Carlos Manuel Chorro Simões Barrico

(Universidade da Beira Interior)

Uma Abordagem ao Problema de Caminho Mais Curto Multiobjectivo – Aplicação ao Problema de Encaminhamento em Redes Integradas de Comunicações

Dissertação de Mestrado em
Sistemas e Automação / Área de Gestão da Informação
sob a orientação do Prof. Doutor Carlos Henggeler Antunes

Departamento de Engenharia Electrotécnica

Faculdade de Ciências e Tecnologia

Universidade de Coimbra

Coimbra, Setembro de 1998

Agradecimentos

Ao Prof. Carlos Henggeler Antunes pela orientação científica deste trabalho e pelo incentivo dado ao longo deste tempo todo.

Ao Prof. Ernesto Queirós Martins, ao José Luís dos Santos e à Marta Pascoal pela ajuda prestada na parte dedicada aos problemas de determinar os k caminhos mais curtos, nomeadamente por terem colocado os seus algoritmos à disposição e por terem lido a parte da tese dedicada a estes problemas particulares.

Ao Prof. José Craveirinha pelo apoio dado na parte relativa ao problema do Encaminhamento.

Aos colegas de trabalho do Departamento de Matemática/Informática da Universidade da Beira Interior pelo apoio manifestado durante todo este tempo.

Os meus agradecimentos são extensíveis a todos aqueles que directa ou indirectamente contribuíram para a conclusão deste trabalho.

Índice Geral

CAPÍTULO 4

O Problema do Caminho Mais Curto com Múltiplos Objectivos

CAPÍTULO 5

Abordagem Interactiva ao Problema de Caminho Mais Curto Multiobjectivo

CAPÍTULO 6

Aplicação ao Problema de Encaminhamento em Redes Integradas de Comunicações

Índice de Figuras

Índice de Algoritmos

Resumo

Nos últimos anos tem-se observado um grande avanço nos processos de resolução de problemas de programação linear multiobjectivo com estrutura de rede, em particular aqueles que são modelados como de caminho mais curto. De facto, com a evolução dos computadores, principalmente em termos de memória e de velocidade de execução, e de algumas técnicas de cálculo, como por exemplo, os algoritmos para determinar os k caminhos mais curtos de uma rede, é hoje possível determinar de forma computacionalmente eficiente (para certos tipos de redes) todas as soluções não dominadas (soluções para as quais não existe outra solução admissível que melhore em simultâneo todos os objectivos) de um problema de caminho mais curto multiobjectivo.

No entanto, uma outra questão surge quando se pretende seleccionar apenas uma solução não dominada, a qual deve ser a "melhor possível" de acordo com as preferências do agente de decisão (AD), que é a entidade responsável pela escolha da solução.

Um dos processos para resolver este último problema consiste em determinar todas as soluções não dominadas e colocá-las à disposição do AD para posterior análise e escolha de uma delas. No entanto, escolher uma entre muitas não é tarefa fácil, até porque, geralmente, muitas delas, apesar de diferentes, têm características semelhantes. Ou seja, muitas vezes é vantajoso, quer do ponto de vista de esforço computacional, quer do ponto de vista do esforço de tratamento da informação exigido ao AD, calcular apenas um conjunto reduzido de soluções que o AD vai identificando durante o processo de pesquisa.

O AD intervém assim no processo de pesquisa de soluções, indicando as suas preferências, para que apenas algumas regiões sejam analisadas (aquelas onde se localizam as soluções que melhor correspondem às suas preferências), não havendo necessidade de determinar todas as soluções não dominadas do problema. Os métodos interactivos caracterizam-se pela existência de fases de decisão e de cálculo, que alternam entre si, até se encontrar uma solução não dominada que agrade ao AD, de tal forma que seja escolhida para solução final.

Neste trabalho é proposto uma abordagem interactiva para a resolução de problemas de caminho mais curto com dois e três objectivos. Esta abordagem foi implementada no quadro operacional de sistemas de apoio à decisão, a qual utiliza a interacção AD–computador de modo que as preferências do AD sejam utilizadas no processo de pesquisa de uma solução final. Os sistemas fornecem ao AD um conjunto de informação associada às soluções não dominadas encontradas, utilizando, para tal, meios gráficos e analíticos, que são apresentados aproveitando as potencialidades do ambiente "Windows".

Também é apresentado um exemplo de aplicação de uma das abordagens algorítmicas ao problema de encaminhamento ("Routing") em redes integradas de comunicações. O encaminhamento está relacionado com operações em tempo real sobre a rede e com um mecanismo de controlo ao nível das chamadas, através do qual um dos caminhos disponíveis é seleccionado para estabelecer a comunicação entre dois pontos (origem e destino) da rede.

Alguns algoritmos tradicionais de encaminhamento consistem em resolver problemas de caminho mais curto com um só objectivo, em que a função objectivo se baseia numa métrica simples (ou numa única função de diferentes métricas). No entanto, o problema torna-se mais complexo quando se introduzem restrições para satisfazer os vários requisitos de qualidade de serviço (QoS).

O problema de encaminhamento é modelado como um problema de caminho mais curto com múltiplos objectivos, no qual as diferentes métricas relevantes são explicitamente incorporadas na formulação do problema. A abordagem algorítmica consiste em optimizar funções objectivo de somas pesadas e em utilizar um algoritmo eficiente dos k caminhos mais curtos. Os requisitos de QoS são expressos como restrições adicionais "leves" nos valores das funções objectivo em termos de limites requerido e aceitável, os quais definem regiões de preferência no espaço dos objectivos. Estas regiões são então pesquisadas para determinar caminhos (se existirem) que satisfaçam aqueles requisitos.

C A P Í T U L O 1

Introdução Geral

1. O problema multicritério

A resolução de muitos dos problemas que surgem no dia a dia consiste em escolher entre várias alternativas viáveis; ou seja, tomar decisões. A maioria destes problemas é de difícil resolução, uma vez que envolvem múltiplos critérios, geralmente conflituosos entre si. No entanto, em situações de grande pressão, como sejam as situações de crise, opta-se usualmente por se utilizar modelos muito simples, em que apenas um dos critérios de decisão assume relevância, colocando de lado os restantes aspectos do problema, reduzindo–o a um problema de optimização do objectivo associado ao critério "relevante" — geralmente o que se encontra associado à questão económica.

Exemplos de problemas que envolvem múltiplos critérios são :

(Ex.1) Escolher o melhor local para a construção de uma ponte, em que os critérios poderiam ser o custo, o impacto sobre o rio (ambiental e utilização do rio), o volume de tráfego, o impacto sobre as margens, a estética, o custo da travessia, etc..

(Ex.2) Encontrar o trajecto mais económico, para efectuar a entrega/recolha de produtos aos clientes duma determinada firma, em que os critérios poderiam ser o tempo, a distância, o atraso, o tráfego, etc..

Por existir conflito entre os critérios, a entidade que tem a responsabilidade de decidir, o agente de decisão (AD), tem de ponderar os compromissos a efectuar com vista a encontrar a solução que lhe pareça mais satisfatória. Neste caso, estamos perante um **problema multicritério**.

Durante a década de 50, com o desenvolvimento da Investigação Operacional (IO) como disciplina científica, pensou-se que esta iria desenvolver métodos e técnicas capazes de resolver a maioria dos problemas de decisão que se colocam a vários níveis, quer na indústria, quer no sector de serviços.

Como a maior parte dos problemas com vários critérios surgia no seio das empresas, muitos autores sobrevalorizaram a utilização de modelos matemáticos de optimização, pois consideravam que todos os modelos se baseavam na optimização duma função objectivo : função valor ou utilidade[1]. A formulação da função objectivo era um problema considerado menor, pois a unidade monetária aparecia como a única medida de "benefício social", como se pode verificar, por exemplo, em Hitch : "A medida monetária, apropriada tanto para receitas como para despesas, permite, em geral, a utilização de uma única função objectivo : os lucros da operação ou, a um nível mais lato, os lucros da empresa" [7].

Porém, no início da década de 70, com a complexidade crescente do ambiente sócio-económico que caracteriza as sociedades tecnológicas modernas, verificou-se que quase todos os problemas mais importantes envolviam vários critérios, geralmente conflituosos entre si, tornando-se difícil a formulação, modelação e obtenção da solução para o problema. Agora, as preocupações dos decisores não se limitam apenas ao campo económico em sentido restrito, mas, pelo contrário, já abrangem outros campos como o político, o social, o meio ambiente, o estético e outros, em que a qualidade de vida da sociedade humana ganha foros de cidadania cada vez mais fortes.

Torna-se então necessário ponderar os vários conflitos existentes entre os critérios, de forma a seleccionar a solução que melhor reflicta as preferências do AD.

Assim sendo, surgiu a necessidade de desenvolver técnicas de apoio à decisão capazes de ter em conta a complexidade e natureza conflituosa dos critérios em presença em problemas reais.

O problema multicritério está relacionado com os métodos e procedimentos, pelos quais os vários critérios podem ser formalmente associados no processo de análise. De uma forma geral, estes problemas dividem-se em problemas multiatributo e multiobjectivo. Os primeiros caracterizam-se pela existência de um número finito de alternativas explicitamente conhecidas, a avaliar em presença de múltiplos critérios. Os problemas multiobjectivo referem-se aos casos em que as alternativas são definidas implicitamente por um conjunto de restrições e os critérios são operacionalizados através de funções a optimizar [25], [7].

[1] Expressão analitica construida a partir da parametrizção dos vários critérios.

A situação referida em (Ex.1) corresponde a um problema multiatributo, pois as alternativas a considerar são explicitamente conhecidas à partida. A situação referida em (Ex.2) corresponde a uma problema multiobjectivo, uma vez que as alternativas possíveis são implicitamente definidas por um conjunto de restrições matemáticas.

2. O problema multiobjectivo

De uma forma geral, um problema de programação linear multiobjectivo (PLMO) pode ser formulado do seguinte modo [7] :

$$\text{Maximizar} \quad f_1(x) = c_1 x$$

$$\text{Maximizar} \quad f_2(x) = c_2 x$$

$$\ldots$$

$$\text{Maximizar} \quad f_h(x) = c_h x$$

$$\text{sujeito a} \quad x \in X = \{ x \in \Re^n : x \geq 0,\ A x = b,\ b \in \Re^m \}$$

em que,

h é o número de funções objectivo (critérios),

n é o número de variáveis do problema (de decisão e folga),

m é o número de restrições do problema,

X é a região admissível no espaço das decisões (ou das variáveis),

x é o vector das variáveis do problema ($\mathbf{x} = [\ x_1\ \ x_2\ \ \ldots\ \ x_n\ ]^T$ é solução admissível),

$\mathbf{c_j}$ (j = 1, ..., h) é o vector dos coeficientes da função objectivo f_j ,

A é a matriz dos coeficientes tecnológicos (m×n),

b é o vector dos termos independentes (recursos disponíveis ou requerimentos),

o que não implica perda de generalidade, pois mediante operações convenientes é sempre possível transformar qualquer problema num de maximização.

A região admissível no espaço das funções objectivo (o conjunto de todas as imagens dos pontos em X), pode ser definida da seguinte forma :

$$Z = \{ z \in \Re^h : z = (f_1(x), f_2(x), ..., f_h(x)), x \in X \}.$$

Na resolução de problemas com apenas um objectivo, procura-se encontrar a ***solução óptima***, ou seja, a solução admissível que optimize a função objectivo, cujo valor é único, mesmo que existam soluções óptimas alternativas. Logicamente, em problemas com múltiplos objectivos, esse conceito não é aplicável, uma vez que uma solução admissível que optimize

um dos objectivos, não optimiza, em geral, os restantes objectivos, quando estes estão em conflito.

Na resolução de problemas multiobjectivo, pretende-se encontrar uma ***"melhor" solução de compromisso*** para o AD, que possa constituir uma ***solução final*** do problema de decisão. Desta forma, a noção de solução óptima é substituída pela noção de ***solução não dominada*** (também designada por ***eficiente***, ***óptima de Pareto*** ou ***não inferior***).

Uma solução admissível diz-se **dominada** por outra, se ao passar-se da primeira para a segunda existir "melhoria" de pelo menos um dos objectivos, permanecendo inalterados os restantes. Por outro lado, uma solução **não dominada** caracteriza-se por não existir uma outra solução admissível que melhore simultaneamente todos os objectivos, isto é, a melhoria num objectivo é alcançada à custa de piorar, pelo menos, um dos outros [7]. Matematicamente, tem-se :

i) Sejam $x = [\ x_1\ \ x_2\ \ \dots\ \ x_n\ ]^T$ e $y = [\ y_1\ \ y_2\ \ \dots\ \ y_n\ ]^T$ duas soluções admissíveis de um problema multiobjectivo ($x, y \in X$). A solução x ***domina*** a solução y se e só se

$$f_j(x_1, x_2, \dots, x_n) \geq f_j(y_1, y_2, \dots, y_n) \qquad \text{para qualquer j e,}$$

$$f_j(x_1, x_2, \dots, x_n) > f_j(y_1, y_2, \dots, y_n) \qquad \text{para algum j, com j = 1, 2, …, h}$$

ii) Seja x uma solução admissível de um problema multiobjectivo ($x \in X$). A solução x diz-se ***não dominada*** se e só se não existir outra solução admissível y ($y \in X$) que domine x.

O conceito de eficiência é geralmente utilizado para pontos no espaço de decisão, enquanto o conceito de não dominância é utilizado para a respectiva imagem no espaço das funções objectivo. Ou seja, uma solução não dominada é a imagem de uma solução eficiente [7]. Matematicamente, tem-se :

a) Uma solução admissível x^1 é ***eficiente*** ($x^1 \in X$) se e só se não existe outra solução admissível x^2 ($x^2 \in X$) tal que $f(x^2) \geq f(x^1)$ e $f(x^2) \neq f(x^1)$, em que $f = (f_1, f_2, ..., f_h)$. Caso contrário, x^1 é ***ineficiente*** [25]. Ao conjunto das soluções eficientes, dá-se o nome de ***fronteira eficiente*** ou ***conjunto de soluções não inferiores***.

b) Seja $z^1 \in Z$. Então z^1 é ***não dominado*** se e só se não existe outro $z^2 \in Z$, tal que $z^2 \geq z^1$ e $z^2 \neq z^1$. Caso contrário, z^1 é um vector de objectivos ***dominado*** [25].

Uma solução de compromisso satisfatória para o problema multiobjectivo deverá ser eficiente, cujos valores das funções objectivo associados àquela solução são satisfatórios para o AD e de tal modo que seja aceitável como solução final do processo de decisão. Desta forma, é

apenas sobre o conjunto das soluções eficientes que deve recair a atenção do analista[2] e do AD.

Entre quaisquer duas soluções não dominadas verifica-se que a uma melhoria em pelo menos um dos objectivos se encontra sempre associado um sacrifício em pelo menos um dos outros objectivos. Isto é, verifica-se sempre uma compensação ("trade-off") entre objectivos no conjunto das soluções não dominadas.

3. Métodos para resolver problemas multiobjectivo

Nos últimos anos tem sido dedicado um esforço considerável ao desenvolvimento de métodos para o cálculo de soluções não dominadas, tendo sempre em atenção o contributo que o AD pode fornecer na procura de tais soluções, através do seu sistema de preferências. Desta forma, excluindo o caso trivial em que existe uma solução que optimiza em simultâneo todos os objectivos, podem-se considerar 3 abordagens para resolver um problema multiobjectivo, consoante o sistema de preferências do AD é incorporado *a priori*, *a posteriori* ou *progressivamente*, no processo de decisão [7] :

- métodos em que é feito uma agregação *a priori* de preferências,
- métodos geradores das soluções eficientes (não há articulação de preferências),
- métodos de "articulação progressiva de preferências" (interactivos).

Nos do primeiro tipo, o AD começa por indicar as suas preferências, a partir das quais é possível transformar o problema inicial num problema monocritério, por exemplo, através da construção de uma função utilidade. Desta forma, apesar de existir modelação multicritério do problema, pretende-se optimizar a função utilidade, cuja solução óptima será a solução final. A maior dificuldade deste processo está no facto de não ser possível, na maioria dos casos, obter uma representação matemática da função utilidade por parte do AD. Ou seja, é difícil obter os parâmetros para construir uma função utilidade que agregue, numa única dimensão, todos os critérios em análise.

Nos do segundo tipo, são calculadas todas as soluções eficientes do problema (ou parte), que depois são colocadas à disposição do AD para serem avaliadas. As críticas de que são alvo estes métodos devem-se ao elevado esforço computacional necessário para o cálculo exaustivo das soluções eficientes.

[2] Entidade responsável pela análise ou estudo do problema.

Nos métodos do terceiro tipo, o AD expressa as suas preferências através de um processo de diálogo com a componente procedimental, de forma a conduzir a pesquisa para a zona da região admissível onde se localizam as soluções que melhor correspondem ao seu sistema de preferências. Na prática, este tipo de métodos tem mostrado ser o mais eficaz na pesquisa de uma solução final, através da utilização de processos de interacção Homem-máquina, alternando fases de cálculo com fases de diálogo. A intervenção humana no processo de pesquisa da solução é uma das características que distingue os métodos de programação multiobjectivo dos tradicionais da programação matemática (com um só objectivo).

Os métodos que utilizam o cálculo exaustivo para identificar o conjunto de soluções eficientes exigem um elevado esforço computacional e não têm em atenção a experiência humana, que poderia servir de apoio na resolução do problema, visto existirem múltiplos objectivos em conflito, e consequentemente, um grande grau de subjectividade no problema.

Os métodos interactivos, aproveitando a intervenção do AD, reduzem a zona de pesquisa, de forma quer a minimizar o esforço computacional, quer o esforço do AD no processamento da informação.

4. Sistema de apoio à decisão

O ***apoio à decisão*** é muitas vezes definido como a geração e selecção de opções ou alternativas. Também há quem o relacione com a gestão estruturada de grandes volumes de informação [21]. Outros há ainda, como Andriole, que encaram a questão como um conjunto de funções integradas que suportam todas as fases do processo de tomada de decisões, englobando componentes como a recolha de informação e a posterior avaliação [2].

O ***apoio à decisão***, segundo Roy [22], é uma actividade que, baseando-se em modelos claramente explícitos e totalmente formalizados, procura obter elementos de resposta às questões que um interveniente no processo de decisão coloca a si próprio. Elementos estes que são concorrentes para esclarecer a decisão e, normalmente, para estabelecer um comportamento para aumentar o grau de coerência entre a evolução de todo o processo, a definição dos objectivos e o próprio sistema de valores, ao serviço dos quais é desenvolvida essa actividade.

A partir do final da década de 80, tem-se observado um desenvolvimento considerável de métodos interactivos, os quais têm vindo a constituir meios de apoio à tomada de decisão,

em inúmeros problemas em áreas como a engenharia, a gestão, ou o planeamento nos sectores público e privado. Na verdade, a partir daqui verificou-se, com mais evidência, a convergência de ideias e a eliminação de barreiras entre diferentes disciplinas, que em alguns casos competiam entre si — os cientistas de diferentes áreas passaram a participar cada vez mais em programas de investigação multidisciplinar. As primeiras contribuições vieram das áreas do processamento de informação e da matemática, para depois se estenderem a campos como o da inteligência artificial, do comportamento organizacional e humano, etc. [21].

A característica essencial dos métodos interactivos é a existência de fases de cálculo, que alternam com fases de diálogo. Em cada iteração, o AD analisa uma ou mais soluções, que ao serem confrontadas com as suas preferências, contribuem para a orientação do processo de pesquisa de novas soluções, durante o processo de decisão. Desta forma, podem ser tomados diferentes caminhos, conduzindo o processo de cálculo para a zona da região admissível onde se encontram as "melhores" soluções, segundo a perspectiva do AD. A condição de paragem do processo interactivo é a satisfação do AD com o conhecimento adquirido sobre o problema, que lhe permite tomar uma opção fundamentada, e não o teste de convergência de alguma função utilidade.

O princípio subjacente a um ***sistema de apoio à decisão*** (***SAD***) é ajudar interactivamente o AD em todo o processo de decisão, reduzindo progressivamente o âmbito da pesquisa e minimizando tanto o esforço computacional, como o esforço mental que é exigido ao AD.

A interacção Homem-máquina é uma componente fundamental de um SAD, tendo como objectivos contribuir para a criação do sistema de preferências do AD e ampliar as suas capacidades de processamento de informação e decisão. A interacção Homem-computador tem vindo a ganhar grande importância no domínio do apoio à decisão, em particular em problemas com vários critérios. Isto deve-se à evolução acelerada que os meios tecnológicos têm sofrido, o que permite uma incorporação mais fácil de técnicas de interacção e visualização [3].

De facto, a evolução que a informática tem sofrido, especialmente os computadores, cada vez mais potentes e mais baratos, tem ajudado o desenvolvimento de interfaces Homem-computador, melhorando os aspectos relacionados com o diálogo num SAD. Desta forma, o AD pode dialogar directamente com o computador, utilizando interfaces que se baseiam em representações gráficas dos problemas e que permitem a actuação do utilizador sobre esse gráficos, através de dispositivos adequados, havendo a possibilidade de interrogação e experimentação.

No desenvolvimento computacional de sistemas de apoio à decisão, deve existir um equilíbrio entre as potencialidades que são oferecidas ao utilizador e a facilidade de aprendizagem e uso. Para isso, deve ter-se em atenção os limites das capacidades humanas (sobretudo a memória e paciência), dando relevo às suas principais potencialidades (nomeadamente a inspecção visual) [3].

De forma a tirar partido das capacidades de processamento de informação do AD, as técnicas de diálogo devem utilizar extensivamente gráficos e oferecer um ambiente flexível e amigável servindo-se de todos os recursos do computador, mantendo sempre o utilizador no controlo do processo de pesquisa de soluções. A informação numérica detalhada deve encontrar-se acessível, sempre que requerida pelo AD. O utilizador deve poder controlar o comportamento do programa, através de operações simples.

Segundo Sage [23], sob o ponto de vista tecnológico, um SAD é composto por 3 níveis: os utensílios (computadores, programas e métodos, nomeadamente de investigação operacional, necessários para a implementação do sistema), o gerador (geralmente uma linguagem de alto nível que permite a construção do sistema específico) e os mecanismos de controlo do AD.

5. Problemas com estrutura de rede

Quando se fala em "redes", pensa-se logo numa grande quantidade de dados estruturados de uma forma própria, ocorrendo imediatamente as redes físicas (telefónica, eléctrica, rodoviária, ferroviária, de esgotos, de televisão por cabo, informática, etc.), pois são as que melhor identificam esta estrutura.

