Optimização de um Plano de Marketing Directo

para o Sector da Banca

Daniel Barata

Trabalho de projecto para obtenção do Grau de Mestre em

ESTATÍSTICA E GESTÃO DE INFORMAÇÃO

Instituto Superior de Estatística e Gestão da Informação

Universidade Nova de Lisboa

Orientadores: Professor Doutor Fernando Bação

Professor Doutor Victor Lobo

Novembro, 2010

Resumo

O marketing directo é uma estratégia de *CRM* cada vez mais usada no sector da banca, por isso o número de campanhas existentes neste tipo de negócio tem sido cada vez maior (Cohen, 2002). O plano de marketing de um banco é formado por um conjunto de acções ou campanhas com diferentes objectivos estratégicos e negociado entre os diferentes departamentos envolvidos no processo. Para cada uma das campanhas que formam o plano de marketing são seleccionados os melhores clientes, de acordo com os objectivos da campanha. Estes clientes são seleccionados com base em informação residente em grandes bases de dados e recorrendo muitas vezes a técnicas de *Data Mining* e descoberta de conhecimento conhecidas como *Database Marketing* (Pinto, 2006). No entanto, os clientes a incluir em cada campanha são escolhidos tendo em conta apenas os objectivos dessa campanha. Desta forma alguns clientes podem ser seleccionados para várias campanhas propondo diferentes produtos. Isto faz com que alguns clientes sejam sobrecarregados com várias comunicações. Além disso, outros factores como a capacidade dos canais não são considerados, porque cada campanha é implementada de forma isolada sem conhecer as restantes concorrentes (Cohen, 2002). Esta tese tem como objectivo principal o desenvolvimento de um processo que permita optimizar o plano de campanhas de um banco numa perspectiva holística tendo em vista a maximização do retorno do investimento. O processo deverá permitir seleccionar os clientes a contactar em cada campanha, tendo em consideração as várias combinações possíveis e um conjunto de restrições impostas. Apesar do actual desenvolvimento computacional, vamos verificar que este problema é de elevada complexidade devido à sua grande dimensão. Neste trabalho vai ser apresentado e analisado em detalhe o problema de optimização de campanhas e irá ser avaliado o forte impacto que o processo de optimização poderá ter no retorno de investimento.

Palavras-chave

Marketing Directo; Plano de Marketing; Optimização de Campanhas; *Cross-Selling*; Sistemas de apoio à decisão; *Database Marketing*;

ÍNDICE

ÍNDICE DE FIGURAS E QUADROS

GLOSSÁRIO

ATM (Automated Teller Machine) – São caixas electrónicas que permitem efectuar todo o tipo de operações bancárias. Em Portugal, a maioria são partilhadas por vários bancos por isso também são chamadas de multibanco.

CRM (Customer Relationship Management) – Estratégias implementadas pelas empresas para efectuar a gestão de relacionamento com os clientes.

Cross-Selling – Termo em Inglês que significa venda cruzada, ou seja, é a venda de produtos adicionais ao mesmo cliente. Normalmente, são produtos relacionados ou complementares a produtos que o cliente já tem.

Data Mining – Processo ligado às tecnologias de informação no qual se procura traduzir os dados existentes na empresa em informação útil para o negócio.

Data Warehouse – Conjuntos de bases de dados de grande dimensão onde é armazenada centralmente toda a informação de uma empresa.

Database Marketing – Conjunto de técnicas e metodologias que permitem efectuar uma gestão de marketing e de relacionamento com o cliente.

Datamart – Base de dados de reduzida dimensão (quando comparada com um *data warehouse*) mais focada para uma área de negócio específica.

E-Mail – Correio electrónico, é um meio de envio de mensagens digitais via *internet* ou outras redes computacionais.

RAM (Random Access Memory) – Memória interna e volátil de acesso rápido. É uma espécie de memória primária em dispositivos digitais.

ROI (Return of Investiment) – É o rácio calculado entre o montante ganho num determinado investimento versus o montante investido ou gasto nesse investimento.

SMS (Short Messages Service) – Meio de envio de mensagens via telefones e telemóveis.

SP (Stored Procedure) – Rotina do *MS SQL Server* desenvolvida em código *T-SQL* para implementar uma determinada funcionalidade.

T-SQL (Transact Structured Query Language) – Linguagem de programação usada pelo *MS SQL Server* e especializada em interrogações e operações em bases de dados.

Capitulo I – INTRODUÇÃO

1. Introdução

1.1 Introdução

O marketing directo é uma estratégia cada vez mais usada nas empresas com o objectivo de aumentar a rentabilidade e construir uma relação de sucesso com os clientes (Gronroos, 1996). Num mercado competitivo como o actual onde os clientes são cada vez mais exigentes e informados, é fundamental controlar a relação com os clientes e usar uma estratégia de marketing mais táctica. O marketing directo é uma das ferramentas usada nessa estratégia (Gronroos, 2009). O desenvolvimento das tecnologias de informação e a evolução nas comunicações, com a massificação do uso da internet e do e-mail, contribuíram muito para o desenvolvimento desta área (Stone & Jacobs, 2001). A banca de retalho é uma indústria onde esta estratégia é usada com bastante regularidade. A quantidade de informação que a banca recolhe dos seus clientes e a grande competitividade desta área de negócio, fazem da banca uma área de excelência para a aplicação destas estratégias de Marketing (Cohen, 2002).

A informação existente nas organizações é recolhida e armazenada em grandes bases de dados conhecidos por *data warehouses*. Estes repositórios de dados são estruturas que permitem organizar e manipular os dados usando técnicas de *data mining* e de descoberta de conhecimento com base em algoritmos computacionais (Pinto, 2006).

Os bancos usam a grande quantidade de informação que têm ao seu dispor para melhor conhecerem e compreenderem os seus clientes. Desta forma podem oferecer aos clientes os produtos e serviços que vão ao encontro das suas necessidades, tendo em conta os seus gostos e preferências, criando uma relação de lealdade e confiança (Lovelock & Wirtz, 2006). O surgimento de novos canais, como a *internet*, o *e-mail* ou o *SMS* permitem que estas ofertas sejam efectuadas de forma rápida, personalizada e com custos reduzidos (Cohen, 2002).

Os planos de marketing dos bancos são definidos e alinhados com a estratégia da organização, para a execução destes planos são disponibilizados elevados orçamentos

que permitem efectuar milhares de contactos com ofertas dirigidas e personalizadas aos clientes.

Com este projecto pretendo desenvolver e implementar um processo que permita optimizar de uma forma holística o plano de campanhas de um banco. O objectivo final é seleccionar os melhores contactos e ofertas a executar de acordo com um conjunto de aspectos que têm de ser considerados: a rentabilidade da oferta; os custos envolvidos; a capacidade dos canais; a canibalização das ofertas; a saturação dos clientes; as taxas de resposta esperadas; etc. Esta optimização é feita sobre os contactos previamente seleccionados para cada uma das campanhas, mas numa perspectiva holística do plano de marketing.

1.2 Justificação do Tema

O marketing directo tem um impacto significativo nas vendas de qualquer empresa (Stone & Jacobs, 2001), nos bancos esta é também uma realidade evidente. Todos os anos os bancos disponibilizam elevados orçamentos para executarem os planos de marketing estrategicamente planeados. Este orçamento é distribuído pelas várias campanhas de acordo com as estratégias definidas para os segmentos, produtos, etc. O orçamento é desta forma um dos recursos limitados que tem de ser muito bem gerido. Outro dos recursos também limitado é a capacidade dos canais. Nem sempre os canais têm capacidade para dar resposta aos milhares de contactos que são gerados num plano de marketing, esta capacidade tem de ser gerida e muito bem aproveitada.

Esta necessidade de optimização resulta essencialmente de três factores: dos custos envolvidos face aos orçamentos limitados, da capacidade limitada dos canais envolvidos e da política de comunicação com os clientes. Tendo em consideração estes factores, o processo de optimização tem como principal objectivo a maximização de uma função de acordo com a estratégia da empresa, normalmente é a função *ROI* (*Return on Investment*) (Berry, 2006).

O processo de optimização vai permitir um ganho considerável no *ROI* em marketing directo sem alterar nenhum dos processos operacionais envolvidos (Berry, 2006). No

artigo de Berry (Berry, 2006) o autor apresenta dois casos de estudo onde analisa dois planos de contactos baseados nos mesmos clientes. Estes casos são o exemplo claro do ganho obtido com um processo de optimização deste género. Neste artigo podemos verificar que o segundo plano de contactos, que usou um processo de optimização, obteve relativamente ao primeiro plano, um ganho de vinte e nove por cento e uma redução de custos na ordem dos seis por cento.

Outra grande vantagem do processo de optimização é permitir construir cenários para identificar limitações nos recursos disponibilizados e analisar os impactos de determinadas decisões. Por exemplo, é possível criar vários cenários com diferentes capacidades de canais para verificar até que ponto os canais estão bem dimensionados para as necessidades existentes e analisar o impacto na alteração dessas capacidades. Outro exemplo, é simular diferentes valores de orçamentos para as campanhas, para perceber até que ponto um determinado investimento em algumas campanhas terá ou não um impacto significativo na rentabilidade total do plano de marketing (Cohen, 2002).

O processo de optimização de campanhas é a última fase no processo de gestão de campanhas e trata-se de uma etapa fundamental para uma eficiente gestão e optimização de recursos.

1.3 Objectivo da Investigação

Este trabalho de investigação surge para dar resposta a um problema existente no departamento de marketing da instituição bancária onde eu exerço a minha actividade profissional. Desta forma o objectivo será resolver o problema em concreto da área de gestão de marketing directo deste departamento.

O objectivo principal deste trabalho de projecto é o desenvolvimento de um processo de optimização de campanhas que permita maximizar a rentabilidade de um plano de marketing no sector da banca.

Este processo deverá consistir no desenvolvimento de num programa informático baseado num algoritmo que permita optimizar a selecção de contactos de um plano de marketing, tendo como objectivo a maximização da rentabilidade e tendo em consideração um conjunto de restrições e condições impostas à partida.

Outro objectivo deste trabalho é também apresentar uma análise detalhada do problema de optimização de campanhas de modo a efectuar o devido enquadramento teórico e a respectiva formalização deste problema.

Embora não sendo propriamente um objectivo definido à partida, existe a expectativa que a criação de um processo deste género permita efectuar a simulação de cenários, de modo a analisar eventuais impactos do reforço de investimento neste tipo de estratégias de marketing.

Em termos concretos e mais objectivos, as metas estabelecidas para este trabalho de investigação podem ser colocadas em duas vertentes:

<u>Numa perspectiva teórica:</u>

1. Efectuar uma análise e descrição detalhada do problema com todas as suas características e considerações.
2. Efectuar o enquadramento teórico e a respectiva formalização do problema de optimização de campanhas.

<u>Numa perspectiva prática:</u>

1. Desenvolver um algoritmo que permita criar um processo totalmente automático para seleccionar os melhores contactos a efectuar considerando as seguintes situações:

    a. Respeitar um conjunto de restrições e condições impostas à partida.
    b. Permitir uma rentabilidade superior à obtida com o processo actual.
    c. Permitir atingir a rentabilidade máxima, ou uma rentabilidade próxima da máxima, com a execução do plano de marketing directo.

Capitulo II - REVISÃO DA LITERATURA

2. Revisão da Literatura

2.1 Introdução

Neste capítulo e numa primeira fase, vou fazer uma descrição teórica do processo de gestão de campanhas e do problema que proponho resolver, baseando esta descrição na revisão da literatura efectuada. Esta fase irá levar à respectiva formalização do problema.

Numa fase a seguir, irei efectuar uma comparação deste problema com alguns problemas semelhantes mais conhecidos e efectuar um enquadramento do problema em termos de complexidade computacional.

Depois disso, vou descrever algumas das soluções encontradas na revisão da literatura e explicar as diferentes abordagens e metodologias para este tipo de problemas.

Por último, este capítulo termina com uma síntese e conclusão da revisão de literatura efectuada.

2.2 Enquadramento Teórico

2.2.1 Descrição do processo de gestão de campanhas

O plano de marketing de um banco é planeado para um período de tempo pré-definido e limitado. Este período varia normalmente entre três a doze meses. O plano é constituído por um conjunto de acções de marketing directo que podem ter diferentes objectivos e características (Doyle, 2005). Cada uma destas acções ou campanhas são geridas por um conjunto de pessoas, os gestores de campanhas. O gestor de campanhas é responsável por planear, executar e acompanhar as campanhas que tem ao seu cargo.

O plano de marketing tem um orçamento pré-definido que pode estar distribuído por segmento, produtos, canal, etc. O orçamento é um recurso importante a ter em conta,

na medida em que condiciona fortemente a gestão das campanhas estando directamente ligado com o custo de cada contacto.

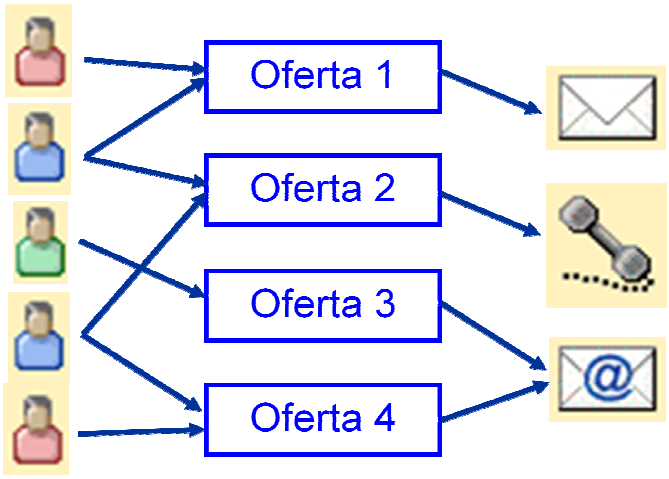
As campanhas que fazem parte do plano são desenhadas e definidas pelos gestores de campanhas em conjunto com os gestores de segmentos e gestores de produtos. Nesta definição acaba por ocorrer uma espécie de negociação entre as partes envolvidas, por existirem diferentes interesses que resultam das diversas funções e visões de negócio. Na definição e planeamento das campanhas são também envolvidas outras áreas mais operacionais como a área de comunicação e a área de canais (Cohen, 2002).

Uma campanha é organizada segundo uma determinada estrutura. Nessa estrutura são definidos, por exemplo, os segmentos, produtos, canais envolvidos, etc. Para cada campanha são ainda definidos os custos de cada contacto, as taxas de sucesso esperadas e os proveitos. Esta definição e feita tendo em consideração os objectivos da campanha.

Uma campanha pode ter diferentes ofertas e formas de comunicação específica para cada uma dessas ofertas (Doyle, 2005). Por exemplo, uma campanha pode envolver dois segmentos, os "Adultos Seniores" e os "Jovens Adultos" e para cada um destes segmentos podem ser colocados produtos diferentes, um cartão crédito prata e um cartão de crédito *gold*. Desta forma será necessário definir os custos e proveitos de acordo com o segmento e o produto envolvido. Para os clientes "Adultos Seniores" pode ser usado o canal de *mail* (correio tradicional) e para os clientes do segmento "Jovens adultos" pode ser usado o canal *e-mail* (correio electrónico). Estes canais podem ser usados como um primeiro contacto e posteriormente existir um novo contacto efectuado pelo canal balcão. Neste caso, o segundo contacto serviria de *follow up* (termo em Inglês para designar um contacto de acompanhamento, normalmente um segundo contacto) (Cohen, 2002).

No sector da banca os canais mais utilizados são: o correio tradicional (carta, mensagens nos extractos mensais e envelopes, etc); o telefone (*call center* e telemarketing); o *e-mail* (correio electrónico); a *internet* (página do banco); os balcões; as lojas de atendimento diferenciado (e.g. lojas habitação); os postos de atendimento automático (*ATMs*) e o *SMS* (Doyle, 2005).

Figura 1. Num plano de marketing cada cliente é alvo de múltiplas ofertas através de diferentes canais.



2.2.2 O processo de selecção de contactos

Depois de definido e planeado, o plano de marketing é implementado pelos gestores de campanhas. Este plano é posto em prática tendo em conta as prioridades e datas de planeamento de cada campanha. Desta forma o gestor de campanhas verifica qual a próxima campanha que deve implementar e selecciona para essa campanha os clientes que são mais apropriados de acordo com os objectivos da campanha e tendo em conta o orçamento previsto para essa campanha (Doyle, 2005).

Normalmente, no processo de selecção de clientes para uma campanha, é efectuado o cálculo de uma taxa de resposta que indica qual a probabilidade de sucesso desse contacto, ou seja, do cliente responder positivamente à oferta. No cálculo desta taxa de resposta são usadas técnicas de *data mining* para a construção de modelos de propensão que se baseiam em informação histórica de cada cliente e no comportamento geral de clientes com as mesmas características (Cohen, 2002).

Num plano de marketing directo são geridos milhares de contactos com os clientes, usando diferentes canais e propondo diferentes produtos e serviços. Estes contactos vão sendo gerados ao longo de semanas e meses, seleccionando os clientes para cada campanha sem ter em conta as restantes campanhas. Desta forma, alguns clientes mais "atractivos" podem ser seleccionados para mais que uma campanha e os contactos acabam por se sobrepor uns aos outros correndo o risco de sobrecarregar e

saturar os clientes com ofertas e comunicações. Estes fenómenos são conhecidos por canibalização e saturação e podem levar a uma situação inversa ao objectivo do plano de marketing, que é o afastamento dos clientes (Selby, 2007).

Para evitar os fenómenos da canibalização e saturação, os clientes podem ser seleccionados para uma campanha apenas se ainda não foram seleccionados para nenhuma campanha anterior. Além disso, podem ser consideradas algumas regras e politicas de restrição de contactos, como por exemplo, verificar se o cliente não se importa de ser contactado por um determinado canal, ou verificar quando foi o último contacto efectuado ao cliente para dar um intervalo de tempo mínimo entre cada contacto (Berry, 2006).

A utilização de algumas regras e políticas de contacto podem evitar a sobrecarga de contactos para os clientes, no entanto esta poderá não ser a forma ideal de o fazer. Neste processo será necessário ter em consideração que alguns dos clientes seleccionados para uma determinada campanha poderão ser mais rentáveis se forem seleccionados para outra campanha que está planeada para implementar a seguir, e que nesse caso, seriam excluídos por já terem sido seleccionados numa campanha anterior (Berry, 2006).

Com a implementação de um processo de optimização de campanhas, o processo de selecção dos clientes para cada campanha terá de ser ligeiramente modificado. A selecção dos clientes terá de passar a ser efectuada de uma única vez para todas as campanhas do plano de marketing, ao contrário do processo referido anteriormente, em que os clientes para cada campanha podiam ser seleccionados ao longo do tempo. Deste modo, seleccionando os clientes para todas as campanhas logo à partida, será possível criar um processo que optimize o plano de marketing de modo a que seja escolhido para cada cliente o contacto com maior rentabilidade, desde que existam condições para o fazer.

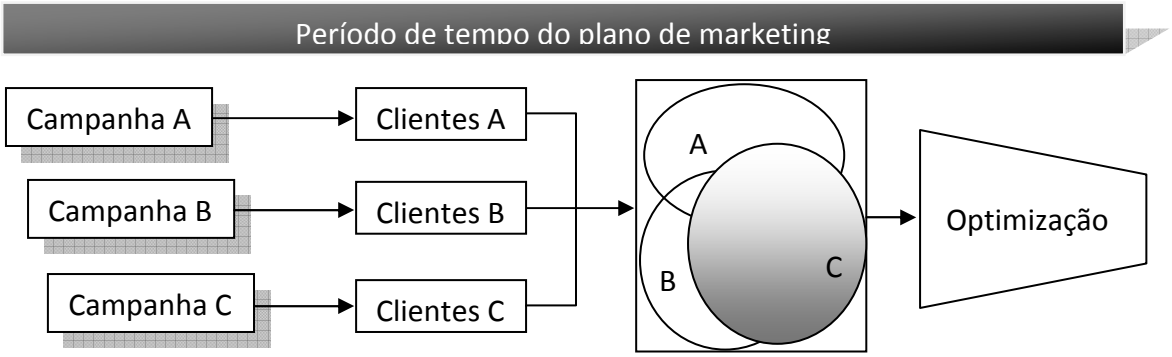
A implementação de um processo de optimização pode também implicar ligeiras alterações nos critérios de selecção dos clientes para cada campanha. Alguns dos critérios de selecção de clientes considerados logo à partida em cada campanha, podem ser desprezados ou alargados. No final vai existir sempre o processo de

optimização que irá garantir que esses critérios serão respeitados. Por exemplo, o nº máximo de contactos por canal ou o nº máximo de contactos para cada cliente é garantido pelo processo de optimização, por isso não necessita de ser um critério a considerar na altura da selecção de clientes para cada campanha.

O processo de optimização de campanhas será a última etapa de um longo processo de implementação do plano de marketing. Este processo para além de resolver a questão da sobrecarga e saturação de contactos para os mesmos clientes, tem ainda o objectivo de optimizar os contactos do plano de marketing do ponto de vista da rentabilidade, tendo em conta os recursos disponíveis (Cohen, 2002).

Com base na anterior descrição do processo de gestão de campanhas foi criado o seguinte fluxograma para melhor entender este processo. Este esquema mostra como deve ser implementado o plano de marketing tendo em conta um processo de optimização.

Figura 2. Processo de implementação de campanhas



2.3 Pressupostos e formalização do Problema

2.3.1 Pressupostos iniciais

Suponhamos que temos um plano de marketing formado por um conjunto de campanhas para ser executado num determinado período no tempo.

Cada campanha deste plano contém um determinado conjunto de oportunidades de contactos. Um contacto é uma comunicação para um cliente, feita através de um canal, com uma oferta de produto ou serviço dirigida a esse cliente.

Cada campanha apenas pode ter uma oferta de um produto ou serviço. Esta questão pode facilmente ser ultrapassada. No caso de querermos remeter para o processo de optimização de campanhas a escolha da melhor oferta para o mesmo cliente, tendo em conta a rentabilidade de todo o plano de marketing, podemos criar uma nova campanha onde são colocados os contactos para os mesmos clientes, mas com uma oferta de produto ou serviço diferente. Desta forma o processo de optimização irá seleccionar os contactos com a maior rentabilidade, tendo em conta as restrições impostas à capacidade de cada canal e ao orçamento de cada campanha.

Do ponto de vista teórico, podemos ter para o mesmo cliente e para a mesma campanha vários canais de comunicação. Nesse caso, o processo de optimização seria responsável por escolher o canal de comunicação para o contacto com esse cliente.

Na realidade, muitas vezes o canal de comunicação é definido à partida para cada campanha. Podendo existir vários grupos de contacto dentro da mesma campanha, onde são utilizados diferentes canais para cada grupo de contacto. Neste caso o canal é único para cada par {campanha, cliente}, sendo que uma campanha poderá usar mais que um canal de comunicação.

Do ponto de vista prático, vamos considerar neste trabalho que o canal de comunicação é definido à partida para cada contacto. Isto porque é o que acontece na realidade na instituição bancária onde eu trabalho e à qual se aplica este trabalho de projecto. Neste caso, vamos ter um número de contactos mais reduzido do que teoricamente seria de prever.

Para cada contacto de uma campanha devemos ter definido os seguintes valores:

1. Valor de custo associado ao contacto, tendo em conta todos os factores que devem ser considerados para o cálculo deste custo (custos fixos e custo variáveis). Por exemplo, o custo do canal de comunicação e a forma de comunicação, a atribuição de brindes, etc.

2. Valor da taxa de sucesso associada ao contacto, que indicará a probabilidade do cliente responder positivamente à oferta, ou seja, a probabilidade do contacto ter sucesso.

3. Valor do proveito de acordo com a oferta associada ao contacto, ou seja, o valor que o sucesso do contacto vai trazer para o banco.

Com estes três valores será calculado o valor de ganho de cada contacto com base na seguinte fórmula:

Valor de ganho = valor de proveito * taxa de sucesso – valor de custo

Desta forma, podemos definir um contacto como um registo formado pela seguinte campos:

Campo	Descrição
Campanha	Indicação da campanha onde pertence o contacto.
Cliente	Indicação do cliente que vai ser alvo do contacto.
Canal	Indicação do canal que vai ser usado para o contacto.
Valor de custo	Valor do custo associado ao contacto e que será considerado no cálculo dos custos da campanha para comparar com o orçamento disponível para a campanha.
Valor de ganho	Valor do ganho associado ao contacto e que será considerado no cálculo da rentabilidade da campanha.

Restrições a considerar

Para este trabalho vamos considerar três tipos de restrições: ao nível da campanha, do cliente e do canal. Estes tipos vão dar origem à seguinte lista de restrições:

1. Cada cliente apenas pode ser contactado no âmbito de uma campanha.

2. O processo de optimização deve ter em conta a capacidade de cada canal.

3. O processo de optimização deve ter em conta o orçamento de cada campanha.

4. O processo de optimização deve ter em conta o número mínimo e máximo de contactos que são seleccionados para cada campanha.

Deste modo é necessário definir à partida para cada campanha qual o orçamento disponível e qual o mínimo de contactos a considerar. Para os canais é também necessário definir qual a capacidade máxima de cada canal.

Objectivo do processo de optimização

Como já foi referido na introdução deste trabalho, o objectivo fundamental do processo de optimização de campanhas é aumentar a ROI, optimizando o investimento que é feito com o marketing directo. Na prática trata-se de maximizar o valor do ganho obtido com os contactos efectuados no plano de marketing, tendo em conta os pressupostos já apresentados.

Para atingir o objectivo proposto o processo de optimização terá que indicar ("decidir") quais os melhores contactos a efectuar de modo a obtermos o valor máximo de ganho, tendo em conta as restrições existentes.

O problema de optimização de campanhas é desta forma um problema de investigação operacional de optimização onde se pretende maximizar uma determinada função, neste caso o lucro, sujeito a determinadas condições impostas (Cohen, 2002).

2.3.2 Formalização do problema

Suponhamos que o nosso plano de marketing é formado por um conjunto de *n* clientes, em *m* campanhas usando *p* canais. Isto faz um total de contactos possíveis, no máximo igual a *n* x *m* x *p*.

A expressão “no máximo” é usada aqui porque nem todos os clientes estão seleccionados para todas as campanhas. Quanto à questão do canal, como já foi referido anteriormente, na prática vamos assumir que o canal é único para cada par {campanha, cliente}, logo vamos ter que *p* = 1. Desta forma temos um conjunto de no máximo *n* x *m* contactos.

Se considerarmos que um contacto é formado por um cliente *j*, numa campanha *i* que usa o canal *k*, esta decisão pode ser representada pela variável x_{ijk}que indica a decisão de efectuar o contacto na campanha *i* ao cliente *j* usando o canal *k* (Cohen, 2002).

Para cada contacto, o processo de optimização deve decidir se esse contacto deve ou não ser seleccionado para ser efectuado.

Desta forma, podemos representar a decisão de efectuar um contacto da seguinte forma:

$x_{ijk} = 1$, caso o contacto na campanha *i* ao cliente *j* pelo canal *k*, seja seleccionado.

$x_{ijk} = 0$, caso o contacto na campanha *i* ao cliente *j* pelo canal *k*, seja rejeitado.

Estamos assim perante um problema de programação linear inteira e binária. Porque se trata de uma função linear onde as variáveis da função que pretendemos optimizar apenas podem tomar o valor de 0 ou 1(Cohen, 2002).

Sendo assim, a função que pretendemos maximizar é formada pela seguinte expressão, em que g_{ijk}é o ganho obtido com o contacto na campanha *i* ao cliente *j* usando o canal *k*.

Maximizar $\sum_{i=1}^{m} \sum_{\mathrm{j}=1}^{\mathrm{n}} \sum_{\mathrm{k}=1}^{\mathrm{p}} g_{ijk} x_{ijk}$

A acrescentar a este objectivo temos ainda o facto de existirem algumas condições que têm de ser respeitadas, são as restrições já referidas anteriormente.

