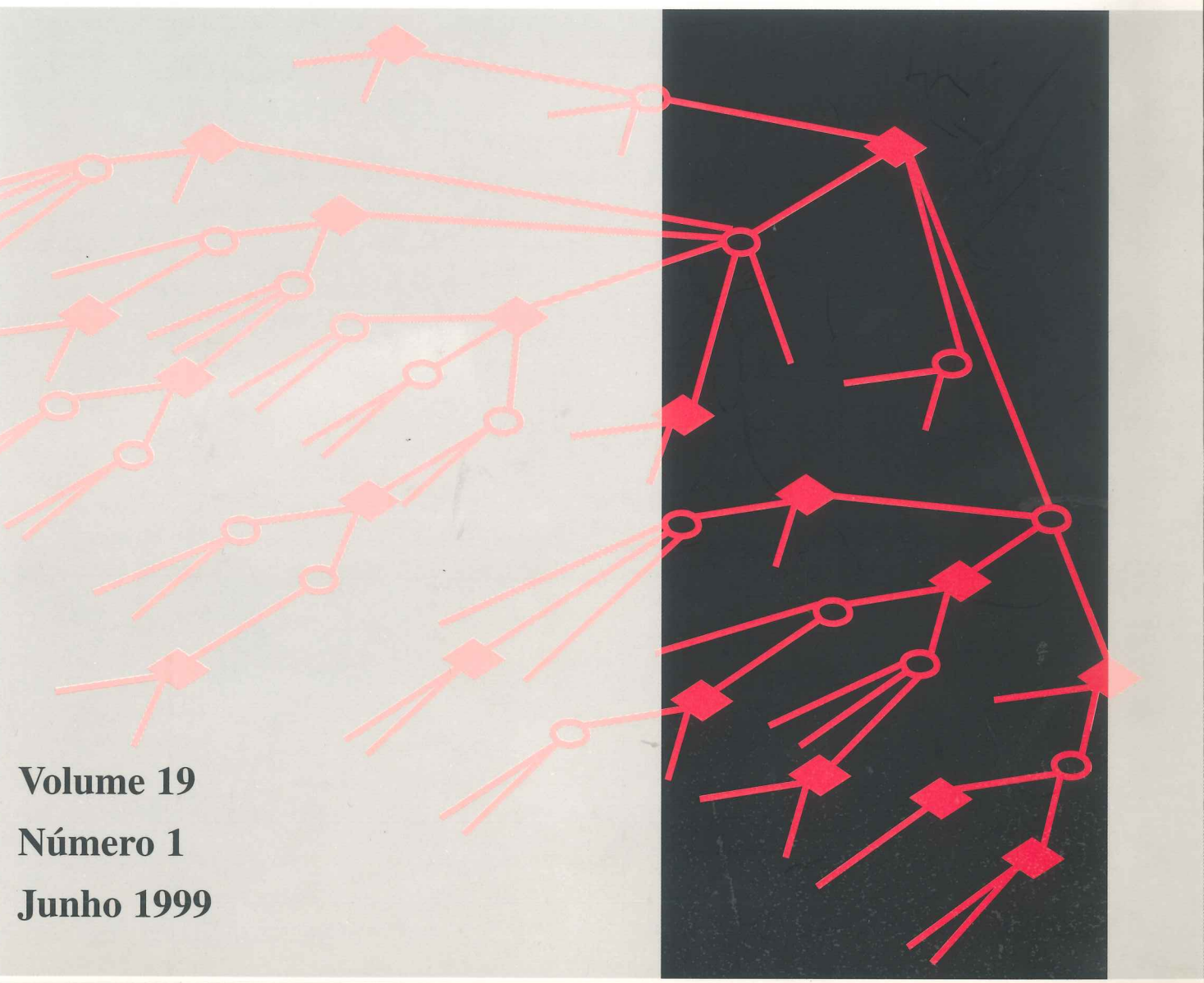
ISSN: 0874-5161

Investigação Operacional



Volume 19
Número 1
Junho 1999

INVESTIGAÇÃO OPERACIONAL

Propriedade:
APDIO — Associação Portuguesa de Investigação Operacional

ESTATUTO EDITORIAL

<<Investigação Operacional>>, órgão oficial da APDIO cobre uma larga gama de assuntos reflectindo assim a grande diversidade de profissões e interesses dos sócios da Associação, bem como as muitas áreas de aplicação da I. O. O seu objectivo primordial é promover a aplicação do método e técnicas da I.O. aos problemas da Sociedade Portuguesa.

A publicação acolhe contribuições nos campos da metodologia, técnicas, e áreas de aplicação e software de I. O. sendo no entanto dada prioridade a bons casos de estudo de carácter eminentemente prático.

Patrocinadores

BP PORTUGUESA

FCT
Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA CIÊNCIA E DA TECNOLOGIA

Fundação Calouste Gulbenkian

ISSN nº 0874-5161
Dep. Legal nº 130 761 / 98
Execução Gráfica: J. F. Macedo - Astrofoto
700 Ex.
99/06

INVESTIGAÇÃO OPERACIONAL

Volume 19 - nº 1 - Junho 1999 Publicação Semestral

A Revista "INVESTIGAÇÃO OPERACIONAL" está registada na Secretaria de Estado da Comunicação Social sob o nº 108335.
Esta Revista é distribuida gratuitamente aos sócios da APDIO. As informações sobre inscrições na Associação, assim como a correspondência para a Revista devem ser enviadas para a sede da APDIO - Associação Portuguesa de Investigação Operacional - CESUR, Instituto Superior Técnico, Av. Rovisco Pais, 1049 - 001 Lisboa.

Este Volume foi subsidiado por:

Fundação para a Ciência e Tecnologia

Fundação Calouste Gulbenkian

BP - Portuguesa

Para efeitos de dactilografia e composição, foram utilizados equipamentos gentilmente postos à disposição pelo Centro de Investigação Operacional (DEIO - FCUL).

Assinatura: 5.000$00

OPTIMIZAÇÃO DA DIVISÃO DE CORRESPONDÊNCIA NOS CENTROS DE TRATAMENTO DE CORREIO

Jacinto Maurício Nunes

CTT - Correios de Portugal, SA
Praça D. Luís I, 30
1208-148 Lisboa

Maria Teresa Almeida

Departamento de Matemática
Instituto Superior de Economia e Gestão (UTL)
Rua do Quelhas, 6
1200-781 Lisboa

Abstract

In this paper three approaches are proposed to generate schedules for the letter sorting machines at CTT-Correios de Portugal SA.

The first approach is based on a single machine scheduling model with no setup times. A constructing algorithm and some heuristic procedures were developed to solve the model.

The second model includes setup times, making the total time dependent on the job processing sequence. It is based on a typical vehicle routing problem with time windows and solved using a comercial package.

The third approach tackles the set of sorting machines as a whole, generating simultaneously the schedules for all machines. It is based on an assignment problem for which comercial software is also available.

Computational results are presented for real data gathered at the Lisbon CTC.

Resumo

Neste trabalho apresentam-se três modelizações para o problema da optimização da alimentação das máquinas de divisão de correio, instaladas pelos CTT nos seus Centros de Tratamento de Correio.

O primeiro modelo tem por base um problema típico de sequenciamento de tarefas numa máquina e para ele foi desenvolvido um algoritmo de tipo construtivo e alguns procedimentos heurísticos para gerar as sequências de lotes de cartas encaminhados para cada máquina, não incluindo tempos de *setup*.

O segundo modelo permite incluir tempos de *setup*, o que torna os tempos totais de processamento dependentes da ordem de divisão dos lotes. Tem por base um problema de optimização de rotas e foi resolvido recorrendo a *software* comercial.

O terceiro modelo destina-se a permitir a gestão integrada do conjunto das máquinas instaladas nos três Centros e baseia-se num problema de afectação, para o qual também existe *software* comercial.

São apresentados resultados computacionais para dados reais do Centro de Tratamento de Correio de Lisboa.

Keywords

Job scheduling, Vehicle routing with time-windows, assignment problem.

1. Introdução

O objectivo deste trabalho foi contribuir para a resolução de um problema operacional surgido na empresa CTT - Correios de Portugal SA, especificamente no subsistema de tratamento automático de correspondências, consequente de uma reformulação de processos e reequipamento que decorreu em simultaneidade com a realização deste estudo e que ainda se desenvolve.

A existência de cerca de mil estações de correios dispersas por todo o país, de perto de 16 mil trabalhadores, de um pouco mais de mil milhões de objectos movimentados por ano e de proveitos e custos operacionais anuais na ordem dos 96 e dos 88 milhões de contos, respectivamente, dão uma ideia da importância da empresa para a economia nacional (indicadores de 1997) [3].

Do ponto de vista da actividade postal, Portugal está dividido, num primeiro nível, em regiões com limites geográficos inspirados na delimitação dos concelhos do país, demarcando a área de actividade de um chamado Centro de Distribuição Postal (CDP).

O local de instalação do CDP é, com poucas excepções, início e fim do trajecto interno dos objectos confiados aos CTT. É da sede do CDP que partem os carteiros para efectuarem os "giros", isto é, os percursos diários de entrega de correio porta-a-porta aos destinatários da área; e inversamente, na recolha, os objectos recebidos dentro dos limites de um CDP, são para lá conduzidos como primeira etapa do processo "produtivo".

Agregando conjuntos de CDP existe um segundo nível da rede postal, menos visível para o utente, formado pelos Centros de Tratamento de Correio (CTC), onde é realizada a triagem, ou divisão, massiva dos objectos originados nos CDP da sua área de influência, e que depois os expede para os CDP de destino.

O processo postal, no que se reveste de importância para este trabalho, pode ser esquematizado nas funções: aceitação, triagem por destinos, transporte e distribuição.

A aceitação é a fase de recolha de objectos, efectuada na área de influência de cada CDP (origem), depositados em marcos e caixas, aos balcões das estações, ou recebidos em casa do cliente.

A triagem começa a fazer-se, manualmente, nos CDP origem com a disjunção dos que têm remetente e destinatário no próprio CDP, com os que têm destino noutro CDP e que são enviados para tratamento no CTC a que o CDP origem está ligado.

Nos CTC executa-se, essencialmente, a divisão de grandes volumes de tráfego; mas podem ser também pontos de trânsito para concentração e reenvio para outros CTC (onde se desenrolará o processo de divisão).

Os transportes têm como função ligar os CDP aos CTC no envio de objectos a tratar, ou o caminho inverso quando estes estão já triados. Suportam igualmente a movimentação inter-CTC e, nas cidades onde estão implantados CTC, fazem ligações directas, entre estações, marcos, caixas e grandes clientes, e respectivos Centros de Tratamento de Correio.

A distribuição é o último elo na cadeia. Inicia-se quando concluída a separação por destinos e os objectos já estão no respectivo CDP. Antes da saída para a distribuição propriamente dita, é necessário que estejam colocados no "giro" que os vai entregar e numa sequência, em cada artéria, de acordo com o percurso dos carteiros.

Até 1978 todo o processamento de divisão dos objectos postais era inteiramente manual. Nesse ano, foi introduzido um código postal de 4 dígitos, identificando as áreas postais já referidas como CDP (cerca de 400). Passou a figurar como elemento fundamental do endereço, viabilizando a divisão mecanizada e trazendo benefícios importantes à função distribuição.

Paralelamente foram construídos 3 Centros de Tratamento de Correio - Lisboa, Coimbra e Porto - onde foram instalados equipamentos para processamento das correspondências.

Os sistemas então disponíveis que foram adquiridos e que se mantêm activos, não lêem os dígitos do código postal, exigindo uma prévia transcrição, a realizar "manualmente" (dado o estado tecnológico, à data da sua compra), por operadores com a função de ler o código postal correspondente ao CDP do destinatário e digitá-lo num teclado. Como resultado, obtém-se um código de barras, colocado no canto inferior direito das correspondências, legível pelas máquinas divisoras que possibilita a triagem por essa unidade postal.

Dado este nível de separação ser ainda demasiado agregado (área coberta por um CDP), o tratamento tem que ser prosseguido com o correio após transporte para o CDP destino, sujeito a operações manuais de divisão e sequenciamento até estar em condições de entrega, pelos carteiros, aos destinatários.

É precisamente aqui, na forma como chegam os objectos aos CDP vindos dos CTC, que reside uma nova oportunidade de incrementar, substancialmente, a produtividade da cadeia de tratamento: eliminando estas tarefas manuais de separação que levam a classificação do correio até subdivisões do CDP nas suas "unidades constituintes" - os referidos "giros". É possível com novos equipamentos levar a divisão até ao nível do "giro" para os CDP de maiores dimensões, equivalendo a passar dos presentes 400 para perto de 3000 códigos de destino (que incluem CDP e cerca de 45% de "giros" de todo o país).

Para poder viabilizar esta maior fragmentação, o código postal será ampliado de 4 para 7 dígitos, para poder identificar CDP+giro, e instalados leitores ópticos (*OCR - Optical Character Reader*), que consigam ler directamente os novos códigos postais, substituindo os operadores na sua tradução para um código de barras interpretável pelas divisoras. Os leitores ópticos além destas funções, realizam em simultâneo uma pré-triagem formando lotes de cartas (onde são agrupados vários destinos), nas múltiplas saídas que possuem e de onde é feita a alimentação das máquinas divisoras (na indexação manual também é executada uma pré-divisão, embora com muito menos saídas).

A percentagem de cartas que é possível distribuir nos CDP destino (após tratamento num CTC), no dia seguinte aquele em que são depositadas na área dos CDP origem (cartas processadas num intervalo de tempo dito igual a D+1), deve muito à forma como os lotes,

"produzidos" nas saídas dos leitores ópticos, são escalonados nas divisoras. Como o número de dias que medeia entre a aceitação dos objectos postais pelos CTT e a entrega ao destinatário é um dos indicadores mais importantes para aferir a qualidade do serviço prestado, é do maior interesse atingir o máximo de cartas que é possível processar em D+1. Esse, o modo como deve ser feita a alimentação de cartas nas máquinas divisoras com o propósito de triar o maior número em D+1, foi o objectivo central do estudo que se apresenta em seguida.

Na secção 2 é feita uma descrição resumida do problema em estudo. Na secção 3 são apresentados os modelos e algoritmos desenvolvidos para a resolução do problema. Na secção 4 são apresentados os resultados computacionais obtidos para o CTC de Lisboa e na secção 5 são feitas as considerações finais.

2. Descrição do Problema

Para os propósitos pretendidos, as operações dentro de um CTC que asseguram a triagem por destinos finais das correspondências esquematizam-se no que se segue:

Recepção dos objectos

A maior concentração de tráfego, encaminhado para um CTC e originado em CDP da sua área de influência, ocorre entre as 17H:30M e as 20H:30M. No entanto, a chegada de objectos com proveniência directa de estações, marcos, caixas e clientes de grande dimensão (com várias tiragens ao longo do dia), e as ligações inter-CTC sustentam um fluxo quase contínuo.

Preparação e separação por tipos

Parte do correio, nomeadamente o depositado nos marcos e caixas, tem volumetria e espessura diversa, isto é, recebe-se uma mistura de objectos "finos", "médios" e "grossos".

Para isolar cada categoria, existem equipamentos mecânicos, que se podem descrever como cilindros rotativos com ranhuras actuando como "filtros", que deixam, ou não, passar cada um dos tipos enumerados.

Obtém-se daqui a classe "finos" (genericamente, correspondências ou "cartas"), individualizada das restantes. Uma pequena curiosidade que se acrescenta, é que é também função destas máquinas deixar as "cartas faceadas" (à saída todas as cartas têm a mesma orientação em relação ao endereço e posição do selo) e obliteradas (com o selo carimbado).

Separação automática de correspondências

- Indexação

Ultrapassadas as duas etapas anteriores atinge-se a chamada indexação.

Basicamente, esta operação é a tradução de um código de destino escrito no endereço num código de barras verticais a colocar no canto inferior direito das correspondências.

Com a entrada em funcionamento dos *OCR* (3 em Lisboa, 2 em Coimbra e 2 no Porto) é a indexação que sofre uma alteração qualitativa, embora a tarefa não se transforme em totalmente automática. Objectos que sejam rejeitados pelos *OCR* são conduzidos para postos de indexação manual (onde se pode colocar apenas o código do CDP).

Como resultado, obtêm-se dois grupos de objectos, onde num se pode levar a sua classificação automática até muito maior individualização, ao passo que no outro há maior exigência de recursos humanos para conseguir a mesma diferenciação (quando as cartas chegarem ao CDP destino).

Nas indexações manual e automática existe, como se disse, simultaneamente uma pré-divisão que distribui os objectos por diversas saídas, formando classes de destinos, e cuja utilidade se pormenorizará adiante.

- Divisão

Feita a indexação, o correio está apto a ser levado para os equipamentos de divisão mecânica (duas divisoras em Lisboa e uma em Coimbra e outra no Porto).

Como a chegada dos objectos postais aos CTC se dá em fluxo quase contínuo ao longo do dia a formação dos lotes de cartas nos leitores ópticos vai sendo também feita ao longo do dia. Para cada lote o momento de saída do leitor óptico determina o momento a partir do qual ele se encontra disponível para se iniciar a sua divisão.

As máquinas de divisão são equipamentos computorizados que comparam os códigos de barras com os registos de um ficheiro carregado na sua memória *RAM* e deslocam as cartas para os cacifos de saída correspondentes. Dispõem de 220 cacifos, o que significa que apenas podem separar em simultâneo 220 destinos. Ora como se pretende individualizar cerca de 3000 e como, por razões de gestão, cada carta passa apenas uma vez pelas divisoras, torna-se clara a necessidade da pré-divisão para garantir que qualquer lote, formado em cada saída dos leitores ópticos na fase de indexação, não inclua mais de 220 destinos distintos. Fazendo a alimentação das divisoras lote a lote, a partir das saídas dos leitores ópticos, consegue-se a separação da totalidade de destinos.

Mais dois detalhes operacionais merecem destaque para a compreensão do problema: de cada vez que se muda a saída do leitor óptico que alimenta uma divisora é necessário "carregar" o correspondente ficheiro na memória desta e proceder ao esvaziamento da totalidade dos seus cacifos antes de passar lotes da nova saída dos leitores ópticos. O tempo necessário para realizar estas duas operações pode ser interpretado como um tempo de *setup* das máquinas de divisão.

A separação de um destino fica concluída quando termina o processamento na máquina de divisão do lote em que ele se integra.

Expedição

Completada a divisão, os objectos são encaminhados para o cais do CTC para serem levados para os vários CDP de destino.

O estabelecimento de horas de começo dos "giros" de distribuição, obriga a horas limite de partida dos veículos de transporte.

Para cada destino é conhecida *a priori* a "hora de corte", definida como o momento mais tarde de conclusão da sua triagem na máquina de divisão. Esta hora é fixada em função dos tempos de percurso entre o CTC e o CDP e do momento até ao qual é necessário ter as cartas

disponíveis no CDP de destino para iniciar o planeamento da distribuição domiciliária em cada dia.

Para que as cartas de um lote estejam todas em condições de ser transportadas para o seu CDP de destino é necessário que a conclusão da triagem do lote não seja posterior à hora de corte do destino com menor hora de corte que pertença a esse lote.

Decorrente do exposto são facilmente identificáveis dois problemas a nível de cada CTC.

O primeiro, que não se irá abordar mas condicionante do segundo, prende-se com o estabelecimento dos agrupamentos de destinos nas saídas dos *OCR*. O segundo com o escalonamento destes agrupamentos de cartas que saem dos *OCR*, e da indexação manual, e que terão que passar pelas divisoras.

Escalonamento nas máquinas de divisão

Suponha-se resolvido o primeiro problema, isto é, estão definidos os agrupamentos de destinos em cada saída dos leitores ópticos. Consequentemente estão definidas as horas de corte e, a partir do diagrama de carga horária de tráfego recebido, as "horas de entrada nas divisoras" para cada lote indexado, cujos destinos que o constituem têm que ser separados.

É daqui sugestiva a necessidade de determinar um escalonamento para a passagem nas divisoras dos diversos lotes, com dois factores determinantes na modelização do problema:

- O cumprimento das horas de corte;
- A chegada de correio à divisão.

Sendo, relembra-se, um processo quase contínuo, a decisão de "passar" um lote nas máquinas de divisão dentro de uma janela de tempo é condicionada pela disponibilidade dos objectos.

Explicitados os aspectos que configuram o fundamental da forma de escalonamento, poderá, complementarmente, enriquecer-se o modelo com a introdução de dois novos parâmetros:

- Prioridades a atribuir a cada grupo (ou saída dos *OCR*);
- Tempo de preparação das divisoras para mudança de grupo.

Fazendo agora a conclusão do anterior, pode formular-se este problema no seguinte modo:

- Organizar a alimentação das máquinas de divisão, isto é determinar a sequência e o período de passagem dos lotes de cartas provenientes da indexação, de modo a minimizar o número dos não processados dentro das horas de corte.

O objectivo é minimizar o número de lotes de cartas separados após as correspondentes horas de corte.

É importante esclarecer aqui que o genuíno objectivo seria minimizar o número de cartas (e não lotes) não tratadas até à sua hora de corte; mas tal implicaria a identificação prévia de cada carta à saída dos *OCR*, para se conhecer a sua hora de corte. Para isso, tornar-se-ia indispensável individualizar as cartas antes do processo de triagem, o que seria absurdo.

O que se deverá assegurar é que a variabilidade das horas de corte dos destinos que integram cada lote seja tão pequena quanto possível.

3. Métodos de Resolução

Considerando que a divisão de cada lote não pode ser interrompida (*nopreemption*) e que os tempos de *setup* das máquinas de divisão (associados à mudança de saída dos leitores ópticos) são desprezáveis, o problema pode ser modelizado como um problema típico de escalonamento da tarefas numa máquina e resolvido de forma construtiva.

Pelo contrário, se os tempos de *setup* forem considerados não negligenciáveis, o tempo total necessário para dividir um conjunto de lotes é também função da respectiva sequência de entrada na máquina divisora (pois a alimentação pode ser feita, ou não, consecutivamente por lotes da mesma saída dos *OCR*). Este tipo de problemas é mais difícil de resolver e, como é salientado em [7], está intimamente relacionado com os problemas de optimização de rotas.

Se além do planeamento da divisão do correio em cada CTC se pretender gerir o conjunto de todas as máquinas de forma integrada pode desenvolver-se um modelo para o conjunto dos CTC baseado num problema de afectação.

Nas secções seguintes são apresentadas as abordagens desenvolvidas para as três alternativas.

3. 1 A alimentação das divisoras como um problema de escalonamento de tarefas

Os problemas de escalonamento de tarefas em máquinas constituem uma vastíssima classe de problemas que têm sido objecto de intensa investigação científica nas últimas décadas, sem que tenha sido possível desenvolver para a esmagadora maioria dos casos algoritmos que garantam a obtenção da solução óptima em tempo polinomial [2],[7]. Uma das poucas excepções é o problema $1 \parallel \Sigma_j U_j$.

3. 1. 1 Problema $1 \parallel \Sigma_j U_j$ e algoritmo de Moore-Hodgson

Considere-se o problema da determinação do escalonamento de n tarefas, $\{J_1, J_2, \ldots, J_n\}$, numa única máquina que minimiza o número de tarefas concluídas com atraso, admitindo que são verificadas as seguintes condições: todas as tarefas estão disponíveis para iniciar o processamento no mesmo instante (*ready date*), para cada tarefa J_j ($1 \leq j \leq n$) são conhecidos a priori o seu tempo de processamento, p_j, e o seu prazo de conclusão d_j (*due date*) e não é possível interromper a execução das tarefas (*nopreemption*). Associando a cada tarefa J_j ($1 \leq j \leq n$) uma variável binária, U_j, que assume o valor 1 se a tarefa é concluída com atraso e 0 caso contrário, o problema pode ser representado por $1 \parallel \Sigma_j U_j$, usando a notação habitual para os problemas de escalonamento. Para este caso a solução pode ser gerada em tempo polinomial de forma construtiva executando o seguinte algoritmo ([5]):

Algoritmo de Moore-Hodgson

Passo 0

Ordenar todas as tarefas por ordem crescente dos respectivos prazos de conclusão formando a sequência inicial $S = (J_{u1}, J_{u2},\ldots, J_{un})$, $d_{u1} \leq d_{u2} \leq \ldots \leq d_{un}$.

Passo 1

Se na sequência corrente nenhuma tarefa é concluída com atraso ir para o Passo 3. Caso contrário identificar a primeira tarefa em atraso, J_{uk}, e ir para o PASSO 2.

Passo 2

Seleccionar em $(J_{u1}, J_{u2},\ldots, J_{uk})$ a tarefa com maior tempo de processamento e excluí-la da sequência corrente. Voltar ao Passo 1.

Passo 3

Juntar à sequência corrente, a seguir à sua última tarefa, todas as tarefas anteriormente excluídas (por ordem arbitrária).

Interpretando o problema da alimentação das máquinas de divisão de correio como um problema do tipo $1 \,||\, \Sigma_j U_j$ foi desenvolvido um algoritmo de tipo construtivo, SeqLotes, para gerar as sequências de lotes a introduzir em cada máquina.

3.1.2 Algoritmo SeqLotes

Considere-se que as tarefas são a triagem dos lotes de cartas provenientes da indexação pelas divisoras e que não se admite a interrupção do seu processamento (*nopreemption*). Para cada lote o prazo é tomado igual à hora de corte do destino de menor hora de corte nele incluído (pelos motivos já explicitados) e o tempo de processamento é calculado em função do número de cartas que o constituem e da velocidade das máquinas (as de Lisboa e Porto dividem 1000 cartas em 2 minutos e a de Coimbra em 3 minutos). Embora o número de cartas por destino varie de dia para dia a empresa dispõe de diagramas de carga horária por destino que podem ser tomados como valores médios. Para concluir a modelização como um problema do tipo $1 \,||\, \Sigma_j U_j$ basta agora ultrapassar duas pequenas dificuldades: o facto de o momento a partir do qual os lotes ficam disponíveis para divisão não ser comum a todos (já que a indexação vai sendo feita ao longo do dia) e o facto de num dos CTC, o de Lisboa, existirem duas máquinas de divisão idênticas a funcionar em paralelo. Para ultrapassar a primeira dificuldade basta considerar o período diário de funcionamento da máquina, digamos [0, T], dividido em intervalos $I_k = [(k-1)\times\Delta, k\times\Delta]$ ($k = 1,\ldots, L$ e $L\times\Delta = T$) e que em cada intervalo estão disponíveis para divisão todos os lotes formados até ao seu início e ainda não divididos, sendo a duração do intervalo, Δ, suficiente para assegurar fiabilidade aos valores de chegada de tráfego estimados. Para ultrapassar a segunda dificuldade pode considerar-se o processamento numa máquina artificial com velocidade dupla da real e com base no escalonamento obtido gerar um escalonamento para as duas máquinas reais como é descrito na secção seguinte.

Algoritmo SeqLotes

Passo 0

Fixar a dimensão, Q, dos lotes de cartas a formar para cada saída dos leitores ópticos e os intervalos I_k (k = 1,..., L) a considerar no período diário de funcionamento. Fazer k := 1.

Passo 1

Formar lotes de dimensão Q com as cartas indexadas até ao início do intervalo I_k e ainda não separadas.

Passo 2

Aplicar o algoritmo de Moore-Hodgson aos lotes formados no Passo 1 até esgotar a duração de I_k ou os lotes disponíveis.

Passo 3

Se k < L fazer k := k + 1 e voltar ao Passo 1.

Caso contrário STOP.

Os intervalos de tempo I_k e a dimensão, Q, dos lotes são definidos de modo a que a duração de I_k seja um múltiplo do tempo gasto para processar Q cartas[1].

Embora não seja estritamente necessário que todos os lotes tenham a mesma dimensão (o número de cartas por lotes será um aspecto a discutir posteriormente) e que todos os intervalos tenham a mesma duração é vantajoso que assim seja.

Nestas condições, associadas com a de *nopreemptiom*, é fácil garantir que cada lote é sempre integralmente dividido num único intervalo I_k, evitando-se a ocorrência da divisão de um lote começar em I_k e continuar em I_{k+1}. Por outro lado, o facto de todos os lotes terem a mesma dimensão (e consequentemente o mesmo tempo de processamento) permite uma implementação muito eficiente do algoritmo de Moore-Hodgson.

Do ponto de vista operacional é também conveniente que lotes provenientes da mesma saída do leitor óptico sejam, se possível, processados em sequência, para evitar as operações que acompanham as mudanças da saída dos *OCR* na alimentação de cada divisora. Como lotes de saídas distintas podem, eventualmente, ter a mesma hora de corte, na implementação do algoritmo de Moore-Hodgson este aspecto foi também considerado, adaptando-se o procedimento de ordenação dos lotes para que, executando em cada intervalo I_k o algoritmo de Moore-Hodgson, se promova a alimentação consecutiva de lotes da mesma saída dos *OCR*.

No Passo 1, em cada intervalo I_k, forma-se (para cada saída dos *OCR*) o maior número de lotes de dimensão Q com as cartas já indexadas e ainda não triadas e retêm-se as restantes para incluir nos lotes a formar em intervalos seguintes (se k < L).

A descrição detalhada da implementação é apresentada em [8].

---

[1] Uma hora é um valor "razoável" para a duração de I_k.

Proposição P1

O escalonamento, S, gerado pelo algoritmo SeqLotes maximiza o número de lotes divididos dentro da respectiva hora de corte para todo o período de funcionamento da máquina.

Demonstração

Por construção no escalonamento S a máquina de divisão só está inactiva se já tiverem sido divididos todos os lotes disponíveis no intervalo de tempo em causa. Dado que a antecipação de um lote preenchendo tempos de inactividade nunca piora o escalonamento, existe sempre um escalonamento óptimo, S^*, com a característica apontada para S.

Sejam S e S^* esses escalonamentos e $n_H(S)$ e $n_H(S^*)$ o número de lotes divididos dentro da respectiva hora de corte em cada um deles. Pretende-se provar que $n_H(S) = n_H(S^*)$.

Se $S \equiv S^*$ o resultado é trivial.

Se $S \neq S^*$ existe pelo menos um lote que em S ocupa uma posição diferente daquela que ocupa em S^*. Seja J_a o primeiro lote nessas condições. Sejam d_a a sua hora de corte, C_a o seu momento de conclusão em S, C_a^* o seu momento de conclusão em S^* e J_b o lote que ocupa a posição equivalente em S^*, como a figura ilustra:

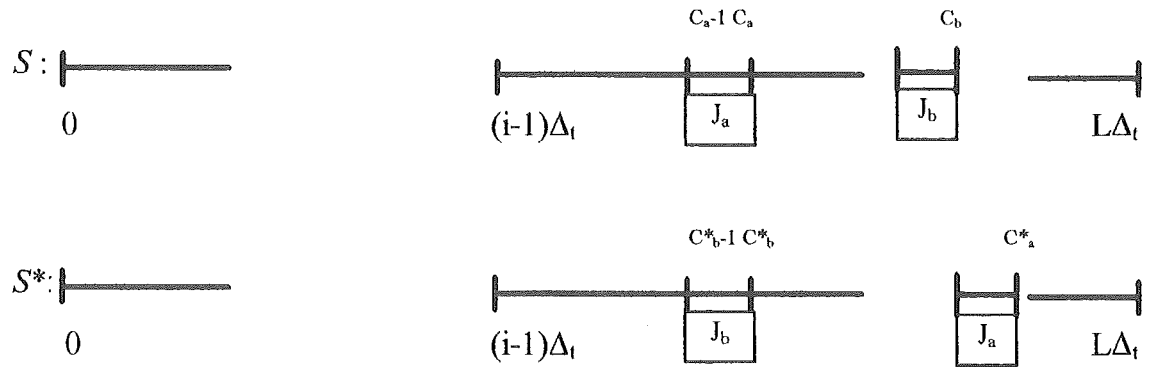


Figura 1 - Posição dos lotes J_a e J_b nos escalonamentos S e S^*

Seja S^1 o escalonamento que resulta de trocar J_a com J_b em S^*. Dois casos se podem dar:

Caso 1: $C_a \leq d_a$

No escalonamento S o lote J_a está a ser dividido dentro da hora de corte e precede o lote J_b. Como S foi construído com o algoritmo SeqLotes, se $d_b < d_a$ então $d_b < C_a$. Dado que $C_a = C_b^*$ tem-se $n_H(S^1) \geq n_H(S^*)$ e, portanto, $n_H(S^1) = n_H(S^*)$, por S^* ser, por hipótese, óptimo. Se $d_b \geq d_a$ a troca de J_a com J_b não gera nenhum novo atraso e tem-se de novo $n_H(S^1) = n_H(S^*)$, tal como anteriormente.

Caso 2: $C_a > d_a$

Como o lote J_a é dividido com atraso e precede J_b, em S, conclui-se que $d_b < C_a$. Como $C_a = C_b^*$ a troca de J_a com J_b não gera nenhum novo atraso e portanto $n_H(S^1) = n_H(S^*)$.

Então $n_H(S^1) = n_H(S^*)$ e o escalonamento S^1 tem mais um lote, J_a, cuja posição coincide com a que ocupa em S.

Se $S \equiv S^1$ então $n_H(S) = n_H(S^1) = n_H(S^*)$ e está provado o resultado.

Caso contrário pode repetir-se para S^1 e S o raciocínio feito para S^* e para S. Executando o raciocínio tantas vezes quantas as necessárias obtem-se uma sequência de escalonamentos S^1, S^2, ... , S^p, tal que $n_H(S^*) = n_H(S^1) = n_H(S^2) = \ldots = n_H(S^p)$ e $S^p \equiv S$. ♦

Caso se pretenda atribuir prioridades distintas a diferentes saídas dos leitores ópticos, o algoritmo SeqLotes pode ser complementado com um procedimento que, sem aumentar o número de lotes separados após a respectiva hora de corte, procede a troca de lotes no escalonamento por forma a antecipar, sempre que possível, o processamento dos lotes com maior prioridade. O procedimento é apresentado com pormenor em [8]. Após a execução deste procedimento são ainda feitas, sempre que possível, trocas que tornem consecutivos lotes de uma mesma saída para evitar operações desnecessárias de esvaziamento das máquinas e de mudança de ficheiros na memória.

3.1.3 O sistema com duas máquinas – caso do CTC de Lisboa

No CTC de Lisboa existem duas máquinas idênticas a funcionar em paralelo. Para poder aplicar o algoritmo SeqLotes a este caso considerou-se uma máquina fictícia com velocidade dupla da real, gerou-se com ele um escalonamento, S_f, e com base neste escalonamento obteve-se uma solução para as duas máquinas reais, $\{S_f^1, S_f^2\}$, fraccionando cada lote ao meio e atribuindo cada metade a uma das máquinas, mantendo a ordem de processamento da máquina fictícia.

Nas máquinas reais os tempos de processamento são duplos dos considerados na máquina fictícia e assim o tempo real de processamento de cada fracção (metade de um lote) é igual ao tempo fictício do correspondente lote. Como, por construção, as máquinas só ficam inactivas quando não existirem lotes disponíveis para triagem, os momentos de conclusão das fracções coincidem com o momento de conclusão do lote fictício.

Sendo $\{J_{I_k 1}, J_{I_k 2}, \ldots, J_{I_k n}\}$ o conjunto de lotes disponíveis para divisão num qualquer intervalo, I_k, representando por $J^1_{I_k m}$ e $J^2_{I_k m}$ as fracções do lote $J_{I_k m}$ atribuídas às máquinas 1 e 2, respectivamente, o procedimento pode ser ilustrado como segue (com n = 3):

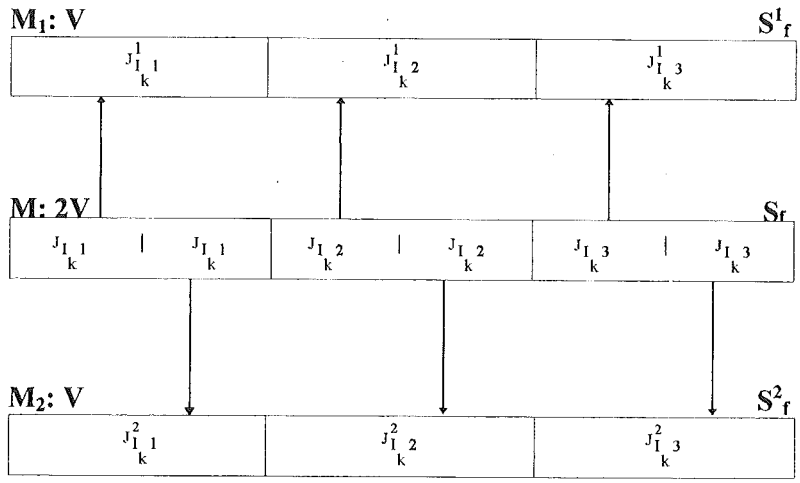


Figura 2 - Escalonamento para duas máquinas a velocidade V a partir de um escalonamento para uma máquina a velocidade 2V

Do ponto de vista teórico o fraccionamento dos lotes ao meio é uma alteração muito significativa, uma vez que se abandona a condição de *nopreemption*. Do ponto de vista prático não levanta questões relevantes e permite, como é mostrado seguidamente na proposição P2, gerar um escalonamento onde o número de lotes processados com atraso não é superior a um escalonamento óptimo R construído para um sistema com duas máquinas em paralelo.

Proposição P2

Seja R um escalonamento óptimo para as duas máquinas iguais em paralelo, M_1 e M_2, gerado sob as condições que se vêm admitindo: *nopreemption*, tempo de funcionamento do sistema fraccionado em intervalos I_k de duração múltipla do tempo de divisão de lotes com Q cartas, e onde um lote começado a processar em I_k é concluído em I_k. Então no escalonamento $\{S_f^1, S_f^2\}$ o número de lotes divididos dentro da hora de corte é pelo menos igual ao de R, isto é: $n_H(R) \leq n_H(\{S_f^1, S_f^2\})$.

Demonstração

Considerem-se as tarefas executadas com R num período $I_k (1 \leq k \leq L)$, $\{J_{I_k1}, J_{I_k2}, \ldots, J_{I_kn}\}$ e ordenem-se por tempos de conclusão, $C_{I_k1} \leq C_{I_k2} \leq \ldots \leq C_{I_km}$, $(1 \leq m \leq n)$.

Tome-se agora uma tarefa, J_{I_ku}, escolhida arbitrariamente, executada, sem perda de generalidade, em M_1 que em R é concluída sem atraso. Para o tempo de conclusão de J_{I_ku} em R vem: $C_{I_ku}(R) \geq \Delta \times (k-1) + \sum_{J_{I_k} z \in X} p$, sendo X o conjunto formado pelas tarefas processadas em I_k na máquina M_1 até à conclusão de J_{I_ku} e p o tempo de divisão de um lote de dimensão Q à velocidade real.

Seja Y o conjunto de tarefas em R, processadas em M_2, concluídas não posteriormente a J_{I_ku}. Então: $C_{I_ku}(R) \geq \Delta \times (k-1) + \sum_{J_{I_k} z \in Y} p$.

Considere-se agora um escalonamento S' para a máquina fictícia com as tarefas realizadas no mesmo período I_k e na sequência correspondente a $C_{I_k1} \leq C_{I_k2} \leq \ldots \leq C_{I_km}$, $(1 \leq m \leq n)$. Como se mantém em S' a ordem de finalização e a velocidade de execução é dupla, vem para o tempo de conclusão da tarefa J_{I_ku} em S'

$$C_{I_ku}(S') = \Delta \times (k-1) + \sum_{J_{I_k} z \in (Y \cup X)} \frac{p}{2} \leq \Delta \times (k-1) + \text{Max}\left\{ \sum_{J_{I_k} z \in Y} p , \sum_{J_{I_k} z \in X} p \right\}$$

e daqui

$$C_{I_ku}(S') \leq C_{I_ku}(R).$$

Como J_{I_ku} não está atrasada em R, então também o não está em S'. Então, conclui-se da mesma forma, todas as tarefas realizadas sem atraso em R, qualquer que seja o período I_k, também o são em S'. Por P1 sabe-se que, na máquina fictícia, se tem $n_H(S') \leq n_H(S_f)$. Como em S_f e em $\{S_f^1, S_f^2\}$ os momentos de conclusão dos lotes e das respectivas fracções coincidem virá: $n_H(R) \leq n_H(S') \leq n_H(S_f) = n_H(\{S_f^1, S_f^2\})$. ♦

3. 2 A alimentação das divisoras como um problema de optimização de rotas

Admita-se que quando a alimentação de uma divisora é mudada de lotes de uma saída j para lotes de uma saída q ($q \neq j$), é necessário um tempo (*setup*) não nulo para proceder à substituição, sendo este tempo igual a zero se $q = j$. Se o tempo de execução de uma tarefa for a soma do tempo de divisão de um lote com o tempo de *setup* da divisora [5], então o tempo de processamento é função da forma como os lotes são sequenciados nas divisoras.

Escalonamentos com tempos de processamento dependentes da sequência com que as tarefas transitam nas máquinas do sistema de processamento (*sequence-dependent processing times*), são afins aos problemas de optimização de rotas de veículos [7].

O problema básico de optimização de rotas com horários de serviço (*time windows*) pode ser genericamente enunciado como sendo o da determinação do conjunto óptimo de rotas que devem ser realizadas por uma frota de veículos, sediada num armazém central, para servir um conjunto de clientes dentro dos seus horários de serviço. São considerados previamente conhecidos os horários e tempos de serviço dos clientes, as suas localizações e encomendas a entregar, a posição do armazém e a capacidade dos veículos. A função de desempenho a optimizar pode incluir vários critérios (custos fixos relativos aos veículos, custos variáveis relativos às distâncias percorridas, penalizações relativas a clientes não servidos, etc.). Esta versão básica do problema, bem como um número quase infindável de variantes, tem sido objecto de muito trabalho de investigação nos últimos anos [4], [6].

Neste estudo, a modelização do problema da alimentação das máquinas de divisão de correspondência como um problema de optimização de rotas foi feita tendo em atenção que se pretendia resolver o modelo recorrendo ao *software* comercial Optrak [9], que na empresa é já usado para outros fins.

Importa aqui destacar que a variante do modelo de distribuição assumido para o Optrak possibilita atribuir *time windows* distintas para cada cliente e também *time windows* diferenciadas para cada encomenda de um mesmo cliente. Consequentemente, são permitidas múltiplas visitas a um mesmo cliente. O Optrak suporta, igualmente, a definição de um tempo de descarga (tempo de serviço) para cada encomenda. Um critério de optimização possível, aplicável neste contexto, é a geração de rotas de modo a satisfazer o máximo de pedidos em tempo.

Assim, para cada CTC, considerou-se uma rede de caminhos completa, cujos nodos representam um armazém central e os clientes. Um cliente corresponde a uma saída dos leitores ópticos e os lotes que se vão formando em cada saída correspondem às encomendas pedidas pelos clientes e que são guardadas no armazém central.

