Universidade Federal da Paraíba

Centro de Informática

Programa de Pós-Graduação em Informática

META-HEURÍSTICAS GRASP E ILS APLICADAS AO PROBLEMA DA VARIABILIDADE DO TEMPO DE RESPOSTA

Wesley Willame Dias Menezes

Orientação: Prof. Dr. Lucídio dos Anjos F. Cabral

João Pessoa, PB

Julho 2014

Apresentação

Este documento descreve a dissertação de mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal da Paraíba, PPGI UFPB, como parte dos requisitos necessários à obtenção do título de Mestre em Informática (Sistemas de Computação).

Agradecimentos

Diante de todas as conquistas e desafios alcançados por meio deste trabalho, tenho muito a agradecer...

A Deus, por propiciar o mestrado em momento tão oportuno, por dar forças e tornar aqueles momentos possíveis quando pareciam impossíveis.

Ao meu pai (*in memoriam*), que embora sequer tenha me visto entrar para o mestrado, mas que certamente teria me dado todo o apoio e conselhos para me sair bem sucedido.

A minha mãe, por toda compreensão e suporte nas atividades que precisei desempenhar.

A minha noiva, pelo companheirismo durante estes poucos anos, por ter tido certa compreensão pela minha omissão em determinados momentos em virtude de outros que naquele momento eram mais importantes.

Ao meu orientador, Professor Doutor Lucídio dos Anjos Formiga Cabral, pelo tempo despendido durante as reuniões que tivemos, pela receptividade que teve em sua casa, pelo compromisso e dedicação para comigo desde o começo, só tenho a elogiá-lo. Entendo perfeitamente as dificuldades que tivemos em identificarmos horários compatíveis, mas isso é o preço que se paga por ter um orientador competente, atarefado e cheio de responsabilidades, sejam elas familiares e/ou profissionais.

Ao gerente da unidade Mário Matos, pela credibilidade que deu a mim e permitiu que continuasse no mestrado quase que ao mesmo tempo em que eu fora chamado para o concurso da Dataprev, certamente foi uma escolha difícil, uma vez que foi nosso primeiro contato. Vou lembrar muito disso.

Ao meu antigo gerente de projeto Antônio Jaime, que recepcionou muito bem no projeto, mesmo tendo conhecimento de algumas limitações de horário em alguns dias da semana.

Ao meu atual gerente de projeto Fabrício Paiva, que não teve mais tanto problema com horário, mas mesmo assim era nítido seu apoio para que eu concluísse, e que assim como eu, vê bons frutos nesta conclusão do mestrado.

Aos amigos do trabalho, por entenderem de certa forma quando eu não podia sair para almoçar com eles, pois precisava sair cedo do trabalho, fosse para estudar para alguma prova, ou reunião com meu orientador ou mesmo para descansar e ter um dia mais produtivo de estudo no dia seguinte.

Aos meus colegas do mestrado, pelos encontros para revisarmos juntos quando uma prova estava por vir, pelos incentivos recíprocos em virtude das dificuldades que passamos juntos.

Meus sinceros agradecimentos a todos vocês.

Resumo

Com o advento dos avanços tecnológicos, cada vez mais se procura soluções que utilizem menos recursos, sejam mais rápidos e de baixo custo. Em virtude disso, este trabalho propôs uma abordagem meta-heurística híbrida utilizando *Greedy Randomized Adaptive Search Procedure* (GRASP) e *Iterated Local Search* (ILS) aplicados ao Problema da Variabilidade do Tempo de Resposta. Este problema pode envolver desde alocação de recursos escassos, como por exemplo, máquinas industriais ou salas de reunião, passando pelo agendamento de clientes de um banco que requerem certas condições, o planejamento das propagandas de TV ou o percurso feito por caminhões de empresas transportadoras, dentre outros. Para a aplicação do procedimento, foram utilizados os movimentos de deslocamento do mesmo, permuta de posição entre símbolos e de um movimento chamado *double brigde*, que é uma mistura dos movimentos de deslocamento e permutação envolvendo símbolos opostos. As estruturas de vizinhança compostas basearam-se nos movimentos descritos anteriormente, variando a quantidade de símbolos envolvidos. Desta forma, os resultados obtidos demonstram que tais procedimentos trouxeram resultados satisfatórios ao problema e condizentes quando comparados com a literatura.

Palavras-chave: Variabilidade do Tempo de Resposta, Greedy Randomized Adaptive Search Procedure, Iterated Local Search

Abstract

With the advent of technological advances, increasingly demand solutions that use fewer resources, are faster and low cost. As a result, this paper proposed a hybrid approach using metaheuristics Greedy Randomized Adaptive Search Procedure (GRASP) and Iterated Local Search (ILS) applied to the Response Time Variability Problem (RTVP). Since this problem may involve allocation of scarce resources, such as industrial machinery or meeting rooms, going for scheduling of banking customers that require certain conditions, planning of TV ads or route taken by vehicles of logistic companies, etc. For application of the procedure, the movements of shifting symbols, swapping positions between symbols and one called double bridge, which is a mix of movements of shifting and swapping involving opposite symbols were used. The neighborhood structures were based on the movements described above, varying the number of symbols involved. Thus, the results obtained demonstrate that such procedures satisfied the problem and brought consistent results when compared with the literature.

Keywords: Response Time Variability, Greedy Randomized Adaptive Search Procedure, Iterated Local Search

Sumário

Lista de Acrônimos e Siglas

DSM-CC	Digital Storage Media Command and Control
DVB	Digital Video Broadcasting
GRASP	Greedy Randomized Adaptive Search Procedure
HAVi	Home Audio Video Interoperability
ILS	Iterated Local Search
LAVID	Laboratório de Aplicações de Vídio Digital
LCR	Lista de Candidatos Restrita
MPEG	Moving Picture Experts Group
MILP	Mixed Integer Linear Programming
QAP	Quadratic Assignment Problem
QIP	Quadratic Integer Programming
RTVP	Response Time Variability Problem
SBTVD	Sistema Brasileiro de TV Digital
VTR	Variabilidade do Tempo de Resposta
VND	Variable Neighborhood Descent
VNS	Variable Neighboorhood Search

Lista de Figuras

Lista de Tabelas

Capítulo 1

1 Introdução

É cada vez mais comum o confrontamento de pessoas com problemas cada vez mais complexos. Esses problemas podem envolver desde alocação de recursos escassos, como por exemplo, máquinas industriais ou salas de reunião, passando pelo agendamento de clientes de um banco que requerem certas condições, o planejamento das propagandas de TV ou o percurso feito por caminhões de empresas transportadoras, dentre outros. Antigamente se trabalhava com margens mais flexíveis de aceitação e, portanto não se fazia necessário tanto esforço para atingir tal meta.

Com o advento dos avanços tecnológicos essas margens de aceitação são cada vez menores, fazendo com que cada vez mais se procure soluções que utilizem menos recursos, sejam mais rápidos e de baixo custo. Com isso dito, aumenta a necessidade de se utilizar técnicas de otimização para que se alcance tais objetivos.

As meta-heurísticas são definidas como sendo um conjunto de procedimentos implementados de tal forma que sejam capazes de conduzirem uma busca sistemática possibilitando escapar de ótimos locais. Com isso, elas são bastante utilizadas na busca por soluções aproximadas tão próximas quanto possível da solução ótima do problema, evitando procedimentos que exijam grandes esforços computacionais.

As meta-heurísticas tem se mostrado presente ao promover soluções de boa qualidade de forma aproximativa a problemas de otimização discreta, comumente conhecidos como problemas NP-difícil. Estes por sua vez são definidos como um conjunto de problemas os quais não existe um algoritmo que os resolvam de forma exata em tempo polinomial (CORMEM et al., 2002).

Para problemas de otimização discreta, a utilização de métodos exatos torna o algoritmo ineficaz, uma vez que encontrar a solução exata do problema requer um grande esforço computacional. Por esta razão, as meta-heurísticas entram em cena, pois muitas vezes é suficiente identificar soluções próximas do ótimo global a baixo custo computacional.

De acordo com Corominas et al., (2007), o Problema da Variabilidade do Tempo de Resposta ocorre em diversas situações na vida real, muitos dos sistemas modernos

compartilham recursos entre diferentes tarefas. E essas tarefas se definem por certa quantidade de trabalho a ser feita. Este problema, caracterizado como um subproblema de sequência justa ocorre sempre que eventos, tarefas, clientes ou produtos precisam ser sequenciados de forma a minimizar a variabilidade de tempo esperado para obter um determinado recurso necessário. Nesse caso, as tarefas precisam ser uniformemente distribuídas de forma que a distância entre duas tarefas consecutivas do mesmo tipo seja o mais regular possível, ou seja, idealmente constantes.

Diante do exposto, o presente estudo teve por objetivo propor uma abordagem de meta-heurística híbrida utilizando GRASP e ILS aplicados ao Problema da Variabilidade do Tempo de Resposta, tendo por base uma variante desse problema, o Problema da Variabilidade no Tempo de Download em Ambientes de TV Digital proposto por Ramos (2012).

Para tanto, a formulação do Problema da Variabilidade do Tempo de Resposta foi descrita da seguinte forma: Dado n números inteiros positivos e d_i o número de cópias do símbolo i, onde $d_1 \leq ... \leq d_n$, sendo D o somatório de d_i. Considerando $s = s_1 s_2 ... s_D$ uma sequência de tamanho D onde o símbolo i ocorra exatamente d_i vezes. Essa sequência será chamada viável. Para $d_i \geq 2$, quaisquer duas ocorrências consecutivas de i se define a distância t como sendo o número de posições que os separam somado a um. Esta distância também pode ser entendida como distância fim a fim entre duas ocorrências. Aqui busca-se minimizar o máximo valor que a distância t possa assumir, considerando todas as cópias para cada um dos símbolos.

1.1 Objetivos

1.1.1 Objetivo geral

Esse estudo teve por objetivo geral o desenvolvimento de um novo método para resolução do problema da variabilidade no tempo de resposta, utilizando-se de abordagens heurísticas para tratar esse problema que é NP-Difícil (COROMINAS et al., 2007).

1.1.2 Objetivos específicos

Como objetivos específicos, podemos citar:

- Implementar uma meta-heurística híbrida GRASP+ILS para o Problema da Variabilidade do Tempo de Resposta.
- Efetuar um levantamento bibliográfico sobre este problema e suas variantes.
- Comparar os resultados obtidos com os trabalhos relacionados, no sentido de demonstrar a eficácia do algoritmo desenvolvido.

1.2 Organização do trabalho

O trabalho divide-se em capítulos da seguinte maneira:

- Capítulo 1: Introduz o problema, uma motivação sobre o assunto, as meta-heurísticas e suas aplicações, como também algumas vantagens de utilizá-las no problema em questão. Descreve também os objetivos gerais e específicos do trabalho.
- Capítulo 2: Apresenta uma fundamentação teórica a respeito do problema e detalha as meta-heurísticas envolvidas. É discutida a visão geral do problema como também as meta-heurísticas utilizadas, no caso *Greedy Randomized Adaptive Search Procedure* (GRASP) e *Iterated Local Search* (ILS).
- Capítulo 3: Apresenta uma revisão dos trabalhos encontrados até o presente momento que se relacionam com o problema.
- Capítulo 4: Aborda a proposta do trabalho, detalhando como o algoritmo a ser implementado.
- Capítulo 5: Descreve a metodologia utilizada, como também o ambiente a ser utilizado no período de desenvolvimento.
- Capítulo 6: Informa sobre as considerações finais do trabalho, assim como sugestão para trabalhos futuros.

Capítulo 2

2 Fundamentação teórica

Este capítulo inicia com uma breve explanação de heurísticas no item 2.1. Na sequência, o item 2.2 trata da definição de meta-heurísticas detalhando principalmente as meta-heurísticas utilizadas no trabalho proposto. Finalizando o capítulo no item 2.3 com a descrição de Sequências Justas, que é a base para o Problema da Variabilidade do Tempo de Resposta.

2.1 Heurísticas

Heurística é um procedimento eficiente intuitivamente, com finalidade de solucionar problemas que tenha custo computacional aceitável [Aguilera et al, 2008]. Sua proposta se dá por reduzir a complexidade deste sem que garanta a otimalidade do resultado, ou seja, utiliza-se de operações simples de julgamento para buscar valores próximos ao ótimo (TONETTO et al., 2006).

Podemos citar como exemplo alguns procedimentos heurísticos aplicados à otimização combinatória, tais como Variable Neighborhood Search (MLADENOVIC et al., 1997) e Tabu Search (GLOVER et., 2006).