Em todos estes casos pretende-se enviar alguma entidade (electricidade, um produto para consumir, uma pessoa ou um veículo, uma mensagem, etc.) de um ponto para outro numa rede específica e da forma mais eficiente possível, tanto no sentido de fornecer um bom serviço aos utilizadores da rede, como de utilizar uma transmissão (normalmente dispendiosa) fácil e eficaz [1].

A resolução destes problemas, assim como de muitos outros, requer uma modelação matemática que utilize uma estrutura em rede para organizar os seus dados, de forma que o modelo se possa apoiar em algoritmos específicos utilizados no tratamento de redes.

A modelação matemática de inúmeros problemas recai num conjunto de modelos de fluxo em redes específicos, dos quais se salientam os seguintes [1] :

1. *Problema do fluxo de custo mínimo*. Pretende-se determinar o menor custo de envio de uma mercadoria através de uma rede, de forma a satisfazer a procura em determinados nós a partir da oferta noutros nós, atendendo a restrições de capacidade nos arcos. Este modelo tem uma quantidade de aplicações muito conhecidas, como sejam : a distribuição de um produto a partir da fábrica até aos armazéns, ou desde os armazéns até aos centros de venda; o fluxo de matérias-primas e de produtos intermédios pelas várias máquinas numa linha de montagem; o encaminhamento dos automóveis através de uma rede urbana de ruas; o encaminhamento de chamadas através de um sistema telefónico.
2. *Problema do caminho mais curto*. Pretende-se determinar o caminho de custo (ou comprimento) mínimo entre um nó origem e um nó destino de uma rede. Algumas das aplicações deste problema são, por exemplo : determinar o caminho de menor comprimento, ou que gaste menos tempo ao percorrê-lo, ou que tenha o máximo de segurança, entre dois nós de uma rede. Além destas aplicações, este modelo também pode ser aplicado a muitos outros problemas, tais como substituição de equipamentos, projectos de catalogação ou inventariação, encaminhamento de mensagens em sistemas de comunicação, etc..
3. *Problema do fluxo máximo*. Procura-se uma solução admissível que envie a máxima quantidade de fluxo de um nó origem para um nó destino, tendo em conta as restrições de capacidade dos arcos da rede. Exemplos deste problema são : determinar o fluxo constante máximo de produtos petrolíferos numa rede de oleodutos, de carros numa rede de estradas, de mensagens numa rede de telecomunicações e de electricidade numa rede eléctrica.
4. *Problema de afectação*. Pretende-se afectar (associar um para um) da forma mais económica (menor custo) tarefas a indivíduos, entendidas estas entidades em sentido geral, em que a quantidade de tarefas é igual à quantidade de indivíduos. Exemplos de problemas de afectação são : associar pessoas a projectos, operações a máquinas, inquilinos a apartamentos e nadadores a provas numa competição de natação.
5. *Problema de transportes*. Pretende-se determinar a forma mais económica de enviar directamente a mercadoria produzida em vários locais (fábricas), em quantidades limitadas (oferta), para os clientes que se encontram geograficamente dispersos, cada um com uma procura a satisfazer, tendo em conta a oferta de cada um dos locais.

6. *Problema de circulação*. Pretende-se determinar um fluxo possível que respeite os limites inferior e superior impostos em cada arco. Desde que nunca se introduza qualquer fluxo crescente na rede, ou se extraia qualquer fluxo dela, todo o fluxo circula pela rede. O objectivo é determinar a circulação que tenha o menor custo. Por exemplo, o esquema de uma lista de encaminhamentos de uma linha aérea comercial, em que qualquer avião circula entre os aeroportos de várias cidades : o limite inferior imposto num arco (i, j) é 1 se a linha aérea precisar de fornecer serviço entre as cidades **i** e **j**, e assim, enviar um avião por este arco (na realidade, os nós representam uma combinação tanto de uma localização física como de um período de tempo até que seja ligado um arco; por exemplo, Lisboa às 8 horas com Madrid às 9 horas).
7. *Problema da árvore abrangente mínima*. Pretende-se identificar a árvore abrangente de valor mínimo, isto é, uma solução (árvore) que contenha todos os nós da rede, cujo somatório dos valores dos arcos seja mínimo. As aplicações deste problema são variadas, como por exemplo : construção de auto-estradas e de vias férreas que "passam" por várias cidades, colocação de oleodutos a ligar locais de perfuração, refinarias e postos de venda ao consumidor, ligação de fios eléctricos a painéis de controlo.

O facto de muitos dos problemas que utilizam estes modelos serem de natureza multiobjectivo, levou a que recentemente se tenha dedicado algum esforço de investigação na formulação e resolução de problemas de optimização em redes, particularmente os de caminho mais curto. Os critérios mais utilizados nestes problemas são : custo, tempo, distância, acessibilidade, satisfação exigida, protecção do ambiente, minimização do risco, segurança, entre outros. Por exemplo, ao pretender-se determinar o melhor trajecto, em termos económicos, relativamente a um sistema de entrega/recolha de produtos numa cidade (considera-se a cidade representada em rede, onde os nós são os locais de entrega/recolha e os arcos as ruas entre aqueles locais), pode-se formular o problema como multiobjectivo (existem vários critérios a considerar : tempo de travessia, distância a percorrer, tráfego, etc.) com estrutura em rede (os dados podem ser representados, em termos visuais, utilizando uma estrutura em rede — mapa simplificado).

6. Objectivos do trabalho

A resolução de muitos problemas de programação linear multiobjectivo com estrutura em rede, requer uma modelação como problema de caminho mais curto. Desta forma, apesar de existirem vários processos para resolver estes problemas, o mais indicado é aquele que

utiliza a interacção com o decisor, uma vez que muitas vezes o mais difícil não é determinar as soluções não dominadas, mas sim escolher apenas uma entre tantas. Na realidade, os processos interactivos são os mais vantajosos, já que o número de soluções não dominadas que necessitam de ser determinadas pode ser muito pequeno, assim como permite evidenciar os compromissos entre os objectivos envolvidos inerentes a um conjunto de soluções não dominadas.

Neste sentido, desenvolveram-se dois sistemas de apoio à decisão, com o objectivo de se identificar a "melhor" solução de compromisso em problemas de caminho mais curto bi e tri-objectivo.

O método associado ao primeiro sistema, consiste em analisar todo o espaço dos objectivos, em que as soluções são determinadas segundo uma determinada direcção de pesquisa, a qual está associada aos valores que optimizam (minimizam) cada um dos objectivos.

O método utilizado no segundo sistema, baseia-se no procedimento interactivo descrito por Current et al. [10] para problemas de caminho mais curto bi-objectivo, ao qual foi associado um algoritmo para determinar os k caminhos mais curtos. Neste sistema, os conceitos associados ao problema bi-objectivo foram adaptados aos problemas com três objectivos.

Os dois sistemas propostos incluem interfaces AD–Computador, as quais utilizam meios gráficos para a interpretação dos resultados que vão sendo obtidos e para o diálogo com o AD. Estes sistemas foram desenvolvidos para computadores pessoais que possuam o sistema operativo "Windows" (95 ou NT). A razão do sistema ser implementado para funcionar em ambiente "Windows", está relacionada com o facto deste sistema operativo estar muito vulgarizado, o que permite a utilização dos sistemas por uma grande quantidade de pessoas, e também por oferecer uns aspectos gráficos bastante agradáveis.

7. Organização da tese

Esta dissertação encontra-se dividida em vários capítulos, os quais traduzem os diferentes aspectos do trabalho realizado subjacente à tese. Desta forma, e resumidamente, podem-se sintetizar aqueles aspectos nos seguintes pontos :

- No capítulo 2 apresentam-se os principais conceitos básicos e terminologia associados a grafos e redes. São também descritas algumas estruturas de dados utilizadas para representar redes em computadores.

- No capítulo 3 é abordado o problema de caminho mais curto com um só objectivo, através da análise de alguns algoritmos para resolver este tipo de problemas. É também apresentado o problema de determinar os k caminhos mais curtos, através da descrição de alguns algoritmos para os resolver.
- No capítulo 4 é apresentado, em termos gerais, o problema de caminho mais curto com vários objectivos. No entanto, o problema com dois objectivos é abordado em pormenor, descrevendo-se alguns métodos e algoritmos utilizados para resolver tais problemas, assim como alguns algoritmos interactivos de apoio ao AD para determinar a "melhor" solução de compromisso destes problemas particulares.
- O capítulo 5 está destinado à apresentação de duas abordagens interactivas ao problema de caminho mais curto com dois e três objectivos, através de dois métodos que servirão de apoio ao AD no cálculo da "melhor" solução de compromisso dos problemas.
- O capítulo 6 apresenta uma aplicação de um dos métodos descrito no capítulo 5 a um problema de encaminhamento ("Routing"). Aquele algoritmo, que é interactivo, é transformado num automático, através da construção de regiões de prioridade a partir de valores correspondentes aos requisitos de qualidade de serviço de cada objectivo.
- O capítulo 7 está reservado à apresentação das conclusões acerca do trabalho realizado, assim como de vias de desenvolvimentos futuros nesta área de investigação.
- Em anexo encontra-se o manual do utilizador, o qual se pretende que sirva de auxílio na utilização da aplicação.

O trabalho realizado deu origem a um artigo que foi submetido para publicação [4] e às seguintes comunicações em reuniões científicas :

- Em Julho de 1997 a comunicação "A DSS for Multiple Objective Routing in Integrated Communication Networks", foi apresentada na XV European Conference on Operational Research, realizada em Barcelona, Espanha.
- Em Junho de 1998 a comunicação "Multiple Objective Routing in Integrated Communication Networks", foi apresentada na XIV International Conference on Multiple Criteria Decision Making, University of Virginia, Charlottesville, Estados Unidos.
- Em Julho de 1998 a comunicação "A Multiple Objective Approach to QoS Routing in Integrated Communication Networks", foi apresentada na conferência internacional Optimization'98, Universidade de Coimbra.

C A P Í T U L O 2

Grafos e Redes

1. Introdução

Um **grafo** é uma representação visual de um determinado conjunto de dados e da ligação existente entre alguns dos elementos desse conjunto. Desta forma, em muitos dos problemas que nos surgem, a forma mais simples de o descrever é representá-lo em forma de grafo, uma vez que a representação visual que apresenta trará vantagens na construção de um modelo matemático com vista à resolução do problema. Por esta razão é que muitas áreas do conhecimento (matemática, telecomunicações, engenharia civil, gestão, etc.) recorrem à teoria de grafos para resolver muitos problemas.

Uma vez que as redes surgem em numerosos contextos e em variadas formas, muitos domínios têm contribuído com ideias importantes, para a evolução do estudo de problemas de fluxo em redes, já que a forma mais simples de modelar matematicamente muitos problemas inerentes a esses domínios é utilizar redes. Apesar desta diversidade ter produzido inúmeros benefícios, como a introdução de perspectivas muito ricas e variadas, também trouxe alguns custos : por exemplo, a literatura da teoria de grafos e de redes não possui uniformização, fazendo com que vários autores adoptem uma grande variedade de convenções e notação. Desta forma, podem-se formular problemas de fluxo em redes de diferentes formas padrão e utilizar muitos grupos alternativos de definições e terminologia. As definições e terminologia aqui adoptadas são, na sua maioria, as que se encontram em Ahuja et al. [1].

2. Conceitos fundamentais de grafos

Um **grafo** é uma estrutura constituída por dois conjuntos finitos, um de vértices (nós ou nodos) e o outro de arestas (arcos ou ramos), e pode ser representado por $G = (N, A)$, em que N e A são os conjuntos de vértices e arestas, respectivamente, com $A \subseteq N \times N$.

Cada aresta é representada por um par (i, j), com $\mathbf{i} \neq \mathbf{j}$ e $\mathbf{i}, \mathbf{j} \in N$, em que **i** é a sua ***cauda*** e **j** a sua ***cabeça***; diz-se que (i, j) ***sai*** do vértice **i** e ***chega*** ao vértice **j**. Uma aresta diz-se **dirigida** (**orientada)** se for representada um par ordenado de vértices distintos e **não dirigida** (**não orientada)** se for representada por um par não ordenado de vértices distintos — uma aresta que liga os vértices **i** e **j** representa-se por (i, j) ou por (j, i). Um arco dirigido (i, j) pode ser visto como uma rua de um só sentido, que permite fluxo apenas de **i** para **j** e um arco não dirigido (i, j) pode ser visto como uma rua de dois sentidos, que permite fluxo em ambas as direcções (de **i** para **j** e de **j** para **i**).

A aresta (i, j) é ***incidente*** nos vértices **i** e **j**. A aresta (i, j) é uma ***aresta de saída*** do vértice **i** e uma ***aresta de chegada*** do vértice **j**. O vértice **j** diz-se **adjacente** ao vértice **i** se (i, j) for uma aresta. Duas **arestas** são **adjacentes** se forem ambas incidentes relativamente ao mesmo vértice. Um **vértice** é de **ordem k** se tiver k arestas a ele adjacente.

A ***lista de adjacência de arestas*** de um vértice **i**, A(i), é o conjunto de arestas que saem de **i**; isto é, $A(i) = \{ (i, j) \in A : j \in N \}$. A ***lista de adjacência de vértices*** de um vértice i, V(i), é o conjunto de vértices adjacentes a **i**; isto é, $V(i) = \{ j \in N : (i, j) \in A \}$. Tanto $|A(i)|$ como $|V(i)|$ correspondem à quantidade de arcos que saem do nó **i**.

Existem 3 tipos de grafos :

- **dirigido** (**orientado**) — todas as arestas são dirigidas,
- **não dirigido** (**não orientado**) — todas as arestas são não dirigidas e,
- **misto** — algumas arestas são dirigidas e outras são não dirigidas.

Um **grafo** diz-se **completo** se entre quaisquer dois vértices existir uma aresta dirigida (grafos dirigidos) ou não dirigida (grafos não dirigidos).

A **densidade** de um **grafo** é a razão entre a quantidade de arestas do grafo e a quantidade de arestas do grafo completo com o mesma quantidade de vértices.

Um grafo $G' = (N', A')$ é um subgrafo de $G = (N, A)$ se $N' \subseteq N$ e $A' \subseteq A$.

Considerem-se dois vértices, **s** e **t**, de um grafo $G = (N, A)$. Um **trajecto** de **s** para **t** é um sucessão de vértices e arestas, $[\mathbf{s} = n_1, (n_1, n_2), n_2, ..., (n_{k-1}, n_k), n_k = \mathbf{t}]$, tal que :

a) $n_i \in N, \forall\, i \in \{ 1, ..., k \}$,

b) $(n_i, n_{i+1}) \in A, \forall\, i \in \{ 1, ..., k-1 \}$.

Alternativamente, um trajecto pode ser representado apenas pela sucessão de vértices ou de arestas. A notação que vai ser seguida é a sucessão de vértices; ou seja, $[\mathbf{s} = n_1, n_2, ..., n_k = \mathbf{t}]$.

Um **caminho** de **s** para **t** é um trajecto de **s** para **t** sem vértices repetidos. Um **ciclo** é um trajecto de **s** para **s**. Um **ciclo simples** é um caminho de **s** para **s**. Um **circuito** (**ciclo dirigido**) é um ciclo formado por arestas dirigidas. Um grafo dirigido sem ciclos diz-se **acíclico**. Um grafo diz-se **ligado** (**conexo**) se existir um trajecto entre quaisquer dois vértices.

Uma **árvore** é um grafo com um e um só caminho entre quaisquer dois vértices (ou seja, é um grafo conexo sem ciclos). Uma **árvore geradora** (**abrangente ou total**) de um grafo G é uma árvore formada por arestas de G e que contenha todos os vértices de G.

3. Conceitos fundamentais de redes

Uma **rede** é um grafo cujos vértices e/ou arestas têm associado valores numéricos (custos, capacidades e/ou oferta e procura). A terminologia utilizada em redes é **nós** (**nodos**) e **arcos**, em vez de vértices e arestas, respectivamente.

Uma rede pode ser representada por $G = (N, A, C)$, em que (N, A) é um grafo e C é o conjunto dos valores numéricos associados aos arcos (*comprimentos* dos arcos); ou seja, ao arco (i, j) está associado o valor c_{ij} (o *comprimento* do arco (i, j) é de c_{ij}). De uma maneira geral, os conceitos utilizados para grafos são extensíveis a redes.

Considere-se um caminho **p** de **s** para **t** na rede G. O *comprimento* do caminho p, c(p), é a soma dos *comprimentos* dos arcos que pertencem àquele caminho; ou seja,

$$c(p) = \sum_{(i,j) \in p} c_{ij} .$$

O conjunto de todos os caminhos de **s** para **t** numa rede G identifica-se por P. Ao caminho de menor *comprimento* dá-se o nome de ***caminho mais curto***.

Define-se **árvore mínima** (**árvore dos caminhos mais curtos**) com raiz em **s**, como a árvore que contém todos os nós da rede G acessíveis a partir de **s**, em que para cada nó **r** o único caminho de **s** para **r** é o caminho mais curto entre aqueles nós.

A **árvore geradora de custo mínimo** é a árvore geradora de G, em que é mínimo o somatório dos custos associados às respectivas arestas.

4. Representação computacional de redes

A eficiência computacional de um algoritmo para resolver problemas de optimização em redes não depende apenas das suas características intrínsecas, mas também, e muito, das estruturas de dados utilizadas para representar a rede no computador (formas de armazenar e manipular os dados associados à rede) e para armazenar os resultados intermédios necessários ao algoritmo. Geralmente, para representar uma rede no computador são necessários dois tipos de informação :

- a topologia da rede (estrutura dos nós e dos arcos),
- os dados (custos, capacidades e oferta/procura) associados aos nós e aos arcos.

Normalmente, o esquema utilizado para armazenar a topologia da rede irá sugerir, naturalmente, uma forma de armazenar a informação associada aos arcos e aos nós.

As representações mais comuns de uma rede são as seguintes : matriz de adjacência, matriz de incidência, listas de adjacência e vectores simulando listas múltiplas.

4.1. Matriz de adjacência

Dada uma rede dirigida $G = (N, A, C)$ com **n** nós e **m** arcos, a matriz de adjacência, ou mais exactamente a matriz de adjacência nó–nó, armazena a rede G numa matriz $M = \{ m_{ij} \}$ quadrada de ordem **n**, em que cada linha e cada coluna corresponde a um nó. Cada elemento $\mathbf{m_{ij}}$ da matriz assume um dos seguintes valores : 1 (se $(i, j) \in A$) ou 0 (se $(i, j) \notin A$).

Por outro lado, para cada parâmetro associado aos arcos da rede (custo, comprimento, capacidade, etc.) terá que existir uma matriz do mesmo tipo e para cada parâmetro associado aos nós terá que existir um vector de **n** elementos.

Nas redes não dirigidas, pode considerar-se que cada arco (i, j) corresponde a dois arcos dirigidos : (i, j) e (j, i). Neste caso a matriz de adjacência é simétrica.

Nos problemas em que a densidade da respectiva rede é muito baixa (matriz com grande quantidade de elementos nulos), como acontece na maioria dos problemas de circulação, este tipo de estrutura é bastante ineficiente, tanto relativamente ao tempo de pesquisa, como ao espaço de memória necessário ao armazenamento dos dados. Isto porque a matriz de adjacência permite armazenar $\mathbf{n^2}$ arcos e a rede apenas tem **m** (ou **2m** caso seja uma rede dirigida).

Apesar disso, a simplicidade desta representação permite a sua utilização para implementar, muito facilmente, os algoritmos de redes. Isto porque pode-se :

- determinar os parâmetros associado a qualquer arco (i, j), tomando simplesmente o elemento (i, j) das respectivas matrizes;
- obter facilmente os arcos que saem do nó **i**, examinando a linha **i** : se o j-ésimo elemento dessa linha tem o valor um (1) então (i, j) é um arco da rede;
- obter os arcos que chegam ao nó **j**, examinando a coluna **j** : se o i-ésimo elemento dessa coluna tem o valor um (1) então (i, j) é um arco da rede.

4.2. Matriz de incidência

Dada uma rede dirigida $G = (N, A, C)$ com **n** nós e **m** arcos, a matriz de incidência, ou mais exactamente a matriz de incidência nó-arco, armazena a rede G numa matriz constituída por **n** linhas e **m** colunas, em que cada linha está associada a um nó e cada coluna a um arco da rede. Cada elemento (i, a) assume um dos seguintes valores : 1 (se o arco **a** sai do nó i), –1 (se o arco **a** chega ao nó i) ou 0 (restantes casos).

Nas redes não dirigidas, os valores negativos (–1) são substituídos por positivos (+1), uma vez que todo o nó é considerado como *cauda* e como *cabeça* de um arco.

Também esta estrutura se torna bastante ineficiente, tanto relativamente ao tempo de pesquisa, como ao espaço de memória necessário para o armazenamento dos dados, uma vez que a quantidade de elementos reservados para a rede é **n.m** e esta tem apenas **m** elementos para serem armazenados (ou **2m** se for uma rede não dirigida). Em ambos os casos a matriz de incidência apenas tem **2m** valores diferentes de zero (dos **n.m** existentes) : cada coluna tem exactamente um (+1) e um (–1) — rede dirigida — ou dois (+1) — rede não dirigida.

Além disso, a quantidade de (+1) numa linha é igual à quantidade de arcos que saem do nó correspondente, qualquer que seja o tipo de rede. Da mesma forma, em redes dirigidas, a quantidade de (–1) numa linha é igual à quantidade de arcos que chegam ao nó

correspondente; consequentemente, a soma de (+1) e de (–1), numa linha, corresponde ao grau do nó que lhe está associado.

4.3. Listas de adjacência de arestas

Dada uma rede dirigida $G = (N, A, C)$ com **n** nós e **m** arcos, este tipo de representação armazena a lista de adjacência de arestas de cada nó numa simples lista ligada (cada elemento da lista de adjacência é uma lista ligada). Uma lista ligada é um grupo de células, cada uma contendo um ou mais campos. A lista de adjacência do nó i, A(i), será uma lista ligada com |A(i)| células, em que cada célula corresponde a um arco (i, j) da rede.

A célula correspondente ao arco (i, j) terá tantos campos quanto a quantidade de informação que é necessário guardar. Um dos campos guardará o nó j; depois existirá um conjunto de campos, onde serão guardados, por exemplo, o custo, a capacidade, o comprimento, etc., do arco; por último, terá que existir um outro campo, designado por *ligação ("link")*, que guardará um ponteiro para a próxima célula desta lista ligada (que corresponde a um outro arco que sai do nó i). A última célula da lista ligada conterá no campo *ligação*, por convenção, o valor zero.