Divida-se, tal como para o algoritmo SeqLotes, o período de funcionamento em intervalos I_k. A partir dos diagramas de carga, já referidos atrás, conhece-se o número médio de cartas que chega ao CTC para cada destino em cada intervalo I_k. Suponha-se, como anteriormente, que as cartas recebidas em I_k apenas estão disponíveis a partir do início de I_{k+1}. Daqui,

determina-se o número de cartas prontas para dividir no início de cada intervalo, para cada saída dos *OCR*. Formem-se lotes de dimensão Q, para cada saída dos *OCR* com as cartas indexadas no início de cada intervalo, considerando as que são insuficientes para formar um lote em I_k como se tivessem chegado em I_{k+1}. Então, para cada dia de trabalho sabe-se (em média) quantas "encomendas" existem, no "armazém central" para "entregar em cada cliente" e a serem entregues num horário de serviço definido pelo diferença de tempo que decorre entre o início do intervalo a partir do qual os lotes estão disponíveis e a sua hora de corte.

Cada máquina de divisão é representada por um veículo.

Quando um veículo está num cliente q para entregar encomendas, isto significa que uma divisora está ocupada a dividir lotes de cartas da saída q dos *OCR*. O tempo necessário à divisão de um lote de Q cartas é entendido neste modelo como o tempo de descarga para cada encomenda (tempo de serviço).

Quando se substitui a alimentação de uma divisora de uma saída q por uma saída j, o "veículo" é obrigado a efectuar um percurso entre q e j.

A manipulação das velocidades e das distâncias entre clientes e entre estes e o armazém, na rede de caminhos, permite assegurar que o tempo de deslocação entre dois clientes, q e j, traduza um tempo de *setup* associado a uma mudança na alimentação numa divisora e que não haja passagens pelo armazém central entre visitas a clientes.

Como cada veículo pode deslocar-se mais que uma vez a qualquer cliente, uma divisora não separará necessariamente todos os lotes de uma saída dos *OCR* consecutivamente.

O armazém central é início e fim do conjunto de rotas, ou seja da laboração.

Como se fixa o número de veículos, pode ser impossível satisfazer todas as encomendas dentro das janelas temporais exigidas pelos clientes. É então objectivo gerar rotas de modo a satisfazer o máximo de pedidos em tempo.

3.3 A alimentação das divisoras como um problema de afectação

Nas abordagens anteriormente apresentadas considerou-se sempre que a triagem das cartas é feita no CTC correspondente ao CDP de origem, de acordo com a prática corrente nos CTT. Em alternativa pode considerar-se o conjunto de todas as máquinas de divisão instaladas nos três CTC como um sistema integrado e admitir que a triagem de cada lote pode ser feita quer no CTC correspondente à origem quer no CTC correspondente ao destino. Para estudar essa alternativa foi desenvolvida uma modelização baseada no problema de afectação que se apresenta de seguida de forma sucinta.

O problema de afectação é habitualmente enunciado como sendo o problema da atribuição ao custo mínimo de *n* tarefas a *n* pessoas de tal forma que cada pessoa realize uma e uma só tarefa e que cada tarefa seja realizada por uma e uma só pessoa. Nos casos em que o número de pessoas disponíveis não coincida com o número de tarefas a realizar podem considerar-se pessoas ou tarefas fictícias (associadas a custos nulos) para equilibrar o problema. Nos casos em que se queira impedir a atribuição de determinadas tarefas a determinadas pessoas pode

sempre fixar-se os custos correspondentes em valores suficientemente elevados (habitualmente representados por ∞) para inviabilizar essas atribuições. O problema de afectação é um problema típico de programação linear para o qual existem algoritmos de resolução particularmente eficientes [1].

Como cada tarefa tem de ser realizada por uma e uma só pessoa vamos considerar cada um dos intervalos de tempo de funcionamento de cada uma das máquinas de divisão subdivididos em subintervalos de comprimento unitário (isto é, igual ao tempo necessário para dividir um lote com a dimensão adoptada na máquina em causa). Esses subintervalos representam as tarefas sendo as pessoas representadas por todos os lotes de cartas com origem no conjunto de todos os CDP (note-se que alternativamente as tarefas poderiam ser representadas pelos lotes e as pessoas pelos subintervalos).

Para a definição dos custos de afectação tem de ter-se em consideração um conjunto de aspectos que se enunciam de seguida.

Cada lote só pode ser dividido no CTC da área de origem ou no CTC da área de destino pois não faz sentido proceder ao seu transporte para outra área. Assim, o custo de afectação de um lote aos subintervalos das máquinas de um CTC que não o da sua origem ou o do seu destino são sempre fixados em ∞.

A afectação de um lote a um subintervalo posterior à hora a partir da qual está disponível e anterior à sua hora de corte tem custo nulo pois representa a situação desejada. Note-se que para um mesmo lote estas horas são distintas no CTC da área do CDP de origem e no CTC da área do CDP de destino (a menos, evidentemente, que os CDP de origem e destino estejam associados ao mesmo CTC).

O custo de afectação de um lote a um subintervalo anterior à hora a partir da qual está disponível tem custo ∞ em todos os CTC. O custo de afectação de um lote a um subintervalo posterior à hora de corte respectiva reflecte a prioridade atribuída ao lote sendo fixado num valor positivo tanto mais alto quanto maior for essa prioridade.

Esta abordagem tem sobre as anteriores a vantagem de permitir que a decisão de dividir cada lote no CTC de origem ou no CTC de destino seja tomada em conjunto com a geração dos escalonamentos respectivos permitindo uma gestão mais integrada do conjunto das máquinas de divisão. Tem no entanto o grande inconveniente de ser muito pouco eficiente do ponto de vista computacional. Como facilmente se constata a percentagem de custos de afectação representados por ∞ é extremamente elevada. Assim as implementações usuais dos algoritmos de resolução do problema de afectação, baseadas em matrizes quadradas de custos, dificilmente comportam a dimensão dos problemas que será preciso resolver para analisar cenários reais para o conjunto de todas as máquinas instaladas nos CTC de Lisboa, Porto e Coimbra.

4. Resultados Computacionais

A aplicação dos métodos descritos anteriormente foi realizada sobre o CTC de Lisboa, com maior volume de tráfego e informação mais actualizada (1995).

O quadro 1 mostra o diagrama de cargas relativo às cartas recebidas. O quadro 2 contém horas de corte e prioridades (uma saída com o valor u é mais prioritária que outra com o valor v < u), por saída dos *OCR*. Os agrupamentos de destinos em cada saída, as horas de corte e prioridades são arbitradas, dado que à data de realização do trabalho ainda se estava na fase de instalar os *OCR*. Os valores de tráfego, atendendo aos agrupamentos, são factuais.

Intervalo I_k ->	1	2	3	4	5	6	7	8	9	10	11	12
Hora ->	12	13	14	15	16	17	18	19	20	21	22	23
Saídas *OCR*												
1	9357	6908	6294	7041	21748	5440	16733	10557	12665	11485	6028	1044
2	3084	2291	2143	2374	7040	1779	5545	3529	4147	3808	2012	340
3	8663	6453	5836	6298	19700	5133	15011	9552	11281	10530	5442	973
4	4083	3022	2841	3044	9451	2431	7173	4599	5383	5078	2728	446
5	3843	2881	2618	2947	8994	2340	6963	4370	5011	4759	2429	460
6	5080	3789	3472	3848	11872	2947	9180	5769	6671	6298	3202	583
7	6694	5028	4554	4916	15063	3858	11684	7383	8714	8258	4220	780
8	2174	1629	1497	1638	5000	1291	3831	2523	2962	2669	1416	262
9	5328	4076	3574	4002	12358	3100	9538	6119	7076	6845	3368	596
10	1866	1326	1338	1420	4355	1104	3317	2018	2440	2275	1141	186
11	12751	9476	8895	9546	29396	7401	22968	14134	16887	15697	8077	1469
12	2344	1725	1630	1765	5458	1274	4161	2525	3038	2843	1530	283
13	10596	8162	7408	7872	24665	6462	18846	12064	14128	13409	6773	1255
14	30366	22631	20661	22375	69814	17671	53216	33885	39897	37605	18975	3338
15	3370	2559	2271	2582	7761	1967	5791	3862	4427	4242	2095	354

Quadro 1
Diagrama de cargas por saída dos leitores ópticos

Intervalo I_k ->	13	14	15	16	17	18	19	20	21	22	23	24
Hora ->	24	1	2	3	4	5	6	7	8	9	10	11
Saídas *OCR*												
1	1512	3163	2291	993	2456	2659	2298	1198	771	1609	2463	15904
2	538	1022	726	331	720	864	783	422	256	559	759	5141
3	1340	2955	2037	920	2099	2385	2116	1135	749	1543	2188	14624
4	684	1394	930	437	966	1147	979	555	351	665	1051	7102
5	633	1259	963	396	990	1150	917	532	317	673	1015	6581
6	756	1727	1158	553	1300	1465	1253	667	414	889	1311	8676
7	1104	2228	1592	665	1610	1887	1611	899	573	1231	1691	11328
8	363	754	527	237	523	593	555	315	187	381	557	3725
9	817	1800	1303	581	1368	1576	1334	662	431	1004	1380	9141
10	265	634	474	218	445	540	486	241	154	329	529	3173
11	2035	4363	3138	1338	3079	3735	3172	1599	1038	2264	3392	21899
12	387	796	561	239	543	646	578	293	178	404	611	4081
13	1715	3626	2558	1103	2664	3027	2540	1440	882	1883	2731	18269
14	4846	10178	7301	3127	7458	8750	7421	3998	2522	5461	7940	51751
15	533	1187	812	357	844	954	853	442	288	574	835	5903

Quadro 1 (cont)

Saídas *OCR*	Hora de corte	Prioridade
1	3H:00M	10
2	4H:24M	8
3	6H:00M	8
4	20H:24M	8
5	1H:36M	8
6	24H:00M	1
7	1H:00M	10
8	3H:48M	9
9	4H:48M	8
10	0H:24M	7
11	23H:24M	6
12	2H:24M	5
13	2H:00M	4
14	4H:00M	3
15	4H:48M	2

Quadro 2
Horas de corte e prioridades

4.1 Soluções geradas com o algoritmo SeqLotes

O quadro 3 sumaria os resultados obtidos com o primeiro método proposto para lotes que são submúltiplos, em tempo equivalente de processamento, de um período de uma hora e na condição de prioridades iguais.

Como no CTC de Lisboa existem duas máquinas idênticas a funcionar em paralelo foi gerado primeiro um escalonamento para uma máquina fictícia com velocidade de 60000cartas/hora e depois, com base nele, gerado o escalonamento para duas máquinas com velocidade de 30000cartas/hora como descrito na secção 3.1.3.

O ensaio de lotes de várias dimensões, justifica-se pela tentativa de equilibrar argumentos operacionais, com a procura de tratar o máximo de cartas em D+1. Se o número escolhido de cartas por lote for "grande", a divisora processa, seguramente, cartas de uma saída do leitor óptico, durante um intervalo de tempo largo, sem necessitar de trocas de programas ou esvaziamento; por outro lado existem períodos de tempo, como se pode ver no quadro 1, em que o fluxo de chegadas é reduzido, e corre-se o risco de nenhuma saída do leitor óptico ter cartas para alimentar as divisoras durante todo o período de uma hora[2]. Se os lotes forem "pequenos", as divisoras estão sempre ocupadas, mas à custa de mudanças frequentes de saída dos leitores ópticos o que corresponde a alterações de programas e esvaziamentos a um ritmo inconveniente.

[2] A não opção por lotes de dimensão variável foi justificada anteriormente.

Cartas /Lote	1000		5000		10000		15000		20000		30000	
Saídas OCR	cartas	%	cartas	%	cartas	%	cartas	%	cartas	%	cartas	%
1	120000	78.8	120000	78.8	120000	78.8	120000	78.8	120000	78.8	120000	78.8
2	24000	47.8	15000	29.9	0	0	15000	29.9	20000	39.9	0	0
3	72000	51.8	75000	54.0	100000	72.0	90000	64.8	80000	57.6	90000	64.8
4	42000	63.2	40000	60.1	40000	60.1	30000	45.1	40000	60.1	30000	45.1
5	49000	78.5	50000	80.1	50000	80.1	45000	72.1	40000	64.1	30000	48.1
6	62000	75.6	60000	73.2	60000	73.2	60000	73.2	60000	73.2	60000	73.2
7	82000	76.5	80000	74.6	80000	74.6	75000	70.0	80000	74.7	60000	56.0
8	28000	80.8	25000	72.1	20000	57.7	15000	43.3	20000	57.7	0	0
9	24000	27.5	30000	34.4	30000	34.4	30000	34.4	20000	22.9	30000	34.4
10	23000	76.1	20000	66.2	20000	66.2	15000	49.7	20000	66.3	0	0
11	156000	75.4	1550000	75.0	150000	72.6	150000	72.6	140000	67.8	150000	72.6
12	30000	80.0	30000	80.0	30000	80.0	30000	80.0	20000	53.3	0	0
13	136000	78.7	135000	78.1	130000	75.2	135000	78.1	120000	69.4	120000	69.4
14	230000	47.1	245000	50.2	240000	49.2	240000	49.2	240000	49.2	300000	61.5
15	0	0	0	0	0	0	0	0	0	0	0	0
Total	1080000	60.6	1080000	60.6	1070000	60.0	1050000	58.9	1020000	57.2	990000	55.5

Quadro 3
Cartas divididas sem atraso por dimensão dos lotes.
Uma máquina, tempos de preparação nulos e prioridades iguais

A coluna cartas representa a quantidade de cada saída do leitor óptico dividida até à sua hora de corte e a percentagem corresponde a relação entre cartas chegadas e divididas sem atraso.

Em anexo apresentam-se os escalonamentos obtidos para lotes de 1000 e 10000 cartas. Neles pode observar-se a redução da percentagem de lotes tratados quando a dimensão cresce para 10000 (por insuficiência de cartas nalguns períodos para constituir lotes desta dimensão), mas à custa de frequentes e indesejáveis alterações nas saídas dos leitores ópticos que alimentam as divisoras, o que ilustra a importância da escolha do número de cartas por lote.

O quadro 4 apresenta os resultados obtidos aplicando aos escalonamentos do quadro 3 o procedimento de trocas desenvolvido para antecipar, se possível, a divisão dos lotes com maior prioridade.

Cartas/ Lote	1000		5000		10000		15000		20000		30000	
Saídas OCR	cartas	%	cartas	%	cartas	%	cartas	%	cartas	%	cartas	%
1	120000	78.8	120000	78.8	120000	78.8	120000	78.8	120000	78.8	120000	78.8
2	42000	83.7	40000	79.7	40000	79.7	30000	59.8	20000	39.9	30000	59.9
3	115000	82.7	115000	82.7	110000	79.1	105000	75.5	120000	86.3	90000	64.8
4	44000	66.2	40000	60.1	40000	60.1	30000	45.1	40000	60.1	30000	45.1
5	49000	78.5	50000	80.1	50000	80.1	45000	72.1	40000	64.1	30000	48.1
6	0	0	0	0	10000	12.6	0	0	0	0	0	0
7	82000	76.5	75000	70.0	80000	74.6	75000	70.0	80000	74.6	60000	56.0
8	28000	80.8	25000	72.1	20000	57.7	15000	43.3	20000	57.7	0	0
9	73000	83.6	70000	80.2	80000	91.7	60000	68.8	60000	68.8	60000	68.8
10	23000	76.1	20000	66.2	20000	66.2	15000	49.7	20000	66.3	0	0
11	156000	75.4	155000	75.0	150000	72.6	150000	72.6	140000	67.8	150000	72.6
12	32000	85.3	30000	80.0	20000	53.3	30000	80.0	20000	53.3	0	0
13	135000	78.1	135000	78.1	130000	75.2	135000	78.1	120000	69.4	120000	69.4
14	179000	36.6	205000	41.9	200000	40.9	240000	49.2	220000	45.0	300000	61.5
15	0	0	0	0	0	0	0	0	0	0	0	0
Total	1080000	60.6	1080000	60.6	1070000	60.0	1050000	58.9	1020000	57.2	990000	55.5

Quadro 4
Cartas divididas sem atraso por dimensão dos lotes
Uma máquina, tempos de preparação nulos e prioridades diferenciadas

Pode ver-se, comparando os quadros 3 e 4, que mantendo o número total de lotes divididos, lotes menos prioritários (exemplo: 6 e 14), são preteridos, sempre que possível, em relação aos de maior prioridade (exemplo: 2, 3 e 9).

Tomando lotes de menor dimensão, D = 1000 ou D = 5000, as saídas 2, 3 e 9 têm um grande aumento de lotes processados quando são consideradas prioridades diferenciadas, relativamente à situação de prioridades iguais. No entanto, com o crescimento do tamanho dos lotes, saídas mais prioritárias mas com menos cartas podem ser "prejudicadas" por outras de menos prioridade mas com número superior de cartas.

4.2 Soluções geradas com base num problema de optimização de rotas

Nos quadros 5 e 6 apresentam-se os resultados obtidos com o *software* comercial Optrak fixando a distância entre clientes em 1 km e a velocidade entre clientes em 6 km/h e 12 km/h para tempos de preparação de 10 e 5 minutos, respectivamente, com as prioridades para os lotes apresentadas no quadro 2.

A opção por lotes com 15000 e 20000 cartas foi determinada pelos resultados obtidos no estudo prévio reportado em [8]. Nesse estudo começou por considerar-se velocidades extremamente elevadas entre os clientes para anular praticamente os tempos de *setup* e obter resultados comparáveis com os gerados pelo algoritmo SeqLotes (que se provou serem óptimos nestas condições). Nessa comparação observou-se que para lotes de dimensão inferior a 15000 cartas os resultados se afastavam mais dos obtidos com o algoritmo SeqLotes (embora em menos de 10%).

Cartas/lote	15000		20000		30000	
Saídas OCR	cartas	%	cartas	%	cartas	%
1	120000	78.8	120000	78.8	120000	78.8
2	30000	59.8	40000	79.7	30000	59.8
3	105000	75.5	100000	72.0	90000	64.8
4	30000	45.1	40000	60.1	30000	45.1
6	45000	72.1	40000	64.1	30000	48.1
6	0	0	0	0	0	0
7	75000	70.0	80000	74.6	60000	56.0
8	15000	43.3	20000	57.7	0	0
9	60000	68.8	60000	68.8	60000	68.8
10	15000	49.7	20000	66.2	0	0
11	150000	72.6	140000	67.8	150000	72.6
12	30000	80.0	20000	53.3	0	0
13	135000	78.1	120000	69.4	120000	69.4
14	240000	49.2	220000	45.0	300000	61.5
15	0	0	0	0	0	0
Total	1050000	68.9	1020000	67.2	990000	65.5

Quadro 5
Cartas divididas sem atraso por dimensão dos lotes
Uma máquina, com tempos de preparação e prioridades diferenciadas

	Intervalos em minutos / Saídas *OCR*								
HORA									
12	0 -60								0 - 60
	—								—
13	0 - 60							0 -15	15 - 60
	S14							—	S14
14	0 - 30	30-60							0 - 60
	—	S11							S14
15	0 - 10	10 - 15	15 - 55	55-60				0 - 55	55 -60
	S11	—	S7	—				S14	—
16	0 - 40	40 - 45	45 - 60						0 - 60
	S6	—	S5						S13
17	0 - 25	25 - 30	30 - 60						0 - 60
	S5	—	S7						S13
18	0 -10	10 - 15	15 - 55	55 - 60				0 - 5	5 - 60
	S7	—	S6	—				—	S11
19	0 - 40	40 - 45	45 - 60						0 - 60
	S12	—	S4						S11
20	0 - 25	25 - 30	30 - 60						0 - 60
	S4	—	S8						S11
21	0 - 10	10 - 15	15 - 55	55 - 60					0 - 60
	S8	—	S7	—					S11
22	0 - 40	40 - 45	45 - 60		0 - 5	5 - 10	10 - 50	50 - 55	55 - 60
	S6	—	S5		S11	—	S10	—	S1
23	0 - 25	25 - 30	30 - 60						0 - 60
	S5	—	S10						S1
24	0 - 10	10 - 15	15 - 60						0 - 60
	S10	—	S13						S1
1	0 - 35	35 - 40	40 - 60						0 - 60
	S13	—	S14						S1
2	0 - 20	20 - 25	25 - 60					0 - 55	55 - 60
	S14	—	S9					S1	—
3	0 - 60								0 -60
	S9								S2
4	0 - 25	25 - 30	30 - 60				0 - 20	20 - 25	25 - 60
	S9	—	S3				S2	—	S3
5	0 - 50	50 - 60						0 - 45	45 - 60
	S3	—						S3	—

Quadro 6
Duas máquinas, 5 minutos de tempo de preparação e prioridades diferenciadas
Escalonamento para lotes de 20000 cartas

4.3 Soluções geradas com base num problema de afectação

Os resultados que se apresentam apenas servem para ilustração do método no caso do CTC de Lisboa já que, como se disse, não existem dados actualizados para os dois outros CTC. A aplicação computacional utilizada foi o TRANSFCL, desenvolvido pela Faculdade de Ciências da Universidade de Lisboa[3].

Apenas se trabalha com lotes de 15000, 20000 e 30000 cartas dado o *software* disponível não permitir lotes de dimensão inferior.

Os valores são idênticos para uma ou duas máquinas, dado que o ganho pela duplicação do número de intervalos usando duas máquinas é anulado pelo tempo de duração de cada intervalo (que passa para o dobro, dado o tempo de processamento de cada tarefa ser multiplicado por 2).

O quadro 7 apresenta os resultados obtidos com os coeficientes de custo fixados de acordo com as prioridades do quadro 2.

Embora o procedimento de trocas que foi aplicado ao escalonamento gerado pelo algoritmo SeqLotes, para ter em consideração as referidas prioridades, não garanta a obtenção de um escalonamento óptimo para essas prioridades, é interessante observar a semelhança entre os quadros 4 e 7.

---

[3] Biblioteca de programas de Investigação Operacional, Departamento de Estatística e Investigação Operacional, Faculdade de Ciências da Universidade de Lisboa.

Lotes	15000		20000		30000	
Saídas OCR	cartas	%	cartas	%	cartas	%
1	120000	78.8	120000	78.8	120000	78.8
2	30000	59.8	40000	79.7	30000	59.8
3	105000	75.5	100000	72.0	90000	64.8
4	30000	45.1	40000	60.1	30000	45.1
5	45000	72.1	40000	64.1	30000	48.1
6	0	0	0	0	0	0
7	75000	70.0	80000	74.6	60000	56.0
8	15000	43.3	20000	57.7	0	0
9	60000	68.8	60000	68.8	60000	68.8
10	15000	49.7	20000	66.2	0	0
11	150000	72.6	140000	67.8	150000	72.6
12	30000	80.0	20000	53.3	0	0
13	135000	78.1	120000	69.4	120000	69.4
14	240000	49.2	220000	45.0	300000	61.5
15	0	0	0	0	0	0
Total	1050000	58.9	1020000	57.2	990000	55.5

Quadro 7
Cartas divididas sem atraso por dimensão dos lotes
Tempos de preparação nulos e prioridades diferenciadas

5. Conclusões

Para que o avanço tecnológico registado nos equipamentos de indexação e divisão de correspondência se traduza num aumento real da produtividade da actividade postal, é necessário resolver um conjunto de novos problemas, entre os quais o da alimentação das divisoras com a correspondência proveniente dos equipamentos de indexação.

Neste estudo foram propostas três abordagens para este problema baseadas em modelos típicos de investigação operacional. Usando para a primeira abordagem um algoritmo desenvolvido no âmbito deste trabalho e recorrendo a *software* comercial para as outras duas, é possível gerar muito rapidamente soluções de boa qualidade para várias alternativas estudadas.

Com o primeiro método, considerando tempos de *setup* nulos, além dum limite para o máximo de cartas que é possível dividir em D+1, a forma como o escalonamento é gerado permite uma regra para alimentação das divisoras (que é a actualmente usada): os lotes devem ser escolhidos por hora de corte crescente.

Não negligenciando tempos de *setup*, a segunda abordagem possibilita concluir que se não deve esperar separar com os actuais equipamentos[4] muito mais que 55% das cartas em D+1 e estabelece, se o número de lotes e destinos não tiver grandes flutuações diárias, igualmente uma disciplina de alimentação para a triagem.

O estudo computacional realizado, embora ainda limitado ao CTC de Lisboa, permitiu também obter indicações sobre o impacte da dimensão dos lotes formados na indexação e das prioridades atribuídas aos diversos destinos. Espera-se assim que quando estiverem disponíveis dados actualizados para todos os CTC os métodos desenvolvidos neste trabalho possam constituir uma ferramenta útil quer para apoiar a actividade corrente dos CTC quer para detectar oportunidades de melhorar a qualidade do serviço prestado.

---

[4] A substituir a curto prazo. Acrescente-se que apenas para o correio azul existe a obrigatoriedade de 95% ser entregue em D+1 e existem linhas de tratamento manual.

Agradecimentos

O primeiro autor agradece aos CTT - Correios de Portugal as facilidades concedidas para realizar este trabalho.

O trabalho da segunda autora foi parcialmente apoiado pelo Centro de Investigação Operacional.

Ambos os autores agradecem a um revisor anónimo as críticas e sugestões que contribuiram para melhorar a apresentação do trabalho.

Referências Bibliográficas

[1] Bazaraa, M.; J.Jarvis; H. Sherali, Linear Programming and Network Flows, John Wiley (1990).
[2] Brucker, P. , Scheduling Algorithms, Springer Verlag (1995).
[3] CTT-Correios de Portugal, Relatório e Contas do ano 1997, (1998).
[4] Desrosieres, J., Y. Dumas, M. Solomon, F. Soumis, Time Constrained Routing and Scheduling, in M. O. Ball, T.L. Magnanti, C.L. Monma, G. L. Nemhauser (eds), Network Routing, Handbooks in Operations Research and Management Science 8 (1995) 35-139, North-Holland, Amesterdam.
[5] French, S., Sequence and Scheduling: an introduction to the mathematics of job-shop, Ellis Horwood (1990).
[6] Gendreau, M., G. Laporte, J.-Y. Potvin, Vehicle Routing: modern heuristics, in E. Aarts, J.K. Lenstra (eds), Local Search in Combinatorial Optimization (1997) 311-336, John Wiley & sons, Chichester.
[7] Lawler, E., Lenstra, J., Rinnooy Kan, A. , Shmoys, D., Sequencing and Scheduling: algorithms and complexity, in S.C. Graves, A.H.G.Rinnooy Kan, P.H. Zipkin (eds), Logistics of Production and Inventory, Handbooks in Operations Research and Management Science 4 (1993) 445-522, North-Holland, Amesterdam.
[8] Nunes, J. M., Problemas de Sequenciamento de Tarefas numa Máquina – aplicação à divisão de correio, Dissertação de Mestrado, Instituto Superior de Economia e Gestão UTL (1997).
[9] Optimiza, Notas sobre o OPTRAK 3.1., Optimiza Lda. (1995).

ANEXO

	Intervalos em minutos / Saídas *OCR*										
HORA											
12	0 - 4 S4	4 - 16 S11	16 -21 S6	21 - 22 S10	22 - 28 S7	28 - 31 S5	31 - 41 S13	41 - 43 S12	43 - 52 S1	52 - 54 S8	54 - 60 S14
13	0 - 3 S4	3 - 13 S11	13 -16 S6	16 - 18 S10	18 - 23 S7	23 - 26 S5	26 - 34 S13	34 - 36 S12	36 - 43 S1	43 - 44 S8	44 - 60 S14
14	0 - 2 S4	2 - 11 S11	11 -15 S6	15 - 16 S10	16 - 21 S7	21 - 24 S5	24 - 32 S13	32 - 33 S12	33 - 39 S1	39 - 41 S8	41 - 60 S14
15	0 - 3 S4	3 - 12 S11	12 -16 S6	16 - 17 S10	17 - 22 S7	22 - 25 S5	25 - 33 S13	33 - 35 S12	35 - 42 S1	42 - 43 S8	43 - 60 S14
16	0 - 10 S4	10 - 40 S11	40 - 52 S6	52 - 57 S10	57 - 60 S7						
17	0 - 2 S4	2 - 9 S11	9 -12 S6	12 - 13 S10	13 - 29 S7	29 - 40 S5	40 - 60 S13				
18	0 - 8 S4	8 - 31 S11	31 - 40 S6	40 - 43 S10	43 - 54 S7	54 - 60 S5					
19	0 - 4 S4	4 - 18 S11	18 - 23 S6	23 - 25 S10	25 - 33 S7	33 - 38 S5	38 - 60 S13				
20	0 - 6 S4	6 - 23 S11	23 - 30 S6	30 - 33 S10	33 - 41 S7	41 - 46 S5	46 - 60 S13				
21	0 - 16 S11	16 - 22 S6	22 - 24 S10	24 - 33 S7	33 - 38 S5	38 - 60 S13					
22	0 - 8 S11	8 -12 S6	12 - 13 S10	13 - 17 S7	17 - 20 S5	20 - 38 S13	38 - 59 S12	59 - 60 S1			
23	0 - 1 S11	1 - 2 S7	2 - 3 S13	3 - 60 S1							
24	0 - 1 S10	1 - 2 S7	2 - 3 S5	3 - 5 S13	5 - 34 S1	34 - 55 S8	55 - 60 S14				
1	0 - 1 S5	1 - 4 S13	4 - 5 S12	5 - 8 S1	8 - 9 S8	9 - 60 S14					
2	0 - 1 S12	1 - 4 S1	4 - 60 S14								
3	0 - 60 S14										
4	0 - 24 S2	24 - 48 S9	48 - 60 S3								
5	0 - 60 S3										

Quadro A1
Uma máquina, sem tempos de preparação e prioridades iguais
Escalonamento para lotes de 1000 cartas

	Intervalos em minutos / Saídas *OCR*										
HORA											
12	0 - 10 S11	10 - 20 S13	20 - 50 S14								
13	0 - 10 S11	10 - 20 S7	20 - 30 S1	30 - 50 S14	50 - 60 S3						
14	0 - 10 S11	10 - 20 S6	20 - 30 S13	30 - 40 S1	40 - 60 S14						
15	0 - 10 S4	10 - 20 S11	20 - 30 S7	30 - 40 S13	40 - 50 S5	50 - 60 S14					
16	0 - 10 S4	10 - 40 S11	40 - 50 S6	50 - 60 S10							
17	0 - 10 S6	10 - 30 S7	30 - 60 S13								
18	0 - 10 S4	10 - 40 S11	40 - 50 S6	50 - 60 S7							
19	0 - 10 S11	10 - 40 S13	40 - 60 S12								
20	0 - 10 S4	10 - 30 S11	30 - 40 S6	40 - 50 S7	50 - 60 S13						
21	0 - 10 S11	10 - 20 S10	20 - 30 S7	30 - 50 S13	50 - 60 S1						
22	0 - 10 S11	10 - 20 S6	20 - 30 S7	30 - 40 S13	40 - 60 S1						
23	0 - 60 S1										
24	0 - 30 S5	30 - 50 S8	50 - 60 S8								
1	0 - 60 S14										
2	0 - 10 S12	10 - 20 S1	20 - 30 S5	30 - 60 S14							
3	0 - 60 S14										
4	0 - 30 S9	30 - 60 S3									
5	0 - 60 S3										

Quadro A2
Uma máquina, sem tempos de preparação e prioridades iguais
Escalonamento para lotes de 10000 cartas

AUTOMATIC REFORMULATION FOR GENERAL MIXED 0-1 LINEAR PROGRAMS

Tim H. Hultberg

Department of Mathematical Modelling
Technical University of Denmark
DK-2800 Lyngby, Denmark

Abstract

The efficiency of solving mixed 0-1 linear programs by linear programming based branch-and-bound algorithms depends heavily on the formulation of the problem. In seeking a better formulation, automatic reformulation employs a range of different techniques for obtaining an equivalent formulation of a given pure or mixed integer programming problem, such that the new formulation has a tighter LP-relaxation than the original formulation. This paper surveys the use of automatic reformulation techniques for general mixed 0-1 linear programs.

Keywords

Mixed 0-1 linear programming, automatic reformulation, branch-and-cut.

1. Introduction

Many practical problems including several well known combinatorial optimization problems can be formulated as mixed 0-1 linear programs. However some mixed 0-1 linear programming formulations can be impossible to solve in reasonable time by using standard linear programming based branch-and-bound algorithms. On the other hand it is known that the convex hull of the set of feasible solutions to a mixed 0-1 linear program is a polyhedron. Therefore a mixed 0-1 linear program could in principle be solved by an LP-algorithm if this polyhedral description was known. Unfortunately it is, in general, NP-hard to find this polyhedral description. The goal of automatic reformulation is to find, in a short time, an equivalent formulation with an LP-relaxation which approximates the convex hull of the set of feasible solutions as well as possible.

The automatic reformulation approach as described in [10] consists of two phases: preprocessing and cut generation. The basic ideas of preprocessing are described in [11] where another related technique called probing is also described. The second phase of automatic reformulation looks for valid inequalities which are violated by some feasible point of the LP-

relaxation. Adding such inequalities, also called cuts, to the formulation tightens the corresponding LP-relaxation.

Combinatorial optimization problems formulated as mixed 0-1 linear programs do often possess a special structure which can be used to characterize families of strong valid inequalitites. The identification of such families together with so-called separation algorithms to find cuts among its menbers have recently led to substantial improvements of exact algorithms for several combinatorial optimization problems [1]. Combinatorial cuts are problem specific by their nature and will not be considered here. Instead we will review the efforts to extend the success of reformulation to general mixed 0-1 linear programming.

The rest of the paper is organized as follows. In section two the mixed 0-1 linear programming problem is defined and some basic notation and concepts are introduced. Furthermore the standard LP-based branch-and-bound approach for solving this type of problem is explained to show the reader the importance of the formulation of the problem when using this kind of algorithm to solve it. Preprocessing is the subject of section three. Section four discusses lifting of cuts and 3 different types of cuts which do not assume underlying structure of the problem: Gomoty cuts, disjunctive cuts and knapsack cuts.

Throughout the paper we will use the following notation. If $S \subseteq R^n$ and $p(x)$ is a proposition defined on R^n then $S_p(x)$ denotes $\{x \in S : p(x)\}$.

2. Background and Preliminaries

A mixed 0-1 linear program, M, is an optimization problem of the form

$$M : \text{maximize}\{cx : x \in S\}$$

where c is a row vector in R^n and S is a subset of $\{0,1\}^p \times R_+^{n-p}$ specified by a system of linear inequalities, i.e.

$$S = \{x \in \{0,1\}^p \times R_+^{n-p} : Ax \leq b\}$$

where $A \in R^{m \times n}$ and b is a column vector in R^m. The variables $x_1, \ldots, x_p$ are called the binary variables and the variables $x_{p+1}, \ldots, x_n$ are called the continuous variables.

We use J to denote the index set of variables $\{1, \ldots, n\}$, $D = \{1, \ldots, p\}$ the index set of the binary variables and $C = \{p+1, \ldots, n\}$ the index set of the continuous variables.

Generally the same set of feasible solutions, S, can be obtained by many different systems of inequalities.

Example 1

Let $A = \begin{bmatrix} 2 & 1 & 1 \\ -1 & 2 & -1 \end{bmatrix}$ and $b = \begin{bmatrix} 3 \\ 0 \end{bmatrix}$ and let $\hat{A} = [\,0 \quad 1 \quad 0\,]$ and $\hat{b} = [0]$. Since

$$\{x \in \{0,1\}^3 : Ax \leq b\} = \{x \in \{0,1\}^3 : \hat{A}x \leq \hat{b}\}$$

$Ax \leq b$ and $\hat{A}x \leq \hat{b}$ provides two different formulations of the same set.

Clearly the two problems $M_1 \equiv \text{maximize}\{cx : x \in \{0,1\}^p \times R_+^{n-p},\ Ax \leq b\}$ and $M_2 \equiv \text{maximize}\{cx : x \in \{0,1\}^p \times R_+^{n-p},\ \hat{A}x \leq \hat{b}\}$ are equivalent if $\{x \in \{0,1\}^p \times R_+^{n-p} : Ax \leq b\} = \{x \in \{0,1\}^p \times R_+^{n-p} : \hat{A}x \leq \hat{b}\}$. In this case we say that M_2 is a **reformulation** of M_1.

Given a particular formulation of M, we obtain a linear program, LPR, by relaxing the integrality requirement of the variables $x_1,\ldots, x_p$,

$$\text{LPR} : \text{maximize}\{cx : x \in P\}$$

where

$$P = \{x \in [0,1]^p \times R_+^{n-p} : Ax \leq b\}$$

We will refer to LPR as the LP-relaxation of M (having in mind that LPR depends upon the formulation of M).

The optimal objective function values of the two programs will be denoted z_M and z_{LPR}.

$$z_M = \max\{cx : x \in S\},\ z_{LPR} = \max\{cx : x \in P\}$$

We note that $S \subseteq P$, since $S = P \cap (\{0,1\}^p \times R_+^{n-p})$. It follows directly that $z_M \leq z_{LPR}$. The difference $z_M - z_{LPR}$ is called the **integrality gap**. If an optimal solution, x^*, to LPR, happens to be in S, x^* is also an optimal solution to M and the integrality gap is zero.

Two different equivalent formulations of a mixed 0-1 linear program will generally have different LP-relaxations, since $\{x \in \{0,1\}^p \times R_+^{n-p} : Ax \leq b\} = \{x \in \{0,1\}^p \times R_+^{n-p} : \hat{A}x \leq \hat{b}\}$ does not imply that $P_1 = \{x \in [0,1]^p \times R_+^{n-p} : Ax \leq b\}$ is equal to $P_2 = \{x \in [0,1]^p \times R_+^{n-p} : \hat{A}x \leq \hat{b}\}$. If $P_2 \subset P_1$ we say that the second formulation of the problem has a tighter LP-relaxation than the first.

Example 2

(Example 1 continued) The two equivalent formulations have different LP-relaxations, in particular

$$\{x \in \{0,1\}^3 : [\,0 \quad 1 \quad 0\,]\,x \leq [0]\} \subset \{x \in \{0,1\}^3 : \begin{bmatrix} 2 & 1 & 1 \\ -1 & 2 & -1 \end{bmatrix} x \leq \begin{bmatrix} 3 \\ 0 \end{bmatrix}\}$$

So the LP-relaxation of the second formulation is the tightest of the two.

Definition 1

$\alpha x \leq \beta$ is a **valid inequality** (for S) if $\alpha x \leq \beta$ for all $x \in S$

Definition 2

A **cut** is a valid inequality $\alpha x \leq \beta$ such that there exists a $\hat{x} \in P$ with $\alpha \hat{x} > \beta$

We will generally be interested in cuts which eliminate x^* (i.e. such that $\alpha x^* > \beta$) where x^* is an optimal solution to LPR.

We observe that if $\alpha x \leq \beta$ is a cut then

$$\begin{bmatrix} A \\ \alpha \end{bmatrix} x \leq \begin{bmatrix} b \\ \beta \end{bmatrix}$$

is a reformulation of M and that the new formulation has a tighter LP-relaxation.

Since we are maximizing a linear function, an optimal solution to M is also optimal to $\text{maximize}\{cx : x \in \text{convh}(S)\}$. The convex hull of feasible solutions to M, convh(S), is contained in P for any formulation of M. Standard results from polyhedral theory [8,12] imply that convh(S) is a polyhedron which means that it can be described by a system of linear inequalities. In other words, there exist an $\bar{A}$ and a $\bar{b}$ such that

$$P = \{x \in [0,1]^p \times R_+^{n-p} : \tilde{A} \leq \tilde{b}\} = \text{convh}(S)$$

This formulation of the feasible region, S, is the best possible since the optimization of any linear objective function over S can be done by solving a linear program with the set of feasible solutions defined by P.

2.1 LP based branch-and-bound

When solving a mixed 0-1 program by a linear programming based branch-and-bound algorithm, the formulation of the problem being solved is crucial for the efficiency. In this section we shall see why this is the case.

Linear programming based branch-and-bound algorithms for solving M are based on two basic facts.

For any $F \subseteq S$

1. $\max\{cx : x \in F\} = \max\{\max\{cx : x \in F_{x_j=0}\}, \max\{cx : x \in F_{x_j=1}\}\}$

   This can be used recursively to divide the search for an optimal solution to M into several smaller subproblems.

2. If $\max\{cx : x \in F\} < cx$ for some $x \in S$ then F does not contain an optimal solution to M.

   This can be used to eliminate F from further search. If we can find an upper bound UB on $\max\{cx : x \in F\}$ such that $UB < cx$ for some $x \in S$, it is not necessary to obtain the exact value of $\max\{cx : x \in F\}$ to be able to eliminate F from the search for an optimal solution. The objective function value of the LP-relaxation of $\max\{cx : x \in F\}$ provides such an upper bound.

Figure 1 sketches a branch-and-bound algorithm for solving mixed 0-1 programs based on these principles.

```
z = -∞
ActNodes = {P}
while ActNodes ≠ Ø do
   pick an element R of ActNodes
   ActNodes = ActNodes\{R}
   (z*,x*) = (max{cx : x∈R}, argmax{cx : x∈R})
   if z* > z then
     if x*∈S then
       (z,x) = (z*,x*)
     else
       pick a 1 ≤ j ≤ p such that 0 < x*_j < 1
       ActNodes = ActNodes∪{R_{x_j=0},R_{x_j=1}}
     end if
   end if
end while
```

Figure 1 - Linear programming based branch-and-bound algorithm for solving M

When the algorithm terminates, x contains the optimal solution of M.

At any point of the execution of the algorithm, the elements of ActNodes are called the active nodes. P is called the root node. When examining a node, two things can happen: 1) the node is fathomed (eliminated from further search) 2) the node is substituted by two child nodes dividing the set of feasible solutions of the node by fixing a binary variable to each of its two possible values. A node is fathomed whenever a) the solution of its LP-relaxation belongs to S b) the LP-relaxation is infeasible or c) the upper bound given by the objective function value of its LP-relaxation is smaller than the objective function value of the best feasible solution found so far.

The tree of nodes generated by the algorithm rooted by the root node is called the search tree. By the level of a node we will understand the depth in the search tree or equivalently the number of fixed binary variables. Since binary variables can only be fixed once, the level of node in the search tree cannot exceed p. If no nodes of level smaller than p could be fathomed, the search tree would be a complete binary tree of depth p implying the solution of 2^{p+1}-1 linear programs; an impossible task for most problems of any realistic size.

In order to keep the search tree to a manageable size we must be able to fathomed nodes of levels smaller than p (the smaller, the better). This can be achieved by having both good lower and upper bounds. The lower bound is the objective function value of the best feasible solution found so far. For this reason many algorithms include heuristics for finding good feasible solutions. The quality of the upper bound of each node can be measured by the integrality gap of the node. As we have seen, the integrality gap depends on the formulation of the problem. While the integrality gap a descendant node can be bigger than in the root node, it is still true that a formulation of the root node with a tight LP relaxation will also lead to tight LP relaxations of the descendant nodes and thereby to good upper bounds.

The strategy for picking the next active node R of ActNodes to examine and the strategy for choosing a binary variable x_j for generating child nodes also affect the size of the search tree. However, no matter what strategies are used, a tighter formulation of M will still decrease the size of the search tree and thereby the execution time of the algorithm.

3. Preprocessing

By using only simples bounds on the variables and a single inequality of the formulation of M at a time, preprocessing techniques can often find a much improved formulation of M without requiring a big computational effort.

3.1 Successive strengthening of bounds

l_j is called a lower and u_j an upper bound on x_j if $l_j \leq x_j \leq u_j$ for all $x \in S$. A lower bound is said to be stronger than another if it is higher. An upper bound is said to be stronger than another if it is lower.