Método de pesquisa local, também conhecido como busca local, é um procedimento frequentemente utilizado por ter como objetivo a busca de uma solução próxima à solução atual em um espaço comum de soluções, levando em consideração propriedades de uma solução ótima local.

Embora heurísticas tenham gerado grande poder computacional ao longo dos anos, mas tais métodos são relativamente frágeis quanto à execução dos seus procedimentos, pois aplicam o conceito de tentativa e erro, tendendo a não escaparem de soluções ótimas locais (Chaves, 2009).

Neste contexto, surgem as meta-heurísticas, que embora compartilhem dos mesmos objetivos que procedimentos heurísticos, mas são algoritmos flexíveis que não fazem uso excessivo dos recursos computacionais.

2.2 Meta-heurísticas

Meta-heurísticas tiveram seu surgimento em meados da década de 80 em um estudo que combinava métodos heurísticos básicos, cujo objetivo também era encontrar soluções em problemas de otimização combinatória [Glover, 1986].

As meta-heurísticas podem ser diferenciadas pelas seguintes características:

- Decisões gulosas, que por vezes geram boas soluções iniciais;
- Utilização de buscas locais por meio de estratégias determinísticas;
- Uso dinânico de memória auxiliar para capturar dados intermediários;
- Inspiração em procedimentos naturais, como a simulação do comportamento de formigas ou mesmo mutação genética.

Em seguida, serão apresentadas as duas meta-heurísticas que serão utilizadas no trabalho: Greedy Randomized Adaptive Search Procedure (GRASP) e Iterated Local Search (ILS).

2.1.1 Greedy Randomized Adaptive Search Procedure (GRASP)

O GRASP (Procedimento de Busca Adaptativa, Aleatória e Gulosa) é um procedimento que utiliza técnicas meta-heurísticas e que divide-se em duas fases principais, foi desenvolvido por Feo et al., (1995) e tem por objetivo retornar a melhor solução dentre todas as encontradas. Seu nome deriva das suas características gulosa, aleatória e adaptativa.

O GRASP inicia na fase de construção, de forma que é gerado um conjunto de soluções viáveis utilizando técnicas gulosas e aleatórias a cada iteração. Gulosa no sentido de nem sempre obter a melhor solução em virtude dos critérios utilizados. Em seguida, inicia-se a fase de busca local onde é feito um refinamento das soluções encontradas a fim de melhorá-las ainda mais, seguindo algum mecanismo de busca local. Uma vez finalizadas ambas as fases, é retornado a melhor das soluções encontradas.

Ainda em se falando da fase de construção, os elementos são colocados um por vez em uma lista de possíveis soluções, e que dela parte as melhores soluções, mais conhecida como

Lista de Candidatos Restrita (LCR), ela denomina desta forma, pois é composta dos β melhores elementos candidatos da lista, que também pode ser encontrado descrito na forma de $\alpha \in [0,1]$.

Segue um descritivo do funcionamento dessa técnica na figura 1:

```
procedimento GRASP(f, g, N, max, s)
1       f* ← ∞
2       para 1 até Max 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      retorne s*
fim procedimento
```

Figura 1 – Pseudocódigo do GRASP

O parâmetro α restringe o tamanho da lista de candidatos de forma que este valor igual à zero gera soluções puramente gulosas, enquanto que se igual a um gera soluções totalmente aleatórias.

O procedimento descrito na figura 2 mostra a fase de construção do algoritmo.

```
procedimento construção(g, α, s)
1       f* ← ∅
2       inicialize o conjunto C
3       enquanto C ≠ 0 faça
4               g(t_min) ← min { g(t) | t ∈ C }
5               g(t_max) ← max { g(t) | t ∈ C }
6               LCR ← { t ∈ C | g(t) ≤ g(t_min) + α(g(t_max) - g(t_min)) }
7               Selecione aleatoriamente um elemento t ∈ LCR
8               s ← s ∪ {t}
9               Atualize conjunto C
10      fim enquanto
11      retorne s
fim procedimento
```

Figura 2 – Pseudocódigo da fase de construção

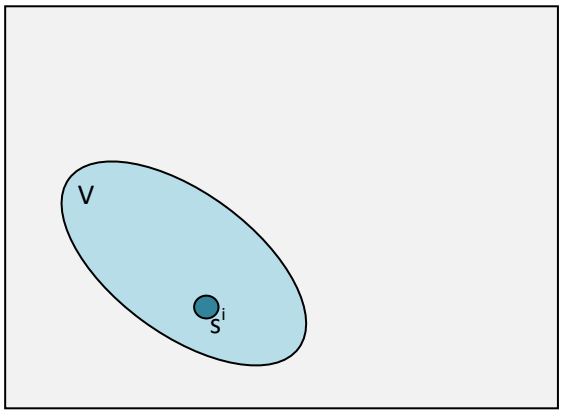
Considerando uma dada vizinhança, as soluções encontradas na fase de construção do GRASP não garante serem ótimos locais, quero dizer, não são necessariamente as melhores soluções da vizinhança. Por esta razão, é feito um segundo procedimento que visa obter melhores soluções quando comparado com as inicialmente encontradas.

Conforme descrito na figura 3, a busca local se dá pela definição de movimentos que são feitos sucessivamente a fim de substituir a solução atual por outra solução visitada que seja melhor.

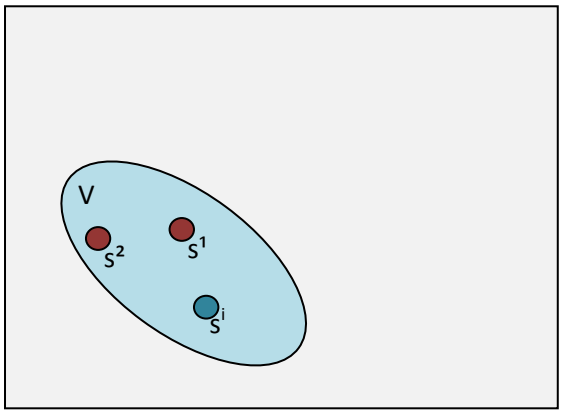
```
procedimento buscaLocal(f, N, s)
1       V ← {s' ∈ N(s) | f(s') < f(s) }
2       enquanto |V| > 0 faça
3               Selecione s'de V
4               s ← s'
5               V ← { s' ∈ N(s) | f(s') < f(s) }
6       fim enquanto
7       retorne s
fim procedimento
```

Figura 3 – Pseudocódigo da fase de refinamento

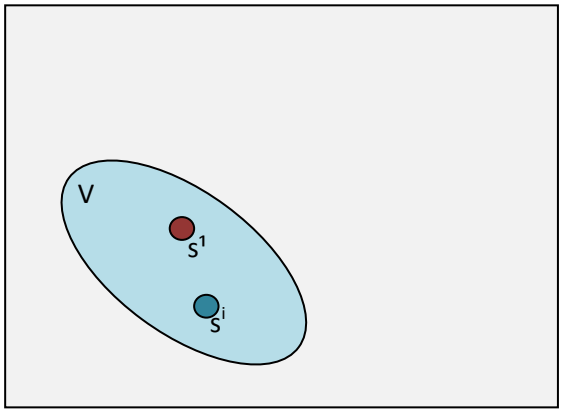
A qualidade das soluções na busca local tem relação direta com a qualidade das soluções encontradas na construção, ou seja, boas soluções iniciais contribuem numa melhor eficiência na busca local.



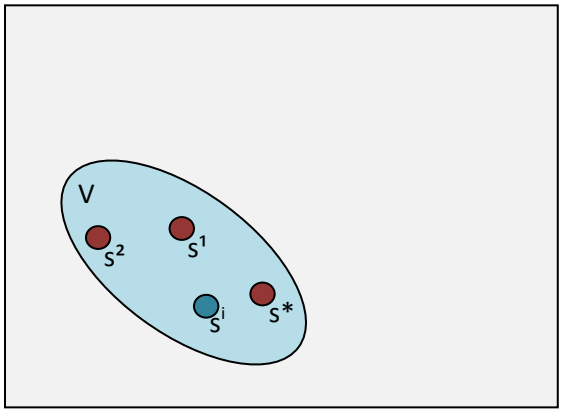
Passo 1 – Identificação da vizinhança e da solução inicial



Passo 3 – Solução s^2 encontrada na vizinhança V de s^1



Passo 2 – Solução s^1 encontrada na vizinhança V de s^i



Passo 4 – Solução sub-ótima s* encontrada na vizinhança V

Figura 4 – Ilustração dos passos do algoritmo GRASP

Considerando a vizinhança V na figura 4, é gerada a solução inicial s^i e em seguida iniciado o processo de iteração de forma que identifica uma solução s^1 melhor que s^i, segundo o critério de refinamento. Depois, o processo é reiniciado e encontrado a solução s^2 da mesma vizinhança, assim sucessivamente. E finalmente, após *max* execuções é encontrado s*,que é a solução sub-ótima encontrada para o problema, segundo os critérios de construção e refinamento instanciados.

2.1.2 Iterated Local Search (ILS)

ILS (Busca Local Iterada) trata-se de um procedimento que também utiliza técnicas meta-heurísticas de busca local. Este, baseia-se em fazer uma busca local por melhores

soluções a partir de soluções ótimas locais por meio de perturbações (LOURENCO et al, 2003).

Essas perturbações são comumente utilizadas de tal forma que permita sair de ótimos locais, possibilitando explorar novas áreas, mas sem perder as características do ótimo local atual.

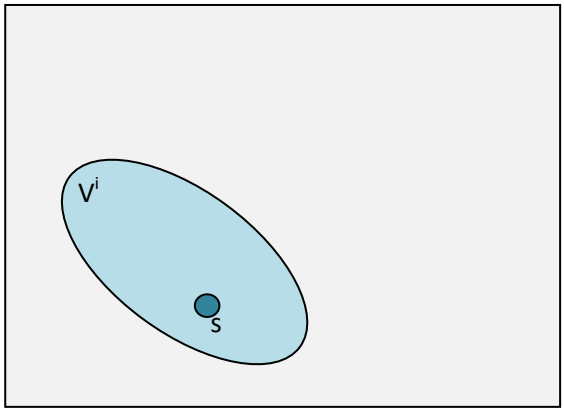
São necessários quatro procedimentos para possibilitar tal implementação:

- Solução inicial: gera a solução inicial
- Busca local: faz a busca local (intensificação)
- Perturbação: modifica a solução atual (diversificação)
- Aceitação: escolhe a próxima solução a ser perturbada

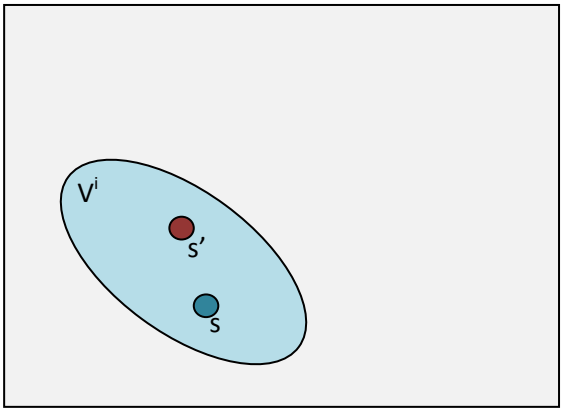
O procedimento de perturbação executa movimentos para diversificar soluções, enquanto que o procedimento de aceitação verifica se a solução encontrada está na condição de ser aceita.

```
procedimento ILS
1       s0 ← geracaoSolucaoInicial()
2       s ← buscaLocal(f, N, s0)
3       enquanto critério de parada nãosatisfeito faça
4               s' ← perturbação(histórico, s)
5               s'' ← buscaLocal(s')
6               s ← criterioAceitacao(s, s'', histórico)
7       fim enquanto
fim procedimento
```

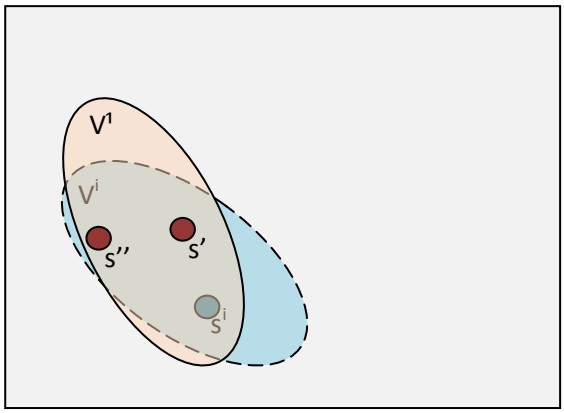
Figura 5 – Pseudocódigo do ILS.



