problema de sequências justas genérico tem como objetivo minimizar a variação das distâncias entre quaisquer duas ocorrências de elementos idênticos na sequência. Para exemplificar, tomemos um conjunto S formado por $n = 4$ elementos: $S = \{A, B, C, D\}$. A demanda é definida pelo conjunto D, onde d_i é a demanda do símbolo $i = 1, 2, 3, 4$. Caso $d_1 = 4$, a situação perfeita para o elemento A na sequência é como mostra a Figura 3.3

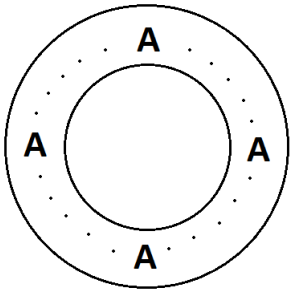

Figura 3.3: Exemplo de Sequência Justa

Caso a demanda de B, C e D também sejam igual a 4, teremos um problema fácil de resolver, tendo a solução trivial mostrada na Figura 3.4

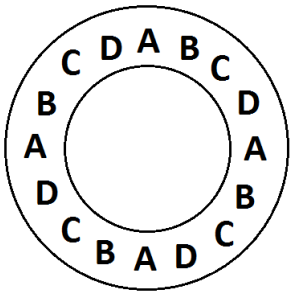

Figura 3.4: Exemplo de Sequência Justa

Porém para demandas aleatórias, o problema se torna complicado. Tomemos como exemplo os seguintes valores para o conjunto de demandas D: $D = \{3, 1, 4, 2\}$ para o mesmo exemplo acima. Uma possível solução é mostrada na Figura 3.5. Porém achar a organização ótima não é uma tarefa trivial. Se tomarmos instâncias com

muitos elementos, será impraticável calcular a solução ótima computacionalmente, já que o problema de sequências pertence à classe de problemas NP-difícil (Kubiak, 2004).

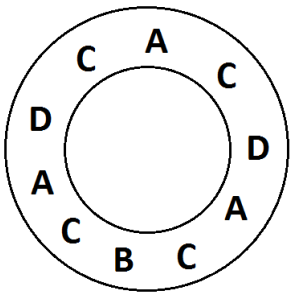

Figura 3.5: Exemplo de Sequência Justa

Posteriormente, Kubiak (2004) definiu como um exemplo de sequência justa o resultado do *Product Rate Variation Problem* (PRV), que é descrito como: Dado um conjunto de n produtos $1, \ldots, i, \ldots, n$, n demandas (inteiros positivos) $d_1, \ldots, d_i, \ldots, d_n$ e n funções convexas e simétricas $f_1, \ldots, f_i, \ldots, f_n$ de variável única, chamada desvio, todas assumindo um mínimo de 0 em 0. A função objetivo é encontrar uma sequência $S = s_1 \ldots s_D, D = \sum_{i=1}^{n} d_i$, de produtos $1, \ldots, i, \ldots, n$, onde o produto i ocorre exatamente d_i vezes que minimiza:

$F(S) = \sum_{i=1}^{n} \sum_{k=1}^{D} f_i(x(S)_{ik} - r_i k)$, onde $x(S)_i k$ = número de ocorrências do produto i no prefixo $s_1 \ldots s_k$, sendo k = $1, \ldots, D$, e $r_i = d_i/D$, $i = 1, \ldots, n$.

3.3.2 Response Time Variability Problem (RTVP)

O *Response Time Variability Problem(RTVP)* é um problema de otimização com uma vasta possibilidade de aplicações. Ele ocorre sempre que eventos, tarefas, clientes ou produtos precisam ser sequenciados para minimizar o tempo de variabilidade que eles esperam para obter os recursos necessários para avançar. Ele é considerado como um sub-problema do problema de Sequências Justas e portanto é considerado também NP-difícil (Corominas, 2007).

O RTVP é formulado como segue. Seja n o número de símbolos, d_i o número de cópias a ser sequenciadas do símbolo i ($i = 1, \ldots, n$) e D o número total de cópias ($\sum_{i=1}^{n} d_i$). Seja s uma solução de uma instância do RTVP que consiste em uma sequência circular de cópias ($s = s_1 s_2 \ldots s_D$), onde s_j é a cópia sequenciada na posição j da sequência s. Para cada símbolo i, em que $d_i \geq 2$, define-se como t_k^i a distância entre as posições em que as cópias $k+1$ e k do símbolo i são encontradas. É considerada a distância entre duas posições consecutivas como igual a 1. Como a sequência é circular, a posição 1 vem imediatamente após a posição D. Portanto, $t_{d_i}^i$ é a distância entre a primeira cópia do símbolo i num ciclo e a última cópia do mesmo símbolo no ciclo precedente.

Seja $\overline{t_i}$ a distância média desejada entre duas cópias consecutivas do símbolo i ($\overline{t_i} = \frac{D}{d_i}$). O objetivo é minimizar a métrica chamada de *response time variability* (RTV), que é definida como a soma dos erros quadrados com respeito às distâncias $\overline{t_i}$. Como os símbolos i cujos $d_i = 1$ não intervém na computação do RTV, assumimos que para cada um desses símbolos o seu t_1^i é igual ao seu $\overline{t_i}$. Sendo assim, RTV é definido pela expressão:

$RTV = \sum_{i=1}^{n} \sum_{k=1}^{d_i} (t_k^i - \overline{t_i})^2$.

3.3.3 O Problema da Variabilidade do Tempo de Download (PVTD)

No Respose Time Variability Problem (Corominas, 2007) é dado um conjunto de símbolos e o problema é construir uma sequência justa dos mesmos, atendendo a todas as demandas e minimizando a distância entre as ocorrências dos símbolos iguais. O Problema da Variabilidade do Tempo de Download, definido neste trabalho, pode ser visto como uma variação do RTVP , onde d_i não é fixo e cada aplicação (símbolo) deve ocorrer ao menos uma vez no carrossel (solução). Além disso, a distância entre as ocorrências de um mesmo símbolo é afetada pelo tamanho das aplicações e não apenas pela sua quantidade. Sendo assim, o PVTD se torna ainda mais complicado do que o Response Time Variability Problem, que é de complexidade NP-difícil (Garey, 1979).

3.4 Modelagem Matemática

O presente modelo matemático é uma extensão da formulação matemática apresentada em [Pessoa, 2008], abrangendo as novas restrições descritas no modelo de negócio detalhado na Seção 3.2. Segue abaixo:

Notação

X: Conjunto de aplicações
N: Numero de aplicações
t_i: Tamanho da aplicacao i em segundos
T_{max}: Tamanho máximo do Carrossel em segundos
q: Número de posições ocupáveis no Carrossel
M: Um número maior que T_{max}
c_i : Classe da aplicação i
v_i: Número de vezes que a aplicação i foi iniciada
k_1: Constante multiplicativa da classe
k_2: Constante multiplicativa do número de vezes que a aplicação foi iniciada

z_i: Prioridade da aplicação i, definidar por:$c_i k_1 + v_i k_2$

Variáveis de Decisão

$$x_{ij} = \begin{cases} 1 & \text{se a aplicação } i \text{ ocorrer na posição } j \\ 0 & \text{caso contrário} \end{cases}$$

d_{ij}^k: Soma dos tempos das aplicações entre as posições i e j, com relação a aplicação k

D_i: Maior tempo (em segundos) entre duas ocorrências consecutivas da aplicação i

max: Maior D_i multiplicado por sua prioridade (z_i)

$$\textit{Minimize } max \tag{3.4.1}$$

$$\text{Sujeito a: } max \geq z_i \times D_i \qquad \forall i \in X \tag{3.4.2}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{q} t_i x_{ij} \leq T_{max} \tag{3.4.3}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j \in [1...q] \tag{3.4.4}$$

$$x_{ij} = x_{i(j+q)} \qquad \forall i \in X \qquad \forall j \in [1...q] \tag{3.4.5}$$

$$d_{kj}^i \geq \sum_{s=1}^{n}\sum_{l=k}^{j-1} t_s x_{sl} - M(1 - x_{ik}) - M(1 - x_{ij}) - \sum_{l=k+1}^{j-1} M x_{il}$$
$$\forall i \in X \qquad \forall k, j \in [1...2q], k \leq j \tag{3.4.6}$$

$$D_i \geq d_{kj}^i \qquad \forall i \in X \qquad \forall k, j \in [1...q], k < j \tag{3.4.7}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in X \qquad \forall j \in [1...2q] \tag{3.4.8}$$

$$D_i, d_{kj}^i \geq 0 \qquad \forall i \in X \qquad \forall k, j \in [1...2q] \tag{3.4.9}$$

$$\sum_{j=1}^{q} x_{ij} \geq 1 \qquad \forall i \in X \tag{3.4.10}$$

Como pode ser observado, o modelo busca otimizar o pior tempo de carregamento de uma aplicação multiplicada por sua prioridade. A prioridade é calculada por dois fatores: a classe da aplicação e o número de vezes que foi utilizada. Com isso, a tendência é que o fator ZxD de cada aplicação seja próximo, e assim o Carrossel seja montado da forma mais justa.

A representação do Carrossel está sendo feita na forma de uma matriz nx$2q$, onde as linhas são as aplicações e as colunas são as posições ocupáveis do carrossel.

Dessa forma, se x_{ij} for verdadeiro, significa que a aplicação i está na posição j do Carrossel. Foram utilizadas duas vezes o número de colunas para que não fosse necessário se girar o carrossel por completo para calcular as distâncias máximas de cada aplicação.

A restrição 3.4.2 está diretamente ligada à função objetivo, enfatizando que o valor de maxD deve ser o maior que todos os (z_ix D_i), para qualquer i. A restrição 3.4.3 está ligada ao limite de tempo que o carrossel pode ter em seu total, isto é, a soma dos tempos de todas as aplicações do carrossel não pode ultrapassar T_{max}. A restrição 3.4.4 garante que em cada posição, apenas e necessariamente uma aplicação estará lá. A restrição 3.4.5 foi feita para que a matriz com $2q$ colunas represente na verdade o carrossel dando dois giros a partir da primeira aplicação. Para isso, foram replicadas as posições a partir da metade até a última.

A restrição 3.4.6 trata justamente de todas as possíveis maiores distâncias entre cada aplicação consigo mesma. O valor de d^i_{kj} é alterado para a distância entre as posições k e j, se a aplicação i estiver nelas, e entre essas posições não houver nenhuma ocorrência da aplicação i. Depois, com a restrição 3.4.7, é calculada a maior distância de cada aplicação para sua possível replicação no carrossel. Caso a aplicação só apareça uma vez no carrossel, a distância será a soma dos tempos das aplicações em cada posição ocupável do carrossel.

A restrição 3.4.8 garante que toda aplicação estará presente no Carrossel. Já as restrições 3.4.9 e 3.4.10 determinam os possíveis valores para cada x, D e d, e também asseguram que as maiores distâncias de cada aplicação devem ser maiores que 0.