Em redes não dirigidas, a informação de cada arco é armazenada em duas células diferentes, uma vez que cada arco pertence a duas listas de adjacência: o arco (i, j) pertence às listas de adjacência do nó **i** e do nó **j** — $(i, j) \in A(i)$ e $(i, j) \in A(j)$, pois $(i, j) = (j, i)$.

Uma vez que é necessário armazenar e aceder a **n** listas ligadas (uma para cada nó), também é necessário uma tabela de ponteiros que apontem para a primeira célula de cada lista ligada. Para tal, define-se uma tabela de dimensão **n**, designada por *primeiro ("first")*, cujo elemento *primeiro (i)* guarda um ponteiro para a primeira célula da lista de adjacência do nó **i**, A(i); se $A(i) = \varnothing$ então faz-se *primeiro (i) = 0*.

4.4. Vectores simulando listas múltiplas

Dada uma rede dirigida $G = (N, A, C)$ com **n** nós e **m** arcos, nas representações que utilizam vectores para simular listas múltiplas a numeração dos arcos segue uma determinada ordem : primeiro são numerados os arcos que saem do nó 1, depois os arcos que saem do nó 2, etc.. Os arcos que saem do mesmo nó, podem ser numerados por qualquer ordem.

As representações mais simples apenas necessitam de dois vectores de dimensão **m** :

- **Ant**, em que Ant (a) = nó cauda do arco **a**,
- **Suc**, em que Suc (a) = nó cabeça do arco **a**.

Para além destes dois vectores, cada um dos parâmetros associados aos arcos da rede necessita de um outro vector, de dimensão **m**, para armazenar os valores desses parâmetros.

Em redes não dirigidas, tal como acontece para a representação com listas de adjacência de arestas (ver secção 4.3 deste capítulo), a informação associada a cada arco é armazenada duas vezes.

No entanto, estas estruturas ainda podem ser melhoradas através das bem conhecidas representações denominadas por ***"forward star form"*** e ***"reverse star form"***.

A *"forward star form"* é uma representação muito semelhante à representação com listas de adjacência de arestas, uma vez que também esta guarda a lista de adjacência de cada nó. No entanto, em vez de listas ligadas são utilizados vectores para guardar a informação associada aos arcos da rede. Assim, determinam-se de forma eficiente, todos os arcos que saem de qualquer nó. Os vectores utilizados são os seguintes :

- **Point**, de dimensão n+1, que se define da seguinte forma :

  Point (1) = 1

  Point (i+1) = Point (i) + quantidade de arcos que saem de i ($1 \leq i \leq n$)

  (Point (n+1) = 1 + m)
- **Suc**, em que Suc (a) = nó cabeça do arco **a**.

Se Point (i) = Point (i+1) então não saem arcos do nó i.

A *"reverse star form"* permite determinar de forma eficiente todos os arcos que chegam a qualquer nó. Os vectores utilizados são os seguintes :

- **RPoint**, de dimensão n+1, que se define da seguinte forma :

  RPoint (1) = 1

  RPoint (i+1) = RPoint (i) + quantidade de arcos que chegam a i ($1 \leq i \leq n$)

  (RPoint (n+1) = 1 + m)
- **Ant**, em que Ant (a) = nó cauda do arco **a**.

Se RPoint (i) = RPoint (i+1) então não chegam arcos ao nó i.

4.5. Comparação entre as várias representações

Como se pode concluir, as representações que utilizam matrizes são muito ineficientes, tornando-se mais adequado a utilização de listas, uma vez que é apenas necessário espaço para representar a informação associada aos arcos que existem, não havendo espaço desperdiçado.

Comparando a representação que utiliza listas de adjacência de arestas com a representação que utiliza vectores para simular listas múltiplas, pode-se concluir que a última necessita de menos espaço para armazenar os dados da rede do que a primeira. No entanto, a primeira é mais eficiente quando se pretende adicionar ou remover arcos (ou nós) à rede.

C A P Í T U L O 3

O Problema do Caminho Mais Curto com um só Objectivo

1. Definição e formulação do problema

Os problemas de caminho mais curto com um só objectivo são fundamentais e frequentes quando se estudam problemas em redes, por exemplo de transportes ou de comunicações. Este problema surge quando se pretende determinar o caminho mais curto, mais barato ou mais fiável, entre um ou vários pares de nós de uma rede.

Estes problemas surgiram a partir de adaptações a uma grande variedade de problemas práticos, não só como modelos únicos mas também como subproblemas de problemas mais complexos. Por exemplo, surgiram nas indústrias de telecomunicações e de transportes sempre que se pretendia enviar uma mensagem, ou um veículo, entre dois locais geograficamente distantes, o mais rápido ou o mais barato possível.

O planeamento do tráfego urbano fornece um outro exemplo importante : os modelos utilizados para o cálculo do fluxo de tráfego padrão são problemas de optimização não linear, ou modelos de equilíbrio complexos. Contudo, a maioria das abordagens algorítmicas para determinar tráfegos urbanos padrão consiste, sob hipóteses simplificadoras, em resolver um grande número de problemas de caminho mais curto como subproblemas (um para cada par de nós origem–destino na rede) [1].

Existem três tipos de problemas de caminho mais curto :

- de um nó para outro,
- de um nó para todos os outros (árvore dos caminhos mais curtos),
- entre todos os pares de nós.

Os dois primeiros tipos são identificados como *problema de caminho mais curto de uma única origem* (ou apenas *caminho mais curto*) e o outro, como *problema de caminho mais curto entre todos os pares de nós*.

Sejam **s** e **t** dois nós de uma rede $G = (N, A, C)$, em que a cada arco está associado apenas um valor não negativo (*comprimento* do arco) — c_{ij} é o comprimrnto do arco (i, j). Seja c(p) o *comprimento* de um caminho **p** de **s** para **t** na rede G.

Com a resolução de um problema de caminho mais curto entre os nós **s** e **t**, na rede G, pretende-se determinar o caminho de valor mínimo existente em P; isto é, determinar $p \in P$, tal que $c(p) \leq c(q)$, para todo o $q \in P$ (P é o conjunto de todos os caminhos de **s** para **t**). Desta forma, pode-se formular este problema da seguinte forma :

$$\text{Minimizar} \quad Z = \sum_{i \in N} \sum_{j \in N} c_{ij}\, x_{ij}$$

sujeito a

$$\sum_{j \in N} x_{sj} = 1$$

$$\sum_{i \in N} x_{ij} - \sum_{k \in N} x_{jk} = 0, \quad \forall j \in N - \{ s, t \}$$

$$\sum_{i \in N} x_{it} = 1$$

em que,

$$x_{ij} = \begin{cases} 1, & \text{se (i, j) pertence ao caminho} \\ 0, & \text{se (i, j) não pertence ao caminho} \end{cases}$$

Algumas observações relacionadas com este tipo de problemas :

- o comprimento de um caminho é maior do que o de qualquer dos seus subcaminhos;
- qualquer subcaminho de um caminho mais curto, é ele próprio um caminho mais curto (princípio da optimalidade);
- para uma rede com **n** nós, qualquer caminho mais curto tem no máximo **n-1** arcos (num caminho não existem nós repetidos).

Para se analisar convenientemente a eficiência dos algoritmos, deve ser estudada a ordem de complexidade de cada um deles e compará-las. A ordem de complexidade de um algoritmo depende da quantidade de operações que ele executa, quando se considera o pior dos casos. Desta forma, a ordem de complexidade pode traduzir o tempo de execução do

algoritmo, para aquele caso. Portanto, a ordem de complexidade será o limite superior de tempo que é necessário para executar qualquer problema de caminho mais curto.

2. Algoritmos para resolver problemas de caminho mais curto

Existem vários algoritmos eficientes para resolver problemas de caminho mais curto com um só objectivo, sendo os mais conhecidos os algoritmos de Dijkstra, de Ford e de Floyd. Os dois primeiros aplicam-se a problemas de caminho mais curto de um nó inicial, **s**, para um nó final, **t** (ou para todos os outros), e baseiam-se num processo de rotulação dos nós; o último aplica-se a problemas de caminho mais curto entre todos os pares de nós.

2.1. Algoritmo de Dijkstra

Este algoritmo, que foi apresentado por Dijkstra, só pode ser aplicado a redes a cujos arcos estão associados apenas valores não negativos. Neste algoritmo, é suposto existir pelo menos um caminho entre **s** e qualquer outro nó.

O algoritmo baseia-se num processo de rotulação dos nós da rede e classificação dos respectivos rótulos. A cada nó **i** é atribuído um rótulo $[\xi_i, \pi_i]$ que pode ser permanente (fixo) ou temporário. Isto é,

$[\xi_i, \pi_i]$ permanente (o caminho mais curto de **s** para **i**)

$\xi_i \leftarrow$ nó que antecede **i** no caminho mais curto de **s** para **i**

$\pi_i \leftarrow$ comprimento do caminho mais curto de **s** para **i**

$[\xi_i, \pi_i]$ temporário (um caminho mais curto de **s** para **i**)

$\xi_i \leftarrow$ nó que antecede **i** no melhor caminho, até ao momento, de **s** para **i**

$\pi_i \leftarrow$ comprimento do melhor caminho, até ao momento, de **s** para **i**

$[\xi_i, \pi_i] = [-, \infty]$ indica que ainda não foi encontrada qualquer caminho de **s** para **i**.

O rótulo temporário de um nó representa um limite superior da distância mais curta de **s** para esse nó, uma vez que o caminho que lhe está associado pode ser ou não o mais curto.

O algoritmo consiste num processo de fixação dos rótulos dos nós da rede, começando pelo **s**, de uma forma ordenada segundo as distâncias de cada nó a **s**. Em cada iteração é escolhido o nó **i** com rótulo temporário com menor valor de π, que se torna permanente, para depois se varrerem todos os nós adjacentes a **i** (que tenham rótulos temporários), para que os seus rótulos sejam actualizados. O algoritmo termina quando não existirem nós com rótulos

temporários (caminho mais curto do nó **s** para todos os outros) ou quando o rótulo do nó **t** passar a permanente (caminho mais curto do nó **s** para o nó **t**).

Quando o algoritmo terminar, o comprimento do caminho mais curto entre **s** e **j** é π_j e o caminho determina-se percorrendo (em sentido inverso) a primeira parte dos rótulos (ξ) de **j** até **s**, da seguinte forma :

Caminho $\leftarrow$ { j }
i $\leftarrow$ j
Enquanto i $\neq$ s **Fazer**
    i $\leftarrow \xi_i$
    Caminho $\leftarrow$ Caminho $\cup$ { i }

A versão do algoritmo de Dijkstra que se descreve em Algoritmo 1 é a original e foi adaptada a partir de Ahuja et al. [1].

$[\xi_s, \pi_s] \leftarrow [s, 0]$ *(caminho mais curto de **s** para **s** tem comprimento 0 e **s** é o nó que antecede **s**)*
$[\xi_i, \pi_i] \leftarrow [-, \infty]$, $\forall$ i $\in$ N – { s } *(o caminho mais curto de **s** para **i** é desconhecido)*
R $\leftarrow$ N *(R = conjunto de nós com rótulo temporário)*
$\overline{R} \leftarrow \varnothing$ *($\overline{R}$ = conjunto de nós com rótulo permanente)*
Enquanto R $\neq \varnothing$ (ou $\overline{R} \neq$ N) **Fazer**
    k $\leftarrow$ nó de R tal que π_k é mínimo *($k : \pi_k = min \{ \pi_x \}, x \in R$)*
    R $\leftarrow$ R – { k } *(k deixou de ter rótulo temporário)*
    $\overline{R} \leftarrow \overline{R} \cup$ { k } *(k passou a ter rótulo permanente)*
    Para todo o j $\in$ N tal que (k, j) $\in$ A **Fazer**
        Se $\pi_k + c_{kj} < \pi_j$ **Então**
            $\pi_j \leftarrow \pi_k + c_{kj}$
            $\xi_j \leftarrow$ k

Algoritmo 1. Caminho mais curto : Dijkstra.

Para se contabilizar o número de operações que serão efectuadas neste algoritmo, pode-se fazer uma análise segundo duas componentes : selecção do nó com rótulo temporário mínimo e actualização dos rótulos. Para tal, considere-se uma rede orientada completa com **n** nós e **m** arcos (pior dos casos) — **m = n.(n–1)**.

Relativamente à primeira componente, são seleccionados **n** nós e a partir de cada um deles são varridos os (n–1) nós (associados aos (n–1) arcos que saem de cada nó), excepto aqueles que já têm rótulo permanente; ou seja, a partir do k-ésimo nó seleccionado são varridos n–k nós ((n–1) – (k–1)). Portanto, o número total de operações é : $n + (n–1) + (n–2) + \cdots + 1 = (n^2 + n) / 2 = O(n^2)$ (ou seja, da ordem de complexidade de n^2 operações).

Em relação à segunda componente, como cada arco só é analisado uma única vez, então esta operação é executada **m** vezes; logo, a ordem de complexidade é O(m).

Desta forma, a ordem de complexidade do algoritmo é de $O(n^2) + O(m) = \mathbf{O(n^2)}$ porque $m < n^2$. Este limite é o melhor que se pode obter em problemas que exijam redes completas ou muito densas para a representação dos seus dados no computador.

Relativamente ao espaço de memória necessário à execução deste algoritmo, é preciso armazenar uma matriz quadrada de ordem **n** e três vectores de dimensão **n**. Ou seja, são necessários, pelo menos, $\mathbf{n^2 + 3.n}$ espaços de memória para guardar informação — $\mathbf{O(n^2)}$.

2.2. Outras versões (mais eficientes) do algoritmo de Dijkstra

O facto das redes envolvidas na grande parte dos problemas reais serem esparsas, leva a que se possa utilizar versões mais eficientes deste algoritmo — as que exploram esta característica.

Pelo que se verificou em relação à ordem de complexidade do algoritmo de Dijkstra, a operação mais dispendiosa é a de selecção de nós. Desta forma, todos as versões que visem o melhoramento deste algoritmo têm que se debruçar sobre este aspecto.

Surgiram então diversas versões baseadas na ordenação dos rótulos (conjunto R), em função dos valores de π : sempre que o rótulo de um nó é alterado, a ordenação do conjunto terá que ser actualizada, pois pode acontecer que a ordem daquele nó também seja alterada. Por outro lado, é desnecessário considerar o conjunto dos nós com rótulos permanentes ($\overline{R}$), uma vez que é o complementar de R (em $\mathbb{N}$).

Portanto, das várias versões que surgiram, destacam-se as seguintes :

a) utilização de uma lista, em que os seus elementos são ordenados crescentemente segundo o valor de π e o elemento escolhido (associado ao nó, cujo rótulo passa a permanente) é o que se encontra à cabeça;

b) utilização de duas listas ligadas (lista duplamente ligada), em que os processos de ordenação e escolha são os mesmos do anterior, mas com esta representação torna-se mais rápido, em particular a ordenação;

c) utilização de tabelas de "hashing" (HEAP — lista de listas), em que os elementos que contém a mesma característica são colocados num mesmo grupo (lista) e em que estes grupos são ordenados pelas diferentes características (lista); depois, o elemento escolhido é o que se encontra à cabeça da primeira lista. Por exemplo, dado um valor C, é construída uma lista de C elementos, em que cada elemento contem uma lista de nós, colocados por qualquer ordem, cujo valor do resto da divisão inteira de π por C é igual : 0 (1ª lista), 1 (2ª lista), 2 (3ª lista), …, C–1 (C-ésima lista).

2.3. Algoritmo de Dijkstra Inverso

No algoritmo de Dijkstra normal, determina-se o caminho mais curto do nó **s** para qualquer outro nó $\mathbf{j} \in N - \{ s \}$. No entanto, se se pretender determinar o caminho mais curto de qualquer nó $\mathbf{j} \in N - \{ t \}$ para um nó final **t**, utiliza-se uma pequena modificação do algoritmo de Dijkstra, que consiste em manter a distância ρ_j associada a cada nó **j**, que é um limite superior para o comprimento do caminho mais curto do nó **j** para o nó **t**. Todo o restante processo é igual ao utilizado para o algoritmo normal.

2.4. Algoritmo de Ford

Ao contrário do algoritmo de Dijkstra, este algoritmo, que foi proposto por Ford, pode ser aplicado a redes cujos arcos têm associado valores negativos, não podendo contudo existir circuitos de valor negativo. Também neste algoritmo é suposto existir pelo menos um caminho entre **s** e qualquer outro nó.

Este algoritmo consiste em rotular os nós da rede, corrigindo-os as vezes que for necessário, até que todos os rótulos satisfaçam a seguinte condição (de optimalidade) :

$$\pi_j \leq \pi_i + c_{ij}, \text{ para todo o } (i, j) \in A,$$

em que π_i corresponde ao valor do caminho mais curto de **s** para **i** e c_{ij} ao valor do arco (i, j).

Neste algoritmo, os rótulos dos nós têm o mesmo significado dos rótulos utilizados no algoritmo de Dijkstra, só que agora os rótulos só se tornam definitivos após terminar a execução do algoritmo. Ou seja, nos estados intermédios do algoritmo apenas existem rótulos temporários.

Tal como acontece com o algoritmo de Dijkstra, também neste, após a conclusão da execução do algoritmo, o caminho mais curto entre os nós **s** e **j** determina-se percorrendo os rótulos (em sentido inverso) desde **j** até **s**.

Em Algoritmo 2, encontra-se a descrição da versão original do algoritmo de Ford, que foi adaptada a partir de Ahuja et al. [1].

$[\xi_s, \pi_s] \leftarrow [s, 0]$ *(caminho mais curto de **s** para **s** tem comprimento 0 e **s** é o nó que antecede **s**)*

$[\xi_i, \pi_i] \leftarrow [-, \infty]$, $\forall\ i \in N - \{ s \}$ *(o caminho mais curto de **s** para **i** é desconhecido)*

$R \leftarrow \{ s \}$

Enquanto $R \neq \varnothing$ **Fazer**

  $R \leftarrow R - \{ i \}$, para i qualquer

  Para todo o arco (i, j) **Fazer**

    Se $\pi_j > \pi_i + c_{ij}$ **Então**

      $\pi_j \leftarrow \pi_i + c_{ij}$

      $\xi_j \leftarrow i$

      Se $j \notin R$

      Então $R \leftarrow R \cup \{ j \}$

Algoritmo 2. Caminho mais curto : Ford.

Considere-se uma rede orientada completa com **n** nós e **m = n.(n–1)** arcos. Para se determinar a ordem de complexidade deste algoritmo, apenas é necessário contabilizar o número de iterações do ciclo, já que em cada iteração são efectuadas 2 operações (de actualização dos rótulos). O ciclo é executado no máximo **n.m** vezes, pois todos os nós e todos os arcos são analisados. Logo, a ordem de complexidade deste algoritmo é **O(n.m)**.

Relativamente ao espaço de memória, o algoritmo necessita de uma matriz quadrada de ordem **n** e de dois vectores de dimensão **n**. Portanto, no total são necessários $\mathbf{n^2 + 2.n}$ espaços de memória para guardar informação — $\mathbf{O(n^2)}$.

2.5. Outras versões (mais eficientes) do algoritmo de Ford

Surgiram posteriormente diversas versões relacionadas com a forma de manipular a lista dos nós sucessores dos arcos que são analisados, das quais se destacam as seguintes :

a) LIFO ("Last In First Out") : a inserção e a remoção de elementos é efectuada na cabeça da lista (o último nó inserido é o primeiro a ser removido) — Pilha (pesquisa em profundidade);

b) FIFO ("First In First Out") : a inserção de um elemento é efectuada pela cauda e a remoção pelo topo (o primeiro nó inserido é o primeiro a ser removido) — Fila (pesquisa por níveis);

c) DEQUEUE : as remoções fazem-se na cabeça da lista e as inserções na cabeça ou na cauda, conforme uma condição é ou não satisfeita (por exemplo : **se** π_j foi alterado **então**, se **j** já pertenceu à lista então j é inserido no topo senão j é inserido na cauda);

d) SCALING : a lista é manipulada como um "Dequeue", utilizando o seguinte critério para decidir se um elemento é inserido no topo ou na cauda : **se** a diminuição de π_j for maior que um dado δ **então**, se **j** já pertenceu à lista então **j** é inserido no topo senão **j** é inserido na cauda.

2.6. Algoritmo de Floyd

Ao pretender-se determinar o caminho mais curto entre todos os pares de nós, pode-se aplicar o algoritmo de Dijkstra (ou Ford) **n** vezes, utilizando cada nó, sucessivamente, como origem. No entanto, existem outros algoritmos para resolver este problema, como é o caso do algoritmo de Floyd, desde que não haja circuitos negativos; neste algoritmo, os arcos podem ter associados valores positivos ou negativos.

Um arco (i, j) designa-se por **arco básico** se constituir o caminho mais curto entre os nós **i** e **j**. Um caminho mais curto entre quaisquer dois nós da rede será totalmente constituído por arcos básicos (embora existam arcos básicos não pertencentes ao caminho mais curto).

O algoritmo de Floyd utiliza a matriz D, de ordem **n**, das distâncias directas mais curtas entre os nós. Os nós que não são adjacentes (não existe arco a ligá-los) têm associado uma distância directa infinita. Como os nós são (por convenção) adjacentes a eles próprios, têm associado um arco de comprimento 0 (zero). Os ciclos próprios são ignorados.

Entre cada par de nós não ligados por um arco básico é criado um arco, através de um processo identificado por "tripla ligação" :

$$d_{ij} \leftarrow \min \{ d_{ij} , \ d_{ik} + d_{kj} \}$$

Fixando um k, a "tripla ligação" é efectuada para todos os nós i, j ≠ k.

Após efectuar a "tripla ligação" para cada k ∈ N e i, j ∈ N – { k }, a rede (na matriz D alterada ao longo deste processo) é apenas constituída por arcos básicos. Logo, o valor associado a cada arco dirigido (i, k) é o caminho mais curto entre aqueles nós.

Para conhecer todos os nós intermédios num dado caminho, mantém-se paralelamente uma matriz P, de ordem **n**, onde o elemento P_{ij} representa o primeiro nó intermédio entre os nós **i** e **j**. No final deste algoritmo, a matriz D corresponde à matriz das distâncias mais curtas entre qualquer par de nós. Em Algoritmo 3 encontra-se a descrição deste algoritmo, que foi adaptado a partir de Ahuja et al. [1].

```
D ← matriz das distâncias directas
P_ij ← j , ∀ i, j ∈ N
Para k = 1, ..., n Fazer
    Para i, j = 1, ..., n tal que i, j ≠ k Fazer
        Se D_ij > D_ik + D_kj Então
            D_ij ← D_ik + D_kj
            P_ij ← P_ik
```

Algoritmo 3. Caminho mais curto : Floyd.