1. Limites de custos da campanha de acordo com o orçamento previsto

Para cada campanha i, temos definido um determinado orçamento b_i esta restrição é formalizada pela seguinte expressão, onde c_{ijk} é o custo associado ao contacto na campanha i do cliente j pelo canal k.

$$\sum_{j=1}^{n}\sum_{k=1}^{p} c_{ijk}x_{ijk} \leq b_i \qquad i \varepsilon M = \{1 \dots m\}$$

2. Limite máximo de capacidade dos canais.

Para cada canal k, temos definido um limite máximo de capacidade u_k, esta restrição é formalizada pela seguinte expressão.

$$\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ijk} \leq u_k \qquad k \varepsilon P = \{1 \dots p\}$$

3. Limites máximos e mínimos de contactos por campanha

Para cada campanha i, temos definido um limite mínimo s_i de contactos (em alguns casos será zero) e um limite máximo r_i de contactos (em alguns casos será relativamente elevado porque a restrição orçamental já limita este número). Estas restrições são formalizadas pelas seguintes expressões:

$$\sum_{j=1}^{n}\sum_{k=1}^{p} x_{ijk} \leq r_i \qquad i \varepsilon M = \{1 \dots m\}$$

$$\sum_{j=1}^{n}\sum_{k=1}^{p} x_{ijk} \geq s_i \qquad i \varepsilon M = \{1 \dots m\}$$

4. Limites máximos de contactos por cliente

Cada cliente só poderá ser contactado no âmbito de uma campanha, isto será o mesmo que dizer que para cada cliente apenas pode ser contactado uma única vez. Esta restrição é formalizada pela seguinte expressão:

$$\sum_{i=1}^{m}\sum_{k=1}^{p} x_{ijk} \leq 1 \qquad j \varepsilon N = \{1 \dots n\}$$

A formalização completa do problema de optimização de campanhas é então colocada da seguinte forma:

Maximizar $\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{k=1}^{p} g_{ijk}x_{ijk}$

Sujeito a

$\sum_{j=1}^{n}\sum_{k=1}^{p} c_{ijk}x_{ijk} \leq b_i$ i ε M = {1 … m}

$\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ijk} \leq u_k$ k ε P = {1 … p}

$\sum_{j=1}^{n}\sum_{k=1}^{p} x_{ijk} \leq r_i$ i ε M = {1 … m}

$\sum_{j=1}^{n}\sum_{k=1}^{p} x_{ijk} \geq s_i$ i ε M = {1 … m}

$\sum_{i=1}^{m}\sum_{k=1}^{p} x_{ijk} \leq 1$ j ε N = {1 … n}

$x_{ijk} \in \{0,1\}$ i ε M, j ε N, k ε P

Um pequeno exemplo

Vejamos um exemplo concreto de tamanho reduzido para melhor percebermos o problema.

Suponhamos, por exemplo, que temos as campanhas m1 e m2, os clientes c1 e c2, e os canais k1 e k2. São criados no total 4 contactos, o_{111}, o_{121}, o_{212} e o_{222}, definidos do seguinte modo:

1. o_{111} - Contacto na campanha m1, ao cliente c1, usando o canal k1
2. o_{121} - Contacto na campanha m1, ao cliente c2 usando o canal k1
3. o_{212} - Contacto na campanha m2, ao cliente c1 usando o canal k2
4. o_{222} - Contacto na campanha m2, ao cliente c2 usando o canal k2

O custo e o ganho associado a cada contacto são os seguintes:

Contacto	**Custo**	**Ganho**
o_{111} - Contacto na campanha m1, ao cliente c1, usando o canal k1	12,4	90,5
o_{121} - Contacto na campanha m1, ao cliente c2 usando o canal k1	10	50,6
o_{212} - Contacto na campanha m2, ao cliente c1 usando o canal k2	14,6	60
o_{222} - Contacto na campanha m2, ao cliente c2 usando o canal k2	11,2	50

Suponhamos que temos as seguintes restrições:

a) A campanha m1 tem um orçamento limitado a 15€
b) O canal k1 só tem capacidade para efectuar 1 contacto
c) A campanha m2 tem de ter pelo menos um contacto
d) Cada cliente só pode ser contactado uma vez

As variáveis x_{ijz} são variáveis binárias que tomam o valor 0 ou 1 de acordo com a decisão de efectuar ou não o respectivo contacto.

Este problema pode ser assim apresentado como um problema de programação inteira, onde o objectivo é maximizar a função *f* tendo em contas as restrições 1 a 4.

O problema é então formalizado da seguinte modo:

Maximizar $f = 90.5\ x_{111} + 50.6\ x_{121} + 60\ x_{212} + 50\ x_{222}$

Sujeito a

$$12.4\ x_{111} + 10\ x_{121} \leq 15$$

$$x_{111} + x_{121} \leq 1$$

$$-x_{212} - x_{222} \leq 0 \quad (x_{212} + x_{222} > 0)$$

$$x_{111} + x_{212} \leq 1$$

$$x_{121} + x_{222} \leq 1$$

$$x_{111},\ x_{122},\ x_{211} \text{ e } x_{222} \in \{0,1\}$$

A solução do exemplo apresentado, pela sua pequena dimensão, é fácil de obter mesmo sem recorrer a nenhum algoritmo matemático. De acordo com as condições impostas, a melhor solução seria escolher os contactos o_{111} e o_{222}, ou seja, neste caso a solução seria:

$$x_{111} = 1 \qquad x_{122} = 0 \qquad x_{211} = 0 \qquad x_{222} = 1$$

Uma forma intuitiva de encontrar a solução óptima para este tipo de problemas é por enumeração exaustiva, ou seja, enumerar todas as combinações possíveis e eliminar as que não cumprem as restrições. Para as restantes avaliar qual é a que dá o maior ganho. Para o exemplo anterior, onde temos apenas quatro variáveis, esta solução até podia ser fácil de implementar. Mas quando estamos perante um problema com alguns milhares de variáveis pode ser uma tarefa muito mais complexa. É o que acontece quando temos um plano de marketing directo de um banco formado por milhares e até milhões de potenciais contactos e temos de escolher quais os contactos que devemos efectuar de modo a maximizar o retorno de investimento.

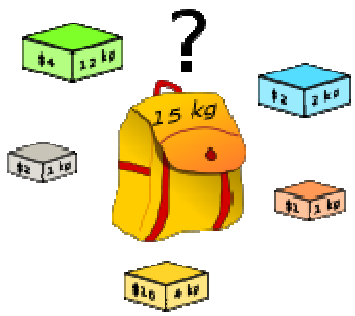
Na investigação operacional este tipo de problemas é bastante conhecido. Alguns exemplos mais conhecidos são, o problema do caixeiro-viajante, o problema da mochila e o problema geral de afectação. São todos casos de programação linear inteira semelhantes ao problema apresentado de optimização de contactos para o departamento de marketing de um banco.

Para compreender melhor este problema vou analisar de seguida dois destes problemas mais conhecidos que têm em muitos aspectos semelhanças com o problema de optimização de campanhas.

2.4 Comparação com problemas conhecidos

2.4.1 *Multiple-Knapsack* Problema (*MKP*)

Figura 3. Que objectos colocar na mochila?



Vamos agora verificar que uma redução do problema de optimização de campanhas é o conhecido problema da mochila (em inglês, *Knapsack problem*).

O problema da mochila é caracterizado pelo facto de termos N objectos com pesos diferentes (w_j) e com diferentes valores (p_j), o problema consiste em decidir quais os objectos a colocar na mochila de modo a maximizar o valor total da mochila, tendo em conta a capacidade máxima de peso que a mochila pode transportar (c).

Como o problema se trata de decidir se cada objecto é colocado ou não na mochila, estamos perante um problema de decisão, ou seja variáveis binárias ($x_j = 0$ ou $x_j = 1$), por isso este problema é chamado de *0-1 Knapsack Problem* (Martello & Toth, 1990).

O problema pode ser formalizado num problema de programação inteira deste modo:

Maximizar $\sum_{j=1}^{n} p_j x_j$

Sujeito a $\sum_{j=1}^{n} w_j x_j \leq c$

$x_j \in \{0,1\}$ $j \in N$

Se consideramos que a mochila pode ter vários compartimentos M = (1 … m) e cada um desses compartimentos tem uma capacidade de peso máxima $C_i = (C_1, \ldots C_m)$, temos então o problema da mochila múltipla (em inglês, *0-1 Multiple-Knapsack problem*). Este problema é formalizado do seguinte modo (Martello & Toth, 1990):

Maximizar $\sum_{i=1}^{m} \sum_{j=1}^{n} p_j x_{ij}$

Sujeito a $\sum_{j=1}^{n} w_j x_{ij} \leq c_i$ $i \in M = \{1 \ldots m\}$

$\sum_{i=1}^{m} x_{ij} \leq 1$ $j \in N = \{1 \ldots n\}$

$x_{ij} \in \{0,1\}$ $i \in M, j \in N$

O problema *0-1 Multiple-Knapsack problem* (*MKP*) é um conhecido problema NP-Difícil (Martello & Toth, 1990) e é exactamente o mesmo do problema de optimização de

campanhas se considerarmos apenas a restrição do limite máximo do orçamento em cada campanha. Neste caso, a mochila seria o plano de marketing a efectuar, os compartimentos da mochila seriam as campanhas (considerando que cada campanha tem apenas um produto) e os clientes seriam os objectos a colocar na mochila, neste caso o plano de marketing. Tal como no problema *MKP*, em que cada objecto tem um determinado valor para a mochila, mas tem um determinado peso dentro de cada compartimento (compartimento esse que tem um limite de peso), também no problema de optimização de campanhas cada contacto de um cliente tem um determinado valor para o plano de marketing, mas representa um determinado custo dentro de cada uma das campanhas (que têm orçamentos limitados).

Apesar do problema de optimização de campanhas ter ainda um conjunto adicional de restrições que podem complicar ainda mais o processo de encontrar a solução óptima, tal como no problema *MKP*, a dicotomia entre a maximização do ganho e o controlo dos custos das campanhas é um dos problemas principais. Se por um lado temos de nos preocupar em maximizar o ganho, escolhendo os contactos que têm um ganho elevado, por outro lado temos de nos preocupar com o controlo dos custos, pois cada campanha tem um orçamento limitado. Além disso, os orçamentos das várias campanhas são definidos de acordo com a estratégia da empresa e os objectivos de cada campanha, logo podem ter valores muito diferentes. Temos de ter sempre presente que o objectivo principal deste processo é a maximização do ganho do plano de marketing como um todo.

Vejamos um exemplo muito simples: suponhamos que temos três contactos para três clientes distintos na mesma campanha, um contacto tem um ganho de 100€ e os outros dois têm um ganho de 70€ cada um. Suponhamos ainda que o contacto de 100€ tem um custo de 10€ e os restantes dois contactos têm um custo de apenas 5€, e que o limite de custos para executar esta campanha é apenas de 10€. Neste caso, apesar do contacto de 100€ ser o que tem um ganho mais elevado, o ideal seria escolher os restantes dois contactos de 70€, pois o ganho no total seria de 140€. O contacto com o maior ganho à partida devia assim ficar de fora, porque se fosse seleccionado esgotava o orçamento para a campanha e desse modo o ganho total seria apenas de 100€.

Encontrar a solução óptima para o problema *MKP*, tal como para o problema simplificado de optimização de campanhas, pode ser uma tarefa muito complicada. Uma forma de encontrar a solução óptima, seria comparar todas as combinações possíveis e verificar qual a que produzia melhores resultados. Esta solução até podia ser considerada para problemas de pequenas dimensões. No entanto, para problemas de larga escala, como é o caso do plano de marketing, o peso computacional desta solução é impraticável. O problema cresce exponencialmente com o número de variáveis (Martello & Toth, 1990), o número de combinações possíveis para um conjunto de n variáveis é de 2^n. Se tivermos em conta que um plano de marketing pode ter milhares de clientes em várias campanhas, o número total de contactos (variáveis) pode ser muito elevado (mais de um milhão), o que daria um total de combinações completamente intratável.

2.4.2 *Generalized Assignment Problem (GAP)*

Outro problema de optimização também muito conhecido e debatido na literatura é o problema geral de afectação (em inglês, *Generalized Assignment Problem*).

O *Generalized Assignment Problem* (*GAP*) é caracterizado por existirem N tarefas que tem de ser afectadas a M agentes, que por exemplo podem ser máquinas, que executam essas tarefas. Cada máquina tem uma determinada capacidade para processar cada tarefa e tem um determinado custo.

O problema consiste em determinar qual a tarefa a atribuir a cada máquina de modo a não exceder a capacidade dessa máquina e de modo a minimizar os custos de execução de todas as tarefas. Este problema tem muitas aplicações na vida real, como por exemplo, a distribuição de tarefas em equipas de projectos, a distribuição de processos em redes de computadores, a alocação de espaços de memória, etc. (Balachandar & Kannan, 2009)

O *GAP* pode ser formulado num problema de programação inteira do seguinte modo:

Minimizar $\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$

Sujeito a $\sum_{j=1}^{n} a_{ij} x_{ij} \leq b_i$ $i \in M = \{1 \ldots m\}$

$\sum_{i=1}^{m} x_{ij} = 1$ $j \in N = \{1 \ldots n\}$

$x_{ij} \in \{0,1\}$ $i \in M, j \in N$

O problema de optimização de campanhas pode ser comparado com o *GAP* se consideramos apenas duas das suas restrições, a restrição relacionada com o orçamento de cada campanha e a restrição de contactar cada cliente apenas uma vez. Nesse caso, o problema seria o de afectar cada cliente a uma determinada campanha de modo a maximizar o ganho da campanha e a não exceder o orçamento previsto dessa campanha, ou seja, tendo em conta a capacidade dessa campanha em termos de orçamento. No caso do *GAP*, o objectivo é minimizar os custos tendo em conta o custo que cada máquina tem para executar cada tarefa, no caso do problema de optimização de campanhas o objectivo é maximizar o valor do ganho tendo em conta o ganho obtido em contactar cada cliente em cada campanha.

Um aspecto importante a ter em conta é o facto da formalização do GAP poder ser facilmente transformada num problema de maximização, multiplicando as variáveis dos custos por -1 (Martello & Toth, 1990), ou seja, outra forma de formalizar o problema é a seguinte:

Maximizar $\sum_{i=1}^{m} \sum_{j=1}^{n} -c_{ij} x_{ij}$

Sujeito a $\sum_{j=1}^{n} a_{ij} x_{ij} \leq b_i$ $i \in M = \{1 \ldots m\}$

$\sum_{i=1}^{m} x_{ij} = 1$ $j \in N = \{1 \ldots n\}$

$x_{ij} \in \{0,1\}$ $i \in M, j \in N$

O GAP é um conhecido problema de optimização considerado NP-Dificil (Balachandar & Kannan, 2009).

2.4.3 Comparação com o problema de Optimização de Campanhas

Apesar da comparação com alguns problemas conhecidos de optimização, o problema de optimização de campanhas tem algumas características especiais que fazem com que tenhamos de analisar o problema tendo em conta essas mesmas características. No entanto, os estudos e investigação existentes para os problemas mais conhecidos, como o *MKP* e *GAP*, podem ser muito úteis para ajudar na análise e resolução deste problema. Por isso, ao longo desta investigação serão referenciados alguns artigos relacionados com estes problemas mais conhecidos.

Uma conclusão que podemos desde logo retirar na comparação do problema com estes problemas mais conhecidos é a sua classificação em termos de complexidade computacional. Dada a sua semelhança com os problemas *MKP* e *GAP*, talvez seja agora mais fácil de afirmar que o problema de optimização de campanhas é também ele um problema NP-Dificil. Aliás, em ambos os casos, o problema de optimização de campanhas pode ser analisado como uma simplificação de cada um destes dois problemas.

A principal motivação para a resolução do problema de optimização de campanhas, tal como nos problemas *MKP* e *GAP*, é a gestão e optimização dos recursos disponíveis para efectuar determinadas tarefas que trazem um valor acrescentado.

Porque os possíveis contactos do plano de marketing já estão identificados à partida, a questão determinante no problema é perceber quais os melhores contactos a efectuar de modo a optimizar os recursos disponíveis para efectuar esses contactos. Este balanço entre o ganho e o custo/capacidade é o factor determinante que está em causa em qualquer um destes problemas de optimização.

2.5 Complexidade computacional

No conjunto dos problemas de optimização existem problemas que na prática não têm uma solução exacta, ou seja, não existe um algoritmo que na realidade consiga encontrar a solução óptima. É importante percebermos e identificarmos este tipo de

problemas para podermos desenvolver e avaliar soluções alternativas (Davis, Sigal, & Weyuker, 1994).

De forma a podermos avaliar a eficiência de um algoritmo foi criado o conceito de "algoritmo de tempo polinomial" (em Inglês, *Polynomial Time Algorithms*). Um algoritmo é considerado de tempo polinomial se é resolvido num período de tempo limitado por uma função polinomial, ou seja, o tempo de execução do algoritmo é dado pela expressão O(p(n)) onde p é uma função polinomial e n é um parâmetro de entrada com o tamanho do problema. Qualquer algoritmo que não seja resolvido num tempo limitado por uma função polinomial é considerado um algoritmo de tempo exponencial (em Inglês, *Exponencial Time Algorithms*) (Garey & Johnson, 1979).

De forma informal, os algoritmos de tempo polinomial são considerados tratáveis e os algoritmos de tempo exponencial são considerados intratáveis. Esta classificação tem em conta a complexidade no cálculo do tempo do algoritmo de acordo com o tamanho inicial do problema (Garey & Johnson, 1979).

Na teoria da complexidade computacional, os problemas de decisão considerados tratáveis pertencem à classe P (do Inglês, *Polinomial time*). Esta classe representa o conjunto de problemas de decisão que têm um algoritmo de tempo polinomial conhecido (Cornelius, 2001).

A classe NP (do Inglês, *Nondeterministic Polinomial time*) representa o conjunto de problemas de decisão cuja solução é verificada em tempo polinomial, incluindo os problemas de decisão para os quais não se conhece um algoritmo polinomial mas que podem ser resolvidos em tempo polinomial por um algoritmo que dá a solução aproximada. É claro que um problema da classe P também pertence à classe NP, pois a sua solução pode ser verificada em tempo polinomial, logo, temos que $P \subseteq NP$ (Cornelius, 2001).

Na classe NP estão os problemas considerados intratáveis e dentro dessa classe estão os problemas NP-completo que são considerados os mais difíceis dentro dos

intratáveis e para os quais não se conhece uma solução determinística capaz de ser executada em tempo polinomial (Garey & Johnson, 1979).

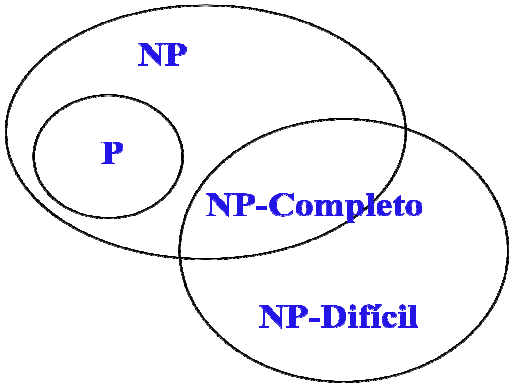
A classe NP-completa é o conjunto de problemas da classe NP para os quais qualquer outro problema da classe NP pode ser reduzido. Ou seja, p ε NP-completo, se qualquer problema da classe NP pode ser reduzido a p. NP-completo é um subconjunto de problemas da classe NP com a seguinte propriedade: se for encontrado um algoritmo polinomial para um destes problemas, então existe um algoritmo polinomial para qualquer problema de NP (Garey & Johnson, 1979).

A classificação de NP-completo referida anteriormente é normalmente aplicada em problemas de decisão, no entanto existem outros problemas de optimização para além dos problemas de decisão. Alias, os problemas de decisão são na realidade uma simplificação de um problema mais geral de optimização, qualquer problema de decisão pode ser formalizado como um problema de optimização. Para problemas gerais de optimização é utilizada a expressão NP-difícil (em Inglês, NP-hard) para classificar os problemas mais complexos de optimização (Garey & Johnson, 1979).

A classificação NP-difícil é aplicada para os problemas que não pertencem necessariamente à classe NP mas que tem uma dificuldade idêntica aos problemas mais difíceis da classe NP, ou seja, os problemas NP-completos. Um problema é NP-difícil se existir um problema em NP ao qual esse problema se possa reduzir polinomialmente. Desta forma, um problema da classe NP-completo pertence à classe NP-difícil e à classe NP (Garey & Johnson, 1979).

A figura apresentada a seguir sintetiza de forma clara as classificações referidas anteriormente relativas à complexidade computacional dos problemas de optimização. Podemos verificar que a classe NP é um conjunto enorme na qual se incluem as classes P e NP-completo. A classe NP-difícil é um outro conjunto que tem alguns elementos em comum com a classe NP e que contém a classe NP-completo (Gurari, 1989).

Figura 4. Classificação dos problemas de optimização em termos de complexidade computacional



Fonte: Adaptação (Gurari, 1989)

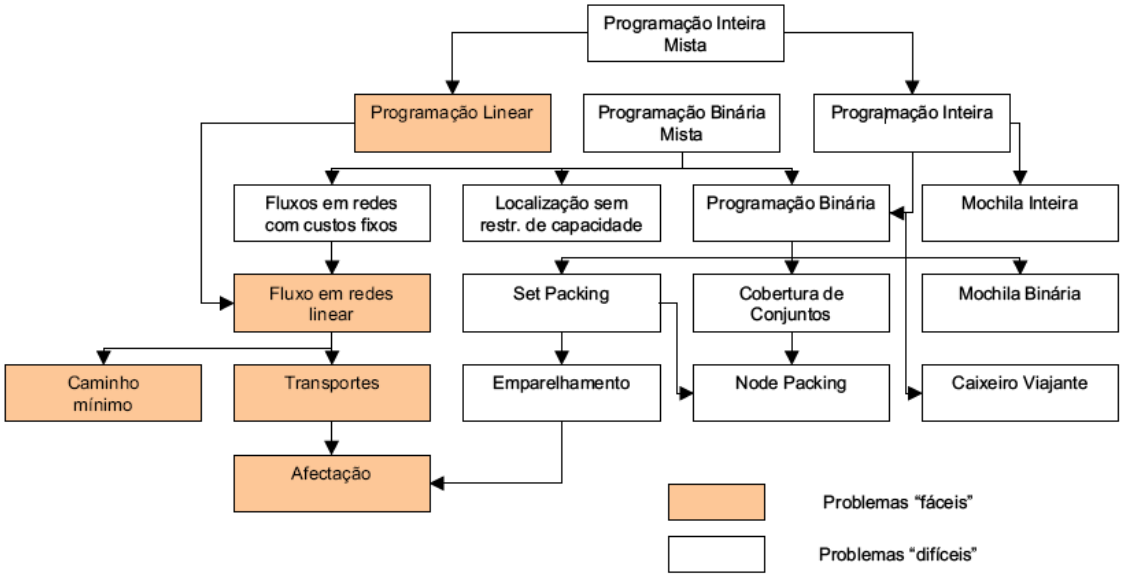
O caso concreto do problema de optimização de campanhas é um problema classificado como NP-difícil e um problema de decisão NP-completo .

Na análise de comparação do problema de optimização de campanhas com outros problemas conhecidos verificámos que este problema é uma simplificação do problema da mochila múltipla ou do problema geral de afectações, que são conhecidos problemas classificados como NP-difíceis (Nobibon, Leus, & Spieksma, 2008). Do mesmo modo estes mesmos problemas enquanto problemas de decisão são classificados de NP-completos (Martello & Toth, 1990).

Para além dos dois problemas já referidos (MKP e GAP), existem muitos outros problemas identificados e classificados como NP-completos ou NP-difíceis, entre eles podemos destacar o problema do caixeiro viajante (em Inglês, *Traveling Salesman Problem*) (Garey & Johnson, 1979), o problema da soma de subconjuntos (em Inglês, *Subset-sum*) e o problema do empacotamento (em Inglês, *Bin-Paking*) (Martello & Toth, 1990).

Como podemos observar no fluxograma em baixo, os problemas de optimização de programação binária, como é o caso do problema de optimização de campanhas, é classificado como NP-dificil.

Figura 5. Hierarquia e classificação de diferentes problemas de optimização



Fonte: (Nemhauser & Wolsey, 1988)

2.6 Abordagens existentes na literatura

Nas pesquisas efectuadas durante a revisão da literatura encontrei vários artigos onde a questão da optimização do plano de marketing é colocada e nos quais são apresentadas algumas soluções. Alguns autores analisam este problema de um ponto de vista mais científico e académico, enquanto que outros apresentam soluções mais práticas propostas por grandes fabricantes de Software multinacionais, como é o caso da *SAS* e da *IBM*.

Existem diferentes formas de colocar e abordar a questão da optimização de campanhas. O processo de gestão de campanhas é um processo que envolve várias etapas, desde o planeamento do plano de marketing até à sua implementação. O problema da optimização depende sempre da forma como este processo está implementado em cada situação. Para a situação em concreto que estamos a analisar nesta tese o processo de gestão de campanhas já foi apresentado anteriormente nas secções 2.2 e 2.3 deste capítulo.

Análise do artigo de *M. D. Cohen* (2002)

Nas pesquisas efectuadas sobre a questão do problema de optimização de campanhas, o artigo que mais se assemelha à situação em concreto que pretendemos resolver nesta tese, na abordagem e colocação do problema, é o artigo de de *M. D. Cohen* (Cohen, 2002) da empresa *SAS Institute Inc*.

Neste artigo o problema é colocada de forma muito idêntica ao apresentado nesta tese nas secções 2.2 e 2.3 deste capítulo. No entanto, neste artigo a questão é colocada a um nível diferente. Neste caso o problema é colocado ao nível da campanha, na selecção dos melhores clientes para os vários tipos de oferta.

A situação é a mesma mas colocada a um nível diferente. Neste artigo de *Cohen* o problema é colocado ao nível da campanha onde existem vários produtos. No nosso caso, o problema é colocado ao nível do plano de marketing onde existem várias campanhas. No entanto, como cada campanha é composta por apenas um produto, a situação é exactamente a mesma.

Neste artigo, *Cohen* começa por descrever a solução actual, que na maioria dos casos, em particular na banca, é usada para tentar resolver este problema. O processo normalmente usado é baseado num algoritmo que escolhe o cliente mais rentável dentro de cada tipo de oferta. Em primeiro lugar é calculado um ganho efectivo aproximado para cada oferta ao cliente *i* do produto *j* pelo canal *k*. Depois são ordenadas as ofertas por tipo de oferta e para cada tipo são ordenados os clientes, por ordem decrescente de rentabilidade que cada cliente tem nesse tipo de oferta. Os clientes vão sendo seleccionados passo a passo até o número desejado de clientes para esse tipo de oferta seja atingido. De seguida é feito o mesmo para o próximo tipo de oferta e assim sucessivamente para todos os tipos.

Este algoritmo é apresentado por *Cohen* como uma fraca solução, uma vez que não tem em consideração algumas restrições de negócio e de recursos, e além disso, a solução encontrada fica longe de ser a óptima, uma vez que considera as ofertas ordenadas dentro de cada produto, a solução só seria óptima para o caso de termos um único produto.

Esta critica que *Cohen* faz no seu artigo ao processo normalmente usado pela maioria dos bancos, é no fundo a motivação que nos leva à necessidade de criar um processo de optimização de campanhas. A questão colocada ao nível do plano de marketing é exactamente a mesma, como as campanhas são tratadas de forma isolada, o processo só seria óptimo no caso do plano de marketing ter apenas uma campanha. O objectivo é optimizar o plano de marketing como um todo.

Numa segunda fase, o autor explica qual seria a solução ideal para o problema. A questão é colocada como um problema de optimização linear inteira, em que o objectivo é maximizar uma função que é colocada como uma função de decisão. O objectivo é decidir quais as ofertas a efectuar a cada cliente e através de que canal, tendo em conta um conjunto de restrições impostas. A formalização do problema é colocada de forma muito idêntica à formalização apresentada nesta tese na secção 2.6 desde capítulo, no entanto, como já foi referido, neste artigo a questão é colocada ao nível da campanha, enquanto que nesta tese é colocada ao nível do plano de marketing.

Existem apenas duas diferenças na forma como este artigo formaliza o problema, a primeira está relacionado com o orçamento da campanha e a segunda relacionada com a taxa mínima do *ROI*.

Relativamente à primeira, como no artigo de *Cohen* a questão de optimização é colocada ao nível da campanha, a restrição ao valor do orçamento é colocada ao nível mais alto, ou seja, existe apenas um valor total de orçamento para a campanha. No nosso caso, a questão do orçamento também é colocada ao nível da campanha só que neste caso é o nível mais baixo, ou seja, existe um orçamento definido para cada campanha que terá de ser respeitado, o orçamento não é um valor único mas é um valor em função de cada campanha.