Capítulo 4

Metodologia

Neste capítulo serão abordadas as metodologias utilizadas para a concepção e implementação do gerador de carrossel otimizado. É mostrado como foi feita a implementação do GRASP e do ILS para resolver o problema proposto. Além disso, é feita a descrição dos arquivos de entrada e saída utilizados para os testes, além de mostrar a forma que as instâncias foram geradas.

4.1 GRASP Aplicado ao Problema

O GRASP, como descrito anteriormente, é um algoritmo guloso, mas ao mesmo tempo randômico e adaptativo. Tem como característica trazer boas construções, que podem ser utilizadas por algum algoritmo de refinamento.

Para se aplicar o GRASP ao problema, inicialmente um carrosel é construído, contendo todas as aplicações em uma ordem aleatória. Em cada passo do GRASP, o algoritmo insere uma nova aplicação no Carrossel, sendo esta retirada aleatoriamente de uma lista restrita de candidatos (LRC). A lista restrita é composta pelas aplicações que tendem a melhorar mais a função objetivo. Nesse caso, após vários testes com diferentes valores, ficou definido que a lista seria composta pelos 25% melhores aplicativos do momento.

A função objetivo é minimizar o valor de Max, que é o pior $z_i \times D_i$. O z_i é referente a prioridade da aplicação i, e D_i é o valor do tempo máximo de espera da aplicação i. Logo, para se calcular a LRC, em vez de calcular o ganho com a inserção de cada aplicação, foram tomadas as aplicações cujo valor de $z_i \times D_i$ estava mais prejudicando o carrossel.

Devemos notar que a cada passo a lista de candidatos é alterada, para tornar o algoritmo adaptativo e gerando soluções diferentes, mas com uma certa qualidade. A rodada do GRASP para quando são inseridas 2*n aplicações em cima do carrossel inicial. Esse número foi o que mostrou trazer os melhores resultados, tanto no trabalho corrente como no de (Pessoa, 2008). Isso porque se percebe que a medida

que se adiciona mais aplicações, pelo fato da solução ser cíclica, os resultados vão se aproximando de similares alcançados previamente com menos aplicações.

Após a construção, o GRASP ainda é responsável por uma busca na vizinhança de soluções, para alcançar o ótimo local. Essa vizinhança foi definida como a troca entre 2 ou 4 posições no carrossel e a remoção / inserção de uma aplicação em qualquer posição no carrossel. O Pseudo-código do programa pode ser visto no Algoritmo 3

Algoritmo *GRASP(n, Aplicacoes)*
1 melhor $\leftarrow\infty$;
2 **para** $1, 2, ..., 10000$ **faça**
3 $Carrossel \leftarrow$ConstroiCarrosselPadrao();
4 **para** $1, 2, ..., 3n$ **faça**
5 CriaLRC($Carrossel, Aplicacoes$);
6 $Elemento \leftarrow$SorteiaElementoLRC(LRC);
7 InsereElemento($Carrossel, Elemento$);
8 **fim para**
9 aux $\leftarrow$BuscaLocal($Carrossel$);
10 **se** $aux < melhor$ **então**
11 $melhor \leftarrow aux$;
12 **fim se**
13 **fim para**
14 **devolva** $s*$;
fim

Algoritmo 3: GRASP aplicado ao problema

4.2 ILS

O ILS, ao contrário do GRASP, é utilizado quando já se possui alguma solução gerada por um outro algoritmo. Ele é responsável por um refinamento nas soluções existentes, sendo capaz de fugir de ótimos locais e assim trazer melhores resultados. Ele faz isso através de perturbações utilizando vizinhanças definidas.

Para aplicar o ILS ao problema, foram definidos três tipos de vizinhança:

- Vizinhança de Remoção: a aplicação de uma perturbação nessa vizinhança se dá através da remoção aleatória de uma das aplicações no carrossel. Ela deve respeitar a restrição de que toda aplicação deve estar presente no carrossel
- Vizinhança de Troca: a perturbação se dá através da troca de posições entre duas ou quatro aplicações distintas no carrossel
- Vizinhança de Inserção: nesse casso uma nova aplicação é inserida no carrossel. Na perturbação que utiliza essa vizinhança, o algoritmo deve checar se o carrossel não viola a restrição de tamanho máximo do carrossel.

Após a aplicação de uma das perturbações, uma busca local é realizada. Um pseudo-código do ILS aplicado ao problema pode ser visto no Algoritmo 4

Como pode ser observado no algoritmo, as perturbações não têm uma sequência lógica, apenas são aplicadas aleatóriamente. Isso faz com que a solução seja diversificada, e assim o algoritmo consiga fugir de ótimos locais.

Algoritmo *ILS(Carrossel, n)*

```
1  para 1, 2, ..., 50 faça
2      dado ←Random(0,3);
3      se dado == 0 então
4          posicao ←Random(0, n − 1);
5          CarrosselModificado ←RemoveAplicacao(carrossel, posicao);
6      fim se
7      se dado == 1 então
8          posicao1 ←Random(0, n − 1);
9          posicao2 ←Random(0, n − 1);
10         CarrosselModificado ←TrocaPosicoes(carrossel, posicao1, posicao2);
11     fim se
12     se dado == 2 então
13         aplicacao ←Random(aplicacoes());
14         posicao ←Random(0, n);
15         CarrosselModificado ←InsereAplicacao(carrossel, aplicacao, posicao);
16     fim se
17     NovoCarrossel = BuscaLocal (CarrosselModificado);
18     se NovoCarrossel.rate < Carrossel.rate entãoCarrossel ←NovoCarrossel;
19     fim se
20 fim para
21 devolva Carrossel;
fim
```

Algoritmo 4: ILS aplicado ao problema

4.3 Fluxograma de Execução do Algoritmo

Para se implementar o gerador de carrossel otimizado foram necessárias algumas etapas, cuja ordem e organização podem ser vistas na figura 4.1.

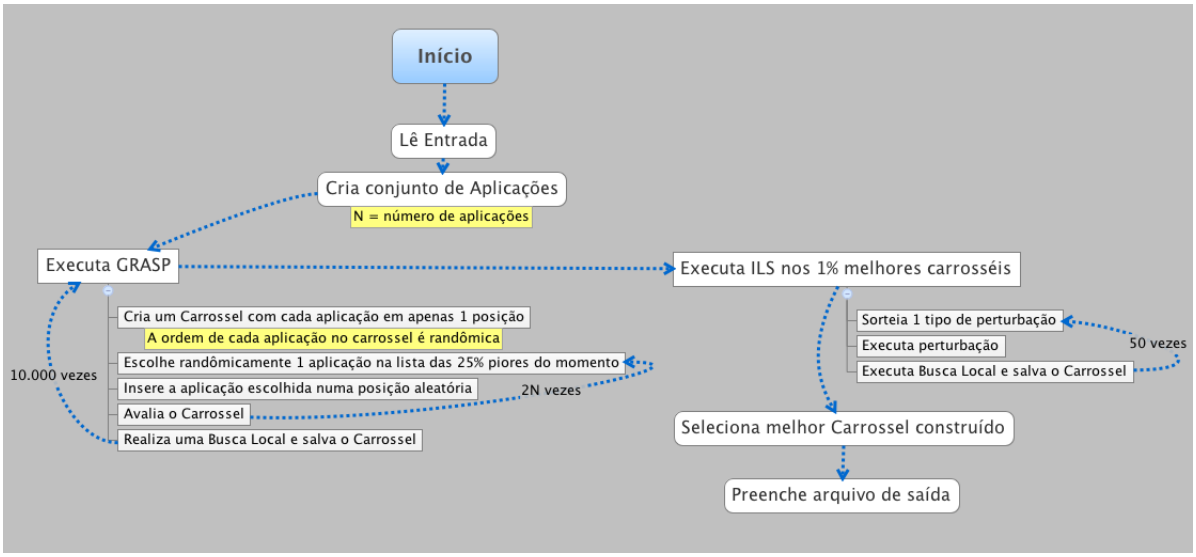

Figura 4.1: Fluxograma do Algoritmo

4.4 Entrada e Saída

As aplicações interativas que vão participar do carrossel deverão estar descritas em um arquivo de texto, que servirá de entrada para o algoritmo. O arquivo contem o número de aplicações, tamanho das aplicações, classes e número de clicks que cada uma já obteve, conforme a sintaxe da figura 4.2.

```
(Número de apps)
Para cada app: Nome | Tamanho em Segundos | Classe | Clicks
```

Figura 4.2: Sintaxe do arquivo de entrada

Um exemplo de um arquivo de entrada pode ser visto na figura 4.3.

```
5
app1 1 3 39
app2 8 1 45
app3 8 2 0
app4 2 1 5
app5 4 5 100
```

Figura 4.3: Exemplo de um arquivo de entrada

Nele, existem 5 aplicações para comporem o Carrossel. A primeira, cujo nome é “app1“, tem tamanho de 1 segundo, é da classe 3, ou seja, o cliente paga 3 vezes mais caro para ter maior prioridade, e já conseguiu 39 acessos dos usuários. As outras aplicações seguem o mesmo raciocínio.

Como saída, devemos ter um arquivo, contendo o número de posições ocupadas no carrossel, além da ordem em que os aplicativos foram dispostos no carrossel. O final desse arquivo também deve conter um número, que será o valor máximo de atraso de uma aplicação, multiplicada a sua prioridade (prioridade = c*k1 + v*k2). Um exemplo desse arquivo pode ser visto na figura 4.4.

```
10
app1 app5 app3 app1 app2 app5 app4 app1 app5 app3
37
```

Figura 4.4: Exemplo de um arquivo de saída

4.5 Geração das Instâncias

Para se testar o algoritmo proposto e realizar comparações com outros métodos, foram utilizados três bancos distintos de instâncias. Todos contém aplicações que respeitam um estudo realizado no Laboratório de Vídeo Digital (Lavid), que possui diversos aplicativos reais de TV Digital. Os bancos foram divididos conforme segue abaixo:

- **Instâncias A1:** Esse banco é composto pelas instâncias utilizadas na comparação com o algoritmo exato. Ele possui 5 instâncias de tamanhos n = 3, 5, 7, 10 e 15, onde n é o número de aplicações presentes no carrossel. As instâncias podem ser visualizadas nos Anexos.

- **Instâncias A2:** Esse banco de instâncias foi utilizado na comparação com o algoritmo de Pessoa (2008). Para poder comparar os resultados de forma justa, foi utilizado o mesmo banco de instâncias do trabalho de Pessoa (2008), presente nos anexos do referido trabalho. Para este caso, foi utilizado o modelo de negócio presente no trabalho de Pessoa (2008). Este banco possui 15 instâncias, onde cada grupo de cinco consecutivas possui tamanho n = 10, 15 e 20 respectivamente.