The availability of strong bounds on the variables of MIP is important to many of the reformulation techniques.

It follows directly from the definition of M that $l_j = 0$, $u_j = 1$ for $1 \le j \le p$ and $l_j = 0$, $u_j = \infty$ for $p+1 \le j \le n$ are lower and upper bounds on x.

By isolating a variable in an inequality we might be able to strengthen the bounds on this variables. If $a_{ij} > 0$, isolating x_j in the i'th equality, we get

$$x_j \le (b_i - \sum_{k \in J \setminus \{j\}} a_{ik}x_k)/a_{ij}$$

Any upper bound on the right hand side is an upper bound on x_j. Finding the lowest possible upper bound on the right hand side would involve the maximization of the right hand side subject to $x \in S$; a problem just as difficult as the original problem. If $S \subseteq \{x \in R^n : l \le x \le u\}$ (that is, if l and u are lower and upper bounds on x) maximizing the right hand side subject to $x \in \{x \in R^n : l \le x \le u\}$ will provide an alternative upper bound x_j which is generally not as strong, but is much faster to find. This maximum is achieved simply by setting each variable x_k for which a_{ik} is positive equal to its lower bound and each variable x_k for which a_{ik} is negative equal to its upper bound.

If $a_{ij} < 0$,

$$x_j \ge (b_i - \sum_{k \in J \setminus \{j\}} a_{ik}x_k)/a_{ij}$$

and a lower bound on x_j can be found in a similar fashion.

We note that if we are able to strengthen a bound on a binary variable, the value of this variable can be fixed.

3.2 Coefficient reduction

In this subsection we shall drop the row indices when referring to a single unspecified inequality of $Ax \le b$ and let $ax \le b$ be a generic inequality of the formulation of M. In other words we let $ax \le b$ represent $\sum_{k \in J} a_{ik}x_k \le b_i$ for some $1 \le i \le m$.

Theorem 1

(Coefficient reduction) Let x_j be a binary variable. An inequality $ax \le b$ of the formulation of M can be replaced by

$$(a_j - \delta)\, x_j + \sum_{k \ne j} a_k x_k \le b - \delta \qquad (1)$$

without affecting the set of feasible solutions if $0 \le \delta \le b - \max\{\sum_{k \ne j} a_k x_k : x \in S_{x_j=0}\}$

Proof:

(1) is valid for $S_{x_j=1}$ for any value of δ. For $x \in S_{x_j=0}$, (1) becomes $\sum_{k \ne j} a_k x_k \le b - \delta$ which is valid for $S_{x_j=0}$ for $\delta \le b - \max\{\sum_{k \ne j} a_k x_k : x \in S_{x_j=0}\}$. This shows that (1) is valid for S for

$\delta \leq b - \max\{\sum_{k\neq j} a_k x_k : x \in S_{x_j=0}\}$. Now let $\bar{S}$ be the set of feasible solutions after $ax \leq b$ has been replaced by (1). $ax \leq b$ is valid for $\bar{S}_{x_j=1}$ for any value of δ. For $x \in \bar{S}_{x_j=0}$ we have $\sum_{k\neq j} a_k x_k \leq b - \delta$ which implies $\sum_{k\neq j} a_k x_k \leq b$ and therefore that $ax \leq b$ is valid for $\bar{S}_{x_j=0}$ for $\delta \geq 0$.

The coefficient reduction reformulation suggested by the theorem can be seen to tighten the LP-relaxation if $\delta > 0$. The maximum tightening is achieved for $\delta = b - \max\{\sum_{k\neq j} a_k x_k : x \in S_{x_j=0}\}$. Obtaining the exact value of $\max\{\sum_{k\neq j} a_k x_k : x \in S_{x_j=0}\}$ is practically as difficult as solving M itself. Instead we can use an upper bound on $\max\{\sum_{k\neq j} a_k x_k : x \in S_{x_j=0}\}$. Such an upper bound can be found quickly by using the bounds on the individual variables. Combining the bounds on the variables with a few of the remaining inequalities of M can often strengthen this upper bound without an excessive computational effort.

Example 3

Consider the constraint

$$10x_1 + 4x_2 + 4x_3 \leq 10$$

of M. Suppose that x_1 is binary variable and that $0 \leq x_2, x_3 \leq 1$ are valid bounds. Then $8x_1 + 4x_2 + 4x_3 \leq 8$ can replace the old constraint since the upper bound of 8 on $\max\{4x_2 + 4x_3 : x \in S_{x_1=0}\}$ is easily obtained from the bounds on x_2 and x_3. Now suppose that

$$x_2 + x_3 \leq 1$$

is another constraint in M. Using this "side constraint" we see that $\max\{4x_2 + 4x_3 : x \in S_{x_1=0}\} \leq 4$ so the original constraint can be replaced by

$$4x_1 + 4x_2 + 4x_3 \leq 4$$

which in turn is equivalent to $x_1 + x_2 + x_3 \leq 1$

The theorem can easily be extended to simultaneous reduction of coefficients (Recall that D is the index of the binary variables and C is the index set of the continuous variables).

Theorem 2

An inequality $ax \leq b$ of the formulation of M can be replaced by

$$\sum_{j\in D} (a_j - \delta_j)x_j + \sum_{j\in C} a_j x_j \leq b - \sum_{j\in D} \delta_j \qquad (2)$$

where $0 \leq \delta_j \leq \max\{0, b - \max\{\sum_{k\neq j} a_k x_k : x \in S_{x_j=0}\}\}$ without affecting the set of feasible solutions.

Theorem 3

(Subset coefficient reduction) Let x_j be a binary variable and let $ax \leq b$ an inequality of the formulation of M. If $K \subset J$ such that $j \notin K$ and $\{i : a_i < 0\} \subseteq K$ then

$$(a_j - \delta)x_j + \sum_{k \in K} a_k x_k \leq b - \delta \qquad (3)$$

is a valid inequality when $0 \leq \delta \leq b - \max\{\sum_{k \in K} a_k x_k : x \in S_{x_j=0}\}$.

Proof:

The inequality $a_j x_j + \sum_{k \in K} a_k x_k \leq b$ is implied by $ax \leq b$. Now the theorem follows by applying coefficient reduction on the implied inequality.

Example 4

Let $x_1 \in \{0,1\}$ and $0 \leq x_2, x_3, x_4, x_5 \leq 1$, and consider the inequality

$$3x_1 + x_2 + x_3 + x_4 + x_5 \leq 4$$

By choosing $K = \{2,3,4\}$ we obtain the valid inequality

$$2x_1 + x_2 + x_3 + x_4 \leq 3$$

which is not available by simple coefficient reduction.

4. Cut Generation

4.1 Lifting of cuts

Algorithms which combine branch-and-bound and automatic reformulation are called branch-and-cut algorithms. Automatic reformulation can in principle be applied at any node of an LP based branch-and-bound algorithm. However the fact that cuts generated at one node are not generally valid at other nodes makes this approach computationally unattractive. Lifting provides a technique to overcome this drawback by obtaining a globally valid inequality from an inequality valid only at a given node.

Theorem 4

Let $\alpha x \leq \beta$ be a valid inequality for $S_{x_j=0}$. Then $\tilde{\alpha}_j x_j + \sum_{k \neq j} \alpha_k x_k \leq \beta$ is a valid inequality for S if $\tilde{\alpha}_j \leq b - \max\{\sum_{k \neq j} \alpha_k x_k : x \in S_{x_j=1}\}$.

Proof:

To show that the inequality is valid for $S = S_{x_j=0} \cup S_{x_j=1}$ we must show that it is valid for both $S_{x_j=0}$ and $S_{x_j=1}$. $\tilde{\alpha}_j x_j + \sum_{k \neq j} \alpha_k x_k \leq \beta$ is valid for $S_{x_j=0}$ for any value of $\tilde{\alpha}_j$. For $x \in S_{x_j=1}$, $\tilde{\alpha}_j x_j + \sum_{k \neq j} \alpha_k x_k = \tilde{\alpha}_j + \sum_{k \neq j} \alpha_k x_k \leq \beta$ which is valid for $S_{x_j=1}$ if $\tilde{\alpha}_j \leq \beta - \max\{\sum_{k \neq j} \alpha_k x_k : x \in S_{x_j=1}\}$.

The strongest inequality is obtained for $\tilde{\alpha}_j \leq \beta - \max\{\sum_{k\neq j} \alpha_k x_k : x \in S_{x_j=1}\}$. But normally we will be satisfied with a value obtained from a relaxation of the maximization problem.

If $\alpha x \leq \beta$ is a valid inequality for $S_{x_j=1}$ we can obtain a valid inequality for S by applying the theorem with x_j complemented ($\alpha x \leq \beta$ is valid for $S_{\hat{x}_j=1}$ where $\hat{x}_j = 1 - x_j$).

If a valid inequality is given for a subset of S with more than one variable fixed, a valid inequality for S is obtained by sequentially lifting each of the fixed variables. Generally, different valid inequalities can be obtained by varying the order in which the fixed variables are considered for lifting.

Example 5

Let $S = \{x \in \{0,1\}^4 : 13x_1 + 11x_2 + 11x_3 + 10x_4 \leq 32\}$. We see that $x_2 + x_3 \leq 1$ is a vlid inequality for $S_{x_1=1,x_4=0} = \{(x_2,x_3) \in \{0,1\}^2 : 11x_2 + 11x_3 \leq 19\}$

1. By lifting $x_2 + x_3 \leq 1$ for $\hat{x}_1$ we get $\tilde{\alpha}_1 = 1 - \max\{x_2 + x_3 : S_{\hat{x}_1=1,x_4=0}\} = 1 - 2 = -1$ so the inequality becomes $-(1 - x_1) + x_2 + x_3 \leq 1$ or $x_1 + x_2 + x_3 \leq 2$. Lifting this for x_4 we get $\tilde{\alpha}_4 = 2 - \max\{x_1 + x_2 + x_3 : S_{x_4=1}\} = 2 - 2 = 0$, so the valid inequality for S becomes

   $$x_1 + x_2 + x_3 \leq 2$$

2. Lifting $x_2 + x_3 \leq 1$ for x_4 first, we get $\tilde{\alpha}_4 = 1 - \max\{x_2 + x_3 : S_{\hat{x}_1=0,x_4=1}\} = 1 - 0 = 1$, so the inequality becomes $x_2 + x_3 + x_4 \leq 1$. Lifting this for $\hat{x}_1$ we get $\tilde{\alpha}_1 = 1 - \max\{x_2 + x_3 + x_4 : S_{\hat{x}_1=1}\} = 1 - 3 = -2$ so the valid inequality for S becomes $-2(1 - x_1) + x_2 + x_3 + x_4 \leq 1$ or

   $$2x_1 + x_2 + x_3 + x_4 \leq 3$$

4.2 Gomory cuts

Gomory [7] pioneered the idea of adding valid inequalities to a given integer linear programming formulation in order to tighten the LP-relaxation until an integer optimal solution is obtained. A Gomory cut, violated by the current solution of the LP-relaxation, can be obtained for each fractional variable within it. Although the Gomory cuts are not immediately applicable to mixed 0-1 linear programs, the idea can be extended to this type of problem [3].

Let x^* be the optimal solution of the LP-relaxation of the current formulation of M and let B be the corresponding basis-matrix. Let x_j be a binary variable such that x_j^* is fractional. Let $y = e_i^T B^{-1} N$ (i.e. the i'th row of the optimal simplex-tableau) where i is the position of x_j in the basis. The i'th row of the optimal simplex-tableau reads

$$x_j + \sum_{k \in D_N} y_k x_k + \sum_{k \in C_N} y_k x_k = x_j^*$$

where D_N denotes the non-basic binary variables and C_N denotes the non-basic continuous variables including slack-variables. Let $\lfloor a \rfloor$ denote the biggest integer less than or equal to a and let f(a) denote $a - \lfloor a \rfloor$. Then the simplex-tableau rows can be rewritten as

$$x_j + \sum_{k \in D_N} \lfloor y_k \rfloor x_k + \sum_{k \in D_N} f(y_k) x_k + \sum_{k \in C_N} y_k x_k = x_j^*$$

The first two terms on the left side are integer valued for any feasible solutions to M. So we get

$$\sum_{k\in D_N} f(y_k)x_k + \sum_{k\in C_N} y_k x_k \equiv x_j^* \qquad \text{(mod1)}$$

This relation remains true if we subtract the integer $\sum_{k\in \hat{D}_N} x_k$ on the left side, for some subset $\hat{D}_N$ of D_N

$$\sum_{k\in D_N\setminus\hat{D}_N} f(y_k)x_k + \sum_{k\in \hat{D}_N} (f(y_k) - 1)x_k + \sum_{k\in C_N} y_k x_k \equiv x_j^* \qquad \text{(mod1)}$$

For this to be true, at least one of the following two conditionsmust be true. 1) the sum of the non-negative left hand side terms is greater than or equal to x_j^*, or 2) the sum of the negative left hand side terms is less than or equal to $x_j^* - 1$. To realize this, assume that both conditions are false, i.e. that the sum of the non-negative left hand sides is strictly less than x_j^* and the sum of the negative terms on the left hand side is strictly greater than $x_j^* - 1$. It follows the total sum of the left hand side terms is strictly between $x_j^* - 1$ and x_j^* and therefore cannot be equal to x_j^* modulo 1.

We get

$$\sum_{k\in D_N\setminus\hat{D}_N} f(y_k)x_k + \sum_{k\in C_N^+} y_k x_k \geq x_j^*$$

or

$$\sum_{k\in \hat{D}_N} (f(y_k) - 1)x_k + \sum_{k\in C_N^-} y_k x_k \leq x_j^* - 1$$

where $C_N^+ = \{k\in C_N : y_k > 0\}$ and $C_N^- = \{k\in C_N : y_k < 0\}$.

Multiplying the last inequality by -1, dividing both inequalitites by their right hand side and finally adding them yields

$$\sum_{k\in D_N\setminus\hat{D}_N} \frac{f(y_k)}{x_j^*} x_k + \sum_{k\in \hat{D}_N} \frac{1-f(y_k)}{1-x_j^*} x_k + \sum_{k\in C_N^+} \frac{y_k}{x_j^*} x_k - \sum_{k\in C_N^-} \frac{y_k}{1-x_j^*} x_k \geq 1 \qquad (4)$$

The left hand side is zero for the optimal solution to the LP-relaxation, since all the variables are non basic, hence the inequality is a cut.

For each $k\in D_N$ we can choose to put k in $\hat{D}_N$ or not. The strongest possible cut is obtained when the coefficients on the left hand side are as small as possible, so k should be put in $\hat{D}_N$ if $\frac{1-f(y_k)}{1-x_j^*}$ is less than $\frac{f(y_k)}{x_j^*}$. Thus it can be seen that the strongest cut is obtained for

$$\hat{D}_N = \{k\in D_N : f(y_k) > x_j^*\}$$

An important property of Gomory cuts for mixed 0-1 linear programming is that they can easily be lifted. Suppose, for example, that (4) is a Gomory-cut for $S_{x_l=0}$ where x_l is a binary variable. By setting $D_N = D_N \cup \{l\}$ and $y_l = e_i^T B^{-1} A_{.l}$ the inequality (4) becomes valid for S. This can be seen by considering the LP-relaxation of M with the constraint $x_l \leq 0$ added. By pivoting the slack variable of this constraint into the basis, x_l becomes non basic and the

validity of the proposed cut follows. Variables fixed at one can be treated similarly. Note that this property is not inherited by mixed integer linear programs because an integer variable which has been used to branch on might not continue to be non basic deeper down in the branch-and-bound search tree.

4.3 Disjunctive cuts

Disjunctive cuts [2] arise by considering valid inequalities for the set of feasible solutions of so called disjunctive programming relaxations of M.

Definition 3

Let d be a row vector in Z^p. Define

$$P(d) = \text{convh}(P_{dx_D \le 0} \cup P_{dx_D \ge 1}) \tag{5}$$

and

$$\text{DPR: maximize}\{cx : x \in P(d)\} \tag{6}$$

DPR is called a **disjunctive programming relaxation** of M.

To see that $S \subseteq P(d)$ and thereby that DPR is indeed a relaxation of M, we note that dx_D is integer valued for all $x \in S$ so $S = S_{dx_D \le 0} \cup S_{dx_D \ge 1}$.

$$S = S_{dx_D \le 0} \cup S_{dx_D \ge 1} \subseteq P_{dx_D \le 0} \cup P_{dx_D \ge 1} \subseteq \text{convh}(P_{dx_D \le 0} \cup P_{dx_D \ge 1}) = P(d)$$

We also note that $P(d) \subseteq P$.

The usefulness of P(d) lies in the fact that valid inequalities for P(d) can easily be characterized.

Lemma 1

Let $P = \{x \in R^n_+ : Ax \le b\}$.

$\alpha x \le \beta$ is valid for $P \Leftrightarrow \exists \lambda \ge 0 : \alpha \le \lambda A$ and $\lambda b \le \beta$

Proof:

"$\Leftarrow$": Assume λ is such that $\alpha \le \lambda A$ and $\lambda b \le \beta$. For $x \in P$ we get $\alpha x \le \lambda Ax$, since $x \ge 0$ and $\lambda Ax \le \lambda b$, since $Ax \le b$ and $\lambda \ge 0$. I.e. we have $\alpha x \le \lambda Ax \le \lambda b \le \beta$.

"$\Rightarrow$": Suppose that $\alpha x \le \beta$ is valid for P. Duality gives us that

$$\max\{\alpha x : Ax \le b, x \ge 0\} = \min\{yb : yA \le \alpha, y \ge 0\}$$

Let x^*, y^* be optimal solutions to the two LP problems above. Then $y^* \ge 0$, $y^*A \ge \alpha$ and $y^*b = \alpha x^* \le \beta$ since $x^* \in P$. Hence $\lambda = y^*$ fulfills the conditions.

Theorem 5

Let $\alpha \in R^n$ and $\beta \in R$. The inequality $\alpha x \le \beta$ is valid for P(d) if and only if there exist row vectors $u,v \ge 0$ and scalars $u_0, v_0 \ge 0$ such that

$$\alpha \le u\hat{A} + u_0[d \;\; 0_{n-p}] \tag{7}$$

$$u\hat{b} \le \beta \tag{8}$$

$$\alpha \le v\hat{A} - v_0[d \;\; 0_{n-p}] \tag{9}$$

$$v\hat{b} \le \beta + v_0 \tag{10}$$

$$\tag{11}$$

where $\hat{A} = \begin{bmatrix} I_{p\times p} & A \\ & 0_{p\times n-p} \end{bmatrix}$ and $\hat{b} = \begin{bmatrix} b \\ 1_p \end{bmatrix}$

Proof:

First note that

$$P_{dx_D\le 0} = \{x\in R^n_+ : \begin{bmatrix} A \\ d\ 0_{n-p} \end{bmatrix} x \le \begin{bmatrix} \hat{b} \\ 0 \end{bmatrix}\}$$

and

$$P_{dx_D\ge 1} = \{x\in R^n_+ : \begin{bmatrix} A \\ -d\ 0_{n-p} \end{bmatrix} x \le \begin{bmatrix} \hat{b} \\ -1 \end{bmatrix}\}$$

and that $\alpha x \le \beta$ is valid for P(d) if and only if it is valid for $P_{dx_D\le 0}$ and for $P_{dx_D\ge 1}$. Now the theorem follows by applying the lemma for $P_{dx_D\le 0}$ and $P_{dx_D\ge 1}$.

For a given d the theorem characterizes a family of valid inequalities for S. We would like to find a cut, $\alpha x \le \beta$, from this family of valid inequalities which eliminates x^* i.e. such that $\alpha x^* > \beta$. To this end we consider the LP

DISJ: maximize$\{\alpha x^* - \beta : u,u_0,v,v_0 \ge 0,\ (8),(9),(10),(11)\}$

which clearly has a solution with strictly positive objective function value if and only if such a cut exists. However if such a cut exists, DISJ is unbounded as any positive multiple of the cut is also a cut. Therefore, in order to use DISJ to find a cut we additionally impose the normalization constraint $\sum_{j\in J} |\alpha_j| \le 1$. This normalization constraint can easily be linearized so that DISJ remains an LP.

Now suppose that we have found a disjunctive cut, $\alpha x \le \beta$. We will show how to strengthen this cut by possibly increasing some of the α_j's. Note that for $j\in D$,

$$\alpha_j = \min\{uAe_j + u_0 d_j,\ vAe_j - v_0 d_j\}. \qquad (12)$$

Each d_j can be individually adjusted in order to increase the value of α_j if possible. Note that the new cut satisfies all the constraints of DISJ (without the normalization constraint) and is therefore valid.

4.4 Knapsack cuts

Knapsack cuts [6] arise by considering a single inequality at a time. They can be expected to be particularly effective when the problem is sparse.

From a given inequality $a_D x_D + a_C x_C \le b$ from the formulation of M, we can obtain a valid inequality, $a_D x_D \le \bar{b}$, in the binary variables only, where $\bar{b}$ is a lower bound on $b - a_C x_C$. Such a bound can be found quickly from the bounds on x_C. This inequality can be turned into a knapsack constraint by complementing the variables for which a_j is negative. For example $\{x_1,x_2\in\{0,1\} : -2x_1 + x_2 \le 0\}$ becomes $\{x_1,x_2\in\{0,1\} : 2(1 - x_1) + x_2 \le 2\}$ and then $\{\bar{x}_1,x_2\in\{0,1\} : 2\bar{x}_1 + x_2 \le 2\}$ introducing $\bar{x}_1 = 1 - x_1$. In the following we will assume that

$$a_D x_D \le \bar{b} \qquad (13)$$

is a knapsack constraint obtained from M in the way described above. A cut obtained with complemented variables can readily be expressed in the original variables.

Definition 4

The pair (U,t) with $U \subset D$ and $t \in D \setminus U$ is called a(1,k)-**configuration** with respect to $a_D x_D \leq \bar{b}$ if

$$\sum_{j \in U} a_j \leq \bar{b}$$

and

$$a_t + \sum_{j \in Q} a_j > \bar{b}$$

for any subset Q of U with k elements and

$$a_t + \sum_{j \in R} a_j \leq \bar{b}$$

for any subset R of U with k-1 elements. If $|U| = k$ then $U \cup \{t\}$ is called a **minimal cover.**

Theorem 6

Let (U,t) be a (1,k)-configuration with respect to $a_D x_D \leq \bar{b}$, $k \leq r \leq |U|$ and let T be a subset of U of cardinality r. Then

$$(r - k + 1)x_t + \sum_{j \in T} x_j \leq r \qquad (14)$$

is a valid inequality.

Proof:

Clearly $\sum_{j \in T} x_j \leq k - 1$ is valid for $S_{x_t=1}$. Lifting this inequality for $\hat{x}_t$ we obtain the lifting coefficient $\alpha_t = k - 1 - r$, since r is an upper bound on $\max\{\sum_{j \in T} x_j : x \in S_{\hat{x}_t=1}\}$. This shows that $(k - 1 - r)(1 - x_t) + \sum_{j \in T} x_j \leq k - 1$ is valid for S.

Example 6

Consider the knapsack constraint

$$13x_1 + 11x_2 + 11x_3 + 10x_4 \leq 32$$

$U = \{2,3,4\}$, $t = 1$ is a (1,2)-configuration with respect to this constraint since $a_2 + a_3 + a_4 = 11 + 11 + 10 \leq 32$ and $a_1 + a_{j1} + a_{j2} > 32$ for any $j_1 \neq j_2 \in S$. Choosing $r = 3$ and $T = \{2,3,4\}$ we obtain the valid inequality

$$2x_1 + x_2 + x_3 + x_4 \leq 3$$

In example 5 we saw how this inequality can also be generated by lifting the inequality $x_1 + x_2 \leq 1$ which is a valid minimal cover inequality for $S_{x_1=1,x_4=0}$.

To generate a cut from this family of valid inequalities for the knapsack constraint, we need a procedure to identify a minimal cover inequality violated by x^*. Such a procedure is provided by the following knapsack problem

$$MC: \min\{\sum_{j \in D} (1 - x_j^*)s_j : \sum_{j \in D} a_j s_j > \bar{b},\ s \in \{0,1\}^p\}$$

It can be seen that the set of indices, j, such that $s_j = 1$ in the optimal solution to MC is a minimal cover inequality violated by x^* if and only if the optimal objective function value is less than one.

Another approach for generating knapsack cuts is presented in [6]. For any $u \in R_+^p$, $u_0 \in R_+$

$$\sum_{j \in D} \lfloor u_0 a_j + u_j \rfloor x_j \leq \lfloor u_0 \bar{b} + \sum_{j \in D} u_j \rfloor \tag{15}$$

is valid inequality for S. Note that (15) is obtained by rounding down the coefficients of a linear combination of the source knapsack constraint and the upper bound constraints of the binary variables (i.e. it is a Chvatal-Gomory cut). A separation procedure (i.e. a procedure for finding a valid inequality violated by x^*) based on this family of valid inequalities can be stated as

$$SP : \max\{ \sum_{j \in D} \lfloor u_0 a_j + u_j \rfloor x_j^* - \lfloor u_0 \bar{b} + \sum_{j \in D} u_j \rfloor : u, u_0 \geq 0 \}$$

This problem can be stated as an integer linear program. As in section 4.3 we need to include a normalization constraint to make SP bounded. We can use $\lfloor u_0 \bar{b} + \sum_{j \in D} u_j \rfloor = K$ for some integer K as a normalization constraint. Then the separation problem becomes

$$\max\{ \alpha x^* : \alpha \leq u_0 a + u,\ u_0 \bar{b} + \sum_{j \in D} u_j - 1 + \varepsilon \leq K,\ u, u_0 \geq 0,\ \alpha_j \text{ integer},\ 1 \leq j \leq p \} \tag{16}$$

where e is some small value (for example e = 0.001). Instead of solving this integer linear program we can use the relaxation obtained be replacing the integrality requirement on α_j by $\alpha_j \geq 1$ for all j's.

6. Conclusion

Inspired by the recent success of branch-and-cut methods for several hard combinatorial optimization problems, there has been a renewed interest in the application of these methods to general mixed integer and 0-1 linear programs. Each of the automatic reformulation techniques presented in this paper has the potential to strengthen the LP-relaxation, leading to considerably reduced execution times for solving general mixed 0-1 linear programs, but the actual gains which are obtained are highly problem dependent. It therefore seems advisable to combine several of these techniques in order to achieve a robust algorithm for solving general mixed 0-1 linear programs.

References

[1] Aardal, K. and van Hoesel, S., Polyhedral techniques in combinatorial optimization, Technical report, Department of Computer Science, Utrecht University (1997).
[2] Balas, E., Ceria, S. and Cornuejols, G., Mixed 0-1 programming by lift-and-project in a branch-and-cut framework, Management Science 42 (1996).
[3] Balas, E., Ceria, S., Cornuejols, G. and Natrajm N., Gomory cuts revisited, OR Letters 19 (1996).
[4] Bixby, R., Cook, W., Cox, A. and Lee, E., Parallel mixed integer programming, Technical Report CRPC-TR95554, Center for Research on Parallel Computation (1995).
[5] Crowder, H., Johnson, E.L. and Padberg, M.W., Solving large-scale zero-one linear programming problems, Operations Research 31 (1983) 803-834.
[6] Glover, F., Sherali, H.D and Lee, Y., Generating cuts from surrogate constraint analysis for zero-one and multiple choice programming, in INFORMS (1995).
[7] Gomory, R.E., Outline of an algorithm for integer solutions to linear programs, Bulletin of the American Mathematical Society 64 (1958) 275-278.
[8] Nemhauser, G.L. and Wolsey, L.A., Integer and Combinatorial Optimization, John Wiley, New York (1988).
[9] Martin, R.K. and Schrage, L., Subset coefficient reduction cuts for 0/1 mixed-integer programming, Operations Research 33 (1985) 505-526.
[10] Van Roy, T. and Wolsey, L., Solving mixed integer programming problems using automatic reformulation, Operations Research 35 (1987) 45-57.
[11] Savelsbergh, M.W.P., Preprocessing and probing techniques for mixed integer programming problems, ORSA Journal on Computing 6 (1994) 445-454.
[12] Schrijver, A., Theory of Linear and Integer Programming, John Wiley, Chichester (1986).

THE OPTIMAL PATH PROBLEM

Ernesto de Queirós Vieira Martins
Marta Margarida Braz Pascoal
Deolinda Maria Lopes Dias Rasteiro
José Luis Esteves dos Santos

Centro de Informática e Sistemas
Departamento de Matemática
Universidade de Coimbra
Portugal

Abstract

In this paper, the optimal path problem will be studied from a global point of view and having no restrictions imposed on the network. The concepts of boundness and finiteness will be presented for the general problem and will be studied in two particular cases. Special emphasis will be given to the optimality principle since it allows one to design a class of algorithms - the labelling algorithms - which determine an optimal path when the weak optimality principle is satisfied. Its importance will be stressed by two problems which, in its turn, are similar in their description and completly different in what concerns their resolution.

Resumo

Neste artigo, o problema do trajecto óptimo será estudado dum ponto de vista global e sem que sejam impostas restrições à rede. Os conceitos de limitação e finitude serão apresentados para o problema geral e estudados para dois casos particulares. Será dada ênfase especial ao princípio de optimalidade, dado que nos permite a concepção de uma classe de algoritmos - os algoritmos de rotulação - que permitem determinar um trajecto óptimo quando se verifica o princípio da optimalidade fraca. A sua importância será realçada com dois problemas que, semelhantes na descrição, são completamente diferentes no que respeita ao modo de resolução.

Keywords

network, path, optimality principle, labelling algorithms.

1. Introduction

In the optimal path problem, a real function is considered which assigns a value to each path that can be defined between a given pair of nodes in a given network; a path with the *best* value in a subset of paths between that pair of nodes is what has to be determined.

As far as it is known, the optimal path problem has not been regarded in the literature as a general problem; that is, the optimal path problem has not been studied considering a general objective function. Several instances have merited consideration from researchers, although most of them assume some restrictions in the network, which imply some loss of generality.

In this paper the optimal path problem is considered from a global point of view and without any restrictions. As a consequence, boundness and finiteness have to be defined and

studied. As far as it is known, no paper deals with these two important concepts, perhaps since the usually assumed restrictions are sufficient to assure both of them.

The importance of the Principle of Optimality is also emphasized in this paper, since it is the support of a class of algorithms - namelly, labelling algorithms - largely studied for some instances of the optimal path problem. However, no authors are known which validate labelling algorithms with that principle. In this paper its importance is stressed by two problems which are similar in their description and completely different regarding their resolution.

Two classes of optimal path problems can be considered: the unconstrained path problems and the constrained ones. While in the former no constraints are considered, in the latter, the optimal path has to satisfy a given set of constraints, [1,11]. As an example of the constrained optimal path problem one can point out the classical travelling salesman problem, [9], in which only paths defined from one node to itself and passing through each node exactly one time can be considered as feasible solutions.

Among the unconstrained problems the classical shortest path and the maximum capacity path problems can be referred to as an example.

Different types of problems can also be considered; for example, ranking paths problems, [10, 14], and multiobjective path problems, [15]. In the first one, it is intended to determine not only the *best* path but also the *K best* paths by order of their values. In the multiobjective path problem a vectorial real function is given and a set of paths has to be determined, being all of them considered as optimal.

2. Notation and definitions

Let (N,A) denote a given network, in which $N = \{v_1,\ldots, v_n\}$ is a finite set composed of n elements known as nodes and $A = \{a_1,\ldots, a_m\} \subseteq N \times N$ is a finite set composed of m elements known as arcs. Sometimes i will simply be used to represent the node v_i. Each arc $a_k \in A$ can also be identified by a pair (i,j), where $i,j \in N$.

Let *i* and *j* be two nodes of (N,A); if all pairs (i,j) in (N,A) are ordered, i.e., if all the arcs of the network are directed, (N,A) is considered as a directed network; if all pairs (i,j) are not ordered, then (N,A) is an undirected network.

With no loss of generality, from now on, one will consider that (N,A) is a directed network; it will also be assumed that there is at most a single arc between each pair of nodes in the network and that there are no arcs of the form (i,i), in which $i \in N$. Notice that these assumptions do not imply some loss of generality, since any network is easily transformed in order to satisfy them.

Let s and t be two different nodes of (N,A), the initial and the terminal node respectively. A path *p* from the node *s* to the node *t* in (N,A) is an alternating sequence of nodes and arcs, which has the form $p = \langle s = v'_1, a'_1, v'_2,\ldots, a'_{\ell-1}, v'_\ell = t\rangle$, where $\ell \geq 2$, and:

- $v'_k \in N$, for any $k \in \{1,\ldots, \ell\}$;
- $a'_k \equiv (v'_k, v'_{k+1}) \in A$, for any $k \in \{1,\ldots, \ell-1\}$.

For convenience, sometimes a single node i is considered as a path ‹i›.

P_{ij} denotes the set of paths defined from i to j in (N,A), for any $i,j \in N$. In order to simplify, P will denote the set P_{st}. From now on, P_{si} and P_{it} are assumed to be nonempty whatever node i in (N,A). Once more, there is no loss of generality with these assumptions, since some node $i \in N$, for which $P_{si} = \emptyset$ or $P_{it} = \emptyset$, can be removed from the network.

A loopless path from s to t is a path from s to t where all the nodes are different and a loop (or cycle) is a path from a node to itself, where all nodes are different, except the first which is also the last.

Let $p \in P_{ux}$ be a path from u to x and $q \in P_{xv}$ a path from x to v. The concatenation of p and q, denoted by $p \Diamond q$, is a path from u to v formed by p until its node x, and followed by the path q from x to v. Given a cycle C, the k-cycle concatenation can be defined by $C^k = C^{k-1} \Diamond C$, with $k \geq 1$ and C^0 a single node of the cycle C.

A finite path is a path with a finite number of nodes (and arcs).

3. The optimal path problem

In the optimal path problem, it is given a directed network (N,A), in which $r \geq 1$ real values, $c_{ij}^1, \ldots, c_{ij}^r$, are associated with each arc $(i, j) \in A$. These values are denominated costs of (i,j). It will also be considered a real function $f : P \rightarrow \mathbb{R}$, denominated objective function (or cost function), in such a way that, for a given path p, f(p) depends on the costs associated with the arcs of p.

One intends to determine a path $p^* \in P^*$ ($P^* \subseteq P$) which optimizes a given function f in P^*, i.e., in such a way that $f(p^*) = \text{opt}\{f(p) : p \in P^*\}$. The path p^* is an optimal solution for such a problem.

Notice that $f : P \rightarrow \mathbb{R}$ is not defined, since its definition depends on the instance to be studied.

3.1 Boundness and finiteness

An optimal path problem is finite if there is an optimal path with a finite number of nodes; otherwise, the problem is said to be non-finite.

When an optimal path problem is non-finite, its optimal solution is considered to be the limit of a sequence of paths, which is associated with a sequence of objective function values converging to the optimal one.

An optimal path problem is bounded if for any optimal solution p^*, $f(p^*)$ is a finite value; otherwise, the problem is unbounded.

3.2 Examples of optimal path problems

In this section, three examples of optimal path problems will be presented. In the first two cases, it will be considered that r = 1, i.e., to each arc of the network is associated one single value; in the last one, r = 2.

3.2.1 The shortest path problem

The shortest path problem is a classical network optimization problem, which has been intensively studied since the fifty's. Hundreds of papers about this problem can be found in specialized literature; among them we call readers attention to [2,4,5,6,7,12,13,15].

It is associated to each arc (i,j) of (N,A) the real value $c_{ij} \in \mathbb{R}$, designated as cost or as distance of (i,j). It is still considered the objective function defined by

$$c : P \to \mathbb{R} : p \to c(p) = \sum_{(i,j)\in p} c_{ij} = \sum_{p} c_{ij}.$$

In the shortest path problem it is intended to determine a path p^* from s to t in (N,A) in such a way that $c(p^*)$ is minimum, i.e., $c(p^*) \leq c(p)$ holds for any path $p \in P$.

In Theorem 1, it will be proved that the shortest path problem is finite if and only if there are no negative cycles (cycles with negative cost) in the given network.

Theorem 1 *Let us assume that P_{si} and P_{it} are nonempty sets for any $i \in N$. The shortest path problem is finite if and only if $c(C) \geq 0$, for any cycle C of (N,A).*

Proof: Let C be a negative cycle, that is, $c(C) < 0$ and let i be a node of C. Since P_{si} and P_{it} are nonempty, there is p_1, a path from s to i, and p_2, a path from i to t, in (N,A). Denoting by $p_1 \Diamond C^0 \Diamond p_2$ the path $p_1 \Diamond p_2$, one concludes that $p_k = p_1 \Diamond C^k \Diamond p_2$, with $k \geq 0$, is a sequence of paths from s until t and $c(p_k) = c(p_1) + c(C^k) + c(p_2)$. It can be concluded that

$$\lim_{k\to\infty} c(p_k) = c(p_1) + c(C) \lim_{k\to\infty} k + c(p_2) = -\infty.$$

Then, the optimal path is $p_\infty = \lim_{k\to\infty} p_k$, which means that p_∞ is not finite. Since c_{ij} is finite for any arc (i,j), then the shortest path problem is not finite.

Let us consider now that $c(C) \geq 0$ for any cycle C of (N,A) and let p^* be an optimal path.

If p^* has no cycles then p^* is a loopless path; therefore it has no repeated nodes and it is finite, too.

Considering now that p^* has, at least, one cycle, then $p^* = p_1 \Diamond C \Diamond p_2$. If $c(C) > 0$, $c(p^*) > c(p_1 \Diamond p_2)$, then p^* would not be optimal. So, it can be concluded that $c(C) = 0$; but since $c(p^*) = c(p_1 \Diamond p_2)$ then $p_1 \Diamond p_2$ is also an optimal path, and p^* has one more cycle than $p_1 \Diamond p_2$. If this procedure is repeated over and over again, it is possible to remove all cycles from p^* and a loopless path will be obtained, which will still be an optimal solution to the problem. Once again, it could be concluded that this is a finite solution. ♦

From Theorem 1 the following results are immediate.

Corollary 1.1 *The shortest path problem is not finite if and only if it is unbounded.*

Proof: If the shortest path problem is not finite, then there is a negative cycle, C, in (N,A). Let one considers $p_\infty = \lim_{k\to+\infty} p_k$, with $p_k = p_1 \Diamond C^k \Diamond p_2$, in which $p_1 \in P_{si}$, $p_2 \in P_{it}$ and i is a node of C. Then $c(p_\infty) = -\infty$ and the shortest path problem is unbounded. If the shortest path problem is finite, then the optimal path has a finite number of nodes and arcs; therefore, it has a bounded value. ♦

Papers in the literature usually assume the non existence of negative cycles, or even that $c_{ij} \geq 0$ holds for any $(i,j) \in A$. From Theorem 1 and Corollary 1.1, it results immediatelly that, under this assumption, they need not study the boundness and finiteness of the shortest path problem.

Corollary 1.2 *There is a loopless path which is a shortest path problem optimal solution if and only if $c(C) \geq 0$, for any cycle C in (N,A).*

Notice that, even if there are no negative cycles in (N,A), there can still be non-finite optimal paths, as long as, it exists a cycle with zero cost with a node belonging to a shortest path, as it can be seen in Figure 1. In fact, the cost of the shortest path from s to t in (N,A) is 2, and there are several shortest paths in this network. Some of these paths are finite, for example $\langle s,1,t\rangle$, which is also a loopless path, while $\lim_{k\to+\infty} \langle s,1\rangle \Diamond C^k \Diamond \langle 1,t\rangle$, where $C = \langle 1,2,1\rangle$ is not finite, but it still has cost 2 since $c(C) = 0$.

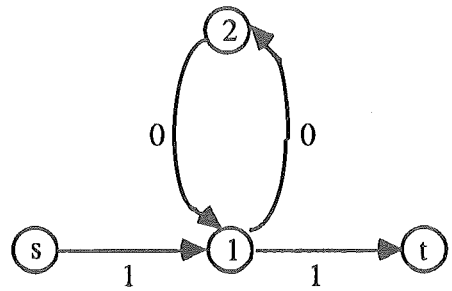


Figure 1 - Network (N,A)

3.2.2 The maximum capacity path problem

Concerning each arc (i,j) of (N,A), one can associate with it the value $u_{ij} \in \mathbb{R}^+$, which is designated as the capacity of the arc (i,j). The objective function is defined by

$$u : P \to \mathbb{R}^+ : p \to u(p) = \min_{(i,j)\in p} \{u_{ij}\} = \min_{p} \{u_{ij}\}.$$

In the maximum capacity path problem, [8], one intends to determine a path p^* from s to t in (N,A), in such a way that $u(p^*)$ is maximum, i.e., $u(p^*) \geq u(p)$ holds any path $p \in P$.

In Theorem 2, it will be proved that the maximum capacity path problem is always finite, and that this does not depend on the given network.

Theorem 2 *Let one assume that P_{si} and P_{it} are nonempty sets for any $i \in N$. The maximum capacity path problem is finite for any network (N,A).*

Proof: Let p^* be an optimal solution to the maximum capacity path problem in a given network (N,A). Then $u(p^*) \geq u(p)$ for any path $p \in P$.

If p^* does not contain cycles, then it is a loopless path and the problem is finite.

Let one assume now that p^* has, at least, one cycle, i.e., $p^* = p_1 \Diamond C \Diamond p_2$. Then:

$$u(p^*) = \min\{u(p_1),u(C),u(p_2)\} \leq \min\{u(p_1),u(p_2)\} = u(p_1 \Diamond p_2).$$

Since p^* is a maximum capacity path, then $u(p^*) = u(p_1 \Diamond p_2)$ and p^* has one more cycle than $p_1 \Diamond p_2$; so $p_1 \Diamond p_2$ is also an optimal path. If this procedure is repeated over and over again, it is

possible to remove all cycles from p^*, and thus to obtain an optimal solution that is a loopless path, which is, in its turn, finite. ♦

The following result is now immediate.

Corollary 2.1 *Under the assumptions of Theorem 2, there is a loopless path that is an optimal solution to the maximum capacity path problem.*

Notice that, similarly to the shortest path problem, non-finite optimal solutions to the maximum capacity path problem can be obtained.

3.2.3 The problem of the shortest path per unit of time

It is associated with each arc (i,j) of (N,A) a pair of values $(c_{ij},t_{ij}) \in \mathbb{R}\times\mathbb{R}^+$, denominated, respectively, the cost and the time of arc (i,j). In the problem of the shortest path per unit of time, one aims to determine a path that minimizes an objective function defined by

$$f : P \rightarrow \mathbb{R} : p \rightarrow f(p) = \frac{\sum_p c_{ij}}{\sum_p t_{ij}}.$$

As Figure 2 shows, this problem can be non-finite and bounded simultaneously.

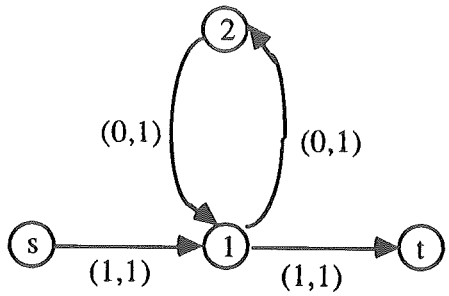


Figure 2 - Network (N,A)

In fact, the optimal solution for the problem of the shortest path per unit of time from s to t in (N,A) is $p_\infty = \lim_{k\rightarrow+\infty} p_k$, in which $p_k = \langle s,1\rangle\Diamond\langle 1,2,1\rangle^k\Diamond\langle 1,t\rangle$, and p_∞ is an non-finite path. However, as $c(p_k) = \frac{1}{1+k}$, path $p_\infty = \lim_{k\rightarrow\infty} p_k$ has finite cost since $c(p_\infty) = \lim_{k\rightarrow\infty} \frac{1}{1+k} = 0$.