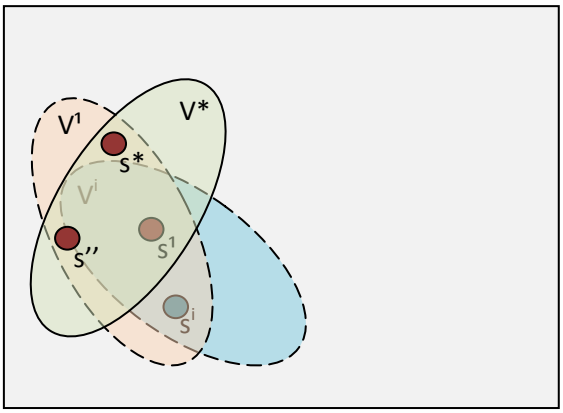
Passo 1 – Identificação da vizinhança e solução inicial a partir da busca local



Passo 2 – Solução s' encontrada na vizinhança V^i a partir da perturbação



Passo 3 – Solução s^2 encontrada na vizinhança V^1 a partir da busca local



Passo 4 – Solução sub-ótima s* encontrada na vizinhança V*

Figura 6 – Ilustração dos passos do algoritmo ILS

Considerando a vizinhança V^i da figura 6, é gerada a solução inicial s^i e em seguida iniciado o processo de iteração de forma que identifica uma solução s^1 melhor que s^i, segundo o critério de refinamento. Depois, diferentemente do que ocorre com GRASP, é feita a perturbação, o processo é reiniciado e encontrado a solução s^2 na vizinhança V^1. E finalmente, uma vez chegada à condição de parada é retornado s*, que é a solução ótima encontrada para o problema, segundo os critérios adotados.

2.3 Sequência Justa

A ideia das sequências justas parece ter começado numa linha de montagem de modelos mistos na Toyota. Foi solicitado que suavizasse as taxas de uso das peças e

produção de todos os modelos para então reduzir a escassez de um lado e estoque excessivo do outro (MONDEN, 1983).

Dito isto, foi percebido na Toyota que diferentes modelos espalhados uniformemente por toda sequência de produção atendia melhor a tais requisitos do que uma sequência em batch que mantém todas as cópias do mesmo modelo mais próximas possíveis.

Miltenburg (1991), formulou tal problema como sendo um problema de programação inteira não linear, cujo objetivo era minimizar o desvio do modelo de produção atual da quantidade desejada do modelo de produção determinado por demandas $d_1, ..., d_N$ dos modelos *1, ..., n*, respectivamente (KUBIAK, 2004).

A partir daí, referenciada como sendo Problema da Variação da Taxa de Produção.

O problema de sequência justa pode ser formalmente definido da seguinte forma: Dado *n* modelos, sendo *n* inteiros positivos $d_1,...,d_i,...,d_n$ (demandas) e *n* funções simétricas e convexas $f_1,..., f_i,..., f_n$ de uma variável chamada desvio, que assumem valor mínimo igual a 0 no ponto 0. Encontrar $S = s_1,...,s_n$, $D = \sum_{i=1}^{n} d_i$ onde o modelo *i* ocorre exatamente d_i vezes que minimize

$$F(S) = \sum_{i=1}^{n}\sum_{k=1}^{D} f_i(x(S)_{ik} - r_i k)$$

onde $x(S)_{ik}$ é igual ao número de cópias do modelo *i* no prefixo $s_1,...,s_k$, podemos referenciá-lo como prefixo-k, k=1,...,D, e $r_i=d_i/D$, i=1,...,n.

O exemplo na figura 7 mostra a situação ideal do símbolo A com três cópias, onde as distâncias entre eles são todas iguais.

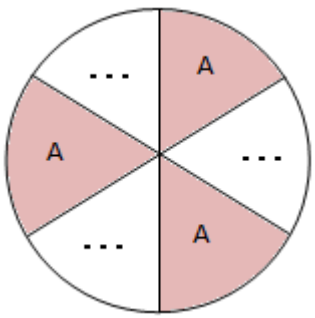


Figura 7 – Sequência justa para D = {3, 0, 0}

De forma semelhante, a figura 8 exemplifica a situação ideal dos símbolos A, B e C, cada um com duas cópias cada, as distâncias em si também são iguais.

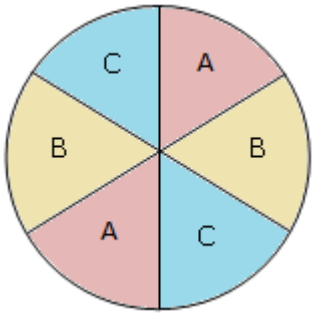


Figura 8 – Sequência justa para D = {2, 2, 2}

Porém, quando as cópias são definidas aleatoriamente ou baseando-se em determinados critérios que atendam certas situações, o problema se assemelha ao exemplo da figura 9 e este por sua vez torna-se um problema da classe NP-Difícil (KUBIAK, 2004), principalmente quando se remete aos conjuntos com o número de elementos mais elevados.

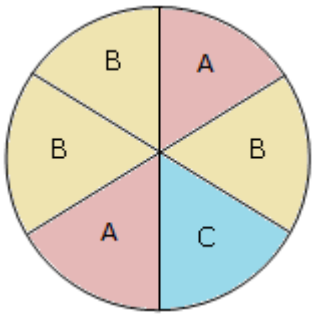


Figura 9 – Sequência justa para D = {2, 3, 1}

Capítulo 3

3 Trabalhos Relacionados

Os primeiros trabalhos a respeito do problema de variabilidade do tempo de resposta se desenvolveram por volta de 2007, quando de lá para cá não houve muitas publicações além das publicações do próprio autor que trouxe o problema em discussão.

Em suas publicações, Corominas et al. (2007), procura mostrar o mesmo problema aplicando diferentes abordagens na busca pela melhor estratégia em resolver o problema.

Já quando se fala de algumas poucas publicações que o referenciam, procura-se contextualizá-lo a um problema mais específico de características próprias.

Abaixo, citaremos alguns dos trabalhos que foram desenvolvidos e que foram tomados como base de conhecimento para o desenvolvimento da proposta em questão.

3.1 Variabilidade do Tempo de Resposta

O Problema da Variabilidade do Tempo de Resposta ocorre em diversas situações na vida real, muitos dos sistemas modernos compartilham recursos entre diferentes tarefas. E essas tarefas se definem por certa quantidade de trabalho a ser feita, por exemplo, o tamanho de um arquivo a ser transmitido ou o número de carros de um modelo em particular que precisa ser produzido em uma linha de montagem mista (COROMINAS et al, 2007).

Este problema, caracterizado como um subproblema de sequência justa, ocorre sempre que eventos, tarefas, clientes ou produtos precisam ser sequenciados de forma a minimizar a variabilidade de tempo esperado para obter um determinado recurso necessário.

Para garantir um sequenciamento justo de recursos comuns entre diferentes tarefas, é necessário dividi-las em tarefas atômicas, como por exemplo, bloco de dados e carros.

Essas tarefas precisam ser uniformemente distribuídas de forma que a distância entre duas tarefas consecutivas do mesmo tipo seja o mais regular possível, ou seja, idealmente constantes.

Em redes ATM (Asynchronous Transfer Mode), cada aplicação (voz, arquivo de dados, vídeo) é dividida em células de tamanho fixo fazendo com que a aplicação possa ser preemptada após cada uma. Além disso, aplicações isócronas, de voz e vídeo, precisam que a distância entre células em um fluxo seja a mais constante possível e que não exceda um valor predefinido no pior caso.

Em uma linha de montagem mista, uma sequencia de diferentes modelos a serem produzidos onde cada modelo é distribuído o mais uniformemente possível, mas atendendo uma quantidade mínima de cada modelo para atender uma demanda. Consequentemente, é reduzida a escassez de alguns modelos e o excesso de outros.

O problema da variabilidade no tempo de resposta frequentemente é vista como um problema de planejamento de distâncias forçada, onde a distância temporal quaisquer duas execuções de uma mesma tarefa não seja maior que uma distância predefinida.

Algumas vezes condições mais restritas são impostas como, por exemplo, a distância temporal seja igual a uma distância predefinida ou com intervalos constantes.

No entanto, o modelo de distância forçada sofre um problema sério que pode não haver uma solução viável que atenda a condição e ao mesmo tempo certifique-se que as tarefas sejam feitas num certo ritmo.

A formulação do Problema da Variabilidade do Tempo de Resposta pode ser escrito da seguinte forma:

i	representação de símbolo qualquer
d_i	número de cópias do símbolo i
D	total de símbolos
$\bar{t}_i$	distância média entre símbolos i

Dado n números inteiros positivos, onde $d_1 \leq ... \leq d_n$, definimos:

$$D = \sum_{i=1}^{n} d_i$$

Considere $s = s_1 s_2 ... s_D$ uma sequência de tamanho D onde i (cliente, produto, tarefa, símbolo, aplicação) ocorra exatamente d_i vezes. Essa sequência será chamada viável.

Para $d_i \geq 2$, quaisquer duas ocorrências consecutivas de i se define a distância t como sendo o número de posições que os separam somado a um. Esta distância também pode ser entendida como distância fim a fim entre duas ocorrências.

Considerando que i ocorrem exatamente d_i vezes em s, então existem exatamente d_i distâncias $t_1^i, ..., t_{d_i}^i$ para i, onde $t_{d_i}^i$ é a distância entre a última e a primeira ocorrência de i.

Logo,

$$t_{d_1}^i + ... + t_{d_i}^i = D$$

e portanto a distância média $\bar{t}_i$ entre i's em s é igual

$$\bar{t}_i = \frac{D}{d_i}$$

que é o mesmo para cada sequência viável s. Então a variabilidade do tempo de resposta para i é definido como

$$RTV_i = \sum_{1 \leq j \leq d_i} (t_j^i - \bar{t}_i)^2$$

e a variabilidade do tempo de resposta como

$$RTV = \sum_{i=1}^{n} RTV_i = \sum_{i=1}^{n} \sum_{1 \leq j \leq d_i} (t_j^i - \bar{t}_i)^2$$

Observe que a variabilidade do tempo de resposta é uma variação ponderada cujos pesos são iguais às demandas, ou seja

$$RTV = \sum_{i=1}^{n} d_i Var_i$$

onde, $Var_i = \frac{1}{d_i} \sum_{1 \le j \le d_i} (t_j^i - \bar{t}_i)^2$

3.1.1 Otimização e Complexidade

Neste contexto, Corominas (2007) procura exemplificar um caso para minimizar a variabilidade no tempo de resposta e a complexidade de tal solução.

3.1.1.1 Caso dois produtos

Em seguida, o s autores, demonstram uma solução que tanto minimiza a variabilidade no tempo de resposta como também maximiza o desvio. Dentre várias alternativas de combinações entre produtos, o autor demonstra através de alguns lemas e provas que a solução ótima só pode ser encontrada quando utilizado números sendo um par e outro ímpar, ou ambos ímpares.

3.1.1.2 Número fixo e complexidade

E para complementar os resultados a respeito da complexidade, o estudo mostra também que o problema pode ser resolvido em tempo polinomial para qualquer número fixo de produtos n e que a complexidade de tal solução é $O(D^{3n+1})$.

3.1.2 Algoritmo Exato

De acordo com Corominas et al. (2007), o PVTR deriva do Problema da Alocação Quadrática (*Quadratic Assignment Problem* - QAP), que por sua vez é formalizado como Programação de Inteiros Quadráticos (*Quadratic Integer Programming* - QIP), e que ele pode ser resolvido utilizando ILOG SOLVER apenas para instâncias muito pequenas, pois em experimentos foi identificado que logo se excedia o tempo aceitável de 180 segundos. O uso do CPLEX com modelo MILP se mostrou mais eficiente e que o limite prático deste modelo é $D = 25$.

3.1.3 Heurística

3.1.3.1 Procedimento de troca

Funciona de tal forma que a troca é feita sempre quando dois elementos próximos resultam em diminuir VTR. Caso a posição 1 seja alcançada sem nenhuma troca, o procedimento é parado. Caso contrário, o próximo da sequência é iniciado.

3.1.3.2 Sequências iniciais

Gargalo

Esta sequência é obtida como resultado do problema do gargalo. Foi usado o algoritmo proposto por Moreno (2002) para resolver tal problema. Outras abordagens foram propostas na literatura e poderiam ter sido usadas para obter sequências possivelmente diferentes (COROMINAS et al, 2004).