- **Instâncias A3:**Esse banco foi utilizado na comparação do algoritmo proposto com um algoritmo genérico. A geração das instâncias se deu com a cópia das 10 primeiras instâncias de Pessoa (2008). Como o mesmo não levou em consideração o número de acessos por aplicação, esse número foi inserido de forma aleatória, sendo que nenhuma aplicação tinha número de acessos maior do que 700. Este banco possui 10 instâncias, onde as cinco primeiras possuem tamanho n = 10 e as cinco últimas n = 15.

Capítulo 5

Resultados

Após a implementação dos algoritmos aproximados propostos e da técnica exata, foram realizadas algumas comparações que serão detalhados nas próximas seções. Para os parâmetros de entrada, foram definidos os seguintes: k_1 (constante multiplicativa da classe) = 1; k_2 (constante multiplicativa do número de acessos da aplicação) = 0.01. O primeiro valor significa que se uma aplicação tem sua classe aumentada em 1, ela terá também sua prioridade aumentada em 1. Já o segundo valor significa que são necessários 100 acessos para que a aplicação tenha sua prioridade aumentada em 1.

5.1 Descrição do Ambiente

O hardware utilizado durante os testes do gerador de carrossel otimizado foi um MacBook Pro, processador Core 2 Duo 2.4GHz, 4GB de memória RAM DDR3, com o sistema operacional Mac OS X.

O algoritmo exato foi escrito no CPLEX for MAC, versão obtida com o licenciamento da IBM Academic Initiative. Já o algoritmo aproximado foi escrito na linguagem Java, com o uso da IDE Netbeans, versão 7.0.

5.2 Comparação com o Algoritmo Exato

Para se comparar com o algoritmo Exato, foi utilizado o banco de instâncias **A1**. Todas as instâncias presentes nesse banco podem ser visualizadas nos Anexos. Os resultados obtidos comprovaram a ineficácia de algoritmos exatos para instâncias de tamanho acima de 7, pois é necessário a geração de um carrossel de forma rápida, já que os dados (número de acessos) estarão em constante mudança durante uma transmissão. A comparação entre os resultados e tempos pode ser vista na Tabela 5.1.

Como pode ser observado, as metaheurísticas quando aplicadas a instâncias relativamente grandes são capazes de retornar resultados bastante satisfatórios em um espaço curto de tempo

Tabela 5.1: Comparação com a Implementação Exata

Instância	Solução Cplex	Tempo(s)	Solução GRASP +ILS	Tempo(s)
Instância de tamanho 3	53571	0.34	53571	0.033
Instância de tamanho 5	78870	9.89	78870	0.793
Instância de tamanho 7	101776	30.313	101776	1.524
Instância de tamanho 10	234010*	1352.78	227230	3.611
Instância de tamanho 15	371295*	2485	355940	8.22

***Melhor solução encontrada em até 7200 segundos**

5.3 Comparação Com o Algoritmo de Pessoa (2008)

Por ter um modelo de negócio diferente de Pessoa (2008), não foi possível comparar a implementação do trabalho corrente com nenhum outro encontrado na literatura. Diante disso, se fez necessário implementar os algoritmos propostos para o modelo de negócio já explorado, presente no trabalho de Pessoa (2008), para então ser feita uma comparação entre os algoritmos.

No trabalho de Pessoa (2008), a função objetivo é maximizar o lucro da emissora, e não leva em consideração o nível de satisfação dos telespectadores. As instâncias foram as do banco **A2**, descrito na Seção 4.5. A Tabela 5.2 mostra a comparação entre os resultados obtidos.

Os resultados obtidos foram bastante satisfatórios, pois além de manter a qualidade da solução, com uma melhora não significativa de 0,16%, alcançou os resultados com um tempo médio 72% menor. Quando analisadas instâncias com 20 aplicações (instâncias 11 até 15), o ganho com o tempo é ainda mais significativo, melhorando em 80%.

Tabela 5.2: Comparação com Pessoa (2008)

# Instância	Resultado(R$) [Pessoa, 2008]	Tempo (s)	Resultado Corrente (R$)	Tempo (s)
1	513,041.36	0.50	519,004.00	1.23
2	856,288.85	0.40	864,912.00	1.45
3	665,352.83	0.60	668,630.00	1.65
4	695,735.43	0.80	695,760.50	1.43
5	596,337.41	0.80	597,085.75	1.65
6	862,590.53	4.90	864,402.00	1.98
7	917,090.16	4.20	917,151.00	2.02
8	936,838.61	5.10	936,838.61	1.89
9	798,168.60	6.20	802,603.00	2.1
10	929,091.26	2.20	929,091.25	2.16
11	1,086,103.89	23.40	1,086,103.89	5.65
12	1,102,782.58	31.00	1,102,782.58	5.23
13	1,118,117.48	20.50	1,118,246.00	6.62
14	1,132,070.52	28.80	1,132,070.52	4.75
15	1,026,302.19	27.70	1,023,192.76	4.56
Total	13,235,911.70	157.1	13,257,873.86	44.37

5.4 Comparação com um Algoritmo de Geração Genérico

Atualmente a maioria das emissoras, se não todas, não utilizam nenhuma abordagem de otimização para a criação dos carrosséis. Esse dado foi constatado ao analisar fluxos de vídeos de diferentes emissoras no Lavid. Isso é devido ao pequeno número de aplicações presentes simultaneamente durante a transmissão. A comparação feita nesta seção busca mostrar a ineficácia dessa abordagem para instâncias relativamente grandes.

As instâncias utilizadas foram as presentes no banco **A3**, cuja descrição pode ser vista na seção 4.5. Como pode ser obsevado na Figura 5.1, há uma boa diminuição do valor da função objetivo quando aplicado o algoritmo proposto. Para todas as diferentes instâncias, houve alguma melhora considerável.

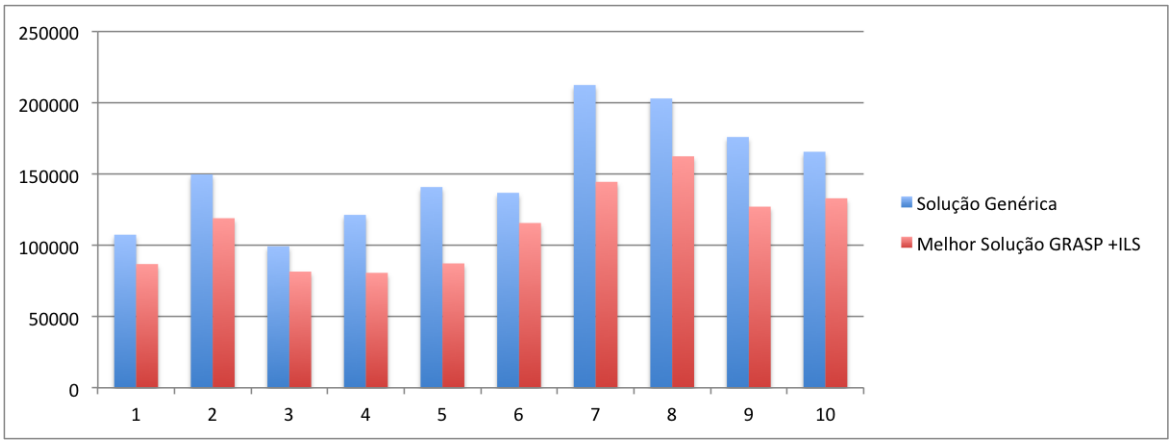

Figura 5.1: Comparação Gráfica com um Algoritmo Genérico

O tempo médio de execução do algoritmo nas diferentes instâncias foi de 1,98 segundos, o que é considerado baixo, perto da demora no cálculo da solução exata. O tempo de execução de cada instância bem como a melhora em percentual do algoritmo GRASP + ILS podem ser observados na Tabela 5.3. O algoritmo foi executado 5 vezes para cada instância e o resultado mostra tanto a média como o melhor caso.

Analisando os resultados, observa-se que a aplicação de um algoritmo inteligente é capaz de melhorar a solução em até 38% (Instância 5). Sendo assim, utilizar a abordagem de geração do carrossel genérica tende a ser bastante ineficiente quando estivermos com um cenário de muitas aplicações sendo transmitidas simultaneamente.

Tabela 5.3: Comparação com um Algoritmo de Geração Genérico

Instância	Solução Genérica	Melhor Solução GRASP +ILS	Tempo(s)	Solução Média GRASP +ILS	Tempo Médio (s)
1	107456	86856	1.13	89946	1.09
2	149787	119036	1.07	119036	1.09
3	99324	81170	1.20	83230	1.22
4	121374	80700	1.42	83902	1.26
5	140910	86045	1.12	89540	1.08
6	136854	114534	2.22	119673	2.02
7	212510	144557	2.51	148565	2.71
8	203050	160152	2.83	164366	2.68
9	176020	127140	3.65	128282	3.89
10	165670	133014	2.52	135234	2.74
Média	151295.5	113320.4	1.96	116177.4	1.98

O algoritmo se mostrou bastante robusto, pois as soluções encontradas na média se aproximaram muito da melhor solução em 5 execuções. A Figura 5.2 mostra exatamente essa diferença, para as 10 instâncias utilizadas na comparação com o algoritmo genérico.

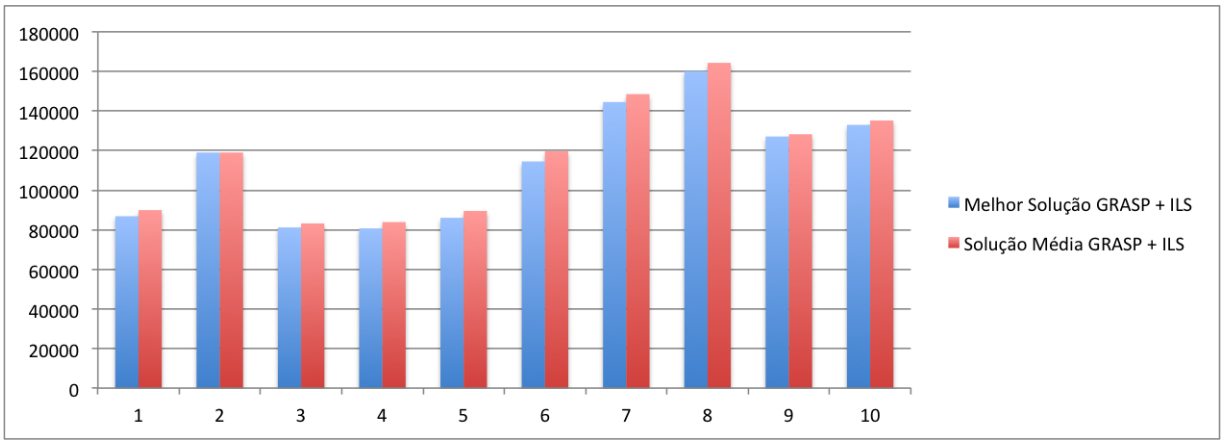

Figura 5.2: Comparação com a Média das Execuções

5.4.1 Comparação entre as Fases de Construção e de Refinamento do GRASP

Durante as execuções dos algoritmos, foram analisados separadamente o desempenho da fase de construção do GRASP e sua fase de refinamento. Os resultados obtidos nessa seção são de acordo com a média das 5 execuções. A Tabela 5.4 mostra que o refinamento chegou a melhorar 17,34% da solução no melhor caso (Instância 4) e 4,11% no pior caso (Instância 1). Assim, pode-se comprovar que o mesmo foi bastante eficiente para o algoritmo proposto.