4. The optimality principle

Some optimal path problems satisfy one or two principles which are known as the strong and as the weak optimality principles which will be stated bellow.

Strong optimality principle: *Every optimal path is formed by optimal subpaths.*

Weak optimality principle: *There is an optimal path formed by optimal subpaths.*

According to these principles, an optimal solution can be determined by finding successive optimal subsolutions. As a consequence, when an optimization problem satisfies the strong optimality principle, a labelling algorithm can be used to solve it. However, labelling algorithms can be applied only if there is an optimal path under those conditions, and they do really determine this optimal path. Then it can be concluded that if a problem satisfies the weak optimality principle, even so, it still can be solved using a labelling algorithm.

It can be easily noted that an optimal path problem which satisfies the strong optimality principle, also satisfies the weak optimality one.

Conditions are established in the next theorem in order to assure that the shortest path problem verifies these two principles (and therefore it can be solved by a labelling algorithm).

Theorem 3 *The shortest path problem satisfies the strong optimality principle if and only if there are no negative cycles in (N,A).*

Proof: Let one assume that there is a negative cycle C in (N,A). From the proof of Theorem 1, $p_\infty = \lim_{k\to+\infty} p_k$, in which $p_k = p_1 \Diamond C^k \Diamond p_2$ is an optimal solution to this problem. Then $q_k = p_1 \Diamond C^k$ is a subpath of p_∞, even though it is not an optimal subpath since $c(q_k) > c(q_{k+1})$, and then, in this case, the shortest path problem does not satisfy the strong optimality principle.

Let one now consider that $c(C) \geq 0$ for any cycle C in (N,A). Then the shortest path problem is finite and an optimal path, p^*, can be chosen which can be regarded as containing a non-optimal subpath, $q \in P_{uv}$; i.e., $c(p^*) \leq c(p)$ for any path $p \in P$, and $p^* = p_1 \Diamond q \Diamond p_2$. Let q^* be an optimal path from u until v; then $c(q^*) < c(q)$ because q is a non-optimal subpath. Considering now the path $w = p_1 \Diamond q^* \Diamond p_2 \in P$, once $c(w) < c(p^*)$, it can be concluded that p^* would not be an optimal solution as assumed. Consequently, p^* is formed by optimal subpaths.

♦

Corollary 3.1 *The shortest path problem satisfies the weak optimality principle if and only if there are no negative cycles in (N,A).*

Notice that the non existence of negative cycles in the network is a necessary and sufficient condition for the weak optimality principle to be satisfied by the shortest path problem.

It can also be noted that the maximum capacity path problem satisfies the weak optimality principle (and therefore it can be solved by a labelling algorithm).

Theorem 4 *The maximum capacity path problem satisfies the weak optimality principle.*

Proof: Let p^* be an optimal finite solution to the maximum capacity path problem, i.e., $u(p^*) \geq u(p)$ for any path $p \in P$. Based on Corollary 2.1, one can assume that p^* is a loopless path. Let one suppose that p^* has a subpath, $q \in P_{uv}$, which is not optimal, i.e., $p^* = p_1 \Diamond q \Diamond p_2$. There is then a maximum capacity path from u to v in (N,A), q^*, which is loopless and

$$u(p_1 \Diamond q^* \Diamond p_2) = \min\{u(p_1), u(q^*), u(p_2)\} \geq \min\{u(p_1), u(q), u(p_2)\} = u(p^*).$$

This means that $p_1 \Diamond q^* \Diamond p_2$ is also an optimal solution to this problem. If all the subpaths of q^* are optimal, the new path has one more optimal subpath than p^* and this process can be repeated until a path that contains only optimal subpaths is obtained. If q^* still contains non optimal subpaths it is also possible to remove them in a similar way. Once p^* is a loopless path and there is a finite number of

loopless paths in (N,A), this process is also finite, and when finished, an optimal path, containing only optimal subpaths, will be obtained. ♦

In general, the maximum capacity path problem does not satisfy the strong optimality principle which will be shown in the problem described in Figure 3.

The following figures exemplify these results. The first one, Figure 3, represents a network (N,A), in which a pair $(c_{ij},u_{ij}) \in \mathbb{R}\times\mathbb{R}^+$ is associated to each arc (i,j); that is, the first value of this pair represents the cost of (i,j) and the second one represents its capacity.

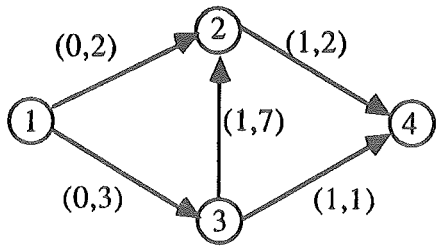


Figure 3 - Network (N,A)

The second figure, Figure 4, represents the tree containing all the paths that can be defined from 1 to 4 in (N,A). A pair, in which the first element represents the cost of the path in the tree from s to i and the second its capacity, is associated with each node i in this tree.

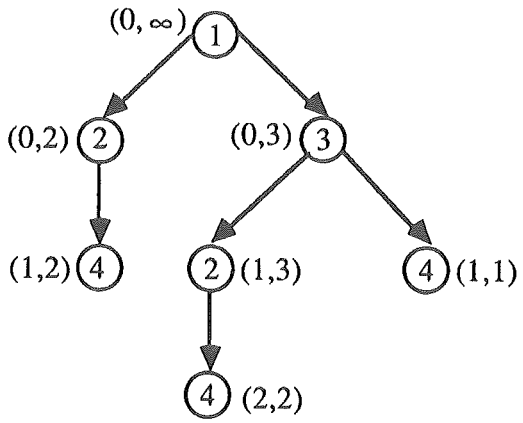


Figure 4 - Tree of the paths from 1 to 4 in (N,A)

In this network, the shortest path problem has two optimal solutions, namely ‹1,2,4› and ‹1,3,4›. As it can easily be verified, all subpaths of this two paths are also optimal paths.

There are also two optimal solutions to the maximum capacity path in (N,A), namely paths ‹1,2,4› and ‹1,3,2,4›. Considering the first one, its subpath from 1 until 2 is ‹1,2›, which has 2 as capacity, while the capacity of ‹1,3,2› is 3. Consequently, there is one optimal solution that is not formed only by optimal subpaths. Every subpath of the optimal path ‹1,3,2,4›, however, is optimal.

Finally, Figure 5 represents the tree containing all paths in (N,A) from 1 to 4, where it is associated with each node i the cost per unit of time - considering u_{ij} as the time for traversing (i,j) from i to j - concerning the path in the tree from s until i.

The path ‹1,3,2,4› is the only optimal solution for the problem of the shortest path per unit of time. Its subpath from 1 until 2 is ‹1,3,2›; however, the optimal path between those two nodes is ‹1,2›.

As a result, the unique optimal solution for this problem contains one subpath that is not optimal, which means that the problem of the shortest path per unit of time does not satisfy neither the strong nor the weak optimality principle.

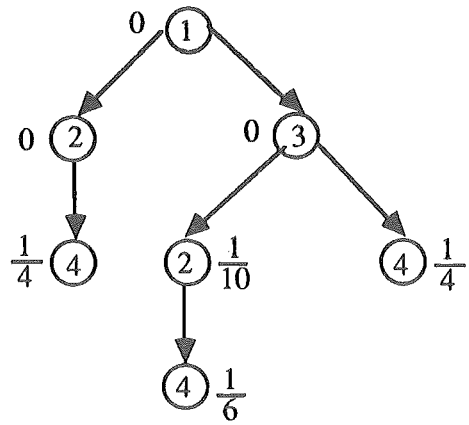


Figure 5 - Tree of the paths from 1 to 4 in (N,A)

4.1 Labelling algorithms

From now on only finite problems will be considered since, only in this case, it is possible to determine computationally an optimal path; moreover, the problem must satisfy the weak optimality principle. In fact, as remarked before, there is a class of algorithms, known as labelling algorithms, which can solve some optimal path problems. These algorithms determine a tree rooted at s, where the path from s until i is optimal for any node $i \in N$. This tree is named optimal tree rooted at s and contains the optimal path from s until t that one have the purpose to determine.

These algorithms construct this tree by determining optimal subpaths. With this purpose in mind, there is a label, π_i, associated with each node $i \in N$, that represents the value of the *best* path determined from s to i at certain moment. This path will be denoted by p_i^*. Furthermore, it is also used a set X, containing nodes of (N,A) with labels that may be improved[1].

To begin with, the set X contains only the initial node, s. The algorithm finishes when there are no more labels to improve, i.e., when X is empty. Whenever a node i is picked out from X and then analysed, the labels of other nodes in (N,A) may be improved and, in this case, placed in X.

After the execution of the algorithm, p_i^* stands for the optimal path from s to i and π_i^* stands for its value.

In the beginning of the algorithm, for any $i \in N-\{s\}$, π_i^* should be a value that could be improved, i.e., should be an upper or lower bound for all the possible values that π_i^* could

[1] Note that the concept of *best* depends on the optimal path problem that is being considered.

assume, which would depend on the optimization problem. Thus, in the shortest path problem, π_i^* is initialized to $+\infty$, while in the maximum capacity path problem π_i^* is initialized to 0 for any $i \in N-\{s\}$. The node s is initialized to the value $f(\langle s \rangle)$, i.e., to the optimal value of a path from s to s.

Notice that labelling algorithms can be easily modified to handle non-finite problems, that is, to determine when a given instance of the optimal path problem is non-finite, [3].

This class of algorithms is described in Algorithm 1 and its validity is proved by Theorem 5. Since the labelling algorithm can be modified to handle non-finite instances of the optimal path problem, there is no loss of generality if finiteness of the problem is assumed.

Theorem 5 *Let us assume that the statement of the weak optimality principle is satisfied by an optimal path problem. Thus, labelling algorithms determine an optimal path.*

Proof: As it was assumed, there is a finite optimal path p^* and, as a consequence, the labelling algorithm is also finite. Let us assume that p^* is not determined by a labelling algorithm, and let $j \in N$ be the first node of p^*, for which the value of π_j determined by that algorithm is greater than π_j^*, the value of the optimal subpath p_j^* of p^*, from s to j. Notice that there exists such a node j in p^*, since the optimal path was not determined by the algorithm. Moreover, j is not the initial node s, since, by definition, π_s is initialized with the value of the optimal path from s to s. Let $i \in N$ be the preceding node of j in the optimal path p^*. Notice that i exists, since $j \neq s$. Since π_i obtained by the algorithm is correct, it can be concluded that i has been placed in X. So, when i was picked out from X, node j was labelled with $f(p_i^* \Diamond \langle i,(i,j),j \rangle)$ which is the value of p_j^*, the optimal subpath of p^* from s to j. This contradicts the assumption made. ♦

Labelling algorithms can be divided in two subclasses, namely, the label correcting and the label setting algorithms.

4.1.1 Label correcting algorithms

The labelling algorithms can be implemented in several different ways, which one depending on the data structure used to represent the set X. This structure determines the node to be removed at each step of the algorithm and how X is manipulated.

When the node i is picked out from X in a non-specified way, its label can be improved later on and, consequently, i can be placed again in X. In this case, the algorithm is a label correcting one. Note that the description of the general labelling algorithms and that specific one of the label correcting algorithms is coincident.

As referred before, several versions of these algorithms can be implemented by use of different structures for X. (See [4,5] for details)

Algorithm 1: *Labelling (correcting) algorithm for the optimal path problem*

For ($i \in N-\{s\}$) Do $\pi_i^* \leftarrow \pm\infty$;
$p_s^* \leftarrow \langle s \rangle$;
$\pi_s^* \leftarrow f(\langle s \rangle)$;
$X \leftarrow \{s\}$
While ($X \neq \emptyset$) Do
  $i \leftarrow$ node of X;
  $X \leftarrow X - \{i\}$;
  For ($(i,j) \in A$ such that $f(p_i^* \Diamond \langle i,(i,j),j \rangle)$ is better than π_j^*) Do
    $p_j^* \leftarrow p_i^* \Diamond \langle i,(i,j),j \rangle$;
    $\pi_j^* \leftarrow f(p_i^* \Diamond \langle i,(i,j),j \rangle)$;
    If ($j \notin X$) Then $X \leftarrow X \cup \{j\}$;
  EndFor
EndWhile

4.1.2 Label setting algorithms

When a node is picked out from X the labels of several other nodes can be *improved.* Therefore it is natural to think of choosing the element in X which has the *best* cost at each step of the labelling algorithm. In this way, instead of just removing an element of X, an element i will be removed in such a way that π_i^* is better than π_j^* for any $j \in X$.

A label setting algorithm is one in which each node is chosen in this manner. This means that, once a node is picked out from set X, its label is permanent, i.e., it will not be improved again. In order to assure that this will happen, it will be assumed that all the nodes labelled from i have a label that is not *better* than π_i^*. In Theorem 6, it will be proved that, in this case, the label of every node removed from X is permanent.

Only when the conditions of Theorem 6 are satisfied, label setting algorithms can be used; thus, they are less general than label correcting ones. For instance, it is well known that label setting algorithms for the shortest path problem are valid if and only if $c_{ij} \geq 0$ holds for any arc $(i,j) \in A$, [6].

Theorem 6 *Let one assume that the label of every node obtained from a node $i \in N$ is not better than the label of i. Then, whenever a node is picked out from the set X, its label is permanent when a label setting algorithm is used.*

Proof: Let one assume that i is a node removed from X and that its label is not permanent, i.e., there is a node j picked out from X, in such a way that, π_i^* can be improved from π_j^*.

If j is an element of X when i had been chosen in X, then π_j^* will not be *better* than π_i^*. Otherwise, if j is not an element of X when i has been chosen in this set, then j will be labelled, directly or indirectly, from some node x which belonged to

X in that moment; that is, π_x^* will not be *better* than π_i^*, which proves the theorem. As a consequence, the label of i cannot be improved, that is, the node i has a permanent label. ♦

Algorithm 2: *Label setting algorithm for the optimal path problem*

For ($i \in N-\{s\}$) Do $\pi_i^* \leftarrow \pm\infty$;
$p_s^* \leftarrow \langle s \rangle$;
$\pi_s^* \leftarrow f(\langle s \rangle)$;
$X \leftarrow \{s\}$
While ($X \neq \emptyset$) Do
  $i \leftarrow$ node of X such that π_i^* is *better* than π_j^*, for every $j \in X$;
  $X \leftarrow X - \{i\}$;
  For ($(i,j) \in A$ such that $f(p_i^* \Diamond \langle i,(i,j),j \rangle)$ is *better* than π_j^*) Do
    $p_j^* \leftarrow p_i^* \Diamond \langle i,(i,j),j \rangle$;
    $\pi_j^* \leftarrow f(p_i^* \Diamond \langle i,(i,j),j \rangle)$;
    If ($j \notin X$) Then $X \leftarrow X \cup \{j\}$;
  EndFor
EndWhile

5. Two optimal path problems

In this section, two problems will be presented in order to reinforce the importance of the optimality principle in the resolution of optimal path problems.

Although being similar in their description, these problems are completely different in what regards the determination of the optimal solution. They are based on two of the problems presented before - the shortest path problem and the maximum capacity path problem. In these two problems, one aims to determine a path from s to t which is optimal in a given subset of P.

As referred above, although both problems - the shortest path and the maximum capacity path problem - satisfy the weak optimality principle, only the first one satisfies the strong optimality principle. In the following sections, it will be proved how this property has influence in the fact that the problems which will be described will satisfy or not these principles.

5.1 The problem of the maximum capacity path in the set of the shortest paths

Let P^* denote the set of the shortest paths from s to t, i.e., $P^* = \{p^* \in P : c(p^*) \leq c(p), \forall p \in P\}$. It is immediate that $P^* \subseteq P$.

In the problem of the maximum capacity path in the set of the shortest paths, it is intended to determine a maximum capacity path in P^*, i.e., it is intended to determine a path $\bar{p} \in P^* \subseteq P$ in such a way that $u(\bar{p}) \geq u(p)$ for any $p \in P^* \subseteq P$. The path $\bar{p}$ is a shortest path from s to t and it is also a maximum capacity path among all the paths of P^*.

Notice that, as the maximum capacity path problem is finite, this problem is also finite since there is a finite shortest path. In this case, there is also an optimal path which is loopless.

Theorem 7 *Let one assume that P_{si} and P_{it} are nonempty sets for any $i \in N$ and that the shortest path problem is finite. The problem of the maximum capacity path in the set of the shortest paths in finite.*

Proof: Let $\bar{p}$ be an optimal path to the problem of the maximum capacity path in the set of the shortest paths, that is, $\bar{p} \in P^*$ and $u(\bar{p}) \geq u(p)$ for any $p \in P^*$.

If $\bar{p}$ does not contain cycles, then it is a loopless path and this problem has a finite optimal solution.

Let one now assume that $\bar{p}$ has a cycle C, i.e., $\bar{p} = p_1 \Diamond C \Diamond p_2$. Once the shortest path problem is finite, $c(C) \geq 0$, and then

$$c(\bar{p}) = c(p_1) + c(C) + c(p_2) \geq c(p_1 \Diamond p_2).$$

Since $\bar{p}$ is a shortest path $c(\bar{p}) = c(p_1 \Diamond p_2)$, and then $p_1 \Diamond p_2 \in P^*$. Besides,

$$u(\bar{p}) = \min\{u(p_1), u(C), u(p_2)\} \leq u(p_1 \Diamond p_2).$$

Analogously, once $\bar{p}$ is a maximum capacity path in P^*, then $u(\bar{p}) = u(p_1 \Diamond p_2)$. Since $p_1 \Diamond p_2$ has less cycles than $\bar{p}$, it can be concluded that it is possible to remove all the cycles in $\bar{p}$, obtaining then an optimal loopless path. Thus, the problem of the maximum capacity path in the set of the shortest paths is finite. ♦

Similarly to other problems, this theorem leads to Corollary 7.1.

Corollary 7.1 *Under the assumptions of Theorem 7, there is a loopless path that is an optimal solution to the problem of the maximum capacity path in the set of the shortest paths.*

It can now be proved that the problem of the maximum capacity path in the set of the shortest paths satisfies the weak optimality principle.

Theorem 8 *Let (N,A) be a network without negative cycles. The problem of the maximum capacity path in the set of the shortest paths satisfies the weak optimality principle.*

Proof: Let one considers a network (N,A) without negative cycles. Then the shortest path problem in (N,A) is finite and satisfies the strong optimality principle. Now, one will prove that there is a maximum capacity path in P^* for which all subpaths are also optimal.

Let $\bar{p}$ be a maximum capacity loopless path in the set of all the shortest paths, i.e., $u(\bar{p}) \geq u(p)$ for any $p \in P^*$. Let one assume that $\bar{p} = p_1 \Diamond q \Diamond p_2$, where $q \in P_{uv}$ is not an optimal path. Since the shortest path problem satisfies the strong optimality principle, q is a shortest path from u to v in (N,A), i.e., $q \in P^*_{uv}$. Then, there is another shortest loopless path, $\bar{q} \in P^*_{uv}$, which has maximum capacity in that set, i.e., $c(\bar{q}) = c(q)$ and $u(\bar{q}) \geq u(q')$ for any $q' \in P^*_{uv}$. As a result, one can consider a path from s to t in (N,A), $p_1 \Diamond \bar{q} \Diamond p_2$. This is a shortest path since

$$c(p_1 \Diamond \bar{q} \Diamond p_2) = c(p_1) + c(\bar{q}) + c(p_2) = c(p_1) + c(q) + c(p_2) = c(\bar{p});$$

$p_1 \Diamond \bar{q} \Diamond p_2$ is also a maximum capacity path in P^* since $\bar{p}$ is an optimal loopless path and

$$u(p_1 \Diamond \bar{q} \Diamond p_2) = \min\{u(p_1),u(\bar{q}),u(p_2)\} \geq \min\{u(p_1),u(q),u(p_2)\} = u(\bar{p}).$$

It is then possible to remove from $\bar{p}$ every subpath that is not optimal and once the number of loopless paths in (N,A) is finite, this process is also finite, allowing one to obtain an optimal path containing only optimal subpaths. ♦

As the problem of the maximum capacity path in the set of the shortest paths satisfies the weak optimality principle, it can thus be solved by a labelling algorithm, after taking into account certain adaptations. These adaptations are presented in Algorithm 3. Note that whenever a tie for the cost of two paths from s to i happens, the path with better capacity is chosen.

Algorithm 3: *Algorithm for the maximum capacity path problem in the set of the shortest paths*

For ($i \in N-\{s\}$) Do $\pi_i^* \leftarrow (+\infty,-\infty)$;
$p_s^* \leftarrow \langle s \rangle$;
$\pi_s^* \leftarrow (0,+\infty)$;
$X \leftarrow \{s\}$;
While ($X \neq \emptyset$) Do
  $i \leftarrow$ node of X;
  $X \leftarrow X - \{i\}$;
  For ($(i,j) \in A$) Do
    If ($\pi_j^1 > \pi_i^1 + c_{ij}$) Then
      If ($j \notin X$) Then $X \leftarrow X \cup \{j\}$;
      $p_j^* \leftarrow p_i^* \Diamond \langle i,(i,j),j \rangle$;
      $\pi_j^* \leftarrow (\pi_i^1 + c_{ij}, \min\{\pi_i^2, u_{ij}\})$;
    Else
      If ($\pi_j^1 = \pi_i^1 + c_{ij}$ and $\pi_j^2 < \min\{\pi_i^2, u_{ij}\}$) Then
        If ($j \notin X$) Then $X \leftarrow X \cup \{j\}$;
        $p_j^* \leftarrow p_i^* \Diamond \langle i,(i,j),j \rangle$;
        $\pi_j^2 \leftarrow \min\{\pi_i^2, u_{ij}\}$;
      EndIf
    EndIf
  EndFor
EndWhile

One can still consider the network (N,A), represented in Figure 3, and the tree containing all its paths from 1 to 4, Figure 4. After using Algorithm 3, it can be obtained the tree represented in Figure 6.

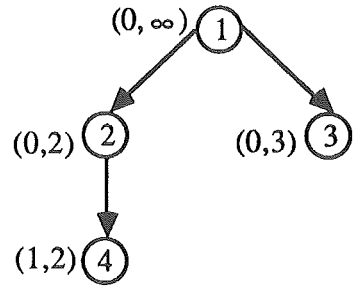


Figure 6 - Tree obtained with a labelling algorithm for the maximum capacity path problem in the set of the shortest paths

5.2 The problem of the shortest path in the set of the maximum capacity paths

In this section, $\overline{P}$ will be used to denote the set of the maximum capacity paths from s to t (once more $\overline{P} \subseteq P$).

In the problem of the shortest path in the set of the maximum capacity paths, one aims to determine a shortest path in the set $\overline{P}$, i.e., a path $p^* \in \overline{P} \subseteq P$, in such a way that $c(p^*) \leq c(p)$ for any path $p \in \overline{P} \subseteq P$. Then p^* is a maximum capacity path and it is thus, the shortest path among all the paths of $\overline{P}$. However, since the maximum capacity path does not satisfy the strong optimality principle (but only the weak optimality one), it is not possible to assure that all the subpaths in p^* have maximum capacity. This is the reason why, in general, the problem of the shortest path in the set of the maximum capacity paths does not satisfy the weak optimality principle.

As an example the network represented in Figure 3 and the tree of all the paths from 1 to 4 in (N,A) showed in Figure 4 can be used again. The path $p^* = \langle 1,2,4 \rangle$ is the unique optimal solution to this problem; however, its subpath $\langle 1,2 \rangle$, with cost 0 and capacity 2, is not an optimal path from 1 to 2, since $\langle 1,3,2 \rangle$ has a higher capacity, 3.

Since this problem does not satisfy neither the strong nor the weak optimality principle, then it can not be solved with a labelling algorithm. This is shown in Figure 7, which represents the tree constructed by a labelling algorithm used to determine the shortest path problem in the set of the maximum capacity paths from 1 to 4 in the network (N,A).

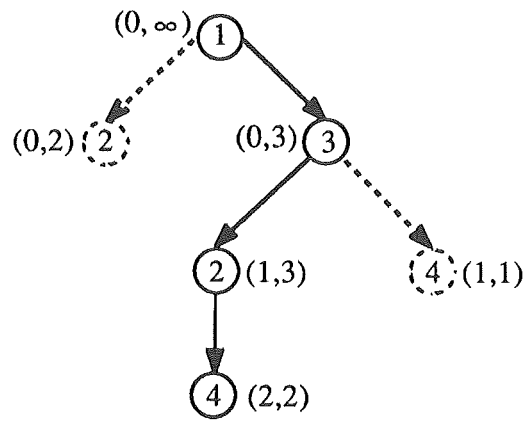


Figure 7 - Tree obtained with a labelling algorithm, analogous to Algorithm 3 for the shortest path problem in the set of the maximum capacity paths

Although it cannot be solved by a labelling algorithm, the problem of the shortest path in the set of the maximum capacity paths can be solved easily such as follows.

Let $\bar{u} = u(\bar{p})$, in which $\bar{p} \in P$ is a maximum capacity path in (N,A) from s to t.

It can be noted that the arcs in (N,A), whose capacity is lower than $\bar{u}$, will not be a part of any optimal solution to this problem. Therefore, after the value $\bar{u}$ is known, those arcs of (N,A) can be ignored. This is stated in Theorem 9.

Theorem 9 *Let $\bar{u} = u(\bar{p})$, in which $\bar{p} \in \bar{P}$, and let p be other maximum capacity path in (N,A) from s to t. Then $u_{ij} \geq \bar{u}$ for any arc $(i,j) \in p$.*

Proof: Let $p \in \bar{P}$ and let one assume that $p = p_1 \Diamond \langle i,(i,j),j \rangle \Diamond p_2$, where $u_{ij} < \bar{u}$. Then

$$u(p) = \min\{u(p_1), u_{ij}, u(p_2)\} < \bar{u},$$

and p would not be a maximum capacity path in (N,A) as it was assumed. ♦

Based on this theorem, it can be concluded that the shortest path in $\bar{P}$ does not contain any arc with a capacity lower than $\bar{u}$. Moreover, let $\bar{A}$ be the set obtained from A removing these arcs, and let $\bar{P}(\bar{A})$ denote the set of the maximum capacity paths from s to t in $(N,\bar{A})$. The following result, which asserts that $\bar{P} = \bar{P}(\bar{A})$, is immediate and it will be presented without proof.

Theorem 10 *The set of the maximum capacity paths from s to t in (N,A) coincides with the set of the maximum capacity paths from s to t in $(N,\bar{A})$, in which $\bar{A} = \{(i,j) \in A : u_{ij} \geq \bar{u}\}$.*

If the initial network, (N,A), is changed according to Theorem 10 it is then possible to determine the shortest path in $\bar{P}$, which is simply the shortest path from s to t in $(N,\bar{A})$. Therefore, this problem can be solved in three phases:

1. Determining $\bar{u}$, the maximum capacity of any path from s to t in (N,A).
2. Removing from (N,A) the arcs with capacity lower than $\bar{u}$.
3. Determining the shortest path from s to t in the resulting network.

These three phases are resumed and presented in Algorithm 4.

Algorithm 4: *Algorithm for the shortest path problem in the set of the maximum capacity paths*

For ($i \in N-\{s\}$) Do $\pi_i \leftarrow -\infty$;
$p_s^* \leftarrow \langle s \rangle$;
$\pi_s \leftarrow +\infty$;
$X \leftarrow \{s\}$;
While ($X \neq \emptyset$) Do
  $i \leftarrow$ node of X;
  $X \leftarrow X - \{i\}$;
  For ($(i,j) \in A$ such that $\pi_j < u(p_i^* \Diamond \langle i,(i,j),j \rangle)$) Do
    $p_j^* \leftarrow p_i^* \Diamond \langle i,(i,j),j \rangle$;
    $\pi_j \leftarrow u(p_i^* \Diamond \langle i,(i,j),j \rangle) = \min\{\pi_i, u_{ij}\}$;
    If ($j \notin X$) Then $X \leftarrow X \cup \{j\}$;
  EndFor
EndWhile
$u^* \leftarrow \pi_t$;
For ($i \in N-\{s\}$) Do $\pi_i \leftarrow +\infty$;
$p_s^* \leftarrow \langle s \rangle$;
$\pi_s \leftarrow 0$;
$X \leftarrow \{s\}$;
While ($X \neq \emptyset$) Do
  $i \leftarrow$ node of X;
  $X \leftarrow X - \{i\}$;
  For ($(i,j) \in A$ such that $u_{ij} \geq u^*$ and $\pi_j > c(p_i^* \Diamond \langle i,(i,j),j \rangle)$) Do
    $p_j^* \leftarrow p_i^* \Diamond \langle i,(i,j),j \rangle$;
    $\pi_j \leftarrow c(p_i^* \Diamond \langle i,(i,j),j \rangle) = \pi_i + c_{ij}$;
    If ($j \notin X$) Then $X \leftarrow X \cup \{j\}$;
  EndFor
EndWhile

Acknowledgments

We acknowledge the referee for criticism and comments, specially on the second draft of the paper.

Research developed within CISUC ("Centro de Informática e Sistemas da Universidade de Coimbra") and partly supported by the Portuguese Ministry of Science and Technology (MCT), under PRAXIS XXI Project of JNICT ("Junta Nacional de Investigação Científica e Tecnológica").

References

[1] Aneja, Y.P. and Nair, K.P.K., The constrained shortest path problem, *Naval Research Logistics Quarterly* 25 (1978) 549-555.
[2] Bellman, R.E., On a routing problem, *Quarterly Applied Mathematics* 1 (1958) 425-447.
[3] Cherkassky, B.V. and Goldberg, A.V., Negative-cycle detection algorithm, Research Report, NEC Research Institute, Princeton, New Jersey (1996).
[4] Cherkassky, B.V., Goldberg, A.V. and Radzik, T., Shortest paths algorithms: theory and experimental evaluation, *Mathematical Programming* 73 (1996) 129-196.
[5] Denardo, E.V. and Fox, B.L., Shortest route methods: reaching, pruning and bruckects, *Operations Research* 27 (1979) 161-186.
[6] Dijkstra, E., A note on two problems in connection with graphs, *Numerical Mathematics* 1 (1959) 395-412.
[7] Gallo, G. and Pallotino, S., Shortest path methods: a unifying approach, *Mathematical Programming Study* 26 (1986) 38-64.
[8] Hansen, P., Bicriterion path problems, in *Multiple Criteria Decision Making: Theory and Applications*, editors: G.Fandel and T.Gal, *Lectures Notes in Economics and Mathematical Systems* 177 (1980) 109-127, Springer Heidelberg.
[9] Lawler, E.L., Lenstra, J.K. and Rinnooy Kan, A.H.G. (eds.), *The Traveling Salesman Problem*, John Wiley & Sons, Inc. (1983).
[10] Martins, E.Q.V., Pascoal, M.M.B. and Santos, J.L.E., *The k shortest paths problem*, submited (1998).
[11] Masch, V.V., The cyclic method of solving the transshipment problem with an additional constraint, *Networks* 10 (1980) 17-31.
[12] Moore, E.F., The shortest path through a maze, *Proceedings of the International Symposium on the Theory of Switching* (1959) 285-292, Harvard University Press.
[13] Pape, U., Implementation and efficiency of Moore algorithms for the shortest root problem, *Mathematical Programming* 7 (1974) 212-222.
[14] Pascoal, M.M.B., *Algoritmos para a enumeração dos k trajectos mais curtos*, Dissertação de Mestrado, Departamento de Matemática, Universidade de Coimbra (1997).
[15] Santos, J.L.E., *O problema do trajecto óptimo multiobjectivo*, Dissertação de Mestrado, Departamento de Matemática, Universidade de Coimbra (1997).

INTEGER SOLUTIONS OF MULTICRITERIA NETWORK FLOW PROBLEMS

Mathias Ehrgott

Fachbereich Mathematik
Universität Kaiserslautern
67663 Kaiserslautern
Germany

Abstract

This paper is concerned with the solution of integer multicriteria network flow problems. The single criterion network flow problem and a sketch of the out-of-kilter method are presented. Important results from (linear) multicriteria optimization are stated and their importance for network flow problems discussed. The main topic of the paper is the presentation of a method to solve integer multicriteria network flow problems. The set of all efficient solutions in objective space is determined in two steps: first the step of all maximal efficient faces of the linear relaxation is determined. The second step consists in finding all integer efficient points. Finally the lexicographic max-ordering theory is proposed to choose a compromise solution.

Keywords

Network flow problems, multicriteria optimization, integer programming.

1. The Network Flow Problem

The general formulation of the (minimum cost) network flow problem is the following:

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij}$$

$$\text{subject to } \sum_{j\in N} x_{ij} - \sum_{j\in N} x_{ji} = 0 \quad \forall i\in N \qquad \text{(NFP)}$$

$$x_{ij} \geq l_{ij} \quad \forall (i,j)\in A$$

$$x_{ij} \leq u_{ij} \quad \forall (i,j)\in A$$

Here, D = (N,A) is a network with node set N, arc set A and variable x_{ij} representing flow along arc $(i,j)\in A$. Many types of algorithms are available for the solution of this problem. Below we will outline the idea of the out-of-killer algorithm, upon which the method for the multicriteria network flow problem developed later is based.

Using dual variables Π_i for the nodes $i\in N$, reduced costs for the arcs are defined as $\bar{c}_{ij} = \Pi_i - \Pi_j - c_{ij}$. From complementary slackness conditions of linear programming it is

straightforward to derive the optimality conditions for the network flow problem: A feasible flow x is optimal if for each arc (i,j) $\bar{c}_{ij} = 0$,

$$x_{ij} = l_{ij} \quad \text{if} \quad \bar{c}_{ij} < 0, \text{ or}$$
$$x_{ij} = u_{ij} \quad \text{if} \quad \bar{c}_{ij} > 0$$

(Nonbasic) variables satisfying these conditions are called "in kilter". The out-of-kilter algorithm is based on the idea of bringing all edges in kilter in order to obtain an optimal solution of the NFP. Because we use the ideas of this algorithm throughout the rest of this paper, we will give a sketch of it now, for details see e.g. [Bazaraa and Jarvis, 1977].

Sketch of the out-of-kilter algorithm

1 Let $x := 0$ and $\Pi := 0$
2 If there is an "out-of-kilter" edge (u,v) goto 3, else STOP: x is an optimal flow
3 Primal phase:
If possible change flow along a cycle containing (u,v), else goto 4
4 Dual phase:
Change dual variables Π and goto 2

In this paper we will discuss the solution of multicriteria integer (NFP). Integrality is an issue that does not create any difficulty in the single objective case: due to the total unimodularity of the constraint matrix of the (NFP) (the node-arc incidence matrix of the network N), it is well known that all basic feasible solutions of NFP are integral. Thus, solving the integer (NFP) is equivalent to solving the linear (NFP). In Section 3 below we will see why the integrality requirement will make the multicriteria NFP much more difficult to solve.

For this paper we shall assume that all problems are non-degenerate. I.e. for all basic feasible solutions, we do neither have $x_{ij} = l_{ij}$ or $x_{ij} = u_{ij}$ for basic variables (primal degeneracy), nor $\bar{c}_{ij} = 0$ for nonbasic variables (dual degeneracy). We will also assume that (NFP) is feasible and (w.l.o.g.) that $l_{ij} \geq 0$ and $u_{ij} < \infty$. Thus the feasible set of (NFP) will be a compact set. Assuming nondegeneracy will avoid technical complications, e.g. in Lemma 4. However, the method can be modified to allow degenerate solutions.

2. Multicriteria (Linear) Optimization

In this section we will briefly summarize the basic definitions and some important results in (linear) multicriteria optimization. Because the multicriteria NFP is a special case, we will refer to these results later.

A linear multiobjective program is given by

$$\begin{aligned} \min \quad & Cx \\ \text{subject to} \quad & Ax = b \\ & x \geq 0 \end{aligned} \qquad \text{(MOLP)}$$

Here, C denotes a Q×n objective or criteria matrix, A is an m×n constraint matrix of full row rank. The right hand side vector is $b \in \mathbb{R}^m$ and $x \in \mathbb{R}^n$ is the vector of variables. Individual

objectives (rows of C) are denoted by C^q. As usual, we will denote the set of feasible solutions of (MOLP) by

$$X = \{x \in \mathbb{R}^n : Ax = b, x \geq 0\}.$$

The following is the fundamental definition of optimality in multicriteria optimization, replacing the single objective notion of minimality.

Definiton 1

- *$x^0 \in X$ is called Pareto optimal if there is no $x \in X$, $x \neq x^0$ such that $Cx \leq Cx^0$ componentwise, with strict inequality for at least one component.*
- *If $x^0 \in X$ is Pareto optimal, then the corresponding image point $y^0 = Cx^0$ is called efficient.*

We will here consider the case of a compact feasible set X, as we encounter in the (NFP), due to our assumptions on the bounds for arc capacities, l_{ij} and u_{ij}. We shall also use the notation Y = CX for the image of the feasible set in criterion space.

The set of efficient points in Y is often referred to as the efficient frontier, a notion we shall adhere to, too. It is well known that the efficient frontier is indeed located on the boundary of Y. Figure 1 depicts Y and the efficient frontier (solid lines) for an (MOLP). Many authors have investigated (MOLP). We will now summarize, without proofs, some of the most important results in the area, which are relevant for this paper.

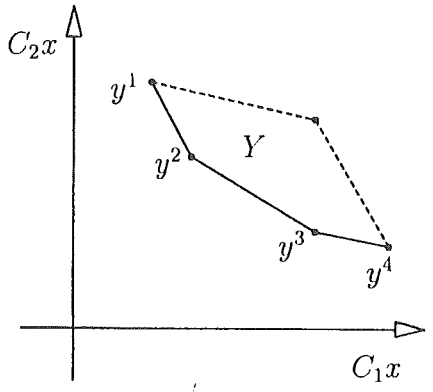


Figure 1 - Feasible set in objective space and efficient frontier

Theorem 1 ([Geoffrion, 1968, Isermann, 1974])

If x^0 is a Pareto optimal solution of (MOLP) then there exists some $\lambda \in \mathbb{R}^Q$, $0 < \lambda_q < 1$,

$\sum_{q=1}^{Q} \lambda_q = 1$, *such that*

$$\lambda^T Cx^0 \leq \lambda^T Cx \quad \forall x \in X.$$

In other words, all Pareto optimal points can be found by solving weighted sum scalarizations of (MOLP) with appropriate positive weights. The reader can easily visualize this result from the drawing in Figure 1.

The next results refer to topological properties of the efficient frontier, and the set of Pareto optimal solutions (the preimage of the efficient frontier under C).

Theorem 2 ([Naccache, 1978])

The efficient frontier of Y is connected.

Theorem 3 ([Warburton, 1983])

The set of Pareto optimal solutions of (MOLP) is connected.

These results are very important as they imply that neither in decision nor in objective space we can have isolated Pareto optimal/efficient solutions. It is important to note that these results are not true for general (nonconvex) objectives. Theorems 2 and 3 emphasize the continuous nature of linear programming. However, due to basic theory of linear programming and the widespread use of simplex type algorithms, basic feasible solutions are the ones which are most important. For compact X this is a finite set. Thus linear programming is on the boundary between continuous and combinatorial optimization. The combinatorial counterpart of Theorem 3 is the following result, due to Isermann.

Theorem 4 ([Isermann, 1977])

For an (MOLP) define a graph G = (N,E) such that the set of nodes N represents the set of Pareto optimal basic feasible solutions. The set of edges E is defined in such a way that an edge exists between two nodes if and only if the corresponding basic solutions can be obtained from each other by one single pivot step. Then G is connected.

Note that G is naturally an undirected graph. The important implication of Theorem 4 is, that the whole continuum of Pareto optimal solutions can be generated by exploring basic feasible solutions alone, since Pareto optimal basic feasible solutions represent extreme points of the feasible set in decision space, Y.

There is a drawback, however. Experiments recently reported by Benson [Benson, 1997] indicate that even for medium sized problems ((MOLP) with 4 objectives, 50 variables and 50 constraints) the number of Pareto optimal basic feasible solutions may be prohibitively large (80,000 on average). It may well be demanding too much of decision makers to choose among these. Therefore, the need for compromise solutions is evident.

But how to choose a compromise solution? We propose the following lexicographic max-ordering approach. The reasons will be illustrated below.

Definition 2

- $x^0 \in X$ *is called max-ordering optimal if*

$$\max_{1 \le q \le Q} (C^q)^T x^0 \le \max_{1 \le q \le Q} (C^q)^T x$$

for all $x \in X$.

- *For* $y \in \mathbb{R}^Q$ *let* $sort(x) = (sort_1(x), \ldots, sort_Q(x))$ *be such that* $sort_1(x) \ge sort_2(x) \ge \ldots \ge sort_Q(x)$. $x^0 \in X$ *is called lexicographic max-ordering (lex-MO) optimal if* $sort(Cx^0) \le_{lex} sort(Cx)$ *for all* $x \in X$, *where* $\le_{lex}$ *denotes the lexicographic order.*

In other words, we first choose a solution which minimizes the worst objective (some kind of "conservative planning" idea). But, there may be many such max-ordering solutions, and most of them need not be Pareto optimal. The idea of lexicographic max-ordering is to iterate this idea to the second worst, third worst objective, etc.

Why does this define a reasonable compromise solution? The answer is given in Theorem 5 from [Ehrgott, 1977b] (see also [Ehrgott, 1995] and [Ehrgott, 1997a]).

Theorem 5

- *If x^0 is a lex-MO optimal solution then x^0 is also Pareto optimal solution and it is a max-ordering solution.*
- *x^0 is a lexicographic max-ordering solution if and only if x^0 satisfies the normalization, regularity and reduction property.*

The properties of Theorem 5 need some explanation. Normalization simply means, that when we have only one objective the optimal solution should define the usual minimum. Regularity is defined as the conservativeness or robustness in the sense of max-ordering optimality. Finally, by reduction we mean that, should the value of one objective for an optimal solution be known, then we can drop this criterion from minimization and impose it as a constraint instead, but still obtain the same optimal solutions for this reduced problem.

From the point of view of decision makers, given the task to choose from such a huge number of possibly "good" (Pareto optimal) alternatives, these three properties seem reasonable. So, once they are accepted as valid, there is only one choice for choosing a solution, namely the lexicographic max-ordering optimal solution.

3. Multicriteria Network Flow Problems

In this main section we will introduce the multicriteria network flow problem. After reviewing existing literature on the topic, we will present the outline of a method to solve the continuous problem in Section 3.1. Section 3.2 describes a method to solve the integer problem. The method will be illustrated by means of an example. Finally, we discuss the problem of determining a lexicographic max-ordering solution as a compromise solution.