Para se determinar a ordem de complexidade deste algoritmo, considere-se uma rede completa, com **n** nós e **m** = n.(n–1) arcos (pior dos casos). Neste caso, a quantidade de operações que são efectuadas depende do número de iterações dos ciclos, uma vez que em cada iteração são efectuadas, no máximo, três operações (incluindo o teste). Portanto, para este caso, o ciclo externo é executado **n** vezes e o interno **2.(n–1)**, o que perfaz um total de $n.2.(n–1) = 2.n^2 – 2.n =$ **$O(n^2)$**.

Relativamente ao espaço de memória exigido, são necessárias duas matrizes quadradas de ordem **n**; logo, são precisos **$2.n^2$** espaços de memória para guardar informação — **$O(n^2)$**.

3. Árvore dos caminhos mais curtos

Quando se pretende determinar os caminhos mais curtos de um nó origem para todos os outros, está-se perante um problema de caminho mais curto. Para armazenar os dados referentes a estes caminhos, apenas é necessário utilizar (n–1) espaços de memória, em que **n** é quantidade nós da rede. Este resultado é consequência do facto de se poder determinar sempre uma árvore com raiz no nó origem, à qual pertencem todos os nós da rede para os quais exista caminho desde o nó origem, com a seguinte propriedade : o único caminho da origem para qualquer nó é o caminho mais curto para aquele nó.

Portanto, para se determinar a árvore dos caminhos mais curtos de um nó origem para todos os outros numa qualquer rede, basta aplicar um dos dois primeiros algoritmos de caminho mais curto, apresentados anteriormente.

4. O problema de determinar os k caminhos mais curtos

Este problema coloca-se quando é necessário conhecer, não só o caminho mais curto entre dois nós de uma rede, mas também o 2^o, o 3^o, ..., o k-ésimo caminho mais curto, tal que o valor do **k**-ésimo caminho é menor ou igual ao valor do **j**-ésimo caminho, com j > k. Pretende-se então determinar uma ordenação (crescente) dos caminhos mais curtos, os quais constituem caminhos mais curtos alternativos ao melhor caminho.

Os problemas de determinar os k caminhos mais curtos pode ser aplicado a casos em que se pretenda determinar o caminho mais curto, mas obedecendo a restrições complexas, em que a melhor solução pode não ser o caminho mais curto. Portanto, determinando-se sucessivamente os caminhos por ordem crescente dos seus comprimentos, pode-se chegar ao caminho mais curto que obedece às restrições impostas.

Em problemas de circulação de veículos interessa, muitas vezes, conhecer caminhos alternativos ao caminho mais curto, em virtude de existir congestionamento ou corte em qualquer troço daquele caminho.

Em problemas de encaminhamento de chamadas, pode interessar conhecer caminhos alternativos ao mais curto, por exemplo, quando alguma ligação deste caminho está saturada.

Um outro tipo de problemas que pode utilizar o algoritmo dos k caminhos mais curtos, está relacionado com o problema do caminho mais curto com múltiplos objectivos.

Formalmente, pode-se definir o problema de determinar os k caminhos mais curtos da seguinte forma :

Seja $G = (N, A, C)$ uma rede com **n** nós e **m** arcos. Considere-se $c : P \rightarrow \Re$ tal que $p \rightarrow c(p) = \sum_p c_{ij}$ é um valor real definido sobre P, que associa um valor a cada caminho de **s** para **t** em G. Dado um inteiro positivo k, pretende-se determinar um conjunto $P^{(k)} = \{ p_1, ..., p_k \} \subset P$ tal que

1) p_i é determinado antes de p_{i+1} , para qualquer $i \in \{ 1, ..., k-1 \}$

2) $c(p_i) \leq c(p_{i+1})$, para qualquer $i \in \{ 1, ..., k-1 \}$

3) $c(p_k) \leq c(q)$, para qualquer $q \in P - P^{(k)}$.

Existem alguns algoritmos para tratar este problema, dos quais serão referidos três deles, de acordo com a sua importância na evolução dos algoritmos :

- baseado no Princípio da Optimalidade — o clássico algoritmo de Dreyfus [12],
- baseado no aumento da rede — desenvolvido por Martins [17],
- baseado na determinação de nós desvios — desenvolvido por Martins et al. [20].

Enquanto que os dois primeiros determinam os k trajectos mais curtos, o último determina apenas caminhos (um caminho é um trajecto sem ciclos).

4.1. Algoritmo de Dreyfus

O objectivo deste algoritmo é determinar os k trajectos mais curtos entre um nó origem (assume-se que é o nó 1) e os restantes (n–1) nós da rede — Algoritmo 4.

Em Algoritmo 4 (em Rodrigues [21]), atenda-se à seguinte notação :

- $u_j[h]$ é o comprimento do h-ésimo trajecto mais curto do nó origem para o nó j;
- μ (i, j, h) é a quantidade de trajectos nos quais (i, j) é o arco final, no conjunto dos h trajectos mais curtos ($\{ 1^o, 2^o, ..., h\text{-ésimo} \}$) do nó origem para o nó j.

Neste algoritmo os nós **j** devem ser processados na ordem da quantidade de arcos existentes nos respectivos trajectos mais curtos a partir da origem, de modo a que o valor $u_i[h+1]$ seja conhecido na altura em que se pretenda calcular $u_j[h+1]$, se $(i, j) \in A$.

$u_1[1] \leftarrow 0$ (*por convenção, o nó origem é o 1*)

Para todos os arcos (i, j) **Fazer**

μ (i, j, 0) $\leftarrow 0$

Determinar a árvore dos caminhos mais curtos a partir do nó origem

Para os nós j = 2, 3, ..., n **e** i = 1, 2, ..., n, com i $\neq$ j **Fazer**

Se i é predecessor de j na árvore dos caminhos mais curtos **Então**

μ (i, j, 1) $\leftarrow \mu$ (i, j, 0) + 1

Senão

μ (i, j, 1) $\leftarrow \mu$ (i, j, 0)

Para h =1, ..., k **Fazer**

Para todo o nó j **Fazer**

Se (i, j) é o arco final do h-ésimo trajecto mais curto da origem até j **Então**

μ (i, j, h) $\leftarrow \mu$ (i, j, h–1) + 1

Senão

μ (i, j, h) $\leftarrow \mu$ (i, j, h–1)

(i, j) $\leftarrow$ último arco do trajecto mais curto da origem para j

$$u_j[h] \leftarrow \min_i \left\{ u_i\left[\mu(i, j, h-1)+1\right] + c_{ij} \right\}$$

Algoritmo 4. Determinar os k trajectos mais curtos : Dreyfus.

Em termos de complexidade, a quantidade de operações que este algoritmo efectua é da ordem de **O(k.n.log n)**, não contabilizando as operações associadas ao cálculo da árvore dos caminhos mais curtos, nem a atribuição de valores iniciais a u e μ.

Este algoritmo necessita de **k.n** espaços de memória para guardar a informação associada a u e m espaços para a associada a μ — em cada iteração é preciso guardar o valor de μ (i, j, h) para cada arco (i, j) — para além do espaço necessário para armazenar a rede. Portanto, o algoritmo necessita de um total de $\mathbf{n^2 + k.n + m}$ espaços de memória para armazenar informação — $\mathbf{O(n^2)}$.

4.2. Algoritmo baseado no aumento da rede

O desenvolvimento deste tipo de algoritmos deveu-se, principalmente, a Martins [17], cuja ideia inicial tem sofrido melhorias ao longo dos tempos. Este tipo de algoritmo consiste em aumentar a rede, sempre que se encontre um trajecto, em que esta alteração está directamente relacionado com o caminho que foi encontrado.

Considerando que se pretende determinar os k trajectos mais curtos entre os nós **s** (origem) e **t** (destino) na rede $G(N, A, C)$, o algoritmo começa por adicionar um "super nó origem" S, um "super nó destino" T e dois arcos de custo nulo, (S, s) e (t, T). Desta forma, em qualquer trajecto de S para T existe um subtrajecto de **s** para **t**, cujos comprimentos são iguais, pois (S, s) e (t, T) têm comprimentos nulos.

O primeiro passo do algoritmo consiste em determinar a árvore dos caminhos mais curtos do nó **s** para todos os outros, a partir da qual se determina o trajecto mais curto de **s** para **t**. Depois, e em cada iteração **h** do algoritmo, a rede é alterada adicionando mais nós (alternativos) e mais arcos (fictícios), sobre a qual é determinado o trajecto mais curto, que coincide com o (h+1)-ésimo trajecto mais curto da rede original.

Assim, para se determinar a sequência dos k trajectos mais curtos, $\{ p_1, p_2, ..., p_k \}$, o algoritmo constrói uma sequência de k nós $\{ t, t', t'', ..., t^{(k-1)'} \}$, em que cada um representa o nó terminal **t** e o nó alternativo ao trajecto mais curto. Além disso, o k-ésimo trajecto mais curto pode ser determinado desde $t^{(k-1)'}$ até **s**, andando para trás, substituindo todos os nós alternativos pelos respectivos nós originais.

Em Algoritmo 5 descreve-se a versão mais recente deste algoritmo (MS), que foi apresentada por Martins e Santos [19]. Para qualquer nó $x \in N$ considere a seguinte notação :

Pred(x) é o nó que antecede x em p — $(Pred(x), x) \in p$

Suc(x) é o nó sucessor de x em p — $(x, Suc(x)) \in p$

π_x é o valor da distância mais curta de **s** para **x**,

$I(x) = \{ (i, x) : (i, x) \in A \}$ é o conjunto dos arcos que chegam a **x**,

$T(x) = \{ i \in N : (i, x) \in I(x) \}$ é o conjunto dos nós cauda dos arcos de I(x),

x′ é o nó alternativo de x,

$\bar{x}$ é o nó da rede original que corresponde ao nó x,

$x^{(h)'}$ é o h-ésimo nó alternativo de x (x tem h nós alternativos) — $x^{(0)'} \equiv x$, $\forall\, x \in N$

Determinar a árvore dos caminhos mais curtos a partir de **s** em G

$p_1 \leftarrow$ caminho mais curto de **s** para **t** na árvore dos caminhos mais curtos

$h \leftarrow 1$

$p \leftarrow p_1$

Enquanto existe um trajecto alternativo a p **e** $h < k$ **Fazer**

$u \leftarrow$ o primeiro nó de p tal que a $\overline{u}$ chega mais do que um arco *{ 1ª fase }*

Se $u' \notin N$ **Então**

$(N, A) \leftarrow (N, A) \cup \{ u' \}$

$T(\overline{u}) = T(\overline{u}) - \{ Pred(u) \}$

$I(\overline{u}) = I(\overline{u}) - \{ (Pred(u), \overline{u}) \}$

$\pi_{u'} \leftarrow \min \{ \pi_x + dist(x, \overline{u}) : (x, \overline{u}) \in I(\overline{u}) \}$

$v \leftarrow Suc(u)$

Senão

$v \leftarrow$ primeiro nó que segue $u \in p$ tal que $u' \in N$

Para cada $w \in \{ v, ..., t^{(h-1)'} \}$ **Fazer** *{ 2ª fase }*

$(N, A) \leftarrow (N, A) \cup \{ w' \}$

$T(\overline{w}) = T(\overline{w}) - \{ Pred(w) \} \cup \{ Pred(w)' \}$

$I(\overline{w}) = I(\overline{w}) - \{ (Pred(w), \overline{w}) \} \cup \{ (Pred(w)', \overline{w}) \}$

$\pi_{w'} \leftarrow \min \{ \pi_x + dist(x, \overline{w}) : (x, \overline{w}) \in T(\overline{w}) \}$

$p \leftarrow$ trajecto mais curto de s para $t^{(h)'}$ em G

$h \leftarrow h + 1$

Algoritmo 5. Determinar os k trajectos mais curtos : baseado no aumento da rede (MS).

Em termos de complexidade, a quantidade de operações que esta versão do algoritmo efectua é da ordem de **O(k.m)**, não tendo em conta as operações associadas ao cálculo da árvore dos caminhos mais curtos.

Comparando com a versão original[1], a quantidade de operações efectuadas não foi melhorado. No entanto, relativamente ao espaço de memória necessário para o algoritmo ser

[1] A versão origial deste algoritmo encontra-se descrita em Martins e Santos [19].

executado, enquanto que a quantidade de operações na versão inicial é da ordem de **O(k.n + k.m)**, a última versão apenas é da ordem de **O(k.n + m)** — ou de **O(k.n)** se k > m/n.

Note-se que esta questão do espaço de memória é muito importante, já que permite a sua aplicação a problemas que envolvam redes muito grandes.

Comparando a versão aqui apresentada (a última) com a original, verifica-se que enquanto na versão original as quantidades de nós e de arcos sofrem um significativo aumento, uma vez que quando se adiciona um nó alternativo x', adicionam-se tantos arcos quantos os que chegam a x menos 1 (se $(y, x) \in A$ e $(y, x) \notin p$ então (y, x') é adicionado à rede), na versão aqui apresentada, apenas o número de nós é aumentado, já que o número de arcos pode ser mantido (ou até ligeiramente decrementado), pois sempre que se adiciona um nó alternativo x', retira-se o arco de **p** que chega a x e, ou não se adiciona qualquer arco (1ª fase) ou apenas se adiciona um arco que liga o nó alternativo de Pred(x) a x (2ª fase).

4.3. Algoritmo baseado na determinação de nós desvios

Este algoritmo, que foi apresentado por Martins et al. [20], utiliza a mudança de variável proposta por Eppstein nos seus algoritmos para problemas sem restrições e o conceito de nó desvio de um trajecto.

Em termos gerais, a iteração **h** deste algoritmo consiste em determinar os desvios que podem dar origem a caminhos, os quais são calculados a partir do h-ésimo caminho mais curto, que foi encontrado nesta versão.

A mudança de variável é apresentada no seguinte resultado : seja ρ_i a distância mais curta de **i** para **t** e $\bar{c}_{ij} = \rho_j - \rho_i + c_{ij}$, $\forall (i, j) \in A$. Então, $\bar{c}(p) = \sum_p \bar{c}_{ij} = \rho_t - \rho_s + c(p)$, $\forall p \in P$.

Para resolver o problema da árvore dos caminhos mais curtos de cada nó $\mathbf{i} \in N - \{ t \}$ para **t**, utiliza-se o algoritmo de Dijkstra Inverso (ver secção 2.3 deste capítulo).

Por outro lado, o conjunto de arcos que saem de cada nó **i** tem de ser ordenado por ordem decrescente dos valores de $\bar{c}_{ij}$, de tal modo que o último arco pertença ao caminho mais curto de **i** para **t**.

Considere-se a seguinte definição de nó desvio de um caminho. Seja **q** o caminho mais curto de **s** para **i** na árvore de todos os trajectos já determinados. Para todo o k > 1, o nó **i** é o nó desvio de p_k se $q \subset p_k$ e for o subcaminho com o maior número de nós tal que $q \subset p_h$, com $h \in \{ 1, ..., k-1 \}$. Por conveniência, o nó **s** é o nó desvio de p_1. Considere-se o seguinte

exemplo : p_1 = [1, 2, 3, 4, 5], p_2 = [1, 2, 4, 5], p_3 = [1, 2, 3, 5] e p_4 = [1, 2, 4, 3, 5] são caminhos de uma rede qualquer. O nó desvio de p_4 é o nó 3, pois q = [1, 2, 3] é o subcaminho com origem em 1 com o maior número de nós entre todos os três caminhos dados.

Em Algoritmo 6 descreve-se a versão mais recente deste algoritmo (MPS), que foi apresentada por Martins et al. [20].

Determinar a árvore dos caminho mais curto para **t** em G

$\bar{c}_{ij} \leftarrow \rho_j - \rho_i + c_{ij}$, para todo $(i, j) \in A$

$A \leftarrow \{ 1, ..., m \}$ tal que $\forall\, i \in \{ 1, ..., m \}$, $\bar{c}_i \leq \bar{c}_{i+1}$ se cauda(i) = cauda(i+1)

$X \leftarrow \{ p_1 \}$

$h \leftarrow 0$

Enquanto $(X \neq \varnothing)$ **e** $(h \leq k)$ **Fazer**

  $p \leftarrow$ trajecto de X tal que $\bar{c}(p) \leq \bar{c}(q)$ para qualquer $q \in X$

  Se p não tem ciclos **Então**

    $h \leftarrow h + 1$

    $p_h \leftarrow p$

  $i \leftarrow$ nó desvio de p

  Repetir

    $p_i \leftarrow$ subcaminho de p de **s** para **i**

    $b \leftarrow$ arco de p cuja cauda é o nó **i**

    Enquanto (i é a cauda de b+1) **e** (b+1 forma um ciclo com p_i) **Fazer**

      $b \leftarrow b + 1$

    Se (i é a cauda de b+1) **Então**

      $j \leftarrow$ cabeça de b + 1

      $\bar{p}_j \leftarrow$ caminho mais curto de **j** para **t**

      $X \leftarrow X \cup \{ p_i \lozenge [i, j] \lozenge \bar{p}_j \}$ ($\lozenge$ = *concatenação*)

    $i \leftarrow$ próximo nó de p

  Até (p_i ser um ciclo) **Ou** (i = t)

Algoritmo 6. k caminhos mais curtos : baseado no cálculo de nós desvios (MPS).

C A P Í T U L O 4

O Problema do Caminho Mais Curto com Múltiplos Objectivos

1. Formulação do problema

Considere-se uma rede $G = (N, A, C)$ com **n** nós e **m** arcos, em que $C = (c^1, c^2, ..., c^h)$ com $h \geq 1$ corresponde aos valores numéricos não negativos associados a cada arco. Ou seja, ao arco (i, j) estão associados os valores $c_{ij} = (c_{ij}^1, \ldots, c_{ij}^h)$.

Sejam **s** e **t** dois nós da rede G. Os valores numéricos associados ao caminho **p** de **s** para **t** na rede G formam um h-uplo de somas, em que cada uma delas corresponde à soma dos valores associados aos arcos que pertencem àquele caminho. Ou seja,

$$c(p) = \left(c^1(p), c^2(p), \ldots, c^h(p)\right) \text{ em que } c^w(p) = \sum_{(i,j) \in p} c_{ij}^w, \quad w \in \{1, 2, \ldots, h\}.$$

O problema do caminho mais curto com vários (h) objectivos entre os nós **s** e **t** da rede G, pode ser formulado da seguinte forma :

$$\text{Min} \quad Z = (Z_1, ..., Z_h)$$

sujeito a

$$\sum_{j \in N} x_{sj} = 1$$

$$\sum_{i \in N} x_{ij} - \sum_{l \in N} x_{jl} = 0, \qquad \forall j \in N - \{s, t\}$$

$$\sum_{i \in N} x_{it} = 1$$

$$x_{ij} \in \{0, 1\}, \quad \forall (i, j)$$

onde,

$$Z_1 = \sum_{i \in N} \sum_{j \in N} c_{ij}^1 \, x_{ij}$$

. . .

$$Z_h = \sum_{i \in N} \sum_{j \in N} c_{ij}^h \, x_{ij}$$

$$x_{ij} = \begin{cases} 1, & \text{se o arco (i, j) pertence ao caminho} \\ 0, & \text{caso contrário} \end{cases} .$$

2. Tipo de soluções do problema

Nos problemas de programação linear com vários objectivos, o conjunto das soluções admissíveis eficientes é um conjunto convexo contínuo, denominado por politopo convexo (quando limitado tem o nome de poliedro convexo), o qual é construído à custa das restrições do problema. Desta forma, qualquer solução vértice eficiente pode ser determinada optimizando uma combinação convexa das funções objectivo (as soluções não vértices na fronteira eficiente podem depois ser caracterizadas à custa dos vértices).

No entanto, como os problemas de caminho mais curto multiobjectivo são de natureza discreta, a fronteira eficiente não é um conjunto convexo contínuo, podendo existir soluções não dominadas que não estejam situadas na fronteira do invólucro convexo (***contorno convexo***), mas sim no seu interior, nos chamados ***desníveis de dualidade*** ("duality gaps"). Às soluções não dominadas que pertencem ao contorno convexo, dá-se o nome de **soluções suportadas**, as quais podem ainda ser **extremas** (ou **vértices**) ou **não extremas**. Às soluções que se encontram nos desníveis de dualidade, dá-se o nome de **soluções não suportadas**.

Seja $Z^{\geq}$ o invólucro convexo de $[Z \oplus \{ z \in \Re^h : z \geq 0 \}]$ (em que $\oplus$ representa a adição de conjuntos e h o número de objectivos do problema).

1. Seja $z \in Z$ uma solução não dominada. Se z pertence à fronteira de $Z^{\geq}$ então z é suportada; caso contrário, z é não suportada (é dominada por uma combinação convexa de outras soluções não dominadas suportadas).
2. Seja $z \in Z$ uma solução não dominada suportada. Se z não puder ser expressa como combinação convexa de outras soluções suportadas, então z é extrema; caso contrário, z é não extrema [25]. Ou seja, z é extrema se e só se existir $\alpha_k > 0$ com k = 1, ..., h, tal que

$$\sum_{k=1}^{h} \alpha_k \, z_k < \sum_{k=1}^{h} \alpha_k \, w_k \, , \; \forall \, w \in Z \; : \; z = (z_1, ..., z_h) \neq (w_1, ..., w_h) = w \; [15].$$

O gráfico da Fig. 1 exemplifica os conceitos de solução não dominada suportada (extrema e não extrema) e não suportada :

a) as soluções 1, 2 e 4 são não dominadas suportadas extremas (vértices),

b) a solução 3 é não dominada suportada não extrema (é uma combinação convexa das soluções 2 e 4, pois encontra-se representada sobre a aresta $\overline{24}$),

c) as soluções 5 e 6 são não dominadas não suportadas (dominadas pelas combinações convexas de 2 com 3 e de 3 com 4, respectivamente).

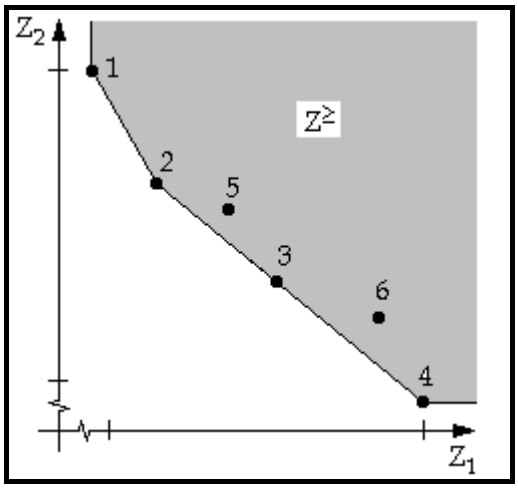


Fig. 1 – Tipo de soluções não dominadas : suportadas e não suportadas.