Tabela 5.4: Comparação entre as Fases de Construção e de Refinamento do GRASP

Instância	GRASP: Construção	GRASP: Construção + Refinamento	Contribuição Refinamento (%)
1	96624	92804	4.11
2	132032	124030	6.45
3	97426	88540	10.03
4	102345	87220	17.34
5	105232	93720	12.28
6	132854	122677	8.29
7	170223	152557	11.57
8	183254	172468	6.25
9	154030	133260	15.58
10	156368	140680	11.15

O custo para o refinamento considerando o tempo de processamento foi relativamente baixo, se comparado ao tempo levado para se construir uma solução. Esse dado pode ser constatado na Figura 5.3. O tempo levado para se realizar o refinamento é em média 1/4 menor do que o tempo necessário para construção do GRASP.

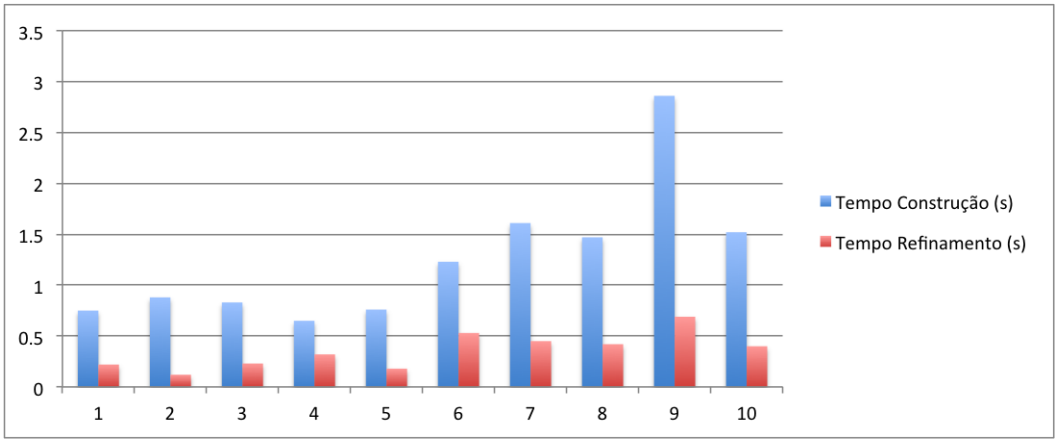

Figura 5.3: Comparação entre os Tempos da Construção e do Refinamento

5.4.2 Comparação entre os Algoritmos GRASP e ILS

Também foi analisado o desempenho do GRASP em relação ao algoritmo de refinamento ILS. O GRASP por si só já trazia resultados bastante satisfatórios, porém ainda assim se conseguiu melhorar significativamente a solução com o ILS. A melhora média (para 5 execuções) obtida em cada instância pode ser observada na Tabela 5.5

Tabela 5.5: Contribuição do Algoritmo ILS para cada Instância

Instância	GRASP	GRASP + ILS	Contribuição ILS (%)
1	92804	89946	3.17
2	124030	119036	4.19
3	88540	83230	6.37
4	87220	83902	3.95
5	93720	89540	4.66
6	122677	119673	2.51
7	152557	148565	2.68
8	172468	164366	4.92
9	133260	128282	3.88
10	140680	135234	4.02

Por ser aplicado apenas nas melhores instâncias da fase de construção, o algoritmo de ILS tem um tempo de execução bastante reduzido, se comparado ao GRASP. Esse dado pode ser observado na Figura 5.4. Esse foi o tempo de execução médio de cada algoritmo nas 5 execuções.

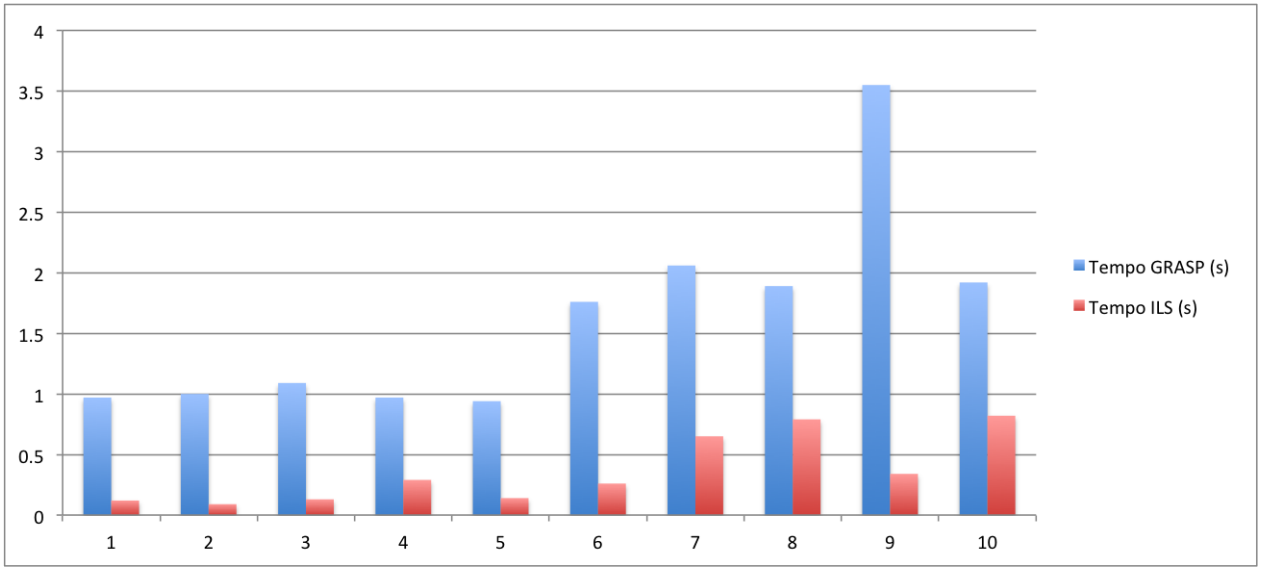

Figura 5.4: Comparação entre os Tempos do GRASP e do ILS

Capítulo 6

Conclusão

O aumento eminente de aplicações pra TV Digital pode trazer um novo problema às emissoras, que terão que se preocupar com a forma de organizar o Carrossel de aplicações. Assim, o trabalho corrente tende a oferecer uma boa ferramenta para que, em um tempo hábil, se calcule um carrossel de qualidade.

Além disso, é de grande importância se pensar na qualidade do serviço para a maioria dos usuários. Ignorar a alta repercussão de uma aplicação pode causar insatisfação de uma grande quantidade de telespectadores. Assim, o trabalho proposto traz um modelo justo que visa satisfazer tanto as empresas contratantes que pagam mais, quanto os usuários.

Após definido o modelo, surge então o Problema da Variabilidade no Tempo de Download aplicado ao ambiente de TV Digital. Para resolvê-lo, o trabalho corrente utilizou as metaheurísticas GRASP e ILS, que se mostraram bastante eficientes, trazendo resultados em curto espaço de tempo, mesmo para instâncias consideravelmente grandes. Ao comparar o algoritmo proposto com outro da literatura, percebeu-se a eficiência dos resultados alcançados, pois alcançou resultados de igual qualidade em uma média de tempo 72% menor.

Quando comparamos o algoritmo proposto com a forma com que os carrosseis são gerados hoje em dia no Brasil, percebemos que um grande ganho pode ser obtido com a utilização de algoritmos de otimização. Os algoritmos genéricos são ineficazes para instâncias de tamanhos considerados e por isso certamente no futuro as emissoras vão buscar otimizar a forma de gerar o carrossel contendo as aplicações transmitidas.

Como trabalho futuro, intenciona-se incorporar uma estratégia de busca local exata, explorando a formulação matemática.

Referências Bibliográficas

[1] ABNT NBR 15601. (2007), Revisada em 07.04.2008. Televisão digital terrestre - Sistema de transmissão

[2] ABNT NBR 15604 (2007), revisada em 07.04.2008. Televisão digital terrestre — Receptores

[3] Balabanian, V. (1996). An Introduction to Digital Storage Media - Command and Control (DSM- CC). Disponível em < http://www.chiariglione.org/mpeg/tutorials/papers/dsmcc/dsmcc.htm >. Acessado em Abril de 2009

[4] Becker, V. (2009). TV Digital: História. Disponível em: < http://ube-167.pop.com.br/repositorio/37874/meusite/tvdigital/historia.html >. Acessado em Dezembro de 2009

[5] Corominas, A., Kubiak, W., Moreno, N. (2007). Response time variability. Journal of Scheduling, Vol. 10, pp. 97-110.

[6] ETSI TR 101 202 V1.2.1. (2003). Digital Video Broadcasting (DVB); Implementation guidelines for Data Broadcasting

[7] Festa, P., Resende, M. G. C., (2002). "GRASP: an annotated bibliography", Essays and Surveys on Metaheuristics (C.C. Ribeiro e P. Hansen, editores), pp. 325-367, Kluwer Academic Publishers

[8] Glover, F., Kochenberger, G. (2003). Handbook of Metaheuristics. p. xi - xii

[9] Garey, M. R., Johnson, D. S., (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Co., San Francisco.

[10] Herrero, C. and Vuorimaa , P. (2003). Optimization Tecniques for Digital Television Applications Broadcastin

[11] ISO/IEC 13818-6, 12-July-1996. Digital Storage Media Command & Control

[12] Kubiak, W., (2004). Fair sequences. In Handbook of Scheduling: Algorithms, Models and Performance Analysis , Leung, J.Y-T., editor, Chapman & Hall/CRC, Boca Raton, Florida.

[13] Lourenço, H. R., Martin, O. C., Stützle, T., (2001) Iterated Local Search – Disponível no Handbook on MetaHeuristics e em: < http://arxiv.org/pdf/math/0102188.pdf >

[14] Médola, A. (2009). Televisão digital e interatividade: uma demanda da convergência midiática. ECO-Pós, v.12, n.2, maio-agosto 2009, p. 7-11

[15] Miltenburg, J.G. (1989). Level schedules for mixed-model assembly lines in just-in-time production systems. Management Science 35, 192 - 207.