The multicriteria network flow problem can be written as follows

$$\min \begin{pmatrix} \sum_{(i,j)\in A} c^1_{ij} x_{ij} \\ \vdots \\ \sum_{(i,j)\in A} c^Q_{ij} x_{ij} \end{pmatrix}$$

$$\sum_{j\in N} x_{ij} - \sum_{j\in N} x_{ji} = 0 \qquad \forall i \in N \qquad \text{(MONFP)}$$

$$x_{ij} \geq l_{ij} \qquad \forall (i,j)\in A$$

$$x_{ij} \leq u_{ij} \qquad \forall (i,j)\in A$$

The feasible set is again denoted by X, the feasible set of the integer (MONFP) is denoted by $X_{int} := X \cap \mathbb{Z}^{|A|}$.

Several authors have investigated this problem. Solutions which can be obtained by a weighted sum scalarization have been considered in [Klingman and Mote, 1982]. Burkard and others [Burkard et al., 1989, Ruhe and Fruhwirth, 1990 have used a sandwich approximation algorithm for convex curves to approximate the efficient frontier of continuous bicriteria network flow problems. Lee and Pulat and later Pulat and Huarng developed a method to completely determine the efficient frontier and all Pareto optimal solutions in the continuous bicriteria case [Lee and Pulat, 1991, Pulat et al., 1992]. Another study by Lee and Pulat treated the bicriteria integer case [Lee and Pulat, 1993]. It is this paper upon which our research builds up. We generalize the method to the case of more than two objectives. However, we will not give any technical proofs, which would be very similar to those given in [Lee and Pulat, 1993]. The structure of integer solutions in the bi- and tricriteria case was also studied in [Mustafa and Goh, 1997] and [Mustafa and Goh, 1998]. Several authors considered (MONFP) with lexicographic optimality [Calvete and Mateo, 1995, Calvete and Mateo, 1996] or max-ordering optimality [Hamacher, 1995]. Results of the latter paper will be used in Section 3.3. Finally, the method is illustrated by examples in Section 4.

Figure 2 illustrates why solving the continuous problem is not enough to find all integer Pareto optimal solutions. The point y is an integer efficient point, because there is no integer point on the edge connecting y^2 and y^3.

Due to the results mentioned above (Theorem 1 and the total unimodularity of the incidence matrix) we know that all extreme points in Y are integer. However, there may be integer points between extreme points, if a pivot step between the corresponding adjacent Pareto optimal basic solutions represents a flow change of more than one unit. In Figure 2 the points $y^2_{(1)}$, $y^2_{(2)}$, and $y^3_{(3)}$ between extreme points y^2 and y^3 are such points. These are also efficient points for the continuous problem. However, introducing the integrality requirement results in points $y^2_{(1,0)}, \ldots, y^2_{(1,3)}$ being efficient. These cannot be found solving a problem with a weighted sum of objectives. Additional problems only encountered with more than two objectives will be explained later.

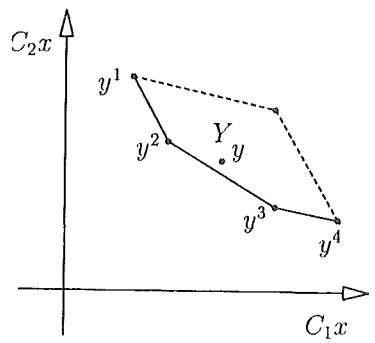


Figure 2 - A bicriteria example for integer efficient points

The general idea to solve the integer (MONFP) contains the following steps.

1. Find an initial efficient extreme point y^1.

Lemma 1

If y^t is an efficient extreme point, $y^{t'}$ is obtained by introducing (u,v) into the basis at y^t and (u,v) is in kilter for all c^q. Then $y^{t'}$ is not efficient.

Proof:

Obvious from the definition of an arc being in kilter. ♦

After having determined the efficient extreme point of Y, we use G_{eff} to determine the whole efficient frontier, which is composed of maximal efficient faces.

Definiton 4

Consider the continuous (MONFP). The convex hull of a subset of extreme points of Y, which is obtained in the boundary of Y is called a face of Y. A face F of Y is called efficient face, if all $y \in F$ are efficient. An efficient face F is maximal if there is no other efficient face F' such that $F \subset F'$.

In order to determine all maximal efficient faces. we check all possible faces (which correspond to cyles in G_{eff}) for efficiency. This makes use of the result:

Lemma 2

A face F is efficient if and only if there exists an inner point $\hat{y}$ of F which is efficient.

Proof:

If a face is efficient, all points of F are efficient. On the other hand suppose $\hat{y}$ is an efficient inner point of F. By Theorem 4 this implies that there exists a $\lambda \in \mathbb{R}^Q$ such that $\hat{y}$ solves the (NFP) with objective $\lambda^T C$. By the linearity of the problem, the whole face F is optimal for this same scalarized problem. ♦

The idea of the efficiency check is illustrated in Figure 3 and formally stated in Lemma 3. An inner point $\hat{y}$ can be obtained by

$$\hat{y} = y^t + 0.5\,(\bar{c}_{(u,v)_1} + \ldots + \bar{c}_{(u,v)_t}).$$

Here y^t is an efficient extreme point of a t-dimensional face and $(u,v)_i$ are the edges introduced into the basis at y^t to move to adjacent extreme point y^i i = 1,..., t.

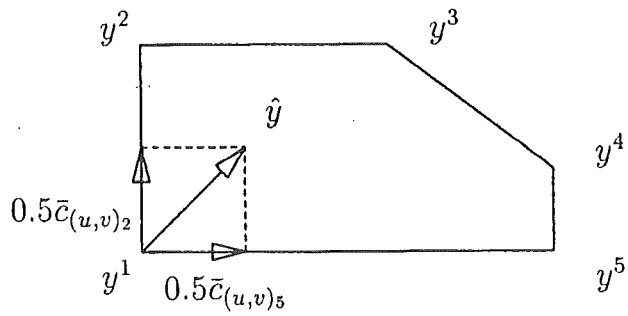


Figura 3 - An inner point of a face

Lemma 3 ([Ecker and Kouada, 1975])

Let $\hat{y}$ be an inner point of a face F of Y and let (x^0,s^0) be an optimal solution of the problem (TP) below. Then $\hat{y}$ is efficient if and only if $\sum_{q=1}^{Q} s_i^0 = 0$.

$$\begin{aligned} \min \quad & \sum_{q=1}^{Q} s_q \\ \text{s.t.} \quad & x \in X \\ & s \geq 0 \\ & Cx - \hat{y} = -Is \end{aligned} \qquad \text{(TP)}$$

3.2 Integer efficient points

In order to find all integer efficient solutions of (MONFP), we proceed in two steps. First, the integer solutions on the efficient frontier are determined. Then, modifying our original (MONFP), we determine solutions in the interior of Y.

During the determination of the efficient extreme points of Y (which are integer efficient points themselves) all those lying on efficient edges can be determined immediately. Note that there may exist integer efficient points on edges leading to non efficient extreme points. This will be determined as points "inside" or "behind" the efficient frontier.

After the determination of the maximal efficient faces, the integer points on the efficient frontier can be computed. Let F' be a t-dimensional maximal efficient face, $2 \leq t \leq Q-1$, and let y^1 be an extreme point of F'. Then select t adjacent efficient extreme points $y^2,\ldots, y^{t+1}$ such that $\{y^1,\ldots, y^{t+1}\}$ are affinely independent. This identifies t nonbasic variables x_{uv_i} at y^1.

Then all integer combinations of flow changes of the variables x_{uv_i} (and the respective cycles) leading from y^1 to the adjacent extreme points $y^2,\ldots, y^{t+1}$ define the set of integer efficient points on F'. The unit flow changes to be considered are between 1 and the flow change to reach y^i, respectively. For an illustration, see Figure 4 with y^1 and adjacent extreme points y^2 and y^5.

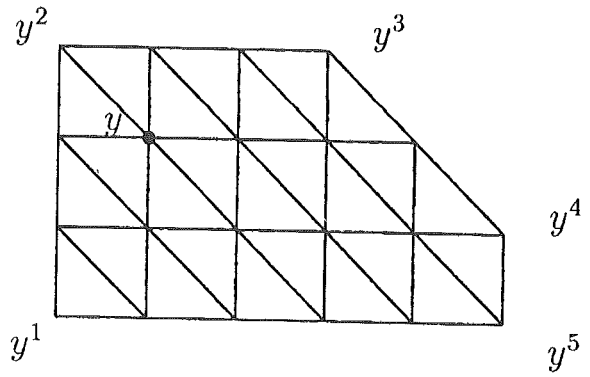


Figure 4 - Integer efficient points on a two-dimensional face

If we call two Pareto optimal integer flows adjacent, if one can be obtained from the other by flow change along a single cycle, we have the following theorem.

Theorem 6

The set of Pareto optimal integer flows corresponding to integer efficient point on the efficient frontier defines a connected graph.

Proof:

According to Theorem 2 the efficient frontier is connected. Theorem 4 says that the graph of Pareto optimal basic solutions is connected. We can therefore restrict ourselves to a maximal efficient face, say F. Let y be an efficient integer point on face F. According to the algorithm, y is obtained by combining flow changes along several cycles. Performing these backwards one after another will generate a sequence of integer points eventually ending in an efficient extreme point $\hat{y}$ of face F (see Figure 4 and the text before). According to the choice of nonbasic variables at $\hat{y}$ none of the flow changes can yield points outside F, thus all intermediate integer points (and corresponding flows) are efficient (resp. Pareto optimal). ♦

The crucial step in determining integer efficient solutions is the computation of those which are behind the continuous efficient frontier. The idea to obtain these is to change the bounds (decrease the upper bound or increase the lower bound) of some nonbasic variables by 1 at some efficient basic solution.

Lemma 4

Let y^t be an extreme point of a face F. Changing the bound of a nonbasic variable one at y^t leads to a parallel translation of a face F containing y^t.

Proof:

The slope of a face is determined by the reduced costs $\bar{c}_{ij}$. Changing the bound of a basic variables by one does not change the structure of the basis, thus $\bar{c}_{ij}$ are not changed and the slope remains the same. That the translation is nonzero follows from the fact that the original solution becomes infeasible by the bound change. ♦

Note that in the degenerate case, we may have $\bar{c}_{ij} = 0$ and therefore the result is not be true in general.

The effect of changing bounds is illustrated in Figure 5.

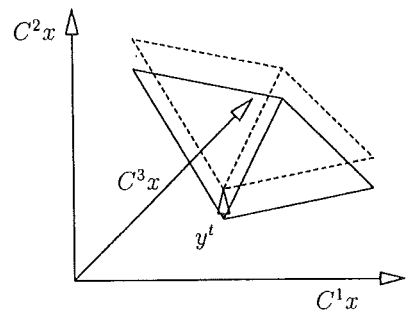


Figure 5 - Parallel translation of faces

and selecting the lexicographic minimum. An efficient way to do that is described in [Ehrgott, 1998].

The essential step in determining p best solutions is to find a second best solution. Then by applying branch and bound methods, further solutions can be determined. Theorem 8 provides the background of an algorithm for the integer lex-MO (MONFP). The details of the ranking method for (NFP) can be found in [Hamacher, 1995] and need not be repeated here.

4. Examples

We illustrate the proposed method by means of two examples with two and three criteria, respectively. Both will be defined for the same network of Figure 6.

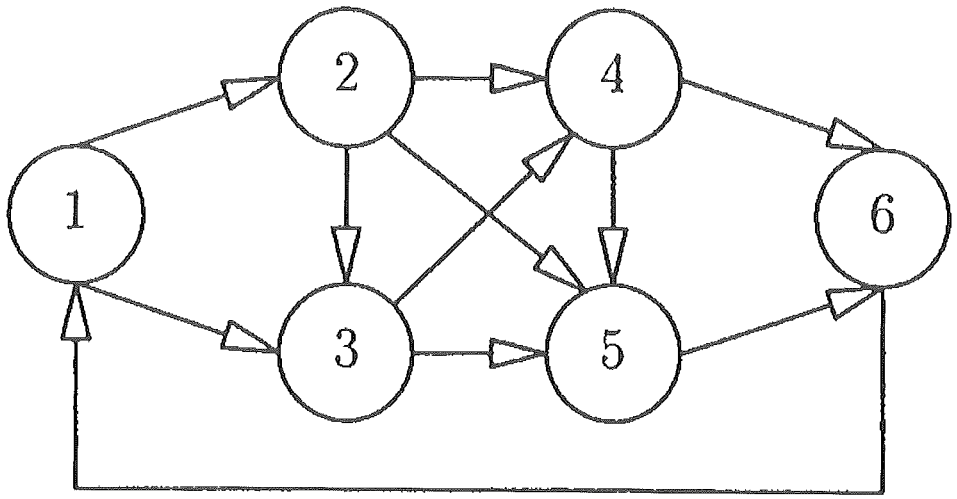


Figure 6 - Network for the examples

The cost coefficients as well as lower and upper bounds are given in the table below.

arc	(1,2)	(1,3)	(2,3)	(2,4)	(2,5)	(3,4)	(3,5)	(4,5)	(4,6)	(5,6)	(6,1)
C^1	5	8	2	6	1	2	6	5	1	2	0
C^2	1	2	7	1	7	4	2	3	9	8	0
l	0	2	0	2	0	0	3	0	0	4	10
u	9	12	8	10	6	10	10	7	10	9	10

Determining an initial extreme point using weights $\lambda_1 = \lambda_2 = \frac{1}{2}$ will yield either y^2 or y^3, depending on the implementation of the algorithm. Exploring adjacent basic feasible solutions, we eventually discover the complete efficient frontier, containing extreme points $y^1,\ldots, y^4$.

In the second step, we observe that only moving from y^2 to y^3 a flow change of more than one unit is performed: nonbasic variables $x_{(2,4)}$ is increased by 4 units. Stepwise increase at this point, together with regions where integer efficient points may be found is illustrated in Figure 7.

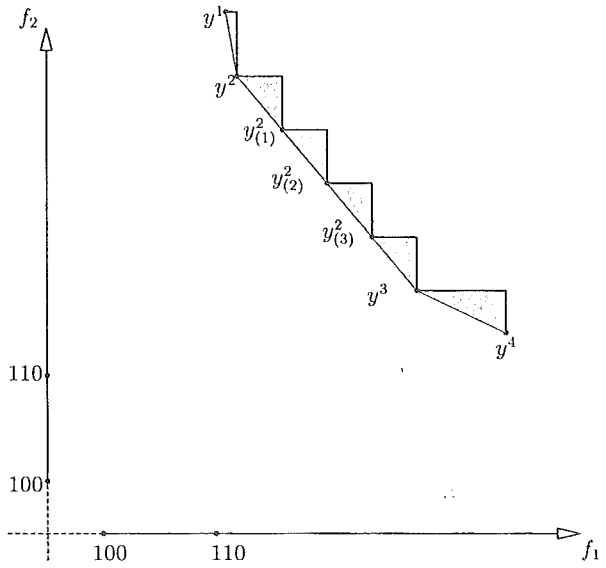


Figure 7 - Integer points on the efficient frontier

To find integer efficient points behind the efficient frontier, we choose y^2. Noting that $x_{(3,4)}$ is not in kilter for both objectives, we increase the lower bound on arc (3,4) from 0 to 1. We have to change the flow in order to obtain a feasible solution. As a result, y^2 is translated to $y^2_{(3,4)}$ [1,0]. Here the 1 denotes flow change along arc (3,4), the 0 flow change along arc (2,4), leading to y^3. When we now perform the same flow changes at the translated point, that we did at y^2 we find $y^2_{(3,4)}$ [1,1] and $y^2_{(3,4)}$ [1,2]. Finally, we note that a further increase of the lower bound does not yield more integer efficient solutions. The same is observed for all other nonbasic variables at the extreme points, thus the problem is solved. Note that conv{$y^2_{(3,4)}$ [1,0],$y^2_{(3,4)}$ [1,2]} is the translation of the efficient face conv{y^1,y^2}.

Calculating a lexicographic max-ordering solution in this problem yields $y^2_{(3)}$ = (124,123) as a compromise solution. Note that this is not a basic feasible solution. The result is shown in Figure 8.

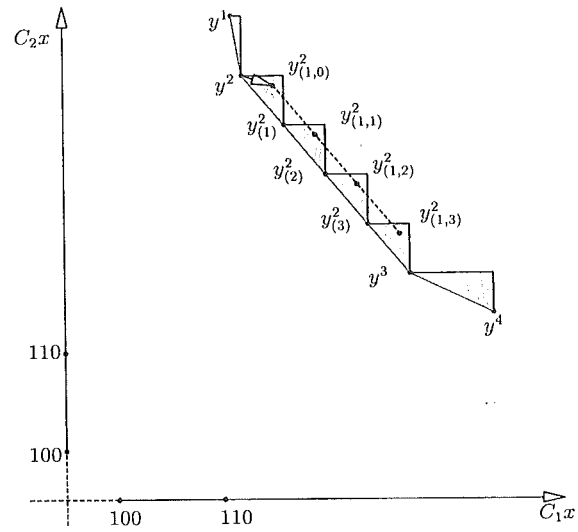


Figure 8 - Integer points on the efficient frontier

For a tricriteria example, we simply add another objective function to the bicriteria example: $C^3 = (1,3,5,9,7,6,2,3,1,2,0)$. We restrict ourselves to the presentation of the results.

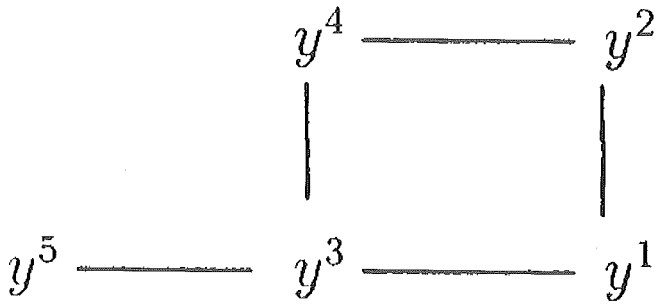


Figure 9 - Graph of efficient extreme points

The graph Geff is shown in Figure 9. The objective space with all 43 integer efficient points (21 on the efficient frontier and 22 others) is shown in Figure 10. The efficient frontier itself consists of a one and a two dimensional face. Integer points on the efficient frontier are black dots, those in the interior of Y are empty dots, the lex-MO solution is indicated in Figure 10 by a circle around the point.

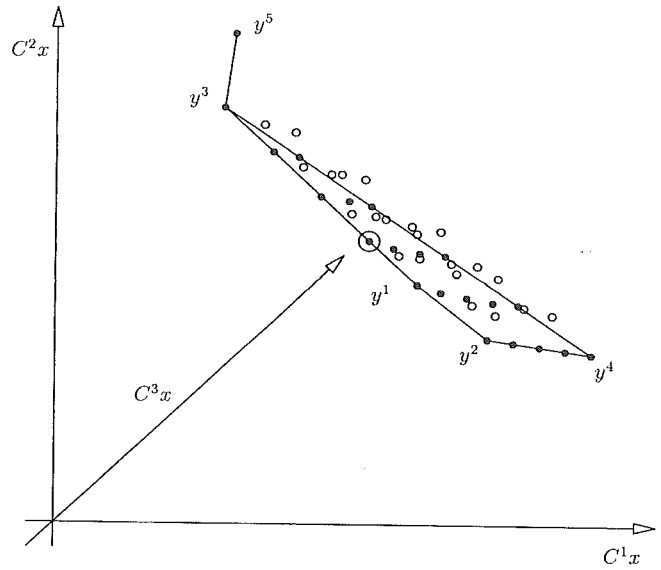


Figure 10 - The (integer) efficient set for the tricriteria example

Acknowledgment

The author wants to express his gratitude to Christina Soltau, who carried out the calculations for the examples.

References

[1] Bazaraa, M. and Jarvis, J., *Linear Programming and Network Flows*, Wiley, New York (1977).

[2] Benson, H., *An outer approximation algorithm for generating all efficient points in the outcome set of a multiple objective linear programming problem*, Technical report, University of Florida (1997).

[3] Bukard, R., Rote, G., Ruhe, G. and Sieber, N., *Algorithmische Untersuchungen zu bikriteriellen kostenminimalen Flüssen in Netzwerken*, Wissenschaftliche Zeitung der technischen Hochschule Leipzig 13 (1989) 333-341.

[4] Calvete, H. and Mateo, M., *An approach for the network flow problem with multiple objectives*, Computers and Operations Research 22 (1995) 971-983.
[5] Calvete, H. and Mateo, M., *A sequential network-based approach for the multiobjective network flow problem with preemptive priorities*, in Tamiz, M. editor, Multi-Objective Programming and Goal Programming - Theory and Applications (1996) 74-86, Springer-Verlag, Berlin.
[6] Ecker, J. and Kouada, I., *Finding efficient points for linear multiple objective programs*, Mathematical Programming 8 (1975) 375-377.
[7] Ehrgott, M., *Lexicographic max-ordering - a solution concept for multicriteria combinatorial optimization*, In Schweigert, D. editor, Methods of Multicriteria Decision Theory, Proceedings of the 5th Workshop of the DGOR-Working Group Multicriteria Optimization and Decision Theory (1995) 55-66.
[8] Ehrgott, M., *A characterization of lexicographic max-ordering solutions*, In Göpfert, A., Seeländer, J. and Tammer, C., editors, Methods of Multicriteria Decision Theory, Proceedings of the 6th Workshop of the DGOR-Working Group Multicriteria Optinization and Decision Theory Alexisbad 1996, vol. 2389 (1997) 193-202, Deutsche Hochschulschriften, Hänsel-Hohenhausen, Egelsbach.
[9] Ehrgott, M., *Multiple Criteria Optimization - Classification and Methodology*, Shaker Verlag, Aachen (1997).
[10] Ehrgott, M., *Discrete decision problems, multiple criteria optimization classes and lexicographic max-ordering*, In Stewart, T. and van den Honert, E. editors, Trends in Multicriteria Decision Making, Lecture Notes in Economics and Mathematical Systems 465 (1998) 31-44, Springer-Verlag, Berlin.
[11] Geoffrion, A., *Proper efficiency and the theory of vector maximization*, Journal of Mathematical Analysis and Applications 22 (1968) 618-630.
[12] Hamacher, H., *K best network flows*, Annals of Operations Research 57 (1995) 65-72.
[13] Isermann, H., *Proper efficiency and the linear vector maximum problem*, Operations Research 22 (1974) 189-191.
[14] Isermann, H., *The enumeration of the set of all efficient solutions for a linear multiple objective program*, Operational Research Quarterly 28 (1977) 711-725.
[15] Klingman, D. and Mote, J., *Solution approaches for network flow problems with multiple criteria*, Advances in Management Science 1 (1982) 1-30.
[16] Lee, H. and Pulat, P., *Bicriteria network flow problems: continuous case*, European Journal of Operational Research 51 (1991) 119-126.
[17] Lee, H. and Pulat, P., *Bicriteria network flow problems: integer case*, European Journal of Operational Research 66 (1993) 148-157.
[18] Mustafa, A. and Goh, M., *Characteristics of the efficient solutions of bicriteria and tricriteria network flow problems*, In Caballero, R., Ruiz, F. and Steuer, R. editors, Advances in Multiple Objective and Goal Programming, Lecture Notes in Economics and Mathematical Systems 455 (1997) 131-139, Springer-Verlag, Berlin.
[19] Mustafa, A. and Goh, M., *Finding integer efficient solutions for bicriteria and tricriteria network flow problems*, Computers and Operations Research 25 (1998) 139-157.
[20] Naccache, P., *Connectedness of the set of nondominated outcomes in multicriteria optimization*, Journal of Optimization Theory and Applications 25 (1978) 459-467.
[21] Pulat, P., Huarng, F. and Lee, H., *Efficient solutions for the bicriteria network flow problem*, Computers and Operations Research 19 (1992) 649-655.
[22] Ruhe, G. and Fruhwirth, B., *ε-optimality for bicriteria programs and its application to minimum cost flows*, Computing 44 (1990) 21-34.
[23] Warburton, A., *Quasiconcave vector maximization: connectedness of the sets of Pareto-optimal and weak Pareto-optimal alternatives*, Journal of Optimization Theory and Applications 40 (1983) 537-557.

DERIVATE-FREE METHODS FOR UNCONSTRAINED NONSMOOTH OPTIMIZATION AND ITS NUMERICAL ANALYSIS

Adil M. Bagirov

Laboratory of High Energies
Joint Institute for Nuclear Research
141980 Dubna Moscow Region
Russia

Abstract

Numerical algorithms for solving the unconstrained problems of Lipschitz optimization are presented. These algorithms are based on continuous approximations to the subdifferential and belong to the numerical algorithms of nonsmooth optimization without calculating derivatives. We give a definition of such approximations and propose a method for its construction. An algorithm for the computation of descent direction of the objective function is described. We consider various procedures for the computation of stepsize. The results of numerical experiments are presented. We compare the suggested algorithms using these results.

Keywords

Lipschitz optimization, Clarke subdifferential, continuous approximation, semismooth function.

1. Introduction

In this paper, we consider the problem

$$f(u) \to \inf, \ u \in E_n, \tag{1}$$

where E_n is the n-dimensional Euclidean space of vectors $u = (u_1,\ldots, u_n)$ and f is a locally Lipschitz function, defined on E_n. We assume that the Clarke subdifferential $\partial f(u)$ cannot be computed for any u.

Minimization methods without calculating derivates were developed well enough only for smooth functions. There are a lot of papers on this subject (see, for example, [6,9,10,12,18,19,20,21]). In these papers, various approaches to the construction of methods without calculating derivatives were suggested. One of these approaches is related to the use of finite - difference approximations of the gradient in numerical methods (see, [12]).

The methods of nonsmooth optimization without calculating derivates were studied sufficiently weakly. Such methods may be useful when the subdifferential of the objective function is unknown or very complex in form. For example, the Clarke subdifferential of marginal functions is very complex in form. For its complete computation at a given point, it is

necessary to find a set of solutions of special problems of mathematical programming. But for the computation of the values of the objective function it is sufficient to find the value of these problems. Of course, the latter problem is simpler.

On the other hand the subdifferential mapping enjoys only upper semicontinuity. As it was pointed out in [23], the lack of continuity of the subdifferential mapping was responsible for the failure of nonsmooth steepest descent algorithms. Consequently, various continuous approximations were proposed, among them the ε-subdifferential has been found to be most successful. Most efficient nonsmooth optimization methods, such as the bundle method and its variations, are based on it (see, [11,13,14,17,22,25,26]).

In this paper, we suggest a method without calculating derivatives for minimizing locally Lipschitz functions. This method is based on continuous approximations to the subdifferential. The continuous approximations under consideration were studied in [4]. To construct the continuous approximations, a notion of discrete gradient, introduced and studied in [1,2], is used. The discrete gradient is one of possible versions of the finite - difference approximation of a subgradient. Only values of the objective function are used for its computation.

The algorithm used to compute the descent direction of the objective function is described. We discuss the problem of choice of stepsize. The use of continuous approximations allows us to use some ideas of smooth optimization. In particular, we use the idea on modelling the objective function along the descent direction by one - dimensional quadratic and piecewise linear functions. Such algorithms with quadratic and cubic functions for the unconstrained smooth optimization were considered in [12]. Some numerical experiments were carried out using the suggested algorithms. We present the results of these experiments and compare the suggested algorithms.

This paper is organized as follows. In Section 2 we give a definition of a continuous approximation to the subdifferential and describe a method for its construction. Section 3 is devoted to the description of an algorithm for the computation of descent direction of the objective function. In Section 4 a method for solving unconstrained problem of Lipschitz optimization is described and its convergence is proved. In Section 5 we discuss the problem of the computation of stepsize. Results of numerical experiments are presented in Section 6. Section 7 provides some conclusions.

We use the following notation: $\langle u,v\rangle = \sum_{i=1}^{n} u_i v_i$ is a scalar product in E_n, $\|u\| = \langle u,u\rangle^{1/2}$ is the Euclidean norm. We shall denote by $f'(u,g)$ the directional derivate of the function f at a point u in a direction $g \in E_n$, by conv A the convex hull of the set $A \subset E_n$ and by cl A the closure of the set A.

Then open and closed δ-ball centered at a point $u \in E_n$ will be denoted by $S_\delta(u)$ and $S^0_\delta(u)$, respectively. More specifically, if $u = 0$, we use S_δ and S^0_δ to denote the open and closed δ-ball centered at the origin, respectively:

$$S_\delta(u) = \{v \in E_n \mid \|v-u\| < \delta\},\ S^0_\delta(u) = \{v \in E_n \mid \|v-u\| \le \delta\},$$

$$S_\delta = \{v \in E_n \mid \|v\| < \delta\},\ S^0_\delta = \{v \in E_n \mid \|v\| \le \delta\}.$$

We denote the Hausdorff distance between the subsets $A, B \subset E_n$ by

$$d_H(A,B) = \max\{\sup_{u \in A} \inf_{v \in B} \|u-v\|, \sup_{v \in B} \inf_{u \in A} \|u-v\|\}.$$

2. Preliminary Results

In this section we give the definition of a continuous approximation to the Clarke subdifferential and describe a method for its construction.

Let f be a scalar locally Lipschitz function defined on E_n. We recall that a locally Lipschitz function f is differentiable almost everywhere and that one can define for it the Clarke subdifferential (see, [7,8]), by

$$\partial f(u) = \mathrm{conv}\{v \in E_n \mid \exists (u^k \to u,\ k \to +\infty) : v = \lim_{k \to +\infty} \nabla f(u^k)\},$$

where $u^k \to u$ are such that $\nabla f(u^k)$ exists. It is shown that the mapping $\partial f(u)$ is upper semicontinuous and bounded on a bounded set.

Let U be a compact subset of the space E_n. We consider a family $V(u,\varepsilon) = V_\varepsilon(u)$ of set-valued mappings depending on a parameter $\varepsilon > 0$. For each $\varepsilon > 0$, $V(\cdot,\varepsilon)$ is a set-valued mapping of U to subsets of E_n. We suppose that $V(u,\varepsilon)$ is a compact convex subset for all $u \in U$ and $\varepsilon > 0$. It is assumed that there exists a number $R > 0$ such that:

$$\sup\{\|v\| : v \in V(u,\varepsilon),\ u \in U,\ \varepsilon > 0\} \le R. \qquad (2)$$

Definition 1

The limit of the family $\{V(u,\varepsilon)\}$ at a point u is defined by the following set:

$$V_L(u) = \{v \in E_n \mid \exists (u^k \to u,\ \varepsilon_k \to +0,\ v^k \in V(u^k,\varepsilon_k)) : v = \lim_{k \to +\infty} v^k\}.$$

It is possible that the limit $V_L(u)$ is not convex even if all the sets $V(u,\varepsilon)$ are convex. So we shall consider $\mathrm{conv}V_L(u)$ the convex hull of the limit $V_L(u)$. It follows from the definition and the inequality (2) that the mapping $\mathrm{conv}V_L$ has compact convex images.

Definition 2

A family $\{V(u,\varepsilon)\}$ is called a continuous approximation to the subdifferential $\partial f(u)$ on U, if the following holds:

1) $V(u,\varepsilon)$ is a Hausdorff continuous mapping with respect to u on U for all $\varepsilon > 0$;
2) The subdifferential $\partial f(u)$ is the convex hull of the limit of the family $V(u,\varepsilon)$ on U, i.e. for all $u \in U$

$$\partial f(u) = \mathrm{conv}V_L(u).$$

Now we will describe a method for the construction of continuous approximations. This method uses a concept of a discrete gradient. First, we recall the definition of a discrete gradient (see, [2]).

Let

$$S_1 = \{g \in E_n \mid \|g\| = 1\},$$

$$G = \{e \in E_n \mid e = (e_1, \ldots, e_n),\ |e_j| = 1,\ j = 1, \ldots, n\},$$

$$P = \{z(\lambda) \mid z(\lambda) \in E_1,\ z(\lambda) > 0,\ \lambda > 0 \text{ and } \lambda^{-1} z(\lambda) \to 0,\ \lambda \to 0\},$$

$$I(g,\alpha) = \{i \in \{1, \ldots, n\} \mid |g_i| \geq \alpha\},$$

where $\alpha \in (0, n^{-1/2}]$ is a fixed number. It is clear that $I(g,\alpha)$ is non-empty for any $g \in S_1$.

We define the operators $H_i^j : E_n \to E_n$ for $i,j = 1, \ldots, n$ by the formula

$$H_i^j g = \begin{cases} (g_1, \ldots, g_j, 0, \ldots, 0), & \text{if } j < i, \\ (g_1, \ldots, g_{i-1}, 0, g_{i+1}, \ldots, g_j, 0, \ldots, 0), & \text{if } j \geq i. \end{cases}$$

Let $e(\beta) = (\beta e_1, \beta^2 e_2, \ldots, \beta^n e_n)$, where $\beta \in (0,1]$. For $u \in E_n$ we consider the vectors

$$u_i^j(g,e,z,\lambda,\beta) = u + \lambda g - z(\lambda) H_i^j e(\beta),$$

where $g \in S_1$, $e \in G$, $i \in I(g,\alpha)$, $z \in P$, $\lambda > 0$, $\beta \in (0,1]$, $j = 0, 1, \ldots, n$, $j \neq i$.

It is clear that $H_i^0 g = 0 \in E_n$ for all $i = 1, \ldots, n$. Therefore,

$$u_i^0(g,e,z,\lambda,\beta) = u + \lambda g.$$

From the definition of the operators H_i^j it follows that $H_i^i g = H_i^{i-1} g$.

Consequently,

$$u_i^0(g,e,z,\lambda,\beta) = u_i^{i-1}(g,e,z,\lambda,\beta).$$

In particular, if $i = n$ then $u_n^n(g,e,z,\lambda,\beta) = u_n^{n-1}(g,e,z,\lambda,\beta)$.

Henceforth, if $\lambda_0 > 0$ is given, the notations $z_1 \leq z_2$ and $z_k \to 0$, $z_k \in P$, $k \to +\infty$, will mean that $z_1(\lambda) \leq z_2(\lambda)$ for any $\lambda \in (0,\lambda_0]$ and $\sup\{z_k(\lambda), \lambda \in (0,\lambda_0]\} \to 0$ as $k \to +\infty$.

Definition 3

The discrete gradient of the function f at the point $u \in E_n$ is the vector $\Gamma^i(u,g,e,z,\lambda,\beta) = (\Gamma_1^i, \ldots, \Gamma_n^i) \in E_n$, $g \in S_1$, $i \in I(g,\alpha)$, with the following coordinates:

$$\Gamma_j^i = [z(\lambda) e_j(\beta)]^{-1} [f(u_i^{j-1}(g,e,z,\lambda,\beta)) - f(u_i^j(g,e,z,\lambda,\beta))],$$

$$j = 1, \ldots, n,\ j \neq i,$$

$$\Gamma_i^i = (\lambda g_i)^{-1} \Big[f(u_i^n(g,e,z,\lambda,\beta)) - f(u) - \sum_{j=1, j\neq i}^{n} \Gamma_j^i (\lambda g_j - z(\lambda) e_j(\beta))\Big].$$

Concrete examples and some explanations for the discrete gradient are given in [1,4,5]. We note the following useful lemma (see, [1]).

Lemma 1

For any $g \in S_1$, $e \in G$, $i \in I(g,\alpha)$, $z \in P$, $\lambda > 0$, $\beta > 0$

$$f(u+\lambda g) - f(u) = \lambda \langle \Gamma^i(u,g,e,z,\lambda,\beta), g \rangle. \tag{3}$$

For fixed $z \in P$, $\lambda > 0$, $\beta > 0$ we will consider the following mapping:

$$D_0(u,z,\lambda,\beta) = \text{cl conv}\{v \in E_n \mid \exists (g \in S_1, e \in G, i \in I(g,\alpha)) : v = \Gamma^i(u,g,e,z,\lambda,\beta)\}.$$

Let

$$B(u) = \bigcap_{z_0 \in P, \lambda_0 > 0, \beta_0 > 0} \operatorname{clconv} \bigcup \{D_0(u,z,\lambda,\beta) : z \in P,\ z \le z_0,\ \lambda \le \lambda_0,\ \beta \le \beta_0\}.$$

Lemma 2

Let f be a Lipschitz continuous function on the set $S_\delta(u)$, $u \in E_n$, $\delta > 0$. Then there exist $z_0 \in P$, $\lambda_0 > 0$ and $R > 0$, such that

$$\sup\{\|v\| : v \in D_0(u,z,\lambda,\beta),\ z \in P,\ z \le z_0,\ \lambda \in (0,\lambda_0],\ \beta \in (0,1]\} \le R.$$

Proof:

For $\delta > 0$ there exist $z \in P$, $\lambda_0 > 0$, such that

$$u_i^j(g,e,z,\lambda,\beta) \in S_\delta(u)$$

for all $z \le z_0$, $\lambda \in (0,\lambda_0]$, $\beta \in (0,1]$, $g \in S_1$, $e \in G$, $i \in I(g,\alpha)$ $j = 0, 1, \ldots, n$, $j \ne i$. Let L be a Lipschitz constant of the function f on the set $S_\delta(u)$. Then from the definition of the discrete gradient it immediately follows that $|\Gamma_j^i| \le L$ for all $j = 1, \ldots, n$, $j \ne i$.

For $j = i$ from (3), taking into account the inequality $|g_i| \ge \alpha$, we get

$$|\Gamma_j^i| \le (\lambda|g_i|)^{-1}[|f(u+\lambda g) - f(u)| + \lambda \sum_{j=1, j \ne i}^{n} |\Gamma_j^i g_j|] \le nL/\alpha.$$

Thus,

$$\|\Gamma^i(u,g,e,z,\lambda,\beta)\| \le (L/\alpha)(\alpha^2 n + n^2 - \alpha^2)^{1/2}$$

for all $g \in S_1$, $e \in G$, $i \in I(g,\alpha)$, $z \in P$, $z \le z_0$, $\lambda \in (0,\lambda_0]$, $\beta \in (0,1]$. Set $R = (L/\alpha)(\alpha^2 n + n^2 - \alpha^2)^{1/2}$. Then we have that $\|v\| \le R$ for all $v \in D_0(u,z,\lambda,\beta)$. ♦

Note that the (locally Lipschitz) continuity of the function f implies the Hausdorff (locally Lipschitz) continuity of the mapping $D_0(u,z,\lambda,\beta)$ with respect to u for fixed $z \in P$, $\lambda > 0$, $\beta \in (0,1]$ [2]. If the function f is continuous, then the mapping $D_0(u,z,\lambda,\beta)$ is Hausdorff continuous with respect to (u,z,λ,β), $u \in E_n$, $z \in P$, $\lambda > 0$, $\beta \in (0,1]$.

We will consider semismooth functions [16]. The following results have been established in [3]:

Theorem 1

Let f be a semismooth function. Assume that the directional derivative $f'(u,g)$ is upper semicontinuous with respect to u for all $g \in E_n$. Then, $\partial f(u) = B(u)$.

Corollary 1

Let f be a semismooth function with the directional derivative $f'(u,g)$ lower semicontinuous with respect to u for all $g \in E_n$. Then, $\partial f(u) = B(u)$.

Corollary 2

Let all the conditions of Theorem 1 or Corollary 1 be satisfied. Then, for any $\varepsilon > 0$ there exist $z_0 \in P$, $\lambda_0 > 0$, $\beta_0 > 0$ such that for all $z \in P$, $z < z_0$, $\lambda \in (0,\lambda_0)$, $\beta \in (0,\beta_0)$

$$d_H(\partial f(u), D_0(u,z,\lambda,\beta)) < \varepsilon.$$

Now we apply the notion of the discrete gradient for the construction of a continuous approximation to the subdifferential.

Corollary 2 implies that for all $\varepsilon > 0$ there exist $z \in P$, $\lambda > 0$, $\beta > 0$ such that

$$D_0(u,z,\lambda,\beta) \subset \partial f(u+S_\varepsilon^0) + S_\varepsilon, \tag{4}$$

where

$$\partial f(u+S_\varepsilon^0) = \bigcup \{\partial f(y) : y \in S_\varepsilon^0(u)\}.$$

Let

$$V(u,\varepsilon) = D_0(u,z,\lambda,\beta), \tag{5}$$

where $z \in P$, $\lambda > 0$, $\beta > 0$ satisfy the condition (4). The mapping $V(u,\varepsilon)$ is continuous with respect to the Hausdorff metric for (u,ε), $\varepsilon > 0$. Indeed it follows from Corollary 2 that for $\varepsilon > 0$ we can find $z \in P$, $\lambda > 0$, $\beta > 0$ such that

$$D_0(u,z,\lambda,\beta) \subset \partial f(u) + S_{\varepsilon/2}. \tag{6}$$

For such $z \in P$, $\lambda > 0$, $\beta > 0$ there exists $\delta > 0$ such that for all $y \in S_\delta(u)$

$$D_0(y,z,\lambda,\beta) \subset D_0(u,z,\lambda,\beta) + S_{\varepsilon/2} \tag{7}$$

Let $\delta < \varepsilon$. We have

$$\partial f(u) \subset \partial f(y + S_\varepsilon^0), \ \forall y \in S_\delta(u).$$

Consequently, from (6) and (7) we get that

$$D_0(y,z,\lambda,\beta) \subset D_0(u,z,\lambda,\beta) + S_{\varepsilon/2} \subset \partial f(u) + S_\varepsilon \subset \partial f(y + S_\varepsilon^0) + S_\varepsilon. \tag{8}$$

Thus $V(y,\varepsilon) = D_0(y,z,\lambda,\beta)$. These inclusions and continuity of the mapping $D_0(u,z,\lambda,\beta)$ with respect to (u,z,λ,β) imply that the mapping $V(u,\varepsilon)$ is continuous at the point (u,ε), $\varepsilon > 0$.

Appying (8), Corollary 2 and upper semi-continuity of the subdifferential $\partial f(u)$ we can conclude that

$$\mathrm{conv} V_L(u) = \partial f(u).$$

Consequently, the family $\{V(u,\varepsilon)\}$ is a continuous approximation to the subdifferential $\partial f(u)$. Thus, we have proved the following assertion.

Theorem 2

Let $U \subset E_n$ be a compact subset and f be a semismooth function with the directional derivative $f'(u,g)$ upper (lower) semicontinuous with respect to u for all $g \in E_n$. Then the family $\{V(u,\varepsilon)\}$ constructed by (4),(5) is a continuous approximation to the subdifferential $\partial f(u)$ on U.

We will denote by Φ a class of locally Lipschitz functions for which the relation $\partial f(u) = B(u)$ holds. Theorem 1 and Corollary 1 show that the class Φ is fairly wide.

From the definition of the mapping $V(u,\varepsilon)$ it follows that for all $f \in \Phi$ we can set

$$V(u,0) = \partial f(u).$$

By the construction of the mapping $V(u,\varepsilon)$ we immediately get the following lemma.

Lemma 3

The mapping $V(u,\varepsilon)$ constructed by (4),(5) is upper semicontinuous with respect to (u,ε) at the point $(u,0)$.

By the construction of the mapping $V(u,\varepsilon)$ we also obtain that at given point $u \in E_n$ for any $\varepsilon > 0$ there exist $\delta > 0$, $z \in P$, $\lambda > 0$, $\beta > 0$ such that for all $y \in S_\delta(u)$

$$V(y,\varepsilon) = D_0(y,z,\lambda,\beta)$$

From this we get the following lemma.

Lemma 4

Let $U \subset E_n$ is a bounded set, $\varepsilon > 0$ and

$$V(u,\varepsilon) = D_0(u,z_u,\lambda_u,\beta_u).$$

Then there exist $z_0 \in P$, $\lambda_0 > 0$, $\beta_0 > 0$ such that

$$z_u \geq z_0,\ \lambda_u \geq \lambda_0,\ \beta_u \geq \beta_0$$

for all $u \in clU$.