Relativamente à segunda diferença, no artigo é acrescentada uma restrição à formalização do problema para garantir que o valor de *ROI* da campanha ultrapasse um mínimo pré-estabelecido. No nosso caso esta questão não se coloca, pois na selecção de clientes para cada campanha já é garantida uma taxa de sucesso mínima para cada contacto ser seleccionado. Neste caso o problema dá algum foco à questão

de selecção dos clientes para a campanha. No nosso caso, a questão de selecção dos clientes não é um problema à partida, pois essa questão é tratada com um processo independente. No nosso caso, a questão fundamental é a optimização dos recursos e a selecção da melhor oferta para cada cliente, tendo em conta que se pretende não perturbar a relação com o cliente com demasiadas campanhas.

A solução ideal para este problema, segundo *Cohen*, passa então por resolver o problema formalizado sob a forma de programação linear inteira binária. Esta solução apresenta um grande problema de computação por ser uma tarefa demasiado morosa e com necessidade de grandes recursos computacionais. O autor apresenta algumas contas, dando como exemplo uma situação com um milhão de clientes e dez produtos, o que daria um total de 10 milhões de variáveis de decisão que resultariam num total de $2^{10,000,000}$ de combinações. O autor refere que usando os métodos habituais de programação inteira binária, como por exemplo o método *branch and bound*, um problema desta dimensão é impossível de resolver.

Por último, o autor apresenta uma solução que considera prática para o problema, que é a solução implementada no *Software* da *SAS* para resolver esta situação.

Esta solução usa um processo de *clustering* onde os clientes são agrupados pela sua proximidade, tendo em conta o valor de rentabilidade e do custo.

Desta forma os clientes são agrupados em grupos com rentabilidade e custos aproximados, transformando assim o problema de programação linear inteira binária num problema de programação linear mais fácil de resolver e de menor dimensão.

O problema deixa de ser uma questão de decisão, onde as variáveis são zero ou um de acordo com a decisão a tomar, e passa a ser a uma questão de identificar quantos elementos de cada grupo devem de ser seleccionados para maximizar o ganho. As variáveis deixam de ser binárias e passam a ser inteiras.

Esta solução é uma heurística aproximada para chegar perto da solução óptima. Neste caso, cada cliente dentro do seu grupo é considerado de forma igual, o que na realidade sabemos que não é verdade. No entanto, esta solução levanta um outro problema que é o facto de que para cada grupo de clientes devem ser seleccionados

apenas uma proporção desses clientes, de acordo com a solução encontrada. Para resolver esta questão será necessário resolver um novo problema de afectação para cada agregação de clientes.

No artigo, *Cohen* refere que a resolução simples do problema de afectação pode não garantir que todas as restrições serão respeitadas, para contornar esse problema teriam de ser adicionadas algumas restrições na resolução do problema de afectação. Mas esta situação não é explicada com mais detalhe no artigo, o próprio autor refere este facto.

O artigo de *Cohen* o problema de optimização de campanhas é colocado de uma forma muita próxima do problema que é colocado nesta tese. O autor descreve a grande dimensão do problema e apresenta uma solução que passa pela redução e simplificação do problema inicial.

No entanto, esta solução apresentada no artigo de *Cohen* implica resolver uma serie de novos problemas. Primeiro, vem o problema no agrupamento dos clientes e a questão de quantos grupos devem ser criados. Provavelmente o número de grupos criados terá de ser relativamente elevado para se obterem bons resultados. O autor refere no artigo que devem ser feitos estudos futuros para esta questão. Depois vem a questão da resolução dos problemas de afectação em cada agrupamento de clientes, esta questão pode envolver a resolução de centenas e até milhares de problemas de afectação.

A solução apresentada neste artigo não foi considerada viável neste meu trabalho de investigação porque implicava a resolução de uma serie de novos problemas e porque o artigo não apresentou solução para alguns desses problemas.

Artigo de *D. A. Selby* (Selby, 2007)

Noutro artigo de *D. A. Selby* (Selby, 2007) publicado no jornal de investigação da *IBM* é apresentado um algoritmo proposto pela *IBM* para resolver o problema de optimização de contactos de um plano de marketing.

No artigo o problema é colocado como uma optimização de eventos, que são contactos com os clientes ao longo do tempo. A questão é escolher para cada cliente o evento que melhor rentabiliza o plano de contactos tendo em consideração a rentabilidade e o custo associado a cada par {cliente, evento} e tendo em conta algumas restrições de orçamentos e limites de eventos.

O algoritmo apresentado é baseado numa heurística em que o par {cliente, evento} com maior potencial de rentabilidade é escolhido à partida numa lista de potenciais clientes e eventos. O passo a seguir é eliminar dessa lista todos os eventos para o cliente previamente seleccionado, partindo do princípio que cada cliente é alvo de apenas um evento. Depois é seleccionado o próximo par {cliente, evento} com maior potencial de rentabilidade, e assim sucessivamente. Em cada selecção de clientes são verificadas as restrições de orçamento e de nº máximo de eventos.

A optimização de eventos é feita executando este algoritmo diariamente e tendo em conta o histórico de eventos. Os eventos de histórico deverão entrar no cálculo do nº de dias para o cliente poder ser alvo de um novo evento.

Segundo o autor, este algoritmo é bastante rápido, o que permite executar o processo numa base diária, e assim seleccionar ao longo do tempo os eventos para cada cliente de modo a evitar os problemas de canibalização e saturação.

Neste artigo o processo de optimização de contactos é colocado como um conjunto de eventos que deverão ser seleccionados para os clientes ao longo do tempo. O problema é colocado numa estratégia de gestão de relacionamento com o cliente, com o objectivo de não saturar o cliente com demasiadas ofertas ou com ofertas menos rentáveis. O conceito de campanhas, que é o resultado das estratégias de marketing da empresa e faz parte da estrutura do plano de marketing, não é tido em consideração. Logo, todas as questões relacionadas com as campanhas não são consideradas neste artigo, como o caso dos contactos máximos e mínimos por campanha, e os orçamentos das campanhas.

Outro aspecto também não referido neste artigo, é a questão das capacidades dos canais de comunicação usados nos eventos, sabemos que alguns dos canais têm capacidades limitadas e este facto pode condicionar a gestão e selecção dos eventos.

O artigo de *Selby* dá uma visão um pouco simplificada do problema de optimização de campanhas, no entanto, o algoritmo apresentado pode ser um ponto de partida para construir um processo de optimização. Como refere o autor, tem a grande vantagem de ser rápido e assim poder ser usado numa base diária, pode ainda ser usado com mais frequência na construção de cenários.

Artigos de *Kim, Yoon e Moon*

Dois dos autores que mais se têm dedicado a este tema de optimização de campanhas são *Kim* e *Yoon*. Num artigo recentemente publicado (Kim & Yoon, 2009), mostram do ponto de vista mais teórico, toda a complexidade computacional deste tipo de problemas, e provam matematicamente que encontrar a solução óptima para este tipo de problemas é uma tarefa muito complexa, que não é exequível num tempo prático e aceitável.

Este artigo vem na sequência de outros anteriores, onde os mesmos autores sugerem alguns algoritmos para resolver esta questão, apesar de alguns desses algoritmos serem considerados completamente intratáveis devido à dimensão e complexidade do problema. Num artigo publicado em Março de 2006 (Kim & Moon, 2006) podemos analisar alguns algoritmos determinísticos para encontrar a solução óptima e outros algoritmos heurísticos que procuram uma solução aproximada para o problema. Um dos algoritmos apresentado neste artigo é baseado em programação dinâmica, que usa um processo recursivo para tentar chegar à solução. No entanto, os autores consideram que para um conjunto de contactos demasiado elevado o processo é de tal modo demorado que se torna intratável.

Num outro artigo (Kim, Yoon, & Moon, 2008), os mesmos autores *Kim* e *Yoon,* juntamente com um outro autor, *Moon*, tentam uma nova abordagem. Através de um método de *Lagrange* tentam diminuir a dimensão do problema e assim transformar o

que consideram um problema intratável num problema de menor dimensão sobre o qual será mais fácil trabalhar. No entanto, ao diminuírem a dimensão do problema, estão a perder algum detalhe da informação e por outro lado estão a aumentar a complexidade matemática do problema.

Outros artigos

Um dos artigos mais completos que encontrei nas minhas pesquisas, em termos de soluções que apresenta, foi o artigo dos autores *Nobibon, Leus e Spieksma* (Nobibon, et al., 2008). Este artigo apresenta cinco heurísticas diferentes para resolver o problema, com algumas abordagens mais alternativas, por exemplo, baseada na marca e no preço do produto. Apesar de algumas destas abordagens não se aplicarem ao negócio da banca, o artigo serviu para perceber que podem existir diferentes visões a abordagens ao problema.

Para além dos artigos já referidos, com soluções concretas e objectivas, investiguei e analisei outros artigos onde pude observar uma abordagem mais conceptual do problema. Por exemplo, no artigo de *Hsu, Tsai, e Chiang* (Hsu, Tsai, & Chiang, 2009), pude analisar as várias questões relacionadas com as dificuldades em medir determinados objectivos de campanhas. Neste artigo é sugerido o uso de sistemas difusos para caracterizar a importância e os objectivos de cada campanha de Marketing.

Existem algumas dificuldades em definir e executar um processo de optimização. Algumas das decisões são efectuadas com base em métricas que são apenas estimativas e por isso têm algumas margens de erro, por exemplo, as taxas de resposta e a definição de custos e proveitos associados a cada oferta.

Existem múltiplas possibilidades de optimização e será sempre difícil encontrar a solução óptima. No entanto, se estivermos conscientes de todos os aspectos importantes e das problemáticas envolvidas, podemos criar vários cenários e construir e aperfeiçoar gradualmente um conjunto de regras que permitem atingir os melhores

resultados e assim justificar com argumentos válidos os investimentos neste tipo de acções.

2.7 Métodos e algoritmos

De um modo geral podemos classificar os métodos de resolução deste tipo de problema em duas categorias principais, por um lado temos os métodos exactos ou determinísticos, que permitem chegar à solução exacta do problema (solução óptima), por outro lado, temos os métodos aproximados, que não permitindo chegar à solução exacta, permitem uma aproximação à solução com resultados igualmente satisfatórios.

Uma forma de encontrar a solução exacta para um problema de programação binária seria por enumeração exaustiva, ou seja, enumerar todas as combinações possíveis e verificar qual a combinação que obtinha a melhor solução satisfazendo as condições impostas. Esta técnica até pode ser usada para pequenos problemas, onde existem poucas combinações, mas para situações de maior escala este processo seria intratável. Para o caso concreto da optimização de campanhas, bastaria termos por exemplo, 10 clientes e 2 campanhas e já teríamos um total de 20 contactos que poderiam combinar entre eles, o que daria um total de 2^{20} combinações possíveis, o que já é um valor bastante elevado. Se escalamos este exemplo para um caso mais real, por exemplo, 2000 clientes e 6 campanhas, obteríamos um número de combinações completamente intratável.

Os métodos exactos fazem esta enumeração de forma explícita ou de forma implícita. Alguns destes métodos fazem a enumeração de forma mais inteligente, procurando eliminar algumas das combinações possíveis e usando uma estratégia de "dividir para conquistar", tornando-se assim mais eficientes. Um destes exemplos é o método *Branch and Bound.* Este método pode ser analisado com detalhe no artigo de Clausen (Clausen, 1999) indicado na bibliografia.

Outros métodos exactos usam uma abordagem de baixo para cima (*bottom-up*), procurando converter o problema inicial em subproblemas de menor dimensão e começando por resolver o problema de menor dimensão para chegar ao problema

acima dele, usando assim um processo recursivo. Este tipo de métodos e algoritmos são conhecidos por programação dinâmica (*dynamic programming*), em que o problema inicial é resolvido por uma serie recursiva de etapas em que cada etapa contribui para a resolução da etapa seguinte (Martello & Toth, 1990).

Os métodos exactos funcionam bem para problemas de pequena e média dimensão, no entanto, para problemas de grande escala estes métodos são pouco eficientes computacionalmente, podendo não obter a solução em tempo útil.

Para problemas de grande escala a solução é então usar um método aproximado para a resolução do problema. Nestes casos a solução obtida não será necessariamente a solução óptima para o problema, mas poderá ser uma solução aproximada com resultados igualmente satisfatórios e obtidos em tempo útil. Estes métodos podem usar variadas técnicas e abordagens, desde abordagens construtivas, de melhoramente, relaxação, etc. Estes métodos aproximados são chamados de *heurísticas (Kim & Moon, 2006).*

Uma das técnicas de resolver este tipo de problemas por um método aproximado é efectuar o que se chama a "relaxação do problema". Neste caso, o problema é simplificado ignorando algumas das restrições ou condições do problema inicial, transformando-o assim num problema mais fácil de resolver. Por exemplo, a relaxação linear considera que os valores das variáveis passam a ser valores reais em vez serem valores inteiros, como inicialmente previsto.

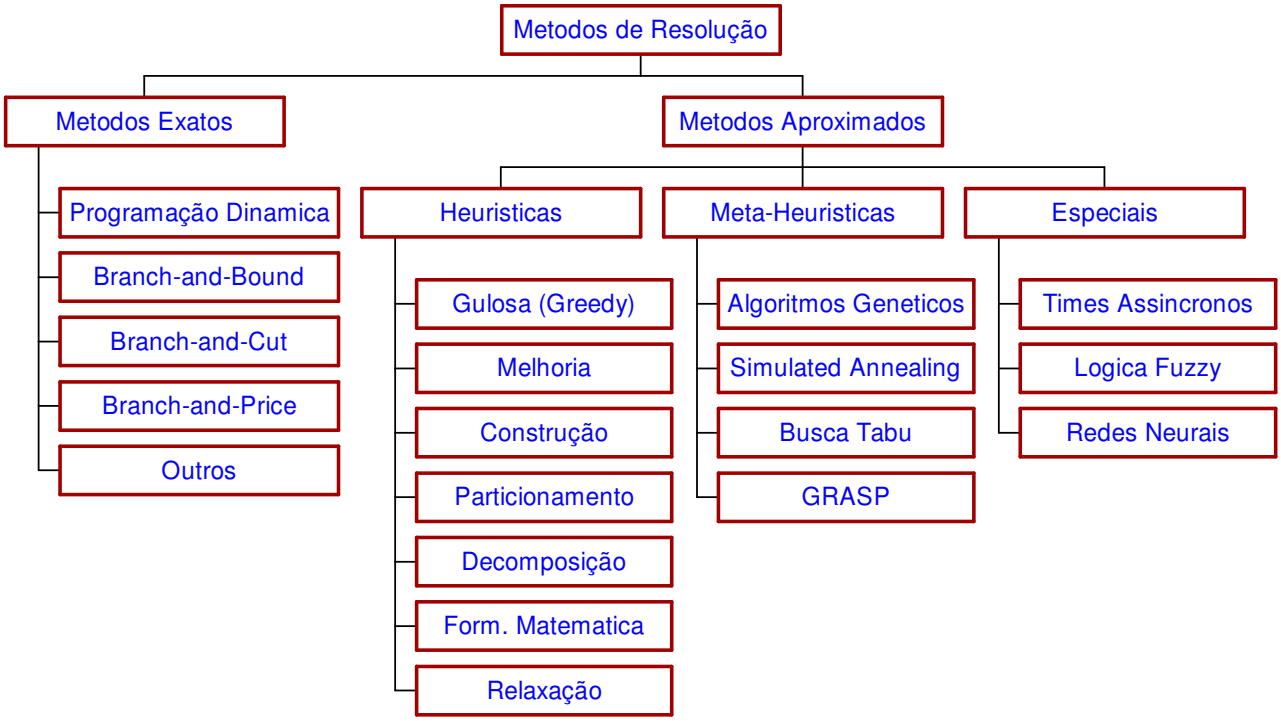
Outra das *heurísticas* bem conhecida e usada para a resolução destes problemas é o chamado algoritmo guloso (*greedy*), usando por exemplo, uma abordagem construtiva. Este algoritmo procura construir uma solução optimizada por um processo de melhoramento iterativo. O algoritmo parte da solução vazia e vai analisar em cada passo do processo a opção mais vantajosa daquela altura, construindo assim uma possível solução, por isso de diz ser uma abordagem construtiva. Esta heurística tem como principal vantagem um tempo de execução muito razoável, quando comparada com outros métodos, sendo um algoritmo muito fácil de implementar.

Um algoritmo guloso pode seguir também uma abordagem destrutiva em vez de uma abordagem construtiva. Na abordagem destrutiva, o algoritmo começa por considerar todos os potenciais candidatos como fazendo parte da solução, de seguida vai excluindo os candidatos com menor impacto na solução óptima, até obter uma lista de candidatos que respeite todas as restrições colocadas, essa é considerada a solução óptima.

Existem ainda outros métodos que podem ser usados para encontrar uma solução aproximada para este tipo de problemas. São algoritmos complexos, usados em *data mining* que se concentram essencialmente na análise dos dados, gerando o conhecimento necessário que lhes permite construir e melhorar uma solução aproximada para o problema. Dois dos mais conhecidos algoritmos deste género são as redes neuronais e os algoritmos genéticos.

A figura 6 procura elencar os algoritmos mais usados para a resolução deste tipo de problemas de optimização, destacando a divisão pelas duas metodologias referidas, os métodos exactos e os métodos aproximados.

Figura 6. Metodos de resolução para problemas de optimização



Fonte: (Rodrigues, 2000)

2.8 Síntese e Conclusão

Este capítulo procurou efectuar uma análise detalhada do problema de optimização de campanhas e da forma como esta questão é colocada e analisada por diferentes autores e investigadores.

O processo de gestão de campanhas descrito nesta tese é o usado para o sector da banca na generalidade, com principal ênfase e orientação para características específicas da instituição onde eu trabalho.

Numa primeira fase foi efectuado o enquadramento teórico do problema, onde foi apresentado e descrito o processo geral de gestão de campanhas e algumas das principais características que determinam a formulação do problema. Nesta fase foi possível perceber e analisar o papel específico do processo de optimização dentro de todo o processo de gestão de campanhas, concluindo-se que se trata de um processo necessário e fundamental para obter melhores resultados com o plano de marketing directo.

Numa segunda fase foi efectuada a análise específica e detalhada do problema, e foi efectuada a respectiva formalização do problema de optimização de campanhas como um problema de programação linear binária da área de investigação operacional.

Nas duas fases seguintes, o problema de optimização de campanhas foi comparado com outros problemas semelhantes mais conhecidos e foi efectuada uma análise e descrição da complexidade deste tipo de problemas.

Foi possível verificar que o problema de optimização de campanhas é semelhante ao problema da múltipla-mochila em muitos aspectos, e tem também algumas semelhanças com o problema geral de afectação. Este tipo de problemas apresenta uma elevada complexidade e na teoria da complexidade computacional são classificados como NP-Difíceis e NP-Completos.

As últimas duas fases deste capítulo foram dedicadas à análise específica de alguns artigos, onde foi investigado este tema, e à descrição das diferentes metodologias e abordagens para a resolução deste tipo de problemas de optimização.

Dos artigos analisados podemos destacar os artigos de *Kim, Yoon e Moon* que são três investigadores que se têm dedicado a este tema com muitos artigos publicados sobre este problema. Estes autores analisam o problema do ponto de vista da sua complexidade e apresentam algumas soluções.

Destaco ainda o artigo de *Choen (Cohen, 2002),* onde o problema de optimização de campanhas é colocado especificamente para o sector da banca. Neste artigo o autor faz uma descrição do processo de gestão de campanhas e do problema de optimização que é muito semelhante à verificada na instituição onde eu trabalho, por isso este artigo serviu para consolidar alguns aspectos no enquadramento teórico do problema.

Em termos de metodologias e abordagens para a resolução deste tipo de problemas, verificámos que existem duas grandes famílias, os métodos exactos e os métodos aproximados. Dentro destas duas grandes abordagens existem inúmeros algoritmos, alguns deles foram descritos de forma resumida.

Neste capítulo de revisão da literatura foi possível perceber que este tema está a ser analisado e investigado por vários autores e que o processo de optimização de campanhas é uma lacuna identificada em grande parte das empresas que aplicam estratégias de marketing directo.

Outra grande conclusão que podemos ainda retirar, é que este problema do ponto de vista da resolução tem uma elevada complexidade computacional, por isso uma das abordagens praticada é encontrar uma solução aproximada para o problema.

Capitulo III – METODOLOGIAS DE INVESTIGAÇÃO

3. Metodologias de Investigação

3.1 Explicação da Metodologias de Investigação

Para concretizar o objectivo, que é o desenvolvimento de um processo de optimização de campanhas, pretendo usar uma metodologia de investigação do tipo *Action Research,* através da análise e implementação do processo numa situação real com a aplicação desta metodologia.

Esta metodologia de investigação é baseada num processo de investigação "acção", ou seja, aprender fazendo, através de uma espiral evolutiva onde se vão analisando os resultados e efectuando acções para obter melhores resultados. Trata-se de um processo colaborativo e democrático onde o conhecimento da organização é integrado para resolver problemas reais e onde a acção é a base do conhecimento (Coghlan & Brannick, 2005).

Existem várias nomenclaturas e diferentes abordagens para a aplicação desta metodologia. Esta espiral de investigação pode, por exemplo, ser definida pelas seguintes etapas (Herr & Anderson, 2005):

1. Elaborar um plano de acção para tentar resolver o problema identificado, de um modo melhor do que existe actualmente.

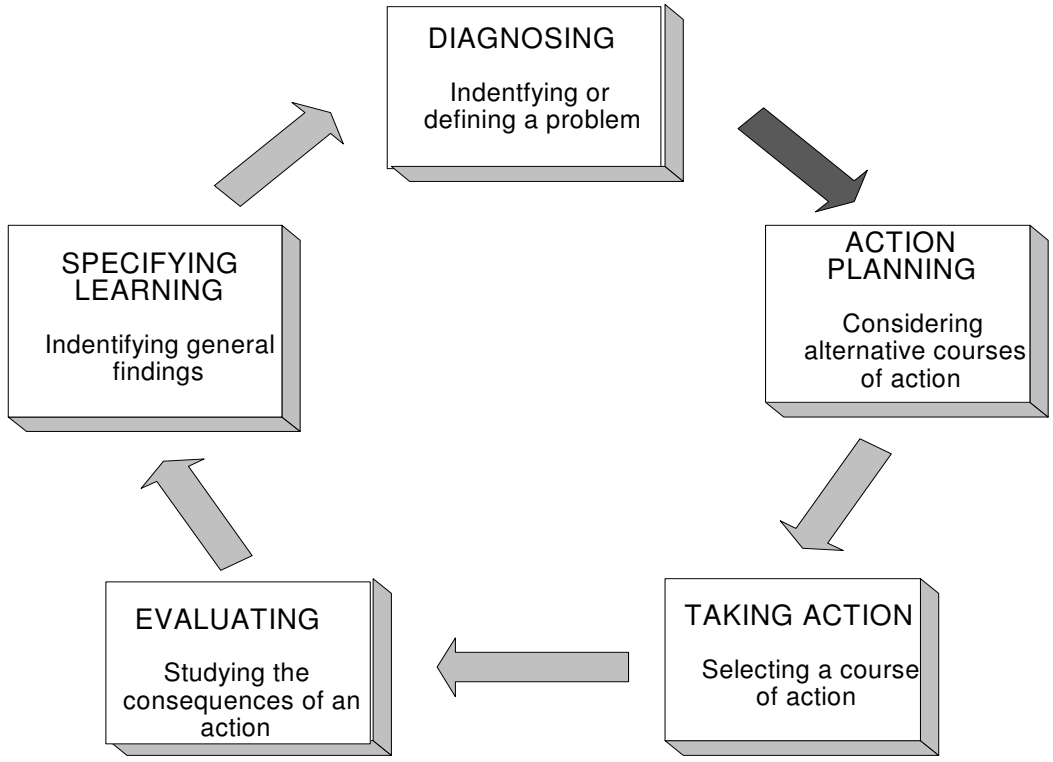
2. Efectuar as acções necessárias para implementar o plano.

3. Observar o resultado das acções tomadas no contexto onde ocorrem.

4. Reflectir e analisar os resultados obtidos no sentido de elaborar um novo plano e efectuar novas acções para melhorar o processo.

Esta metodologia de investigação será um ciclo de aprendizagem onde se vai "aprender fazendo". A figura seguinte ilustra esse ciclo de investigação.

Figura 7. Modelo de "Action Research"



Fonte : (O'Brien, 2001)

Outra forma de definir esta espiral interactiva é encontrada no livro *Action Research* (Stringer, 2007) onde o autor apresenta as fases do que chama a "rotina básica do processo de *Action Research*" (Look-Think-act):

Quadro 1. O processo de *Action Research*

Rotina básica do processo de *Action Research*	
LOOK	Obter os dados e a informação necessária Definir e descrever a situação actual
THINK	Explorar e analisar o que está acontecer Definir e explicar o problema Colocar as questões de investigação
ACT	Planear e reportar as acções a efectuar Implementar essas acções Avaliar e reportar os resultados

As fases apresentadas no quadro anterior são referidas pelo autor como um processo não linear, onde os intervenientes vão explorando as suas actividades e assim construindo um processo constante de observação, reflexão e acção. Em cada final de ciclo os intervenientes devem estar numa etapa superior onde vão voltar a analisar, reflectir a agir.

3.2 Etapas do processo de investigação

A investigação que me proponho efectuar neste trabalho será desenvolvida tendo por base uma situação real e dados reais da instituição bancária onde exerço a minha actividade profissional, aproveitando assim o conhecimento da organização para observar, analisar e implementar esta metodologia.

No seguimento das várias abordagens que os diferentes autores estudados têm sobre a metodologia de *Action Research,* o meu processo de investigação será guiado pelas seguintes etapas:

Etapa 1. Diagnostico e observação do processo actual e garantir o acesso aos dados para a investigação.

Nesta primeira fase vou procurar os vários interlocutores do processo dentro da organização e procurar marcar entrevistas e reuniões para discutir e perceber como funciona o actual processo (Stringer, 2007).

Neste caso, terei de contactar com o responsável da área de marketing directo e dentro da equipa identificar os gestores de campanhas para interagir com eles no sentido de perceber como funciona o actual processo de gestão do plano de marketing do banco. Juntamente com os vários gestores de campanhas terei de observar e discutir o modo como é feito o processo de gestão das campanhas e a metodologia actual de resolver o problema da gestão de recursos e de prioritização das diferentes

ofertas. Nesta fase é importante conversar e discutir com os diferentes participantes no processo para ficar a conhecer como este processo funciona.

Relativamente aos dados para a investigação, já consegui garantir junto do responsável da direcção de marketing do banco o total acesso aos dados, bem como a disponibilidade da equipa de marketing directo para participar no processo de investigação.

Para este processo de investigação vou necessitar de ter acesso a um conjunto de contactos que fazem parte do plano de marketing para um determinado período, e conhecer algumas das regras e politicas praticadas, por exemplo, a capacidade de cada canal, políticas de contacto utilizadas, definição de custos e proveitos das ofertas, etc.

Etapa 2. Explorar e analisar o processo actual e definir correctamente o problema. Planear as acções necessárias para a resolução do problema.

Nesta etapa devo analisar e discutir com os intervenientes do processo todos os aspectos importantes a considerar de modo a construir uma visão correcta da situação actual e dos problemas existentes. Toda a informação recolhida na 1ª etapa deve ser aqui destilada de modo a identificar os conceitos chave (Stringer, 2007). Depois disso devo propor uma ou mais soluções para melhorar a situação e tentar resolver o problema de forma sustentável. A solução proposta deve passar pela construção de um algoritmo para optimizar o plano de marketing. Para isso devo identificar de forma concreta quais são os objectivos a atingir e propor uma forma de atingir esses objectivos. Penso que será fundamental nesta etapa efectuar um planeamento de todas as acções a implementar até chegar à primeira solução do problema, desde a recolha dos dados, até à obtenção dos resultados.

Etapa 3. Implementar as acções necessárias para a resolução do problema.

Nesta etapa deve-se passar dos problemas para as soluções (Stringer, 2007). Nesta fase devo passar para a acção e implementar a solução proposta seguindo o

planeamento da fase anterior. É nesta fase que devo efectuar a codificação e execução do algoritmo proposto para tentar resolver o problema da optimização do plano de marketing e procurar atingir os objectivos definidos.

Etapa 4. Analisar os resultados, observar as falhas e voltar a planear o inicio de um novo ciclo de acções.