Aleatório

É uma variante da sequência do gargalo, porém trocando a posição de cada cópia da sequência de D elementos por outra qualquer da mesma sequência, podendo inclusive ser o mesmo símbolo.

Inserção

Imagine que $d_1 \leq \ldots \leq d_n$. Considere $n-1$, os problemas com dois elementos $P_{n-1} = (d_{n-1}, d_n)$, $P_{n-2} = (d_{n-2}, \Sigma_{j=n-1}^{n} d_j)$, ..., $P_1 = (d_1, \Sigma_{j=2}^{n} d_j)$. Para cada problema P_{n-1}, P_{n-2}, ..., P_1, o primeiro elemento é o original, ou seja $n-1$, $n-2$, ..., 1, e o segundo o mesmo elemento fictício para todos os problemas, denotado por $*$. Seja as sequências S_{n-1}, S_{n-2}, ..., S_1 solução ótima para os problemas P_{n-1}, P_{n-2}, ..., P_1, respectivamente. Perceba que a sequência S_j, para $j = n-1$, ..., 1 é composta pelos elementos j e $*$. Assim, a sequência para o problema original pode ser obtido recursivamente substituindo $*$ em S_1 por S_2 para obter $S_1^{"}$. Perceba que $S_1^{"}$ é composto pelos elementos 1, 2 e $*$. Em seguida, $*$ é substituído por S_3 em $S_1^{"}$ para obter $S_1^{"}$, composto por 1, 2, 3 e $*$. Finalmente, a sequência S_{n-1} substitui os $*$ remanescentes, obtendo a sequência de inserção, onde o elemento i ocorre exatamente d_i vezes.

3.1.4 Experimentos computacionais

O referido estudo, foi feito com objetivo de avaliar duas métricas, erro de transferência e VTR. O experimento consiste em aplicar a heurística de troca às sequências iniciais descritas e analisar tais métricas. O código foi implementado em C++, executado em PC arquitetura x86, processador 500Mhz e 128MB RAM (COROMINAS et al, 2007).

Foram geradas 6650 instâncias fixando o total de unidades D e o número de elementos n, e selecionando aleatoriamente o número de cópias de cada elemento, uniformemente distribuídos.

D = 100	n = { 3,10k | k = 1,...,9 }
D = 500	n = { 3,5,10,50k | k = 1,...,9 }
D = 1000	n = { 10,50,100k | k = 1,...,9 }
D = 1500	n = { 100k | k = 1,...,14 }

Os resultados computacionais mostraram que, em média, os valores mais baixos alcançados para PVTR foram principalmente utilizando as sequências iniciais gargalo (54% dos casos) e aleatória (41% dos casos).

Segue as tabelas dos resultados que mostram os valores de VTR inicial e final para cada grupo:

Tabela 1 – Resultados para $D = 100$

n	Bottleneck		Random		Insertion	
	Inicial	Final	Inicial	Final	Inicial	Final
3	43	26	1377	71	31	26
10	539	86	8868	143	306	101
20	2763	105	10424	142	498	104
30	3447	139	15598	120	1539	85
40	4908	172	32549	70	3307	71
50	8621	94	32280	51	2609	57
60	12100	35	31413	31	1840	31
70	15111	17	23834	25	687	11
80	20029	10	17576	15	187	4
90	20296	3	10381	3	17	2

Fonte: Corominas et al., (2007).

Tabela 2 – Resultados para $D = 500$

n	Bottleneck		Random		Insertion	
	Inicial	Final	Inicial	Final	Inicial	Final
3	82	73	2959	608	108	92
5	1081	177	27032	1224	285	183
10	3403	409	82248	2441	993	443
50	63498	1948	1000332	2384	29925	2256
100	153664	1404	1492918	1207	106108	1313
150	234034	4086	3372643	871	300787	1353
200	461694	1497	3730674	593	344989	1036
250	719887	496	4069915	345	380971	496
300	1245583	255	3800175	289	244113	236
350	1578706	86	2573962	306	88601	56
400	3079086	38	2788306	137	32116	20
450	1996168	41	1063597	149	1580	9

Tabela 3 – Resultados para $D = 1000$

n	Bottleneck		Random		Insertion	
	Inicial	Final	Inicial	Final	Inicial	Final
10	17904	601	253685	5435	1546	625
50	180503	3208	2706679	7407	49101	4242
100	636545	4946	7130615	6375	253319	5866
200	1020847	8412	13476818	3517	873579	5032
300	1418033	6120	16271934	2266	1376361	2717
400	3574364	3913	35047637	1195	3584645	3395
500	6232671	1162	35083775	862	3290617	1313
600	7681994	496	21386442	962	1321738	375
700	12206931	179	20496095	952	706786	130
800	18083157	334	15952258	697	184422	84
900	15549203	82	8299817	472	14832	18

Fonte: Corominas et al., (2007).

Tabela 4 – Resultados para $D = 1500$

n	Bottleneck		Random		Insertion	
	Inicial	Final	Inicial	Final	Inicial	Final
100	885866	8780	13310978	13475	318918	11125
200	3206865	10985	28313115	10532	1554419	12273
300	3243331	13577	51059006	5475	3611346	9623
400	4996223	23905	73166821	4048	6624499	8586
500	6458663	10564	82218609	3517	8396917	7558
600	11085096	4866	98681397	2208	9990472	4804
700	14593169	2051	116870944	1628	10742017	3627
800	24081386	1309	110641048	1369	8495253	1501
900	36781821	1007	112399627	1086	7107165	1068
1000	49367053	298	100131267	990	4335813	442
1100	67535885	150	90861641	893	2475907	259
1200	78982366	116	71570932	620	966875	70
1300	83546651	82	52228958	632	201881	29
1400	60688751	65	27742011	360	14809	8

Fonte: Corominas et al., (2007).

MILP foi aplicado a 389 instâncias, para D=10, 15, 20 e 25 e diferentes valores de n e a conclusão que chegou foi que 39 das instâncias, o tempo de execução foi abortado em virtude de ter alcançado o limite de 180 segundos e foram excluídas das estatísticas de comparação, e a heurística encontrou ótimo para 249, das 350 remanescentes.

D	n	[1]	[2]	Insertion			Bottleneck			Random			[6]
				[3]	[4]	[5]	[3]	[4]	[5]	[3]	[4]	[5]	
10	3	8	0	7	0,500	4	8	0,000	0	8	0,000	0	8
	4	7	0	5	0,857	4	5	0,857	4	5	0,857	4	7
	5	7	0	3	1,429	4	2	2,286	6	4	0,857	2	7
	6	5	0	4	0,400	2	5	0,000	0	5	0,000	0	5
	7	3	0	3	0,000	0	2	0,667	2	3	0,000	0	3
	8	2	0	2	0,000	0	2	0,000	0	2	0,000	0	2
	9	1	0	1	0,000	0	1	0,000	0	1	0,000	0	1
Total D=10		33	0	25			25			28			33
15	3	12	0	10	1,333	10	11	0,167	2	9	0,833	4	12
	4	16	0	12	1,375	8	14	0,875	8	14	0,375	4	16
	5	14	0	6	3,571	16	8	1,000	4	10	0,571	2	12
	7	13	0	5	3,231	16	1	5,846	16	5	2,769	10	7
	9	9	0	7	0,667	4	3	4,000	14	7	0,444	2	9
	11	4	0	4	0,000	0	3	0,500	2	2	2,000	6	4
	13	2	0	2	0,000	0	2	0,000	0	2	0,000	0	2
Total D=15		70	0	46			42			49			62
20	3	19	0	15	0,947	8	16	0,421	4	12	2,105	16	18
	4	17	0	11	1,765	14	13	0,471	2	11	0,824	4	14
	5	17	3	6	3,429	14	7	2,000	14	6	3,286	12	10
	6	20	3	3	6,471	20	4	4,941	14	10	2,000	8	12
	7	18	3	5	4,400	12	2	6,267	14	4	4,267	10	7
	8	18	0	5	3,333	8	4	7,000	18	5	3,444	10	8
	9	17	0	5	3,765	16	1	1,094	34	4	4,940	24	8
	11	13	0	7	1,846	8	2	4,615	8	2	3,692	8	7
	13	9	0	7	0,444	2	4	1,556	4	3	1,778	4	8
	15	6	0	5	0,333	2	3	1,000	2	4	1,000	4	5
	18	2	0	2	0,000	0	2	0,000	0	2	0,000	0	2
Total D=20		156	9	71			58			63			99
25	3	18	2	13	1,625	12	16	0,000	0	9	2,250	14	16
	5	17	8	3	4,444	14	4	4,000	14	3	4,444	16	7
	7	20	14	2	4,000	14	1	5,333	10	0	5,000	8	2
	9	18	6	2	9,167	30	0	13,500	22	1	6,667	18	3
	11	18	0	0	10,110	38	0	19,560	52	2	7,000	16	3
	13	15	0	3	5,067	18	0	14,530	36	2	4,933	16	5
	16	13	0	6	1,538	6	3	4,615	12	5	2,923	8	9
	19	8	0	6	0,500	2	4	1,250	4	5	0,750	2	7
	22	3	0	3	0,000	0	3	0,000	0	3	0,000	0	3
Total D=25		130	30	38			31			30			55
Total		389	39	180			156			170			249

Tabela 5 – Comparativo dos resultados entre heurísticas

Legenda:

[1] Número de instâncias

[2] Números de instâncias excluídas

[3] Números de instâncias das quais a heurística encontrou solução ótima

[4] Diferença média entre heurística e valores ótimos

[5] Diferença máxima entre heurística e valores ótimos

[6] Números de instâncias das quais alguma heurística encontrou solução ótima

3.1.5 Conclusão

O trabalho de Corominas (2007) procura mostrar os resultados de VTR utilizando sequências iniciais obtidas a partir de técnicas diferentes, das quais ele informa que Bottleneck, Random e Insertion foram as que obtiveram melhores resultados. Tais resultados foram utilizados como base comparativa para o trabalho proposto.

3.2 Geração de Carrossel no Sistema Brasileiro de TV Digital

Já em 2008, ainda muito recente em relação à publicação do trabalho sobre o problema da variabilidade do tempo de resposta, Pessoa (2008), fez um trabalho sobre a geração de carrossel de aplicações utilizando o Sistema Brasileiro de TV Digital, onde ele propôs um modelo de negócio que visasse o lucro da emissora por meio da disponibilização de aplicativos interativos durante a programação de TV. Com esse modelo, era definida estaticamente a prioridade dos aplicativos de forma a gerar um único fluxo do carrossel para toda a transmissão de um programa.

3.2.1 Sistemas de TV Digital

O trabalho começa com uma breve explanação sobre sistemas de TV digital, cuja definição pode ser dita como um conjunto de especificações com intuito de permitir a transmissão e recepção de sinais digitais. A composição do sistema se dá por uma estação de transmissão, que por sua vez é composto por codificador, gerador de carrossel (padrão DSM-CC), multiplexador (padrão MPEG-2 Sistemas) e modulador, o meio pelo qual o sinal é transmitido e o receptor dos sinais (controlado pelo middleware), mais conhecido por set-up box.

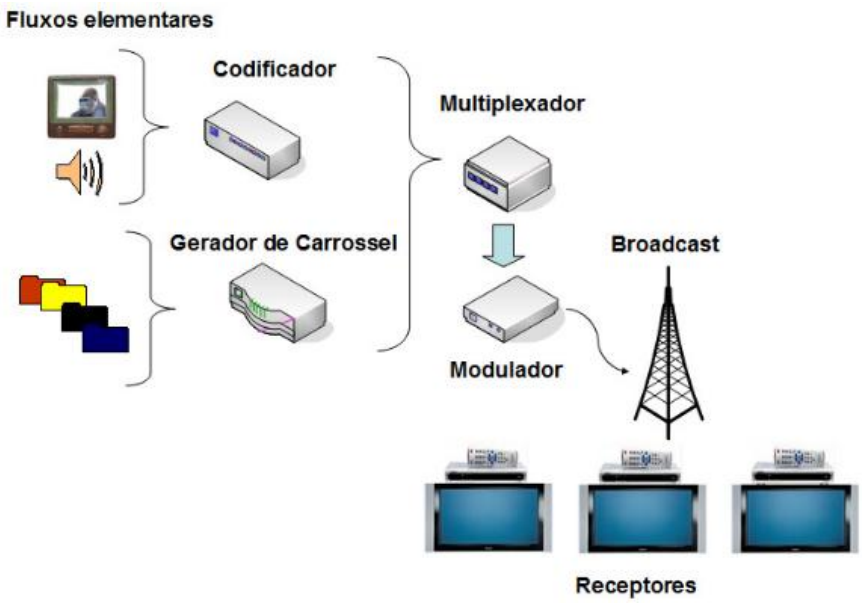


Figura 10 – Elementos básicos de um Sistema de TV Digital

Para permitir a interoperabilidade entre os elementos, foram criados padrões, dentre eles o Europeu (DVB – Digital Vídeo Broadcasting), o Americano (ATSC – Advanced Television Systems Committee), e o Japonês (ISDB – Integrated Services Digital Broadcasting). [PESSOA, 2008]

O middleware, elemento que roda do receptor, também passou por padronização e foram definidos MHP, ACAP, ARIB e Ginga, respectivamente para os sistemas DVB, ATSC, ISDB e SBTVD.