3. Computation of the Descent Direction

In this section we discuss the problem of the computation of a descent direction. First, we recall the following lemma (see, [5]).

Lemma 5

Let $U \in E_n$ and for given $z \in P$, $\lambda > 0$, $\beta > 0$

$$\min\{\|v\| : v \in D_0(y,z,\lambda,\beta)\} = \|v^0\| > 0.$$

Then for $g^0 = -\|v^0\|^{-1} v^0$

$$f(u+\lambda g^0) - f(u) \leq -\lambda \|v^0\|.$$

As it follows from Lemma 5, for the computation of the descent direction, we have to solve the following problem

$$\|v\| \to \min,\ v \in D_0(u,z,\lambda,\beta). \tag{9}$$

Problem (9) is difficult, because in general case the complete computation of the set $D_0(u,z,\lambda,\beta)$ is not possible. We can substitute it by the following problem:

$$\|v\| \to \min,\ v \in \overline{D}, \tag{10}$$

where $\overline{D}$ is the convex hull of a finite number of points and

$$\overline{D} \subset D_0(u,z,\lambda,\beta).$$

Effective methods for solving problem (10) (see, for example, [24]) are available.

For the computation of the descent direction of the function f the following Algorithm is suggested. Let be given $z \in P$, $\lambda > 0$, $\beta > 0$, the number $c \in (0,1)$ and small enough number $\delta > 0$.

Step 1 - Choose any $g^1 \in S_1$, $e \in G$, $i \in I(g^1,\alpha)$ and compute the discrete gradient $v^1 = \Gamma^i(u,g^1,e,z,\lambda,\beta)$. Set $\bar{D}_1(u) = \{v^1\}$ and $k = 1$.

Step 2 - Compute the vector $\|w^k\| = \min\{\|w\| : w \in \bar{D}_k(u)\}$. If

$$\|w^k\| \le \delta, \tag{11}$$

then, stop.

Step 3 - Compute the search direction by $g^{k+1} = -\|w^k\|^{-1} w^k$.

Step 4 - If

$$f(u+\lambda g^{k+1}) - f(u) \le -c\lambda\|w^k\|, \tag{12}$$

then, stop.

Step 5 - Compute the discrete gradient $v^{k+1} = \Gamma^i(u,g^{k+1},e,z,\lambda,\beta)$, $i \in I(g^{k+1},\alpha)$, construct the set $\bar{D}_{k+1}(u) = \mathrm{conv}\{\bar{D}_k(u) \bigcup \{v^{k+1}\}\}$, set $k = k+1$ and go to Step 2.

First, we will show that if both conditions for stopping the algorithm do not hold, then the new discrete gradient $v^{k+1} \notin \bar{D}_k(u)$, that is the algorithm in this case allows to improve the approximation of the set $D_0(u,z,\lambda,\beta)$. Indeed, in this case $\|w^k\| > \delta$ and

$$f(u+\lambda g^{k+1}) - f(u) > -c\lambda\|w^k\|.$$

Therefore, from (3) we get that

$$f(u+\lambda g^{k+1}) - f(u) = \lambda\langle\Gamma^i(u,g^{k+1},e,z,\lambda,\beta), g^{k+1}\rangle = \lambda\langle v^{k+1},g^{k+1}\rangle > -c\lambda\|w^k\|.$$

From this we have

$$\langle v^{k+1},w^k\rangle < c\|w^k\|^2. \tag{13}$$

On the other hand, since $w^k = \mathrm{argmin}\{\|v\| : v \in \bar{D}_k(u)\}$, so from necessary condition for a minimum it follows that for any $w \in \bar{D}_k(u)$

$$\langle w^k,w-w^k\rangle \ge 0$$

$$\langle w^k,w\rangle \ge \|w^k\|^2.$$

Then from this and (13) we obtain that $v^{k+1} \notin \bar{D}_k(u)$.

Now we will show that the described algorithm is finite. For this it is sufficient to get an upper estimation for the number of computed discrete gradients, m, after which the following inequality is fulfilled:

$$\|w^m\| \le \delta. \tag{14}$$

It is clear that for arbitrary $t \in [0,1]$

$$\|w^{k+1}\|^2 \le \|tv^{k+1}+(1-t)w^k\|^2$$

or

$$\|w^{k+1}\|^2 \le \|w^k\|^2 + 2t\langle w^k,v^{k+1}-w^k\rangle + t^2\|v^{k+1}-w^k\|.$$

Lemma 2 implies that there exists $R > 0$ such that

$$\|v^{k+1}-w^k\| \le 2R.$$

Then, taking into account the inequality (13), we have

$$\|w^{k+1}\|^2 \le \|w^k\|^2 - 2t(1-c)\|w^k\|^2 + 4t^2R^2.$$

For $t = (1-c)(2R)^{-2}\|w^k\|^2 \in (0,1)$ we get

$$\|w^{k+1}\|^2 \le \{1-[(1-c)(2R)^{-1}\|w^k\|]^2\}\|w^k\|^2. \quad (15)$$

For the given $\delta \in (0,R)$ we will estimate the number of computed discrete gradients, m, after which the inequality (14) is fulfilled.

From (15) and the condition $\|w^k\| > \delta$, $k = 1,\ldots, m-1$ it follows the inequality

$$\|w^{k+1}\|^2 \le \{1-[(1-c)(2R)^{-1}\delta]^2\}\|w^k\|^2.$$

Set $\Gamma = 1-[(1-c)(2R)^{-1}\delta]^2$. It is clear that $\Gamma \in (0,1)$. Consequently,

$$\|w^m\|^2 \le \Gamma\|w^{m-1}\|^2 \le \ldots \le \Gamma^{m-1}\|w^1\|^2 \le \Gamma^{m-1}R^2.$$

If $\Gamma^{m-1}R^2 \le \delta^2$, then the inequality (14) is also fulfilled and therefore,

$$m \le 2\log_2(\delta/R)/\log_2\Gamma + 2.$$

Thus, we have proved the following assertion.

Theorem 3

Let f be a locally Lipschitz function, at a point $u \in E_n$, there exists $R > 0$ such that

$$\max\{\|v\| : v \in D_0(u,z,\lambda,\beta)\} \le R,$$

and $c \in (0,1)$, $\delta \in (0,R)$ be given numbers. Then after m computations of discrete gradients one of the conditions (11), (12) should be fulfilled, where

$$m \le 2\log_2(\delta/R)/\log_2\Gamma + 2,$$

$$\Gamma = 1-[(1-c)(2R)^{-1}\delta]^2.$$

4. The Method and its Convergence

In this section we suggest the method for solving problem (1) by using the mapping $V(u,\varepsilon)$ and study its convergence.

Let be given the numbers $c_1 \in (0,1)$, $c_2 \in (0,c_1]$ and the sequences $\{\delta_k\}$, $\{\varepsilon_k\}$, $\delta_k > 0$, $\varepsilon_k > 0$, $\delta_k \to 0$, $\varepsilon_k \to 0$, $k \to +\infty$. We assume that the choice of $\varepsilon_k > 0$ corresponds to the choice of some $z_k \in P$, $\lambda_k > 0$ and $\beta_k \in (0,1]$.

The method for solving problem (1) can be described as follows.

Step 1 - Choose any starting point $u^k \in E_n$ and set $k = 0$.

Step 2 - Set $s = 0$ and $u_s^k = u^k$.

Step 3 - Apply the algorithm for the computation of the descent direction at $u = u_s^k$, $\delta = \delta_k$, $z = z_k$, $\lambda = \lambda_k$, $\beta = \beta_k$, $c = c_1$. After stopping this algorithm, for some finite $m > 0$, an element $\|v_s^k\| = \min\{\|v\| : v \in \overline{D}_m(u_s^k)\}$ and a search direction $g_s^k = -\|v_s^k\|^{-1}v_s^k$ are computed such that either

$$f(u_s^k + \lambda_k g_s^k) - f(u_s^k) \le -c_1\lambda_k\|v_s^k\|, \quad (16)$$

or $\|v_s^k\| \le \delta_k$.

Step 4 - If

$$\|v_s^k\| \le \delta_k, \quad (17)$$

then set $u^{k+1} = u_s^k$, $k = k+1$ and go to Step 2.

Step 5 - Construct the following iteration

$$u_{s+1}^k = u_s^k + \sigma_s g_s^k.$$

Compute σ_s in

$$\sigma_s = \text{argmax}\{\sigma \geq 0 \mid f(u_s^k+\sigma g_s^k) - f(u_s^k) \leq -c_2\sigma\|v_s^k\|\}. \quad (18)$$

Step 6 - Set s = s+1 and go to Step 3.

Theorem 4

Assume that $f\in\Phi$ and the set $M(u^0) = \{u\in E_n / f(u) \leq f(u^0)\}$ is bounded for starting points $u^0\in E_n$. Then every accumulation point of $\{u^k\}$ belongs to the set $U^0 = \{u\in E_n / 0\in\partial f(u)\}$.

Proof:

Since the function f is locally Lipschitz continuous and the set $M(u^0)$ is bounded so $f_* = inf\{f(u) : u\in E_n\} > -\infty$. If the condition (16) takes place then taking into account that $0 < c_2 \leq c_1$ we have

$$f(u_s^k+\lambda_k g_s^k) - f(u_s^k) \leq -c_2\lambda_k\|v_s^k\| < 0.$$

This inequality always implies that $\sigma_s \geq \lambda_k$. From the boundedness of the set $M(u^0)$ and Lemma 4 we get that there exists $\bar{\lambda} > 0$ such that for given k, $\lambda_k(u) \geq \bar{\lambda}$ for all $u\in M(u^0)$. Consequently $\sigma_s \geq \bar{\lambda}$ for all s. Then we have

$$\|v_s^k\| \leq (c_2\bar{\lambda})^{-1}[f(u_s^k)-f(u_{s+1}^k)]. \quad (19)$$

On the other hand $f(u_{s+1}^k) < f(u_s^k)$ for all k,s = 0, 1, 2,... and consequently, the limit

$$\lim_{s\to+\infty} f(u_s^k) \geq f_*$$

exists for any fixed k. Then from (19) it follows that there exists such finite $s(k) > 0$ that $\|v_{s(k)}^k\| \leq \delta_k$. Thus, in each stage the condition (17) holds after finite number of steps. At that $u^{k+1} = v_{s(k)}^k$ and therefore, $\min\{\|v\| : v\in\bar{D}_m(u^{k+1})\} \leq \delta_k$. Since $\bar{D}_m(u^{k+1}) \subset D_0(u^{k+1},z_k,\lambda_k,\beta_k)$ so

$$\bar{D}_m(u^{k+1}) \subset V(u^{k+1},\varepsilon_k).$$

Then we get that

$$\min\{\|v\| : v\in V(u^{k+1},\varepsilon_k)\} \leq \delta_k. \quad (20)$$

By the construction of the sequence $\{u^k\}$ it follows that $u^k\in M(u^0)$ for all k. Since $M(u^0)$ is the bounded set, the sequence $\{u^k\}$ is also bounded. Consequently, it has at least one accumulation point. Suppose u^* is the accumulation point of the sequence $\{u^k\}$ and assume that $u^k \to u^*$ as $k \to +\infty$, without loss of generality. The upper semicontinuity of the mapping V at the point $(u^*,0)$ implies that for any $\gamma > 0$ there exist $\Gamma > 0$, $\bar{\varepsilon} > 0$ such that

$$V(u,\varepsilon) \subset V(u^*,0) + S_\gamma \quad (21)$$

for all $u\in S_\Gamma(u^*)$ and $\varepsilon\in(0,\bar{\varepsilon})$. Since $u^k \to u^*$ and $\varepsilon_k \to +0$ so there exists k_0 so that $u^k\in S_\Gamma(u^*)$, $\varepsilon_k\in(0,\bar{\varepsilon})$ for all $k \geq k_0$. On the other hand since $V(u^*,0) = \partial f(u^*)$ so from (21) it follows that for all $k \geq k_0$

$$V(u^{k+1},\varepsilon_k) \subset \partial f(u^*) + S_\gamma.$$

From this and (20) we have

$$\min\{\|v\| : v \in \partial f(u^*)\} \le \delta_k + \gamma.$$

Since $\delta_k \to +0$ as $k \to +\infty$ and γ is an arbitrary number then we have that

$$\min\{\|v\| : v \in \partial f(u^*)\} = 0$$

or $0 \in \partial f(u^*)$. ♦

5. Computation of the Stepsize

In this section we discuss the problem of the computation of the stepsize σ_s in Step 5 of the above described method. For this we suggest the following three algorithms. Set $\theta = \|v_s^k\|$, where v_s^k is defined as in Step 3 of the method.

Algorithm 1

Let be given number $\gamma \in (0,1)$. We begin with $\sigma_s = \theta$ and if for $u_s^k + \sigma_s g_s^k$

$$f(u_s^k+\sigma_s g_s^k) - f(u_s^k) \le -c_2\sigma_s\|v_s^k\|, \tag{22}$$

then the algorithm stops. Otherwise we compute the next σ_s by $\sigma_s = \gamma\sigma_s$ and again check the condition (22) and so on. Since the condition (22) takes place for $\lambda_k > 0$ so if $\sigma_s \le \lambda_k$ then the algorithm stops and we take $\sigma_s = \lambda_k$.

Algorithm 2

In this algorithm we substitute the objective function along the descent direction by the one-dimensional quadratic function. We set $\sigma_s = \theta$. If $\sigma_s \le \lambda_k$ then we take $\sigma_s = \lambda_k$ and the algorithm stops. Otherwise we check the condition (22). If the condition (22) satisfies for $f(u_s^k+\sigma_s g_s^k)$ then the algorithm stops. Otherwise we define

$$\varphi(t) = f(u_s^k+tg_s^k).$$

Originally we have the following information on the function $\varphi(t)$:

$$\varphi(0) = f(u_s^k), \quad \varphi(\lambda_k) = f(u_s^k+\lambda_k g_s^k), \quad \varphi(\sigma_s) = f(u_s^k+\sigma_s g_s^k).$$

We denote by

$$\Gamma_1 = \lambda_k^{-1}[\varphi(\lambda_k) - \varphi(0)], \quad \Gamma_2 = \sigma_s^{-1}[\varphi(\sigma_s) - \varphi(0)].$$

Since $f(u_s^k+\sigma_s g_s^k)$ does not satisfy the condition (22) then

$$\varphi(\sigma_s) > \varphi(0) - c_2\sigma_s\|v_s^k\|. \tag{23}$$

Having this information on the function $\varphi(t)$ we substitute it by the following one - dimensional quadratic function:

$$\Psi(t) = at^2 + bt + \varphi(0),$$

where

$$a = \frac{\Gamma_2-\Gamma_1}{\sigma_s-\lambda_k}, \quad b = \frac{\sigma_s\Gamma_1-\lambda_k\Gamma_2}{\sigma_s-\lambda_k}.$$

The inequality (23) implies that $\Gamma_2 > -c_2\|v_s^k\|$. Then takingo into account that $\Gamma_1 \le -c_2\|v_s^k\|$ and $\sigma_s > \lambda_k$ we get that $a > 0$ and $b < -c_2\|v_s^k\|$. We compute the point $t_* = -b/2a$ for which $\Psi'(t_*) = 0$. Since $a > 0$, then t_* minimizes the function $\Psi(t)$. Moreover, $t_* > 0$ because $b < 0$. Therefore, we take t_* as a new value for σ_s. If $\sigma_s \le \lambda_k$ then we take $\sigma_s = \lambda_k$ and the

algorithm stops. If $\varphi(\sigma_s) = f(u^k+\sigma_s g_s^k)$ satisfies the condition (22) then the algorithm stops. Otherwise we repeat the algorthim for the new value of σ_s.

Algorithm 3

In this algorithm when the condition (23) takes place we substitute $\varphi(t)$ by the following one-dimensional piecewise linear function:

$$\Psi(t) = \max\{a_1t+b_1, a_2t+b_2\}.$$

Let $\gamma \in (0,1)$ be a given number. It is required that at the points $t = 0$ and $t = \lambda_k$

$$a_1t+b_1 > a_2t+b_2.$$

We also require that $a_2 > 0$. Thus, we get that $a_1 = \Gamma_1$, $b_1 = \varphi(0)$ and the coefficients a_2 and b_2 satisfy the following conditions:

$$b_2 < b_1, \quad a_2\lambda_k+b_2 < a_1\lambda_k+b_1, \quad a_2 > 0. \tag{24}$$

It is clear that $a_1 < -c_2\|v_s^k\|$. We set $\sigma_s = \theta$. If $f(u_s^k+\sigma_s g_s^k)$ does not satisfy the condition (22), then for the estimation of a_2 and b_2 we need to know one more value of the function $\varphi(t)$. Therefore, we compute the value of this function at the point $t = \gamma\sigma_s$. If $\varphi(\gamma\sigma_s) = f(u_s^k+\gamma\sigma_s g_s^k)$ satisfies the condition (22), then we take $\sigma_s = t$ and the algorithm stops. Otherwise, we can define a_2 and b_2 using $\varphi(\sigma_s)$ and $\varphi(\gamma\sigma_s)$. Thus, we have

$$a_2 = \frac{\varphi(\sigma_s)-\varphi(\gamma\sigma_s)}{\sigma_s(1-\gamma)}, \quad b_2 = \frac{\varphi(\gamma\sigma_s)-\gamma\varphi(\sigma_s)}{1-\gamma}.$$

If a_2 and b_2 do not satisfy the conditions (24) then we set $\sigma_s = \gamma\sigma_s$. If $\sigma_s \le \lambda_k$ then we take $\sigma_s = \lambda_k$ and the algorithm stops. Otherwise we repeat the algorithm. If a_2 and b_2 satisfy the conditions (24), then we compute

$$t_* = \frac{b_1-b_2}{a_2-a_1},$$

which minimizes the function $\Psi(t)$. It is clear that $t_* > 0$. We take t_* as a new value for σ_s. If $t_* \le \lambda_k$ then the algorithm stops and we take $\sigma_s = \lambda_k$. If $\sigma_s > \lambda_k$ and $f(u_s^k+\sigma_s g_s^k)$ satisfies the condition (22) then the algorithm stops. Otherwise we repeat it for the new value of σ_s.

6. Results of Numerical Experiments

In order to verify the practical efficiency of the suggested algorithms a number of numerical experiments have been carried out. In these experiments we considered problems with convex and nonconvex objective functions. So in Problems 1-6 the objective functions are convex (see, [15]) and in Problems 7-11 they are nonconvex. In Problem 12 we consider the marginal function of the special form. This problem is simple, but it is important for the comparison of algorithms. The codes have been written in Microsoft Fortran-90. Numerical experiments have been carried out in PC IBM AT 386 with Main Processor 80386DX, 40 MHz.

For solving the subproblem (10) the Wolfe method (see, [24]) is used. We take $c_1 = 0.2$ for all problems and $c_2 \in [0.001, 0.2]$. Parameters z, λ, β are chosen by the following form. $z_k(\lambda) = z(\lambda) = \lambda^{1.4}$, $\beta_k = \beta = 1$ for all k and problems. $\lambda_{k+1} = t\lambda_k$, $k = 0, 1, \ldots$, where $t = 0.75$ and $\lambda_0 = 0.001$ for all problems. In Algorithm 1 we considered the various values of the

number $\gamma \in (0,1)$ and chose the best results. So we got the best results for the Problems 1-5, 7-9,12 at $\gamma = 0.45$ and for the Problems 6,10,11 at $\gamma = 0.6$.

For description results of numerical experiments we use the following notations: $f = f(u)$ is the objective function, u^0 is the starting point, u^* the point of local minimum, $f_* = f(u^*)$, n the number of variables, N is the number of the problem, $\delta(u^k) = f(u^k) - f_*$ is the precision of the point u^k, m_1 is the number of iterations for achievement of the precision $\delta > 0$, that is

$$m_1 = \sum_{\delta(u^k) \geq \delta} s(k),$$

m_2 is the number of objective function evaluations at the line searching and m_3 is the total number of objective function evaluations for achievement of the precision $\delta > 0$.

Results of numerical experiments with $\delta = 10^{-2}$, $\delta = 10^{-3}$, $\delta = 10^{-4}$ are presented in Tables 1, 2 and 3, respectively.

To carry out numerical experiments we use the following problems:

Problem 1

$f(u) = \max\{u_1^2+u_2^4, (2-u_1)^2 + (2-u_2)^2, 2e^{-u_1+u_2}\}$, $u \in E_2$, $u^0 = (1,-0.1)$,
$u^* = (1.1390,0.8996)$, $f_* = 1.952224$.

Problem 2

$f(u) = \max\{f_i(u), i = 1,2,3\}$, $f_1(u) = u_1^2+u_2^2$, $f_2(u) = u_1^2+u_2^2+10(-4u_1-u_2+4)$,
$f_3(u) = u_1^2+u_2^2+10(-u_1-2u_2+6)$, $u \in E_2$, $u^0 = (-1,5)$, $u^* = (1.2,2.4)$, $f_* = 7.2$.

Problem 3

$f(u) = 4|u_1-u_2| + |u_1+u_2| + |u_2-u_3| + |u_2+u_3| + |u_3-u_4| + |u_3+u_4|$,
$u \in E_4$, $u^0 = (1,2,1,1)$, $u^* = (0,0,0,0)$, $f_* = 0$.

Problem 4

$f(u) = \max\{u_i^2, i = 1,\ldots, n\}$, $u \in E_n$, $u^0 = (i, i = 1,\ldots,]n/2[, -i, i =]n/2[+1,\ldots, n)$,
$u^* = (0,\ldots, 0)$, $f_* = 0$.

Problem 5

$f(u) = \max\{|u_i|, i = 1,\ldots, n\}$, $u \in E_n$, $u^0 = (i, i = 1,\ldots,]n/2[, -i, i =]n/2[+1,\ldots, n)$,
$u^* = (0,\ldots, 0)$, $f_* = 0$.

Problem 6

$$f(u) = \sum_{j=1}^{100} \left| \sum_{i=1}^{n} (u_i - u_i^*) t_j^{i-1} \right|, \quad u \in E_n, \ t_j = 0.01j, \ j = 1,\ldots, 100,$$

$u^0 = (0,\ldots, 0)$, $u^* = (1/n,\ldots, 1/n)$, $f_* = 0$.

Problem 7

$f(u) = |u_1-1| +100\, |u_2-|u_1||$, $u \in E_2$, $u^0 = (-1.2,1)$, $u^* = (1,1)$, $f_* = 0$.

Problem 8

$f(u) = |u_1-1| +100\,|u_2-|u_1|| +90\,|u_4-|u_3|| + |u_3-1| +10.1(|u_2-1| + |u_4-1|) + 4.95(|u_2+u_4-2| - |u_2-u_4|)$, $u\in E_4$, $u^0 = (1,3,3,1)$, $u^* = (1,1,1,1)$, $f_* = 0$.

Problem 9

$f(u) = \max\{u_i^2,\ i = 1,\ldots, n\} + \min\{u_i^2,\ i = 1,\ldots, n\}$, $u\in E_n$,

$u^0 = (i,\ i = 1,\ldots,]n/2[,\ -i,\ i =] n/2[+1,\ldots, n)$, $u^* = (0,\ldots, 0)$, $f_* = 0$.

Problem 10

$$f(u) = \sum_{j=1}^{20} |\sum_{i=1}^{n} (u_i-u_i^*)t_j^{i-1}| - \max\{|\sum_{i=1}^{n} (u_i-u_i^*)t_j^{i-1}|,\ j = 1,\ldots, 20\},\ u\in E_n,$$

$t_j = 0.05j$, $j = 1,\ldots, 20$, $u^* = (1/n,\ldots, 1/n)$, $f_* = 0$.

Problem 11

$$f(u) = \sum_{j=1}^{100} |\sum_{i=1}^{n} (u_i-u_i^*)t_j^{i-1}| - \max\{|\sum_{i=1}^{n} (u_i-u_i^*)t_j^{i-1}|,\ j = 1,\ldots, 100\},\ u\in E_n,$$

$t_j = 0.01j$, $j = 1,\ldots, 100$, $u^* = (1/n,\ldots, 1/n)$, $f_* = 0$.

Problem 12

$f(u) = \max\{\|x\|,\ x\in Z(u)\}$, $u,x\in E_n$,

$Z(u) = \mathrm{conv}\{A^i(u),\ i = 1,\ldots, 8\}$,

$A^1(u) = (A_1^1,\ldots, A_n^1)$, $A_j^1 = |u_j| +1$, $j = 1,\ldots, n$, $A^2(u) = -A^1(u)$,

$A^3(u) = (A_1^3,\ldots, A_n^3)$, $A_j^3 = (-1)^j\,(u_j^2+1)$, $j = 1,\ldots, n$, $A^4(u) = -A^3(u)$,

$A^5(u) = (A_1^5,\ldots, A_n^5)$, $A_j^5 = e^{|u_j|}$, $j = 1,\ldots, n$, $A^6(u) = -A^5(u)$,

$A^7(u) = (A_1^7,\ldots, A_n^7)$, $A_j^7 = (-1)^j\, e^{u_j^2}$, $j = 1,\ldots, n$, $A^8(u) = -A^7(u)$,

$u^0 = (1,\ldots, 1)$, $u^* = (0,\ldots, 0)$, $f_* = n^{1/2}$.

N	n	Algorithm 1			Algorithm 2			Algorithm 3		
		m_1	m_2	m_3	m_1	m_2	m_3	m_1	m_2	m_3
1	2	14	108	164	44	217	393	6	39	59
2	2	11	114	151	22	104	188	7	32	57
3	4	27	204	363	81	527	1088	36	180	376
4	5	10	20	80	5	10	40	10	20	80
4	10	21	42	273	10	20	130	21	42	273
4	15	37	76	668	15	30	270	84	219	1563
5	5	20	34	159	17	19	126	19	25	144
5	10	61	76	747	60	69	729	60	67	727
5	15	133	186	2329	128	150	2198	128	143	2191
6	5	64	1103	1882	62	317	1084	58	266	899
6	10	60	1203	2353	61	310	1561	78	305	2393
6	15	59	991	3810	46	208	2444	80	319	3189
6	20	59	1195	3634	47	249	2516	72	296	3308
7	2	56	293	655	40	107	363	23	37	195
8	4	40	192	848	71	219	1342	47	141	876
9	5	8	16	64	4	8	32	8	16	64
9	10	19	38	247	9	18	117	19	38	247
9	15	34	69	613	15	30	270	46	104	840
10	5	47	675	1117	42	182	569	43	192	590
10	10	45	739	1384	64	268	1322	63	282	1455
10	15	64	983	2382	62	263	1885	49	256	1460
11	5	61	1091	1737	57	275	877	76	381	1117
11	10	52	1015	1917	66	312	1948	74	315	1909
11	15	75	1287	4962	63	304	2932	93	371	4019
12	5	3	11	29	2	4	16	3	9	27
12	10	3	5	38	2	6	28	3	5	38
12	15	6	24	135	5	13	93	5	13	93

Table 1 - Comparison of Algorithms for δ = 1.0E-0.2

N	n	Algorithm 1			Algorithm 2			Algorithm 3		
		m_1	m_2	m_3	m_1	m_2	m_3	m_1	m_2	m_3
1	2	19	148	229	58	288	532	15	97	160
2	2	13	143	190	32	146	284	10	45	85
3	4	28	215	383	89	553	1214	43	197	480
4	5	12	24	96	5	10	40	12	24	96
4	10	27	54	351	10	20	130	27	54	351
4	15	44	90	794	15	30	270	92	235	1707
5	5	24	66	225	21	39	180	21	31	167
5	10	71	148	989	70	114	924	67	88	855
5	15	151	318	2899	144	216	2640	134	161	2335
6	5	91	1465	3121	84	389	1878	78	327	1600
6	10	118	2209	7357	104	460	4454	127	513	5190
6	15	96	1533	8184	108	474	7902	129	492	8061
6	20	134	2577	14571	71	320	5931	122	473	9775
7	2	63	358	757	44	114	402	24	41	204
8	4	55	257	1212	84	246	1618	58	170	1104
9	5	10	20	80	4	8	32	10	20	80
9	10	25	50	325	10	20	130	25	50	325
9	15	41	83	739	15	30	270	52	116	948
10	5	60	806	1526	49	212	726	60	250	1010
10	10	64	957	2461	123	437	4220	94	387	2931
10	15	126	1791	7947	101	382	5163	84	378	4572
11	5	88	1458	2926	95	388	2103	109	490	2004
11	10	125	2348	7483	112	485	4747	94	364	3258
11	15	117	1968	10200	103	463	7091	144	578	8612
12	5	5	26	61	3	9	27	6	25	61
12	10	6	29	105	5	22	87	8	28	136
12	15	14	95	349	6	18	114	8	27	155

Table 2 - Comparison of Algorithms for δ = 1.0E-0.3

N	n	Algorithm 1			Algorithm 2			Algorithm 3		
		m_1	m_2	m_3	m_1	m_2	m_3	m_1	m_2	m_3
1	2	22	169	265	63	305	574	16	103	171
2	2	16	165	227	36	160	318	14	61	123
3	4	44	288	695	103	583	1453	56	212	704
4	5	15	30	120	5	10	40	15	30	120
4	10	31	62	403	10	20	130	31	62	403
4	15	52	106	938	15	30	270	99	249	1833
5	5	35	110	409	31	54	300	21	31	167
5	10	80	183	1248	82	142	1168	77	100	1031
5	15	167	403	3593	159	251	3179	140	164	2519
6	5	104	1647	3736	120	490	3250	109	414	2788
6	10	279	6159	21298	222	988	12180	180	721	8641
6	15	165	2865	16740	218	945	18353	194	789	14408
6	20	194	3868	24262	197	811	24848	177	695	17792
7	2	75	425	910	54	128	492	38	49	330
8	4	70	380	1638	96	273	1885	72	193	1413
9	5	12	24	96	4	8	32	12	24	96
9	10	28	56	364	10	20	130	28	56	364
9	15	48	97	865	15	30	270	59	129	1088
10	5	94	1191	2665	164	652	3556	69	267	1236
10	10	102	1550	4892	227	837	9634	169	724	6783
10	15	168	2456	11909	175	656	11676	143	615	9998
11	5	200	3401	8335	269	1019	7658	152	658	3100
11	10	221	4568	15589	177	743	9020	168	685	7963
11	15	210	3862	22237	146	613	13194	227	975	17162
12	5	12	61	193	9	24	123	16	53	249
12	10	19	95	534	12	36	308	15	44	339
12	15	24	143	812	17	44	586	21	63	759

Table 3 - Comparison of Algorithms for δ = 1.0E-0.4

7. Concluding Remarks

In this paper, we propose three algorithms for solving unconstrained problem of Lipschitz optimization. These algorithms belong to those of nonsmooth optimization without calculating derivatives. The difference between them is in the computation of the step size. In Algorithm 1, we use a usual backtracking step. In Algorithm 2, the objective function is approximated by a one-dimensional quadratic function, and in Algorithm 3 it is approximated by a one-dimensional piecewise linear function along the descent direction. Some numerical experiments were carried out. In these experiments, max-type and min-type functions of quadratic ones (Problems 1,2,4,9,12), L_1-type functions (Problems 3,6,7,8) and mixed functions, which contain both max-type or min-type functions and L_1-type functions (Problems 5,10,11), were used. We present the results of numerical experiments for three values of $\delta : \delta = 0.01$, $\delta = 0.001$ and $\delta = 0.0001$. This allows one to study better the behavious of the suggested algorithms as a precise solution is approached. The results of the numerical experiments show that Algorithm 2 has an advantage to solve the minimization problems of max-type and min-type functions of quadratic ones and Algorithm 3 has an advantage to solve the minimization problems of L_1-type and mixed functions. Problems 1 and 2 are an exception here. Algorithm 3 was the best one for them. One can suppose that the objective functions in these problems are well approximated by a one-dimensional piecewise linear functions along the descent direction.

The comparison of the results, presented in Tables 1,2 and 3 shows that the advantage of Algorithms 2 and 3 becomes more obvious as a precise solution is approached. Comparing these results, we see that the number of computed discrete gradients in all algorithms at one iteration increases as a precise solution is approached, whereas the number of function evaluations in line searching at one iteration, as a rule, does not increase. Algorithms 2 and 3 use a much smaller number of function evaluations than Algorithm 1 in line searching.

The results of the numerical experiments also depend on the choice of the parameters $z \in P$, $\lambda > 0$, $\beta > 0$. Choosing them different for different functions, we can get better results for the concrete problem. But we took them the same for all the problems in the experiments.

It should be noted that the suggested algorithms allow us to solve all the problems tested with a given precision. This implies that they seem to be reliable algorithms.

As our main result, we show in this paper that one can construct more effective algorithms of nonsmooth optimization using the one-dimensional quadratic and piecewise linear models of the objective function along the descent direction.

Acknowledgments

I would like to express my sincere gratitude to Prof. L.N. Vicente and Prof. F. Jarre for the support. I am grateful to an anonymous referee for comments and suggestions that improved the paper.

References

[1] Bagirov, A.M., A method of approximating a subdifferential,Zh. Vichisl. Mat. Mat. Fiz. 32 (1992) 652-658; English transl. in: Russian Journal Computational Mathematics and Mathematical Physics 32 (1992) 561-566.

[2] Bagirov, A.M., Continuous approximation to a subdifferential of a function of a maximum, Kibernetika i sistemniy analiz 4 (1993) 180-184; English transl. in: Cybernetics and System Analysis 4 (1994) 626-630.

[3] Bagirov, A.M. and Gasanov, A.A., A method of approximating a quasidifferential, Zh. Vichisl. Mat. Mat. Fiz. 35 (1995) 511-519; English transl. in: Russian Journal Computational Mathematics and Mathematical Physics 35 (1995) 403-409.

[4] Bagirov, A.M., Continuous subdifferential approximation and its construction, Indian Journal of Pure and Applied Mathematics (1998).

[5] Bagirov, A.M., A method for minimizing convex functions based on continuous approximations to subdifferential, Journal of Optimization Methods and Software 9 (1998) 1-17.

[6] Brent, R.P., Algorithms for Minimization Without Derivatives, Prentice-Hall, Englewood Cliffs, New Jersey (1973).

[7] Clarke, F.H., Generalized gradients and applications, Trans. Amer. Math. Soc. 205 (1975) 247-262.

[8] Clarke, F.H., Optimization and Nonsmooth Analysis, John Wiley, New York (1983).

[9] Conn, A.R. and Toint, Ph. L., An algorithm using quadratic interpolation for unconstrained derivative free optimization, in: G.Di Pillo and F. Gianessi, eds., Nonlinear Optimization and Applications, Plenum Publishing, New York (1996) 27-47.

[10] Conn, A.R., Sheinberg, K. and Toint, Ph.L., Recent progress in unconstrained nonlinear optimization derivatives, Math. Programming, Th. M. Liebling and D. Werra eds., Series B 79 (1997) 397-414.

[11] Demyanov, V.F. and Vasilyev, L.V., Nondifferentiable Optimization, Nauka, Moscow (1981) (in Russian).

[12] Dennis, J.E. and Shnabel, R.B., Numerical Methods for Unconstrained Optimization and Nonlinear Equations, Prentice-Hall, Englewood Cliffs, New Jersey (1983).

[13] Kiwiel, K.C., Methods of Descent for Nondifferentiable Optimization, Lecture Notes in Mathematics 1133 (1985), Springer-Verlag, Berlin.

[14] Lemarechal, C., Extensions Diverses des Methodes de Gradient et Application, These d'etat, Paris (1980).

[15] Lemarechal, C., Numerical experiments in nonsmooth optimization, in: Progress in nondifferentiable optimization, ed. E.A. Nurminski, CP-82-5, International Institute for Applied System Analysis: Laxenburg, Austria (1982) 61-84.

[16] Mifflin, R., Semismooth and semiconvex functions in constrained optimization, SIAM Journal on Control and Optimization 15 (1977) 957-972.

[17] Polak, E. and Mayne, D.Q., Algorithm models for nondifferentiable optimization, SIAM Journal on Control and Optimization 23 (1985) 477-491.

[18] Powell, M.J.D., An efficient method for finding the minimum of a function of several variables without calculating derivatives, Computer Journal 17 (1964) 155-162.

[19] Powell, M.J.D., Recent advances in unconstrained optimization, Math. Programming 1 (1971) 26-57.

[20] Powell, M.J.D., A view of unconstrained minimization algorithms that do not require derivatives, ACM Transactions on Mathematical Software 1 (1975) 97-107.

[21] Powell, M.J.D., Approximation Theory and Methods, Cambridge University Press, Cambridge, UK (1981).

[22] Rzhevski, S.V., Monotone Methods of Convex Programming, Nauk. Dumka, Kiev, Ukraine (1993) (in Russian).

[23] Wolfe, P.H., A method of conjugate subgradients of minimizing nondifferentiable convex functions, Math. Programming Study 3 (1976) 145-173.

[24] Wolfe, P.H., Finding the nearest point in a polytope, Math. Programming 11 (1976) 128-149.

[25] Xu, H., Rubinov, A.M. and Glover, B.M., Continuous approximations to generalized Jacobians with application to nonsmooth least-squares minimization, Working Paper 17/96, School of Informations Technology and Mathematical Sciences, University of Ballarat, Australia (1996).

[26] Zowe, J., Nondifferentiable optimization: a motivation and a short introduction into the subgradient and the bundle concept, in NATO SAI Series, V. F15, Computational Mathematical Programming, K. Schittkowski ed., Springer-Verlag, Berlin (1985).

MODELOS DINÂMICOS BAYESIANOS PARA EXTREMOS

Maria José Schuwartz Ferreira

Departamento de Estatística-UFES/ES
Universidade Federal do Espírito Santo
Av. Fernando Ferrari, s/n
29060-900 Vitória-ES

Reinaldo Castro Souza

Pontifícia Universidade Católica do Rio de Janeiro
DEE-PUC/RJ
Av. Marquês de São Vicente, 225
22451-041 Rio de Janeiro-RJ

Gutemberg Hespanha Brasil

Departamento de Estatística-UFES/ES
Universidade Federal do Espírito Santo
Av. Fernando Ferrari, s/n
29060-900 Vitória-ES

Abstract

The analysis of extreme values data vis classical arguments, makes a through use of the so called extreme value distribution. An alternative approach, known in the hydrologic litterature as P.O.T. (Peaks Over Threshold), takes into account the values that exceed a given threshold mardk. However, both approaches do not allow for serial depedence in the data. In this paper, we develop forecasting models for extreme values time series. The models use the dynamic linear model as the model formulation and bayesian inference for parameter estimation and updating procedures. They are, therefore, included in the class of bayesian dynamic generalized models for extreme values time series. It is important to mention that the derivation is obtained directly without the use of any kinf of numerical approximations, even though the extreme values distribution is not a regular member of the exponential family of distributions. An applications is shown, where the problems related to the forecast of extreme values are discussed.

Resumo

As abordagens clássicas para estudos de valores extremos fazem uso das chamadas "distribuições de valores extremos". Uma abordagem alternativa, conhecida como P.O.T. (Peaks Over Threshold) desenvolvida por hidrologistas considera os valores que excedem um dado patamar (Threshold value). Essas metodologias não consideram explicitamente as propriedades seriais dos dados. Neste trabalho desenvolvemos modelos de previsão para valores extremos. Eles utilizam o modelo linear dinâmico como formulação subjacente e a inferência Bayesiana: são modelos dinâmicos Bayesianos generalizados para valores extremos. Embora essas distribuições não façam parte da família exponencial, toda a análise é feira explicitamente, sem aproximações numéricas. Problemas clássicos da previsão de extremos são apresentados em uma aplicação.

Keywords

Extreme values, bayesian dynamic models, gumbel distribution.

1. Introdução

Estatísticas de valores extremos são relevantes em muitas áreas e, uma grande variedade de aplicações é apresentada em Kinnison (1985). A grande maioria das aplicações utiliza os chamados métodos clássicos de estimação e previsão para extremos. Estes métodos apresentam algumas deficiências, como suposições de independência, a necessidade de transformar os dados e outras, apesar de proverem resultados muitas vezes satisfatórios. Ver por exemplo, Carter e Challenor (1978), Turner (1982), Smith (1984) e Ferreira e Souza (1988).

O objectivo principal deste artigo é desenvolver um modelo dinâmico Bayesiano para valores extremos aplicado a séries temporais. Nessa abordagem a série é descrita por componentes não-observáveis e na análise Bayesiana dinâmica é utilizado um processo sequencial de estimação. Pretende-se com esse procedimento estabelecer uma modelagem que considere explicitamente os elementos simplificadores e, muitas vezes restritivos, adotados na modelagem "clássica".

Na seção 2, descrevemos sucintamente dois dos métodos clássicos de estimação e previsão para extremos presentes na literatura; a exposição é ilustrada com um exemplo. Na seção 3, resumimos os resultados da inferência Bayesiana para o caso de amostra aleatória da distribuição de Gumbel. Algumas aplicações são feitas; Ferreira, Souza & Brasil (1989). Finalmente, na seção 4, desenvolveremos um modelo dinâmico Bayesiano para problemas de extremos de séries temporais.

A modelagem Bayesiana dinâmica foi introduzida por Harrison e Stevens (1971, 1976) para o caso de modelos lineares. Surgiram depois trabalhos estendendo a teoria (para a família exponencial) a modelos dinâmicos não lineares, como em Souza (1979), Migon (1984), West, Harrison e Migon (1985) e West e Harrison (1986). Pole, West e Harrison (1988) mostraram que o tipo de análise proposta por West, Harrison e Migon (1985) permite tratar distribuições não pertencentes à família exponencial. Descrevemos brevemente esse método na seção 4. Uma implementação desse método entretanto pode conduzir a cálculos numéricos não-tratáveis. É importante verificar se as informações adicionais sobre as distribuições podem levar a soluções mais simples.

No principais resultados deste trabalho é construída uma aplicação dessas ideias num caso específico, no qual a distribuição das observações é uma distribuição de extremos. Esse caso tem características próprias e as distribuições envolvidas são obtidas analiticamente. O resultado final é uma implementação do modelo dinâmico Bayesiano para estudo de valores extremos.

2. Alguns métodos para o estudo de valores extremos de séries temporais

2.1 Introdução

Estatísticas de valores extremos são relevantes em muitas áreas e uma grande variedade de aplicações é apresentada em Kinnison (1985). Nesta seção discute-se alguns métodos para

estudar valores extremos de séries temporais. Em geral, a definição dos extremos depende do processo envolvido e do tipo de método que será usado. Por exemplo, veja Yevjevich (1985): se a média temporal é uma série de vazões diárias, obtidas durante vários anos numa determinada seção de um rio, os extremos poderiam ser definidos como:

(i) as razões máximas anuais; ou,

(ii) os excessos de vazão em relação a um dado nível; ou,

(iii) os valores correspondentes aos excesssos de vazão acima de um dado nível; ou,

(iv) os intervalos de tempo correspondentes a estes excessos de vazão (intervalos entre extremos); ou,

(v) as n vazões de toda a amostra.

Como se pode ver, a definição do extremo depende fortemente do processo subjacente e do tipo de ensaio ou experimento proposto.

Nesta seção, discutiremos simplificadamente alguns métodos clássicos mais comuns. Os métodos clássicos mais utilizados são: (i) método de Gumbel; (ii) método P.O.T. e (iii) método P.O.T. modificado. As metodologias serão ilustradas através de um exemplo, a série de alturas de ondas no Porto de Praia Mole em Vitória, Espírito Santo, Brasil e apresentados na seção 5.1.2.