Os resultados obtidos serão avaliados e analisados de modo a identificar eventuais problemas a serem corrigidos. A forma de avaliar os resultados terá uma abordagem quantitativa através da medição dos resultados do plano de marketing normal, sem usar nenhum processo de optimização, e confrontar esses resultados com os resultados obtidos através do algoritmo de optimização desenvolvido. Este e os outros objectivos devem ser avaliados e medidos.

A análise e avaliação dos resultados devem servir de base para reiniciar um novo ciclo de investigação onde devo voltar a observar o processo e a planear novas acções para obter uma solução ainda melhor.

3.3 Recursos e Materiais

Os dados serão recolhidos num banco português no qual eu exerço a minhas funções profissionais e ao qual se aplica este *Action Research*.

O trabalho é desenvolvido usando um computador portátil com um processador *intel core duo* de 2.4 GHz e 4 GB de *RAM*, com o sistema operativo Windows Vista.

O Software utilizado para recolha, tratamento e analise de dados será o *Microsoft SQL Server 2005* e o *MATLAB* versão 7.6 (R2008a).

O Software utilizado para tratamento e gestão das referências bibliográficas será o *EndNote X2*.

O software utilizado para a apresentação e formatação deste trabalho é o *Microsoft Office 2007*.

Capitulo IV – IMPLEMENTAÇÃO DE UMA SOLUÇÃO

4. Implementação de uma solução

4.1 Introdução

Neste capítulo vou apresentar o processo de desenvolvimento e implementação de um algoritmo para resolver o problema de optimização de campanhas. No capítulo da revisão da literatura foram apresentadas as questões mais teóricas, nomeadamente o enquadramento teórico e a formalização do problema. Neste capítulo é descrito o processo prático de todo este projecto de investigação, que envolve a análise e o desenvolvimento de um programa informático para a resolução do problema que estou a analisar.

Nas primeiras secções deste capítulo vou apresentar as ferramentas usadas, o processo de obtenção de dados e a construção e definição do modelo de dados. Numa segunda fase, vou apresentar o desenvolvimento do processo distribuído em várias etapas que evidenciam a evolução do processo de acordo com a metodologia de investigação referida no capítulo anterior.

4.2 O processo de desenvolvimento

O primeira situação a analisar no processo de desenvolvimento é a questão da obtenção dos dados, além disso é necessário escolher as ferramentas informáticas para trabalhar os dados e desenvolver o algoritmo ou o programa para a resolução do problema. Depois de termos os dados e as ferramentas necessárias podemos então iniciar o processo de desenvolvimento.

4.2.1 Ferramentas usadas

Normalmente, quando falamos de dados e de informação as ferramentas usadas na generalidade das empresas para efectuar esta gestão estão relacionadas com sistemas de gestão e administração de bases de dados.

Na instituição bancária onde eu trabalho a ferramenta usada no departamento de marketing para gerir os dados que dão suporte às acções de marketing directo, é a aplicação *Microsoft Windows SQL Server 2000*. Por isso a primeira decisão para este trabalho de projecto foi usar esta ferramenta como principal estrutura de suporte e manuseamento de dados. Primeiro, porque será mais fácil para aplicar a solução na instituição onde eu trabalho, e segundo, devido à minha grande experiência com esta ferramenta. No entanto, para este trabalho foi usada uma versão mais recente, o *SQL Server 2005*.

O sistema de base de dados *SQL Server 2005* usa uma linguagem de interrogação a bases de dados com o nome *T-SQL* (*Transact Structured Query Language*), esta foi uma das linguagens de programação usada para desenvolver uma parte do processo, no tratamento e manuseamento dos dados. Além disso, o *SQL Server 2005* disponibiliza uma aplicação com o nome *SQL Server Management Studio*, que também foi usada para o desenvolvimento do programa e para a construção do modelo de dados para suportar a solução.

Para além do *SQL Server 2005*, foi usada ainda a ferramenta *Matlab* na versão 7.6 (R2008a). O *Matlab* tem associada uma linguagem de programação que também foi usada para implementar uma parte do processo. Foram ainda usadas as rotinas *TOMLAB/CPLEX* versão 7.3, que são um conjunto de programas para *Matlab* para a resolução de problemas de programação linear e quadrática.

4.2.2 Modelo de dados e obtenção dos dados

Para desenvolver o algoritmo para este trabalho foi necessário efectuar a obtenção de dados. A primeira etapa desse processo foi definir uma estrutura que permitisse alojar e manusear esses mesmos dados. Essa estrutura numa linguagem de bases de dados é chamada de modelo de dados.

Tendo em conta que a principal ferramenta usada neste trabalho para a gestão e suporte dos dados é um sistema geral de base de dados, vou usar ao longo desta tese uma terminologia associada a este tipo de aplicações. Por isso serão usados com

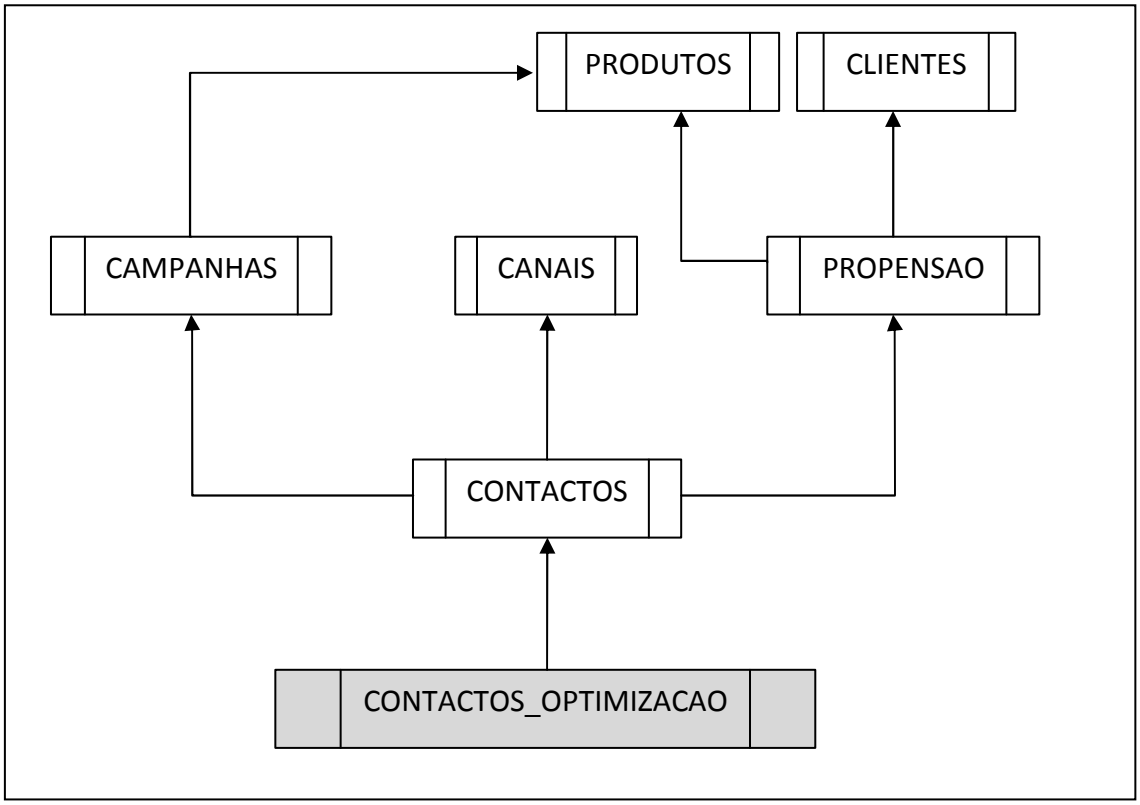
frequência os seguintes termos: registo ou linha, campo ou coluna, tabela, modelo de dados, etc.

A obtenção dos dados foi feita com base em dados reais existentes na instituição bancária onde eu trabalho, onde obtive permissão para analisar um conjunto de cerca de dois milhões de contactos criados para seis campanhas. Com base nesses dados foram analisadas as regras existentes em cada variável para depois criar um programa de geração de cenários. Este programa permitiu gerar vários conjuntos de dados, de diferentes dimensões para testar e avaliar os resultados do algoritmo desenvolvido.

Como foi referido no capítulo da revisão da literatura, um contacto é um registo definido pelos campos: campanha, cliente e canal. Estes campos formam a chave que identifica univocamente cada contacto. Além disso, para cada contacto deve ser indicado o custo do contacto e o ganho associado ao sucesso desse contacto.

Deste modo, a figura 8 mostra um diagrama de entidades e relações com um primeiro modelo (ainda resumido) criado para o tratamento dos dados necessários para este trabalho. Em anexo pode ser consultado o modelo de dados completo e final.

Figura 8. Modelo de dados resumido para o problema de optimização de campanhas.



Descrição do Modelo de dados

O primeiro modelo de dados contém apenas as tabelas necessárias para a obtenção de dados, à excepção da tabela CONTACTOS_OPTIMIZACAO que já foi criada para os futuros contactos seleccionados pelo processo de optimização.

Neste modelo de dados existe uma tabela principal que contem todos os contactos inicialmente seleccionados para todas as campanhas, é a tabela CONTACTOS. Existem ainda várias tabelas adicionais com os dados dos clientes, campanhas, canais, produtos e propensão.

Quadro 2. Lista de tabelas do modelo de dados inicial

TABELA	DESCRIÇÃO
CLIENTES	Tabela com os dados relativos a cada cliente
PRODUTOS	Tabela com os dados relativos aos produtos
PROPENSAO	Tabela com a propensão de cada cliente para cada produto
CAMPANHAS	Tabela com os dados relativos às campanhas
CANAIS	Tabelas com os dados relativos aos canais
CONTACTOS	Tabela com todos os potenciais contacto à partida
CONTACTOS_OPTIMIZACAO	Tabela final com os contactos seleccionados pelo processo de optimização

Um primeiro conjunto de dados foi criado com base em dados reais que me foram facultados pela instituição bancária onde eu trabalho. A partir da análise deste primeiro conjunto de dados foi possível perceber as relações e as características dos dados e identificar um conjunto de regras que permitiram gerar novos conjuntos de dados. Esta tarefa foi muito simplificada pela minha experiência profissional em trabalhar com este tipo de dados e pela facilidade que tive em questionar o responsável pela área de gestão de marketing directo da instituição onde trabalho.

A partir das regras identificadas foi desenvolvido um programa que recebe como parâmetros de entrada o número de campanhas e o número de clientes, e gera de forma automática um novo modelo de dados, criando uma tabela de potenciais contactos. Deste modo é possível criar vários cenários, gerando diferentes conjuntos

de dados com diferentes características e dimensões, que permitem testar o algoritmo desenvolvido.

Para simplificar o processo, e como já foi referido no capítulo da revisão de literatura, neste trabalho estou a considerar que cada campanha tem apenas um produto, logo a tabela de produtos pode ser eliminada do modelo e usada apenas a tabela de campanhas, partindo do princípio que cada campanha terá sempre um produto associado (um e apenas um).

Como também já foi referido no capítulo da revisão da literatura, por uma questão de dimensão dos dados, vamos assumir que cada canal de contacto é definido à partida. Sendo assim, para cada par {campanha, cliente} existe apenas um canal associado. Os canais considerados neste modelo de dados são: *Mail, E-Mail*, Balcão, Telemarketing e *SMS*. Estes cinco canais são transversais aos vários cenários de modelos de dados gerados, ou seja, em todos os cenários são considerados estes cinco canais.

Cada um dos modelos de dados foi criado com um prefixo diferente em todas as tabelas, deste modo podem coexistir vários modelos na mesma base de dados. Por exemplo, o modelo de dados TESTE01 foi criado com duas campanhas e dez clientes, gerando um conjunto de vinte potenciais contactos distribuídos por cinco canais.

Deste modo, para o modelo de dados TESTE01, foram criadas as seguintes tabelas:

TESTE01_CANAIS

Esta tabela é fixa em todos os modelos gerados e tem sempre os cinco canais já apresentados. Para cada canal é definido o número de contactos máximos a efectuar no período, este valor é gerado em cada modelo e será um dos valores a considerar nas restrições do problema de optimização.

Canal_ID	Canal_Nome	N_Maximo
100	Mail	500
200	E-Mail	1000
300	Telemarketing	250
400	Balcão	50
500	SMS	2000

TESTE01_CAMPANHAS

Esta tabela tem dois registos que identificam as duas campanhas do modelo. Como cada campanha tem um, e apenas um produto, nesta tabela é também definido o produto da campanha (eliminando assim a tabela de produtos do modelo). Para cada campanha é definido um proveito e um valor máximo de orçamento para gastar. O valor do proveito será usado no cálculo do ganho de cada contacto, o valor máximo do orçamento será um dos valores a considerar nas restrições do problema de optimização.

Campanha_ID	Campanha_Nome	Valor_Proveito	Valor_Orcamento
CA1	Campanha para o produto 1	43,8	40
CA2	Campanha para o produto 2	140	97,4

TESTE01_CLIENTES

Esta tabela tem dez registos que identificam os clientes existentes neste modelo de dados.

Cliente_ID	Cliente_Nome
CLI01	Cliente 1
CLI02	Cliente 2
CLI03	Cliente 3
CLI04	Cliente 4
CLI05	Cliente 5
CLI06	Cliente 6
CLI07	Cliente 7
CLI08	Cliente 8
CLI09	Cliente 9
CLI10	Cliente 10

TESTE01_PROPENSAO

Esta tabela tem o cruzamento de todos os clientes com todos os produtos (neste caso campanhas) existentes no modelo. Neste modelo, esta tabela tem vinte registos (2 campanhas x 10 clientes) que definem a taxa de propensão que cada cliente tem para adquirir determinado produto. As taxas de propensão são valores reais entre 0,5 e 1,

uma vez que apenas são considerados clientes/produtos com taxas superiores ou iguais a 50%.

Cliente_ID	Campanha_ID	Valor_Propensao
CL01	CA1	0.5
CL01	CA2	0.67
CL02	CA1	0.89
CL02	CA2	0.87
CL03	CA1	0.74
CL03	CA2	0.58
CL04	CA1	0.65
CL04	CA2	0.80
CL05	CA1	0.87
CL05	CA2	0.74
CL06	CA1	0.94
CL06	CA2	0.59
CL07	CA1	0.67
CL07	CA2	0.89
CL08	CA1	0.87
CL08	CA2	0.74
CL09	CA1	0.93
CL09	CA2	0.91
CL10	CA1	0.83
CL10	CA2	0.68

TESTE01_CONTACTOS

Esta tabela tem todos os potenciais contactos a efectuar. Cada contacto é formado por uma campanha, um cliente, um canal, e os valores de custo e ganho do contacto. Os contactos são gerados pelo cruzamento de todos os clientes com todos os produtos (campanhas), calculando o valor de custo com base no canal escolhido. O valor do ganho é calculado com base no proveito de cada produto (campanha) e na taxa de propensão definida na tabela de propensão. O cálculo destes valores já foi apresentado com mais detalhe no capítulo de revisão da literatura.

Neste caso, a tabela de contactos tem vinte potenciais contactos.

Contacto_ID	Cliente_ID	Campanha_ID	Canal_ID	Valor_Custo	Valor_Ganho
1	CL01	CA1	500	8,02	51,30
2	CL01	CA2	200	8,15	26,51
3	CL02	CA1	400	4,23	46,96
4	CL02	CA2	200	8,15	27,12
5	CL03	CA1	400	4,23	48,83
6	CL03	CA2	300	10,25	25,64
7	CL04	CA1	400	4,23	50,71
8	CL04	CA2	400	8,42	28,08
9	CL05	CA1	400	4,23	52,58
10	CL05	CA2	300	10,25	27,47
11	CL06	CA1	300	13,71	53,73
12	CL06	CA2	300	10,25	28,08
13	CL07	CA1	200	16,11	52,27
14	CL07	CA2	300	10,25	28,69
15	CL08	CA1	300	13,71	55,61
16	CL08	CA2	400	18,42	31,14
17	CL09	CA1	400	24,23	46,96
18	CL09	CA2	100	20,40	30,38
19	CL10	CA1	500	18,02	55,04
20	CL10	CA2	300	20,25	32,36

4.2.3 Uma primeira abordagem com *MATLAB*

Tendo por base alguns dos artigos referidos na revisão da literatura, nomeadamente o artigo dos autores *Nobibon, Leus* e *Spieksma* (Nobibon, et al., 2008), a primeira abordagem para a resolução do problema de optimização de campanhas foi uma tentativa de usar a ferramenta *MATLAB* e algumas das rotinas orientadas especificamente para este tipo de problemas.

Assim, depois de algumas pesquisas que efectuei na Internet consegui obter uma licença para estudantes e limitada das rotinas *TOMLAB* para o *MATLAB*, que contêm o *solver* (termo em Inglês para designar as rotinas que resolvem este tipo de problemas) *CPLEX* para resolução de problemas de programação linear.

As rotinas *TOMLAB* são uma plataforma de desenvolvimento modelar aplicada ao *MATLAB* para a resolução de problemas de optimização. O *CPLEX* é uma das rotinas

disponibilizada nesta plataforma que permite a resolução de problemas de grande escala de programação linear e quadrática (*Tomlab*).

Para o caso específico de problemas de programação linear binária, o *TOMLAB* contem a rotina *BINTPROG*. Esta rotina é equivalente à rotina com o mesmo nome que também existe na *Toolbox* de optimização do *MATLAB*, mas que neste caso permite um número maior de variáveis. A rotina original do *MATLAB* deu erro de "*Out of Memory*" a partir das 3.000 variáveis, no entanto, a rotina do *TOMLAB* só a partir das 6.000 variáveis deu o mesmo erro. Além disso, a rotina do *TOMLAB* apresentou tempos de resposta cerca de 10 vezes mais rápidos.

A rotina *BINTPROG* do *TOMLAB* é uma rotina específica, que usa uma rotina mais geral, o *CPLEX,* para resolver problemas de optimização linear inteira binária.

Esta rotina permite a resolução de problemas do tipo:

Minimizar $f^T \times x$

Sejeito a $A \times x \leq b$

$Aeq \times x = beq$

Em que $x \in \{0,1\}$

A rotina *BINTPROG* tem os seguintes parâmetros de entrada:

f – Vector com os coeficientes da função objectivo

A – Matriz composta pelos coeficientes das inequações que representam as restrições

b – Vector com os valores da direita das inequações

Aeq - Matriz composta pelos coeficientes das equações que representam as restrições

beq – Vector com os valores da direita das equações

A rotina vai devolver os seguintes parâmetros de saída para as seguintes variáveis:

x - Matriz de valores zero e um, com o resultado do processo de optimização.

fval – Valor óptimo para a função objectivo. No caso do problema de optimização de campanhas é o máximo de ganho que se pode obter com o processo de optimização.

ExitFlag – Indicador com o resultado do algoritmo, pode ter os valores:

1, se o algoritmo converge para a solução x.

0, se o número máximo de iterações foi excedido.

-2, se o problema não tem solução.

-4, se foi atingido o número máximo de nós e o algoritmo não converge para uma solução.

-5, se foi atingido o tempo máximo (*MaxTime*), sem que o algoritmo convirja para uma solução.

-6, se foi atingido o número máximo de iterações num nó para resolver o problema relaxado de programação linear (*MaxRLPIter*), sem que o algoritmo convirja para uma solução.

De forma a poder utilizar a rotina *BINTPROG* do *TOMLAB* para a resolução do problema de optimização de campanhas, foi necessário criar um processo para transformar os dados alojados na base de dados do *SQL Server,* em variáveis no *MATLAB*, que serviriam de parâmetros de entrada para a rotina. Além disso, foi necessário criar numa sequência de código em *MATLAB,* para que a rotina *BINTPROG* fosse invocada com os parâmetros gerados a partir do modelo de dados e que permitisse depois visualizar os resultados obtidos.

Como já foi referido, foi criado um primeiro programa para gerar cenários de forma automática e preencher as tabelas do modelo de dados de acordo com um número de campanhas e de clientes pretendidos. Neste momento, era necessário criar um outro

programa que, com base num desses cenários gerados, e a partir do modelo de dados em *SQL Server*, criasse um ficheiro do tipo *MATLAB*, com extensão ".m", que permitisse invocar a rotina do *TOMLAB* para a resolução do problema e posterior visualização dos resultados obtidos.

Para usarmos a rotina *BINTPROG,* existe ainda um problema, esta rotina é para casos em que o objectivo é minimizar uma função. No entanto, para o problema de optimização de campanhas o objectivo é maximizar uma função (neste caso é o total de ganho). Para contornar esta situação e transformar o problema de maximização num problema de minimização, é necessário inverter os coeficientes da função multiplicando-os por -1. Sendo assim, o problema de optimização de campanhas deverá ser colocado do seguinte modo:

Minimizar $(-1) \times f^T \times x$

Sejeito a $A \times x \leq b$

$Aeq \times x = beq$

Em que $x \in \{0,1\}$

O passo a seguir para a resolução do problema foi definir os parâmetros de entrada para a rotina. Para isso foi necessário usar a formalização do problema já apresentada no capítulo da revisão da literatura, e com base nessa formalização construir as variáveis em *MATLAB* que serviram de parâmetros de entrada.

Primeiro parâmetro: vector f

Na formalização do problema podemos verificar que a função objectivo é dada pela seguinte expressão:

Maximizar $\sum_{i=1}^{m} \sum_{\mathrm{j}=1}^{\mathrm{n}} \sum_{\mathrm{k}=1}^{\mathrm{p}} g_{ijk} x_{ijk}$

Desta forma, os coeficientes da função são dados pelo valor de ganho de cada contacto (g_{ijk}), formando assim um vector com m x n x p posições. Este vector deverá ser o valor para a variável f que servirá de parâmetro de entrada. No entanto, como já referido anteriormente, vamos assumir que o canal para cada contacto é único. Desta forma alguns dos coeficientes desta expressão serão zero, o que faz com que a dimensão p, relacionada com o canal, seja na realidade igual a um. Deste modo o vector f não irá ter m x n x p posições, mas sim apenas m x n.

Segundo parâmetro: matriz A

Continuando a analisar a formalização do problema de optimização de campanhas, podemos verificar que as restrições colocadas sobre a forma de inequações são dadas pelas seguintes expressões:

1. $\sum_{j=1}^{n}\sum_{k=1}^{p} g_{ijk}x_{ijk} \leq b_j$ i ε M = {1 … m}
2. $\sum_{i=1}^{m}\sum_{j=1}^{n} x_{ijk} \leq c_k$ k ε P = {1 … p}
3. $\sum_{j=1}^{n}\sum_{k=1}^{p} x_{ijk} \leq r_i$ i ε M = {1 … m}
4. $\sum_{j=1}^{n}\sum_{k=1}^{p} -x_{ijk} \leq s_i$ i ε M = {1 … m}
5. $\sum_{i=1}^{m}\sum_{k=1}^{p} x_{ijk} \leq 1$ j ε N = {1 … n}

$x_{ijk} \in \{0,1\}$

Neste caso, a restrição 4 foi invertida multiplicando a variável binária x_{ijk} por -1 de modo a que o sinal ficasse igual às restantes restrições.

Este conjunto de expressões será usado para a definição da matriz A e do vector b que serão as variáveis usadas como parâmetros de entrada da rotina *BINTPROG*.

A matriz A será formada com os coeficientes do lado esquerdo do sinal ≤ e será uma matriz com um número de colunas igual ao número de contactos existentes à partida. O número de linhas será equivalente ao somatório de todas as linhas necessárias para definir cada uma das restrições.

O número de linhas da matriz A será definido da seguinte forma:

Para cada restrição será necessário um número de linhas de acordo com a dimensão a que diz respeito essa restrição. Por exemplo, se a restrição é ao nível da campanha, será necessária uma linha para cada campanha.

A restrição 3 não será considerada, porque vamos partir do princípio que a forma mais normal de restringir o número de contactos em cada campanha será restringir o orçamento disponível para a campanha, não sendo necessária esta restrição. Em todo o caso será fácil acrescentar esta restrição uma vez que é muito idêntica à restrição 4, para garantir o número mínimo de contactos por campanha.

Deste modo, voltando a analisar a formalização do problema, verificamos que temos quatro restrições. Duas são ao nível das campanhas, uma ao nível dos canais e outra ao nível dos clientes. Sendo assim, o número de linhas da matriz *A* será dado pela expressão:

$$(m \times 2) + (n \times 1) + (p \times 1)$$

Terceiro parâmetro: vector b

O vector b será formado pelos valores do lado direito do sinal ≤ e será um vector com o número de posições igual ao número de linhas da matriz *A*. O modo de definir este vector será análogo ao modo descrito anteriormente para definir as linhas da matriz *A*. Neste caso os valores a usar serão os do lado direito das inequações.

No caso do problema de optimização de campanhas, não temos restrições que sejam formalizadas sobre a forma de equações. Deste modo a matriz *Aeq* e o vector *beq* serão nulos e assim não serão considerados nos parâmetros de entrada para a rotina *BINTPROG*.

Exemplo

Para melhor compreender este processo de definição dos parâmetros para a rotina *BINTPROG*, vamos voltar a usar o exemplo já apresentado na formalização do

problema. Recordo que este exemplo de tamanho muito reduzido é composto por duas campanhas, dois clientes e dois canais.

Para este exemplo, existem quatro restrições que são as seguintes:

1. A campanha m1 tem um orçamento limitado a 15€
2. O canal k1 só tem capacidade para efectuar 1 contacto
3. A campanha m2 tem de ter pelo menos um contacto
4. Cada cliente só pode ser contactado uma vez

O exemplo apresentado é formalizado da seguinte forma:

Maximizar $f = 90.5\ x_{111} + 50.6\ x_{121} + 60\ x_{212} + 50\ x_{222}$

Sujeito a $12.4\ x_{111} + 10\ x_{121} \leq 15$

$x_{111} + x_{121} \leq 1$

$-x_{212} - x_{222} \leq 0 \quad (x_{212} + x_{222} > 0)$

$x_{111} + x_{212} \leq 1$

$x_{121} + x_{222} \leq 1$

$x_{111},\ x_{122},\ x_{211}$ e $x_{222}\ \varepsilon\ \{0,1\}$

Para este exemplo, o modelo de dados teria quatro contactos definidos da seguinte forma:

Tabela de CONTACTOS

Contacto_ID	Cliente_ID	Campanha_ID	Canal_ID	Valor_Custo	Valor_Ganho
1	c1	m1	k1	12,4	90,5
2	c2	m1	k1	10	50,6
3	c1	m2	k2	14,6	60
4	c2	m2	k2	11,2	50

Tabela de CANAIS

Canal_ID	Canal_Nome	N_Maximo
k1	Mail	1
k2	E-Mail	Null

Para este pequeno exemplo com apenas quatro contactos, as variáveis com os parâmetros de entrada para a rotina *BINTPROG* seriam definidas do seguinte modo:

1. O primeiro parâmetro é formado pelo vector de coeficientes com os ganhos de cada contacto, ordenados pelo identificador da campanha e depois pelo identificador do cliente. Neste caso teríamos:

   f = (90.5, 50.6, 60, 50)

2. O segundo parâmetro será a matriz *A* em que o número de colunas da matriz é igual ao número de contactos, neste caso quatro. O número de linhas depende das restrições do problema.

A matriz *A* é construída da seguinte forma:

No primeiro passo, deve ser criada uma matriz de correspondência entre a lista de contactos ordenada pela mesma ordem usada na construção do primeiro parâmetro f e as diferentes dimensões que serão usadas nas várias restrições do problema.

Neste caso, a matriz de correspondência deve ficar da seguinte forma:

f	90.5	50.6	60	50
OFERTA_ID	1	2	3	4
CAMPANHA_ID	m1	m1	m2	m2
CLIENTE_ID	c1	c2	c1	c2
CANAL_ID	k1	k1	k2	k2

Matriz de correspondência

Com base nesta matriz podemos observar onde se enquadra cada uma das dimensões. Cada dimensão corresponde a uma linha, apenas temos de verificar em que colunas estão os códigos para os quais pretendemos criar a linha da matriz *A*. Por exemplo, a

dimensão “cliente” está na linha 2, para o cliente 1 temos as colunas 1 e 3, para o cliente 2 temos as colunas 2 e 4.

No segundo passo, para cada uma das dimensões, vamos usar a matriz anterior para criar uma linha da matriz por cada restrição que queremos considerar.