As imagens inversas das soluções não dominadas suportadas denominam-se por pontos eficientes suportados (no espaço das decisões) e das não dominadas não suportadas por pontos eficientes não suportados [25].

Os algoritmos que têm sido propostos até agora utilizam processos diferentes para determinar soluções não dominadas, consoante sejam suportadas extremas ou suportadas não extremas e não suportadas. No entanto, na pesquisa daquelas soluções, utiliza-se sempre o problema do caminho mais curto com um só objectivo.

O processo para determinar as soluções não dominadas suportadas extremas consiste em resolver um problema de caminho mais curto com um só objectivo, em que a função objectivo é uma combinação convexa das p funções objectivo do problema multiobjectivo.

Para se determinar as soluções não dominadas não suportadas têm sido apresentados, ao longo dos tempos, alguns processos diferentes, mas todos exigindo um elevado esforço computacional.

3. Um problema real de caminho mais curto com múltiplos objectivos

Muitos problemas reais podem ser modelados como um problema de caminho mais curto multiobjectivo, em que os dados do modelo são organizados recorrendo a uma estrutura em rede.

Como exemplo, veja-se um caso de circulação de veículos, mencionado em Rodrigues [21]. Pretende-se determinar o melhor caminho entre dois nós de uma rede de estradas (por exemplo, entre um ponto onde se carregam componentes que devem ser transportadas para uma fábrica localizada num outro ponto da rede), tendo em conta um compromisso entre dois objectivos : distância percorrida e tempo gasto. O tempo gasto num percurso pode ser influenciado por atrasos na entrega da mercadoria, uma vez que o tempo de circulação em cada artéria (arco da rede) pode variar muito, devido a problemas de congestão de tráfego, acidentes, obras de manutenção, filas de espera para pagamentos de portagens, etc..

Geralmente, pretende-se determinar uma solução cujos valores associados a cada objectivo se encontrem o mais próximo possível dos valores respectivos mínimos, de modo a rentabilizar o lucro da empresa responsável pela entrega das componentes a transportar. Neste caso, está-se perante um problema de caminho mais curto bi-objectivo.

Ao acrescentar-se mais um objectivo, por exemplo o número de artérias percorridas (ou quantidade de nós visitados), o problema transforma-se num problema de caminho mais curto tri-objectivo, se se pretender uma solução em que o número de nós a visitar seja também a menor possível.

Se se acrescentar o nível de perigosidade em termos de acidentes rodoviários (relacionado com o número de acidentes nas artérias), o problema transforma-se num de caminho mais curto tetra-objectivo, pois pretende-se uma solução cujo valor associado a este objectivo seja também o menor possível.

4. O método NISE

Uma das abordagens mais importantes no âmbito dos problemas de caminho mais curto com vários objectivos é o método NISE ("**N**on**I**nferior **S**olutions **E**stimation"), que foi proposto por Cohon [9]. Este método destina-se a determinar uma aproximação do conjunto de soluções não inferiores e foi descrito, sobretudo, para problemas com duas funções objectivo.

Suponha-se que se pretende resolver um problema de caminho mais curto com dois objectivos, em que a função objectivo pode ser expressa da seguinte forma :

$$\text{Minimizar} \quad Z = (Z_1, Z_2) .$$

As soluções não dominadas são determinadas utilizando o método da soma pesada das funções objectivo, reduzindo, desta forma, o problema original a um com apenas uma função objectivo. Para tal, efectua a soma das duas funções depois de se ter multiplicado uma delas por um parâmetro (peso) positivo w, ficando :

$$\text{Minimizar} \quad Z(w) = w.Z_1 + Z_2 \ .$$

Desta forma, se o valor atribuído a w for superior a 1 então está-se a favorecer o primeiro objectivo em detrimento do segundo, acontecendo o inverso se w for inferior a 1, pois o peso que se está a atribuir ao segundo objectivo é constante e igual a 1.

Pode-se generalizar a utilização de pesos, atribuindo pesos a ambas as funções objectivo, obtendo-se :

$$\text{Minimizar} \quad Z(w_1, w_2) = w_1.Z_1 + w_2.Z_2 .$$

No entanto, se se dividir as duas funções por um mesmo valor positivo (por exemplo, w_2), obtém-se a equação anterior, fazendo $w = w_1/w_2$. Desta forma, conclui-se que neste tipo de problemas o que interessa efectivamente é a relação entre os pesos atribuídos aos objectivos e não o seu valor absoluto.

O método NISE utiliza, em cada iteração, o seguinte critério para seleccionar os pesos : escolher valores de tal modo que a próxima solução esteja o mais distante possível da recta que passa pelas duas soluções previamente escolhidas.

O método começa por determinar as duas soluções não dominadas (vértices) que optimizam (minimizam) isoladamente cada uma das funções objectivo : as soluções A e B que minimizam Z_1 e Z_2, respectivamente (Fig. 2).

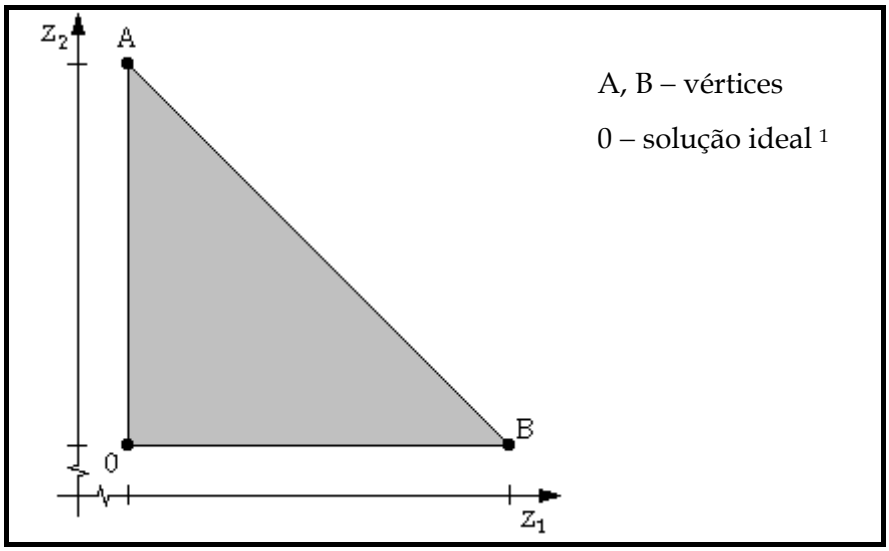


Fig. 2 – Método NISE : zona de pesquisa de soluções não dominadas (vértices).

A próxima solução não dominada a ser determinada, se existir, deve ser de tal modo que a sua representação deve situar-se o mais distante possível do segmento $\overline{AB}$, mas no interior do triângulo $\Delta(0,A,B)$. Como facilmente se pode verificar, não existem soluções à esquerda do segmento $\overline{0A}$, pois se existisse alguma solução nessa zona ela teria sido encontrada quando se optimizou a função objectivo Z_1 isoladamente. Da mesma maneira, se conclui que não podem existir soluções abaixo do segmento $\overline{0B}$.

Por outro lado, o segmento $\overline{AB}$ servirá de base ao cálculo da próxima solução não dominada. Com efeito, aplicando a regra utilizada pelo método NISE para o cálculo dos pesos w_1 e w_2, estes são tais que

$$-\frac{w_1}{w_2} = \alpha ,$$

em que α é o declive da recta que contém o segmento $\overline{AB}$. A direcção da busca é perpendicular ao segmento $\overline{AB}$. Desta forma, qualquer nova solução admissível determinada por este método, só poderá situar-se no interior do triângulo $\Delta(0,A,B)$.

[1] Denomina-se por **solução ideal** aquela que optimizaria simultaneamente todas as funções objectivo (os seus valores obtêm-se considerando os valores das soluções que optimizam separadamente cada função objectivo), a qual é não admissível sempre que as funções objectivo estejam em conflito.

Como no cálculo dos pesos apenas interessa a relação entre eles, e sendo conhecido o declive da recta que contém o segmento $\overline{AB}$, basta igualar ao valor unitário um dos pesos para se calcular o outro. Neste caso, o declive da referida recta é dado por :

$$\alpha = \frac{Z_2(B) - Z_2(A)}{Z_1(B) - Z_1(A)}.$$

Como este declive é negativo (numerador negativo e denominador positivo) e os pesos devem ser escolhidos de forma que $-(w_1/w_2) = \alpha$, o quociente entre os pesos será um valor positivo. Isto é,

$$-\frac{w_1}{w_2} = \alpha = \frac{Z_2(B) - Z_2(A)}{Z_1(B) - Z_1(A)} = -\frac{Z_2(A) - Z_2(B)}{Z_1(B) - Z_1(A)} \Rightarrow \begin{cases} w_1 = Z_2(A) - Z_2(B) \\ w_2 = Z_1(B) - Z_1(A) \end{cases}.$$

Uma vez que é a relação entre os pesos que interessa, fazendo $w_2 = 1$, vem :

$$w_1 = \frac{Z_2(A) - Z_2(B)}{Z_1(B) - Z_1(A)}.$$

Resolvendo o problema cuja função objectivo é :

Minimizar $Z(A, B) = w_1.Z_1 + Z_2$

seria determinada a solução C e obter-se-ia uma nova aproximação para a região das soluções admissíveis não dominadas, zona a cheio na Fig. 3 — o triângulo $\Delta(0,0',0'')$ corresponde à região onde não se encontram soluções admissíveis, com o segmento $\overline{0'0''}$ paralelo a $\overline{AB}$.

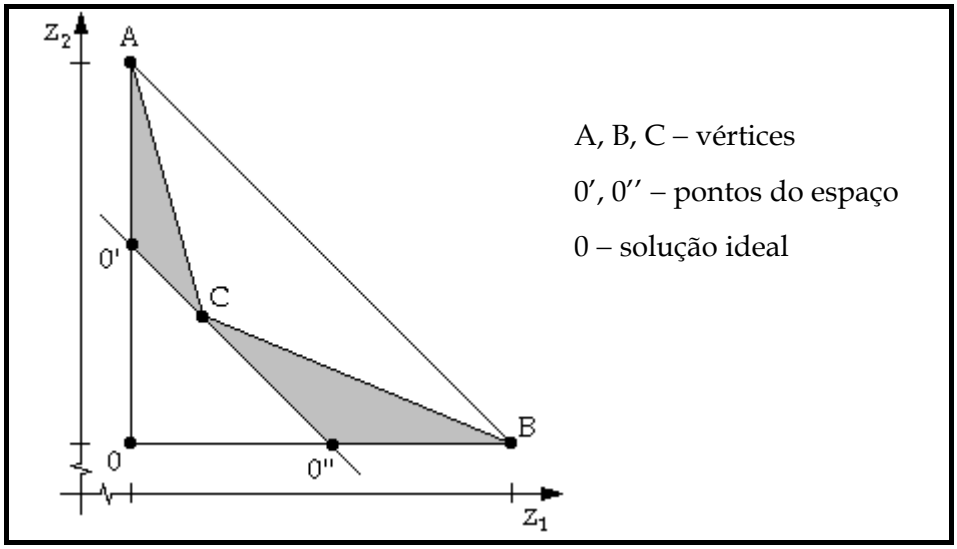


Fig. 3 – Método NISE : determinação de um vértice a partir de dois outros.

Se se prosseguir com o método, são possíveis duas novas direcções de busca : uma perpendicular ao segmento $\overline{AC}$, que percorrerá o triângulo $\Delta(0',A,C)$, e outra perpendicular ao segmento $\overline{BC}$, que percorrerá o triângulo $\Delta(0'',B,C)$.

Em termos genéricos, em cada iteração os valores dos pesos são os seguintes :

$$w_1 = \frac{Z_2(i) - Z_2(i+1)}{Z_1(i+1) - Z_1(i)}$$

$$w_2 = 1$$

em que **i** corresponde à i-ésima solução não dominada suportada extrema, tendo em conta que estas estão numeradas pela ordem decrescente do respectivo valor de Z_2 (a solução A é a que tem o maior valor de Z_2).

O método só termina, quando o máximo erro possível for inferior a determinado valor preestabelecido. Pela Fig. 3 pode-se verificar que as extensões das novas buscas estão relacionadas com as alturas dos triângulos (considera-se como bases dos triângulos os segmentos $\overline{AC}$ e $\overline{BC}$, para cada caso); o erro máximo corresponde à maior altura.

5. O problema de caminho mais curto bi-objectivo

Nesta secção serão descritos alguns métodos para determinar soluções não dominadas suportadas extremas e não suportadas, assim como algumas abordagens interactivas para ajudar o AD a identificar a "melhor" solução de compromisso em problemas de caminho mais curto bi-objectivo.

5.1. Formulação do problema

Considere-se uma rede $G = (N, A, C)$ com **n** nós e **m** arcos, em que $C = (c^1, c^2)$. Ou seja, a cada arco (i, j) estão associados 2 valores não negativos, que correspondem ao "comprimento" do arco.

Sejam **s** e **t** dois nós da rede G. Os valores associados ao caminho **p** de **s** para **t** na rede G, são um par de somas, em que cada uma delas corresponde à soma dos valores associados aos arcos que pertencem àquele caminho; ou seja,

$$c(p) = (c^1(p), c^2(p)), \text{ em que, } c^1(p) = \sum_{(i,j) \in p} c_{ij}^1 \text{ e } c^2(p) = \sum_{(i,j) \in p} c_{ij}^2 .$$

Seja P_i o conjunto de todos os caminhos do nó **i** para o nó **t**, em que $P_i \neq \emptyset$, $\forall\, i \in N$. Seja ainda P o conjunto de todos os caminhos de **s** para **t**. O problema do caminho mais curto com dois objectivos, entre os nós **s** e **t**, pode ser formulado da seguinte forma :

[P1] Min $Z = (Z_1, Z_2)$ **(5.1)**

sujeito a

$$\sum_{j \in N} x_{sj} = 1 \qquad \textbf{(5.2)}$$

$$\sum_{i \in N} x_{ij} - \sum_{k \in N} x_{jk} = 0, \quad \forall j \in N - \{s, t\} \qquad \textbf{(5.3)}$$

$$\sum_{i \in N} x_{it} = 1 \qquad \textbf{(5.4)}$$

$$x_{ij} \in \{ 0, 1 \}, \quad \forall\, (i, j) \qquad \textbf{(5.5)}$$

onde,

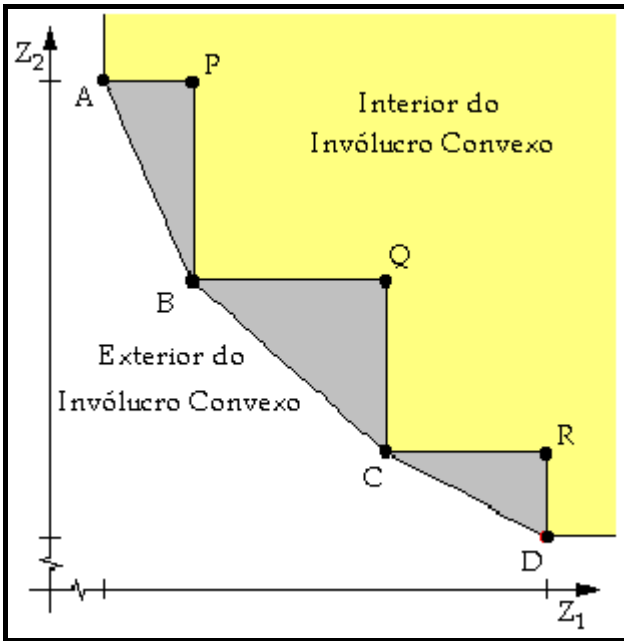
$$Z_1 = \sum_{i \in N} \sum_{j \in N} c_{ij}^1 \, x_{ij}$$

$$Z_2 = \sum_{i \in N} \sum_{j \in N} c_{ij}^2 \, x_{ij}$$

$$x_{ij} = \begin{cases} 1, & \text{se o arco } (i, j) \text{ pertence ao caminho} \\ 0, & \text{caso contrário} \end{cases} .$$

5.2. Representação do espaço dos objectivos

Todas as soluções admissíveis formam um conjunto convexo que se chama polígono convexo, uma vez que este conjunto é limitado (politopo convexo limitado). Desta forma, estas soluções encontram-se na sua fronteira (suportadas) e no seu interior (não suportadas). No entanto, como mostra a Fig. 4, enquanto que todas as soluções suportadas são não dominadas, as não suportadas são dos dois tipos. De facto, analisando a Fig. 4, verifica-se que qualquer solução que se encontre na região mais clara do invólucro convexo (solução não suportada) é dominada por soluções suportadas (por exemplo, a solução C domina qualquer solução que se encontre à direita da recta que contém o segmento $\overline{CQ}$ e acima da recta que contém o segmento $\overline{CR}$).



Δ(A,B,P), Δ(B,C,Q), Δ(C,D,R) – zonas onde se podem localizar soluções não dominadas não suportadas

A, B, C, D – soluções não dominadas suportadas

P, Q, R – pontos do espaço (geralmente não correspondem a qualquer solução)

Fig. 4 – Bi-objectivo : representação das soluções no espaço dos objectivos.

Uma ***zona de desnível de dualidade*** é definida por dois vértices adjacentes e por um ponto formada pelos valores máximos das duas funções objectivo, relativamente àquelas soluções, formando um triângulo (por exemplo, Δ(A,B,P), Δ(B,C,Q) e Δ(C,D,R) formam três zonas de desníveis de dualidade na Fig. 4).

5.3. Métodos para determinar soluções não dominadas

Um processo usual é gerar todo o conjunto de soluções não dominadas para que o AD possa *a posteriori* seleccionar a "melhor" solução existente naquele conjunto. No entanto, esta abordagem torna-se complicada, pois este conjunto pode ser muito extenso e, portanto, muito difícil de determinar.

Uma outra forma de determinar as soluções não dominadas, consiste em identificar, em fases distintas e com processos diferentes, as soluções não dominadas suportadas extremas e as não suportadas. Podem, então, considerar-se dois tipos de métodos para determinar estas soluções, cada um correspondendo a cada tipo de soluções.

5.3.1. Métodos para determinar soluções não dominadas suportadas extremas

Nesta secção serão descritos quatro métodos para determinar soluções não dominadas suportadas extremas, dos quais três foram apresentados em 1985 por Henig [15].

5.3.1.1. Método de Henig adaptado a partir de Shin e Hartley

Este método consiste em gerar uma sequência finita $\{ \beta_k \}$, $1 = \beta_1 \geq \beta_k > \beta_{k+1} \geq 0$ em paralelo com a lista $\{ (c_k^1, c_k^2) \}$, onde (c_k^1, c_k^2) é o valor do vértice associado a cada $\beta \in (\beta_k, \beta_{k+1})$. Este método é uma adaptação do utilizado no algoritmo desenvolvido por Shin [24] e Hartley [14], para um modelo de programação dinâmica estocástica.

Começa-se por determinar o caminho $p^* \in P$, tal que $c_1^1 = c^1(p^*) \leq c^1(p)$ para todo o $p \in P$ e $c_1^2 = c^2(p^*) \leq c^2(p)$ com $p \in P$ e $c^1(p^*) = c^1(p)$. Este *caminho* é o *lexicograficamente mais curto* relativamente a $\beta_1 = 1$ e pode ser determinado aplicando duas vezes um algoritmo de caminho mais curto : uma para determinar todos os caminhos mais curtos relativamente ao primeiro objectivo e em seguida, entre todos aqueles caminhos, determinar o mais curto relativamente ao segundo objectivo.

Depois, determina-se a árvore dos caminhos mais curtos de qualquer nó para **t**, relativamente ao primeiro objectivo ($c^1(p_j)$, $j \in N$) e, naquela árvore, determinam-se também os valores dos caminhos mais curtos para **t** em relação ao segundo objectivo ($c^2(p_j)$, $j \in N$).

Dado β_k suponhamos que o valor do caminho mais curto de **j** para **t**, associado a β_k é

$(c^1(p_j), c^2(p_j))$.

Seja

$$\alpha = \min \frac{-(c_{ij}^1 + c^1(p_j) - c^1(p_i))}{(c_{ij}^2 + c^2(p_j) - c^2(p_i))} \qquad \textbf{(5.6)}$$

em que o mínimo é considerado sobre todos os $(i, j) \in A$, tal que o denominador é negativo.

Seja (i^*, j^*) o arco associado ao valor mínimo da equação (5.6) e actualize-se a árvore dos caminhos mais curtos para **t**, substituindo o arco com origem em i^* pelo arco (i^*, j^*). Desta forma, obtêm-se os caminhos mais curtos até **t** associados a

$$\beta_{k+1} = \frac{1}{1+\alpha}.$$

Este processo continua até

$$c_{ij}^2 + c^2(p_j) - c^2(p_i) \geq 0, \ \forall (i, j) \in A$$

indicando que foi determinado o caminho lexicograficamente mais curto associado a $\beta = 0$.

5.3.1.2. Método de Henig semelhante ao de Denardo

Este método consiste em gerar uma sequência finita $\{ \beta_k \}$ com $0 = \beta_0 < \beta_k < \beta_1 = 1$ e a lista que lhe é associada $\{ (c_k^1, c_k^2) \}$ dos valores dos caminhos não dominados extremos. Este método assemelha-se ao utilizado por Denardo [11] para determinar um multiplicador de Lagrange, numa rede com uma restrição adicional.

Inicialmente, determinam-se os caminhos lexicograficamente mais curtos de **s** para **t**, associados a $\beta_0 = 0$ e $\beta_1 = 1$. Estes valores denominam-se por (c_0^1, c_0^2) e (c_1^1, c_1^2).

Seja $L = \{ (c_0^1, c_0^2), (c_1^1, c_1^2) \}$ e $S = \varnothing$. Na iteração principal, calcula-se

$$\alpha = \frac{c_{\beta 2}^1 - c_{\beta 1}^1}{c_{\beta 1}^2 - c_{\beta 2}^2}$$

onde $(c_{\beta 1}^1, c_{\beta 1}^2)$ e $(c_{\beta 2}^1, c_{\beta 2}^2)$ são os dois primeiros elementos de L.

Seja (c_β^1, c_β^2) o valor do caminho mais curto associado a $\beta = 1/(1+\alpha)$. Se $\beta \times c_\beta^1 + (1-\beta) \times c_\beta^2 = \beta \times c_{\beta 1}^1 + (1-\beta) \times c_{\beta 1}^2$, então transfere-se $(c_{\beta 1}^1, c_{\beta 1}^2)$ de L para S. Caso contrário, insere-se (c_β^1, c_β^2) em segundo lugar na lista L. Em ambos os casos, a iteração principal é repetida até L ser singular. Assim, $S \cup L$ contém os valores dos vértices do problema.