Digital Storage Media - Command and Control (DSM-CC)

DSM-CC é um protocolo para transmissão de dados multiplexados com conteúdo audiovisual, definido na parte 6 do padrão MPEG-2 (PESSOA, 2008).

Este padrão tem suporte à transmissão de dois mecanismos de dados, carrossel de objetos e de dados, este permite transmitir grandes volumes de dados ciclicamente sem que haja uma descrição semântica e formato dos dados, ficando a responsabilidade por parte do receptor e é utilizado pelo padrão Americano. Para casos mais complexos, o carrossel de objetos complementa dados atribuindo-lhes semântica e forma, semelhante a um sistema de arquivos, e é utilizado nos padrões Europeu e Brasileiro.

Aplicações de TV Digital

O referido autor descreve que as duas classificações para aplicações são declarativas e procedurais, e que dependem do middleware ao que vão ser executados.

As declarativas focam na apresentação e sincronização dos objetos de mídia, enquanto que procedurais são mais complexas, mas oferecem mais recursos e maior robustez, pois permite o desenvolvimento de aplicações em diversos fins.

O autor explica também algumas das APIs existentes e algumas das suas principais funcionalidades:

- JavaTV: acesso e seleção de serviços ou informação dos mesmos, controle dos elementos de recepção, acesso aos dados de difusão e gerência do ciclo de vida das aplicações;

- DAVIC: extensão ao controle de exibição de áudio e vídeo provido pelo framework Java Media, manipulação dos elementos de Transport Stream, interfaces às características de um sistema de acesso condicional, acesso ao sintonizador dos receptores, gerenciamento dos recursos, escassos do set-up Box, armazenamento de aplicações, acesso às seções privadas do MPEG-2 Sistemas;
- HAVi: subconjunto do Java AWT 1.1, Alpha blending, Video/Audio Layering, suporte à controle remoto, conjunto próprio de componentes, suporte à diferenças de Pixel Aspect Ratio, Screen Aspect Ratio e Screen Size
- DVB: gerenciamento da conexão do canal de retorno, armazenamento de informações, manipulação de eventos, segurança e gerenciamento de aplicações, acesso estendido aos arquivos transmitidos via DSM-CC, extensões para o controle de exibição de áudios e vídeos, extensão às funcionalidades de acesso condicional do DAVIC.

3.2.2 Apresentação do problema

De acordo com Pessoa (2008), a taxa máxima de transmissão de dados é limitada a 6Mbps. E em virtude dessa limitação e a alta gama de aplicações, que muitas vezes incluam arquivos de áudio, vídeo, imagens ou mesmo banco de dados, fazendo com que o tamanho do carrossel aumente consideravelmente, chegando alcançar trinta vezes o limite máximo da transmissão de dados. Outro ponto levado em consideração é a limitação na capacidade de armazenamento dos set-up boxes, e que devido à política pública de inclusão social do governo Brasileiro, os aparelhos devem ter seus custos reduzidos, consequentemente afetando a quantidade de recursos disponíveis no aparelho. E diante do exposto, o autor infere que as aplicações poderão sofrer atrasos na execução em virtude do tratamento às aplicações que vierem à frente.

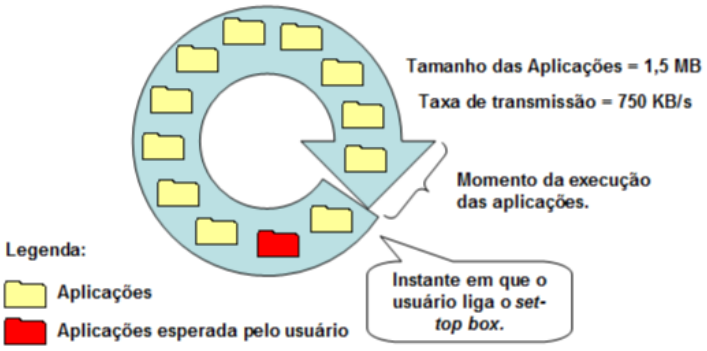


Figura 11 – Representação da transmissão de um carrossel

O exemplo acima mostra como um simples carrossel pode gerar atrasos intoleráveis no contexto televisivo, gerando quebras de contrato, multas exorbitantes e perdas financeiras às emissoras, visto que os telespectadores habitualmente não têm paciência de frente para TV e isso poderia causar mudança de canal, desligamento do terminal ou mesmo aversão a certas aplicações.

O autor então define quarto macro propriedades:

- Todas as aplicações serão inicialmente “candidatas” e poderão estar presentes ou não no carrossel;
- Elas terão frequências diferentes de acordo com seu benefício;
- A distância entre aplicações idênticas dentro do carrossel será otimizada de forma a reduzir seu atraso máximo;
- A receita obtida pela emissora será resultado da subtração do valor recebido com a veiculação das aplicações e a multa paga por atraso de cada uma delas.

E que em virtude das duas primeiras propriedades, que é um tipo especial do problema clássico da Mochila 0-1, então o problema pode ser considerado da classe NP-Difícil (PESSOA, 2008).

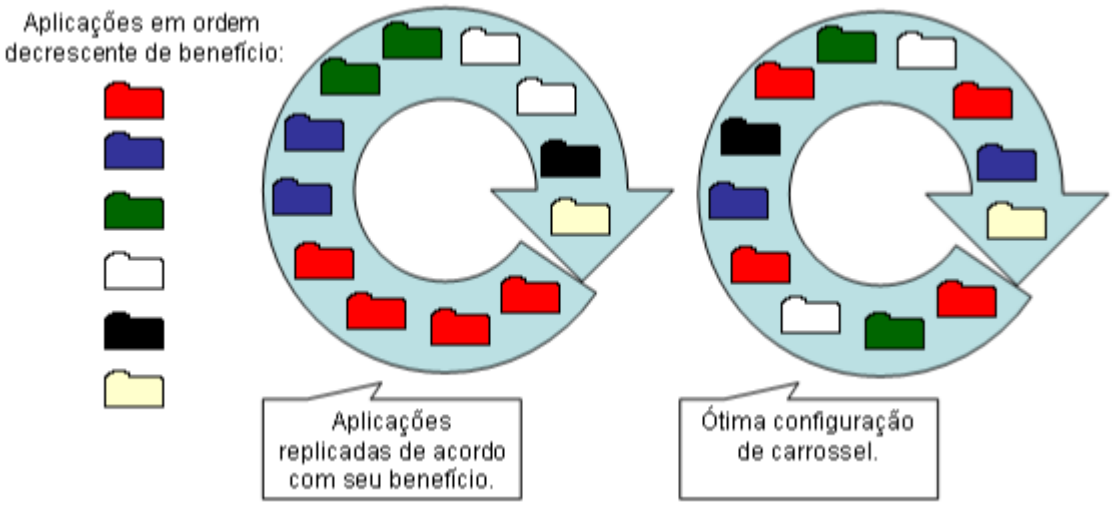


Figura 12 – Otimização do atraso no Carrossel

O número de operações necessárias para encontrar a solução ótima é dado pela fórmula da permutação com repetição, resultando em um algoritmo de complexidade $\theta(n!)$, considerando que n é o somatório das frequências das aplicações selecionadas.

A complexidade do algoritmo demonstra intratabilidade e então o trabalho entra como proposta do uso de heurística no auxílio na busca por uma boa solução em um tempo aceitável.

3.2.3 Modelo de Negócio Proposto

No modelo de negócio atual, os principais parâmetros utilizados para estabelecer preço são tempo de publicidade e horário de exibição. O trabalho propõe que o tamanho da aplicação seja levado em consideração e que o trabalho proposto haja sobre a minimização do atraso na execução das aplicações de TV Digital, aumentando assim a receita das emissoras, propondo regras de negócio objetivando a especificidade de valores e parâmetros para a transmissão dessas aplicações, como também definindo uma forma de calcular a multa pelos atrasos.

O primeiro cenário prevê que um pagamento inicial seria feito em relação ao tamanho da aplicação, e diversas faixas de valores poderiam determinar prioridades das aplicações baseadas em quanto cada anunciante optar em pagar por KB transmitido, o tempo de

manipulação seria a soma do instante da execução mais o atraso e subtraído do período de contrato, e finalmente que a multa seria aumentada gradativamente por segundo de atraso.

Com relação ao segundo cenário, este diferentemente do primeiro cenário, se excluiria o valor pago para transmissão, o que significaria que aplicações grandes arcariam com custos maiores, e as faixas de valores não teriam ligação com o tamanho da aplicação.

3.2.4 Metodologia

O trabalho propõe um modelo matemático que visa reduzir o tempo máximo entre dois downloads subsequentes de um mesmo Xlet. A solução tende distribuir as cópias de maneira uniforme ao longo da sequência do carrossel.

GVND

Foi usado GRASP na fase de construção e leva em consideração o valor pago pela veiculação, o tempo de veiculação, o tamanho da aplicação e o espaço ocupado pela aplicação na iteração. O parâmetro utilizado que demonstrou melhores resultado foi $\alpha = 0{,}5$. A condição de parada definida foi de duas a três vezes o número de aplicações. Já em relação à busca local foi utilizado VND, cujas estruturas de vizinhança utilizadas foram permuta de dois elementos, exclusão de uma aplicação em uma posição específica e inserção de aplicações em uma das posições do carrossel. Os movimentos utilizados foram determinísticos, adotando o Método Primeiro de Melhora (First Improvement Method), ou seja, a solução é atualizada tão logo o primeiro vizinho seja capaz de melhorá-la.

GVNS

Também foi usado GRASP na fase de construção, tal como na situação anterior. Na busca local foi utilizado VNS, cujas estruturas de vizinhança diferenciam do GVND apenas não utilizar a vizinhança por exclusão e também porque a permuta não ocorre de forma exaustiva, mas apenas um número aleatório de vezes entre 1 e 10, variando entre um vizinho e outro caso este valor supere o valor 5. Já os movimentos utilizados foram inserir

uma ou duas das aplicações selecionadas, excluir uma ou duas das aplicações selecionadas e a permuta conforme condições de aleatoriedade explicada anteriormente.

3.2.5 Resultados

O autor relata que não foram encontrados problemas semelhantes na literatura e a base de controle foi obtida a partir da implementação dos geradores de carrossel atuais, avaliando a solução por meio de modelagem matemática. Os resultados foram apresentados para ambos os cenários propostos. A linguagem utilizada para implementação do algoritmo foi C++ e os experimentos foram feitos em Linux 32bits, kernel 2.6.24-16, distribuição Ubuntu 8.04, processador Core 2 Duo T5250 de 1.5GHz.

Foram gerados três grupos de instâncias contendo 10, 15 e 20 aplicações, respectivamente, para cada cenário. Eles foram obtidos aleatoriamente obedecendo a limites de tamanho da aplicação entre 250KB e 2,5MB e atraso máximo tolerado de 12 segundos. Para o cenário A, o tempo total de veiculação foi fixado em 120 segundos, enquanto que para o cenário B o valor foi de 180 segundos. A taxa reservada para transmissão de dados foi estabelecida em 750KB/s, baseada na taxa real do Sistema Brasileiro de TV Digital.