[16] Monden, Y. (1983). Toyota Production Systems. Industrial Engineering and Management Press, Norcross, Ga.

[17] Pessoa, B. (2008). Metaheurísticas Aplicadas à Geração de Carrossel no Sistema Brasileiro de Tv Digital. Webmedia - Proceedings of the 14th Brazilian Symposium on Multimedia and the Web

[18] Pereira, L. A., Bezzera, E. P. (2008). Televisão Digital: do Japão ao Brasil - Culturas Midiáticas - Ano I

[19] Resende, M. (1998) "Greedy Randomized Adaptative Search Procedures (GRASP)". Technical Report, ATT Labs Research

[20] Schrijver, A. (1988). Theory of Linear and Integer Programming

[21] Souza, M. J.. (2011). Inteligência Computacional para Otimizaçàò, Notas de aula, 2011/1 - Universidade Federal de Ouro Preto

[22] Tien-Ying Kuo, Yi-Chung Lo and Chih-Chun Lai (2008). Optimal Carousel Stream for Interactive Digital TV Service - 2008 IEEE international symposium on broadband multimedia systems and broadcasting

[23] Vinoski, S. (1997). CORBA: Integrating diverse applications within distributed heterogeneous enviroments. Communications Magazine, IEEE

[24] Yamada, F., (2004) Sistema de TV Digital. In Revista Mackenzie de Engenharia e Computação – Ano 5, Número 5.

Apêndice A

Instâncias A1

Tabela A.1: Instância 1

Índice	Tamanho	Classe	Acessos
1	1232	13	190
2	7653	5	230
3	2321	6	110

Tabela A.2: Instância 2

Índice	Tamanho	Classe	Acessos
1	753	4	520
2	5032	3	110
3	403	5	55
4	3201	9	130
5	2102	3	330

Tabela A.3: Instância 3

Índice	Tamanho	Classe	Acessos
1	720	10	320
2	3940	5	130
3	1302	7	511
4	330	3	302
5	203	7	600
6	4002	6	200
7	2032	8	92

Tabela A.4: Instância 4

Índice	Tamanho	Classe	Acessos
1	2032	2	212
2	1023	7	450
3	4212	8	206
4	2032	5	103
5	452	10	131
6	4233	5	220
7	5302	10	54
8	902	3	201
9	5032	6	402
10	3402	2	220

Tabela A.5: Instância 5

Índice	**Tamanho**	**Classe**	**Acessos**
1	4801	5	300
2	890	8	190
3	5045	4	630
4	2303	2	350
5	1110	1	220
6	1103	10	50
7	3203	8	110
8	405	10	50
9	7021	8	110
10	3045	9	90
11	556	2	520
12	3402	2	140
13	2312	3	210
14	455	7	430
15	4532	6	30

Apêndice B

Instâncias A3

Tabela B.1: Instância 1

Índice	Tamanho	Classe	Acessos
1	2066	1	87
2	1524	2	342
3	2031	7	120
4	2279	6	44
5	2048	2	410
6	2045	2	231
7	889	2	504
8	813	5	24
9	672	1	211
10	1096	6	109

Tabela B.2: Instância 2

Índice	Tamanho	Classe	Acessos
1	1095	2	82
2	1637	4	201
3	1619	4	32
4	1417	9	131
5	2244	8	312
6	762	9	56
7	2470	8	52
8	2427	3	123
9	374	9	56
10	1816	6	212

Tabela B.3: Instância 3

Índice	Tamanho	Classe	Acessos
1	1096	7	130
2	2158	3	230
3	244	2	342
4	1463	5	12
5	679	4	302
6	1502	5	102
7	364	9	87
8	1688	10	90
9	772	2	340
10	1434	8	24

Tabela B.4: Instância 4

Índice	Tamanho	Classe	Acessos
1	649	1	532
2	1721	5	13
3	1418	3	43
4	1640	8	304
5	746	10	28
6	1761	8	203
7	2195	6	44
8	2078	3	66
9	287	1	309
10	179	9	12

Tabela B.5: Instância 5

Índice	Tamanho	Classe	Acessos
1	899	9	23
2	1717	4	302
3	307	10	23
4	1534	10	34
5	1531	1	382
6	1540	6	23
7	1959	2	38
8	1045	8	231
9	1844	1	410
10	2022	1	75

Tabela B.6: Instância 6

Índice	Tamanho	Classe	Acessos
1	1101	4	203
2	325	4	154
3	254	3	230
4	1282	1	290
5	842	5	232
6	1863	5	13
7	256	2	182
8	2428	9	78
9	529	9	92
10	2063	4	80
11	341	7	231
12	745	5	203
13	245	6	66
14	1817	9	82
15	1456	8	108

Tabela B.7: Instância 7

Índice	Tamanho	Classe	Acessos
1	265	2	540
2	770	8	201
3	2138	5	232
4	1975	7	21
5	1525	5	98
6	770	6	87
7	957	1	163
8	1192	1	280
9	182	10	48
10	1782	9	74
11	1111	1	302
12	2041	6	290
13	2489	9	89
14	2105	2	134
15	2131	2	178

Tabela B.8: Instância 8

Índice	Tamanho	Classe	Acessos
1	1655	7	293
2	1309	6	230
3	1958	1	40
4	2437	2	180
5	2203	5	122
6	1987	4	160
7	1666	8	65
8	1074	6	94
9	915	9	86
10	1647	7	312
11	715	4	236
12	380	8	187
13	312	7	123
14	1889	9	28
15	1805	4	68

Tabela B.9: Instância 9

Índice	**Tamanho**	**Classe**	**Acessos**
1	1286	9	56
2	587	1	344
3	939	8	21
4	2251	7	156
5	1881	5	178
6	2320	1	192
7	1846	4	54
8	558	3	350
9	562	10	38
10	1515	3	128
11	573	6	196
12	679	10	38
13	303	9	130
14	347	3	350
15	2258	4	128

Tabela B.10: Instância 10

Índice	**Tamanho**	**Classe**	**Acessos**
1	394	8	169
2	186	3	420
3	1034	9	87
4	648	10	67
5	194	10	48
6	1838	8	230
7	765	10	78
8	1059	4	430
9	1866	8	21
10	563	7	130
11	2486	1	156
12	1834	2	187
13	1926	9	88
14	1075	7	198
15	893	1	230

METAHEURÍSTICAS GRASP E ILS APLICADAS AO PROBLEMA DA VARIABILIDADE DO TEMPO DE DOWNLOAD APLICADO AO AMBIENTE DE TV DIGITAL

Daniel Gonçalves Ramos
Universidade Federal da Paraíba
CCEN – Cidade Universitária, João Pessoa
danielramos@lavid.ufpb.br

Bruno Jefferson de S. Pessoa
Universidade Federal da Paraíba
bruno.pessoa@di.ufpb.br

Lucídio dos Anjos F. Cabral
Universidade Federal da Paraíba
lucidio@di.ufpb.br

Guido Lemos
Universidade Federal da Paraíba
guido@lavid.ufpb.br

RESUMO

A transmissão de aplicativos interativos no ambiente de TV Digital é feita através do padrão Carrossel DSM-CC. Esse padrão permite que os dados sejam enviados de forma cíclica, a fim de que o usuário possa recebê-los por completo independente do momento que iniciou seu download. A forma com que os dados estarão disponíveis no carrossel tem um impacto significativo no tempo de espera do usuário, o que dá origem a uma variante do *Response Time Variability Problem*, denominado de Problema da Variabilidade do Tempo de Download (PVTD). Este artigo propõe um modelo de negócio para definir quais aplicações terão prioridade no carrossel, além de propor um novo modelo matemático que visa minimizar o atraso no download dessas aplicações. O trabalho corrente ainda descreve a implementação das metaheurísticas GRASP e ILS aplicadas ao PVTD, bem como compara resultados com trabalhos similares na literatura.

PALAVARAS CHAVE. Carrossel DSM-CC, Aplicações Interativas, GRASP, ILS.

Pesquisa Operacional, TV Digital

ABSTRACT

Transmissions of interactive television applications are made using the DSM-CC Carousel protocol. This standard enables data to be sent cyclically so that a user can receive all data transmitted, no matter the start time of the download. The way each interactive application is disposed on the carousel has impact on users' waiting times, which gives rise to a variant of the Response Time Variability Problem, called Download Time Variability Problem (DTVP). This article proposes a business model to define which applications will have priority in a carousel and a new mathematical model to minimize the users' waiting times. This work describes the implementation of GRASP and ILS metaheuristics applied to the DTVP, and compares the result to similar works in the literature.

KEYWORDS. DSM-CC Carousel, Interactive Applications, GRASP, ILS

Operational Research, Digital TV

1. Introdução

O forte avanço na TV Digital proporcionou um aumento significativo nas pesquisas na área. A possibilidade de interação com a TV criou uma enorme expectativa acerca dessa nova tecnologia. Dessa forma, as pessoas além de assistirem seus programas prediletos em alta definição, podem usar a TV como meio para fazer compras, alugar filmes, opinar em enquetes, entre outras infinidades de formas de interação.

Tudo isso só foi possível graças ao uso de poderosos algoritmos de compressão (MPEG). O vídeo que antes tinha baixa qualidade e ocupava uma largura de banda de 6MHz, agora é transmitido em alta definição e ainda há sobras nessa mesma banda de 6MHz [ABNT, 2007]. Com isso, é possível transmitir, além do vídeo, variedades de informações, que vão de aplicações interativas até outros vídeos em menor qualidade.

O caráter de transmissão de dados em carrossel, ou seja, o mesmo conteúdo sendo enviado para todos os receptores de forma cíclica, permite que todos os receptores, independente da hora que se conectarem a um canal, sejam capazes de capturar os dados que estão sendo enviados em um determinado momento. Esse modelo de transmissão é ilustrado na Figura 1.

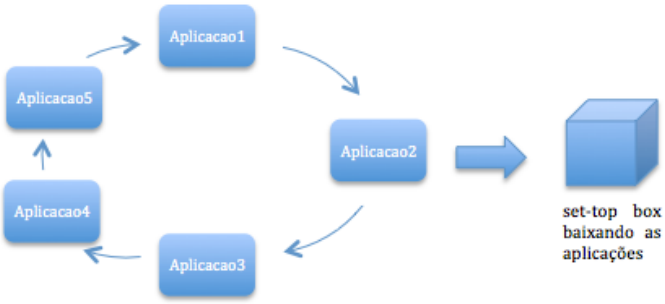

Figura 1 - Carrossel em TV Digital

Porém a forma com que o carrossel vai estar organizado pode acarretar em grandes atrasos por parte dos receptores, que precisarão esperar um tempo elevado para carregar o aplicativo desejado. O ideal seria que os *set-top boxes* tivessem memória ilimitada, de forma que toda aplicação transmitida pela emissora pudesse ficar armazenada nos mesmos. Mas essa situação está longe da realidade no Brasil, dado que, por questões econômicas, os *set-top boxes* produzidos aqui possuem pouca memória de armazenamento.