5.3.1.3. Método de Henig que utiliza a operação Ext

Seja $\mathbf{G_i}$ o conjunto de valores dos caminhos não dominados suportados extremos de **i** para **t**. Note-se que se o arco (i, j) e o caminho $p_j \in P_j$ formam um caminho mais curto de **i** para **t**, associado a β, então p_j é um caminho mais curto de **j** para **t**, associado a β. Isto verifica a seguinte equação :

$$G_i = \text{Ext} \{ \overline{G}_i \}, \; i \in N, \; G_t = \{ (0, 0) \}$$

onde

$$\overline{G}_i = \{ (c_{ij}^1, c_{ij}^2) + G_j \; : \; j \in N \}.$$

A operação **Ext** refere-se à escolha dos pontos não dominados extremos, que pertencem ao contorno convexo de $\overline{G}_i$.

O processo consiste em determinar os conjuntos G_i, com $i \in N$, em que **i** varia entre **t** e **s**. No fim, G_s contém os valores dos caminhos não dominados extremos do problema.

5.3.1.4. Versão do método NISE apresentada por Cohon

Esta versão, que foi apresentada por Cohon [9] em 1978, resolve, em cada iteração, um problema de caminho mais curto com um único objectivo, cuja função objectivo é uma combinação convexa das funções objectivo originais. Desta forma, cada solução determinada é não dominada para o problema P1. Portanto, este método consiste na resolução do seguinte problema (que é P1 modificado) :

$$\text{Min} \quad Z = w_1.Z_1 + w_2.Z_2 \qquad \textbf{(5.7)}$$

sujeito a

(5.2) – (5.5)

em que

$$w_1 = \left| \frac{Z_2(A) - Z_2(B)}{Z_1(A) - Z_1(B)} \right|$$

$$w_2 = 1$$

$Z_i(x)$ é o valor do i-ésima função objectivo associado à solução x

Z(A) e Z(B) são soluções não dominadas adjacentes.

Em cada iteração do método é calculado o erro máximo para a aproximação, o qual é utilizado para determinar o esquema de ponderação a ser usado para determinar a próxima solução não inferior. Por outro lado, pode-se controlar a exactidão da aproximação, através de um critério de erro predeterminado.

5.3.2. Métodos para determinar soluções não dominadas não suportadas

Se para determinar as soluções não dominadas suportadas existem alguns métodos, para determinar as não suportadas já isso não acontece. De facto, existem muito poucos métodos para determinar estas últimas soluções, tornando-se a sua identificação o "calcanhar de Aquiles" dos problemas de caminho mais curto com vários objectivos.

Apesar disto, será feito referência a dois métodos : um que se baseia no método NISE com restrições e o outro no problema dos k caminhos mais curtos.

5.3.2.1. Método NISE com restrições adicionais

O método NISE, como foi apresentado antes, apenas consegue determinar as soluções não dominadas suportadas extremas. Desta forma, este método consegue determinar as soluções A, B, C e D, mas não as soluções que possam existir nos interiores dos triângulos Δ(A,B,P), Δ(B,C,Q) e Δ(C,D,R) — Fig. 4.

No entanto, Current et al. [10] apresentaram um processo para determinar soluções não dominadas que se encontrem nas zonas de desníveis de dualidade, utilizando uma técnica que consiste em resolver um problema de caminho mais curto com um só objectivo, com restrições adicionais. Neste caso, recomenda-se, sempre que possível, a redução da zona de desnível de dualidade a pesquisar.

Suponha-se, por exemplo, que se pretende pesquisar soluções não dominadas na zona de desnível de dualidade representada pelo triângulo Δ(B,C,Q) na Fig. 4. Esta zona de desnível de dualidade encontra-se representada isoladamente na Fig. 5.

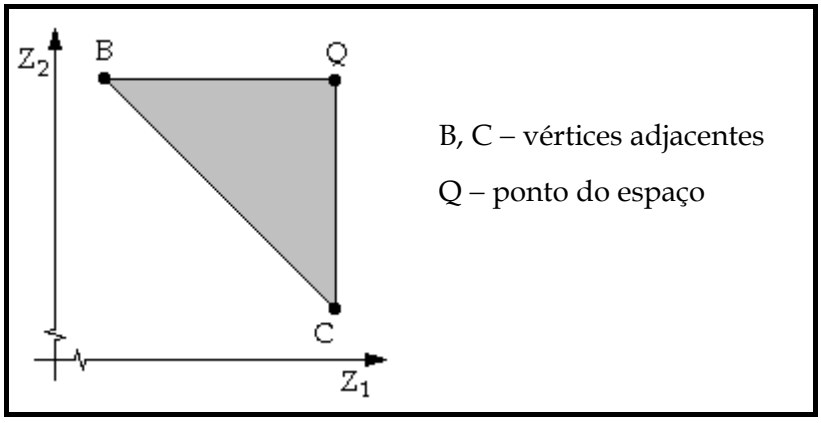


Fig. 5 – Bi-objectivo : representação de uma Zona de Desnível de Dualidade.

Para tal, pode-se resolver, por exemplo, o seguinte problema de caminho mais curto :

[P2] Min $Z = w_1.Z_1 + w_2.Z_2$ **(5.8)**

sujeito a

(5.2) –(5.5)

$Z_1 \leq Z_1(C) - \varepsilon$ **(5.9)**

em que

$w_1 = \left| \frac{Z_2(Q') - Z_2(B)}{Z_1(Q') - Z_1(B)} \right|$ é o valor absoluto do declive do segmento $\overline{BQ'}$

(onde Q' é um ponto pertencente ao segmento $\overline{CQ}$)

$w_2 = 1$

ε = um pequeno valor positivo.

Este problema corresponde a minimizar a função de somas pesadas, para a qual a linha que passa por B e Q' é uma linha de custo constante; ou seja, o gradiente desta função é perpendicular à linha referida.

Note-se que se poderia resolver outros problemas, bastando para tal alterar as restrições impostas : $Z_2 \leq Z_2(B) - \varepsilon$ (em vez da restrição (5.9)) e associar w_1 ao declive do segmento $\overline{CQ''}$ (onde Q'' é um ponto pertencente ao segmento $\overline{BQ}$).

Suponha que ao resolver-se o problema P2, com Q' = Q, foi determinada a solução **E** (Fig. 6). Então são formadas novas zonas de pesquisa, pertencentes à zona de desnível de dualidade analisada, de acordo com a solução encontrada (região a cheio na Fig. 6). Por outro lado, abaixo da recta que passa por E, paralela ao segmento $\overline{BQ}$, não foram encontradas soluções e qualquer solução que se encontre no rectângulo que contém E e Q é dominada por E.

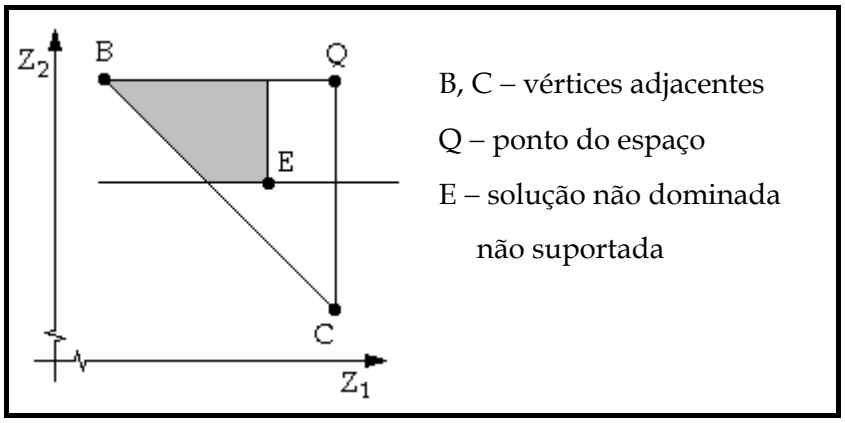


Fig. 6 – Bi-objectivo : actualização de uma Zona de Desnível de Dualidade.

Note-se que as soluções determinadas ao resolver P2 encontram-se no interior da referida zona de desnível de dualidade, uma vez que :

- abaixo da recta que contém o segmento $\overline{BC}$ não existem soluções (B e C são vértices adjacentes),
- antes de ser identificada qualquer solução acima do segmento $\overline{BQ}$, terá que ser determinada a solução B (portanto, esta solução serve de "vigia"),
- P2 não determina soluções que se encontrem à direita da recta que passa pelo segmento $\overline{CQ}$ nem a solução C, devido à restrição (5.9).

Refira-se que pode continuar-se a pesquisa de soluções nesta zona de desnível de dualidade, resolvendo o problema P2 em que a restrição (5.9) é substituída por $Z_1 \leq Z_1(E) - \varepsilon$.

5.3.2.2. Utilizando algoritmos dos k caminhos mais curtos

Estes métodos são utilizados para determinar as soluções não dominadas que se encontram no interior de uma dada zona de desnível de dualidade (não suportadas). A função objectivo utilizada é a mesma que é utilizado no método NISE com restrições, e que é construída a partir dos dois vértices que caracterizam aquela zona de desnível de dualidade.

Os algoritmos utilizados para determinar os k caminhos mais curtos são extremamente eficientes computacionalmente e identificam, muito rapidamente, todas as soluções admissíveis, segundo uma dada direcção de pesquisa. Desta forma, são determinadas todas as soluções não dominadas não suportadas de uma qualquer zona de desnível de dualidade.

Este processo de pesquisa resolve um problema de determinação dos k caminhos mais curtos correspondente ao seguinte problema :

$$\text{Min} \quad Z = w_1.Z_1 + w_2.Z_2 \qquad \textbf{(5.10)}$$

sujeito a

(5.2) –(5.5)

em que

$$w_1 = \left| \frac{Z_2(A) - Z_2(B)}{Z_1(A) - Z_1(B)} \right|$$

$$w_2 = 1$$

Z(A) e Z(B) são os vértices que definem a zona de desnível de dualidade.

Este processo é executado até se determinar um caminho, cujo valor da função objectivo seja maior ou igual do que o valor da função objectivo associado ao ponto do espaço correspondente ao terceiro vértice do triângulo que representa a zona de desnível de dualidade escolhida.

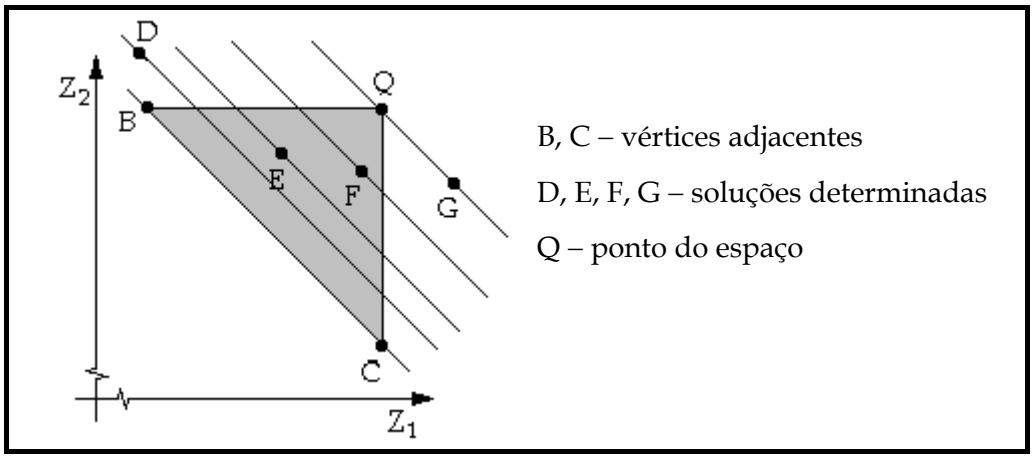


Fig. 7 – Bi-objectivo : utilização dos k caminhos mais curtos.

Para o exemplo apresentado na Fig. 7 o referido vértice é o ponto Q — esta zona de desnível de dualidade é representada pelo triângulo Δ(B,C,Q).

Ao aplicar-se o processo descrito à zona de desnível de dualidade representada na Fig. 7, em que w_1 é o valor absoluto do declive do segmento $\overline{BC}$, são determinadas as soluções B, C (ou C, B), D, E, F e G, por esta ordem. O processo termina quando se determina a solução G, pois o valor da função objectivo correspondente a esta solução é igual ao do ponto Q. Analisando as características das soluções determinadas, verifica-se que :

- B e C são soluções não dominadas suportadas extremas,
- E e F são não dominadas não suportadas (pertencem à zona de desnível de dualidade),
- D não pertence a esta zona de desnível de dualidade, mas à definida por A e B (Fig. 4).
- G é dominada pela solução C.

5.4. Algoritmos interactivos para determinar a "melhor" solução de compromisso

Vários autores, como Hansen [13], Clímaco e Martins [5], Martins [18] e Henig [15] apresentaram algoritmos para determinar todas as soluções não dominadas de um problema de caminho mais curto bi-objectivo. No entanto, todos eles reconhecem a complexidade computacional do problema, e por isso, com excepção para Martins, propuseram algoritmos para determinar uma aproximação do conjunto de soluções não dominadas. Por outro lado, nenhum destes autores recomendou a interacção directa com o AD.

De uma maneira geral, os algoritmos existentes consistem em duas fases, as quais correspondem aos dois tipos de soluções não dominadas existentes (Fig. 4) :

1ª fase : determinar alguns vértices (ou todos caso o AD deseje), minimizando combinações convexas das duas funções objectivo envolvidas,

2ª fase : determinar algumas soluções não dominadas não suportadas, correspondentes a zonas de desníveis de dualidade onde se localizam as soluções mais de acordo com as preferências do AD.

Desta forma, os algoritmos distinguem-se uns dos outros, pelas técnicas utilizadas em cada fase, podendo alguns deles utilizar a mesma técnica na mesma fase.

Todos os algoritmos que irão ser analisados baseiam-se em sistemas interactivos de apoio à decisão, e têm por objectivo reduzir o esforço computacional necessário à identificação de uma solução final pois, desta forma, não é necessário que se determinem todas as soluções não dominadas do problema, mas apenas um subconjunto, o qual irá conter a "melhor" solução de compromisso.

5.4.1. Algoritmo de Current, ReVelle e Cohon

Este algoritmo, que foi apresentada por Current et al. [10] em 1988, utiliza, na primeira fase do processo, a versão do algoritmo NISE, que corresponde ao método apresentado por Cohon (ver secção 5.3.1.4), e na segunda fase, também o algoritmo NISE, mas agora correspondente à versão com restrições adicionais (ver secção 5.3.2.1).

No entanto, o algoritmo NISE foi modificado de forma a identificar um subconjunto do conjunto de soluções não dominadas através da interacção com o AD. Desta forma, o algoritmo concentra a procura de soluções eficientes em áreas, do conjunto das soluções não dominadas, que parecem mais prometedoras ao AD.

Em cada iteração o AD é interrogado para indicar, caso o deseje, se alguma região do conjunto não inferior pode ser eliminada em posteriores pesquisas (por não lhe interessar), para estabelecer a direcção na pesquisa de novas soluções no Contorno Convexo ou para escolher as Zonas de Desníveis de Dualidade a serem analisadas. Desta forma, as soluções que pertencem às regiões eliminadas não irão ser consideradas. A eliminação de regiões pode ser baseada no valor de uma das funções objectivo ou na importância dos compromissos entre os objectivos na região.

Na primeira fase do algoritmo aplica-se o método NISE apresentado por Cohon, podendo-se determinar todos os vértices do problema, caso o AD o deseje. Nas duas primeiras iterações, são identificados os vértices que optimizam separadamente cada função objectivo. Nas restantes, aplica-se o algoritmo NISE associado aos dois vértices adjacentes escolhidos pelo AD. Esta fase termina quando todos os vértices forem encontradas, ou então, por indicação do AD, por achar que as soluções que possam existir nas regiões não analisadas não têm interesse para ele (decisão baseada na sua experiência).

Note-se que o algoritmo NISE permite conhecer, numa dada região, se não existem mais vértices, o que acontece quando a solução encontrada é um dos vértices que serviu para construir a função objectivo a minimizar, o que significa que aqueles vértices definem uma Zona de Desnível de Dualidade.

Para identificar as soluções não dominadas que possam existir numa determinada Zona de Desnível de Dualidade, este algoritmo utiliza uma versão do método NISE com restrições adicionais. No entanto, como a pesquisa de soluções nestas regiões requer um elevado esforço computacional, é recomendada a redução da Zona de Desnível de Dualidade, eliminando a região que represente, para o AD, compromissos indesejáveis entre os objectivos (Fig. 8).

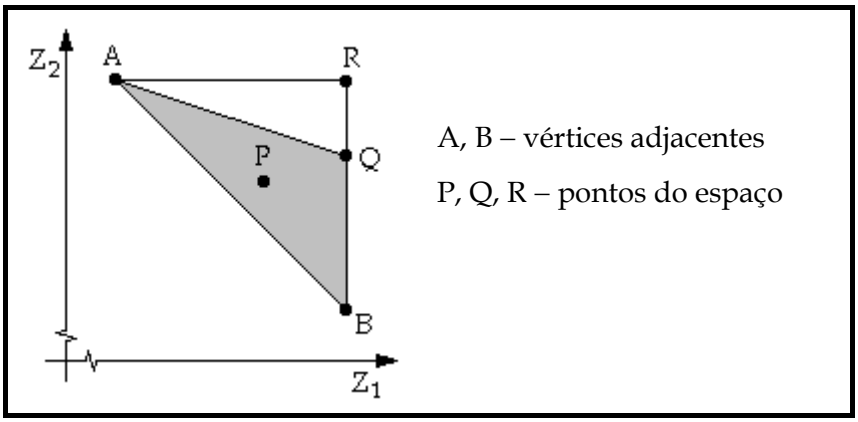


Fig. 8 – Bi-objectivo : redução de uma Zona de Desnível de Dualidade.

Analisando a Fig. 8, pode-se concluir que o aumento necessário no objectivo Z_1 para diminuir (melhorar) em uma unidade o objectivo Z_2, é maior ao deslocar-se de A para qualquer ponto P da zona de desnível de dualidade, do que ao deslocar-se de A para B, já que o valor absoluto do declive do segmento $\overline{AP}$ será menor que o do segmento $\overline{AB}$. Assim sendo, o compromisso de Z_1 para Z_2 é limitado pelos declives dos segmentos $\overline{AR}$ e $\overline{AB}$.

Desta forma, pode-se reduzir a região de pesquisa nesta zona de desnível de dualidade, através da indicação, por parte do AD, do compromisso máximo aceitável de Z_1 para Z_2. Supondo-se, por exemplo, que o AD indica um compromisso máximo aceitável de 3, então qualquer solução que se encontre acima do segmento $\overline{AQ}$ pode ser eliminada, uma vez que o declive desta linha é –1/3. Ou seja, qualquer solução que se encontre acima daquela linha, terá um compromisso superior a 3.

Para determinar a solução não dominada abaixo do segmento $\overline{AQ}$ (em Δ(A,B,Q)), deve ser aplicado o algoritmo correspondente ao método NISE com restrições associado às soluções A e B. Este problema corresponde a minimizar a função de somas pesadas, para a qual a linha que passa pelo segmento $\overline{AQ}$ é uma linha de custo constante; ou seja, o gradiente desta função é perpendicular ao segmento referido.

Como resultado da resolução deste problema foi determinada a solução C, a partir da qual se actualizam as regiões de interesse onde podem existir mais soluções não dominadas, pois C domina qualquer uma que se situe no rectângulo em que C e R são dois dos seus vértices; isto é, qualquer solução x tal que $Z_1(x) \geq Z_1(C)$ e $Z_2(x) \geq Z_2(C)$ é dominada pela solução C. Por outro lado, não existem soluções abaixo da linha paralela ao segmento $\overline{AQ}$ que passa por C (Fig. 9).

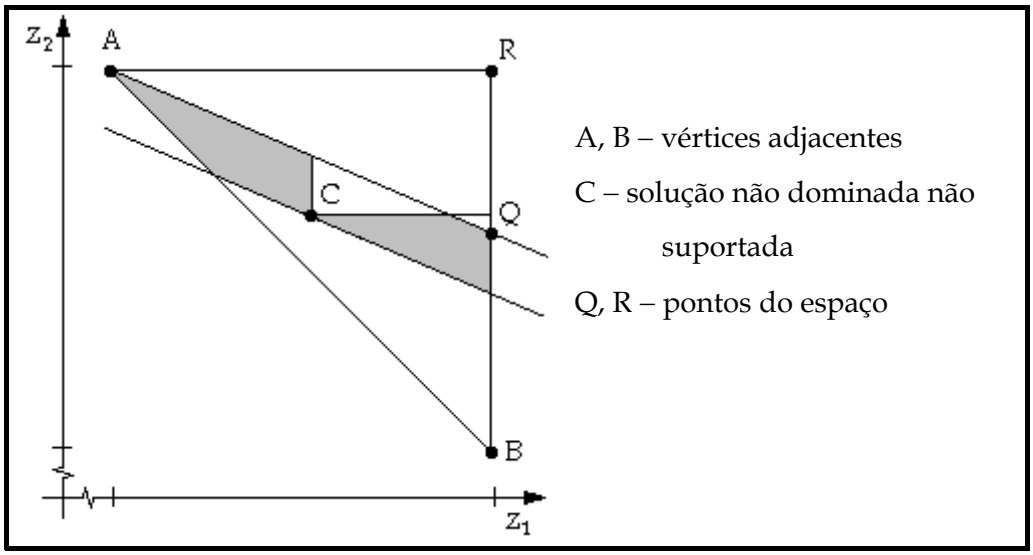


Fig. 9 – Bi-objectivo : actualização de uma Zona de Desnível de Dualidade.

Para encontrar uma outra solução, aplica-se novamente o algoritmo NISE com restrições, correspondente ao problema P2, em que a restrição (5.9) é substituída por $Z_1 \leq Z_1(C) - \varepsilon$, caso se pretenda analisar a região à esquerda de C, ou então, o mesmo problema P2 com $w_1 = 0$ (logo, $w_2 = 1$), se for pretensão analisar a região à direita de C. No primeiro caso, o resultado é a determinação da solução A, o que provoca o término do processo, já que esta solução encontra-se sobre o segmento $\overline{AQ}$ (tem um compromisso igual a 3).