Instância	Lucro (R$)	Instância	Lucro (R$)
1	494.213,63	1	1.366.756,61
2	847.817,20	2	902.172,71
3	665.352,83	3	434.001,04
4	686.040,24	4	1.193.405,92
5	564.443,99	5	1.251.874,58
6	836.188,44	6	1.526.063,89
7	802.245,38	7	1.109.602,25
8	852.626,47	8	2.388.033,33
9	749.796,63	9	2.275.026,54
10	858.005,81	10	1.592.715,83
11	974.174,94	11	2.251.665,61
12	907.228,55	12	1.013.190,90
13	840.132,76	13	1.125.306,10
14	842.525,36	14	1.175.151,46
15	778.131,17	15	1.642.403,28

Tabela 6 – Função de avaliação da solução atual para ambos os cenários

Instância	Construção (R$)	Busca Local (R$)	Lucro (R$)	
			Média	Melhor
1	414.211,60	509.387,44	513.041,36	515.989,53
2	705.635,22	856.038,20	856.288,85	856.288,85
3	516.979,05	665.329,39	665.352,83	665.352,83
4	538.257,44	694.706,79	695.174,89	695.735,43
5	477.678,99	589.872,73	592.469,45	594.338,99
6	682.317,05	858.169,05	858.764,58	858.764,58
7	690.503,37	911.352,72	917.090,16	917.090,16
8	768.281,67	928.191,87	936.838,61	936.838,61
9	650.239,47	794.900,84	795.508,55	795.866,14
10	750.583,29	923.346,99	929.091,26	929.091,26
11	837.842,29	1.078.909,64	1.086.103,89	1.086.103,89
12	829.801,39	1.084.989,10	1.102.782,58	1.102.782,58
13	807.366,08	1.100.071,37	1.115.208,93	1.118.117,48
14	853.078,39	1.114.485,46	1.132.070,52	1.132.070,52
15	745.710,51	1.007.029,11	1.023.192,76	1.023.192,76

Tabela 7 – Detalhamento da contribuição das fases do GVND (Cenário A)

Instância	Construção (R$)	Busca Local (R$)	Lucro (R$)	
			Média	Melhor
1	1.231.601,24	1.442.990,54	1.444.388,39	1.444.422,50
2	773.246,55	941.164,90	942.499,83	947.747,50
3	379.215,50	461.559,83	463.990,51	465.521,94
4	1.000.428,86	1.209.520,98	1.211.545,69	1.212.781,11
5	1.161.043,11	1.406.245,38	1.417.998,33	1.421.742,22
6	1.431.606,04	1.806.732,59	1.823.341,14	1.831.659,44
7	933.280,57	1.224.935,71	1.237.983,23	1.243.751,74
8	2.091.213,20	2.556.859,59	2.570.717,88	2.584.005,28
9	2.087.928,89	2.465.181,42	2.467.266,28	2.467.909,44
10	1.406.732,39	1.805.464,04	1.816.651,68	1.820.392,22
11	2.093.175,91	2.716.915,97	2.734.422,72	2.742.299,44
12	965.567,90	1.268.693,49	1.275.506,02	1.280.092,29
13	1.129.776,72	1.524.103,15	1.541.331,76	1.544.994,17
14	1.106.521,00	1.445.246,15	1.458.245,45	1.459.861,67
15	1.572.051,73	1.961.835,81	1.974.004,50	1.978.230,00

Tabela 8 – Detalhamento da contribuição das fases do GVND (Cenário B)

Instância	Construção (R$)	Busca Local (R$)	VNS (R$)	Lucro (R$)	
				Média	Melhor
1	409.101,36	507.825,73	514.247,26	517.808,66	518.979,43
2	694.941,67	837.205,53	851.716,16	857.749,15	862.260,56
3	514.645,88	657.793,92	663.678,57	665.939,63	667.848,09
4	433.320,19	548.249,44	557.798,52	695.455,16	695.735,43
5	474.773,57	584.527,79	592.313,56	595.503,55	596.337,41
6	854.487,14	974.563,96	1.001.599,28	859.464,95	862.590,53
7	692.994,95	886.869,39	910.084,11	917.090,16	917.090,16
8	731.803,99	881.557,06	911.846,54	936.838,61	936.838,61
9	655.532,88	780.990,32	794.081,07	796.193,86	798.168,60
10	747.110,70	913.481,94	922.988,66	929.004,87	929.091,26
11	841.862,69	1.025.155,87	1.073.932,26	1.086.103,89	1.086.103,89
12	765.597,50	933.593,32	993.823,06	1.102.782,58	1.102.782,58
13	809.947,60	1.024.867,33	1.087.060,42	1.117.046,46	1.118.117,48
14	836.072,65	1.016.947,11	1.086.196,97	1.131.412,24	1.132.070,52
15	745.227,62	941.048,03	996.195,30	1.023.035,67	1.026.302,19

Tabela 9 – Detalhamento da contribuição das fases do GVNS (Cenário A)

Instância	Construção (R$)	Busca Local (R$)	VNS (R$)	Lucro (R$)	
				Média	Melhor
1	1.246.675,22	1.428.601,03	1.443.095,38	1.444.397,83	1.444.422,50
2	749.964,07	935.284,74	943.822,91	948.480,17	949.246,39
3	376.093,93	457.572,38	463.888,94	467.933,10	468.613,89
4	1.003.527,38	1.204.225,27	1.214.758,84	1.220.426,89	1.221.282,78
5	1.133.582,67	1.385.909,23	1.409.049,84	1.421.366,44	1.421.966,67
6	1.429.333,15	1.772.398,68	1.816.741,86	1.826.132,56	1.826.866,67
7	923.492,29	1.215.230,30	1.235.849,12	1.242.505,57	1.243.751,74
8	2.098.575,01	2.535.746,42	2.571.980,65	2.588.682,56	2.594.719,17
9	2.058.489,19	2.435.643,74	2.463.142,46	2.468.637,67	2.471.131,67
10	1.426.168,86	1.777.763,85	1.810.607,69	1.820.337,11	1.820.392,22
11	2.107.504,81	2.644.228,16	2.724.166,34	2.742.282,28	2.747.952,22
12	978.464,27	1.223.674,53	1.272.779,02	1.281.475,44	1.283.045,00
13	1.131.540,24	1.449.999,25	1.538.009,85	1.544.633,14	1.545.533,61
14	1.101.995,09	1.396.052,93	1.451.124,36	1.459.463,56	1.459.861,67
15	1.557.389,37	1.919.161,27	1.966.142,62	1.979.368,19	1.983.706,39

Tabela 10 – Detalhamento da contribuição das fases do GVNS (Cenário B)

O estudo também mostrou tabelas comparativas entre os algoritmos propostos, tanto considerando a média dos valores como também dos melhores valores.

Instância	Solução Atual (R$)	GVND (R$)	GVNS (R$)	Diferença Percentual (R$)		
				Ganho 1	Ganho 2	Ganho 3
1	494.213,63	513.041,36	517.808,66	3,81	4,77	0,96
2	847.817,20	856.288,85	857.749,15	1,00	1,17	0,17
3	665.352,83	665.352,83	665.939,63	0,00	0,09	0,09
4	686.040,24	695.174,89	695.455,16	1,33	1,37	0,04
5	564.443,99	592.469,45	595.503,55	4,97	5,50	0,54
6	836.188,44	858.764,58	859.464,95	2,70	2,78	0,08
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	795.508,55	796.193,86	6,10	6,19	0,09
10	858.005,81	929.091,26	929.004,87	8,28	8,27	-0,01
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	1.115.208,93	1.117.046,46	32,74	32,96	0,22
14	842.525,36	1.132.070,52	1.131.412,24	34,37	34,29	-0,08
15	778.131,17	1.023.192,76	1.023.035,67	31,49	31,47	-0,02

Tabela 11 – Comparação da média dos lucros entre os algoritmos (Cenário A)

Instância	Solução Atual (R$)	GVND (R$)	GVNS (R$)	Diferença Percentual (R$)		
				Ganho 1	Ganho 2	Ganho 3
1	1.366.756,61	1.444.388,39	1.444.397,83	5,68	5,68	0,00
2	902.172,71	942.499,83	948.480,17	4,47	5,13	0,66
3	434.001,04	463.990,51	467.933,10	6,91	7,82	0,91
4	1.193.405,92	1.211.545,69	1.220.426,89	1,52	2,26	0,74
5	1.251.874,58	1.417.998,33	1.421.366,44	13,27	13,54	0,27
6	1.526.063,89	1.823.341,14	1.826.132,56	19,48	19,66	0,18
7	1.109.602,25	1.237.983,23	1.242.505,57	11,57	11,98	0,41
8	2.388.033,33	2.570.717,88	2.588.682,56	7,65	8,40	0,75
9	2.275.026,54	2.467.266,28	2.468.637,67	8,45	8,51	0,06
10	1.592.715,83	1.816.651,68	1.820.337,11	14,06	14,29	0,23
11	2.251.665,61	2.734.422,72	2.742.282,28	21,44	21,79	0,35
12	1.013.190,90	1.275.506,02	1.281.475,44	25,89	26,48	0,59
13	1.125.306,10	1.541.331,76	1.544.633,14	36,97	37,26	0,29
14	1.175.151,46	1.458.245,45	1.459.463,56	24,09	24,19	0,10
15	1.642.403,28	1.974.004,50	1.979.368,19	20,19	20,52	0,33

Tabela 12 – Comparação da média dos lucros entre os algoritmos (Cenário B)

Tempo de Execução

Instância	GVND (s)	GVNS (s)	Instância	GVND (s)	GVNS (s)
1	0,5	21,8	1	1,5	20,7
2	0,4	27,5	2	0,4	33,0
3	0,6	44,6	3	0,6	39,6
4	0,8	29,6	4	0,8	81,5
5	0,8	31,8	5	0,8	33,5
6	4,9	123,3	6	4,9	68,5
7	4,2	100,5	7	4,2	62,9
8	5,1	111,2	8	5,1	205,6
9	6,2	78,5	9	6,2	164,7
10	2,2	214,3	10	2,2	88,0
11	23,4	185,2	11	23,4	223,8
12	31,0	405,8	12	31,0	185,4
13	20,5	172,6	13	20,5	169,5
14	28,8	157,1	14	28,8	200,8
15	27,7	153,7	15	27,7	459,2

Tabela 13 – Tempos de execução dos algoritmos GVND e GVNS em ambos os cenários

3.2.6 Conclusão

O trabalho de Pessoa (2008), que é uma variante do VTR, procura mostrar os resultados de forma que beneficie o lucro das empresas, levando em consideração o tempo de execução do algoritmo proposto. Ele faz um comparativo entre duas meta-heurísticas híbridas, tanto em relação ao lucro quanto aos tempos de execução obtidos por elas. Tais resultados do tempo de execução foram utilizados como base comparativa para o trabalho proposto.

3.3 Meta-heurísticas GRASP e ILS aplicadas ao Problema da Variabilidade do Tempo de Download em Ambientes de TV Digital

O autor propôs um modelo de negócio que visasse tanto o lucro da emissora por meio da disponibilização de aplicativos interativos durante a programação de TV, como também a satisfação da maioria dos usuários por meio da maior disponibilização das aplicações com maiores números acessos.

3.3.1 Apresentação do Problema

O trabalho é uma extensão do trabalho mencionado anteriormente por Pessoa (2008), numa perspectiva em ajustar o carrossel, de forma que também leve em consideração a utilização das aplicações por parte dos usuários.

3.3.2 Método de Negócio Proposto

Diferentemente do modelo proposto por Pessoa (2008), o autor propõe que além de todos os critérios já utilizados para ter uma função objetivo focada no lucro da emissora, que é interessante também levar em conta o número de usuários que vão utilizar a aplicação.

Este modelo é focado em propaganda e foi inspirado no Google Adwords, modelo de propaganda na Internet que cobra de acordo com o número de clicks em links patrocinados (RAMOS, 2012).

Então a proposta é que seja aplicado um preço inicial em função do tamanho da aplicação, horário de transmissão e da classe da aplicação. Estas classes seriam valores fixos a serem pagos por KB a fim de se atribuir prioridades, onde classes mais altas teriam prioridades maiores.

Uma vez definido o preço inicial, o preço variável ficaria em função do número de acessos. A vantagem disso é que força as emissoras a darem maior prioridade às aplicações mais utilizadas pelos usuários e às aplicações de maior classe, pois elas darão mais lucro e diminuirão o tempo de espera da maioria dos usuários.

O autor também explica que com isso é possível estabelecer um teto de gastos por parte da empresa contratante, para não pagar tão caro e obterem um maior controle à utilização da aplicação.

3.3.3 Metodologia

Como mencionado anteriormente, o referido estudo trata-se de uma extensão do trabalho de Pessoa (2008), que também abrange um modelo matemático estendido, considerando as novas restrições descritas no modelo de negócio. O autor utilizou as meta-heurísticas GRASP e ILS para resolver o problema proposto.