Sendo assim, é possível favorecer algumas aplicações dentro do carrossel, melhorando a experiência da maioria dos usuários e satisfazendo as empresas contratantes e a emissora. Para isso, é necessário se definir um modelo de negócio justo. O trabalho propõe um modelo inovador, e traz a modelagem matemática para o mesmo.

Definidas as prioridades das aplicações interativas dentro do carrossel, o problema passa a ser como organizar essas aplicações para refletir o modelo definido. Esse problema pode ser entendido como uma variação do *Response Time Variability Problem* (Corominas, 2009), o qual pertence à classe de problemas chamada sequências justas, que emergiu paralelamente ao problema de escalonamento, uma variante dos problemas de escalonamento. O objetivo desse problema é a construção de sequências justas utilizando n símbolos, onde o símbolo $i (i = 1, \ldots, n)$ deve ocorrer d_i vezes numa sequência e cada símbolo de qualquer subsequência está alocado em posições justas (Kubiak, 2004).

Este trabalho está organizado da seguinte forma: a seção 2 descreve o funcionamento do padrão DSM-CC; a seção 3 traz uma descrição mais detalhada do PVTD; na seção 4, é feita uma revisão da literatura e posteriormente a seção 5 mostra detalhes do novo modelo de negócio proposto; a modelagem matemática é mostrada na seção 6, seguida pela implementação das metaheurísticas GRASP e ILS, mostrada na seção 7; por fim, nas seções 8 e 9 são mostrados os resultados e as considerações finais, respectivamente.

2. O Video TS e o Padrão DSM-CC

Para se transmitir um fluxo de áudio e vídeo utilizando o padrão MPEG-2 é necessário

que esses dados sejam empacotados e multiplexados, de forma que mais de um fluxo seja transmitido simultaneamente. Isso é possível graças a um robusto padrão de controle, que faz com que cada fluxo elementar tenha apenas um tipo de informação, como áudio, vídeo, ou dados.

Cada pacote que forma um fluxo elementar é formado por um cabeçalho e um *payload*, que é a carga útil desse pacote. O cabeçalho possui informações que identificam o mesmo unicamente no fluxo, e a qual fluxo ele pertence. A informação mais importante, que define o fluxo do pacote, é o PID (Packet ID).

Já o padrão DSM-CC foi desenvolvido para dar suporte à transmissão de serviços multimídia. Um protocolo aberto é essencial para a entrega em larga escala desses serviços. A possibilidade de envio de sistemas de arquivos completos para um receptor é um dos principais motivos do sucesso desse padrão (Balabanian, 1996).

Os dados carregados no Carrossel provém normalmente de um único PID, havendo instâncias mais complexas que o carregam em mais de um PID. Para o primeiro caso, identificar os pacotes do carrossel se torna fácil, bastando apenas criar um filtro para um determinado PID.

3. Problema da Variabilidade do Tempo de Download

A forma com que as aplicações vão estar dispostas no Carrossel tem um impacto importante na espera dos usuários. Favorecer algumas aplicações pode melhorar a experiência da maioria deles, além de satisfazer as empresas que pagam mais pela veiculação de suas aplicações. Dentro de um carrossel, uma aplicação pode ter prioridade sobre as outras. Para tal, basta replicar as aplicações mais importantes dentro do carrossel. A figura 2.0 mostra um exemplo onde a aplicação vermelha tem maior prioridade sobre as demais. Assim, o tempo máximo que o usuário espera para carregar a aplicação com prioridade pode cair consideravelmente.

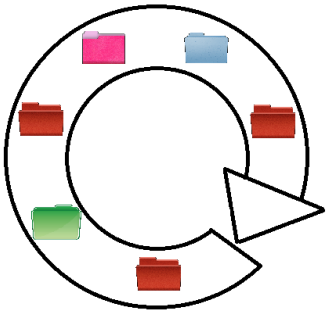

Figura 2 - Carrossel com Prioridade

Definir quais aplicações devem ter prioridade no carrossel é uma tarefa complicada, já que isso deve respeitar o contrato das emissoras, e as aplicações mais utilizadas pelos usuários devem ter um tempo de espera menor. Além disso, a prioridade pode ser alterada instantaneamente, caso a aplicação esteja recebendo muitos acessos. Esse problema requer que seu modelo de negócio seja justo e atenda as necessidades descritas.

Dadas as prioridades das aplicações, o problema passa a ser gerar o carrossel de forma otimizada (Figura 3). Isso também é uma tarefa complexa, já que existe um número exponencial de formas para organizar as aplicações, visto que as mesmas podem ser replicadas inúmeras vezes. Para se ter noção desse número, supondo que existem n aplicações que podem ser colocadas em $q \geq n$ posições no carrossel, o número de combinações possíveis é maior do que $q! \times n!$.

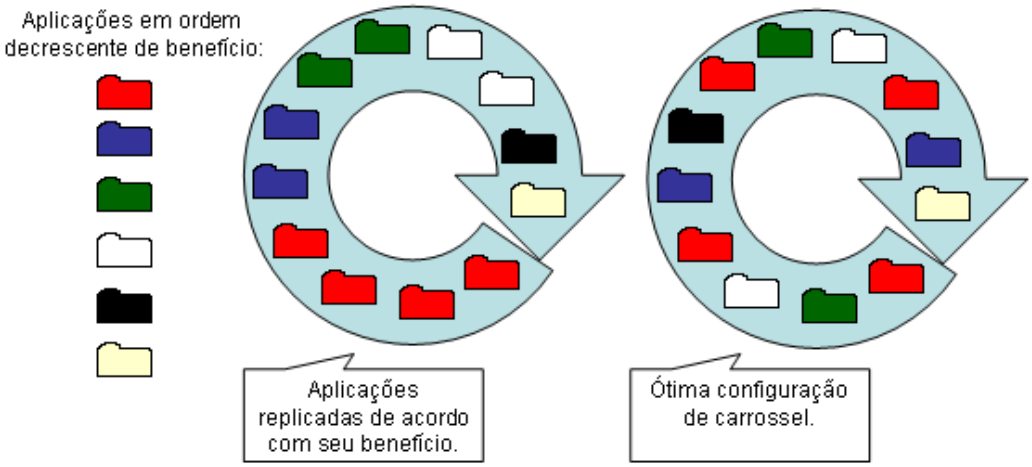

Figura 3 - Carrossel ótimo

Dessa forma, testar todas as combinações possíveis de se formar um carrossel ótimo, dadas as aplicações e suas prioridades, demoraria um tempo proibitivo, caso se tenha uma quantidade considerável de aplicações. Assim, é necessário se aplicar um algoritmo inteligente, capaz de gerar boas soluções em um tempo aceitável.

No *Respose Time Variability Problem* (Corominas, 2009) é dado um conjunto de símbolos e o problema é construir uma sequência justa dos mesmos, atendendo a todas as demandas e minimizando a distância entre as ocorrências dos símbolos iguais. O Problema da Variabilidade do Tempo de Download pode ser visto como uma variação deste, onde d_i não é fixo e cada aplicação (símbolo) deve ocorrer ao menos uma vez no carrossel (solução). Além disso, a distância entre as ocorrências de um mesmo símbolo é afetada pelo tamanho das aplicações e não apenas pela sua quantidade. Sendo assim, o PVTD se torna ainda mais complexo do que o *Response Time Variability Problem*, que é de complexidade NP-difícil (Garey, 1979)

4. Trabalhos Relacionados

Buscando otimizar a organização das aplicações dentro de um carrossel DSM-CC, apenas um trabalho foi encontrado na literatura. Esse trabalho teve grande influência durante todo o processo de desenvolvimento descrito neste artigo. Para Pessoa (2008), o carrossel deveria ser gerado de forma a otimizar o lucro da emissora, propondo um modelo para contratação dos serviços de propaganda por meio de aplicativos interativos. Com esse modelo, são definidas as prioridades dos aplicativos de forma estática, ou seja, não leva em consideração a utilização das aplicações durante a transmissão.

Como consequência, uma única formação do carrossel é construída para transmissão durante cada programa. As prioridades de cada aplicação são determinadas a partir do quanto se ganha pelo envio da mesma. Além disso, uma multa é calculada em cima do valor pago pela empresa contratante, caso o atraso máximo seja superior a um valor fixado.

O trabalho de Pessoa (2008) utiliza as metaheurísticas GRASP + VND. O calculo do ganho da inserção de uma aplicação no carrossel levou em conta o valor pago pela mesma, o tamanho e o número de vezes que ela já apareceu no carrossel. Uma aplicação poderia nunca ser inserida no carrossel. O tamanho máximo do Carrossel levou em conta o tempo total disponível para transmissão. Além disso, nos testes realizados nesse trabalho, verificou- se que o melhor número de aplicações máxima do carrossel era em torno de duas ou três vezes o número total de aplicações.

5. Modelo de Negócio Proposto

A interatividade na TV Digital trouxe consigo um novo modelo de negócio para as emissoras. Estas, que antes comercializavam espaço para propaganda, diferenciando os preços de acordo com horário e tempo de exibição, agora vão poder explorar a banda extra de seus canais

para comercializar aplicativos de terceiros.

Empresas interessadas em divulgar suas aplicações deverão primeiro encontrar uma emissora para fazer a transmissão. Porém será possível que mais de uma aplicação esteja no ar ao mesmo tempo, e a forma de comercialização deve levar em conta o tamanho da aplicação, o horário, o tempo que permanecerá disponível, e o número de usuários que vão utilizá-la.

O modelo proposto é focado em propaganda e tem como inspiração o Google Adwords, um modelo de propaganda na internet que cobra de acordo com o número de clicks nos links patrocinados. Esse modelo é satisfatório pra ambas as partes envolvidas, já que o contratante só paga pelos clicks que recebeu, e a Google só recebe se expor as propagandas adequadamente.

Um preço inicial por aplicação deveria ser cobrado em função do tamanho da aplicação, do horário de transmissão e da classe da aplicação. Cada programa do canal teria um preço inicial associado. O contratante poderia escolher em quais programas deseja que sua aplicação seja exibida, já que determinados horários podem não ser interessantes para o mesmo.

As classes de aplicação são designadas de acordo com o valor pago por elas. Cada classe define um valor fixo a ser pago por KB (unidade de tamanho em bytes) da aplicação. As classes são definidas de 1 a 10, onde na classe 1, o contratante paga 1 real por KB de aplicação. Na classe 2, o contratante pagaria 2 reais por KB de aplicação, e assim sucessivamente. Dessa forma, o contratante que optasse pela classe mais alta, teria sua aplicação com maior prioridade inicialmente.

Dado um custo inicial por aplicação, o preço pago passaria a ser em função do número de usuários que vão utilizar aquela aplicação. Dessa forma, as aplicações que tiverem grande repercussão e utilização gerarão lucro agradável para os contratantes, que deverão pagar um preço maior para a emissora.