Neste caso, a matriz A pode ser reapresentada do seguinte modo:

F	90.5	60	50.6	50
OFERTA_ID	1	2	3	4
RESTRIÇÃO 1	12.4	10	0	0
RESTRIÇÃO 2	1	1	0	0
RESTRIÇÃO 3	0	0	-1	-1
RESTRIÇÃO 4	1	0	1	0
RESTRIÇÃO 5	0	1	0	1

Por exemplo, a primeira restrição é ao nível da campanha, esta restrição indica que a campanha 1 tem um orçamento limitado até 15€. Como podemos observar na matriz de correspondência, os contactos da campanha 1 estão nas colunas 1 e 2 da matriz, logo os custos associados a esses contactos devem ser colocados nessas colunas deixando as restantes colunas com o valor zero. Desta forma criamos a primeira linha da matriz A que corresponde ao vector [12.4 10 0 0]. Para a segunda restrição temos o vector [1 1 0 0], e assim sucessivamente.

Para este exemplo a matriz *A* seria então a seguinte:

A =

12.4	10	0	0
1	1	0	0
0	0	-1	-1
1	0	1	0
0	1	0	1

O terceiro parâmetro é o vector *b* formado pelos valores máximos considerados nas restrições. A ordem destes valores no vector deve ser a mesma que foi considerada na construção das linhas da matriz *A*.

Neste caso o vector *b* seria o seguinte *b* = [15 1 0 1 1].

A formalização apresentada anteriormente para este pequeno problema pode então ser definida da seguinte forma:

Minimizar $(-1) * f^T * x$

-90.5	-50.6	-60	-50	X	X_{111}
					X_{121}
					X_{211}
					X_{222}

Sujeito a $A * x \leq b$

12.4	10	0	0	x	X_{111}	≤	15
1	1	0	0		X_{121}		1
0	0	-1	-1		X_{211}		0
1	0	1	0		X_{222}		1
0	1	0	1				1

Depois de definir os parâmetros de entrada necessários para a rotina *BINTPROG*, o próximo passo foi invocar esta rotina para a resolução do problema e depois visualizar os resultados. Numa primeira fase, a visualização dos resultados foi feita com base na instrução *fprintf* do *MATLAB* para mostrar o conteúdo das variáveis *fval e ExitFlag,* que são parâmetros de saída da rotina.

Numa fase posterior, foi acrescentado código ao programa para exportar o conteúdo da variável *x (*que é outro parâmetro de saída da rotina), para uma tabela de *SQL Server*. Desta forma passou a ser possível identificar quais foram os contactos seleccionados pela rotina de optimização.

Nesta altura do projecto já tinham sido desenvolvidos dois programas. Um para criar cenários e gerar automaticamente vários modelos de dados para diferentes problemas. E outro que criava de forma automática um ficheiro de *MATLAB*, de extensão ".m", com base num dos modelos de dados gerado pelo programa anterior. Estes dois programas permitiam resolver para cada cenário, o respectivo problema de programação binária com base na rotina *BINTPROG do TOMLAB/CPLEX*.

O passo a seguir foi começar a testar os programas com diferentes cenários para depois analisar os resultados. Para isso foram gerados vários cenários de testes. Como referido anteriormente, os modelos de dados gerados para cada cenário foram criados com um prefixo em todas as tabelas. Deste modo é possível identificar o modelo de dados de cada cenário. Por exemplo, todas as tabelas do modelo de dados para o cenário 1 foram criadas com o prefixo TESTE01, para o cenário 2 com o prefixo TESTE02, e assim sucessivamente.

O primeiro cenário criado foi o TESTE01 com apenas dez clientes e duas campanhas, formando um total de vinte contactos. Neste caso temos vinte variáveis binárias. A rotina do *TOMLAB/CPLEX* teve uma resposta bastante rápida (2 Segundos) e o valor de ganho obtido foi de 255,39€.

A seguir foram gerados novos cenários fazendo aumentar o número de clientes e de campanhas, e com isso aumentando o número de contactos ou variáveis. O número máximo de variáveis para o qual foi possível obter resposta foi de 6.000 variáveis. Com mais de 6.000 variáveis a rotina *BINTPROG* começou a dar o erro de "*Out of Memory*". O cenário TESTE07 com mil clientes e seis campanhas, formando assim um total de seis mil variáveis, foi o modelo de dados com mais variáveis que consegui usar com esta rotina. No cenário TESTE08 com sete mil variáveis, o programa já deu o erro de "*Out of Memory*".

O próximo quadro apresenta a lista de cenários gerados para testar este processo.

Quadro 3. Lista de cenários gerados para testar a rotina *BINTPROG.*

Cenário	Nº Clientes	Nº Campanhas	Nº Canais	Nº Variáveis	Tempo (Seg.)
TESTE01	10	2	5	20	2 seg.
TESTE02	100	3	5	300	5 seg.
TESTE03	150	4	5	600	12 seg.
TESTE04	200	6	5	1.200	15 seg.
TESTE05	500	6	5	3.000	40 seg.
TESTE06	1.000	4	5	4.000	75 seg.
TESTE07	1.000	6	5	6.000	150 seg.
TESTE08	1.000	7	5	7.000	*Out of Memory*

Depois destes testes ficou bastante claro que esta rotina não podia ser usada para o caso real do problema de optimização de campanhas. Mesmo com uma versão de 64 bits do *MATLAB* e com um computador com mais memória, no caso real poderíamos ter que trabalhar com milhões de variáveis e esta rotina não poderia ser usada para essa dimensão. Sendo assim, era necessário criar um novo processo que permitisse chegar a uma solução do problema. Mesmo que fosse apenas uma solução aproximada da solução óptima, mas que funcionasse com problemas de grande dimensão.

4.2.4 Desenvolvimento do algoritmo

Escolha do método de resolução para o algoritmo

Como foi apresentado no capítulo de revisão da literatura, existem várias metodologias e abordagens para a resolução de problemas de optimização. Como também foi referido, um dos métodos mais usado para a resolução de problemas de grande escala é o algoritmo guloso (*greedy).* Este método é uma heurística que usa, por exemplo, uma abordagem construtiva, e que analisa iterativamente a melhor opção em cada passo do processo.

O algoritmo guloso foi o método escolhido para a resolução do problema de optimização de campanhas. Esta escolha é justificada por duas situações: primeiro, por ser um algoritmo fácil de explicar e de implementar, por isso seria fácil de implementar na instituição onde eu trabalho. Segundo, por ser um método bastante eficiente, mesmo para problemas de grande escala. Este algoritmo apenas necessita de percorrer uma vez (no máximo) todos os contactos. Mesmo para uma situação de alguns milhões de contactos, isto será possível com um tempo relativamente reduzido.

Num ambiente real, e dada a volatilidade do plano de marketing, este algoritmo pode ser necessário correr com muita frequência. Sempre que existam alterações ao plano inicialmente previsto, são geradas novas oportunidades de contacto que é necessário integrar no plano de marketing a optimizar. Por isso é fundamental que o algoritmo possa correr de forma célere, num período de tempo relativamente curto.

Descrição do algoritmo

Vamos começar por analisar algumas questões relacionadas com este tipo de algoritmos. Neste caso vamos considerar um algoritmo guloso com uma abordagem construtiva, apesar das mesmas questões se colocarem numa abordagem destrutiva.

A primeira etapa de um algoritmo guloso para este tipo de problemas, é ordenar a lista de potenciais contactos. Esta ordenação deve permitir escolher o melhor contacto em cada passo do processo, quando se percorre essa lista a partir do topo e de forma sequêncial. Esta tarefa seria bastante facilitada se as decisões a tomar fossem fraccionais, ou seja, se fosse possível escolher apenas uma fracção do contacto, por exemplo, efectuar apenas 1/3 do contacto i. Mas sabemos que neste caso apenas podemos escolher duas opções, ou efectuar o contacto ($x_i = 1$) ou não efectuar o contacto ($x_i = 0$), por isso estamos perante um problema de optimização binária.

A questão fundamental para o sucesso do algoritmo guloso é encontrar um critério de ordenação que permita ordenar os contactos da forma ideal, tendo em conta as restrições existentes. A escolha do critério de ordenação vai ser responsável por identificar em cada passo, o próximo contacto a considerar para uma possível selecção. É claro que o contacto só será seleccionado se não colocar em causa nenhuma das restrições consideradas, mas a ordem pela qual vai ser posto à prova é um aspecto crucial.

E qual será o critério de ordenação mais correcto?

Temos à partida três critérios candidatos mais evidentes:

1. O critério mais intuitivo e evidente seria ordenar a lista de contactos por ordem de ganhos decrescentes, ou seja, do contacto com um maior ganho para o contacto com menor ganho.

2. Outro critério lógico seria ordenar a lista por ordem de custos crescentes, ou seja, do contacto com um menor custo para o contacto com maior custo.

3. Outro critério, talvez o mais correcto, seria relativizar o ganho com o custo e ordenar por esse rácio de forma decrescente, ou seja, do contacto com um maior rácio ganho/custo para o contacto com esse menor rácio.

De forma a analisarmos estas três opções, vamos ver um exemplo concreto de dimensão reduzida para melhor perceber o problema.

Suponhamos que temos apenas a campanha C1, com um orçamento máximo previsto de 100 euros, que será a única restrição a considerar. E suponhamos que temos uma lista de seis potenciais contactos para optimizar.

O seguinte quadro mostra os ganhos e custos de cada contacto e a solução obtida com os três critérios de ordenação referidos anteriormente.

Quadro 4. Algoritmo guloso com três critérios diferentes de ordenação

Lista de seis contactos				**Critérios de Ordenação**			
Contacto	Ganho	Custo	Ganho/Custo	Ganho	Custo	Ganho/Custo	Solução Óptima
1	400	100	4	1	0	0	0
2	350	50	7	0	0	1	1
3	180	45	4	0	1	0	1
4	40	20	2	0	1	1	0
5	100	10	10	0	1	1	0
6	20	5	4	0	1	1	1
Total Custos				100	80	85	100
Total Ganho				400	340	510	550
Campanha C1 com um orçamento máximo de 100 euros							

Como podemos verificar na análise do quadro, nenhum dos três critérios usados permite chegar à solução óptima.

Vamos analisar as três soluções e a solução óptima com mais detalhe:

i) No caso do critério de ordenação usado ser o valor do ganho, o contacto com maior ganho é o contacto 1. Neste caso, como o custo desse contacto é de 100 euros, esgotamos logo o orçamento da campanha e não podemos

seleccionar mais nenhum contacto. Com esta solução obtemos um ganho de apenas 400 euros, que é o valor de ganho do contacto 1.

ii) No caso do critério de ordenação usado ser o valor do custo, vamos poder seleccionar os contactos 3, 4, 5, e 6 até esgotar o valor do orçamento de 100 euros. Neste caso, o valor ganho com estes contactos seria de apenas 340 euros, inferior ao ganho obtido com a solução anterior ordenada pelo valor do ganho.

iii) No caso do critério de ordenação usado ser o rácio entre o valor do ganho e o custo, vamos poder seleccionar os contactos 2, 4, 5, e 6 até esgotar o valor do orçamento de 100 euros. Neste caso o valor ganho com estes contactos já seria de 510 euros, superior ao ganho obtido com as soluções anteriores ordenadas pelo valor do ganho e do custo respectivamente.

Para este caso, a solução óptima seria seleccionar os contactos 2, 3, e 6. Esta solução permitiria aproveitar na totalidade o valor do orçamento de 100 euros, e obtinha um ganho no total de 550 euros com os três contactos. Este valor é superior a qualquer dos três valores conseguidos anteriormente.

Da análise deste pequeno exemplo, podemos concluir que nenhum dos três critérios de ordenação referidos anteriormente pode chegar à solução óptima. Neste caso, o critério que ficou mais próximo da solução óptima foi o critério de ordenação pelo rácio entre o ganho e o custo. Mas como veremos mais à frente neste trabalho, na maioria dos testes efectuados não foi assim.

Podemos observar facilmente com este pequeno exemplo, que a solução óptima depende da forma de “encaixar” os vários contactos dentro do orçamento disponível, tendo em conta a selecção dos contactos com um maior ganho. Nas soluções i) , ii) e iii) o valor do orçamento não foi todo aproveitado, por isso não houve um “encaixe” perfeito dos contactos dentro do orçamento. Na solução óptima, os contactos 2, 3 e 6

aproveitam a totalidade do orçamento, ao mesmo tempo que proporcionam um valor de ganho superior.

Podemos concluir que a selecção dos contactos tendo por base esta abordagem construtiva, de escolher a melhor opção em cada momento, poderá não ser a melhor solução. A solução óptima depende de como os contactos seleccionados se vão "encaixar" entre si, logo o processo tem de ser analisado com um todo, tendo em conta todas as combinações possíveis. Neste pequeno exemplo, analisar todas as combinações possíveis seria uma tarefa simples, que até pode ser feita manualmente. No entanto, como já foi referido, para problemas com muitos milhões de combinações, essa tarefa seria impraticável, mesmo com meio computacionais evoluídos.

Com base em alguns artigos que encontrei na literatura para resolver problemas idênticos, nomeadamente o problema da múltipla mochila apresentado no capítulo da revisão da literatura, resolvi testar outros critérios de ordenação para analisar se assim obtinha melhores resultados.

No artigo de *Puchinger, Raidl, e Ulrich* sobre o problema da múltipla mochila (Puchinger, Raidl, & Pferschy, 2006) são referidos pelos autores como potenciais critérios de ordenação para uma heurística gulosa, os seguintes critérios de ordenação:

1) $\frac{Ganho_{ij}}{\sum_{i=1}^{m} Custo_{ij}}$

2) $\frac{Ganho_{ij}}{\sum_{i=1}^{m} \frac{custo_{ij}}{b_i}}$

A fórmula 1 dá o ganho relativo, que é o rácio entre o ganho de cada contacto para a campanha i, cliente j e o somatório dos custos do cliente em todas as campanhas. Desta forma a ordenação tem em consideração não só o ganho como também o custo, não só desse contacto mas de todos os contactos do cliente.

A fórmula 2 é parecida com a anterior, mas tendo em consideração a magnitude do valor de orçamento disponível em cada campanha, que são os valores b_i. Deste modo

as diferenças nas restrições referentes aos orçamentos das campanhas são consideradas, trata-se simplesmente de uma redução à mesma escala.

Também no artigo de *Akçay, Li e Xu,* onde é apresentado um algoritmo guloso para o problema da múltipla mochila (Akçay, Li, & Xu, 2007), os autores apresentam uma heurística a que chamaram *primal effective capacity heuristic (PECH).* Esta heurística usa uma ordenação baseada na capacidade efectiva calculada para cada objecto a colocar na mochila. No nosso caso, para o problema de optimização de campanhas, estamos a falar da capacidade efectiva para cada cliente. A capacidade efectiva de um cliente é dada pela fórmula:

Capacidade efectiva do cliente j:

$$C_j = min_i \left\{ \frac{Max_i}{Custo_{ij}} : Custo_{ij} > 0 \right\}, \forall j \in E$$

O conjunto E é formado por todos os clientes ainda não seleccionados em nenhum contacto em cada passo do algoritmo. Max_i é o valor máximo do orçamento para cada campanha i, e o valor $Custo_{ij}$ é o custo que cada cliente i tem na campanha j. Depois de calculada a capacidade efectiva de cada cliente, a ordenação seria feita pelo valor do ganho ponderado por essa capacidade, ou seja, a ordenação seria feita por $Ganho_{ij} * C_j$ de forma decrescente.

De modo a testar o algoritmo com base nos diferentes critérios de ordenação apresentados, foram criadas várias versão do algoritmo, usando as diferentes ordenações para avaliar e analisar os resultados obtidos.

Estes algoritmos foram desenvolvidos usando a linguagem de programação *T-SQL*. Tratam-se de *Stored Procedures* do *SQL Server 2005*.

O quadro 5 apresenta a lista de *Stored Procedures* que foram desenvolvidas para testar o algoritmo com os diferentes critérios de ordenação.

Quadro 5. Lista de *Stored Procedures* criadas para testar os diferentes critérios de ordenação.

Nome da SP	Descrição
SP_OPTIMIZACAO_1	Ordenação com base no valor do ganho por ordem decrescente
SP_OPTIMIZACAO_2	Ordenação com base no ganho por ordem decrescente e depois com base no custo por ordem crescente.
SP_OPTIMIZACAO_3	Ordenação com base no ganho relativo ao custo por ordem decrescente (fórmula 1)
SP_OPTIMIZACAO_4	Ordenação com base no ganho relativo ao custo por ordem decrescente, com os custos na mesma escala (fórmula 2)
SP_OPTIMIZACAO_5	Igual à anterior mas com a ordenação dinâmica, actualizada em cada passo.
SP_OPTIMIZACAO_6	Ordenação com base no ganho ponderada pela capacidade efectiva do cliente por ordem decrescente - *Primal effective capacity heuristic (PECH)* (fórmula 3)
SP_OPTIMIZACAO_7	Igual à anterior mas com a ordenação dinâmica, actualizada em cada passo.

Até ao momento foi analisada a primeira etapa do algoritmo guloso, que diz respeito à questão da ordenação. Vamos analisar de seguida a segunda etapa que é o processo de iteração entre os elementos da lista de contactos devidamente ordenados.

A segunda etapa do algoritmo guloso é percorrer a lista ordenada dos contactos e em cada passo avaliar a possibilidade de seleccionar esse contacto. Para isso é necessário verificar em cada passo se nenhuma das restrições que fazem parte do problema é ultrapassada. Nomeadamente, a restrição ao orçamento da campanha, a capacidade do canal e o número máximo de contactos para o cliente.

Numa primeira fase, não vamos considerar a restrição para o número mínimo de contactos a efectuar em cada campanha. Esta questão não se trata de uma restrição mas sim de uma condição imposta, será um problema que será tratado e descrito mais à frente neste trabalho.

Para verificar em cada passo se as restrições estão a ser respeitadas, será necessário criar tabelas adicionais no modelo de dados com as estatísticas para cada dimensão. Em cada iteração do algoritmo deverão ser actualizadas essas estatísticas para cada uma das dimensões. Por exemplo, é necessário criar uma tabela com a estatística para

a dimensão “campanha”. Esta tabela terá duas estatísticas por campanha, uma é o número de contactos da campanha até ao momento, a outra é o valor total do custo da campanha até ao momento.

Em cada passo do algoritmo, percorrendo a lista ordenada de contactos, será identificado um potencial contacto para fazer parte da solução. Nessa altura serão efectuadas as seguintes operações:

1. Verificar se o contacto seleccionado não vai ultrapassar nenhuma das restrições impostas, ou seja, se é válido para ser seleccionado.

2. Se o contacto for válido, em primeiro lugar será necessário actualizar as tabelas de estatísticas com a informação desse contacto seleccionado. De seguida o contacto deverá ser marcado como “seleccionado” e passamos para o próximo contacto da lista. À partida todos os contactos estarão marcados como “não seleccionado”.

3. Se o contacto não for válido, simplesmente passamos para o próximo contacto.

Para explicar melhor este processo, vou usar novamente um pequeno exemplo com apenas quatro contactos e uma campanha. Neste exemplo os ganhos e custos são distribuídos do seguinte modo:

Contacto	Ganho	Custo
1	400	80
2	350	50
3	60	15
4	140	20

Para este exemplo vamos apenas colocar uma restrição, que é o orçamento da campanha, neste caso seria de 100 euros.

Para controlar o valor de custo da campanha teríamos então de ter uma tabela de estatística para a campanha.

A tabela de estatística teria a seguinte forma:

Campanha_ID	Valor_Custo	Max_Custo
1	0	100

A primeira etapa do algoritmo é então ordenar a lista de contactos segundo um determinado critério. Vamos considerar, por exemplo, o valor do ganho. A tabela de contactos ordenada ficaria da seguinte forma:

Contacto	Ganho	Custo
1	400	80
2	350	50
4	140	20
3	60	15

A segunda etapa do algoritmo é começar a percorrer a tabela ordenada de contactos e a executar as operações descritas anteriormente.

Passo 1 – Primeira Iteração

1. Vamos verificar se o contacto 1 é valido para a campanha.

   Para isso vamos usar a tabela de estatística e verificar o valor de custo da campanha até ao momento. Neste caso, como estamos no primeiro passo, o valor de custo ainda é zero.

   Depois, vamos somar este valor de custo até ao momento com o valor de custo do potencial contacto onde estamos colocados, e comparar essa soma com o valor máximo de custo para a campanha, definido na tabela de estatística. Neste caso vamos então comprar 0+80 com 100. Como $80 \leq 100$ então o contacto ainda está dentro do orçamento da campanha, logo é valido para ser seleccionado.

2. Como o contacto é valido, vamos então actualizar a tabela de estatística da campanha, somando o valor de custo ao valor existente na tabela.

   O contacto 1 é marcado como "seleccionado" e seguimos para o próximo contacto da lista ordenada.

Neste momento a tabela de estatística ficaria da seguinte forma:

Campanha_ID	Valor_Custo	Max_Custo
1	80	100

Passo 2 – Segunda Iteração

1. Vamos verificar se o contacto 2 é valido para a campanha.
   Vamos usar a tabela de estatística e verificar o valor de custo da campanha até ao momento, que agora já tem o valor 80. De seguida, vamos somar este valor com o valor de custo do potencial contacto onde estamos colocados, e comparar essa soma com o valor máximo de custo para a campanha. Vamos então comparar 80+50 com 100. Como 130 > 100 então o contacto não está dentro do orçamento da campanha, logo não é valido para ser seleccionado.

2. Como o contacto não é valido, não vamos actualizar a tabela de estatística da campanha, a tabela ficaria igual.
   O contacto 2 fica marcado como "não seleccionado" e seguimos para o próximo contacto da lista ordenada.

No passo a seguir o processo seria idêntico, iríamos verificar que o contacto 4 é valido para a campanha e a tabela de estatística seria actualizada da seguinte forma:

Campanha_ID	Valor_Custo	Max_Custo
1	100	100

No último passo, para o último contacto, o contacto 3 não seria seleccionado uma vez que a tabela de estatística nos indicaria que o orçamento da campanha tinha sido atingido.

Com este pequeno exemplo analisámos como o algoritmo vai seleccionado os contactos em cada iteração do processo e desta forma vai construindo uma solução para o problema.

De acordo com as restrições colocadas na formalização do problema de optimização de campanhas, foram adicionadas ao modelo de dados as seguintes tabelas de estatísticas (exemplos para o cenário de teste 1):

TESTE01_MDATA_RESTRICAO_CAMPANHA_COUNT

Tabela com informação da estatística relativa ao número de contactos seleccionados para cada campanha. Além da informação de estatística, a tabela contem informação dos limites máximos e mínimos de contactos considerados para cada campanha.

Campanha_ID	Valor	Min	Max
1	0	5	8
2	0	0	8

TESTE01_MDATA_RESTRICAO_CAMPANHA_VALOR

Tabela com informação da estatística relativa ao valor de custos associados a cada campanha. Esta tabela contém ainda a informação do limite máximo de custos para cada campanha, ou seja, o orçamento disponível para cada campanha.

Campanha_ID	Valor	Max
1	0	40
2	0	97,4

TESTE01_MDATA_RESTRICAO_CANAL_COUNT

Tabela com informação da estatística relativa ao número de contactos seleccionados para cada canal. A tabela contém ainda a informação do limite máximo de capacidade de cada canal.

Canal_ID	Valor	Max
100	0	1
200	0	2
300	0	6
400	0	6
500	0	1

TESTE01_MDATA_RESTRICAO_CLIENTE_COUNT

Tabela com informação da estatística relativa ao número de contactos seleccionados para cada cliente. Cada cliente apenas pode ser contactado uma única vez, logo o valor é no máximo 1 para todos os casos.

Cliente_ID	Valor	Max
1	0	1
2	0	1
3	0	1
4	0	1
5	0	1
6	0	1
7	0	1
8	0	1
9	0	1
10	0	1

Para facilitar o processo de actualização das estatísticas, foi criada uma *Stored Procedure* (*SP*) por cada tabela de estatística. Estas rotinas permitem actualizar as estatísticas em cada iteração do algoritmo. Deste modo o processo fica mais flexível e modelar, ou seja, se for necessário acrescentar uma nova restrição no processo, é só criar a respectiva tabela de estatística e a respectiva *SP* associada a essa mesma restrição.

Estamos neste momento em condições de apresentar a formalização do primeiro algoritmo para resolver o problema de optimização de campanhas.

O algoritmo é formalizado usando uma linguagem de pseudo-código com uma terminologia ligada a sistemas de base de dados.

Algoritmo 1. – Algoritmo guloso para a optimização de campanhas

1. *CREATE TABLE CONTACTOS_INI*
   SELECT Campanha_ID, Cliente_ID, Canal_ID, Valor_Ganho, Valor_Custo, Flag_Select = 0
   FROM CONTACTOS
   ORDER BY Valor_Ganho DESC
2. *OPEN TABLE CONTACTOS_INI*
3. *MOVE FIRST RECORD*
4. *WHILE NOT EOF*
5. *@contacto = CURRENT RECORD*
6. *@flag_ok = Contacto_Valido (@contacto)*
7. *IF @flag_ok = 1 THEN*
8. *Actualizar_Estatistica (@contacto)*
9. *IF @flag_ok = 1 THEN*
10. *UPDATE CURRENT RECORD SET Flag_Select = 1*
11. *MOVE NEXT RECORD*
12. *END WHILE*

Para testar o algoritmo 1 foram usados os mesmos cenários já apresentados anteriormente para testar a rotina do *TOMLAB/CPLEX,* e foram criados mais cenários para testar problemas com de maior escala.

No total foram gerados quinze cenários diferentes, alguns dos quais com vários milhões de contactos. Para os primeiros sete cenários foi possível comparar os resultados do algoritmo 1, com os resultados do algoritmo determinístico obtidos com a rotina do *TOMLAB/CPLEX*.

Para criar os novos cenários, foi novamente usado o programa criado inicialmente para gerar cenários automaticamente. No entanto, nesta altura do desenvolvimento foi necessário adaptar esse programa para gerar também as tabelas de estatísticas para cada modelo.

Por uma questão de optimização do algoritmo, o modelo de dados também foi ligeiramente alterado. Os campos relativos aos limites usados nas restrições foram

colocados nas tabelas de estatística e retirados das tabelas onde estavam originalmente. Por exemplo, o valor do orçamento de cada campanha deixou de estar na tabela de campanhas e passou para a respectiva tabela de estatística, tal como foi apresentado anteriormente para o caso do cenário do TESTE01. Nos anexos é possível consultar o modelo de dados completo que foi usado neste projecto.

Os valores para os limites usados nas restrições foram também gerados de forma automática para cada cenário. Para gerar estes valores foram usadas as seguintes regras.

1. Para o orçamento de cada campanha i :

$$b_i = \frac{\sum_{j=1}^{n} \sum_{k=1}^{p} c_{ijk}}{2} , \forall j \in M$$

c_{ijk} é o custo associado a cada potencial contacto inicial.

2. Para a capacidade máxima de cada canal k :

$$u_k = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} y_{ijk}}{5} , \forall k \in P$$

$y_{ijk} = 1$, se a campanha i, cliente j e canal k é um potencial contacto inicial

$y_{ijk} = 0$, caso contrario

Desta forma e com base nas fórmulas apresentadas, podemos verificar que o orçamento de cada campanha é a soma de todos os custos dos contactos iniciais da campanha a dividir por dois. Ou seja, o orçamento de cada campanha é metade dos custos de todos os potenciais contactos iniciais dessa campanha. Este facto mostra que são definidos à partida para cada campanha, muito mais contactos (neste caso o dobro) do que na realidade serão seleccionados pelo processo de optimização.

Do mesmo modo, podemos verificar que a capacidade máxima de cada canal é limitada a 25% (um quinto) do total de contactos seleccionados à partida para cada canal.

Além das fórmulas apresentadas anteriormente para cada restrição, em alguns casos pontuais foram ainda alterados de forma manual um ou mais casos específicos, de forma a criar cenários com restrições diferentes.

O quadro 6 mostra os quinze cenários gerados em termos de dimensão. Como podemos verificar, a dimensão vai crescendo gradualmente, começando no primeiro cenário com apenas com vinte contactos e terminando no cenário quinze com mais de quatro milhões de contactos.

Quadro 6. Lista de cenários criados para testar o processo de optimização de campanhas.