GRASP

Após alguns experimentos, ficou definido que $\alpha = 0{,}25$, a função objetivo é minimizar o pior caso de $z_i \times D_i$, que significa a prioridade z da aplicação i multiplicada pelo seu tempo médio de espera D_i. A condição de parada é três vezes o número de aplicações. Ainda foi utilizada estrutura de vizinhança de troca entre duas ou quatro posições no carrossel e remoção/inserção de uma aplicação numa posição qualquer, visando alcançar um ótimo local.

ILS

As estruturas de vizinhança utilizadas foram remoção aleatória de uma aplicação, desde que satisfaça as restrições, a permuta entre duas ou quatro aplicações distintas e a inserção de uma nova aplicação, também sem ferir as restrições.

3.3.4 Resultados

Foram utilizados três grupos de instâncias, os quais o primeiro possui cinco instâncias de tamanhos $n = 3,5,7,10,15$, onde n é o número de aplicações presentes no carrossel, utilizado em comparação com o algoritmo exato, o segundo foi utilizado o mesmo que em Pessoa (2008), e o último é uma cópia do segundo, mas levando em consideração o número de acessos por aplicação, que foram gerados aleatoriamente não excedendo 700. À constante multiplicativa da classe lhe foi atribuído o valor 1, enquanto que a constante multiplicativa do número de acessos foi 0,01. O modelo exato usou CPLEX for MAC para resolver as instâncias quando possível e o algoritmo aproximativo em Java, ambos num MacBook Pro, processador Core 2 Duo 2.4GHz e 4GB RAM DDR3.

Em seguida o referido autor, fez vários comparativos entre os resultados da sua função objetivo, que é minimizar o intervalo de tempo entre duas ocorrências da mesma aplicação multiplicado pela prioridade, dos quais seguem alguns na tabela 14:

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

Tabela 14 – Resolução usando o modelo com CPLEX

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

Tabela 15 – Contribuição do algoritmo ILS sobre o valor da função objetivo

No trabalho proposto por Ramos (2012) também foi mencionado que a comparação com o algoritmo de Pessoa (2008), obteve resultados pouco significativos, mas um ganho relevante em relação ao tempo médio, de 72% melhor.

# instância	Resultado (R$) Pessoa (2008)	Tempo (s)	Resultado (R$) Ramos (2012)	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 16 – Comparação com [Pessoa, 2008]

3.2.6 Conclusão

O trabalho de Ramos (2012), que é uma variante do trabalho de Pessoa (2008), introduz um novo modelo de negócio que também os usuários das aplicações, aplica também uma meta-heurística diferente e procura fazer um comparativo dos lucros das empresas e dos tempos de execução entre ambos os trabalhos. Tais resultados no tempo de execução foram utilizados como base comparativa para o trabalho proposto.

Capítulo 4

4 Métodos Propostos

Através das pesquisas feitas, com o intuito de propor uma solução para o problema de minimização da Variabilidade do Tempo de Resposta, foi observado que as meta-heurísticas GRASP e ILS poderiam gerar soluções viáveis para tal problema. Por isso, este capítulo apresenta as estratégias utilizadas para se obter uma sequência inicial a ser utilizada pelo algoritmo, como também apresenta o algoritmo proposto para a resolução do problema, utilizando as meta-heurísticas citadas acima.

4.1 Algoritmo Proposto

Foi implementado um algoritmo que utiliza as meta-heurísticas GRASP e ILS de forma que resolvesse instâncias maiores do problema da variabilidade do tempo de resposta.

4.1.1 Fase de Construção

Inicialmente, se propôs na fase de construção do GRASP que fosse utilizado sobre sequências iniciais obtidas a partir de três estratégias distintas: Inserção, Plotada e Aleatória. Iterativamente seria criada a LCR calculando o valor de VTR sendo $\alpha = 0.1$ (10% do tamanho da lista), variando o número de cópias de cada símbolo, fixando o número de símbolos distintos e respeitando o limite máximo de cópias dos símbolos como condição de parada.

Considere uma sequência inicial obtida por alguma das estratégias informadas anteriormente

A	B	A	C	D	B	A	C	B	D

VTR = 21

Aplica-se, por exemplo, $\alpha = 0.4$ para definir LRC

A	**B**	**A**	**C**	D	B	A	C	B	D

VTR = 21

Considere que a solução inicial está conforme abaixo:

A	B	A			

VTR = 9

É escolhido um elemento da LRC aleatoriamente, por exemplo, C na posição 3 e acrescentado à solução

A	B	A	C		

VTR = 12

Este processo é repetido até que se obtenha uma solução completa, e então é passado para o próximo passo, de busca local.

4.1.2 Busca Local

Então, a busca local foi aplicada de tal forma que respeitasse a condição do número de cópias de cada símbolo e que resultasse em melhoria da solução, ou seja, minimizando o VTR.

Os movimentos propostos são:

- ***Swap*:** alguns elementos são permutados de posição entre si aleatoriamente.

A	B	A	C	B	B

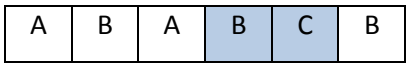
VTR = 22

Dois elementos A na posição 0 e C na posição 3 são escolhidos e permutados de posição entre si.

C	B	A	A	B	B

VTR = 28 ×

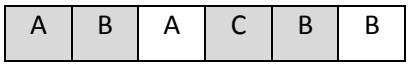
Dois elementos C na posição 3 e B na posição 4 são escolhidos e permutados de posição entre si.



VTR = 20 ✓

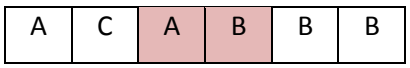
- ***Shift*:** um grupo de elementos consecutivos é movido em conjunto algumas posições anteriores ou posteriores.

  Diferentemente do movimento de permutação, o movimento de *Shift* ocorre com um grupo de 2 ou mais elementos que é movido para frente ou para trás um determinado número de posições.



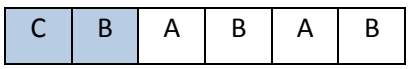
VTR = 22

Os elementos A na posição 0 e B na posição 1 são escolhidos para moverem 2 posições pra frente.



VTR = 26 ×

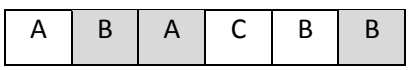
Os elementos C(3) e B(4) são escolhidos para moverem 3 posições pra trás.



VTR = 20 ✓

- ***Double Bridge*:** um grupo de elementos consecutivos é permutado por outro que esteja inversamente posicionado à posição do grupo em questão.

  É um movimento semelhante ao *Shift*, porém são selecionados 2 grupos inversos entre si, quando dividido o vetor em 4.

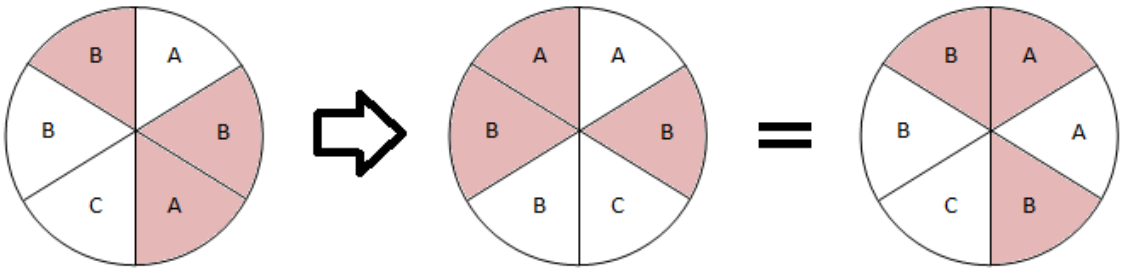


VTR = 22

Os elementos B na posição 1 e A na posição 2 são escolhidos para trocarem de posição com B na posição 5.

A	B	C	B	B	A

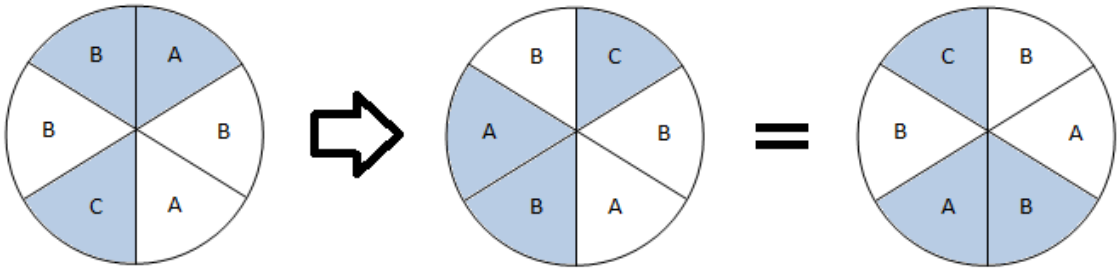
VTR = 28 ×



Os elementos B na posição 5 e A na posição 0 são escolhidos para trocarem de posição com C na posição 3.

C	B	A	B	A	B

VTR = 20 ✓



4.1.3 Fase de Refinamento

Neste contexto, aplica-se a meta-heurística ILS utilizando os mesmos movimentos descritos acima de permutação de posições, inserção e remoção de cópias, com intuito de fugir de ótimos locais, aceitando resultados que não melhorem a solução. E logo uma nova busca local foi feita assim que alguma das perturbações era aplicada.

As estruturas de vizinhança utilizadas foram:

- *Swap*
    - Troca de um elemento da lista por outro escolhido aleatoriamente.
    - Troca de um ou dois elementos da lista por outros escolhidos aleatoriamente.

- *Shift*
    - Reposição de três elementos na lista, pulando de um a três elementos.
    - Reposição de três a quarto elementos na lista, pulando de um a três elementos.

- *Double Bridge*
    - Considerando um círculo, reposicionamento dos elementos que estão entre 0° e 30° com os que estão entre 180° e 210°. (aprox 9% dos elementos).
    - Considerando um círculo, reposicionamento dos elementos que estão entre 0° e 60° com os que estão entre 180° e 240°. (aprox 17% dos elementos).
    - Considerando um círculo, reposicionamento dos elementos que estão entre 0° e 90° com os que estão entre 180° e 270°. (aprox 25% dos elementos).

Capítulo 5

5 Resultados Computacionais

Abaixo será apresentado o ambiente que foi utilizado para desenvolver e testar as aplicações, a forma de obtenção das instâncias dos testes, assim como os resultados dos experimentos computacionais realizados, por fim o comparativo entre os procedimentos envolvidos e dos resultados obtidos.

Os procedimentos foram avaliados individualmente quanto ao valor obtido da função objetivo (sendo considerados os valores mínimo e médio), como também o esforço computacional medido em tempo (sendo considerados os valores mínimo, médio e máximo) necessário para obter tais valores aceitáveis (estabelecendo tempo máximo total de 180 segundos), e então foram comparados entre si e também comparados com os resultados encontrados na literatura.

Os experimentos foram executados utilizando o cenário descrito na seção 5.1, enquanto que as instâncias são descritas na seção 5.2.

Na seção 5.3, foram apresentados os experimentos computacionais resultantes dos procedimentos utilizando apenas a meta-heurística ILS, assim como as parametrizações utilizadas e os resultados obtidos.

Já em relação à seção 5.4, são descritos os experimentos resultantes das meta-heurísticas GRASP e ILS trabalhadas em conjunto, as parametrizações e os resultados.

5.1 Descrição do ambiente

O algoritmo proposto foi desenvolvido na linguagem de programação C++, com o uso da versão 4 da Dev-C++.

O hardware utilizado durante todo o processo foi um notebook Dell Inspiron 14R, cujo processador é Intel® Core™ 2 Duo 64 T6600 2.20GHz e memória RAM 4GB DDR2. O sistema operacional utilizado é Windows 7 Ultimate com arquitetura x64.

Todos os experimentos computacionais foram executados no mesmo ambiente.

5.2 Instâncias

O algoritmo utilizou como instâncias de testes, as que foram utilizadas por [RAMOS, 2012], assim como também foram geradas novas instâncias para que fosse possível obter novos comparativos de acordo com os resultados obtidos. As instâncias A1 à A3 variam o número de símbolos n = 3, 5, 7, 11 e 16.

- Instâncias A1: Composto por instâncias semelhante às utilizadas no trabalho de [COROMINAS, 2007].
- Instâncias A2: Composto por instâncias semelhantes às utilizadas no trabalho de [PESSOA, 2008].
- Instâncias A3: Composto por instâncias semelhantes às utilizadas no trabalho de [RAMOS, 2012].
- Instâncias A4: Composto por instâncias fixando o número de símbolos n (3, 5, 7, 11, 16, 19, 22) e o total de unidades em 20 e 25, variando aleatoriamente o número de cópias de cada símbolo.