Esse modelo força as emissoras a darem prioridade às aplicações mais utilizadas pelos usuários e às aplicações de maior classe, já que essas darão maior lucro, diminuindo o tempo de espera da maioria dos usuários. No final, as três partes envolvidas saem beneficiadas.

Esse modelo também abre margem para controle de uso da aplicação por parte das empresas contratantes, que poderão estipular um limite superior de uso, para não pagarem tão caro. Dessa forma, elas podem controlar o quanto que desejam que suas aplicações sejam utilizadas.

6. Formulação Matemática

O presente modelo matemático é uma extensão da formulação matemática apresentada em (Pessoa, 2008), abrangendo as novas restrições descritas no modelo de negócio detalhado na seção 5. Segue abaixo:

Notação:

X: $Conjunto\ de\ Aplicações$
N: $Número\ de\ Aplicações$
t_i: $Tamanho\ da\ aplicação\ i\ em\ milissegundos$
T_{max}: $Tamanho\ máximo\ do\ Carrossel\ em\ milissegundos$
q: $Número\ de\ posições\ ocupáveis\ no\ Carrossel$
M: $Um\ número\ maior\ que\ T_{max}$
c_i: $Classe\ da\ aplicação\ i$
v_i: $Número\ de\ vezes\ que\ a\ aplicação\ i\ foi\ iniciada$
k_i: $Constante\ multiplicativa\ da\ classe$
k_2: $Constante\ multiplicativa\ do\ número\ de\ vezes\ que\ a\ aplicação\ foi\ iniciada$
z_i: $c_1k_1 + v_ik_2$

Variáveis de Decisão:

$$x_{ij} = \begin{cases} 1 & se\ a\ aplicação\ i\ ocorrer\ na\ posição\ j \\ 0 & caso\ contrário \end{cases}$$

d_{ij}^{k}: $Soma\ dos\ tempos\ das\ aplicações\ entre\ as\ posições\ i\ e\ j, com\ relação\ a\ aplicação\ k$
D_i: $Maior\ tempo\ (em\ milissegundos)\ entre\ duas\ ocorrências\ consecutivas\ da\ aplicação\ i$
$maxD$: $Maior\ D_i$

Função objetivo e restrições:

$$Minimizar\ maxD \tag{7.0}$$

Sugeito a:

$$\text{maxD} \geq z_i \times D_i \qquad \forall_i \in X \tag{7.1}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{q} t_i x_{ij} \leq T_{max} \tag{7.2}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall_j \in [1 \dots q] \tag{7.3}$$

$$x_{ij} = x_{i(j+q)} \qquad \forall_i \in X \qquad \forall_j \in [1 \dots q] \tag{7.4}$$

$$d_{kj}^{i} \geq \sum_{s=1}^{n} \sum_{l=k}^{j-1} t_s x_{sl} - M(1 - x_{ik}) - M(1 - x_{ij}) - \sum_{l=k+1}^{j-1} M x_{il} \qquad \forall_i \in X \qquad \forall_{k,j} \in [1 \dots 2q], k \leq j \tag{7.5}$$

$$D_i \geq d_{kj}^{i} \qquad \forall_i \in X \qquad \forall_{k,j} \in [1 \dots 2q], k < j \tag{7.6}$$

$$\sum_{j=1}^{q} x_{ij} \geq 1 \qquad \forall_i \in X \tag{7.7}$$

$$x_{ij} \in \{0,1\} \qquad \forall_i \in X \qquad \forall_j \in [1 \dots 2q] \tag{7.8}$$

$$D_i, d_{kj}^{i} \geq 0 \qquad \forall_i \in X \qquad \forall_{k,j} \in [1 \dots 2q], \tag{7.9}$$

Como pode ser observado, o modelo busca otimizar o pior tempo de carregamento de uma aplicação multiplicada por sua prioridade. A prioridade é calculada por dois fatores: A classe da aplicação e o número de vezes que foi utilizada. Com isso, a tendência é que o fator ZxD de cada aplicação seja próximo, e assim o Carrossel seja montado da forma mais justa.

A representação do Carrossel está sendo feita na forma de uma matriz $n \times 2q$, onde as linhas são as aplicações e as colunas são as posições ocupáveis do carrossel. Dessa forma, se x_{ij} for verdadeiro, significa que a aplicação i está na posição j do Carrossel. Foram utilizadas duas vezes o número de colunas para que não fosse necessário se girar o carrossel por completo para calcular as distâncias máximas de cada aplicação.

A restrição 7.1 está diretamente ligada a função objetivo, enfatizando que o valor de *max* deve ser o maior que todos os $(z_i \times D_i)$, para qualquer *i*. A restrição 7.2 está ligada ao limite de tempo que o carrossel pode ter em seu total, isto é, a soma dos tempos de todas as aplicações do carrossel não pode ultrapassar $TMAX$. A restrição 7.3 garante que em cada posição, apenas e necessariamente uma aplicação estará lá. A restrição 7.4 foi feita para que a matriz com $2q$ colunas represente na verdade o carrossel dando dois giros a partir da primeira aplicação. Para isso, foram replicadas as posições a partir da metade até a última.

A restrição 7.5 trata justamente de todas as possíveis maiores distâncias entre cada aplicação consigo mesma. O valor de d_{kj}^{i} é alterado para a distância entre as posições *k* e *j*, se a aplicação *i* estiver nelas, e entre essas posições não houver nenhuma ocorrência da aplicação *i*.

Depois, com a restrição 7.6, é calculada a maior distância de cada aplicação para sua possível replicação no carrossel. Caso a aplicação só apareça uma vez no carrossel, a distância será a soma dos tempos das aplicações em cada posição ocupável do carrossel.

A restrição 7.7 garante que toda aplicação estará presente no Carrossel. Já as restrições 7.8 e 7.9 determinam os possíveis valores para cada x, D e d, e também asseguram que as maiores distâncias de cada aplicação devem ser maiores que 0.

7. GRASP + ILS

Para resolver o problema abordado neste artigo, foram utilizadas as metaheurísticas GRASP, para construção de uma boa solução, e ILS, para o refinamento da solução encontrada. Os resultados podem ser vistos na seção 8 deste artigo.

7.1. GRASP

O GRASP (*Greedy Randomized Adaptive Search Procedure*) é uma metaheurística que utiliza uma heurística gulosa e aleatória para tentar alcançar um resultado satisfatório. Ela constrói algumas soluções iniciais, e através de uma busca local, procura encontrar uma solução quase-ótima (Rezende, 1998).

Um pseudo-código do GRASP está reproduzido abaixo:

Algoritmo GRASP(f(.),g(.),N(.),GRASPMax,s)

```
1.    f*←∞;
2.    para 1,2,...,GRASPMax faça
3.          Construção(g(.), α, s);
4.          BuscaLocal(f (.), N (.), s);
5.                se f(s) < f* então
6.                      s* ← s;
7.                      f* ← f(s);
8.                fim se
9.    fim para
10.   devolva s*;
```

fim

Para se construir uma solução de forma aleatória e gulosa, é utilizada uma lista, chamada de LCR (Lista restrita de candidatos). A construção se dá por iterações, onde em cada uma delas, a lista é recomposta. A construção da lista é feita se utilizando um fator α, que é o grau de aleatoriedade. Quanto maior o alfa, mais elementos terão a lista, e mais aleatória será a solução. Selecionam-se os α melhores elementos que otimizem de forma gulosa a solução atual (ou seja, para aquela nova iteração, os elementos que deixem a melhor solução possível para o próximo passo). Daí se escolhe aleatoriamente um dos elementos da lista para se compor a solução.

Como pode ser observado, o GRASP é capaz de retornar a melhor solução encontrada após o *GRASPMax* execuções do algoritmo. Para cada iteração, são construídas novas soluções e refinadas com a utilização da busca local. Para o algoritmo proposto, além da busca local padrão, foi utilizada a metaheurística ILS para refinamento da solução.

7.2. ILS

O algoritmo ILS (*Iterated Local Search*) se baseia no fato de que as soluções ótimas globais nem sempre são encontradas por uma busca local. Isso porque podem existir ótimos locais que confundem o algoritmo de busca, levando-o a concluir que boas soluções foram encontradas, quando na verdade vizinhanças próximas podem possuir soluções bem melhores (Lourenço, 2001).

A aplicação do ILS é feita a partir de uma perturbação da solução encontrada, a fim de encontrar uma vizinhança distante, que possua um melhor ótimo local. A aplicação de várias perturbações, percorrendo o maior numero de vizinhanças possíveis, nos dá uma maior probabilidade de encontrar o ótimo global. Um pseudo-código que efetua o algoritmo ILS pode

ser visto no algoritmo 2.

Algoritmo ILS

1. *s0* ←GeraSoluçãoInicial;
2. *s* ←BuscaLocal(*s0*);
3. enquanto os critérios de parada não estiverem satisfeito faça
4. *s'* ←Perturbação(*histórico,s*);
5. *s''* ←BuscaLocal(*s'*);
6. *s* ←CritérioAceitação(*s*, *s''*, *histórico*);
7. fim enquanto
8. devolva *s*;

fim

7.3. GRASP e ILS aplicados ao problema

Para se aplicar o GRASP ao problema, inicialmente um carrossel é construído, contendo todas as aplicações em uma ordem aleatória. Em cada passo do GRASP, o algoritmo insere uma nova aplicação no Carrossel, sendo esta retirada aleatoriamente de uma lista restrita de candidatos (LRC). A lista restrita é composta pelas aplicações que tendem a melhorar mais a função objetivo. Nesse caso, após vários testes com diferentes valores, ficou definido que a lista seria composta pelos 25% melhores aplicativos do momento.

A função objetivo é minimizar o valor de Max, que é o pior $z_i \times D_i$. O z_i se refere à prioridade da aplicação i, e D_i é o valor do tempo médio de espera da aplicação i. Logo, para se calcular a LRC, em vez de calcular o ganho com a inserção de cada aplicação, foram tomadas as aplicações cujos valores de $z_i \times D_i$ estavam mais prejudicando o carrossel.

Devemos notar que a cada passo a lista de candidatos é alterada para tornar o algoritmo adaptativo e gerando soluções diferentes, mas com uma certa qualidade. A cada iteração do GRASP são inseridas $3n$ aplicações. Esse número foi o que mostrou trazer os melhores resultados, tanto no trabalho corrente como no de Pessoa (2008).

Após a construção, o GRASP ainda é responsável por uma busca na vizinhança de soluções para alcançar o ótimo local. Essa vizinhança foi definida como a troca entre 2 ou 4 posições no carrossel e a remoção/inserção de uma aplicação em uma posição qualquer.