Na seção 2.2, descrevemos um método conhecido como método de Gumbel, descrito, por exemplo em Carter e Challenor (1978). Na seção 2.3 apresentamos o método "Peaks Over Threshold" (POT), Turner (1982) e Smith (184), que definem como extremos os valores que excedem a um dado nível, agrupados ou não. Algumas modificações deste método, conhecido como "P.O.T. modificado" encontram-se em Smith (184), e não são discutidas neste texto.

Nos estudos tradicionais sobre extremos, um dos objetos de interesse é a estimativa do valor H(Q), valor que é ultrapassado em média uma vez em cada Q anos (isto corresponde a especificar a probabilidade de ocorrência deste valor, ou de valores maiores, como 1/Q). Nos exemplos deste trabalho, estaremos interessados na altura de onda que retorna em 50 anos, H(50).

2.2 Método de Gumbel

Este método é baseado na abordagem clássica de Gumbel (1958). Resumidamente, no método de Gumbel, onde cada um dos anos de uma série temporal é dividido em N grupos, digamos N = 12, e toma-se o valor máximo de cada grupo. Define-se M_{ij} como o máximo do i-ésimo grupo do j-ésimo ano, i = 1, 2,..., N e j = 1, 2,..., P, onde P é o último ano disponível. Ajustando-se para cada grupo (M_{i1}, M_{i2},..., M_{ip}) uma distribuição de Gumbel, $F_i(y)$, pode-se estimar $H_i(Q)$ (o valor que retorna em Q anos no i-ésimo período). No caso do

exemplo do porto de Praia Mole temos (Grupos - 1 a 12 e, Anos -1982, 1983,...):

$$\text{Grupos}\begin{bmatrix} & \text{Anos} & & \\ M_{1,1} & M_{1,2} & M_{1,3} & \cdots \\ M_{2,1} & M_{2,2} & M_{2,3} & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ M_{12,1} & M_{12,2} & M_{12,3} & \cdots \end{bmatrix}$$

Note que os valores consecutivos no tempo aparecem em coluna e que uma dada linha congrega os máximos em períodos correspondentes, ano após ano. Ajusta-se a cada uma das linhas desta matriz uma distribuição de Gumbel $F_i(y)$. Estima-se $H_i(Q)$ utilizando essa distribuição e define-se H(Q) como o máximo dos $H_i(Q)$.

Este método evita a necessidade de estudar previamente a sazonalidade da série visto que, ajusta-se uma distribuição de Gumbel aos máximos obtidos em Janeiro, ano após ano; uma distribuição de Gumbel (com possivelmente outros parâmetros) aos máximos obtidos em Fevereiro, ano após ano, e assim por diante; porém ele não detecta uma possível tendência, nem uma possível alteração dos padrões sazonais.

A descrição e algumas aplicações deste método podem ser encontradas em: Carter, D.J.T. and Challenor, P.G. (1978) e Ferreira, M.J.S. (1991).

Uma estimativa de H(Q) pode ser obtida do seguinte modo. Considere Q = 50 anos. Então, $H_i(Q)$ é definida pela expressão

$$P[Y > H_i(50)] \equiv 1 - F_i[H_i(50)] = 1/50 \qquad (2.1)$$

e

$$F_i(y_i) = P[Y \leq y_i] = \exp\left[-\exp\left[-\frac{(y_i-\eta_i)}{\varnothing_i}\right]\right] \quad -\infty < \eta_i < \infty;\ \varnothing_i > 0 \qquad (2.2)$$

é a função distribuição acumulada de Gumbel, onde η_i é o parâmetro de locação da distribuição e $\varnothing_i$ é o parâmetro de escala. Este parâmetros são estimados pelo método da máxima verosimilhança, Carter & Challenor (1978) e Ferreira (1991).

Para obter uma altura de referência anual, podemos raciocinar de duas maneiras:

(i) $H_a(50)$, definida como o máximo dos $H_i(50=, i = 1, 2,..., N$; nesse caso, $H_i(Q)$ é dada por:

$$H_i(Q) \equiv y_i = \eta_i - \varnothing_i.\ln\left[-\ln\left(1 - \frac{1}{Q}\right)\right] \qquad (2.3)$$

(ii) supondo independência entre as linhas, define-se $H_b(50) = y_b$, a partir da distribuição conjunta F(y) das linhas, do seguinte modo:

$$P(Y_1 > y_b \text{ ou } Y_2 > y_b \text{ ou} \ldots Y_N > y_b) = 1 - F_1(y_b).F_2(y_b)\ldots F_N(y_b) = 1/50. \qquad (2.4)$$

Para calcular $H_b(50) = y_b$ um bom procedimento é o método de Newton-Raphson. A tabela 2.1 mostra os valores de $H_a(50)$ e $H_b(50)$ para os dados do Porto de Praia Mole. Ferreira (1991) apresenta todos os cálculos. O valor escolhido pode ser, por exemplo, o maior deles: 8,05 metros.

N	$H_a(50)$	$H_b(50)$
1	6,61	6,61
2	6,61	6,65
3	6,74	7,30
4	6,98	7,38
6	6,93	7,46
9	7,10	7,85
12	7,08	7,74
18	7,14	8,05

Tabela 2.1 - Valores dos estimadores de $H_a(50)$ e $H_b(50)$ - Método de Gumbel

2.3 Método P.O.T. (Peaks Over Threshold)

O Método Peaks Over Threshold (P.O.T.) foi desenvolvido por hidrologistas. Algumas referências para essa metodologia são: North (1980), Turner (1982) e Smith (1984). Neste método em geral, é necessário transformar previamente os dados, pois ele foi desenvolvido para séries estacionárias, i.e. são retirados das séries os possíveis componentes sazonais e de tendência.

A seguir define-se um patamar TL (Threshold Limit), acima do qual os valores são considerados de interesse. Os valores acima do patamar aparecem geralmente em grupos de instantes consecutivos. O valor máximo em um dado grupo é denominado o pico desse grupo. Considera-se somente esses picos (máximos dos grupos). Os tempos correspondentes aos picos formam, sob certas condições, um processo de Poisson; Turner (1982) e Smith (1984).

Denotando-se:

Y_i - a diferença entre o valor do pico e o patamar TL e,

T_i - o instante correspondente ao pico,

os valores dos Y_i são modelados por uma distribuição exponencial e, os T_i formam um processo de Poisson com parâmetro λ.

Segue-se então que as ocorrências de valores acima do patamar (valores excedentes) em intervalos disjuntos são eventos independentes e o número desses valores num dado intervalo tem uma distribuição de Poisson.

A diferença entre o pico e o patamar é modelada por uma distribuição exponencial. Esta escolha é devida à propriedade de "falta de memória", ou seja $P(Y > y+c \mid Y > c) = P(Y > y)$. Essa propriedade torna as conclusões, em certa medida, independentes do patamar escolhido. Outra justificativa para a escolha da distribuição exponencial pode ser encontrada em Smith (1984). A partir destas hipóteses é feita então a estimativa de H(Q).

O método P.O.T. é muito subjetivo:

(i) não existe critério para escolha da altura do patamar (TL), na modelagem dos valores excedentes e no processo de agrupamento dos dados.

(ii) não trata explicitamente os componentes de sazonalidade e tendência; e,

(iii) considera as observações como independentes e identicamente distribuídas.

Tentativas feitas pr Smith (1984) - método P.O.T. modificado, para definir o tamanho dos grupos contribuíram para resolver esse problema, à custa de tornar este método difícil de ser usado.

3. Inferência Bayesiana para uma Distribuição de Extremos

3.1 Inferência Bayesiana para valores extremos

Apresentamos sumariamente os resultados da inferência Bayesiana para uma amostra aleatória de valores provenientes de uma distribuição de Gumbel. Este resultados é interessante de ser observado antes do modelo dinâmico. A seção 3.2 é finalizada com uma estimativa de H(50).

Suponha uma amostra aleatória de tamanho t, $\underline{Y} = (Y_1, Y_2, \ldots, Y_t)$ de um modelo de Gumbel com função densidade de probabilidade (fdp) dada por

$$P[Y|\eta,\emptyset] = \frac{1}{\emptyset}.e^{\frac{-(y-\eta)}{\emptyset}} \exp\left[-e^{\frac{-(y-\eta)}{\emptyset}}\right] = \frac{1}{\emptyset}.\exp\left[-\left(\frac{y-\eta}{\emptyset}\right)\right].\exp\left[-\exp-\left(\frac{y-\eta}{\emptyset}\right)\right], \quad -\infty < y < \infty$$

Onde: η, $(-\infty < \eta < \infty)$, é o parâmetro de locação e $\emptyset$, $(\emptyset > 0)$, o parâmetro de escala. Observa-se que η é a moda da distribuição. No desenvolvimento supomos $\emptyset$ conhecido.

Na terminologia de Harrison & Stevens (1976) e West & Harrison (1989) temos, nesse caso, um modelo estático:

Equação das Observações: $Y_t \sim$ Gumbel $(\eta_t,\emptyset_t) \equiv$ Gumbel $(\eta,\emptyset)$; $\emptyset \in R^+$, conhecido

Equação do Sistema: $\eta_t = \eta_{t-1} = \eta, \ \forall t = 1, 2, \ldots$

A actualização sequencial de $\eta_t = \eta_{t-1} = \eta$, pode ser obtida utilizando-se: (i) uma priori não-informativa para η, (ii) o modelo observacional de Gumbel, e (iii) o teorema de Bayes para obtenção da posteriori e η.

Priori não-informativa

A priori não-informatica para o parâmetro η pode ser estabelecida utilizando-se a regra de Jeffreys, Box & Tiao (1973): $P[\eta] \propto |I(\eta)|^{1/2}$; inde $I(\eta)$ é a informação de Fischer para η. A priori não-informativa de Jeffreys fica:

$$P(\eta|\emptyset) \propto |I(\eta)|^{1/2} = \frac{1}{\emptyset}.$$

A verosimilhança da amostra é dada por:

$$P[\underline{Y}|\eta,\emptyset] = \frac{1}{\emptyset^t}.\exp\left[-\sum_{i=1}^{t}\frac{y_i-\eta}{\emptyset}\right].\exp\left[-\sum_{i=1}^{t}\exp\left\{\frac{y_i-\eta}{\emptyset}\right\}\right]$$

A distribuição a posteriori de η é calculada via teorema de Bayes (em forma fechada):

$$P(\eta|\underline{Y},\emptyset) \propto P(\underline{Y}|\eta,\emptyset).P(\eta|\emptyset)$$

$$P(\eta|\underline{Y},\emptyset) = \frac{\exp\frac{t.\eta}{\emptyset}}{\emptyset.(t-1!)}\left(\sum_i \exp-\frac{y_i}{\emptyset}\right)^t.\exp\left[-\sum_{i=1}^{t}\exp\left(-\frac{y_i-\eta}{\emptyset}\right)\right]$$

Os dois primeiros momentos da distribuição a posteriori de η podem ser calculados através das próprias definições:

$$E(\eta|\emptyset,\underline{Y}) = \emptyset.\left[-C+\sum_{k=1}^{t-1}\frac{1}{k}-\ln\left(\sum_{i=1}^{t}\exp(-\frac{y_i}{\emptyset})\right)\right] \quad (3.1)$$

$$V(\eta|\emptyset,\underline{Y}) = \emptyset^2.\left[\frac{\pi^2}{6}-\sum_{k=1}^{t-1}\frac{1}{k^2}\right] \quad (3.2)$$

A moda M da distribuição a posteriori pode ser calculada derivando-se a densidade a posteriori com relação a η e igualando-se a zero:

$$M(\eta|\emptyset,\underline{Y}) = \emptyset.\left[\ln(t)-\ln\left(\sum_{i=1}^{t}\exp(-\frac{y_i}{\emptyset})\right)\right]$$

Demonstra-se também que, a média da distribuição a posteriori de η, $E(\eta|\underline{Y},\emptyset)$, é um estimador não-tendencioso de η; entretanto, a moda é um estimador tendencioso de η; Ferreira (1991, pag. 53-54).

Distribuição preditiva

A função de densidade preditiva pode ser determinada através dos argumentos Bayesianos tradicionais:

$$P[Y_{t+1}|\underline{Y}_t] = \int_{-\infty}^{\infty} P(Y_{t+1}|\eta,\emptyset,\underline{Y}_t).P(\eta|\emptyset,\underline{Y}_t)d\eta$$

$$P[Y_{t+1}|\underline{Y}_t] = \frac{t}{\emptyset}.\left(\sum_{i=1}^{t}\exp-\frac{y_i}{\emptyset}\right)^t.\exp\left(-\frac{y_{t+i}}{\emptyset}\right).\left(\sum_{i=1}^{t}\exp(-\frac{y_i}{\emptyset})+\exp(-\frac{y_{t+1}}{\emptyset})\right)^{-(t+1)}$$

A média e a variância da distribuição preditiva podem ser calculadas via função geratriz de momentos, e são dadas por:

$$E[Y_{t+1}|\underline{Y}_t] = \emptyset.\left[\sum_{k=1}^{t-1}\frac{1}{k}-\ln\left(\sum_{i=1}^{t}\exp(-\frac{y_i}{\emptyset})\right)\right]$$

$$V[Y_{t+1}|\underline{Y}_t] = \emptyset^2.\left[\frac{\pi^2}{3}-\sum_{k=1}^{t-1}\frac{1}{k^2}\right]$$

Todos os cálculos desta seção encontram-se desenvolvidos detalhadamente em Ferreia (1991, capítulo 3).

3.2 Cálculo de H(50)

Nesta seção estimamos o valor H(50), que é ultrapassado em média uma vez em cada 50 anos, para a série de alturas de ondas, no Porto de Praia Mole, ES/Brasil. O esquema utilizado é análogo ao método (a) de Gumbel, da (seção 2.2) com a diferença de que o parâmetro η, é estimado pela média da distribuição a posteriori, $E(\eta|\emptyset,\underline{Y})$, calculada na seção 3.1.

Lembrando a expressão (2.3), temos:

$$H_i(50) = \eta_i - \emptyset_i.\ln[-\ln(1-1/50)] = \eta_i + 3{,}90.\emptyset_i \quad (3.3)$$

Observando-se que M_{ij} representa o máximo do i-ésimo grupo no j-ésimo ano, vamos considerar que os máximos M_{i1}, M_{i2},... em períodos correspondentes, ano após ano, provêm de uma distribuição de Gumbel com parâmetros η_i e $\emptyset_i$. Seja $M_i = (M_{i1}, M_{i2}, \ldots)$. Este vetor pode então ser considerado como o vetor de observações, mais precisamente como uma amostra aleatória, analogamente à seção anterior.

Com esta notação, a partir da equação (3.1), podemos escrever:

$$E[\eta_i|\emptyset_i,M_i] = \emptyset_i.\left[-C+\sum_{k=1}^{t-1}\frac{1}{k}-\ln\left(\sum_{j=1}^{t}\exp(-\frac{M_{ij}}{\emptyset_i})\right)\right] \qquad (3.4)$$

A partir das equações (3.3) e (3.4) e estimando $\emptyset_i$ pela variância amostral, temos a estimativa de H_i (50). A tabela 3.1 mostra essas estimativas.

$\emptyset_i$	$E(\eta_i\|\emptyset_i,M_i)$	H_i (50)
0,50	3,10	5,05
0,43	3,02	4,70
0,22	2,67	3,53
0,46	3,61	5,40
0,05	4,03	4,23
0,75	4,30	7,23
0,45	3,31	5,07
0,30	3,05	4,22
0,84	3,20	6,48
0,33	2,65	3,94
0.46	4,73	6,52
0,22	2,68	3,54

Tabela 3.1 - Estimativas de $\emptyset_1$, η_i e H_i (50) (estimativa Bayesiana-amostra aleatória)

A partir da tabela 3.1 definimos H(50= = Máx H_i (50) = 7,23 metros resultado que pode ser confrontado com o valor H_b (50) = 8,05 da seção 2.2.

4. Modelo Dinâmico Bayesiano para Valores Extremos

4.1 Introdução

Harrison & Stevens (1971,1976) introduziram a abordagem Bayesiana para os modelos lineares dinâmicos (representação em espaços de estados), estendendo e incorporando novos pontos de vista aos trabalhos de Kalman (1960), Kalman e Bucy (1961) e Kalman (1963).

A modelagem de séries temporais via "MLD's - modelos lineares dinâmicos" foi estendida no sentido de tornar o procedimento de modelagem mais flexível, por exemplo: simplificando a especificação da matriz de covariância do sistema e/ou abrangendo modelos dinâmicos não lineares no contexto da família exponencial; Souza (1979), Ameen & Harrison (1984,1985), Migon (1984), West, Harrison & Migon (1985) e West & Harrison (1986,1989).

Neste modelos, as propriedades da família exponencial aparecem na forma de uma conjugação entre as distribuições a priori e a posteriori, o que permite que todo o processo de inferência seja realizado de forma analítica sem o recurso a aproximações. Assim, o tratamento numérico da atualização, estimação e predição é relativamente simples.

Pole, West & Harrison (1988) mostram que, em princípio, o mesmo tipo de modelagem proposta por West, Harrison & Migon (1985), pode ser utilizada para distribuições não pertencentes à família exponencial. Na seção 4.2, apresentamos o modelo dinâmico Bayesiano generalizado. Na seção 4.3 desenvolvemos um modelo dinâmico Bayesiano para valores extremos, particularmente a distribuição de Gumbel, o qual é uma instância do modelo anterior, tendo no entanto características próprias, pois as distribuições envolvidas exibem propriedade semelhante à conjugação, embora não sejam da família exponencial.

4.2 Modelo dinâmico Bayesiano generalizado

O modelo apresentado nesta seção é uma extensão do Modelo Dinâmico Linear de West, Harrison e Migon (1985). O Modelo Dinâmico Generalizado é dado por:

Modelo Observacional

$$P(Y_t|\eta_t);\ g(\eta_t) \cong \lambda_t = F_t(\underline{\theta}_t) \tag{4.2.1}$$

e Equação de Estado

$$\underline{\theta}_t = \underline{G}_t\,(\underline{\theta}_{t-1}) + \underline{w}_t,\ \underline{w}_t \sim (0, \underline{W}_t) \tag{4.2.2}$$

Onde para cada instante t são conhecidas as funções $F_t(\cdot)$,

$\underline{G}_t\,(\cdot)$, e as variância V_t e $\underline{W}_t$;

y_t são as observações (supostas univariadas);

$\underline{\theta}_t$ é o valor (n×1) de parâmetros desconhecidos, denominado vetor de estado do sistema;

$F_t(\cdot)$ é uma função, em geral não linear; duas vezes diferenciável e conhecida para todo t;

$g(\cdot)$ é uma função de ligação monótona e diferenciável, também conhecida para todo t;

λ_t é uma variável aleatória;

$\cong$ sinal de associação, denominado relação guia;

$\underline{G}_t\,(\cdot)$ é uma função vetorial, em geral não linear;

$\underline{W}_t$ é uma matriz (n×n), covariâncias do sistema;

$\underline{\theta}_0 \sim [\underline{m}_0, \underline{C}_0]$ é o estado inicial.

A análise da série temporal é feita em termos das componentes não observáveis: tendência (μ_t), sazonalidade (γ_t), ciclo (Ψ_t) e componente irregular (v_t). Em geral, o modelo de decomposição tem duas formas:

$$Y_t = \mu_t \,.\, \gamma_t \,.\, \Psi_t \,.\, v_t, \quad \text{modelo multiplicativo} \tag{4.2.3}$$

ou

$$Y_t = \mu_t + \gamma_t + \Psi_t + v_t, \quad \text{modelo aditivo} \tag{4.2.4}$$

O modelo adotado será o modelo aditivo. Vamos descrever suas componentes.

Componentes de tendência (μ_t) - é uma função que varia suavemente no tempo caracterizada pelos parâmetros:

μ_{1t}, que representa o nível da tendência no instante t; e

β_t, que representa o crescimento entre os instantes t-1 e t (fator de crescimento).

Desse modo, o vetor de parâmetros é dado por $\underline{\theta}_{1,t}^T = [\mu_{1t},\beta_t]^T$.

Componente sazonal (γ_t). - é caracterizada como sendo a característica do processo em repetir um certo tipo de comportamento dentro de um período sazonal (s) que se repete anualmente.

A modelagem da sazonalidade será feita através de fatores sazonais. Será estabelecida a seguinte restrição

$$E\left[\sum_{i=1}^{s} \gamma_{it}\right] = 0, \text{ onde s é a periodicidade e } \gamma_{it},\ i = 1,\ldots, s, \text{ são os fatores sazonais.}$$

O vetor de parâmetros é dado por $\underline{\theta}_{2,t}^T = [\gamma_{1t},\gamma_{2t},\ldots,\gamma_{st}]^T$

Componente cíclica - são movimentos oscilatórios em torno da tendência, podendo ser periódicos ou não, com período maior que um ano. Um ciclo tem início e fim. A componente cíclica é definida pela componente cíclica total Ψ_t e o termo associado Ψ_t^*, onde

$$\Psi_{.t} = \rho.[\cos(\lambda.\Psi_{t-1}) + \text{sen}(\lambda.\Psi_{t-1}^*)]$$
$$\Psi_t^* = \rho.[\text{sen}(\lambda.\Psi_{t-1}) + \cos(\lambda.\Psi_{t-1}^*)]$$

onde: $0 \leq \lambda \leq \pi$, é a frequência da senóide e $0 < \rho \leq 1$, é o fator de amortecimento da amplitude. Se considerarmos ρ e λ conhecidos para todo t, tem-se um modelo linear. O vetor de parâmetros é dado por $\underline{\theta}_{3,t}^T = [\Psi_t,\Psi_t^*]^T$

Componente irregular - são os movimentos irregulares a partir da tendência constituindo um componente ruído branco.

A distribuição amostral ou observacional é uma distribuição qualquer denotada por

$$P(Y_t|\eta_t,\emptyset_t),$$

onde η_t e $\emptyset_t$ são parâmetros da distribuição, com $\emptyset_t > 0$. Usualmente η_t é um parâmetro de locação e $\emptyset_t$ é um parâmetro de escala; Y_t é a variável aleatória que representa as observações da série.

Sejam $E(Y_t|\eta_t,\emptyset_t)$ e $V(Y_t|\eta_t,\emptyset_t)$ respectivamente a média e a variância das observações. Há necessidade de uma distribuição a priori para o parâmetro η_t. Existem métodos para especificar a distribuição a priori, mas esta pode ser arbitrária. A função de densidade de probabilidade, o valor esperado e a variância dessa priori, estão representadas genericamente, pelas equações (4.2.5) (4.2.6) e (4.2.7), respectivamente.

$$P(\eta_t|D_{t-1},\emptyset_t) \qquad (4.2.5)$$
$$E(\eta_t|D_{t-1},\emptyset_t) \qquad (4.2.6)$$
$$V(\eta_t|D_{t-1},\emptyset_t) \qquad (4.2.7)$$

onde D_{t-1} representa a informação acumulada até t-1 (muito frequentemente $D_{t-1} = \{Y_0, Y_1, \ldots, Y_{t-1}\}$ constitui apenas o conjunto das observações até t-1). Observe que, para cada caso, deve ser especificado o modelo genérico $P(\eta_t|D_{t-1},\emptyset_t)$ e calculados seus momentos (4.2.6) e (4.2.7). Da densidade amostral $P(Y_t|\eta_t,\emptyset_t)$ e da densidade da priori do parâmetro η_t (4.2.5), determinamos a distribuição marginal para Y_t dados D_{t-1} e $\emptyset_t$, ou seja

$$P(Y_t|D_{t-1},\emptyset_t) = \int P(Y_t|D_{t-1},\emptyset_t,\eta_t).P(\eta_t|D_{t-1},\emptyset_t)d\eta_t \qquad (4.2.8)$$

A evolução do instante t-1 para o instante t e a relação entre o parâmetro η_t e o valor do estado $\underline{\theta}_t$, serão descritas a seguir.

A variável aleatória λ_t é relacionada ao vetor de estado $\underline{\theta}_t$, por

$$\lambda_t = F_t(\underline{\theta}_t),\ \lambda_t|D_{t-1} \sim [f_t,q_t] \qquad (4.2.9)$$

Podemos relacionar η_t a λ_t por: $\qquad g(\eta_t) \cong \lambda_t = F_t(\underline{\theta}_t) \qquad (4.2.10)$

Como dissemos anteriormente $\cong$ é apenas um sinal de associação, denominado relação guia, que não obriga a serem iguais as distribuições a priori das variáveis $g(\eta_t)$ e λ_t. De fato, esta igualdade seria uma restrição muito severa. A relação entre $g(\eta_t)$ e λ_t é usada simplesmente como um guia para determinar a priori para η_t, (veja West, Harrison e Migon (1985), pág. 76).

Desse modo a relação $g(\eta_t) \cong \lambda_t$ deve ser suficientemente flexível de modo a possibilitar a incorporação na priori de $g(\eta_t)$ de informações adicionais. A evolução temporal do modelo é dada pela equação (4.2.2). A distribuição a posteriori do vetor de estado no instante t-1 é em geral desconhecida. Seus dois primeiros momentos, no instante anterior, são conhecidos por hipótese.

$$(\underline{\theta}_{t-1}|D_{t-1}) \sim [\underline{m}_{t-1},\underline{C}_{t-1}] \qquad (4.2.11)$$

Através de (4.2.2) e (4.2.10) determinamos os dois momentos da distribuição priori de $\underline{\theta}_t$, ou seja $(\underline{\theta}_{t-1}|D_{t-1})$.

Então

$$(\underline{\theta}_t|D_{t-1}) \sim [\underline{a}_t,\underline{R}_t] \qquad (4.2.12)$$

onde

$$\underline{a}_t = E(\underline{\theta}_t|D_{t-1}) = \underline{G}_t\,(\underline{m}_{t-1}) \qquad (4.2.13)$$

$$\underline{R}_t = Var(\underline{\theta}_t|D_{t-1}) = \underline{H}_t.\underline{C}_{t-1}.\underline{H}_t^T + \underline{w}_t \qquad (4.2.14)$$

onde $\underline{H}_t$ é a matriz (n×n) de primeiras derivadas de $\underline{G}_t\,(\underline{\theta}_t)$ avaliada em $\underline{\theta}_{t-1} = \underline{m}_{t-1}$.

Na expressão (4.2.14) $\underline{w}_t$ representa o aumento da incerteza de $\underline{\theta}_t$ do instante t-1 para o instante t. Representaremos este acréscimo aditivo da incerteza de uma forma equivalente multiplicativa, usando fatores de desconto. Detalhes dessa abordagem podem ser encontrados em Ameen e Harrison (1985), West e Harrison (1986,1989) e Brasil (1989). Equivalentemente podemos escrever (4.2.14) como:

$$\underline{R}_t = \underline{\Delta}_t.\underline{H}_t.\underline{C}_{t-1}.\underline{H}_t^T.\underline{\Delta}_t. \qquad (4.2.15)$$

onde $\underline{\Delta}_t$ é uma matriz diagonal, (n×n), de fatores de desconto. A partir deste ponto o procedimento de atualização de $\underline{\theta}_t$ e λ_t segue a mesma linha do modelo de West, Harrison e Migon (1985). Não é necessário conhecer a distribuição conjunta de λ_t e $\underline{\theta}_t$. Esta distribuição

será dada pelos dois primeiros momentos. Assumiremos que a distribuição marginal $\lambda_t|D_{t-1}$ seja conhecida em qualquer aplicação particular, ou seja,

$$\left.\begin{matrix}\lambda_t \\ \\ \\ \underline{\theta}_{t-1}\end{matrix}\right| \sim \left[\begin{bmatrix} f_t \\ \underline{a}_t \end{bmatrix}, \begin{bmatrix} q_t & \underline{S}_t^T \\ \underline{S}_t & \underline{R}_t \end{bmatrix}\right]$$

$$E(\lambda_t|D_{t-1}) = F_t^T(\underline{a}_t) = f_t \tag{4.2.17}$$

$$V(\lambda_t|D_{t-1}) = \underline{h}_t^T.\underline{R}_t.\underline{h}_t = q_t \tag{4.2.18}$$

onde $\underline{h}_t$ é o vetor (n×1) de primeiras derivadas de $F_t(.)$ avaliadas em $\underline{\theta}_t = \underline{a}_t$.

$$S_t = Cov[\underline{\theta}_t, \lambda_t|D_{t-1}] = \underline{R}_t.\underline{h}_t \tag{4.2.19}$$

Para atualizar os componentes do modelo, será necessário estimar inicialmente os dois primeiros momentos da posteriori para λ_t, levando em conta que esta distribuição a posteriori é aproximada pela posteriori de $g(\eta_t)$, conforme a relação guia.

Para o vetor de estado $\underline{\theta}_t$, os momentos são atualizados via métodos Lineares Bayesianos [Hartigan (1969) e Goldstein (1976)]. As relações de recorrência são exatamente as mesmas apresentadas por West, Harrison e Migon (1985).

$$(\underline{\theta}_t|D_t) \sim [\underline{m}_t, \underline{C}_t] \tag{4.2.20}$$

onde

$$\underline{m}_t = \underline{a}_t + \underline{S}_t.\left[\frac{E(\lambda_t|D_t)-f_t}{q_t}\right]$$

$$\underline{C}_t = \underline{R}_t - \underline{S}_t.\underline{S}_t^T.\left[1 - \frac{V(\lambda_t|D_t)}{q_t}\right].q_t^{-1}$$

Finalmente, os momentos de λ_t nestas expressões são estimados por estatísticas provenientes da distribuição a posteriori de $g(\eta_t)$, conforme a relação guia.

4.3 Modelo dinâmico Bayesiano para valores extremos

Desenvolveremos aqui um modelo que é uma realização do esquema geral da seção 4.2. Nesse modelo, a distribuição a priori inicial do parâmetro de locação é uma priori não informativa de Jeffreys, que é uma distribuição imprópria. Isto não afeta o desempenho do modelo, pois todo o desenvolvimento depende da distribuição a posteriori, da distribuição preditiva e do procedimento "Linear Bayes" de atualização do vetor de estado, o qual não utiliza de modo essencial os momentos da distribuição a priori de $g(\eta_t)$. Alternativamente poder-se-ia adotar um modelo no qual a distribuição da priori do parâmetro de locação é a chamada "priori de referência", Bernardo (1979) e Pole & West (1987). Este desenvolvimento encontra-se em Ferreira (1991).

No modelo a distribuição das observações é uma distribuição de Gumbel, cujo f.d.p. é

expressa por:

$$P(Y_t|\eta_t,\emptyset_t) = \frac{1}{\emptyset_t}.\exp\left[-\exp\left(-\frac{(y_t-\eta_t)}{\emptyset_t}\right)\right].\exp\left[-\frac{(y_t-\eta_t)}{\emptyset_t}\right] \quad (4.3.1)$$

$-\infty < \eta_t < \infty$; $-\infty < Y_t < \infty$ e $\emptyset_t > 0$;

onde: η_t é a moda da distribuição e $\emptyset_t$ é o parâmetro de escala.

Mostra-se que (Johnson & Kotz, 1970), o valor esperado e a variância da distribuição são dadas por:

$$\mu_t = E(Y_t|\eta_t,\emptyset_t) = \eta_t + C.\emptyset_t, \quad (4.3.2)$$

onde C é a constante de Euler, C = 0,57721566490

$$V(Y_t|\eta_t,\emptyset_t) = \frac{\pi^2}{6}.\emptyset_t^2 \quad (4.3.3)$$

Apresentamos na figura 4.1 o gráfico da densidade de Gumbel (modelo observacional).

$\eta = 10, \emptyset = 1$ e $-\infty < y < \infty$

$\eta = 10$, $\phi = 1$ e $-\infty < y < \infty$

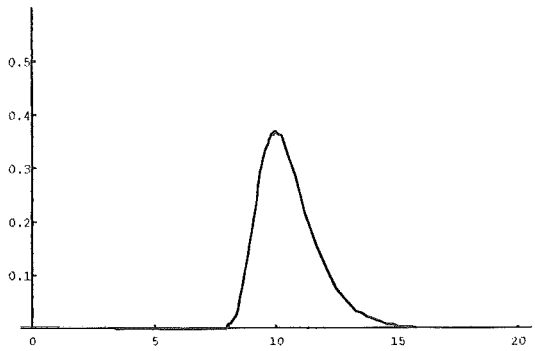


Figura 4.1 - Gráfico da densidade de probabilidade de Gumbel

Reparametrizando a distribuição de Gumbel pela média μ, teremos:

$$P(Y_t|\mu_t,\emptyset_t) = \frac{1}{\emptyset_t}.\exp\left[-\exp\left(-\frac{(y_t+C.\emptyset_t-\mu_t)}{\emptyset_t}\right)\right].\exp\left[-\frac{(y_t+C.\emptyset_t-\mu_t)}{\emptyset_t}\right] \quad (4.3.4)$$

$$E(Y_t|\mu_t,\emptyset_t) = \mu_t \quad (4.3.5)$$

$$V(Y_t|\mu_t,\emptyset_t) = \frac{\pi^2}{6}.\emptyset_t^2 \quad (4.3.6)$$

No desenvolvimento subsequente não utilizaremos a distribuição conjunta de μ_t e $\emptyset_t$. O parâmetro μ_t receberá tratamento Bayesiano. Quanto a $\emptyset_t$, será estimado recursivamente, a partir de hipóteses adequadas sobre a variância das observações. No caso da distribuição normal, e mais geralmente para a família exponencial, uma análise Bayesiana completa pode ser encontrada em Degroot (1987) e West Harrison e Migon (1985), respectivamente.

Priori não informativa

A distribuição a priori não informativa de Jeffreys para μ é calculada de modo análogo ao da seção 3.2, o que resulta:

$$P(\mu_t|D_{t-1}) \propto \frac{1}{\emptyset_t} \quad (4.3.7)$$

A distribuição preditiva um passo à frente feita a partir dessa priori é também uma distribuição imprópria, como é visto a seguir:

$$P(Y_t|D_{t-1}) = \int P(Y_t|\mu_t,\emptyset_t).P(\mu_t|D_{t-1})d\mu_t$$

$$P(Y_t|D_{t-1}) \propto \int_{-\infty}^{\infty} \frac{1}{\emptyset_t}.\frac{1}{\emptyset_t} .\exp\left[-\exp\left(-\frac{(y_t+C.\emptyset_t-\mu_t)}{\emptyset_t}\right)\right].\exp\left[-\frac{(y_t+C.\emptyset_t-\mu_t)}{\emptyset_t}\right].d\mu_t$$

$$P(Y_t|D_{t-1}) \propto \frac{1}{\emptyset_t} \qquad (4.3.8)$$

Distribuição a posteriori

Pelo teorema de Bayes, a distribuição a posteriori de μ_t é dado por:

$$P(\mu_t|D_t) \propto P(Y_t|\mu_t,\emptyset_t).P(\mu_t|D_{t-1})$$

Substituindo as expressões da distribuição a priori e a verossimilhança, temos:

$$P(\mu_t|D_t) \propto \frac{1}{\emptyset_t} .\exp\left[-\exp\left(-\frac{(y_t+C.\emptyset_t-\mu_t)}{\emptyset_t}\right)\right].\exp\left[-\frac{(y_t+C.\emptyset_t-\mu_t)}{\emptyset_t}\right].\frac{1}{\emptyset_t}$$

Após calcular a constante de normalização, a distribuição a posteriori é dada por

$$P(\mu_t|D_t) = \frac{1}{\emptyset_t} .\exp\left[-\exp\left(-\frac{(y_t+C.\emptyset_t-\mu_t)}{\emptyset_t}\right)\right].\exp\left[-\frac{(y_t+C.\emptyset_t-\mu_t)}{\emptyset_t}\right] \qquad (4.3.9)$$

A figura 4.2 representa o gráfico da distribuição a posteriori de μ_t.

$$y_t = 10, \emptyset = 1 \text{ e, } -\infty < \mu_t < \infty$$

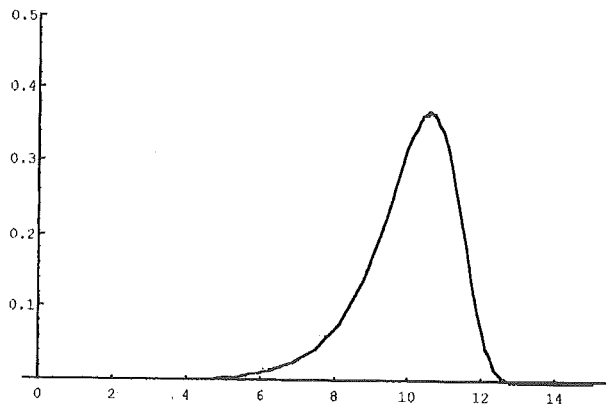


Figura 4.2 - Gráfico da função de densidade de probabilidade a posteriori para μ_t

Calculo da média e a variância desta distribuição a posteriori.

Média da distribuição a posteriori de μ_t

Através do uso da transformação

$$\exp\left(\frac{\mu_t}{\emptyset_t}\right) = z_t \qquad (4.3.10)$$

e de integrais encontradas em Gradshteyn e Ryzhik (1965), pág. 573, podemos então escrever o valor esperado da posteriori de μ_t como [Ferreira (1991)]:

$$E(\mu_t|D_t) = y_t \qquad (4.3.11)$$

Variância da posteriori de μ_t.

$$V(\mu_t|D_t) = E(\mu_t^2|D_t) - [E(\mu_t|D_t)]^2$$

Utilizando-se a transformação (4.3.10) e integrais de Gradshteyn e Ryzhik (1965), pág. 578, segue-se que:

$$E[\mu_t^2|D_t) = y_t^2 + \emptyset_t^2.\frac{\pi^2}{6}$$

Desse resultado e de (4.3.11) segue

$$V[\mu_t|D_t] = \emptyset_t^2.\frac{\pi^2}{6} \qquad (4.3.12)$$

Com estes resultados, passaremos a descrever o modelo. Por comodidade, resumimos alguns resultados anteriores.

Modelo observacional

- A distribuição das observações é uma Gumbel, conforme (4.3.4):

$$P(Y_t|\mu_t,\emptyset_t) = \frac{1}{\emptyset_t}.\exp\left[-\exp\left(-\frac{(y_t+C.\emptyset_t-\mu_t)}{\emptyset_t}\right)\right].\exp\left[-\frac{(y_t+C.\emptyset_t-\mu_t)}{\emptyset_t}\right]$$

$$E(Y_t|\mu_t,\emptyset_t) = \mu_t \qquad -\infty < Y_t < \infty$$

$$V(Y_t|\mu_t,\emptyset_t) = \frac{\pi^2}{6}.\emptyset_t^2 \qquad -\infty < \mu_t < \infty$$

$$\emptyset_t > 0$$

- Priori não informativa do parâmetro μ_t, dado por (4.3.7):

$$P(\mu_t|D_{t-1}) \propto \frac{1}{\emptyset_t}$$

- Posteriori do parâmetro μ_t, expressões (4.3.9), (4.3.11) e (4.3.12):

$$P(\mu_t|D_t) = \frac{1}{\emptyset_t}.\exp\left[-\exp\left(-\frac{(y_t+C.\emptyset_t-\mu_t)}{\emptyset_t}\right)\right].\exp\left[-\frac{(y_t+C.\emptyset_t-\mu_t)}{\emptyset_t}\right]$$

$$E(\mu_t|D_t) = y_t$$

$$V(\mu_t|D_t) = \emptyset_t^2.\frac{\pi^2}{6}$$

Evolução temporal do modelo

Como na seção 4.2, o modelo observacional e a equação de estado são dadas respectivamente por:

$$P(Y_t|\mu_t);\ g(\mu_t) \equiv \lambda_t = \underline{F}_t.\underline{\theta}_t \qquad (4.3.13)$$

$$\underline{\theta}_t = \underline{G}_t\,(\underline{\theta}_{t-1}) + \underline{w}_t,\ \underline{w}_t \sim (0,\underline{W}_t) \qquad (4.3.14)$$

- vetor de estado: $\underline{\theta}_t = [u,\beta,\gamma_1,\gamma_2,\ldots,\gamma_s,\Psi,\Psi^*]_t$ (4.3.15)

Posteriori do vetor de estado no instante anterior, conhecida:

$$(\underline{\theta}_{t-1}|D_{t-1}) \sim [\underline{m}_{t-1},\underline{C}_{t-1}] \qquad (4.3.16)$$

Priori do vetor de estado no instante t: $(\underline{\theta}_{t-1}|D_{t-1}) \sim [\underline{a}_t,\underline{R}_t]$ (4.3.17)

onde:

$$\underline{a}_t = \underline{G}_t\,(\underline{m}_{t-1}) \qquad (4.3.18)$$

$$\underline{R}_t = \underline{\Delta}_t.\underline{H}_t.\underline{C}_{t-1}.\underline{H}_t^T.\underline{\Delta}_t \qquad (4.3.19)$$

$$\underline{S}_t = \underline{R}_t.\underline{F}_t = \underline{R}_t.\underline{F} \qquad (4.3.20)$$

e $\underline{\Delta}$ é uma matriz de fatores de desconto.

No nosso caso, o modelo é linear, com

$$\underline{G} = \begin{vmatrix} 1 & 1 & 0 & \dots & \dots & \dots & 0 & 0 & 0 \\ 0 & 1 & 0 & \dots & \dots & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & \dots & \dots & \dots & 0 & 0 & 0 \\ \cdot & \cdot & \cdot & & & & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & & & & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & & & & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & & & & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & & I_{S-1} & & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & & & & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & & & & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & & & & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & & & & \cdot & \cdot & \cdot \\ 0 & 0 & 0 & \dots & \dots & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & \dots & \dots & \dots & 0 & \rho\cos(\lambda) & \rho\text{sen}(\lambda) \\ 0 & 0 & 0 & \dots & \dots & \dots & 0 & -\rho\text{sen}(\lambda) & \rho\cos(\lambda) \end{vmatrix}$$

e $\underline{F}_t^T = \underline{F}^T = [1,0,1,0,\dots,0,1,0]$

Do modelo observacional definimos:

$$\lambda_t = \underline{F}^T.\underline{\theta}_t, \text{ com } \lambda_t|D_{t-1} \sim [f_t,q_t] \qquad (4.3.21)$$

$$E[\lambda_t|D_{t-1}] = \underline{F}^T.\underline{a}_t = f_t$$

$$V[\lambda_t|D_{t-1}] = \underline{F}^T.\underline{R}_t.\underline{F} = q_t$$

- Atualização do vetor de estado:

A atualização se processa como na seção 4.2: $(\underline{\theta}_t|D_t) \sim [\underline{m}_t,\underline{C}_t]$

onde

$$\underline{m}_t = \underline{a}_t + \underline{S}_t.\left[\frac{E(\lambda_t|D_t)-f_t}{q_t}\right] = \underline{a}_t + \underline{S}_t.\frac{(y_t-f_t)}{q_t} \qquad (4.3.22)$$

$$\underline{C}_t = \underline{R}_t - \underline{S}_t.\underline{S}_t^T.\left[1 - \frac{V(\lambda_t|D_t)}{q_t}\right].q_t^{-1} \qquad (4.3.23)$$

Essa atualização é fundamentada no método "linear Bayes". No caso a relação guia é $\lambda_t \cong \mu_t$, pois o modelo observacional está parametrizado pela média. Então $E(\lambda_t|D_t)$ pode ser estimada por $E(\mu_t|D_t) = y_t$. Quanto a $V(\lambda_t|D_t)$ na expressão (4.3.23), mostraremos a seguir que pode ser estimado por $V(\mu_t|D_t)$.

