METAHEURÍSTICAS GRASP E ILS APLICADAS AO PROBLEMA DA VARIABILIDADE DO TEMPO DE DOWNLOAD EM AMBIENTES 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 de Souza Filho
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 neste trabalho 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 os resultados com um trabalho da 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 in this work 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, uma grande variedade 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.

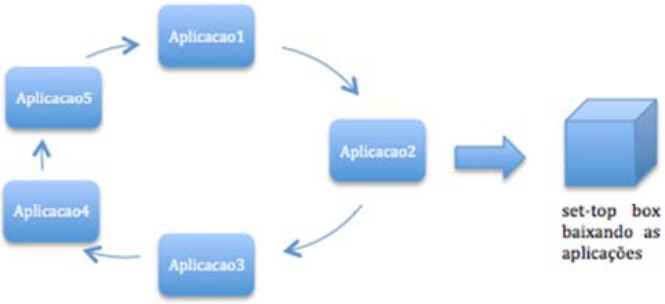

Figura 1 - Carrossel em TV Digital

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 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).

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 os 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 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.

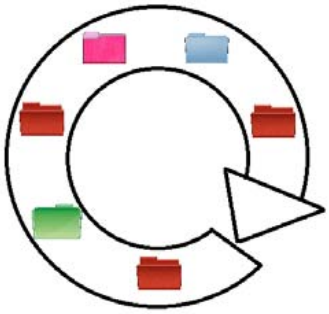

Figura 2 - Carrossel com Prioridade

Definir quais aplicações devem ter prioridade no carrossel é uma tarefa complicada, tendo em vista que deve-se respeitar o contrato das emissoras e, ao mesmo tempo, garantir um tempo de espera menor para as aplicações mais utilizadas pelos usuários. 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 às 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 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 $q! \times n!$.

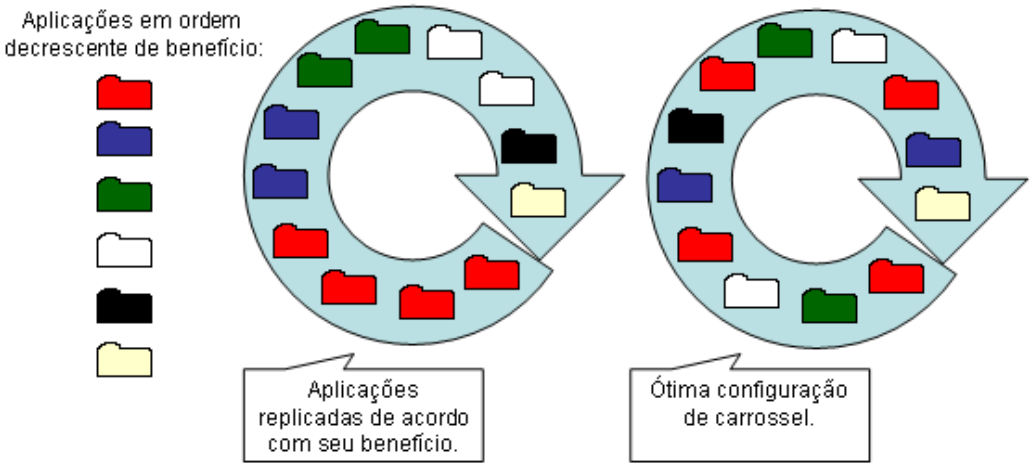

Figura 3 - Carrossel ótimo

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 levando 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 tempo máximo fixado previamente.

O trabalho de Pessoa (2008) utiliza as metaheurísticas GRASP (*Greedy Randomized Adaptive Search Procedure*) + VND (*Variable Neighborhood Descent*). 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 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 modelo é satisfatório para 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 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 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 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.

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:

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

Sujeito 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 à função objetivo, enfatizando que o valor de $maxD$ 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 T_{max}. 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 (Resende, 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 seguinte:

Algoritmo ILS
1. *s0* ←GeraSoluçãoInicial;
2. *s* ←BuscaLocal(*s0*);
3. enquanto os critérios de parada não estiverem satisfeito faça
4. *s'* ←Perturbação(*s*);
5. *s''* ←BuscaLocal(*s'*);
6. *s* ←CritérioAceitação(*s*, *s''*);
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 distribuídas 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.

A função objetivo é minimizar o valor de *maxD*, 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, onde $\alpha = 25\%$, ou seja, a lista era composta por apenas 25% de todas as aplicações.

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 $2n$ 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 um 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 escapar de ótimos locais de baixa qualidade. Após a aplicação das perturbações, uma busca local é realizada.

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

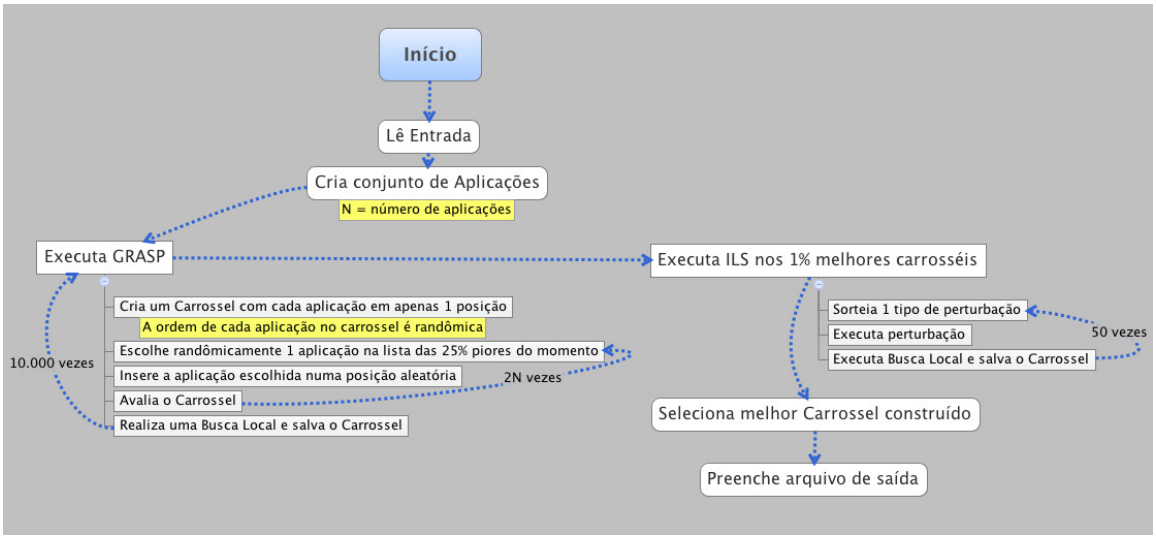

Figura 5 - Execução do algoritmo completo

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 na linguagem Java 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.

Tanto o CPlex quanto o algoritmo em Java foram executados em uma máquina Core 2 Duo 2.14 GHz, com 4GB de RAM, rodando o sistema operacional Windows. 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 trabalho futuro, 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
Resende, 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