Depois de geradas soluções boas com o GRASP, o ILS é aplicado, buscando aprimorar ainda mais a solução encontrada. Para isso, foram definidos três tipos de vizinhança:

Vizinhança de Remoção: a aplicação de uma perturbação nessa vizinhança se dá através da remoção aleatória de uma das aplicações no carrossel. Ela deve respeitar a restrição de que toda aplicação deve estar presente no carrossel

Vizinhança de Troca: a perturbação se dá através da troca de posições entre duas ou quatro aplicações distintas no carrossel

Vizinhança de Inserção: nesse casso uma nova aplicação é inserida no carrossel. Na perturbação que utiliza essa vizinhança, o algoritmo deve checar se o carrossel não viola a restrição de tamanho máximo do carrossel.

As perturbações não têm uma sequência lógica, apenas são aplicadas aleatóriamente. Isso faz com que a solução seja diversificada, e assim o algoritmo consiga fugir de ótimos locais. Após a aplicação das perturbações, uma busca local é realizada.

Um esquema do algoritmo completo pode ser visto na Figura 5.

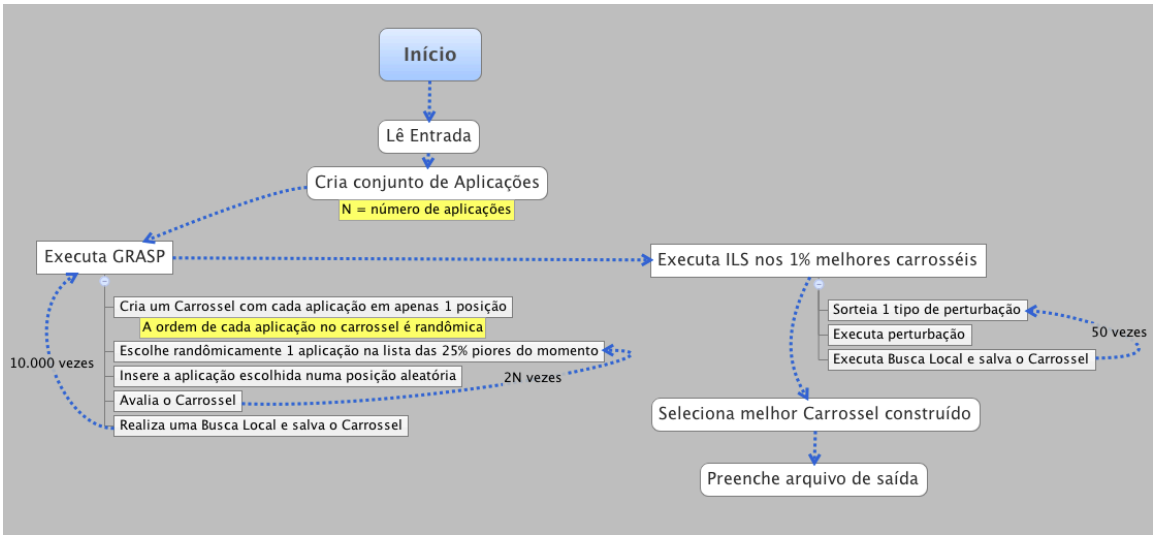

Figura 5 - Execução do algoritmo completo

8. Resultados

Inicialmente, para resolver o problema proposto foi utilizada uma abordagem exata de Programação Linear Inteira. Porém esta se mostrou ineficiente para instâncias maiores, pois, conforme a abordagem proposta, o tempo para calcular uma solução deve ser baixo. Com isso, foram implementadas as metaheurísticas propostas e feita uma comparação com instâncias de tamanhos n = 3, 5, 7, 10, 15, onde n é o número de aplicações no carrossel. As instâncias foram geradas por um algoritmo aleatório, onde o tamanho de cada aplicação respeita um limite mínimo e máximo, definido através de um estudo de todas as aplicações presentes no Laboratório de Vídeo Digital (Lavid) da UFPB. A descrição das instâncias pode ser vista nos anexos. A comparação entre os resultados e tempos pode ser observada na Tabela 1.

Instância	Solução Cplex	Tempo(s)	Solução GRASP +ILS	Tempo(s)
Instância de tamanho 3	53571	0.34	53571	0.033
Instância de tamanho 5	78870	9.89	78870	0.793
Instância de tamanho 7	101776	30.313	101776	1.524
Instância de tamanho 10	234010*	1352.78	227230	3.611
Instância de tamanho 15	371295*	2485	355940	8.22

***Melhor solução encontrada em até 7200 segundos**

Tabela 1 - Comparação com a solução exata

Por ser um trabalho pouco explorado na literatura, só foi encontrado uma abordagem para resolver o problema de otimização das aplicações dentro de um carrossel. Porém, no trabalho de Pessoa (2008), o modelo de negócio foi definido com base no lucro da emissora (maximizar o lucro). Com isso, seria inviável comparar seus resultados com os resultados obtidos com o trabalho corrente, utilizando um outro modelo.

Diante disso, se fez necessário implementar os algoritmos propostos para o modelo de negócio já explorado, presente no trabalho de Pessoa (2008), para então ser feita uma comparação entre os algoritmos. Sendo assim, a Tabela 2 mostra a comparação entre os tempos e resultados obtidos com os algoritmos propostos nesse trabalho e os tempos e resultados obtidos com o trabalho semelhante. As instâncias utilizadas foram utilizadas foram definidas em (Pessoa, 2008). O objetivo é maximizar o lucro da emissora.

Os resultados obtidos foram bastante satisfatórios, pois além de manter a qualidade da

solução, com uma melhora não significativa de 0,16%, alcançou os resultados com um tempo médio 72% menor. Quando analisadas instâncias com 15 aplicações (instâncias 10 até 15), o ganho com o tempo é ainda mais significativo, melhorando em 80%.

# Instância	Resultado(R$) [Pessoa, 2008]	Tempo (s)	Resultado Corrente (R$)	Tempo (s)
1	513,041.36	0.50	519,004.00	1.23
2	856,288.85	0.40	864,912.00	1.45
3	665,352.83	0.60	668,630.00	1.65
4	695,735.43	0.80	695,760.50	1.43
5	596,337.41	0.80	597,085.75	1.65
6	862,590.53	4.90	864,402.00	1.98
7	917,090.16	4.20	917,151.00	2.02
8	936,838.61	5.10	936,838.61	1.89
9	798,168.60	6.20	802,603.00	2.1
10	929,091.26	2.20	929,091.25	2.16
11	1,086,103.89	23.40	1,086,103.89	5.65
12	1,102,782.58	31.00	1,102,782.58	5.23
13	1,118,117.48	20.50	1,118,246.00	6.62
14	1,132,070.52	28.80	1,132,070.52	4.75
15	1,026,302.19	27.70	1,023,192.76	4.56
Total	13,235,911.70	157.1	13,257,873.86	44.37

Tabela 2 - Comparação com [Pessoa, 2008]

9. Considerações Finais

A possibilidade de se transmitir aplicativos interativos por parte das emissoras permitiu a criação de um novo modelo de negócio. O trabalho corrente buscou criar um modelo justo para todas as partes envolvidas: a emissora, a empresa contratante e o usuário.

Com o modelo definido, surge então o Problema da Variabilidade no Tempo de Download aplicado ao ambiente de TV Digital. Para resolvê-lo, o trabalho corrente utilizou as metaheurística GRASP e ILS, que se mostraram bastante eficientes, trazendo resultados em curto espaço de tempo, mesmo para instâncias consideravelmente grandes. Ao comparar o algoritmo proposto com outro da literatura, percebeu-se a eficiência dos resultados alcançados, pois alcançou resultados de igual qualidade em uma média de tempo 72% menor.

Como trabalhos futuros, intenciona-se incorporar uma estratégia de busca local exata, explorando a formulação matemática.

Referências

ABNT NBR 15601, (2007). Revisada em 07.04.2008. Televisão digital terrestre - Sistema de transmissão

ABNT NBR 15604, (2007). Revisada em 07.04.2008. Televisão digital terrestre – Receptores

Balabanian, V. , Casey, L., (1996). An Introduction to Digital Storage Media Command and Control (DSM- CC). Disponível em: <http://mpeg.chiariglione.org/technologies/mpeg-2/mp02-dsm/dsmcc/dsmcc.htm>

Corominas, A., Garcia-Villoria, A., Pastor, R., (2009). Using Tabu Search for the Response Time Variability Problem - XIII Congreso de Ingeniería de Organización.

Festa, P., Resende, M. G. C., (2002). "GRASP: an annotated bibliography", Essays and Surveys on Metaheuristics (C.C. Ribeiro e P. Hansen, editores), pp. 325-367, Kluwer Academic Publishers.

Garey, M. R., Johnson, D. S., (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Co., San Francisco.

Kubiak, W., (2004). Fair sequences. In Handbook of Scheduling: Algorithms, Models and Performance Analysis , Leung, J.Y-T., editor, Chapman & Hall/CRC, Boca Raton, Florida.

Lourenço, H. R., Martin, O. C., Stützle, T., (2001) Iterated Local Search – Disponível no Handbook on MetaHeuristics e em: < http://arxiv.org/pdf/math/0102188.pdf >
Pessoa, B., Souza Filho, G. L., Cabral, L., (2008). Metaheurísticas Aplicadas à Geração de Carrossel no Sistema Brasileiro de Tv Digital - 14th Brazilian Symposium on Multimedia and the Web
Rezende, M. (1998) "Greedy Randomized Adaptative Search Procedures (GRASP)". Technical Report, ATT Labs Research
Yamada, F., (2004) Sistema de TV Digital. In Revista Mackenzie de Engenharia e Computação – Ano 5, Número 5.

Anexos - Instâncias Utilizadas na Comparação Exata

Instância 1

Índice	Tamanho (KB)	Classe	Acessos
1	1232	3	190
2	7653	5	230
3	2321	6	110

Instância 2

Índice	Tamanho (KB)	Classe	Acessos
1	753	4	520
2	5032	3	110
3	403	5	55
4	3201	9	130
5	2102	3	330

Instância 3

Índice	Tamanho (KB)	Classe	Acessos
1	720	10	320
2	3940	5	130
3	1302	7	511
4	330	3	302
5	203	7	600
6	4002	6	200
7	2032	8	92

Instância 4

Índice	Tamanho (KB)	Classe	Acessos
1	2032	2	212
2	1023	7	450
3	4212	8	206
4	2032	5	103
5	452	10	131
6	4233	5	220
7	5302	10	54
8	902	3	201
9	5032	6	402
10	3402	2	220

Instância 5

Índice	Tamanho (KB)	Classe	Acessos
1	4801	5	300
2	890	8	190
3	5045	4	630
4	2303	2	350
5	1110	1	220
6	1103	10	50
7	3203	8	110
8	405	10	50
9	7021	8	110
10	3045	9	90
11	556	2	520
12	3402	2	140
13	2312	3	210
14	455	7	430
15	4532	6	30