Cenário	Nº Clientes	Nº Campanhas	Nº Canais	Nº Contactos
TESTE01	10	2	5	20
TESTE02	100	3	5	300
TESTE03	150	4	5	600
TESTE04	200	6	5	1.200
TESTE05	500	6	5	3.000
TESTE06	1.000	4	5	4.000
TESTE07	1.000	6	5	6.000
TESTE08	1.000	7	5	7.000
TESTE09	10.000	10	5	100.000
TESTE10	10.000	20	5	200.000
TESTE11	50.000	10	5	500.000
TESTE12	100.000	10	5	1.000.000
TESTE13	100.000	20	5	2.000.000
TESTE14	200.000	14	5	2.800.000
TESTE15	300.000	14	5	4.200.000

Testes ao algoritmo 1

O algoritmo 1 foi testado para os quinze cenários gerados usando diferentes critérios de ordenação. No total foram testados sete critérios diferentes de ordenação, apresentados anteriormente no quadro 6.

O seguinte quadro mostra os resultados obtidos pelo algoritmo 1 para os quinze cenários com base nos diferentes critérios de ordenação.

Quadro 7. Resultados obtidos pelo algoritmo 1 com base nos diferentes critérios de ordenação.

Cenário	Nº Contactos	Total Ganho SP_OPTIMIZACAO_1	Total Ganho SP_OPTIMIZACAO_2	Total Ganho SP_OPTIMIZACAO_3
TESTE01	20	225,88	225,88	254,40
TESTE02	300	6.571,36	6.571,36	6.481,67
TESTE03	600	12.972,60	12.972,60	12.553,20
TESTE04	1.200	150.723,55	150.768,29	150.689,79
TESTE05	3.000	294.746,26	294.746,26	294.826,77
TESTE06	4.000	313.465,32	313.980,68	309.565,40
TESTE07	6.000	412.797,22	407.316,82	407.059,13
TESTE08	7.000	1.796.075,73	1.796.170,96	1.783.504,41
TESTE09	100.000	3.999.475,60	3.999.297,63	3.960.121,18
TESTE10	200.000	4.057.061,04	4.057.330,06	4.036.433,51
TESTE11	500.000	21.233.878,30	21.232.415,81	21.081.042,35
TESTE12	1.000.000	42.444.119,08	42.444.735,86	42.180.604,34
TESTE13	2.000.000	37.591.325,96	37.590.671,74	37.420.783,51
TESTE14	2.800.000	81.481.008,37	81.479.539,63	81.194.613,06
TESTE15	4.200.000	119.479.912,44	119.487.185,93	118.701.893,05

Quadro 8. (Continuação do quadro anterior)

Cenário	Nº Contactos	Total Ganho SP_OPTIMIZACAO_4	Total Ganho SP_OPTIMIZACAO_5	Total Ganho SP_OPTIMIZACAO_6	Total Ganho SP_OPTIMIZACAO_7
TESTE01	20	254,40	252,45	254,40	252,45
TESTE02	300	6.519,76	6.581,05	6.581,05	6.581,05
TESTE03	600	12.662,39	12.939,65	12.921,27	12.921,27
TESTE04	1.200	150.482,34	150.545,50	150.800,12	150.504,99
TESTE05	3.000	295.068,21	295.710,37	294.304,05	295.710,37
TESTE06	4.000	303.940,05	316.008,58	295.691,83	315.887,72
TESTE07	6.000	407.316,82	401.617,70	402.523,55	377.406,90
TESTE08	7.000	1.784.445,29	1.774.899,08	1.776.238,87	1.765.952,00
TESTE09	100.000	3.951.262,21	3.925.713,05	3.909.018,80	3.926.277,37
TESTE10	200.000	4.035.407,68	--	4.042.437,39	--
TESTE11	500.000	21.056.310,29	--	20.690.579,56	--
TESTE12	1.000.000	42.126.474,14	--	41.714.885,49	--
TESTE13	2.000.000	37.398.633,47	--	37.233.803,85	--
TESTE14	2.800.000	81.141.923,04	--	80.741.704,19	--
TESTE15	4.200.000	118.570.189,03	--	116.914.898,17	--

Os algoritmos com base nas rotinas 5 e 7, como se tratam de ordenações dinâmicas que têm de ser reordenadas em cada passo, são muito mais lentos. Por isso apenas foram testados para os cenários inferiores a cem mil contactos.

Da análise dos quadros podemos verificar que as rotinas que apresentam os melhores resultados para mais cenários, são as rotinas 1 e 2. Ou seja, a ordenação simples pelo valor do ganho, ou então pelo valor do ganho seguido pelo valor do custo, são as que apresentam os melhores resultados.

Quando comparadas entre si as rotinas 1 e 2, podemos verificar que a rotina 1 tem em 5 cenários os melhores resultados. Enquanto que a rotina 2 tem os melhores resultados em 6 cenários. Nos restantes cenários ficam empatadas com o mesmo valor de ganho. Por isso a rotina que apresenta no geral um melhor resultado, quando comparados todos os cenários, é a rotina 2. Esta rotina usa o critério de ordenação pelo valor do ganho por ordem decrescente, seguido pelo valor do custo por ordem crescente.

Garantir número mínimo de contactos por campanha

Uma questão que ainda estava por resolver no problema de optimização de campanhas é questão de garantir um número mínimo de contactos para determinadas campanhas. Esta condição não é uma restrição do processo, antes pelo contrário, é uma condição imposta ao processo.

A solução escolhida para garantir um número mínimo de contactos para algumas campanhas, foi garantir esse mínimo de contactos à partida para cada campanha onde essa condição é imposta.

Desta forma, é feito um primeiro passo onde são identificadas as campanhas onde é imposta a condição de garantir um número mínimo de contactos. Num segundo passo, são seleccionados para cada campanha nessas condições, e de forma individual, os melhores contactos dessa campanha, tendo em conta as restrições impostas, com base num dos critérios de ordenação.

O algoritmo para garantir o número mínimo de contactos para as campanhas é definido do seguinte modo:

Algoritmo 2. – Garantir o número mínimo de contactos para cada campanha

```
1.  FOR EACH  @campanha_id   IN  CAMPANHAS
2.        @minimo_campanha =  SELECT  valor_minimo
                              FROM CAMPANHAS
                              WHERE Campanha_ID = @ campanha_id
3.        IF @minimo_campanha > 0
4.            CREATE TABLE CONTACTOS_CAMPANHA_INI
5.            SELECT TOP @minimo_campanha, Campanha_ID, Cliente_ID, Canal_ID,
                        Valor_Ganho, Valor_Custo, Flag_Select = 0
              WHERE Campanha_ID = @campanha_id ORDER BY Valor_Ganho DESC
6.        OPEN TABLE CONTACTOS_INI
7.        MOVE FIRST RECORD
8.        WHILE NOT EOF
9.            @contacto  = CURRENT RECORD
10.           @flag_ok = Contacto_Valido (@contacto)
11.           IF @flag_ok =  1 THEN
12.               Actualizar_Estatistica (@contacto)
13.           IF @flag_ok =  1 THEN
14.               UPDATE CURRENT RECORD SET Flag_Select = 1
15.       MOVE NEXT RECORD
16.       END WHILE
17. END FOR
```

O algoritmo 2 será assim o primeiro passo do processo de optimização de campanhas. Só depois ser executado este algoritmo para garantir os mínimos para cada campanha, pode ser executado o processo do algoritmo 1, para optimizar os restantes contactos inicialmente identificados para o plano de marketing.

Nesta fase do trabalho o processo de optimização é então composto pelos algoritmos 1 e 2. Em ambos os algoritmos o critério de ordenação escolhido foi a ordenação pelo valor do ganho de forma decrescente, seguida pelo valor do custo por ordem ascendente.

Para os cenários de menor dimensão, concretamente para os cenários 1 a 7, é possível comparar os valores de ganho obtidos através dos algoritmos 1 e 2, com a solução óptima obtida através da rotina determinística em *MATLAB*.

O próximo quadro mostra a diferença de valor de ganho entre a solução óptima obtida com a rotina do *MATLAB* e o algoritmo 1, usando o critério de ordenação pelo valor de ganho descendente, que é o critério com melhores resultados entre os cenários 1 a 7.

No quadro 9 podemos observar que para o cenário 7 com seis mil contactos, a diferença de ganho entre a solução óptima do *MATLAB* e a solução obtida com a heurística do algoritmo 1 é de cerca de 13 mil euros. Ou seja, para um cenário onde o total de ganho é aproximadamente 400 mil euros, estamos a perder cerca de 13 mil euros. Estamos a falar de uma perda de cerca de 3% face à solução óptima.

Quadro 9. Resultados obtidos com o algoritmo 1 comparados com a solução óptima do *MATLAB*.

Cenário	Nº Contactos	Total Ganho Algoritmo 1	Total Ganho *Matlab*	Diferença	Diferença %
TESTE01	20	225,88	255,39	29,51	13,1%
TESTE02	300	6.571,36	6.625,78	54,42	0,8%
TESTE03	600	12.972,60	13.136,00	163,40	1,3%
TESTE04	1.200	150.723,55	151.296,00	572,45	0,4%
TESTE05	3.000	294.746,26	296.896,00	2.149,74	0,7%
TESTE06	4.000	313.465,32	324.117,00	10.651,68	3,4%
TESTE07	6.000	412.797,22	426.069,00	13.271,78	3,2%

Vamos analisar de seguida alguns destes cenários mas considerando ainda a restrição de garantir um mínimo de contactos para algumas das campanhas em cada cenário.

No quadro 10 são analisados os resultados para alguns dos cenários onde foram garantidos os mínimos para algumas das campanhas. Os mínimos para as campanhas foram encontrados com base nas contagens de contactos por campanha obtidas com a primeira solução, onde não foram considerados os mínimos nas restrições.

Por o exemplo do cenário 7, as contagens com o número de contactos por campanha obtidas depois de executado o algoritmo 1, são apresentadas no próximo quadro.

Quadro 10. Resultados obtidos com o algoritmo 1 para o cenário 7.

Campanha_ID	Nº Contactos - Algoritmo 1
1	15
2	260
3	354
4	9
5	18
6	221

Para o exemplo do cenário 7 e com base nestes primeiros resultados, vamos considerar, por exemplo, no mínimo 100 contactos para as campanhas 1, 4 e 5. Podemos verificar no quadro anterior que para estas campanhas apenas tinham sido seleccionados 15, 9 e 18 contactos, respectivamente.

Continuando a analisar o exemplo do cenário 7, para garantir os mínimos para as campanhas 1, 4 e 5, foi necessário numa primeira fase, correr o algoritmo 2 de modo a garantir um mínimo de 100 contactos para cada uma destas campanhas.

Numa segunda fase foi executado o algoritmo 1 para seleccionar os restantes contactos.

Com este processo foi obtido um ganho total de 341.573,83 para este cenário. Com a rotina do *MATLAB* e garantindo de igual modo os mínimos para as campanhas 1, 4 e 5, foi obtido um total de ganho no valor de 377.764,00.

Como podemos verificar no quadro 11 a diferença entre os resultados ainda é bastante significativa, neste caso estamos a falar de uma perda superior a 10% face à solução óptima.

Quadro 11. Resultados obtidos com o algoritmo 1 e com mínimos garantidos para algumas campanhas.

Cenário	Nº Contactos	Total Ganho Algoritmo 1*	Total Ganho *Matlab*	Diferença	Diferença %
TESTE05	3.000	270.083,71	281.350,00	11.266,29	4,2%
TESTE06	4.000	308.406,57	323.240,00	14.833,43	4,8%
TESTE07	6.000	341.573,83	377.764,00	36.190,17	10,6%

*O algoritmo 1 foi executado depois de terem sido garantidos os mínimos para cada campanha com o algoritmo 2.

De forma a melhorar os resultados da solução obtida com o algoritmo 1, o próximo passo neste trabalho de projecto foi tentar introduzir um novo algoritmo, tendo por base um conceito que encontrei num dos artigos pesquisados na fase de revisão da literatura.

O artigo de *Puchinger, Raidl, e Ulrich* sobre o problema da múltipla mochila (Puchinger, et al., 2006) apresenta o conceito de "núcleo do problema" (*The core concept for multidimensional knapsack problem*). Neste artigo, os autores descrevem este conceito, como o conjunto de itens para os quais é difícil decidir se devem fazer parte da solução ou não. O conjunto formado por esses itens, onde deverá estar centrado o processo de decisão é considerado o núcleo do problema, é sobre este conjunto que o processo de optimização deverá ter um papel fundamental.

Se aplicarmos este conceito ao problema de optimização de campanhas, o núcleo do problema será formado pelo conjunto de contactos para os quais é mais difícil decidir. Será sobre esse conjunto que deverá incidir o algoritmo de optimização.

Como já verificámos anteriormente, um dos principais problemas do processo de optimização de campanhas é a sua dimensão. Para problemas de dimensão reduzida é possível obter a solução óptima através de uma rotina para o *MATLAB*. Como já foi apresentado para alguns dos cenários testados, para um número de contactos não superior a 6 mil, foi possível encontrar a solução óptima.

Desta forma, se para o problema de optimização de campanhas for possível reduzir o problema inicial e identificar um conjunto de contactos que são o núcleo do problema e formar um conjunto com menos de 6 mil contactos, talvez seja possível alcançar uma solução melhorada, fazendo incidir a rotina determinista em *MATLAB* sobre esse núcleo de contactos.

Optimização da primeira solução

De forma a poder optimizar a solução obtida com o primeiro algoritmo, foi criado um novo processo com base no conceito de "núcleo do problema" de forma a melhorar a solução obtida.

Depois de executarmos o algoritmo 1, ficamos com um conjunto de contactos que fazem parte da solução. Estes contactos são colocados numa tabela específica do modelo de dados, é a tabela CONTACTOS_OPTIMIZACAO. Na tabela de contactos iniciais, onde estão todos os contactos à partida identificados, os contactos seleccionados ficam marcados com uma *flag* a indicar se foram seleccionados. Esta tabela, com todos os contactos, é a tabela CONTACTOS (ver modelo de dados em anexo).

Uma forma de identificar os contactos que fazem parte do "núcleo do problema", ou seja, o conjunto de contactos para os quais o processo de decisão seria mais difícil, é identificar os x "piores" contactos dentro dos contactos da primeira solução e identificar os y "melhores" contactos que ficaram de fora da solução. Desta forma seriam identificados os contactos da fronteira de decisão, ou seja, onde o processo de decisão poderia ter um papel mais preponderante.

Neste caso, os x "piores" contactos seriam os *TOP* x da tabela CONTACTOS_OPTIMIZACAO ordenados pelo valor de ganho de forma ascendente. Desta forma seriam identificados os x contactos com os menores valores de ganho.

De forma idêntica, os y "melhores" contactos dos que ficaram de fora da primeira solução seriam identificados com um *TOP* y da tabela CONTACTOS onde a *flag_seleccionado* = 0 e ordenados de forma decrescente pelo valor do ganho.

De forma a não enviesar a escolha dos "piores" e "melhores" contactos, a selecção foi efectuada para cada par {Campanha_ID, Canal_ID}. Desta forma foi garantida uma amostra de contactos abrangendo todas as campanhas e canais. Isto quer dizer que, para cada par {Campanha_ID, Canal_ID} foi efectuado um *TOP* x dos "piores" contactos considerados na primeira solução e um *TOP* y dos "melhores" contactos que ficaram de fora da primeira solução.

O quadro 12 exemplifica a forma de identificar os TOP x "piores" contactos da tabela dos contactos já optimizados. Por exemplo, vamos considerar x = 2.

O quadro 13 exemplifica a forma de identificar os TOP y "melhores" contactos ainda não seleccionados da tabela dos contactos. Por exemplo, vamos considerar y = 2.

Quadro 12. Identificar os *TOP* x "piores" contactos na tabela CONTACTOS_OPTIMIZACAO

Campanha_ID	Canal_ID	Cliente_ID	Valor_Ganho	
CA1	100	CL01	22,64	*Top 2*
CA1	100	CL04	25,60	
CA1	100	CL05	26,51	
CA1	100	CL06	26,51	
CA1	100	CL09	27,12	
…	…	…	…	
CA1	200	CL20	29,12	*Top 2*
CA1	200	CL21	28,08	
CA1	200	CL23	30,08	
CA1	200	CL24	46,96	
…	…	…	…	
CA2	100	CL05	46,96	*Top 2*
CA2	100	CL08	48,83	
CA2	100	CL09	49,83	
CA2	100	CL20	50,71	
CA2	100	CL21	52,71	
…	…	…	…	
CA2	200	CL22	50,30	*Top 2*
CA2	200	CL23	51,30	
CA2	200	CL26	52,58	
CA2	200	CL30	56,58	
…	…	…	…	

Quadro 13. Identificar os *TOP* y "melhores" contactos na tabela CONTACTOS (*flag_seleccionado* = 0)

Campanha_ID	Canal_ID	Cliente_ID	Valor_Ganho	
CA1	100	CL07	24,12	*Top 2*
CA1	100	CL02	23,51	
CA1	100	CL03	22,51	
CA1	100	CL11	21,69	
CA1	100	CL12	20,64	
…	…	…	…	
CA1	200	CL15	28,12	*Top 2*
CA1	200	CL16	26,08	
CA1	200	CL17	25,08	
CA1	200	CL18	16,96	
…	…	…	…	
CA2	100	CL01	40,96	*Top 2*
CA2	100	CL02	40,83	
CA2	100	CL04	39,83	
CA2	100	CL10	30,71	
CA2	100	CL11	22,71	
…	…	…	…	
CA2	200	CL15	48,30	*Top 2*
CA2	200	CL16	46,30	
CA2	200	CL17	42,58	
CA2	200	CL18	36,58	
…	…	…	…	

Depois de identificados estes dois conjuntos, foi criada uma nova tabela no modelo de dados apenas com os contactos neles identificados. Os valores de x e y foram definidos de tal forma que o conjunto final de contactos (núcleo do problema) não fosse superior a seis mil. Desta forma seria possível usar a rotina do *MATLAB* para determinar a solução óptima para este conjunto de contactos.

Para a construção desta nova tabela foram criadas duas tabelas auxiliares, CONTACTOS_OPTIMIZACAO_DELETED e CONTACTOS_INSERTED. Na primeira tabela foram colocados os x "piores" contactos da primeira solução, estes contactos foram apagados da primeira tabela de optimização CONTACTOS_OPTIMIZACAO. Na segunda tabela foram colocados os y "melhores" contactos que tinham ficado de fora na primeira solução.

Desta forma, para cada cenário foi criada a tabela CONTACTOS_OPTIMIZACAO_FINAL, que tem apenas os contactos considerados para a optimização final, ou seja, os contactos do chamado "núcleo do problema" (consultar o modelo de dados em anexo). Por exemplo, para o cenário 1 foi criada a tabela TESTE01_CONTACTOS_OPTIMIZACAO_FINAL.

Outro aspecto a ter em atenção no modelo de dados, é o facto das tabelas de estatística terem de ser actualizadas com os respectivos reajustamentos nos valores das estatísticas. Por exemplo, o valor da tabela MDATA_RESTRICAO_CANAL_COUNT teve de ser reajustado tendo em conta os contactos apagados da primeira solução para cada canal.

Depois de identificados os contactos considerados para o "núcleo do problema" e depois de actualizadas as tabelas de estatísticas, era necessário criar o ficheiro para o *MATLAB* poder executar a rotina *BINTPROG* com base nestes dados. Para isso foi usado novamente o programa que gera automaticamente um ficheiro *MATLAB* (com extensão ".m") com base no modelo de dados.

É apresentado de seguida o algoritmo mais formal para identificar os contactos que irão fazer parte do processo de optimização final.

Algoritmo 3. – Determinar os contactos para o núcleo do problema

```
1. - - Parâmetros de entrada: @cenario_id
2. - - Identificar os TOP x "piores" contactos da primeira solução
3. FOR EACH  @campanha_id, @canal_id   IN  CONTACTOS_OPTIMIZACAO
        @topx = calcular_top(@cenario_id)
        CREATE TABLE CONTACTOS_OPTIMIZACAO_DELETED
        SELECT      TOP @topx
                    Campanha_ID, Cliente_ID, Canal_ID, Valor_Ganho
        FROM CONTACTOS_OPTIMIZACAO
        WHERE Campanha_ID = @ campanha_id and Canal_ID = @canal_id
              and Flag_Select = 0
        ORDER BY Valor_Ganho ASC
        DELETE CONTACTOS_OPTIMIZACAO
        SELECT      TOP @topx
                    Campanha_ID, Cliente_ID, Canal_ID, Valor_Ganho
        WHERE Campanha_ID = @ campanha_id and Canal_ID = @canal_id
              and Flag_Select = 0
        ORDER BY Valor_Ganho ASC
4. END FOR
   - - Identificar os TOP y "melhores" contactos que ficaram de fora da primeira solução
5. FOR EACH  @campanha_id, @canal_id   IN  CONTACTOS
        @topx = calcular_top(@cenario_id)
        CREATE TABLE CONTACTOS_INSERTED
        SELECT      TOP @topx
                    Campanha_ID, Cliente_ID, Canal_ID, Valor_Ganho
         WHERE Campanha_ID = @ campanha_id and Canal_ID = @canal_id
        ORDER BY Valor_Ganho DESC
6. END FOR
7. CREATE TABLE CONTACTOS_OPTIMIZACAO_FINAL
     SELECT   Campanha_ID, Cliente_ID, Canal_ID,Valor_Custo, Valor_Ganho
     FROM CONTACTOS_OPTIMIZACAO_DELETED
    UNION ALL
    SELECT   Campanha_ID, Cliente_ID, Canal_ID,Valor_Custo, Valor_Ganho
    FROM CONTACTOS_INSERTED
```

Testes ao algoritmo de optimização final

Para os cenários analisados anteriormente, vamos agora analisar os resultados obtidos depois de aplicar este novo algoritmo de optimização final sobre a primeira solução.

No próximos quadros podemos analisar os resultados obtidos com o algoritmo de optimização final quando aplicado à solução obtida com o algoritmo 1.

No quadro 14 podemos analisar os incrementos dos valores de ganho depois de aplicar o algoritmo de optimização final e a aproximação com os valores da solução óptima.

Podemos observar que, por exemplo para o cenário 7, o algoritmo de optimização final vai permitir um incremento no valor de ganho de quase 5 mil euros. E para os cenários de maior dimensão, por exemplo, para os cenários 11 e 12, o algoritmo de optimização final vem acrescentar um valor de ganho superior a 16 mil euros.

Quadro 14. Resultados obtidos pelo algoritmo de optimização final para a primeira solução

Cenário	Nº Contactos	Total Ganho Algoritmo 1	Total Ganho Optimização Final	Incremento no Valor de Ganho
TESTE01	20	225,88	255,39	29,51
TESTE02	300	6.571,36	6.625,78	54,42
TESTE03	600	12.972,60	13.136,00	163,40
TESTE04	1.200	150.723,55	151.293,76	570,21
TESTE05	3.000	294.746,26	295.967,32	1.221,06
TESTE06	4.000	313.465,32	316.712,92	3.247,60
TESTE07	6.000	412.797,22	417.419,12	4.621,90
TESTE08	7.000	1.796.075,73	1.807.572,22	11.496,49
TESTE09	100.000	3.999.475,60	4.011.794,03	12.318,43
TESTE10	200.000	4.057.061,04	4.066.031,75	8.970,71
TESTE11	500.000	21.233.878,30	21.250.130,60	16.252,30
TESTE12	1.000.000	42.444.119,08	42.460.671,37	16.552,29
TESTE13	2.000.000	37.591.325,96	37.598.216,43	6.890,47
TESTE14	2.800.000	81.481.008,37	81.490.596,83	9.588,46
TESTE15	4.200.000	119.479.912,44	119.495.586,71	15.674,27

No quadro 15 podemos analisar as diferenças de valores de ganho que são obtidas com o algoritmo de optimização final face aos valores obtidos com a solução óptima obtida com o *MATLAB*. Podemos observar que estas diferenças são mínimas e que

melhoram bastante face à mesma analise feita num dos quadros anteriores (quadro 8) para o algoritmo 1.

Quadro 15. Resultados obtidos com o algoritmo de optimização final comparados com a solução óptima do *MATLAB*.

Cenário	Nº Contactos	Total Ganho Optimização Final	Total Ganho *Matlab*	Diferença	Diferença %
TESTE01	20	255,39	255,39	0,00	0,0%
TESTE02	300	6.625,78	6.625,78	0,00	0,0%
TESTE03	600	13.136,00	13.136,00	0,00	0,0%
TESTE04	1.200	151.293,76	151.296,00	2,24	0,0%
TESTE05	3.000	295.967,32	296.896,00	928,68	0,31%
TESTE06	4.000	316.712,92	324.117,00	7.404,08	2,34%
TESTE07	6.000	417.419,12	426.069,00	8.649,88	2,07%

Os resultados obtidos nos quadros anteriores não tiveram em consideração a restrição de garantir os mínimos de contactos para as campanhas. No próximo quadro podemos agora observar os resultados obtidos com o algoritmo de optimização final, para alguns dos cenários onde são garantidos um número mínimo de contactos para algumas das campanhas. Estas restrições são as mesmas que já foram consideradas na análise anterior para o caso do algoritmo 1 (ver analise no quadro 11).

Os resultados do algoritmo de optimização final para estes casos são ainda mais significativos. Podemos observar que no caso do algoritmo 1 as diferenças para os cenários 5, 6 e 7 eram de 4.2%, 4.8% e 10.6%, respectivamente, e agora com o algoritmo de optimização final temos umas diferenças para a solução óptima de apenas 2.5%, 3.4% e 6.1% para os mesmos cenários.

Quadro 16. Resultados obtidos com o algoritmo de optimização final, com os mínimos garantidos para algumas campanhas.

Cenário	Nº Contactos	Total Ganho Optimização Final	Total Ganho *Matlab*	Diferença	Diferença %
TESTE05	3.000	274.482,13	281.350,00	6.867,87	2,5%
TESTE06	4.000	312.552,97	323.240,00	10.687,03	3,4%
TESTE07	6.000	356.107,58	377.764,00	21.656,42	6,1%

Estamos neste momento em condições de apresentar um processo geral de optimização de campanhas. Este processo é definido por uma estrutura de várias etapas e algoritmos que visam alcançar uma solução aproximada para o problema que é proposto resolver com esta tese.

Este processo de optimização está então estruturado nos seguintes passos:

1. Aplicar um algoritmo heurístico guloso a cada campanha de forma a garantir um número mínimo de contactos para as campanhas. Este processo é implementado pelo algoritmo 2 apresentado anteriormente (ver o código em anexo).

2. Aplicar um algoritmo heurístico guloso para todos os contactos inicialmente identificados, de forma a seleccionar os melhores contactos para cada campanha. Desta forma obtemos uma primeira aproximação da solução óptima. Esta heurística é implementada pelo algoritmo 1 apresentado anteriormente (ver o código em anexo).

3. Determinar o conjunto de contactos que formam o "núcleo do problema" e actualizar o modelo de dados com base neste conjunto de contactos. Este processo é implementado pelo algoritmo 3 apresentado anteriormente.

4. Com base no modelo de dados actualizado e com base na tabela que identifica o conjunto de contactos que forma o "núcleo do problema", gerar o ficheiro para *MATLAB*, com extensão ".m" que permite resolver o problema de optimização para este conjunto restrito de contactos. Este ficheiro é gerado por um dos programas desenvolvido em *T-SQL* apresentado em anexo.

5. Usar a ferramenta *MATLAB* para executar o ficheiro com extensão ".m" gerado no passo anterior e extrair os resultados da rotina *BINTPROG* para um ficheiro Excel.

6. Actualizar o modelo de dados com base nos resultados obtidos no ficheiro Excel e construir assim a solução final, tendo por base os contactos resultantes do passo 2 e substituindo alguns desses contactos por outros encontrados no passo 4.

Este processo definido por estes seis passos ou etapas, pode ser completamente automatizado e implementado numa sequência automática de chamadas a cada um dos algoritmos intervenientes. Será este o processo a que chamamos "o processo de optimização de campanhas".

O processo normal de execução de campanhas

Para podermos comparar os resultados das soluções obtidas pelo algoritmo 1 e pelo algoritmo de optimização final, com os resultados obtidos pelo processo normal, sem qualquer processo de optimização, foi necessário simular o funcionamento do processo normal para os cenários gerados.

O processo normal de execução do plano de campanhas, tal como foi referido no capitulo de revisão da literatura, é a execução das campanhas por ordem de prioridade e de acordo com os critérios definidos individualmente para cada campanha.