5.4.2. Algoritmo de Clímaco e Rodrigues

Este algoritmo, que foi apresentado por Clímaco e Rodrigues [8] em 1988, surgiu na sequência de duas abordagens associadas ao problema de caminho mais curto bi-critério :

1) Clímaco e Martins [5], Martins [17] e Clímaco [6] desenvolveram um método para gerar todos os caminhos não dominados, baseado num algoritmo para determinar os k caminhos mais curtos e num adequado teste de não dominância;
2) Current et al. [10] apresentaram um método interactivo como alternativa aos métodos que geram todos os caminhos não dominados.

Este algoritmo foi apresentado em duas versões, correspondendo a duas maneiras diferentes de analisar as zonas de desníveis de dualidade : uma utiliza apenas parte daquela zona (tal como em [10]) e a outra utiliza toda a zona.

A primeira fase do processo é comum às duas versões e, tal como no método proposto por Current et al. [10], utilizam a mesma versão do algoritmo NISE, apresentado por Cohon, para determinar alguns ou todos os vértices.

Na segunda fase do processo, é utilizado o algoritmo para determinar os k caminhos mais curtos descrito por Martins [17], para determinar as soluções não dominadas existentes nas zonas de desníveis de dualidade. Para tal, são apresentados dois processos para o fazer, associados a cada versão do algoritmo :

1ª) tal como acontece no método de Current et al. [10], é solicitado ao AD que fixe o compromisso máximo aceitável de Z_1 para Z_2. Por exemplo, se apenas se está interessado em soluções contidas no triângulo Δ(A,B,Q), utiliza-se o algoritmo referido cuja função objectivo a minimizar é uma soma pesada, para a qual a linha que passa pelo segmento $\overline{AQ}$ é de custo constante (gradiente da função perpendicular ao segmento $\overline{AQ}$) — Fig. 8.

2ª) funciona da mesma maneira que a versão anterior, mas considera-se agora toda a zona de desnível de dualidade (o AD não impõe qualquer compromisso). Por exemplo, se se está interessado em todas as soluções contidas no triângulo Δ(A,B,R), utiliza-se o mesmo algoritmo, mas agora a função objectivo utilizada é uma soma pesada, para a qual a linha que pelo segmento $\overline{AB}$ é de custo constante (gradiente perpendicular a $\overline{AB}$) — Fig. 8.

Excluindo o caso em que a pesquisa de soluções termina quando o AD desejar, esta pesquisa só termina, na primeira versão, quando for encontrada a solução A, pois é a última solução dentro da região pretendida (Fig. 8), e na segunda, quando se determina uma solução que se encontre na zona onde se tem a certeza que não existem mais soluções não dominadas, o que acontece quando se passar para além da linha carregada (Fig. 10) — esta linha é paralela ao segmento $\overline{AB}$ e contém o ponto pertencente a alguma região de interesse que se encontra a maior distância do segmento $\overline{AB}$.

Desta forma, enquanto na primeira versão a solução D nunca é determinada, na segunda isso já acontece (Fig. 10). No entanto, a segunda versão parece ser mais adequada quando se pretende pesquisar em pequenas zonas de desníveis de dualidade, uma vez que neste caso, o tempo que se ganha ao reduzir a área de pesquisa é insignificante, atendendo aos tempos conseguidos com os algoritmos existentes para determinar os k caminhos mais curtos na determinação de tais caminhos.

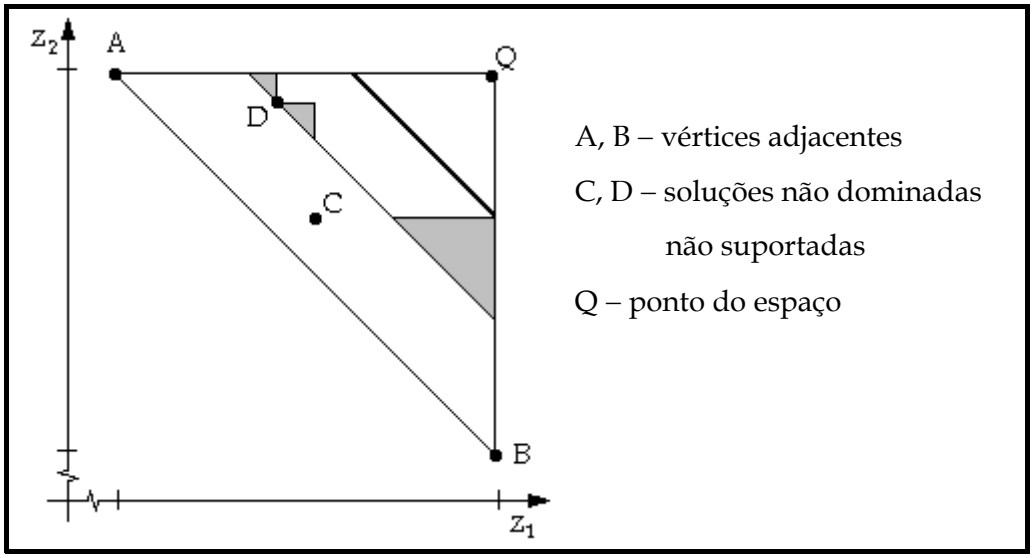


Fig. 10 – Bi-objectivo : soluções de uma Zona de Desnível de Dualidade.

C A P Í T U L O 5

Abordagem Interactiva ao Problema de Caminho Mais Curto Multiobjectivo

1. Introdução

Neste capítulo é apresentada uma abordagem ao problema de caminho mais curto bi e tri-objectivo, que tira partido da interacção AD–computador, de forma a determinar a "melhor" solução de compromisso do problema, de acordo com as preferências do AD. Desta forma, é possível chegar a uma solução final minimizando o esforço computacional e o esforço de tratamento da informação imposta ao AD, uma vez que não há necessidade de determinar todas as soluções não dominadas do problema. De facto, por um lado, logo que seja apresentada ao AD uma solução não dominada que o satisfaça totalmente, deixa de ser necessário continuar com a pesquisa de mais soluções (mesmo correndo-se o risco de poder existir uma outra que o satisfaça ainda mais); por outro lado, há a possibilidade de o AD dirigir a pesquisa para regiões onde este considere que se localizam soluções mais de acordo com as suas preferências, eliminando outras por as considerar sem interesse.

Como existe interacção entre o AD e o computador, em que este irá aproveitar e seguir as indicações do primeiro, terá que ser facultada ao AD certa informação relevante, que servirá de auxílio quer para indicar futuras direcções de pesquisa de outras soluções não dominadas, quer para eliminar regiões que considere menos interessantes.

Toda esta informação será apresentada servindo-se dos aspectos gráficos proporcionados pelo WINDOWS (95 e NT). Para tal, recorreu-se à linguagem de programação por objectos denominada DELPHI PASCAL da BORLAND.

Os gráficos utilizados representam as soluções no espaço dos objectivos, mostrando as soluções não dominadas do problema que vão sendo encontradas. Desta forma, a cada tipo de problema está associado um tipo diferente de gráfico.

No problema bi-objectivo, o gráfico utilizado para representar as soluções consiste num sistema de dois eixos (F1 e F2), onde cada solução é representada por um ponto (x, y) — **x** e **y** são os valores dos objectivos 1 e 2, respectivamente, relativamente àquela solução. Cada eixo tem duas marcas correspondentes aos valores mínimo e máximo não dominados que cada função objectivo pode atingir (Fig. 11.(a)). Por outro lado, todas as soluções são representadas com diferentes cores, para que cada solução seja também identificada pela cor em todos os gráficos onde esteja representada[1].

Para o problema tri-objectivo, as soluções são representadas num gráfico com três eixos (F1, F2 e F3), onde o ângulo entre quaisquer dois eixos é de 120 graus e o ponto central corresponde à solução ideal. Cada solução é representada por um triângulo com uma determinada cor, cujos vértices se situam nos respectivos eixos e correspondem às diferenças, normalizadas em cada dimensão, entre os valores dos objectivos e os valores mínimos destes (solução ideal) — Fig. 11.(b). Por exemplo, na Fig. 11.(b) a solução ideal tem os valores (40, 40, 140) e a solução representada (rendilhado) tem (320, 1220, 270); portanto, a solução marca 280 (= 320 – 40), 1180 (= 1220 – 40) e 130 (= 270 – 140) unidades nos eixos F1, F2 e F3, respectivamente.

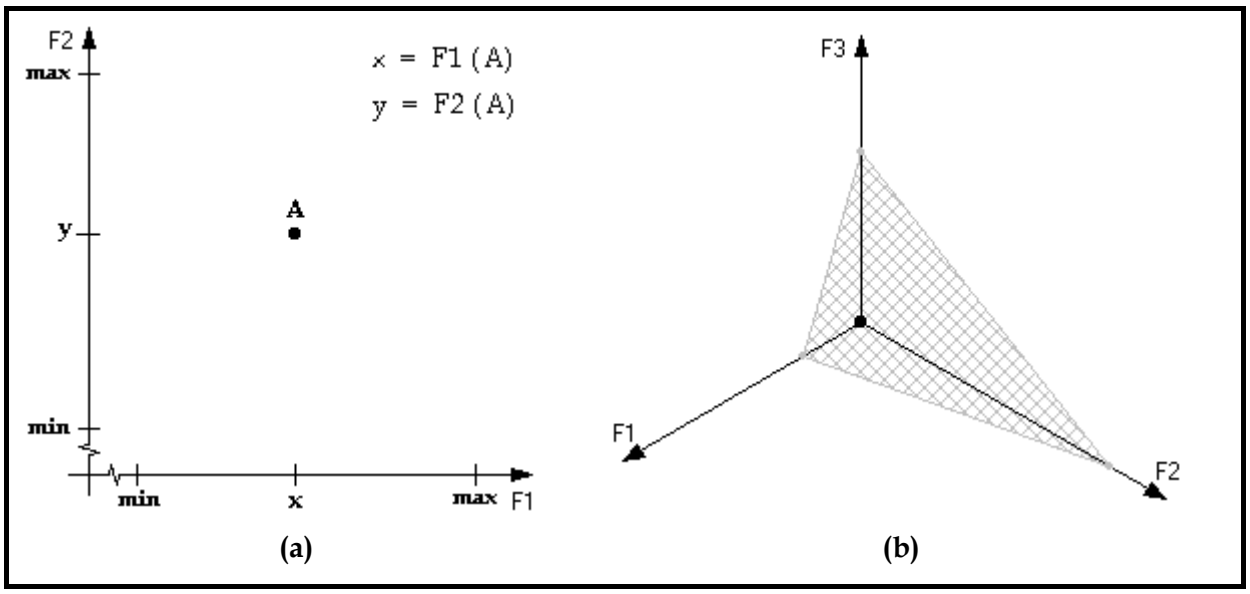


Fig. 11 – Gráficos para representar as soluções nos problemas **(a)** bi e **(b)** tri-objectivo.

[1] Porém, nalguns gráficos apresentados nesta tese, as cores são omitidas por não existir ambiguidade.

2. Definição dos problemas

Em qualquer dos problemas, podem ser determinadas soluções não dominadas optimizando uma função escalar que é uma combinação convexa de **h** (2 ou 3) objectivos, onde o coeficiente de custo associado a cada arco (i, j), $\mathbf{c_{ij}}$, é uma soma pesada não negativa de coeficientes de custo relativos a cada função objectivo. Ou seja, uma solução (vértice) não dominada pode ser obtida resolvendo o seguinte problema :

[P1] Minimizar $Z = \sum_{i \in N} \sum_{j \in N} c_{ij}\ x_{ij}$

sujeito a

$$\sum_{j \in N} x_{sj} = 1 \qquad \textbf{(1)}$$

$$\sum_{i \in N} x_{ij} - \sum_{k \in N} x_{jk} = 0, \quad \forall j \in N - \{ s, t \} \qquad \textbf{(2)}$$

$$\sum_{i \in N} x_{it} = 1 \qquad \textbf{(3)}$$

$$x_{ij} = 0, 1 \quad \forall\ i, j \in N \qquad \textbf{(4)}$$

com $c_{ij} = \sum_{k=1}^{h} \lambda_k\ c_{ij}^k$, onde $\lambda = (\lambda_1, ..., \lambda_h) \in \Lambda = \{ \lambda_k \geq 0, k = 1, ..., h; \sum_{k=1}^{h} \lambda_k = 1 \}$ e $h \in \{ 2, 3 \}$.

Refira-se, no entanto, que existem combinações convexas particulares para as quais as soluções resultantes não são estritamente não dominadas. Estas situações ocorrem sempre que exista um óptimo alternativo para qualquer $\lambda_k = 0$ (por exemplo, quando se determinam os óptimos individuais das **h** funções objectivo).

No problema bi-objectivo, esta situação surge apenas ao determinar a solução óptima de uma das funções objectivo, quando existe mais do que uma solução óptima para esse objectivo. A forma de ultrapassar esta questão, consiste em determinar todas as soluções óptimas alternativas e escolher a não dominada, utilizando o valor do outro objectivo associado a cada uma dessas soluções. Neste caso, apenas existe uma solução estritamente não dominada, pois o valor da função objectivo a optimizar é o mesmo para todas elas, sendo não dominada a solução que tiver menor valor relativamente ao outro objectivo.

Se a solução óptima do problema P1 é única, então corresponde a um caminho vértice não dominado. Desta forma, na resolução do problema P1, apenas se determinam vértices. Estas soluções pertencem ao Contorno Convexo de um conjunto de soluções não dominadas, no espaço das funções objectivo com **h** dimensões.

Uma solução não dominada localizada no interior do Invólucro Convexo (isto é, que não é vértice), não pode ser obtida desta forma, uma vez que é dominada por uma combinação convexa de vértices, e assim, não pode ser solução de P1 (não existe $\lambda \in \Lambda$ que defina um hiperplano de suporte para ela). Por esta razão, apesar de serem não dominadas, são geralmente denominadas por soluções convexamente dominadas ou soluções não suportadas, as quais se encontram dentro das denominadas Zonas de Desníveis de Dualidade. Mas, e uma vez que elas são efectivamente não dominadas, devem ser consideradas como potenciais soluções de compromisso e, consequentemente, o método de procura deve adaptar-se ao seu cálculo. Estas soluções são, então, determinadas através de um algoritmo para determinar os k caminhos mais curtos.

O algoritmo para determinar os k caminhos mais curtos utilizado na abordagem aqui apresentada é a versão mais recente do MPS, que foi desenvolvido por Martins et al. [20] (ver secção 4.3 do Capítulo 4). A escolha deste algoritmo deve-se ao facto de ser muito recente (1997) e muito eficiente. Este algoritmo calcula 500 000 trajectos, em redes euclidianas não orientadas com 10 000 nós e 100 000 arcos, em cerca de 0.35 segundos de tempo CPU (em redes orientadas determina 600 000 trajectos em cerca de 0.15 segundos). Em redes completas não orientadas com 1 000 nós, este algoritmo determina 580 000 trajectos em cerca de 2.1 segundos de tempo CPU (em redes orientadas, determina 780 000 trajectos em cerca de 2.01 segundos).

A abordagem proposta apresenta, para qualquer dos problemas (bi e tri-objectivo), dois métodos diferentes para determinar a "melhor" solução de compromisso :

a) analisando todo o espaço dos objectivos, determinando soluções segundo uma determinada direcção de pesquisa,

b) analisando o Contorno Convexo e as Zonas de Desníveis de Dualidade.

No primeiro método, o passo inicial consiste em determinar a direcção de pesquisa de soluções. No passo principal, cada iteração consiste em determinar uma solução de acordo com aquela direcção de pesquisa.

O segundo método é composto por duas fases, consoante se pretenda determinar :

1ª) soluções não dominadas que pertencem ao Contorno Convexo,

2ª) soluções não dominadas que pertencem às Zonas de Desníveis de Dualidade.

3. As soluções dos problemas

As categorias de soluções dos problemas são dominada, não dominada do Contorno Convexo (vértice) e não dominada das Zonas de Desníveis de Dualidade. Por outro lado, introduzimos dois conceitos que se aplicam aos vértices de um problema multiobjectivo, que são o de **vértices adjacentes** e **definitivamente adjacentes**. Num problema com **h** objectivos, a adjacência é verificada entre combinações de **h** vértices.

Diz-se que **h** vértices são **adjacentes**, se abaixo do hiperplano definido pelos **h** vértices ainda não foi encontrado qualquer vértice, mas que pode existir (uma das formas de verificar se existe algum vértice nessas circunstâncias, é resolver o problema P1, cujos pesos correspondem ao gradiente daquele hiperplano). Aqueles **h** vértices dizem-se **não adjacentes** se existe pelo menos outro vértice abaixo do hiperplano definido pelos **h** vértices.

Diz-se que **h** vértices são **definitivamente adjacentes**, se já se tem a garantia que não existe qualquer vértice abaixo do hiperplano definido pelos **h** vértices (pode-se verificar, resolvendo o problema P1, em que os pesos utilizados correspondem ao gradiente do hiperplano referido, e não se encontrar qualquer novo vértice).

A partir destes dois conceitos, uma Zona de Desnível de Dualidade é completamente caracterizada por **h** vértices definitivamente adjacentes, num problema com **h** objectivos.

3.1. Problema bi-objectivo

Diz-se que **dois** vértices são ***adjacentes***, se abaixo da recta que passa por dois vértices não se conhece qualquer vértice (mas que pode existir), e diz-se que são ***definitivamente adjacentes***, se já se tem a certeza que não existe qualquer vértice abaixo daquela recta. Ou seja, **dois** ***vértices*** são ***definitivamente adjacentes***, se ao resolver-se o problema P1 (fazendo h=2), cujos pesos correspondem ao gradiente da recta que passa por esses dois vértices, não for encontrado qualquer novo vértice.

Relativamente às definições de Contorno Convexo e de Zona de Desnível de Dualidade, o primeiro é composto por todas as combinações de h vértices definitivamente adjacentes, e a segunda é caracterizada por dois vértices definitivamente adjacentes. Desta forma, a Zona de Desnível de Dualidade caracterizada pelos vértices **x** e **y**, consiste na região representada por um triângulo, cujos vértices são : x, y e o ponto construído à custa dos valores máximos de cada objectivo relativamente a x e y (regiões a cheio na Fig. 12).

Na Fig. 12 encontram-se ilustrados os vários tipos de soluções para o problema bi-objectivo : dominada e não dominada (do Contorno Convexo e de Zonas de Desníveis de Dualidade), bem como a solução ideal.

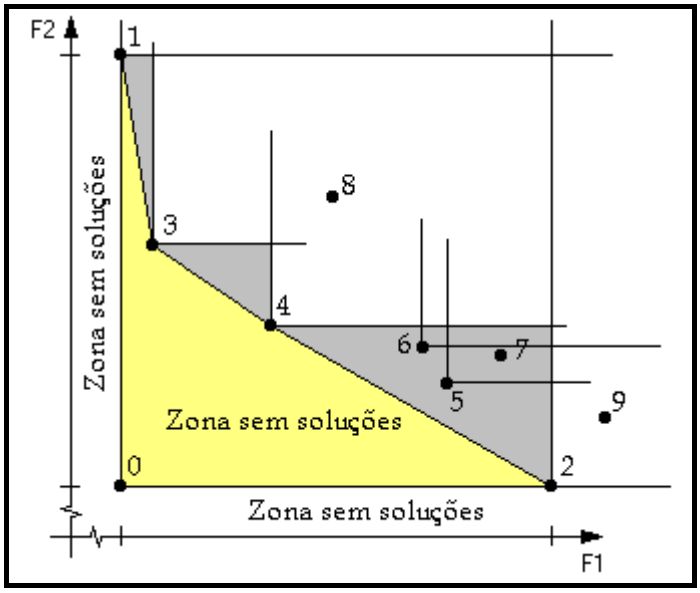


Fig. 12 – Bi-objectivo : tipos de soluções.

As soluções 1, 2, 3 e 4 pertencem ao Contorno Convexo. As soluções 1 e 2 foram determinadas optimizando, separadamente, as funções objectivo 1 e 2, respectivamente. A solução 3 foi determinada optimizando a função escalar cujos pesos correspondem ao gradiente da recta que passa pelas soluções 1 e 2. A solução 4 foi encontrada ao optimizar-se a função escalar cujos pesos correspondem ao gradiente da recta que passa pelas soluções 2 e 3.

Desta forma, os vértices 1 e 2 deixaram de ser adjacentes entre si (foram-no quando eram únicas), porque a partir deles foi determinado o vértice 3. Da mesma forma, os vértices 2 e 3 também deixaram de ser adjacentes entre si. Por outro lado, (1, 3), (3, 4) e (2, 4) são pares de vértices definitivamente adjacentes entre si, uma vez que a partir deles não foram determinados quaisquer vértices. Por esta razão, estes pares de vértices definem Zonas de Desníveis de Dualidade.

As soluções 5, 6 e 7 pertencem à Zona de Desnível de Dualidade definida pelas soluções 2 e 4. No entanto, enquanto que as soluções 5 e 6 são não dominadas, a solução 7 é dominada (pela 5).

As soluções 8 e 9 são dominadas, pois nem pertencem ao Contorno Convexo, nem a qualquer Zona de Desnível de Dualidade (a 8 é dominada pelas soluções 3 e 4 e a 9 pela solução 2).

A solução 0 é a solução ideal, que optimizaria simultaneamente todas as funções objectivo, mas que não é admissível. A *solução ideal* obtém-se considerando os valores das soluções que optimizam separadamente cada função objectivo, mas sem ter associado qualquer caminho, já que tal solução geralmente é não admissível (quando admissível é a única não dominada).

3.2. Problema tri-objectivo

Diz-se que **três** vértices são ***adjacentes***, se abaixo do plano que contém os três vértices não se conhece qualquer vértice (mas que pode existir), e diz-se que são ***definitivamente adjacentes***, se já se tem a garantia que não existe qualquer vértice abaixo do referido plano. A forma de verificar se existe qualquer vértice abaixo do plano que contém aqueles três vértices, consiste em resolver o problema P1 (fazendo h=3), cujos pesos utilizados correspondem exactamente ao gradiente do plano que contém aqueles três vértices.

Quando uma das componentes do gradiente do plano que contém os três vértices é negativa, o peso correspondente é feito nulo (o que significa um relaxamento do gradiente) para evitar a obtenção de soluções dominadas. Contudo, dada a perturbação do gradiente original assim introduzido, não pode ser garantido que, usando uma função objectivo soma ponderada com esse conjunto de pesos que não conduza a qualquer vértice, este não exista abaixo do plano definido por aqueles vértices.

A definição de Contorno Convexo utilizada no problema bi-objectivo, também é válida neste problema, isto é, o conjunto de todos os vértices do problema.