- Estimativa de $\emptyset_t$:

O parâmetro $\emptyset_t$ é estimado a partir de $q_t = V(\lambda_t|D_{t-1})$, da seguinte maneira:

$$\frac{\pi^2}{3}\emptyset_t^2 = V(Y_t|D_{t-1}) = V(\lambda_t|D_{t-1}) = q_t$$

A primeira igualdade acima é obtida tomando a variância na distribuição preditiva do passo anterior. (o cálculo da distribuição preditiva se encontra no item seguinte). Assim, escolhemos como $\emptyset_t$ o valor que iguala a variância q_t (dada pelo preditor linear λ) à variância de $Y_t|D_{t-1}$, interpretada como a preditiva do passo anterior.

Uma consequência desse modelo para Ø é que $V(\lambda_t|D_t) = V(\mu_t|D_t)$. De fato:

$$V(\lambda_t|D_t) = V(\lambda_t|\mu_t,Ø) = (\pi^2/6).Ø_t^2,$$

a partir do modelo observacional. Por outro lado,

$$V(\mu_t|D_t) = (\pi^2/6).Ø_t^2,$$

a partir da distribuição a posteriori.

Voltando à equação (4.3.23) temos a expressão da variância do vetor de estado $\underline{\theta}_t$, ou seja,

$$V(\underline{\theta}_t|D_t) = \underline{C}_t.$$

$$\underline{C}_t = \underline{R}_t - \underline{S}_t.\underline{S}_t^T.\left[1 - \frac{V(\lambda_t|D_t)}{q_t}\right].q_t^{-1} = \underline{R}_t - \underline{S}_t.\underline{S}_t^T.\left[1 - \frac{(\pi^2/6).Ø_t^2}{(\pi^2/3).Ø_t^2}\right].q_t^1$$

$$\underline{C}_t = \underline{R}_t - \underline{S}_t.\underline{S}_t^T.\left[1 - \frac{1}{2q_t}\right]$$

É interessante observar que a relação $[V(\lambda_t|D_t)/V(\lambda_t|D_{t-1})]$ é constante e igual a 1/2 no nosso caso. As expressões completas de atualização do vetor de estado $\underline{\theta}_t$ são:

$$\underline{m}_t = \underline{a}_t + \underline{S}_t.\frac{(y_t - f_t)}{q_t}$$

$$\underline{C}_t = \underline{R}_t - \underline{S}_t.\underline{S}_t^T.\left[1 - \frac{1}{2q_t}\right]$$

Distribuição Preditiva

Os itens já descritos permitem definir completamente o modelo $P(Y_t|\mu_t,Ø_t)$. A partir deste modelo descreve-se na seção 5, como se obter uma estimativa para H(50). No entanto, para efeito de previsão a curto prazo é útil conhecer a distribuição preditiva um passo à frente $(Y_{t+1}|D_t)$. Por definição

$$P(Y_{t+1}|D_t) = \int P(Y_{t+1},\mu_t,D_t).P(\mu_t|D_t)d\mu_t$$

Ferreira (1991, págs.82-87) desenvolve o cálculo da distribuição preditiva $P(Y_{t+1}|D_t)$, bem como de seus dois primeiros momentos:

$$P(Y_{t+1}|D_t) = \frac{\exp\left(-\frac{y_{t+1}}{Ø_t}\right).\exp\left(-\frac{y_t}{Ø_t}\right)}{Ø_t.\left[\exp\left(-\frac{y_{t+1}}{Ø_t}\right)+\exp\left(-\frac{y_t}{Ø_t}\right)\right]^2}$$

$$E[Y_{t+1}|D_t] = y_t \qquad (4.3.25)$$

$$V[Y_{t+1}|D_t] = \frac{\pi^2}{3}.Ø_t^2 \qquad (4.3.26)$$

A figura 4.3 representa o gráfico da distribuição preditiva.

$$y_t = 10, \emptyset = 1 \text{ e, } -\infty < y_t < \infty$$

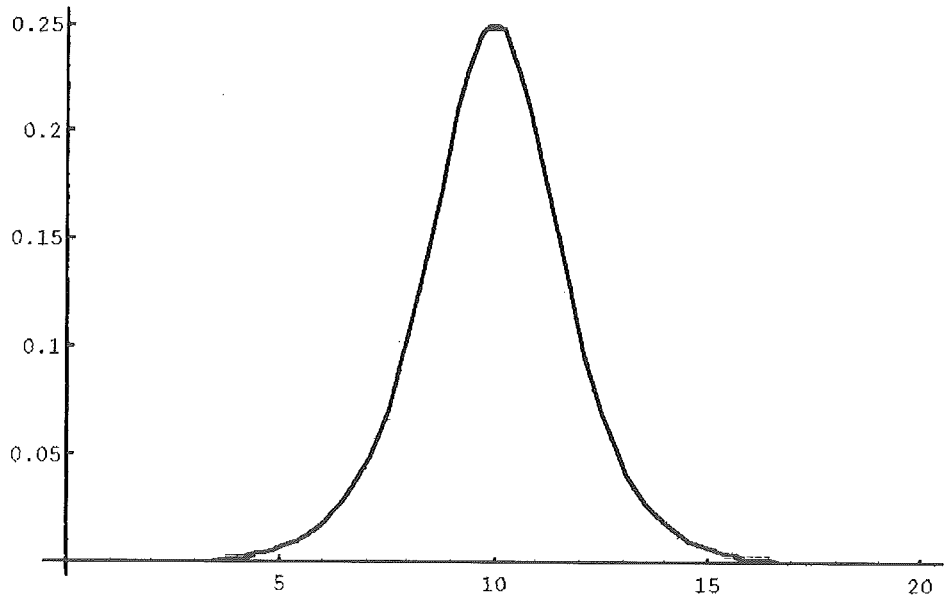


Figura 4.3 - Gráfico da função densidade preditiva

Observação: a densidade da preditiva obtida nessa item pode ser escrita como:

$$P(Y_t|D_{t-1}) = \int P(Y_t,\mu_{t-1}).P(\mu_{t-1}|D_{t-1})d\mu_{t-1}$$

$$P(Y_t|D_{t-1}) = \frac{\exp\left(-\frac{y_t}{\emptyset_t}\right)\exp\left(-\frac{y_{t-1}}{\emptyset_t}\right)}{\emptyset_t.\left[\exp\left(-\frac{y_t}{\emptyset_t}\right)+\exp\left(-\frac{y_{t-1}}{\emptyset_t}\right)\right]^2}$$

No desenvolvimento da distribuição preditiva estamos supondo que $\emptyset_t \cong \emptyset_{t-1}$, o que significa assumir uma variação suave para $\emptyset$. Essa densidade substitui, para todos os propósitos práticos, a densidade imprópria (4.3.8), pois representa, de fato, toda a informação necessária para prever Y_t a partir das observações anteriores, eliminada a incerteza em μ_{t-1}. Em particular, temos $V(Y_t|D_{t-1}) = \pi^2\emptyset_t^2/3$, igualdade que foi usada como base para a estimativa de $\emptyset_t$.

5. Aplicações a algumas Séries e Conclusões

Nesta seção estamos interessados em verificar o comportamento do modelo desenvolvido na seção 4. Na seção 5.1, aplicamos o modelo dinâmico Bayesiano para extremos a duas séries reais, a saber, a série de vendas de um determinado produto farmacêutico e a série de alturas de ondas no Porto de Praia Mole, Vitória, Espírito Santo, Brazil. Na seção 5.2, determinamos o valor que é ultrapassado em média uma vez em cada 50 anos, H(50), para a série de alturas de ondas de Praia Mole, e concluímos com uma comparação dos métodos utilizados em todo o trabalho para a estimativa de H(50).

5.1 Aplicação a séries reais

5.1.1 Série de vendas de um protocolo farmacêutico

Os dados desta série consistem das vendas máximas mensais, nos EUA, de um produto farmacêutico, no período de maio de 1987 a abril de 1989. As tabelas 5.1, 5.2 e a Figura 5.1, fornecem, respectivamente, a série de máximos, suas características quantitativas e o gráfico.

Mês	1987	1988	1989
Janeiro	—	302.284	294.782
Fevereiro	—	217.268	206.826
Março	—	224.098	209.536
Abril	—	214.995	197.954
Maio	147.469	197.795	—
Junho	206.624	198.193	
Julho	215.144	211.057	
Agosto	200.948	196.724	
Setembro	213.642	223.869	
Outubro	219.242	221.578	
Novembro	235.772	228.576	
Dezembro	259.238	274.158	

Tabela 5.1 - Máximos mensais de vendas de um produto farmacêutico (EUA)-Mai/87-Abr/89

Estatística	Valor
Média	221.575,92
Mediana	215.069,50
Valor mínimo	147.469,00
Valor máximo	302.284,00
Desvio padrão	33.225,97

Tabela 5.2 - Algumas características da série de vendas

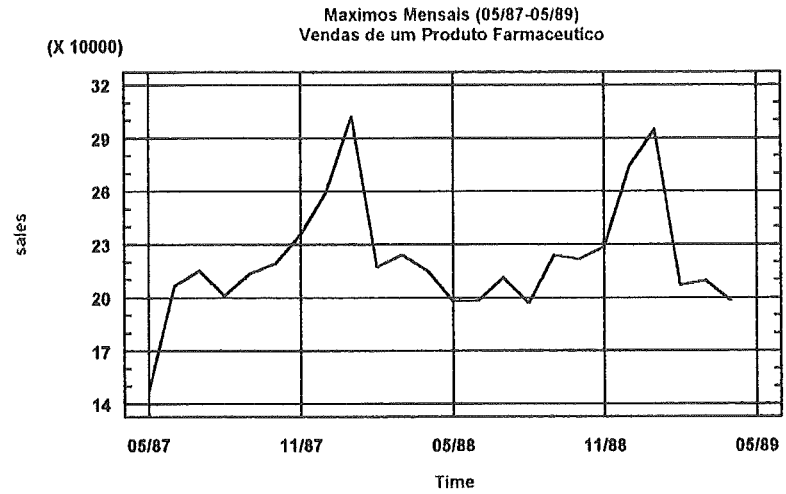


Figura 5.1 - Série de vendas de um produto farmacêutico

Apesar da pequena quantidade de dados disponível, podemos observar que o padrão sazonal dessa série é bem definido. Adotamos o modelo com tendência e sazonalidade, i.e., o vetor de estado $\underline{\theta}_t$ definido como: $\underline{\theta}_t = .[\mu_t, \beta_t, \gamma_{1t}, \gamma_{2t}..., \gamma_{12t}]$.

Os valores da priori inicial estabelecida para o modelo dinâmico Bayesiano foram:

$\underline{m}_0$ = [180;0;-15,03;-39,25;-28.15;-11,81;-24,51;0.08;-1,12;8,86;5,68;75,35;-10,01;-5,08]

$\underline{C}_0$ = Diag[625;9;100;100;...;100]

A tabela 5.3 mostra o erro quadrático médio (EQM) para diversos valores dos fatores de desconto da tendência e da sazonalidade. Considerando o menor MSE relativo [RMSE = $\frac{1}{m}\sum_{t=1}^{m}\left({}^{e_t}/_{Y_t}\right)^2$], apresentado na tabela, o melhor resultado é para o fator de desconto da tendência $\delta_1 = \delta_{TEND} = 0.60$ e o fator de desconto da sazonalidade $\delta_2 = d_{SAZ} = 0.99$. Podemos observar pela figura 5.2 que o ajustamento do Modelo Dinâmico Bayesiano de Extremos para esses dados foi muito bom.

Note-se que o desempenho do Modelo Dinâmico Bayesiano, que supõe normalidade fica aquém do modelo para extremos: O menor RMSE foi $8{,}399E^{-3}$. Ver a figura 5.3 gerada pelo software BATS (Pole, West & Harrison, 1994) para fatores de desconto $\delta_1 = 0.60$ e $\delta_2 = 0.99$.

F.D.SAZ / F.D.TEND	0.90	0.94	0.97	0.99
0.60	$1.5471E^{-3}$	$1.5180E^{-3}$	$1.4974E^{-3}$	$1.4843E^{-3}$
0.70	$1.6426E^{-3}$	$1.6178E^{-3}$	$1.5989E^{-3}$	$1.5861E^{-3}$
0.80	$1.7319E^{-3}$	$1.6968E^{-3}$	$1.6761E^{-3}$	$1.6643E^{-3}$
0.90	$1.9100E^{-3}$	$1.8244E^{-3}$	$1.7788E^{-3}$	$1.7546E^{-3}$
0.95	$2.0680E^{-3}$	$1.9358E^{-3}$	$1.8653E^{-3}$	$1.8282E^{-3}$
0.99	$2.2427E^{-3}$	$2.0600E^{-3}$	$1.9618E^{-3}$	$1.9100E^{-3}$

Tabela 5.3 - Erro quadrático médio relativo X fatores de desconto: modelo dinâmico Bayesiano para extremos

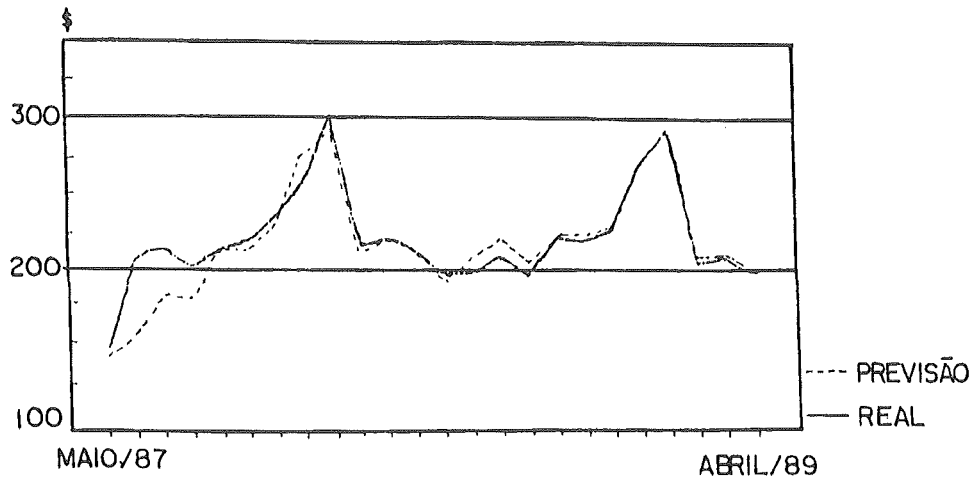


Figura 5.2 - Máximos mensais de vendas de um produto farmacêutico - previsão um passo à frente: modelo dinâmico de extremos ($\delta_1 = \delta_{TEND} = 0.60$ e $\delta_2 = \delta_{SAZ} = 0.99$)

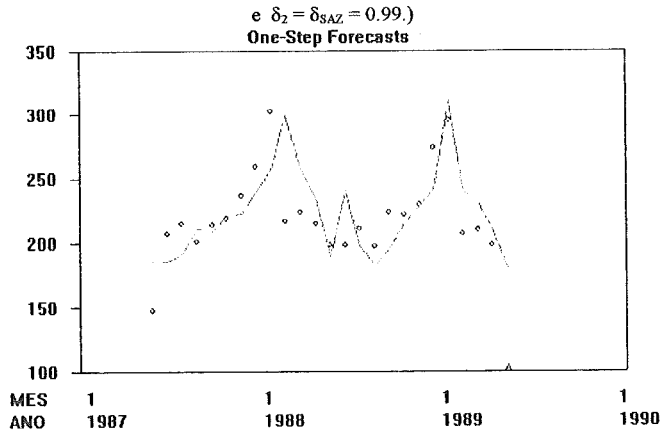


Figura 5.3 - Máximos mensais de vendas de um produto farmacêutico - previsão um passo à frente: modelo dinâmico (BATS) ($\delta_1 = \delta_{TEND} = 0.60$ e $\delta_2 = \delta_{SAZ} = 0.99$)

5.1.2 Séries de ondas do Porto de Praia Mole (Vitória-ES)

A série de alturas de ondas no Porto de Praia Mole, já foi estudada em seções anteriores. As tabelas 5.4, 5.5 e figura 5.5 fornecem, respectivamente, a série de altura máxima mensais, as principais características quantitativas e o gráfico desta série.

Mês	1982	1983	1984	1985	1986
Janeiro	—	2.95	2.60	3.40	2.45
Fevereiro	—	2.25	2.15	2.80	2.04
Março	—	2.80	3.19	2.50	2.80
Abril	4.25	3.45	2.69	3.30	—
Maio	3.90	3.50	2.59	3.45	
Junho	2.59	2.59	3.19	2.80	
Julho	3.50	3.40	3.90	4.69	
Agosto	3.90	3.90	4.00	4.00	
Setembro	4.30	6.05	4.89	3.80	
Outubro	3.06	4.39	3.40	3.40	
Novembro	2.80	3.85	3.09	3.40	
Dezembro	2.69	4.39	3.09	4.25	

Tabela 5.4 - Alturas máximas mensais de ondas (Ab/82-Mar/86)

Fonte: Instituto de Pesquisa Hidroviárias (INPH), Portobrás, Brasil

Estatística	Valor
Média	3,38
Mediana	3,40
Valor mínimo	2,04
Valor máximo	6,05
Desvio padrão	0,79

Tabela 5.5 - Algumas características da série de ondas

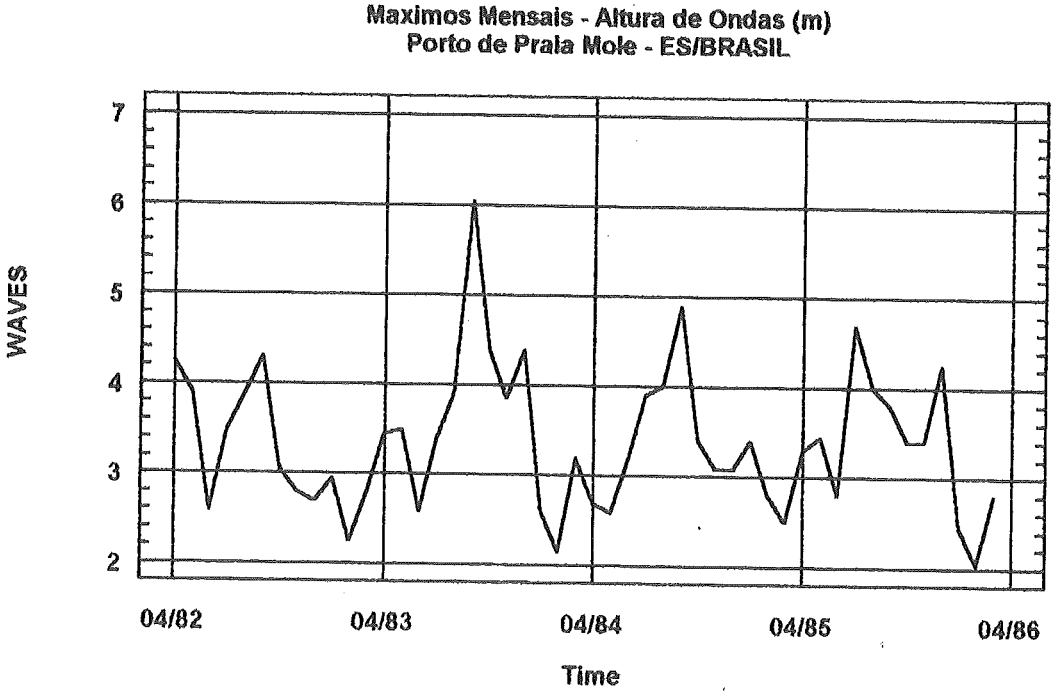


Figura 5.5 - Gráfico da série de ondas - Porto de Praia Mole (Vitória/ES)

Adotou-se um modelo com tendência e sazonalidade, como na seção anterior. Os valores da priori inicial estabelecido no Modelo Dinâmico Bayesiano de Extremos para esta série são:

$\underline{m}_0$ = [3.3;0.0;0.0;...;0.0]

$\underline{C}_0$ = Diag[0.6;0.1;0.2;-0.2/11;...;-0.2/11]

A Tabela 5.3 mostra o erro quadrático médio alguns valores dos fatores de desconto da tendência e da sazonalidade. Considerando o menor erro quadrático apresentado nesta tabela, o melhor resultado é para o fator de desconto da tendência (δ_{TEND}) e sazonalidade (δ_{SAZ}), respectivamente, $\delta_1 = 0.80$ e $\delta_2 = 0.99$. É importante ressaltar que, neste caso, o ajuste do Modelo Linear Dinâmico, que supõe normalidade, foi ligeiramente melhor: MSE = 0,0532.

Apresentamos na figura 5.6 o gráfico do ajustamento do modelo com os fatores de descontos $\delta_1 = 0.80$ e $\delta_2 = 0.99$ e, na figura 5.7 outro gráfico com fatores de descontos $\delta_1 = 0.95$ e $\delta_2 = 0.97$, respectivamente, para a tendência e sazonalidade. Nas figuras 5.8 e 5.9 temos os gráficos das variações dos parâmetros μ_t e $\emptyset_t$, com fatores de descontos de 0.95 e 0.97, para a tendência e a sazonalidade, respectivamente.

F.D.SAZ (δ_2) / F.D.TEND (δ_1)	0.94	0.97	0.99
0.70	—	0.06471	0.06590
0.80	0.06397	0.06336	0.06296
0.90	—	0.06486	0.06466
0.95	0.0685	0.06664	0.06557

Tabela 5.3 - Erro quadrático médio X fatores de descontos

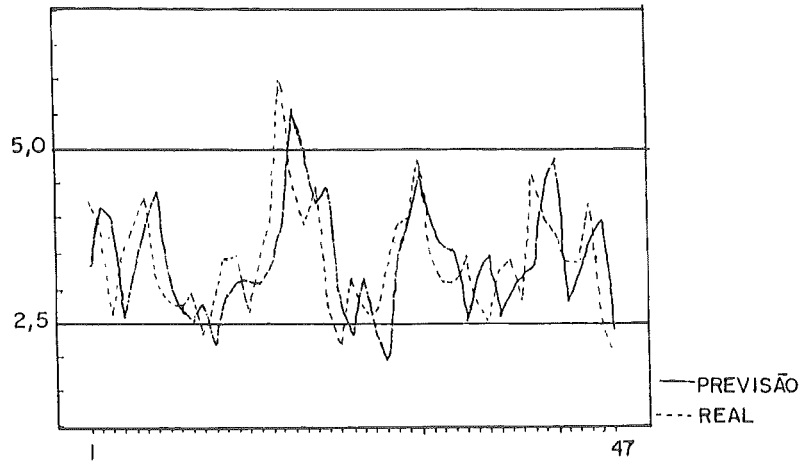


Figura 5.6 - Gráfico de alturas de ondas ($\delta_{TEND} = \delta_1 = 0.80$ e $\delta_{SAZ} = \delta_2 = 0.99$)

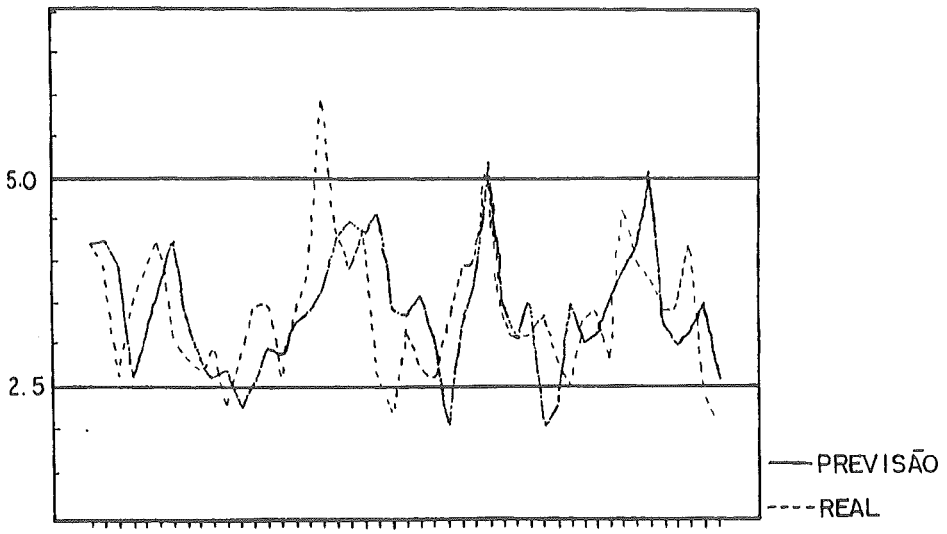


Figura 5.7 - Gráfico de alturas de ondas ($\delta_{TEND} = \delta_1 = 0.95$ e $\delta_{SAZ} = \delta_2 = 0.97$)

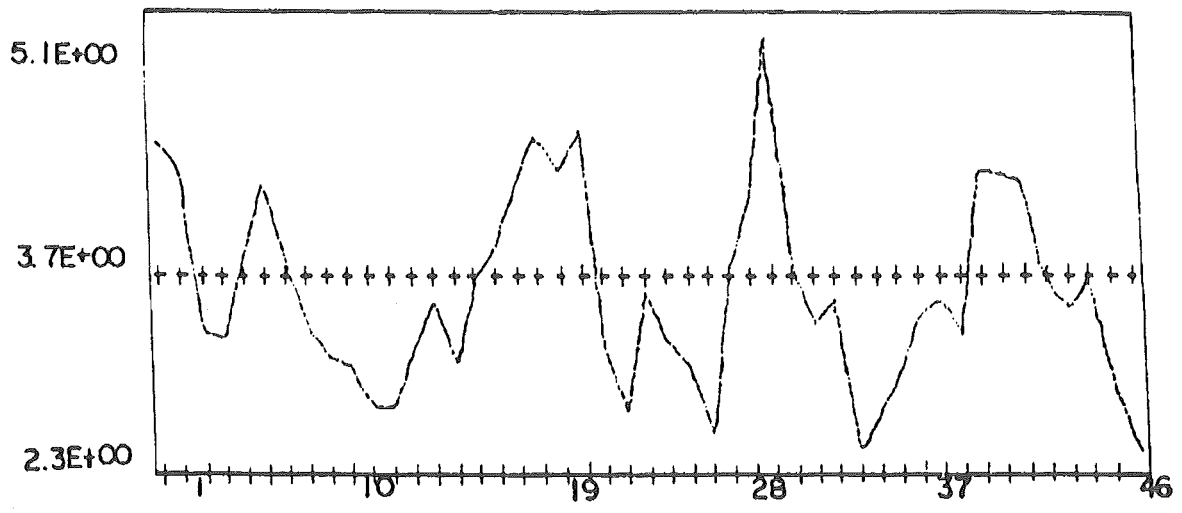


Figura 5.8 - Variação da média μ_t ($\delta_{TEND} = \delta_1 = 0.95$ e $\delta_{SAZ} = \delta_2 = 0.97$)

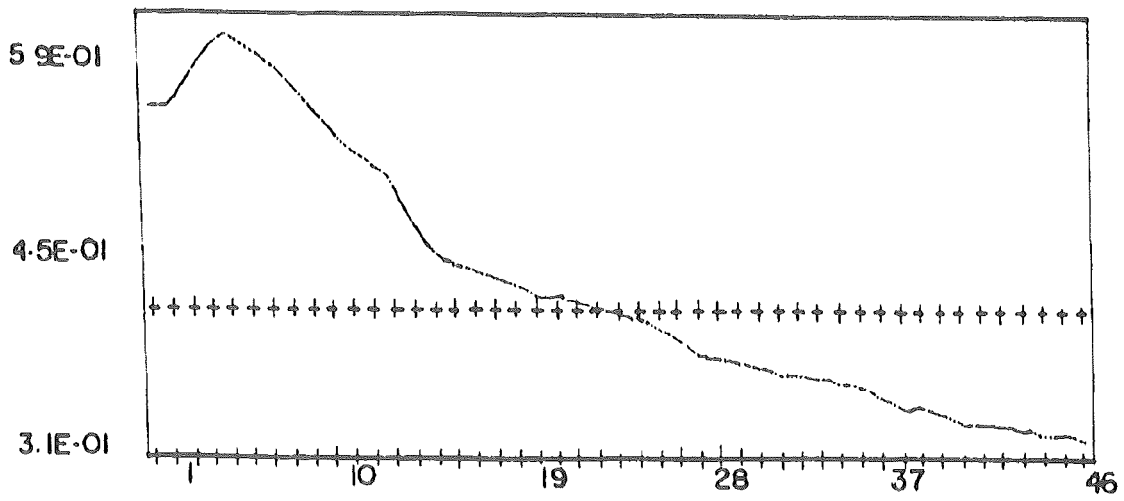


Figura 5.9 - Variação do parâmetro $\emptyset_t$ ($\delta_{TEND} = \delta_1 = 0.95$ e $\delta_{SAZ} = \delta_2 = 0.97$)

5.2 Cálculo de H(50) e conclusões

Na seção 2, apresentamos dois métodos clássicos para tratar problemas de extremos: o método clássico de Gumbel e o Peaks Over Threshold (P.O.T.). Turner (1982) e Smith (1984) estudaram estes métodos, com o objetivo de estimar a altura de onda que é ultrapassada em média uma vez em 50 anos, denotada por H(50).

No método de Gumbel, ajustam-se várias distribuições de Gumbel $F_i(y)$ e, a partir delas, estimamos vários $H_i(Q)$, definindo-se H(Q) como o máximo dos $H_i(Q)$. Este método evita a necessidade de estudar previamente a sazonalidade da série, porém ele não detecta uma possível tendência, nem uma possível alteração dos padrões sazonais.

No método de Peaks Over Threshold (P.O.T.), em geral, é necessário transformar previamente os dados, pois ele é idealizado para séries estacionárias. A seguir definimos um patamar. Os valores acima do patamar aparecem em grupos de instantes consecutivos. O valor máximo em um dado grupo é denominado o pico desse grupo. Consideram-se somente esses picos, as outras observações são descartadas. Os tempos correspondentes aos picos formam, sob certas condições, um processo de Poisson, veja Turner (1982) e Smith (1984). Segue-se então que as ocorrências de valores acima do patamar (valores excedentes) em intervalos disjuntos são eventos independentes e o número desses valores num dado intervalo tem uma distribuição de Poisson. Esses fatos contribuem para a estimativa de H(50).

A diferença entre o pico e o patamar é modelada por uma distribuição exponencial. Esta escolha é devida à propriedade de "falta de memória", ou seja $P(Y > y + c|Y > c) = P(Y > y)$. Essa propriedade torna nossas conclusões, em certa medida, independente do patamar escolhido. Outra justificativa para a escolha da distribuição exponencial pode ser encontrada em Smith (1984). A partir destas hipóteses é feita então a estimativa de H(50).

Como vimos na seção 2, não existe critério para escolha da altura do patamar e tamanho dos grupos. O tamanho de cada grupo está ligado à duração da condição de extremo, a qual pode

ser tão importante num dado problema quanto o próprio valor do extremo (por exemplo, vento sobre uma estrutura). Em determinadas situações é necessário modelar todos os valores excedentes, e não apenas os picos. Tentativas feitas por Smith (1984) para definir o tamanho dos grupos contribuíram para resolver esse problema, à custa de tornar este método difícil de ser usado. No modelo dinâmico essa questão pode ser respondida com simplicidade, diminuindo o período de referência para a tomada do máximo.

A necessidade de estudar previamente as componentes de tendência e sazonalidade constitui também uma limitação do método P.O.T. Turner (1982) e Smith (1984) mencionam como alternativa a utilização de um processo de Poisson não homogêneo. No modelo dinâmico aquelas componentes são consideradas no próprio corpo do modelo, os dados são tratados tal como eles se encontram. As transformações de dados usados, por exemplo, por Turner (1982), dependem em grande parte da experiência do pesquisador. A decisão de logaritmar os dados, bem como a decisão de limitar as frequências na componente sazonal são exemplos desse fato. Em contraste, no modelo dinâmico Bayesiano, s subjetividade é incorporada naturalmente nas distribuições a priori dos parâmetros.

Até certo ponto, o modelo dinâmico Bayesiano com observações provenientes da distribuição de Gumbel, é um retorno ao método clássico de Gumbel, embora não seja suposto que os padrões sazonais estejam fixos. Nossa proposta é usar os valores de η e $\emptyset$ pelo modelo dinâmico como base para a estimativa de H(50), dispensando a estrutura dada pela matriz (5.1). Esses valores apreendem o comportamento da série, na medida em que reúnem informações acumuladas (as distribuições a posterior e preditiva sintetizam toda a informação no tempo t) podendo captar uma tendência e possíveis modificações nos padrões sazonais. Portanto, para cada um desses pares $(\eta_t,\emptyset_t)$ estimamos $H_t(50)$ e definimos H(50) como o máximo dos $H_t(50)$. Como na seção 2.2, temos:

$$F_t(y_t) = P[Y \leq y_t]\ 1 - 1/N = 1 - 1/50 = 0{,}98$$

onde $Y_t = H_t(50)$. Logo,

$$F_t(y_t) = \exp[-\exp-((y_t - \eta_t)/\emptyset_t)] = 0.98$$

Daí segue-se que

$$H_t(50) = Y_t = \eta_t - \emptyset_t.\ln[-\ln(0{,}98)] = \eta_t + 3.90.\emptyset_t.$$

No nosso caso, optamos por considerar $H(50) = \text{Max}\{H_t(50)\}$ no último ano disponível da série, embora, em geral possa ser usado qualquer período englobando os últimos anos da série. O valor de H(50) calculado na versão correspondente ao gráfico apresentado na figura 5.7 é H(50) = 6.48 metros.

É claro que, após o cálculo anual de $H_t(50)$, há a possibilidade de modelagem no tempo desse novo parâmetro, na procura de ciclos, por exemplo; no entanto, entendemos que o máximo acumulado fornece um parâmetro estático apropriado.

Alguns comentários sonre os dados e os modelos. Os dados da série de ondas apresentam um ajustamento melhor para o Modelo Linear Dinâmico (distribuição normal) relativamente ao

Modelo Dinâmico para Extremos (distribuição de Gumbel). Na verdade, os dados da série de ondas não rejeitam a hipótese de normalidade (teste Kolmogorov-Smirnov e qui-quadrado). O mesmo não se pode dizer dos dados da série de vendas, que é extremamente pequena (24 observações): o Modelo Dinâmico Bayesiano que adota diretamente uma distribuição de Gumbel apresenta um melhor ajuste.

Quanto ao valor de H(50) determinado pelas metodologias descritas, o valor mais alto foi obtido pelo método de Gumbel clássico [$H_a(50) = 7.14$m e $H_b(50) = 8.05$m] enquanto que a estimativa Bayesiana (estática) foi de 7.23m (seção 3.2). Observe-se que o estimador Bayesiano, neste caso, considera toda a amostra disponível.

O Modelo Dinâmico Bayesiano é adaptativos e atribui um peso maior aos dados mais recentes. O valor de H(50) obtido via distribuição preditiva em t = Abril/86 foi de 6.48m, um valor pouco conservador (note que o máximo histórico foi de 6.05m); mas isso é devido ao fato de adotarmos um fator de descontos para a tendência muito baixa ($\delta_T = 0{,}60$), um valor pouco recomendado, resultado do pequeno tamanho da série.

Finalmente, podemos observar que há muitas variações do modelo proposto, dependendo das possíveis informações a priori sobre η e $\emptyset$, que irão se refletir nas distribuições a priori; (por exemplo, um modelo com a chamada priori de referência). A adoção de uma determinada distribuição serve a vários propósitos, possibilitando intervenções no modelo, regularizando um problema numérico, ou até simplificando expressões analíticas no modelo.

Um exemplo particularmente interessante que pode ser obtido dentro das mesmas linhas deste trabalho é o caso do modelo observacional Gumbel com priori exponencial. Outra questão a ser desenvolvida é o tratamento Bayesiano para estimar simultaneamente os parâmetros η e $\emptyset$.

Os autores gostariam de agradecer os comentários e sugestões dos pareceristas da revista.

Referências

[1] Ameen, J.R.M. and Harriosn, P.J., Discount Weighted Estimation, Journal of Forecasting 3 (1984) 285-296.

[2] Ammen, J.R.M. and Harrison, P.J., Normal Discount Bayesian Models, in Bayesian Statistics 2 (1985) 271-298, J.M. Bernardo et al eds., North-Holland.

[3] Bernardo, J.M., Refrence Posterior Distributions for Bayesian Inferences, J.R. Statist. 41 (1979) 113-147.

[4] Box, G.E.P. and Tiao, G.C., Bayesian Inference in Statistical Analysis, Addison-Wesley, Reading, Massachusetts (1973).

[5] Brasil, G.H., Modelagem Dinâmica Bayesiana da Componente Cíclica na Formulação Estrutural de Séries Temporais, Tese de doutorado, Departamento de Engenharia Elétrica (Sistemas), Pontifícia Universidade Católica do Rio de Janeiro (1989).

[6] Carter, D.J.T. and Challenor, P.G., Return Wave heights at Seven Stones and Famita estimated from monthly maximum, Institute of Oceanographic Sciences Report 66 (1978).

[7] Challenor, P.G., Model for the variation of return values throughout the year, Institute of Oceanographic Sciences WACAS32 (1981).

[8] Cox, D.R., Some statistical methods connected with series of events, J.Roy. Statist. Soc. 27 (1955) 129-164.

[9] Degroot, M.H., Probability and Statistics, end ed., Addison-Wesley, Massachusetts (1987).

[10] Ferreira, M.J.S. and Souza, R.C., Estimativa a altura de onda que retorna em 50 anos no Porto de Praia Mole, trabalho apresentado na SBMAC (1988).

[11] Ferreira, M.J.S., Souza, R.C. and Brasil, G.H., Inferência Bayesiana para uma distribuição de Extremos, Anais do XII SOBRAPO (1989) 304-309, Ceará.

[12] Ferreira, M.J.S., Modelos Bayesianos para extremos, Tese de doutorado, Departamento de Engenharia Elétrica (Sistemas), Pontifícia Universidade Católica do Rio de Janeiro (1991).
[13] Gradshteyns, I.S. and Ryzhik, I.M., Table of Integrals, Lecture Notes in Mathematics 529 (1965), Springer-Verlag, Berlin.
[14] Goldstein, M., Bayesian Analysis of regression Problems, Biometrika 63 (1976) 51-58.
[15] Gumbell, E.J., Statistics of Extremes, Columbia Univ. Press, New York (1958).
[16] Harrison, P.J. and Stevens, C.F., Short-Term Sales Forecasting, Op. Res. Quart. 22 (1971) 341-362.
[17] Harrison, P.J. and Stevens, C.F., Bayesian Forecasting, J.Roy Statist. Soc. 38 (1976) 205-247 (with discussion).
[18] Hartigan, J.A., Linear Bayesian Methods, J.R. Statist. Soc. 31 (1969) 446-454.
[19] Hill, B.M., A simple general approach to inference about the tail of a distribution, Ann Statist. 3 (1975) 1163-1174.
[20] Johnson, N.I. and Kotz, S., Distributions in Statistics Continuous Univariate Distributions, Hougton Mifflin Company.
[21] Kalman, R.E., A new approach to linear Filtering and Prediction Problems, J. of Basic Eng. 82 (1960) 35-45.
[22] Kalman, R.E. and Bucy, R.S., New results in Linear Filtering and Prediction Theory, J. of Basic Eng. 83 (1961) 95-108.
[23] Kalman, R.E., New Methods in Wiener Filtering Theory, in Proceedings of the First Symposium on Engineering Applications of Random Function Theory and Probability, eds. J.L. Radaneff and F. Kazin, New York, John Wiley (1963).
[24] Kinnison, Robert R., Applied Extreme Value Statistics, Macmillan, USA (1985).
[25] Migon, H.S., An approach to Non-Linear Forecasting Problems with Applications, Phd Thesis, Warmick University Coventry, UK (1984).
[26] North, M., Time-dependent stochastic models of floods, J. Hydraulics Division, A.S.C.E. (1980) 649-655.
[27] Pole, A. and West, M., Reference Analysis of the DLM, University of Warwick, Warwick Res. Report (1987).
[28] Pole, A., West, M. and Harrison, P.J., Non-normal and Non-linear Dynamic Bayesian Modelling, Departmentof Statistics, University of Warwick, Warwick Res Report 112 (1988), England (In Bayesian Analysis of Time Series and Dynamic Models, J.C. Spall (ed.) Marcel Dekker, New York).
[29] Pole, A., West, M. and Harrison, J., Applied Bayesian Forecasting and Time series Analysis, Chapman & Hall (1994).
[30] Pickands, J., Statistical inference using extreme order statistics, Ann. Statist. 3 (1975) 119-131.
[31] Smith, R.L., Threshold Methods for sample extremes, Department of Mathematics, Imperial College, London SW7 (1984).
[32] Souza, R.C., A Bayesian Entropy Approach to Forecasting, Phd Thesis, Statistics Dept., Warwick University, Coventry UK (1979).
[33] Turner, M.J., Estimation of the fifty year return wave height at Seven Stones, M.Sc. and D.I.C. Report, Department of Mathematics, Imperial College (1982).
[34] Weisman, I., Estimation of parameters and large quantiles, based on the K largest observations, J. Amer-Statist. Soc. 73 (1978) 812-815.
[35] Weisman, I., Estimation of tail parameters under Type I censoring, Commun Statist. Theor. Math. 9 (1980) 1165-1175.
[36] West, M., Harrison, P.J. and Migon, H.S., Dynamic Genralized Linear Models and Bayesian Forecasting, J.A.S.A. 80 (1985) 73-97 (with discussion).
[37] West, M. and Harrison, P.J., Monitoring and Adaptation in Bayesian Forecasting Models, J.A.S.A. 81 (1986) 741-750.
[38] West, M. and Harrison, P.J., Bayesian Forecasting and Dynamic Models, Springer-Verlag (1989).
[39] Yevjevich, V., International Water Resources Institute, School of Engineering and Applied Science, George University, Washington, D.C. 20052, USA (1985).

INSTRUÇÕES AOS AUTORES

Os autores que desejam submeter um artigo à Investigação Operacional devem enviar três cópias desse trabalho para:

Prof. Joaquim J. Júdice
Departamento de Matemática
Universidade de Coimbra
3000 Coimbra, Portugal

Os artigos devem ser escritos em Português ou Inglês. A primeira página deve conter a seguinte informação:

- Título do artigo
  - Autor(es) e instituição(ões) a que pertence(em)
  - Abstract (em inglês)
  - Resumo
  - Keywords (em inglês)
  - Título abreviado

As figuras devem aparecer em separado de modo a poderem ser reduzidas e fotocopiadas. As referências devem ser numeradas consecutivamente e aparecer por ordem alfabética de acordo com os seguintes formatos:

Artigos: autor(es), título, título e número da revista (livro com indicação dos editores), ano, páginas.

Livros: autor(es), título, editorial, local de edição, ano.

ÍNDICE

Apdio

CESUR - Instituto Superior Técnico
Av. Rovisco Pais - 1049 - 001 LISBOA
Telef. 840 74 55 - Fax. 840 98 84
http://www.inescc.pt/~apdio