Desta forma, para podermos simular o processo normal, vamos considerar que cada campanha é executada de forma individual e que são implementadas pela ordem do ID da campanha. De modo a termos clientes garantidos para todas as campanhas, vamos distribuir os clientes disponíveis em cada cenário pelo número de campanhas de cada cenário. Por exemplo, o cenário 7 tem 1.000 clientes e 6 campanhas, neste caso vamos considerar que cada campanha vai ter 200 clientes e que as campanhas são implementadas pela ordem do ID, ou seja, primeiro a campanha 1, depois a campanha 2, etc. Em todo o caso, mesmo com este processo de seleccionar os contactos para cada campanha de forma individual, é necessário garantir que as restrições são garantidas, isto é, que a capacidade dos canais não é excedida, que o orçamento de cada campanha é respeitado, e que cada cliente apenas é contactado uma única vez.

Este processo é exactamente o mesmo que foi usado no algoritmo 2 para garantir os mínimos para cada campanha, só que neste caso os valores mínimos são o número de contactos que pretendemos seleccionar para cada campanha. Desta forma é possível simular o processo normal de execução das campanhas, através do algoritmo 2 que foi criado anteriormente para garantir o número mínimo de contactos para cada

campanha. Relembro que este algoritmo ordena os contactos de uma determinada campanha por ordem decrescente de ganho e percorre esses contactos verificando em cada passo se deve ou não seleccionar o contacto para essa campanha, de acordo com as restrições impostas. Para isso usa umas tabelas de estatísticas que são actualizadas em cada iteração. Este algoritmo difere do algoritmo 1, na medida em que o algoritmo 1 ordena toda a lista de contactos, para todas as campanhas, neste caso o algoritmo 2 ordena a lista de contactos apenas da campanha em causa. Neste caso, algoritmo 2 é executado várias vezes, uma vez para cada campanha do plano de marketing.

O primeiro passo para simular este processo é identificar o número de contactos que pretendemos colocar em cada campanha, para isso vamos simplesmente dividir o número de clientes pelo número de campanhas existente. Por exemplo, para o cenário 7, onde temos 1.000 clientes e 6 campanhas, vamos considerar que cada campanha deverá ter 200 clientes. Começamos por executar o algoritmo 2 para a campanha 1, onde vamos seleccionar 200 clientes para essa campanha. A seguir fazemos o mesmo para a campanha 2 e assim sucessivamente.

Algoritmo 4. – Simulação do processo de execução do plano de campanhas pela forma normal, sem usar o processo de optimização.

- - Parâmetros de entrada: @cenario_id

- - Identificar o número de contactos para cada campanha do cenário

1. *@n_clientes = SELECT Count(Distinct Cliente_ID) FROM CONTACTOS*
2. *@n_campanhas = SELECT Count(Distinct Campanha_ID) FROM CONTACTOS*
3. *@n_contactos = @n_clientes / @n_campanhas*

- - Executar o Algortimo 2 para cada campanha

4. *FOR EACH @campanha_id IN CONTACTOS*
5. *EXEC Algoritmo_2 @campanha_id , @n_contactos*
6. *END FOR*

No quadro 17 podemos analisar os valores obtidos pelo processo normal de execução do plano de campanhas para cada cenário. Podemos comparar esses resultados com os valores obtidos pelo processo de optimização.

Quadro 17. Resultados obtidos com o algoritmo de optimização final comparados com a solução obtida pelo processo normal de execução de campanhas, sem usar o processo de optimização.

Cenário	Nº Contactos	Total Ganho Optimização Final	Total Ganho *Processo Normal*	Diferença	Diferença %
TESTE01	20	255,39	225,88	29,51	13,1%
TESTE02	300	6.625,78	5.866,07	759,71	13,0%
TESTE03	600	13.136,00	12.122,77	1.013,23	8,4%
TESTE04	1.200	151.293,76	118.639,42	32.654,34	27,5%
TESTE05	3.000	296.391,62	193.866,69	102.524,93	52,9%
TESTE06	4.000	319.640,41	229.917,50	89.722,91	39,0%
TESTE07	6.000	419.643,34	294.625,08	125.018,26	42,4%
TESTE08	40.000	1.807.572,22	1.326.917,50	480.654,72	36,2%
TESTE09	100.000	4.011.794,03	2.924.574,41	1.087.219,62	37,2%
TESTE10	200.000	4.066.031,75	2.918.987,73	1.147.044,02	39,3%
TESTE11	500.000	21.250.130,60	14.895.663,92	6.354.466,68	42,7%
TESTE12	1.000.000	42.460.671,37	34.131.024,51	8.329.646,86	24,4%
TESTE13	2.000.000	37.598.216,43	29.575.018,01	8.023.198,42	27,1%
TESTE14	2.800.000	81.490.596,83	53.332.239,24	28.158.357,59	52,8%
TESTE15	4.200.000	119.495.586,71	85.358.992,22	34.136.594,49	40,0%

Da análise do quadro 17 podemos verificar que os totais de ganho obtidos pelo processo de optimização de campanhas são bastante melhores que os resultados obtidos pelo processo de execução de campanhas normal.

No entanto, não podemos esquecer que este processo aqui considerado “normal”, é apenas uma forma objectiva de tentar simular o funcionamento normal de uma área de gestão de campanhas. Normalmente o número de contactos que entram em cada campanha, além de ser o resultado de várias análises pontuais feitas pelo gestor da campanha, é muitas vezes o resultado de um processo de negociação por parte do gestor da campanha com as várias áreas intervenientes no processo. Como por exemplo, os gestores de produtos, a área de comunicação, a área de gestão de segmentos, etc.

Considerando a dificuldade em simular o processo normal de execução de campanhas, a melhor forma de avaliar este processo seria analisar alguns casos reais que já tinham sido implementados na instituição onde eu trabalho. No entanto, nesse caso a dificuldade seria em simular o processo de optimização de campanhas para esses casos reais. Isso obrigava a efectuar uma extracção e selecção prévia de contactos para as campanhas, tendo em conta que haveria um processo de optimização no final. Este processo implicava uma análise e nova implementação dessas campanhas, o que se tornou impraticável para este trabalho.

Apesar do processo dito "normal" não ser possível de simular neste ambiente de testes com vários cenários. Podemos concluir com esta análise, que a selecção dos clientes para uma campanha, quando é feita de forma isolada, tendo em conta apenas as condições na altura de execução da campanha, e tendo em conta apenas essa campanha, sem considerar as campanhas planeadas para o futuro, pode ter um impacto bastante negativo no *ROI* do plano de marketing.

2.7 Síntese e Conclusão

O desenvolvimento de um processo para a resolução do problema de optimização de campanhas é o objectivo principal deste trabalho de tese. Neste capítulo foi apresentado o processo de desenvolvimento de uma possível solução para este problema, seguindo uma metodologia de investigação apresentada anteriormente como "*Action Research*", o que significa que o processo de investigação foi evoluindo gradualmente à medida que se implementava na prática a solução investigada.

Modelo de dados

A primeira fase do processo de desenvolvimento foi a obtenção de dados. Para isso foi desenvolvido um modelo de dados em *SQL Server* e foi criado um programa que permitiu criar vários cenários gerando e alimentando um modelo de dados para cada cenário.

O modelo de dados foi criado com base no modelo real, actualmente existente no *datamart* de apoio à gestão de campanhas da instituição onde eu trabalho. O programa para alimentar o modelo de dados e gerar os vários cenários foi desenvolvido com base num conjunto de regras, também elas retiradas do ambiente real na instituição onde eu trabalho.

Processo de desenvolvimento

Depois de ser definido o modelo de dados e de terem sido gerados vários cenários de teste, foi então iniciado um processo evolutivo de desenvolvimento formado por várias etapas. Este processo terminou com a definição de um processo geral de optimização de campanhas que permite resolver o problema colocado.

De forma resumida podemos sintetizar o processo de desenvolvimento em três etapas fundamentais:

A primeira etapa foi implementar um processo determinístico que permitisse obter a solução óptima para o problema. Este processo foi conseguido através de uma biblioteca de rotinas para o *MATLAB* chamada de *TOMLAB*, neste caso foi usada a rotina *BINTPROG*. No entanto, esta solução revelou-se eficaz apenas para cenários de reduzida dimensão, para cenário de dimensão maior a rotina deu erro de falta de memória.

A segunda etapa foi desenvolver um algoritmo heurístico que permitisse num tempo praticável obter uma aproximação da solução óptima, mesmo para os cenários de maior dimensão. Baseado em vários artigos da revisão da literatura, foi desenvolvido um algoritmo guloso. Este algoritmo baseou-se na ordenação dos contactos e usou tabelas de estatísticas para controlar as restrições. O algoritmo mostrou-se bastante eficiente mesmo para problemas de elevadíssima dimensão, com milhões de contactos.

A terceira etapa foi um processo de melhoramento da solução obtida na etapa anterior. A ideia foi aplicar o processo determinístico desenvolvido na primeira etapa, mas apenas a um conjunto reduzido de contactos situados na fronteira da decisão, ou

seja, seleccionando alguns dos piores contactos que já tinham sido escolhidos no processo anterior, e alguns dos melhores contactos que tinham ficado fora dessa selecção. Este conjunto restrito de contactos foi chamado o "núcleo do problema".

A solução final para o processo de resolução do problema de optimização de campanhas é desta forma formada por estas três etapas.

Avaliação e análise de resultados

Uma tarefa importante ao longo de todo o processo de desenvolvimento foi a avaliação e a análise dos resultados.

Esta avaliação foi efectuada tendo em consideração o impacto que o processo de optimização poderia vir a ter no *ROI* do plano de campanhas. Para os casos de menor dimensão, onde foi possível obter a solução óptima, foi importante analisar as diferenças entre os resultados obtidos e a solução óptima do problema.

Na elaboração destes testes foi ainda analisada a situação em que existem restrições relacionadas com o número de contactos mínimos a efectuar em cada campanha. Como esta questão foi uma característica que condicionou fortemente o algoritmo desenvolvido, foram efectuados dois conjuntos de testes distintos. Um primeiro conjunto onde esta restrição não é colocada em nenhuma campanha, e um segundo onde esta restrição é colocada em algumas das campanhas.

Programas desenvolvidos

Para a implementação do processo de optimização de campanhas foram desenvolvidos neste capitulo vários programas e algoritmos.

1. A construção de um programa para gerar de forma automática o modelo de dados e preencher esse modelo de acordo com os parâmetros introduzidos. O programa foi desenvolvido em Script *T-SQL* para *SQL Server 2005*, uma vez que é a plataforma onde assentam os dados.

2. A construção de um programa que permitisse criar de forma automática, para cada um dos cenários gerados, um ficheiro com extensão ".m" para o *MATLAB* interpretar e resolver o problema através de um processo determinístico. Este programa foi desenvolvido em *T-SQL* para *SQL Server 2005* e o resultado é um ficheiro com extensão ".m" para posteriormente correr no *MATLAB*.

3. A construção de um algoritmo para garantir um número mínimo de contactos para algumas das campanhas onde essa condição seja imposta. Este algoritmo foi desenvolvido em *T-SQL* para *SQL Server 2005*, usando uma heurística gulosa.

4. A construção de um algoritmo para a optimização das campanhas tendo em consideração um conjunto de restrições impostas. Este algoritmo foi desenvolvido em *T-SQL* para *SQL Server 2005*, usando uma heurística gulosa.

5. A construção de um programa que permita identificar o conjunto de contactos para o "núcleo do problema" e permita gerar um ficheiro com extensão ".m" para o *MATLAB* resolver esse problema. O programa deve substituir os contactos seleccionados melhorando assim a primeira solução. Este programa foi desenvolvido em *T-SQL* para *SQL Server 2005.*

6. A construção de um algoritmo que permite simular o processo de implementação do plano de marketing sem usar nenhum algoritmo de optimização, ou seja, seleccionando os contactos de acordo com as características individuais de cada campanha e pela ordem de entrada de cada campanha. Este programa usa o algoritmo do ponto 3 para garantir os mínimos para cada campanha.

Capitulo V – CONCLUSÕES E RECOMENDAÇÕES

5. Conclusões e Recomendações

5.1 Síntese e conclusões finais

O problema

O problema de optimização de campanhas é um problema complexo que envolve a utilização de metodologia de *Database Marketing* e ferramentas de *Business Intelligence* e suporte à decisão.

Este problema é uma consequência das estratégias de *CRM* implementadas pelas empresas e surge devido ao elevado número de acções de marketing directo que são implementadas nesta área.

Em primeiro lugar, a resolução deste problema envolve um planeamento estratégico e a definição de uma metodologia para implementação e execução do plano de marketing da empresa. A execução de campanhas e a geração dos contactos deve ser feita com base num conjunto de pressupostos e condições tendo em conta a existência de um processo geral de optimização.

A resolução prática do problema passa por um processo de decisão onde é necessário seleccionar quais os melhores contactos a efectuar tendo em conta um conjunto variado de condições e restrições a considerar.

Neste trabalho de projecto, o problema de optimização de campanhas foi analisado e colocado como um problema de investigação operacional, nomeadamente como um problema de optimização.

O problema foi analisado e comparado com outros problemas conhecidos de investigação operacional. Foi formalizado como um problema de programação linear binária, na medida em que é um problema de decisão onde a função a optimizar é linear e as variáveis de decisão assumem os valores binários, zero e um.

O principal objectivo deste trabalho de projecto foi dar resposta a uma necessidade existente no departamento de marketing da instituição onde trabalho, onde o

problema de optimização de campanhas se tem colocado. Por isso, a apresentação e formalização do problema foi motivada e orientada para as necessidades específicas existentes nessa área. Embora esta formalização do problema se possa aplicar a muitas situações em várias empresas.

Na formalização do problema foram consideradas um total de cinco restrições. Três restrições ao nível das campanhas, uma restrição ao nível dos canais e outra ao nível dos clientes. No entanto, outras restrições poderiam ter sido colocadas, inclusive a outros níveis, por exemplo, ao nível do tipo de campanha.

O processo de desenvolvimento

Depois de devidamente formalizado, o problema foi analisado e foi iniciado o processo de desenvolvimento de uma solução. Neste processo de desenvolvimento foi usada uma metodologia de investigação do tipo *Action Research*, o que significa que o processo de investigação e desenvolvimento foi baseado na implementação prática das possíveis soluções. Tratou-se desta forma de um processo evolutivo, onde se foram analisando os resultados, sempre com o objectivo de melhorar a solução anterior e de dar resposta a todas as questões colocadas na formalização do problema. Desta forma, as diferentes questões foram sendo resolvidas em vários passos, procurando atingir uma solução o mais aproximada possível da solução óptima.

Uma etapa preliminar antes de se iniciar o desenvolvimento de uma solução foi a necessidade de criar um programa para gerar de forma automática vários cenários, desta forma foi possível criar problemas de diferentes dimensões para testar e analisar as soluções. No total foram gerados e usados 15 cenários de teste.

O primeiro algoritmo

O primeiro passo no desenvolvimento de uma solução foi tentar resolver o problema através de um algoritmo determinístico usando a ferramenta *MATLAB* e uma das rotinas da biblioteca *TOMLAB/CPLEX*. Este algoritmo garantia a solução óptima, no

entanto, devido à grande escala do problema, esta solução acabou por se revelar pouco eficaz, na medida em que apenas permitia a resolução de problemas de pequena escala. Mesmo assim, esta solução foi implementada e testada em cenários de dimensão até seis mil variáveis, ou seja, seis mil oportunidades de contacto. Para problemas de maior escala o algoritmo começou dar problema de falta de memória para carregamento dos dados nas matrizes de parâmetros de entrada.

O segundo passo do processo de desenvolvimento foi a construção de uma heurística baseada num algoritmo guloso (*greedy*). Esta heurística tem por base uma ordenação dos contactos e para o efeito foram testados diferentes algoritmos com vários critérios de ordenação. Para testar esta heurística foram usados os 15 cenários gerados, alguns dos quais com vários milhões de contactos.

Um dos problemas do algoritmo anterior é que não garantia o número mínimo de contactos para as campanhas, por isso foi necessário criar um algoritmo específico para este fim, que seria o primeiro passo antes de se aplicar o algoritmo geral de optimização.

Outro problema no desenvolvimento do algoritmo, foi a escolha dos critérios de ordenação. A escolha deste critério influenciava muito a solução, por isso foi necessário testar vários critérios de ordenação e avaliar os resultados obtidos. Depois de analisados os resultados, a escolha foi simplesmente ordenar pelo valor de ganho. Surpreendentemente este critério foi no geral o que apresentou melhores ganhos.

Optimização final

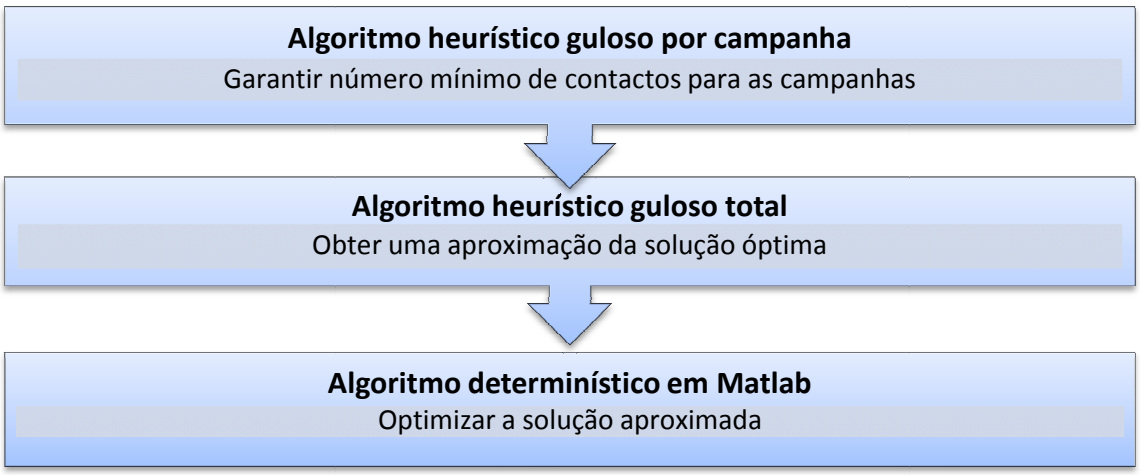
O algoritmo determinístico implementado em *MATLAB* não permitiu a resolução de problemas de elevada escala, por isso foi implementado um algoritmo heurístico guloso para obter uma solução aproximada do problema.

Baseado num conceito encontrado num dos artigos de revisão da literatura, o conceito de "núcleo do problema" (Puchinger, et al., 2006), surgiu então a ideia de aproveitar o algoritmo desenvolvido em *MATLAB* para optimizar a solução aproximada obtida com o algoritmo guloso. A ideia é encontrar o "núcleo do problema", que neste caso são os

contactos situados nas fronteiras da decisão, tendo em conta um critério de ordenação, e para o núcleo do problema aplicar o algoritmo determinístico. É claro que este conjunto de contactos que formam o "núcleo do problema" teria de ter uma dimensão reduzida de tal forma que se pudesse aplicar o algoritmo determinístico.

Resumindo, o processo de optimização de campanhas envolve as seguintes etapas:

Figura 9. Etapas do processo de optimização de campanhas



Resultados finais

Para podermos avaliar os resultados obtidos nos vários cenários gerados para o problema de optimização de campanhas, foi necessário criar um processo para simular a execução do plano de campanhas sem usar nenhum algoritmo de optimização.

É necessário temos em consideração que a metodologia de execução de campanhas, no caso de não existir um processo de optimização geral, é uma metodologia com características muito diferentes.

No caso de não existir um processo de optimização, as campanhas são executadas por uma ordem de prioridade pré-definida e tendo em conta apenas os objectivos de cada campanha. O número de contactos que neste caso são gerados para cada campanha, depende de uma análise do gestor de campanhas e de uma negociação com os gestores de produtos e com a área de comunicação.

Neste trabalho de projecto, foi considerado que o número de contactos a efectuar para cada campanha, no caso de não existir um processo de optimização, seria encontrado, dividindo de igual modo o número de clientes existente em cada cenário pelo número de campanhas desse cenário.

Num dos quadros apresentados (quadro 17) podemos observar que as diferenças dos resultados obtidos por este processo de optimização, quando comparados com os resultados obtidos pelo processo normal, com os pressupostos anteriormente referidos, são bastante significativas, pela positiva. As diferenças no valor de ganho, são na maioria dos casos, na ordem dos trinta ou quarenta por cento, chegando a atingir num dos caso os cinquenta por cento.

É claro que na realidade estes não são os ganhos reais alcançados com o processo de optimização, porque o número de contactos para cada campanha pelo processo normal é definido com base em critérios que não conseguimos replicar em ambiente de testes. No entanto, esta comparação dá-nos alguma evidência do papel que um processo de optimização pode vir a ter no lucro de uma empresa que usa este tipo de estratégias.

5.2 Dificuldades encontradas

A primeira grande dificuldade foi a obtenção dos dados para a construção dos vários cenários. Foi necessário um primeiro trabalho na instituição onde eu trabalho para recolher os dados e a informação necessária junto dos gestores de campanhas, de modo a poder desenvolver um algoritmo que permitisse gerar os cenários de forma lógica e coerente com a realidade.

Outra dificuldade foi a obtenção de uma licença para a utilização das rotinas *TOMLAB*. Este Software é muito restrito e especifico e não foi fácil obter o prolongamento da primeira licença de avaliação. Foi necessário enviar um e-mail a um dos responsáveis da *TOMLAB* a explicar a necessidade de um prolongamento da licença, e mesmo assim foi-me enviada uma nova licença para um período muito curto de utilização.

Outra das dificuldades encontradas, foi a comparação dos resultados obtidos com o processo normal de gestão de campanhas, sem a utilização de um processo de optimização. Os pressupostos que devem ser considerados no caso de não existir um processo de optimização de campanhas são muito diferentes, por isso é muito difícil simular esse cenário. Neste caso, a simulação do processo normal foi efectuada considerando que os clientes existentes seriam repartidos de forma igual pelas várias campanhas do plano de marketing.

5.3 Recomendações de trabalho futuro

O trabalho desenvolvido neste projecto procurou dar resposta a uma necessidade específica da empresa onde eu exerço a minha actividade profissional. Neste contexto, todo o trabalho desenvolvido foi orientado tendo por base a minha experiência profissional e as necessidades especificas da empresa onde trabalho.

Este tipo de actividades começam a ser cada vez mais utilizadas em diferentes negócios para além da banca, o marketing directo é hoje uma estratégia usada em muitas empresas como forma de ganhar competitividade. Por isso este problema vai afectar muitas dessas áreas. Penso que seria interessante analisar este problema noutros contextos e em diferentes áreas de negócio para formalizar o problema de forma mais genérica e abrangente tendo em conta as diferentes realidades.

Para além da questão mais teórica relacionada com a formalização do problema, a parte prática da construção do algoritmo é um problema muito interessante, principalmente para programadores e profissionais mais ligadas à área da computação. Nesse aspecto penso que seria interessante usar algumas metodologias menos convencionais para a resolução deste problema, como as redes neuronais ou os algoritmos genéticos.

REFERÊNCIAS BIBLIOGRÁFICAS

Akçay, Y., Li, H., & Xu, H. S. (2007). Greedy algorithm for the general multidimensional knapsack problem. *Annals of Operations Research, 150*(1), 17-29.

Balachandar, S. R., & Kannan, K. (2009). A new heuristic approach for the large-scale Generalized assignment problem. *International Journal of Mathematical and Statistical Sciences, 1*(4).

Berry, J. (2006). Benefits of Using a Decision Engine to Optimise Campaign Planning for Direct Marketing. *Database Marketing & Customer Strategy Management, 13*(4), 319-323.

Clausen, J. (1999). Branch and Bound Algorithms - Principles and Examples. *Department of Computer Science, University of Copenhagen*.

Coghlan, D., & Brannick, T. (2005). *Doing action research in your own organization* (Second ed.). London: Sage.

Cohen, M. D. (2002, Jul 23-26). *Exploiting response models - optimizing cross-sell and up-sell opportunities in banking.* Paper presented at the *8th International Conference on Knowledge Discovery and Data Mining (KDD 2002)*, Edmonton, Canada.

Cornelius, T. L. (2001). *Computer Aided Design, Engineering, and Manufacturing: Systems techniques and computational methods* (Vol. Systems Techniques and Computational Methods). Florida: CRC Press.

Davis, M., Sigal, R., & Weyuker, E. J. (1994). *Computability, complexity, and languages: fundamentals of theoretical computer science* (Second ed.). London: Academic Press.

Doyle, S. (2005). Business requirements for campaign management - A sample framework. *Database Marketing & Customer Strategy Management, 12*(2), 177-192.

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

Gronroos, C. (1996, Mar 29-31). *Relationship marketing: Challenges for the organization.* Paper presented at the *International Conference on Relationship Marketing*, Berlin, Germany.

Gronroos, C. (2009). Marketing as promise management: regaining customer management for marketing. [Article]. *Journal of Business & Industrial Marketing, 24*(5-6), 351-359.

Gurari, E. M. (1989). *An Introduction to the Theory of Computation*. New York: Computer Science Press.

Herr, K., & Anderson, G. L. (2005). *The action research dissertation: a guide for students and faculty*. London: Sage.

Hsu, T. H., Tsai, T. N., & Chiang, P. L. (2009). Selection of the optimum promotion mix by integrating a fuzzy linguistic decision model with genetic algorithms. [Article]. *Information Sciences, 179*(1-2), 41-52.

Kim, Y. H., & Moon, B. R. (2006). Multicampaign assignment problem. [Article]. *Ieee Transactions on Knowledge and Data Engineering, 18*(3), 405-414.

Kim, Y. H., & Yoon, Y. (2009). A Note on Mathematical Modelling of Practical Multicampaign Assignment and Its Computational Complexity. [Article]. *CoRR, abs/0906.5475*.

Kim, Y. H., Yoon, Y., & Moon, B. R. (2008). A Lagrangian approach for multiple personalized campaigns. [Article]. *Ieee Transactions on Knowledge and Data Engineering, 20*(3), 383-396.

Lovelock, C., & Wirtz, J. (2006). Managing Relationships and Building Loyalty. *Services Marketing : People, Techonoly, Strategy* (sixth ed.). New Jersey: Prentice Hall.

Martello, S., & Toth, P. (1990). *Knapsack Problem - Algorithms and computer implementations*. Chichester: JOHN WILEY & SONS.

Nemhauser, G. L., & Wolsey, L. A. (1988). *Integer and Combinatorial Optimization*. New York: Wiley-Interscience.

Nobibon, F. T., Leus, R., & Spieksma, F. C. R. (2008). *Models for the optimization of promotion campaigns: Exact and heuristic algorithms*. Leuven, Belgium: Faculty of Business and Economics

O'Brien, R. (2001). An Overview of the Methodological Approach of Action Research. *Brazil*. Retrieved from http://www.web.ca/~robrien/papers/arfinal.html

Pinto, F. J. d. M. (2006). *A descoberta de conhecimento em bases de dados como suporte a actividades de business intelligence : Aplicação na área do Database Marketing.* Universidade do Minho.

Puchinger, J., Raidl, R. G., & Pferschy, U. (2006). The Core Concept for the Multidimensional Knapsack Problem *Evolutionary Computation in Combinatorial Optimizatio* (Vol. 3906/2006, pp. 195-208). Budapest: Springer Berlin / Heidelberg.

Rodrigues, R. d. F. (2000). NP_COMPLETUDE: Universidade do Amazonas, Departamento de Ciência da Computação.

Selby, D. A. (2007). Marketing event optimization. [Article]. *Ibm Journal of Research and Development, 51*(3-4), 409-419.