5.3 Experimentos computacionais utilizando GRASP e ILS

Os grupos de instâncias foram executados variando aleatoriamente os movimentos possíveis de perturbação, detalhados a seguir.

- *Shift* de três ou quatro elementos movendo entre uma e três posições
- *Swap* de um ou dois elementos
- *Double Bridge* variando entre 30, 60 e 90 graus.

Foram mensurados os valores da VTR, média dos resultados, valor máximo obtido e também a diferença do resultado em relação à solução ótima, não excedendo o limite de 180 segundos.

A unidade utilizada para representar os resultados de esforço computacional foi o tempo medido em segundos com 2 casas decimais.

Aproveitamentos de execuções, médios e máximos representam, respectivamente, a porcentagem das execuções que resultaram em alguma solução antes de ser abortado

devido à limitação de tempo, a melhoria média dos resultados obtidos comparativamente e por fim a melhoria máxima dos resultados obtidos.

	Tempo		Aproveitamento		
	Mínimo	**Médio**	**Execuções**	**Médio**	**Máximo**
A2	0,0520	0,1760	100,00%	67,95%	67,95%
A3	0,0230	0,1667	100,00%	64,11%	64,11%
A4	0,0210	0,2094	93,33%	70,16%	71,49%

Tabela 17 – Instância com três símbolos

	Tempo		Aproveitamento		
	Mínimo	**Médio**	**Execuções**	**Médio**	**Máximo**
A2	0,0460	0,5577	100,00%	54,86%	54,86%
A3	0,0810	1,1525	97,20%	48,38%	50,26%
A4	0,1960	1,7130	88,46%	48,53%	51,13%

Tabela 18 – Instância com cinco símbolos

	Tempo		Aproveitamento		
	Mínimo	**Médio**	**Execuções**	**Médio**	**Máximo**
A2	1,3540	2,0729	80,00%	56,86%	59,56%
A3	1,5900	2,7191	91,67%	71,35%	72,15%
A4	1,9420	2,6901	71,42%	58,49%	60,99%

Tabela 19 – Instância com sete símbolos

	Tempo		Aproveitamento		
	Mínimo	**Médio**	**Execuções**	**Médio**	**Máximo**
A2	2,6920	4,2705	57,95%	52,13%	55,27%
A3	1,2710	3,3052	61,22%	53,20%	59,71%
A4	1,4320	4,0143	55,55%	49,56%	53,58%

Tabela 20 – Instância com onze símbolos

	Tempo		Aproveitamento		
	Mínimo	**Médio**	**Execuções**	**Médio**	**Máximo**
A2	4,9250	15,3760	71,89%	40,76%	59,75%
A3	4,1640	15,6312	65,75%	36,95%	58,88%
A4	3,7710	16,1706	69,23%	38,04%	57,20%

Tabela 21 – Instância com dezesseis símbolos

n	A1		A2		A3		A4	
	Inicial	**Final**	**Inicial**	**Final**	**Inicial**	**Final**	**Inicial**	**Final**
3	13120	690	12634	852	14438	953	13358	890
5	30147	10415	28520	3391	30157	3380	30223	3487
7	40714	26386	42928	12368	42873	12153	26439	3051
11	84517	44608	69108	30818	74943	33212	76942	34200
16	97561	25678	106241	32790	102764	31520	106099	32646
19	114856	14357	124437	15654	125952	16144	118320	14790
22	146133	5577	148623	3600	134246	3668	135955	2880

Tabela 22 – Comparação dos resultados da função objetivo para cada instância

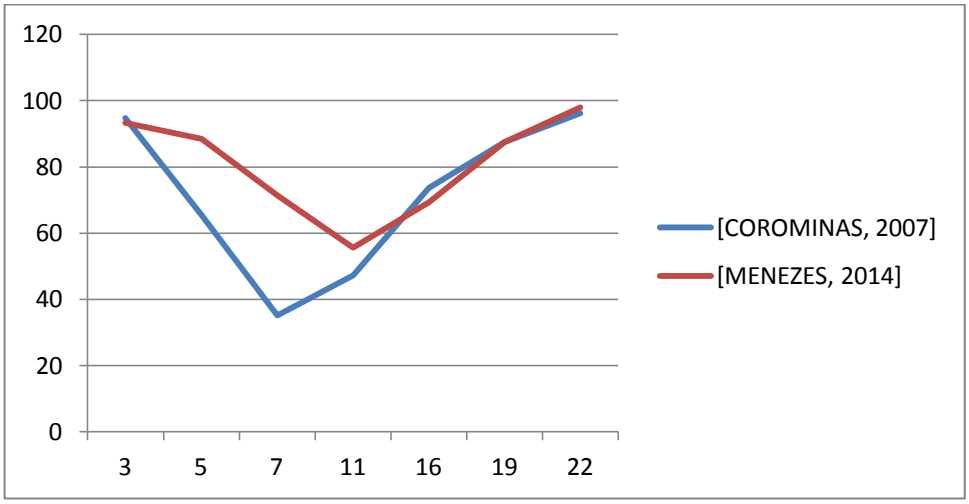


Figura 13 – Comparativo do aproveitamento de resultado entre instâncias A1 e A4 (porcentagem/número de símbolos)

Em seu estudo, Corominas (2007) não detalha em termos do tempo de execução, apenas cita que suas execuções utilizando MILP não excedem 180 segundos, só foi possível verificar a eficácia da proposta na visão do aproveitamento de resultados, ou seja, a porcentagem na redução da função objetivo, que é minimizar a variabilidade no tempo de resposta, demonstrado na figura 14.

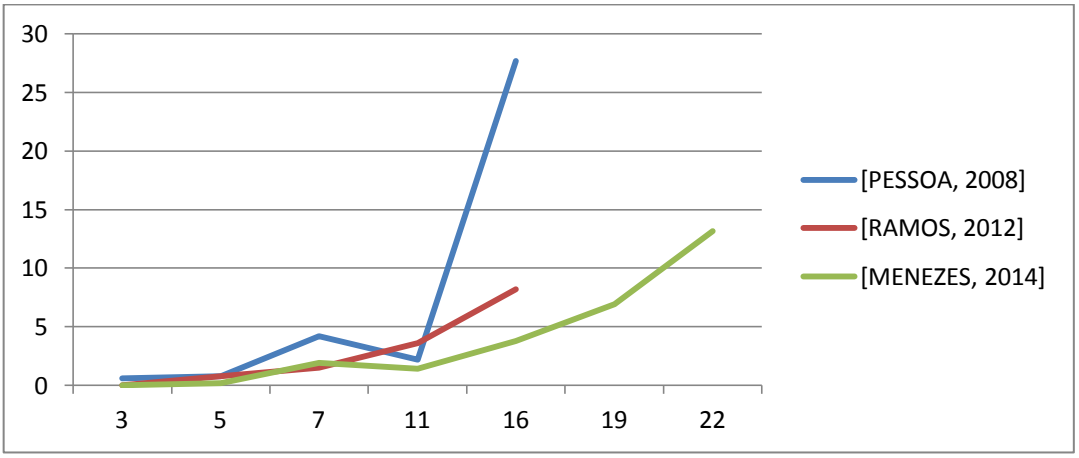


Figura 14 – Comparativo do tempo médio das execuções entre as instâncias A2 à A4 (tempo(s)/símbolo)

Com relação ao tempo de execução mostrados na figura 15, verificamos que o trabalho proposto obtém resultados satisfatórios em tempos inferiores tanto em relação ao trabalho proposto por Pessoa (2008), principalmente com instâncias de maior valor, quanto ao trabalho de Ramos (2012), que demonstra resultados bem equiparados.

Embora os tempos das perturbações com melhor aproveitamento para as instâncias onde n = 7 e 11 tenham sido maiores, mas dependendo dos critérios de satisfação, podem ser considerados irrelevantes se considerado o ganho com o aproveitamento dos resultados obtidos em geral.

Capítulo 6

6 Considerações Finais

O presente estudo propôs a utilização dos procedimentos Greedy Randomized Adaptive Search Procedure (GRASP) e Iterated Local Search (ILS) para aplicação no Problema da Variabilidade do Tempo de Resposta.

Ambas motivadas pela sua utilização bem sucedida em problemas variantes ao da Variação do Tempo de Resposta, e então optadas como forma de verificar sua eficácia no problema original.

Posteriormente, por meio dos resultados, foi visto que tais procedimentos também trazem resultados satisfatórios ao problema, de forma que trazem melhorias nos resultados de até 36,19% e em uma média de 3,92 segundos.

Trabalhos futuros

Como sugestão para trabalhos futuros, aplicar paralelismo no algoritmo proposto com intuito de verificar sua viabilidade técnica em experiências que requerem mais poder de processamento.

Outra sugestão seria aplicar um conjunto de funções de teste para verificar possíveis parametrizações a serem aplicadas em problemas específicos, variantes ao Problema da Variabilidade do Tempo de Resposta.

REFERÊNCIAS

AGUILERA, B.; ROLI, A.; SAMPELS, M. Hybrid Metaheuristics: An Emerging Approach to Optimization. **Springer**. (Studies in Computational Intelligence), 2008.

CHAVES, A.A. Uma Meta-heurística Híbrida com Busca por Agrupamentos Aplicados a Problemas de Otimização Combinatória. **Dissertação (Mestrado)** - Instituto Nacional de Pesquisas Espaciais, 2009.

CORMEM, T.; LEISERSON, C.; RIVEST, R.; STEIN, C. Algoritmos - Teoria e Prática. Editora Campus, vol. 2, 763-791, 2002.

COROMINAS, A.; KUBIAK, W.; MORENO, N. Response time variability, 2004. Disponível em: < http://upcommons.upc.edu/e-prints/handle/2117/572 >. Acessado em Julho 2013.

COROMINAS, A.; KUBIAK, W.; MORENO, N. Response time variability. **Journal of Scheduling**, v. 10, p. 97-110, 2007.

FEO, T.A.; RESENDE, M.G.C. Greedy randomized adaptive search procedures. **Journal of Global Optimization**, p.109–133, 1995.

GAREY, M. R.; JOHNSON, D. S. Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Co, 1979.

GLOVER, F.; MARTI, R. Tabu Search. Metaheuristic Procedures for Training Neutral. **Networks**, v. 36, p. 53–69, 2006.

KUBIAK, W. Fair sequences. Handbook of Scheduling: Algorithms, Models and Performance Analysis, **Leung**, J. Y-T, 2004. Disponível em: < http://goo.gl/9Xqg5V >. Acessado em Maio de 2014.

LOURENÇO, H. R., MARTIN, O. C., STÜTZLE, T. Iterated Local Search – Handbook on MetaHeuristics, 2001. Disponível em: < http://arxiv.org/pdf/math/0102188.pdf >. Acessado em Abril de 2014.

LOURENCO, H. R.; MARTIN, O. C.; STÜTZLE, T. Iterated Local Search. Handbook on Metaheuristics. **Springer**, Heidelberg, 2003.

MILTENBURG, J.G. Level schedules for mixed-model assembly lines in just-in-time production systems. **Management Science**, 35, 192 – 207, 1989.

MLADENOVI'C, N.; HANSEN, P. Variable Neighborhood Search. Computers & Operations. **Research**, v. 24, n. 11, p. 1097–1100, 1997.

MONDEN, Y. Toyota Production Systems Industrial Engineering and Management Press, **Norcross**, GA, 1983.

MORENO, N. Solving the Product Rate Variation Problem (PRVP) of large dimensions as an assignment problem, Doctoral Thesis, DOE, ETSEIB-UPC, 2002.

PESSOA, B. Meta-heurísticas Aplicadas à Geração de Carrossel no Sistema Brasileiro de Tv Digital. **Dissertação (mestrado)**. Universidade Federal da Paraíba, Programa de Pós Graduação em Informática, 2008.

RAMOS, D.G. Meta-heurísticas GRASP e ILS aplicadas ao problema da Variabilidade do Tempo de Download em ambientes de Tv Digital. **Dissertação (mestrado)**. Universidade Federal da Paraíba, Programa de Pós Graduação em Informática 2012.

RESENDE, M.G.C. Greedy Randomized Adaptative Search Procedures (GRASP). **Technical Report, ATT Labs Research**, 1998.

SCHRIJVER, A. Theory of Linear and Integer Programming, 1988.

TONETTO, L.M. O papel das heurísticas no julgamento e na tomada de decisão sob incerteza. **Estudos de Psicologia**, v. 23, n. 2, p. 181–189, 2006.