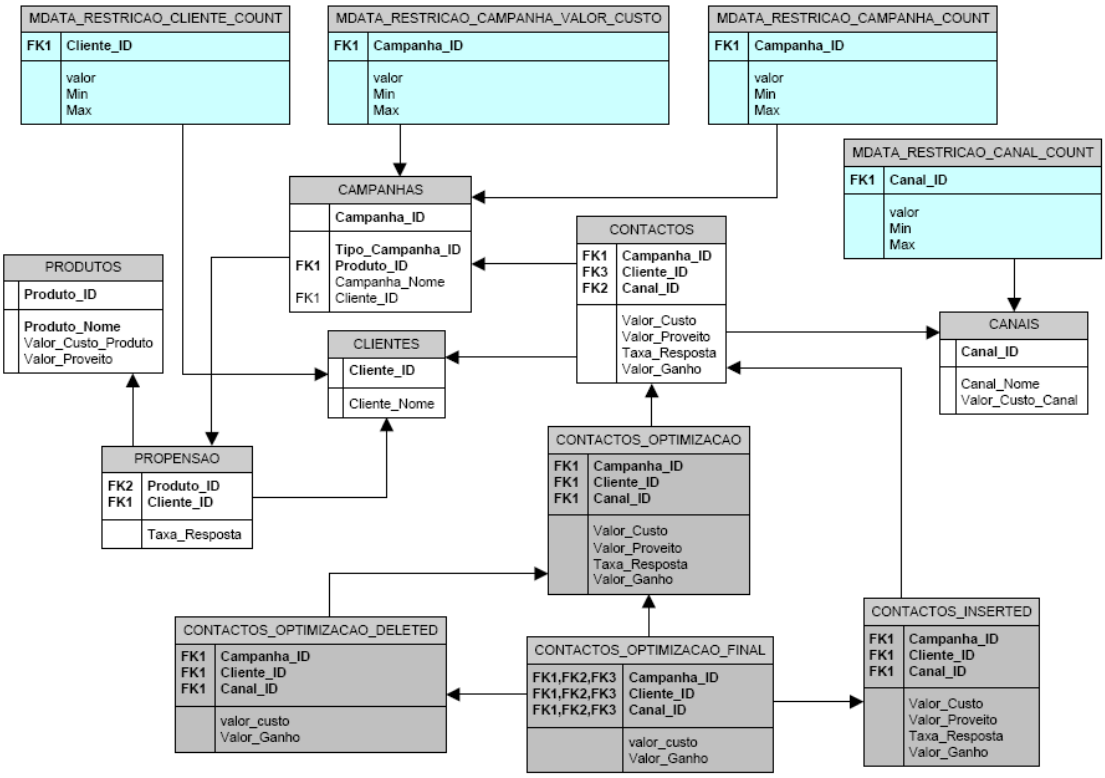
Stone, B., & Jacobs, R. (2001). *Successful Direct Marketing Methods* (Seventh ed.). New York: McGraw-Hill.

Stringer, E. T. (2007). *Action Research* (Third ed.). London: Sage.

Tomlab (2010-05-16). Tomlab Optimization Retrieved 2010-08-03, from http://tomopt.com/tomlab/

ANEXOS

1. Modelo de dados



2. Programa para o algoritmo 1

```
--Processo de Optimizao

DECLARE @Campanha_ID int

DECLARE @Cliente_ID int

DECLARE @Produto_ID int

DECLARE @Canal_ID int

DECLARE @Valor_Custo Decimal(18,2)

DECLARE @Valor_Proveito Decimal(18,2)

DECLARE @Taxa_Resposta Decimal(18,2)

DECLARE @Valor_Ganho Decimal(18,2)

SET NOCOUNT ON

Truncate table OFERTAS_OPTIMIZACAO

update MDATA_RESTRICAO_CANAL_COUNT set valor_count=0

update MDATA_RESTRICAO_CAMPANHA_COUNT set valor_count=0
```

```
update MDATA_RESTRICAO_PRODUTO_COUNT set valor_count=0

update MDATA_RESTRICAO_CAMPANHA_CLIENTE_COUNT set valor_count=0

DECLARE optimizacao CURSOR FOR

SELECT Campanha_ID,Cliente_ID ,Produto_ID ,Canal_ID,Valor_Custo,Valor_Proveito,Taxa_Resposta,Valor_Ganho

FROM OFERTAS

order by valor_ganho desc

OPEN optimizacao

FETCH NEXT FROM optimizacao

INTO @Campanha_ID,@Cliente_ID,@Produto_ID,@Canal_ID,@Valor_Custo,@Valor_Proveito,@Taxa_Resposta,@Valor_Ganho

WHILE @@FETCH_STATUS = 0

BEGIN

IF (dbo.FUNC_RESTRICAO_CAMPANHA_COUNT(@Campanha_ID)=1

        and dbo.FUNC_RESTRICAO_CANAL_COUNT(@Canal_ID)=1

        and dbo.FUNC_RESTRICAO_PRODUTO_COUNT(@Produto_ID)=1

        and dbo.FUNC_RESTRICAO_CAMPANHA_CLIENTE_COUNT(@Campanha_ID,@Cliente_ID)=1)

BEGIN

        Insert into OFERTAS_OPTIMIZACAO values(@Campanha_ID,@Cliente_ID,@Produto_ID,@Canal_ID)

        update MDATA_RESTRICAO_CANAL_COUNT set valor_count=valor_count+1 where canal_id=@Canal_ID

        update MDATA_RESTRICAO_CAMPANHA_COUNT set valor_count=valor_count+1 where campanha_id=@Campanha_ID

        update MDATA_RESTRICAO_PRODUTO_COUNT set valor_count = valor_count+1 where produto_id=@Produto_ID

        update MDATA_RESTRICAO_CAMPANHA_CLIENTE_COUNT set valor_count = valor_count+1 where
campanha_id=@Campanha_ID and cliente_id=@cliente_ID

END

FETCH NEXT FROM optimizacao

INTO @Campanha_ID,@Cliente_ID,@Produto_ID,@Canal_ID,@Valor_Custo,@Valor_Proveito,@Taxa_Resposta,@Valor_Ganho

END

CLOSE optimizacao

DEALLOCATE optimizacao
```

3. Programa para o algoritmo 2

```
Create procedure [dbo].[PROC_OPTIMIZACAO_CAMPANHA](@campanha_id int, @min int)
as
--Processo de Optimização para a campanha = @campanha_id
--preencher com o minimo de @min contactos
DECLARE @Cliente_ID int
DECLARE @Tipo_Campanha_ID int
DECLARE @Canal_ID int
DECLARE @valor_custo Decimal(18,2)
DECLARE @Valor_Proveito Decimal(18,2)
DECLARE @Taxa_Resposta Decimal(18,2)
DECLARE @Valor_Ganho Decimal(18,2)
DECLARE @count int
set @count = 0
SET NOCOUNT ON

if @count < @min
BEGIN
DECLARE optimizacao CURSOR scroll static FOR
SELECT Cliente_ID ,Tipo_Campanha_ID ,Canal_ID,valor_custo,Valor_Proveito,Taxa_Resposta,Valor_Ganho
FROM OFERTAS where Campanha_ID = @campanha_id
order by Valor_Ganho desc

OPEN optimizacao

FETCH NEXT FROM optimizacao
INTO @Cliente_ID,@Tipo_Campanha_ID,@Canal_ID,@valor_custo,@Valor_Proveito,@Taxa_Resposta,@Valor_Ganho

WHILE @@FETCH_STATUS = 0
BEGIN
        if @count >= @min
                FETCH LAST FROM optimizacao
                INTO
@Cliente_ID,@Tipo_Campanha_ID,@Canal_ID,@valor_custo,@Valor_Proveito,@Taxa_Resposta,@Valor_Ganho
        else
        BEGIN
        IF DBO.FUNC_RESTRICOES_VALIDAR(@campanha_id,@cliente_id,@Tipo_Campanha_ID,@canal_id,@valor_custo,1)=1
        BEGIN
                Insert into OFERTAS_OPTIMIZACAO
values(@Campanha_ID,@Cliente_ID,@Tipo_Campanha_ID,@Canal_ID,@valor_custo,@Valor_Ganho)
                Exec PROC_UPDATE_ESTATISTICAS
@campanha_id,@cliente_id,@Tipo_Campanha_ID,@canal_id,@valor_custo,1
                set @count = @count + 1
        END
        END
        FETCH NEXT FROM optimizacao
        INTO @Cliente_ID,@Tipo_Campanha_ID,@Canal_ID,@valor_custo,@Valor_Proveito,@Taxa_Resposta,@Valor_Ganho
END
CLOSE optimizacao
DEALLOCATE optimizacao

Delete OFERTAS from OFERTAS a INNER JOIN OFERTAS_OPTIMIZACAO b
on a.campanha_id=b.campanha_id and a.cliente_id=b.cliente_id

END
```

4. Programa para gerar cenários. Exemplo para o cenário 7.

```
USE TESE_BD
SET NOCOUNT ON

Declare @n int --Nº de clientes
Declare @m int --Nº de campanhas
set @n=1000
set @m=6
-----------------------------------------------------------------
--Gerar tabela de clientes com N clientes
-----------------------------------------------------------------
if exists(select name from sysobjects where name = 'TESTE07_CLIENTES' and xtype= 'U')
	Drop table TESTE07_CLIENTES

create table TESTE07_CLIENTES(Cliente_ID int, Cliente_Nome varchar(80))

Declare @i int
set @i=1
while @i <= @n
begin
	Insert into TESTE07_CLIENTES values(@i,'Cliente ' + convert(varchar(10),@i))
	set @i=@i+1
end
-----------------------------------------------------------------
--Gerar tabela de campanhas com M campanhas
-----------------------------------------------------------------
if exists(select name from sysobjects where name = 'TESTE07_CAMPANHAS' and xtype= 'U')
	drop table TESTE07_CAMPANHAS
create table TESTE07_CAMPANHAS(Campanha_ID int,Tipo_Campanha_ID int,Campanha_Nome varchar(80),Valor_Proveito
decimal(15,2))

Declare @j int
set @j=1
while @j <= @m
begin
	Insert into TESTE07_CAMPANHAS values(@j, -1, 'Campanha ' + convert(varchar(10),@j), -1)
	set @j=@j+1
end

DECLARE @U INT
DECLARE @L INT
SET @L = 1 -- The lowest random number
SET @U = 6 -- The highest random number
set rowcount 1
while exists(select top 1 * from TESTE07_CAMPANHAS where Tipo_Campanha_ID < 1)
begin
	update TESTE07_CAMPANHAS
	set Tipo_Campanha_ID=ROUND(((@U - @L -1) * RAND() + @L), 0)
	where Tipo_Campanha_ID < 1
End
set rowcount 0

DECLARE @Upper0 decimal(15,2)
DECLARE @Lower0 decimal(15,2)
SET @Lower0 = 200.00 ---- The lowest random number
SET @Upper0 = 600.00 ---- The highest random number
set rowcount 1
while exists(select * from TESTE07_CAMPANHAS where Valor_Proveito < 1)
begin
	update TESTE07_CAMPANHAS
	set Valor_Proveito = ((@Upper0 - @Lower0 -1) * RAND() + @Lower0)
	where Valor_Proveito < 1
end
set rowcount 0
-----------------------------------------------------------------
--Gerar ofertas (n*m)
-----------------------------------------------------------------
if exists(select name from sysobjects where name = 'TESTE07_OFERTAS' and xtype= 'U')
```

```
	Drop table TESTE07_OFERTAS
create table TESTE07_OFERTAS
	(Campanha_ID int,Tipo_Campanha_ID int,Cliente_ID int,Canal_ID int,
	Valor_Custo decimal(15,2),Valor_Proveito decimal(15,2),Taxa_Resposta decimal(15,2),Valor_Ganho decimal(15,2))

Insert into
TESTE07_OFERTAS(Campanha_ID,Tipo_Campanha_ID,Cliente_ID,Canal_ID,Valor_Custo,Valor_Proveito,Taxa_Resposta,Valor_Ganh
o)
select campanha_id,Tipo_Campanha_ID,cliente_id,-2 as canal_id,0,Valor_Proveito,-1,-1
from TESTE07_CLIENTES cross join TESTE07_CAMPANHAS
----------------------------------------------------------------
--Calcular as taxas de resposta
----------------------------------------------------------------
Declare @t int
set @t = (select count(*)/1000+1 from TESTE07_OFERTAS)
set rowcount @t
while exists(select top 1 * from TESTE07_OFERTAS where Taxa_Resposta<0)
begin
	update TESTE07_OFERTAS
	set Taxa_Resposta=rand((((cliente_id+1)) % 1000000) * (datepart(ms, GETDATE()) +1))
	where Taxa_Resposta<0
End

while exists(select top 1 * from TESTE07_OFERTAS where Taxa_Resposta=0)
begin
	update TESTE07_OFERTAS
	set Taxa_Resposta=rand((((cliente_id+1)) % 1000000) * (datepart(ms, GETDATE()) +1))
	where Taxa_Resposta=0
End
set rowcount 0

while exists(select top 1 * from TESTE07_OFERTAS where Taxa_Resposta<0.5)
begin
	update TESTE07_OFERTAS
	set Taxa_Resposta=Taxa_Resposta*2
	where Taxa_Resposta<0.5
End
----------------------------------------------------------------
--Afectar o Canal_ID
----------------------------------------------------------------
DECLARE @Upper INT
DECLARE @Lower INT
SET @Lower = 1 -- The lowest random number
SET @Upper = 6 -- The highest random number
Declare @t1 int
set @t1 = (select count(*)/1000+1 from TESTE07_OFERTAS)
set rowcount @t1
while exists(select top 1 * from TESTE07_OFERTAS where Canal_ID < 1)
begin
	update TESTE07_OFERTAS
	set Canal_ID=ROUND(((@Upper - @Lower -1) * RAND() + @Lower), 0) * 100
	where Canal_ID < 1
End
set rowcount 0
----------------------------------------------------------------
--Atribuir os Custos por campanha e canal
----------------------------------------------------------------
drop table #custos
select  Campanha_ID,canal_id,convert(decimal(18,2), -1.00) as valor_custo
into #custos
from TESTE07_OFERTAS group by Campanha_ID,canal_id order by 1,2,3

DECLARE @Upper1 decimal(15,2)
DECLARE @Lower1 decimal(15,2)
SET @Lower1 = 10.00 ---- The lowest random number
SET @Upper1 = 50.00 ---- The highest random number
set rowcount 1
while exists(select * from #custos where valor_custo<1)
begin
	update #custos
	set valor_custo = ((@Upper1 - @Lower1 -1) * RAND() + @Lower1)
```

```
		where valor_custo<1
end
set rowcount 0

update TESTE07_OFERTAS
set valor_custo = b.valor_custo
from TESTE07_OFERTAS a inner join #custos b
		on a.campanha_id=b.campanha_id and a.canal_id=b.canal_id
-----------------------------------------------------------------
Update TESTE07_OFERTAS
set valor_ganho = Valor_Proveito * Taxa_Resposta - Valor_Custo
-----------------------------------------------------------------
--Tabelas de restrições
-----------------------------------------------------------------
--Por campanha - COUNT
if exists(select name from sysobjects where name = 'TESTE07_MDATA_RESTRICAO_CAMPANHA_COUNT' and xtype= 'U')
		Drop table TESTE07_MDATA_RESTRICAO_CAMPANHA_COUNT

select Campanha_ID, 0 as valor, 0 as Min, convert(int, @n) as Max
into TESTE07_MDATA_RESTRICAO_CAMPANHA_COUNT from TESTE07_OFERTAS group by Campanha_ID

--Por campanha - SUM(custo)
if exists(select name from sysobjects where name = 'TESTE07_MDATA_RESTRICAO_CAMPANHA_VALOR_CUSTO' and xtype= 'U')
		Drop table TESTE07_MDATA_RESTRICAO_CAMPANHA_VALOR_CUSTO

select	Campanha_ID, convert(decimal(18,2), 0) as valor,
			convert(decimal(18,2), 0) as Min, convert(decimal(18,2), sum(valor_custo)/2.00) as Max
into TESTE07_MDATA_RESTRICAO_CAMPANHA_VALOR_CUSTO from TESTE07_OFERTAS group by Campanha_ID
--Por CANAL
if exists(select name from sysobjects where name = 'TESTE07_MDATA_RESTRICAO_CANAL_COUNT' and xtype= 'U')
		Drop table TESTE07_MDATA_RESTRICAO_CANAL_COUNT

select Canal_ID, 0 as valor, 0 as Min, @n/5 as Max
into TESTE07_MDATA_RESTRICAO_CANAL_COUNT from TESTE07_OFERTAS group by Canal_ID
--Por CLIENTE
if exists(select name from sysobjects where name = 'TESTE07_MDATA_RESTRICAO_CLIENTE_COUNT' and xtype= 'U')
		Drop table TESTE07_MDATA_RESTRICAO_CLIENTE_COUNT

select Cliente_ID, 0 as valor, 0 as Min, 1 as Max
into TESTE07_MDATA_RESTRICAO_CLIENTE_COUNT from TESTE07_CLIENTES
-----------------------------------------------------------------
```

5. Programa para gerar o ficheiro de MATLAB. Exemplo para o cenário 7.

```
-----------------------------------------------------------------
--Gerar Ficheiro de Input para o Matlab rotina BintProg
-----------------------------------------------------------------
drop table #prim_0
drop table #prim_1
drop table #prim_2
drop table #prim_3
drop table #prim_4

Declare @i as int --contador
Declare @campanha as int --campanha_id
Declare @n as int --nº de clientes
Declare @k as int --nº de canais
Declare @c as int --nº de campanhas
Declare @file as varchar(800)
Declare @cmd as varchar(8000)
Declare @texto as varchar(8000)
set @file = 'E:\Daniel\Mestrado ISEGI - 3º Semestre\TESE PROJECTO\TESE\MATLAB\INPUT_FILE_SQL07.m'

--Titulo do Ficheiro
set @texto = '%Ficheiro Matlab'
```

```
set @cmd='echo '+@texto+' > "'+@file+'""'
Exec master..xp_cmdshell @cmd,no_output

--Limpar variáveis
set @texto = 'clear all;'
set @cmd='echo '+@texto+' >> "'+@file+'""'
Exec master..xp_cmdshell @cmd,no_output

--Definir o vector da função a maximizar
set @texto = 'f=['
select Campanha_ID,Cliente_ID,Tipo_Campanha_ID,Canal_ID,valor_custo,Valor_Ganho
into #prim_0 from TESTE07_OFERTAS where 0=1

while exists(select top 1 * from TESTE07_OFERTAS a left join #prim_0 b
                                        on a.campanha_id=b.campanha_id and a.cliente_id=b.cliente_id and a.canal_id=b.canal_id
                                        where b.campanha_id is null)
begin

select top 500 @texto = @texto + convert(varchar(10), a.valor_ganho*-1)+' '
from TESTE07_OFERTAS a left join #prim_0 b
on a.campanha_id=b.campanha_id and a.cliente_id=b.cliente_id and a.canal_id=b.canal_id
        where b.campanha_id is null
order by a.campanha_id, a.cliente_id, a.canal_id

Insert into #prim_0
select top 500 a.Campanha_ID,a.Cliente_ID,a.Tipo_Campanha_ID,a.Canal_ID,a.valor_custo,a.Valor_Ganho
from TESTE07_OFERTAS a left join #prim_0 b
on a.campanha_id=b.campanha_id and a.cliente_id=b.cliente_id and a.canal_id=b.canal_id
        where b.campanha_id is null
order by a.campanha_id, a.cliente_id, a.canal_id

if exists(select top 1 * from TESTE07_OFERTAS a left join #prim_0 b
                                        on a.campanha_id=b.campanha_id and a.cliente_id=b.cliente_id and a.canal_id=b.canal_id
                                        where b.campanha_id is null)
        select @texto = @texto + '...'
else
        select @texto = rtrim(@texto) + '];'

set @cmd='echo '+@texto+' >> "'+@file+'""'
Exec master..xp_cmdshell @cmd,no_output

set @texto = ''

End

--Definir a matriz das restrições
--Restrição 1 - Cada cliente apenas pode estar em uma campanha
--Uma linha da matriz para cada cliente
set @texto = 'A=['
set @i = 1
set @n = (select count(distinct cliente_id) from TESTE07_OFERTAS)
while @i <= @n
begin
        select top 3000 @texto=@texto+case when cliente_id=@i then '1 ' else '0 ' End
        from TESTE07_OFERTAS order by campanha_id, cliente_id, canal_id

        select top 3000 * into #prim_1 from TESTE07_OFERTAS order by campanha_id, cliente_id, canal_id

        set @texto=@texto+'...'
        set @cmd='echo '+@texto+' >> "'+@file+'""'
        Exec master..xp_cmdshell @cmd,no_output

        set @texto=''

        select top 3000 @texto=@texto+case when a.cliente_id=@i then '1 ' else '0 ' End
        from TESTE07_OFERTAS a left join #prim_1 b
                on a.campanha_id=b.campanha_id and a.cliente_id=b.cliente_id and a.canal_id=b.canal_id
        where b.campanha_id is null
        order by a.campanha_id, a.cliente_id, a.canal_id

        set @texto=@texto+';'
```

```
		set @cmd='echo '+@texto+' >> "'+@file+'"'
		Exec master..xp_cmdshell @cmd,no_output

		set @i=@i+1
		set @texto=''

		drop table #prim_1
end

--Restrição 2 - Cada canal apenas pode ter o nº max de contactos definidos
--Uma linha da matriz para cada canal
set @i = 1
set @k = (select count(distinct canal_id) from TESTE07_OFERTAS)
while @i <= @k
begin
		if exists(select top 1 * from TESTE07_OFERTAS where canal_id=@i*100)
		begin
			select top 3000 @texto=@texto+case when canal_id=@i*100 then '1 ' else '0 ' End
			from TESTE07_OFERTAS order by campanha_id, cliente_id, canal_id

			select top 3000 * into #prim_2 from TESTE07_OFERTAS order by campanha_id, cliente_id, canal_id

			set @texto=@texto+'...'
			set @cmd='echo '+@texto+' >> "'+@file+'"'
			Exec master..xp_cmdshell @cmd,no_output

			set @texto=''

			select top 3000 @texto=@texto+case when a.canal_id=@i*100 then '1 ' else '0 ' End
			from TESTE07_OFERTAS a left join #prim_2 b
				on a.campanha_id=b.campanha_id and a.cliente_id=b.cliente_id and a.canal_id=b.canal_id
			where b.campanha_id is null
			order by a.campanha_id, a.cliente_id, a.canal_id

			set @texto=@texto+';'
			set @cmd='echo '+@texto+' >> "'+@file+'"'
			Exec master..xp_cmdshell @cmd,no_output
		end

		set @i=@i+1
		set @texto=''

		drop table #prim_2
end

--Restrição 3 - Cada campanha apenas pode ser gasto o orçamento definido
--Uma linha da matriz para cada campanha
set @i = 1
set @c = (select count(distinct campanha_id) from TESTE07_OFERTAS)
while @i <= @c
begin
		select a.Campanha_ID,a.Cliente_ID,a.Tipo_Campanha_ID,a.Canal_ID,a.valor_custo,a.Valor_Ganho
		into #prim_3 from TESTE07_OFERTAS a where 0=1

		while exists(select top 1 * from TESTE07_OFERTAS a left join #prim_3 b
					on a.campanha_id=b.campanha_id and a.cliente_id=b.cliente_id and
a.canal_id=b.canal_id
					where b.campanha_id is null)
		begin

			select top 500 @texto=@texto+case when a.Campanha_ID=@i then convert(varchar(10),a.Valor_Custo)+' '
else '0 ' End
			from TESTE07_OFERTAS a left join #prim_3 b
				on a.campanha_id=b.campanha_id and a.cliente_id=b.cliente_id and a.canal_id=b.canal_id
			where b.campanha_id is null order by a.campanha_id, a.cliente_id, a.canal_id

			Insert into #prim_3
			select top 500 a.Campanha_ID,a.Cliente_ID,a.Tipo_Campanha_ID,a.Canal_ID,a.valor_custo,a.Valor_Ganho
			from TESTE07_OFERTAS a left join #prim_3 b
			on a.campanha_id=b.campanha_id and a.cliente_id=b.cliente_id and a.canal_id=b.canal_id
				where b.campanha_id is null
```

```
		order by a.campanha_id, a.cliente_id, a.canal_id

		if exists(select top 1 * from TESTE07_OFERTAS a left join #prim_3 b
					on a.campanha_id=b.campanha_id and a.cliente_id=b.cliente_id and
a.canal_id=b.canal_id
					where b.campanha_id is null)
			select @texto = @texto + '...'
		else
			if @i = @c and (select count(distinct campanha_id) from
TESTE07_MDATA_RESTRICAO_CAMPANHA_COUNT where min>0) = 0
				select @texto = rtrim(@texto) + '];'
			else
				select @texto = @texto + ';'

		set @cmd='echo '+@texto+' >> "'+@file+'"'
		Exec master..xp_cmdshell @cmd,no_output

		set @texto=''
	end

	set @i=@i+1
	set @texto=''

	drop table #prim_3
end

--Restrição 4 - Para cada campanha garantir um minimo de contactos para a campanha
--Uma linha da matriz para cada campanha
set @i = 1
set @campanha = 0
set @c = (select count(distinct campanha_id) from TESTE07_MDATA_RESTRICAO_CAMPANHA_COUNT where min>0)
while @i <= @c
begin
	set @campanha = (select top 1 campanha_id from TESTE07_MDATA_RESTRICAO_CAMPANHA_COUNT where min>0
and campanha_id>@campanha order by campanha_id)
	select * into #prim_4 from TESTE07_OFERTAS where 0=1

	while exists(select top 1 * from TESTE07_OFERTAS a left join #prim_4 b
				on a.campanha_id=b.campanha_id and a.cliente_id=b.cliente_id and
a.canal_id=b.canal_id
				where b.campanha_id is null)
	begin

		select top 500 @texto=@texto+case when a.Campanha_ID=@campanha then '-1 ' else '0 ' End
		from TESTE07_OFERTAS a left join #prim_4 b
			on a.campanha_id=b.campanha_id and a.cliente_id=b.cliente_id and a.canal_id=b.canal_id
		where b.campanha_id is null order by a.campanha_id, a.cliente_id, a.canal_id

		Insert into #prim_4
		select top 500 a.* from TESTE07_OFERTAS a left join #prim_4 b
		on a.campanha_id=b.campanha_id and a.cliente_id=b.cliente_id and a.canal_id=b.canal_id
			where b.campanha_id is null
		order by a.campanha_id, a.cliente_id, a.canal_id

		if exists(select top 1 * from TESTE07_OFERTAS a left join #prim_4 b
					on a.campanha_id=b.campanha_id and a.cliente_id=b.cliente_id and
a.canal_id=b.canal_id
					where b.campanha_id is null)
			select @texto = @texto + '...'
		else
			if @i = @c select @texto = rtrim(@texto) + '];' else select @texto = @texto + ';'

		set @cmd='echo '+@texto+' >> "'+@file+'"'
		Exec master..xp_cmdshell @cmd,no_output

		set @texto=''
	end

	set @i=@i+1
	set @texto=''
```

```
		drop table #prim_4
end

--Definir o vector das restrições
--Restrição 1 - Cada cliente apenas pode estar em uma campanha
--Uma linha da matriz para cada cliente
set @texto = 'b=['
select @texto=@texto+convert(varchar(10),Max)+';'
from TESTE07_MDATA_RESTRICAO_CLIENTE_COUNT order by cliente_id

--Restrição 2 - Cada canal apenas pode ter o nº max de contactos definidos
--Uma linha da matriz para cada canal
select @texto=@texto+convert(varchar(10),Max)+';'
from TESTE07_MDATA_RESTRICAO_CANAL_COUNT
where canal_id in (select distinct canal_id from TESTE07_OFERTAS)
order by canal_id

--Restrição 3 - Cada campanha apenas pode ser gasto o orçamento definido
--Uma linha da matriz para cada campanha
select @texto=@texto+convert(varchar(10),Max)+';'
from TESTE07_MDATA_RESTRICAO_CAMPANHA_VALOR_CUSTO
order by campanha_id

--Restrição 4 - Para cada campanha garantir um minimo de contactos a efectuar
--Uma linha da matriz para cada campanha
select @texto=@texto+'-'+convert(varchar(10),Min)+';'
from TESTE07_MDATA_RESTRICAO_CAMPANHA_COUNT where min>0
order by campanha_id

set		@texto=left(@texto,len(@texto)-1)
set @cmd='echo '+@texto+']; >> "'+@file+'"'
Exec master..xp_cmdshell @cmd,no_output

--
set @texto = 'A=sparse(A);'
set @cmd='echo '+@texto+' >> "'+@file+'"'
Exec master..xp_cmdshell @cmd,no_output
--
set @texto = 'tic;'
set @cmd='echo '+@texto+' >> "'+@file+'"'
Exec master..xp_cmdshell @cmd,no_output
--
set @texto = '[x,fval] = bintprog(f,A,b);'
set @cmd='echo '+@texto+' >> "'+@file+'"'
Exec master..xp_cmdshell @cmd,no_output
--
set @texto = 'toc;'
set @cmd='echo '+@texto+' >> "'+@file+'"'
Exec master..xp_cmdshell @cmd,no_output
--
set @texto = 'fprintf(''%g \n'',fval*-1)'
set @cmd='echo '+@texto+' >> "'+@file+'"'
Exec master..xp_cmdshell @cmd,no_output
```