No que respeita à definição de Zona de Desnível de Dualidade já isso não acontece, apesar da sua caracterização ser semelhante, isto é, por três vértices definitivamente adjacentes. No entanto, definimos uma Zona de Desnível de Dualidade da seguinte forma : uma solução não dominada **x** pertence à Zona de Desnível de Dualidade definida pelos vértices A, B e C ([A, B, C]), se se encontrar no interior da pirâmide cujos vértices são A, B, C e P, em que P é o ponto construído com os valores máximos de cada função objectivo relativamente às soluções A, B e C (Fig. 13).

Por exemplo, as soluções A = (1600, 40, 180), B = (320, 1220, 270) e C = (860, 1110, 150) e o ponto P = (1600, 1220, 270) formam uma pirâmide, a qual define uma Zona de Desnível de Dualidade representada na Fig. 13. Desta forma, qualquer solução não dominada que se encontre no interior desta pirâmide, pertence a esta Zona de Desnível de Dualidade.

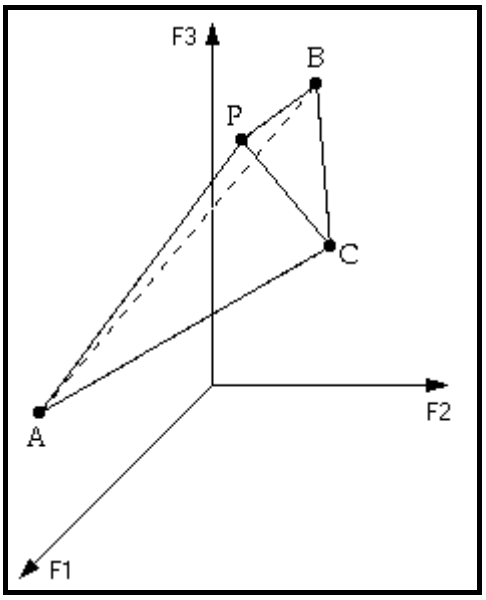


Fig. 13 – Tri-objectivo : representação duma Zona de Desnível de Dualidade.

Atendendo às definições de Zona de Desnível de Dualidade, enquanto no problema bi-objectivo qualquer solução não dominada pertence a uma Zona de Desnível de Dualidade específica, no problema tri-objectivo isso já não acontece, uma vez que podem existir soluções não dominadas que não pertençam a qualquer Zona de Desnível de Dualidade.

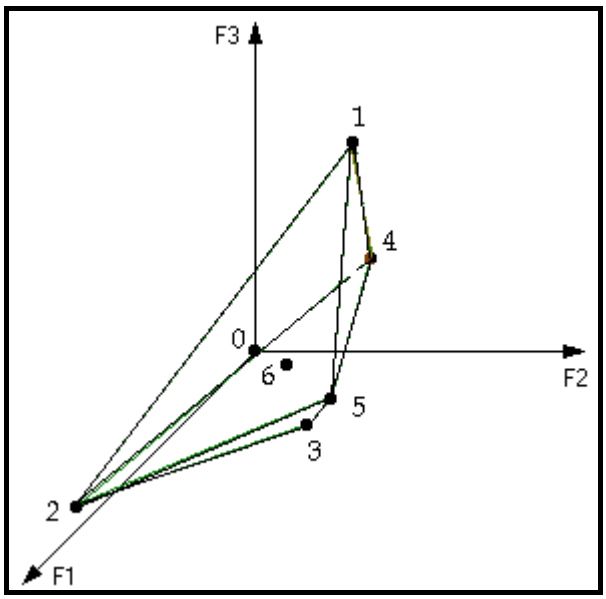


Fig. 14 – Tri-objectivo : tipo de soluções.

A Fig. 14 mostra alguns tipos de soluções num problema tri-objectivo : não dominada do Contorno Convexo e não dominada de uma Zona de Desnível de Dualidade, bem como a solução ideal. A solução 0 é a ideal, as soluções 1, 2, 3, 4 e 5 pertencem ao Contorno Convexo

(únicas), as quais formam quatro Zonas de Desníveis de Dualidade : [1, 2, 4], [1, 4, 5], [2, 3, 5] e [2, 4, 5]. A solução 6 pertence à Zona de Desnível de Dualidade identificada por [2, 4, 5] e apenas a esta (de acordo com os cálculos efectuados com a aplicação associada à abordagem aqui apresentada).

O gráfico da Fig. 15 contém as cinco soluções do Contorno Convexo encontradas.

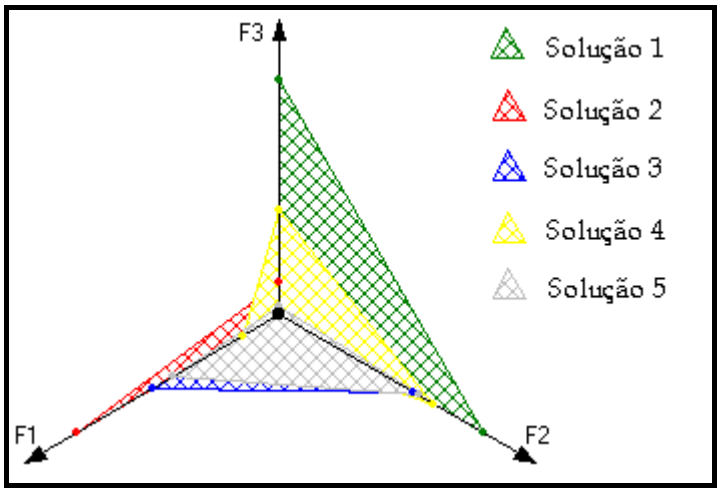


Fig. 15 – Tri-objectivo : gráfico das soluções do Contorno Convexo.

O gráfico da Fig. 16 contém a solução 6 inserida na Zona de Desnível de Dualidade a que pertence, que é a definida pelos vértices 2, 4 e 5 (também representadas).

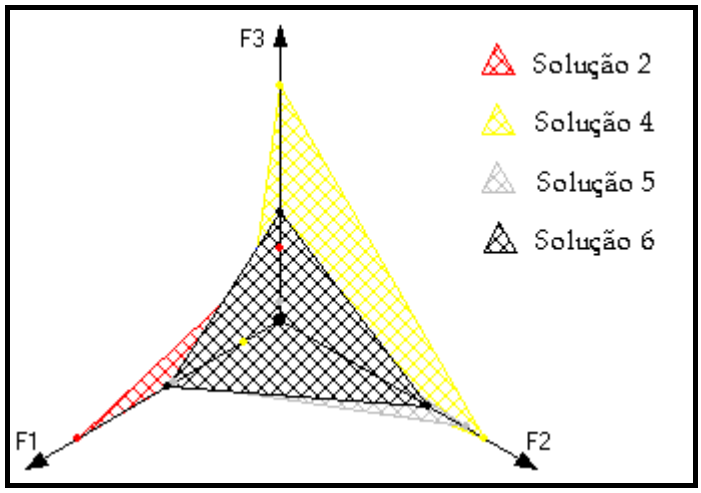


Fig. 16 – Tri-objectivo : soluções de uma Zona de Desnível de Dualidade.

4. Método de procura de soluções em todo o espaço dos objectivos

Este processo permite determinar soluções não dominadas, analisando todo o espaço dos objectivos, não havendo possibilidade de eliminar regiões do espaço de pesquisa.

O passo inicial deste método consiste em determinar a direcção de pesquisa de soluções não dominadas, a qual corresponde ao gradiente do hiperplano definido por **h** pontos do espaço dos objectivos, cada um dos quais composto por apenas um valor não nulo, o qual corresponde ao valor óptimo de cada um dos objectivos. Por exemplo, se as soluções que optimizam individualmente os objectivos 1, 2 e 3 de um problema tri-objectivo têm associado um custo de **a**, **b** e **c**, respectivamente, então a direcção de pesquisa corresponde ao gradiente do plano que contém os pontos (a, 0, 0), (0, b, 0) e (0, 0, c).

No passo principal, cada iteração consiste em determinar uma solução não dominada, resolvendo o problema P1, para determinar os k caminhos mais curtos, cuja função escalar é construída a partir dos pesos definidos da forma descrita, utilizando o algoritmo MPS.

Este processo termina quando todas as soluções não dominadas forem determinadas, ou então quando o AD o decidir.

No entanto, a utilização deste método coloca uma questão. Será que se pode verificar a não dominância de uma solução, apenas tendo em conta as soluções já encontradas segundo aquela direcção de pesquisa ? Ou por outras palavras, será que uma solução encontrada num certo momento, e segundo uma dada direcção de pesquisa, não pode dominar uma outra encontrada antes ? De facto uma solução determinada, nas circunstâncias descritas, num certo instante não domina qualquer solução que tenha sido determinada antes, como se prova a seguir.

Sejam x e y duas soluções determinadas por esta ordem, tais que os seus valores relativamente à função escalar são Cx e Cy, respectivamente (com $Cx \leq Cy$, pois como x foi determinado antes de y, corresponde a um caminho mais curto). Por outro lado, os valores dos **h** objectivos associados a estas duas soluções são $cx = (c_1x, \ldots, c_hx)$ e $cy = (c_1y, \ldots, c_hy)$, tais que $Cx = \lambda_1 c_1x + \ldots + \lambda_h c_hx$ e $Cy = \lambda_1 c_1y + \ldots + \lambda_h c_hy$, com $\lambda_i \geq 0$ e $i = 1, \ldots, h$.

Suponha-se que y domina x. Então, $c_iy \leq c_ix$ para qualquer $\mathbf{i} \in \{ 1, \ldots, h \}$ e existe pelo menos um $\mathbf{j} \in \{ 1, \ldots, h \}$ tal que $c_jy < c_jx$.

Logo,

$\lambda_j c_jy < \lambda_j c_jx$ com $\lambda_j \geq 0$ para $j \in \{ 1, \ldots, h \}$ e

$\lambda_i c_iy \leq \lambda_i c_ix$ com $\lambda_i \geq 0$ para $i = 1, \ldots, h$ e $i \neq j$.

Consequentemente,

$\lambda_j c_j y < \lambda_j c_j x$, com $j \in \{1, ..., h\}$ e

$$\sum_{j \neq i=1}^{h} \lambda_i c_i y \leq \sum_{j \neq i=1}^{h} \lambda_i c_i x .$$

Desta forma, como $\lambda_j c_j y < \lambda_j c_j x$ então

$$\sum_{i=1}^{h} \lambda_i c_i y < \sum_{i=1}^{h} \lambda_i c_i x \quad (Cy < Cx).$$

Assim sendo, $\lambda_1 c_1 x + ... + \lambda_h c_h x = Cx > Cy = \lambda_1 c_1 y + ... + \lambda_h c_h y$, o que é uma contradição.

Logo, y não pode dominar x.

No problema bi-objectivo, cada iteração do passo principal do método consiste em resolver o seguinte problema para determinar os k caminhos mais curtos :

[P2] Minimizar $Z = \lambda_1 Z_1 + \lambda_2 Z_2$

sujeito a **(1)** – **(4)**

onde,

$$\lambda_1 = \frac{b}{a+b}$$

$$\lambda_2 = 1 - \lambda_1$$

a e b são os valores que optimizam individualmente os objectivos 1 e 2, respectivamente.

No problema tri-objectivo, cada iteração do passo principal do método consiste em resolver o seguinte problema para determinar os k caminhos mais curtos :

[P3] Minimizar $Z = \lambda_1 Z_1 + \lambda_2 Z_2 + \lambda_3 Z_3$

sujeito a **(1)** – **(4)**

onde,

$$\lambda_i = \frac{|u_i|}{|u_1 + u_2 + u_3|} \quad (i=1,2,3)$$

$$\vec{u} = (u_1, u_2, u_3) = \vec{AB} \times \vec{AC}$$

A = (a, 0, 0), B = (0, b, 0) e C = (0, 0, c)

a, b e c são os valores que optimizam individualmente os objectivos 1, 2 e 3, respectivamente.

Os passos principais do método para determinar soluções não dominadas em todo o espaço dos objectivos, segundo uma determinada direcção de pesquisa, encontram-se descritos com detalhe em Algoritmo 7.

Passo 1. Determinar as soluções que optimizam separadamente cada um dos objectivos, utilizando o algoritmo de Dijkstra para cada um deles.

Passo 2. Se não existe qualquer solução ou as soluções são a mesma então terminar (apenas existe uma solução não dominada).

Passo 3. Questionar o AD se pretende impor restrições nos valores das funções objectivo. Tomando estas restrições e os valores máximos não dominados de cada objectivo (apenas para problemas bi-objectivo), define-se a região de interesse.

Passo 4. Construir a função escalar soma ponderada, cujos pesos correspondem ao gradiente do hiperplano definido pelos pontos que cortam os eixos de coordenadas nos valores que optimizam cada objectivo.

Passo 5. Determinar a primeira solução não dominada, resolvendo o problema P2/P3, cuja função objectivo é a função escalar construída no Passo 4, utilizando o algoritmo MPS. Acrescentá-la ao conjunto das não dominadas e saltar para o Passo 10.

Passo 6. Determinar a próxima solução não dominada (k-ésimo caminho mais curto), resolvendo o problema P2/P3, cuja função objectivo é a função escalar construída no Passo 4, utilizando o algoritmo MPS. Se não existe qualquer solução então terminar.

Passo 7. Se toda a região de interessa ficou totalmente analisada então terminar.

Passo 8. Se a solução encontrada é dominada então voltar ao Passo 6.

Passo 9. Se a solução encontrada está fora de um dos limites impostos, então acrescenta-se ao conjunto das não dominadas que estão fora dos limites impostos e volta-se ao Passo 6; caso contrário, acrescenta-se ao conjunto das não dominadas.

Passo 10. Caso o AD pretende determinar mais soluções, voltar ao Passo 6.

Algoritmo 7. Determinar soluções em todo o espaço dos objectivos.

5. Método de procura no Contorno Convexo

O algoritmo proposto nesta abordagem, baseia-se no método NISE desenvolvido por Cohon (ver secção 4.3 do Capítulo 3). Em cada iteração deste método é resolvido um problema de caminho mais curto com um só objectivo, P1, em que a função objectivo é uma

combinação convexa das **h** funções objectivo originais, cujos pesos utilizados correspondem ao gradiente do hiperplano definido por **h** vértices adjacentes.

Existem várias vantagens em utilizar este método, entre as quais se destacam as seguintes :

- existem vários algoritmos eficientes para resolver os problemas de caminho mais curto com um só objectivo, apresentados em cada iteração do método;
- a solução (caminho) óptima de um problema daqueles é não dominada no problema original (com **h** funções objectivo);
- o conjunto de soluções encontradas é fornecido ao AD com a informação relativa aos valores das soluções e aos compromissos entre os objectivos inerentes às várias regiões do conjunto não inferior. É a partir desta informação que o AD pode eliminar regiões que considere sem interesse, ou então dirigir a pesquisa de novas soluções para zonas em que lhe parece mais favorável satisfazer as suas preferências. Desta forma, podem ser necessárias poucas iterações para se identificar uma solução final.

Na fase inicial, o método proposto determina as **h** soluções não dominadas que optimizam separadamente cada função objectivo envolvida, utilizando o algoritmo MPS para determinar os k caminhos mais curtos, em vez do algoritmo de Dijkstra, de forma a se prever a existência de óptimos alternativos (isto é, pode existir mais do que uma solução que optimize um dos objectivo), o que faz com que algumas daquelas soluções possam não ser estritamente não dominadas. Por exemplo, para o caso bi-objectivo, entre todas as eventuais soluções alternativas, apenas uma é estritamente não dominada.

Por outro lado, note-se que só se pode passar à fase seguinte deste processo caso existam pelo menos dois e três vértices, respectivamente para o problema bi e tri-objectivo, já que é apenas nestas condições que é possível determinar direcções de pesquisa de soluções.

Na fase principal, cada iteração do método consiste em questionar o AD para que este indique a direcção de pesquisa de novos vértices, resolvendo o mesmo problema, mas agora utilizando o algoritmo de Dijkstra. Para tal, o AD tem que indicar **h** vértices adjacentes, que serão utilizados para definir a direcção de pesquisa. Este processo termina quando o AD achar que não precisa de determinar mais soluções do Contorno Convexo, ou quando todos os vértices forem determinados, o que acontece quando, para qualquer conjunto de **h** vértices adjacentes, a utilização da função escalar construída para a resolução do problema P1 não encontre uma nova solução.

Neste caso, a existência de óptimos alternativos não acarreta qualquer problema, uma vez que as soluções ao serem transportadas para o problema original (multiobjectivo), leva a que os valores associados a cada um dos objectivos sejam diferentes. Logo, não há dominância de umas soluções em relação às outras. Ao resolver-se o problema P1, cujos pesos correspondem ao gradiente do hiperplano definido por **h** vértices adjacentes, pode acontecer que aqueles **h** vértices sejam soluções óptimas alternativas para o problema e, no entanto, todas elas são não dominadas.

5.1. Problema bi-objectivo

Determinar as soluções do Contorno Convexo de um problema bi-objectivo, consiste em resolver o seguinte problema :

[P4] Minimizar $Z = \lambda_1 Z_1 + \lambda_2 Z_2$

sujeito a **(1)** – **(4)**

onde,

$$\lambda_1 = \frac{\left| Z_2(A) - Z_2(B) \right|}{\left| Z_1(A) - Z_1(B) \right| + \left| Z_2(A) - Z_2(B) \right|}$$

$$\lambda_2 = 1 - \lambda_1$$

$Z_i(X)$ é o valor da i-ésima função objectivo para o vértice X, com i = 1, 2

Z(A) e Z(B) são vértices adjacentes.

Com os pesos definidos desta forma, a solução encontrada, ao resolver o problema P4, encontra-se abaixo do segmento cujas extremidades são as duas soluções adjacentes A e B, ou então uma destas soluções (Fig. 17). Desta forma, se a solução encontrada é nova (diferente de A e B), então pertence ao Contorno Convexo; caso contrário (é uma daquelas) é garantido que estas duas soluções são definitivamente adjacentes, definindo, assim, uma Zona de Desnível de Dualidade. Por outro lado, é garantido que, se forem analisadas todas as combinações de soluções adjacentes, então todos os vértices do problema ficam encontrados.

Refira-se que se os pesos forem outros e a solução encontrada for A ou B, não é garantido que estas sejam definitivamente adjacentes, como se pode verificar pela análise da Fig. 17, onde se mostra que das três direcções de pesquisa (cada uma associada a uma combinação de pesos diferente e representada por uma linha), apenas uma determina a solução devida (C), que corresponde à linha paralela ao segmento de recta $\overline{AB}$.

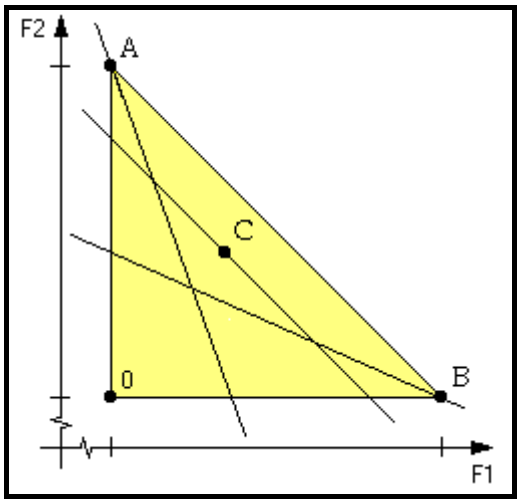


Fig. 17 – Bi-objectivo : definição dos pesos associados à função escalar.

Os principais passos do método para determinar soluções não dominadas do Contorno Convexo, encontram-se descritos, com detalhe, em Algoritmo 8.

Passo 1. Determinar os vértices que optimizam separadamente cada um dos objectivos, utilizando o algoritmo MPS para cada um deles.

Passo 2. Se não existe qualquer vértice ou os vértices são o mesmo então terminar (apenas existe uma solução não dominada).

Passo 3. Caso o AD não pretenda determinar mais vértices então terminar ou iniciar (ou reiniciar) a análise de uma qualquer Zona de Desnível de Dualidade já conhecida.

Passo 4. Questionar o AD acerca dos vértices que pretende utilizar para determinar a direcção de pesquisa de novos vértices (indicar dois que sejam diferentes e adjacentes).

Passo 5. Determinar a função escalar soma ponderada, cujos pesos correspondem ao gradiente da recta que passa pelos dois vértices indicados.

Passo 6. Resolver o problema P4, cuja função objectivo é a função escalar construída no passo anterior, utilizando o algoritmo de Dijkstra.

Passo 7. Se o vértice encontrado é novo então acrescenta-se ao conjunto das soluções não dominadas. Voltar ao Passo 3.

Algoritmo 8. Bi-objectivo : determinar soluções do Contorno Convexo.

Note-se que se o vértice encontrado no Passo 7 for novo, então isso implica que os dois vértices indicados no Passo 4 deixam de ser adjacentes entre si; caso contrário estes vértices passam a definitivamente adjacentes, formando assim uma Zona de Desnível de Dualidade.

5.2. Problema tri-objectivo

Determinar as soluções do Contorno Convexo de um problema tri-objectivo, consiste em resolver o seguinte problema :

[P5] Minimizar $Z = \lambda_1 Z_1 + \lambda_2 Z_2 + \lambda_3 Z_3$

sujeito a **(1)** – **(4)**

onde,

$$\lambda_i = \frac{|u_i|}{|u_1 + u_2 + u_3|} \quad (i=1,2,3) \quad \text{se } u_i \geq 0, \ \forall i \in \{1,2,3\}$$

ou

$$\begin{cases} \lambda_i = \dfrac{|u_i|}{|u_i + u_j|} \\ \lambda_j = \dfrac{|u_j|}{|u_i + u_j|} \qquad (i,j \neq k) \qquad \text{se } u_k < 0, \ k \in \{1,2,3\} \\ \lambda_k = 0 \end{cases}$$

com $\vec{u} = (u_1, u_2, u_3) = \vec{AB} \times \vec{AC}$, em que A, B e C são vértices adjacentes.

Da mesma maneira que se concluiu para o problema bi-objectivo, também os pesos utilizados neste problema garantem que todas as soluções do Contorno Convexo são determinadas, desde que cada combinação de pesos corresponda ao gradiente do plano definido por três vértices adjacentes.

Por outro lado, tal como acontece para o problema bi-objectivo (quando uma das combinações de pesos não corresponde ao gradiente da recta que passa por dois vértices adjacentes — Fig. 17), também neste caso se os pesos não forem aqueles e a solução encontrada ao resolver o problema P5 for um daqueles três vértices, então não é garantido que aqueles três vértices sejam definitivamente adjacentes.

Refira-se ainda que, ao resolver-se o problema P5 pode acontecer que seja determinado um vértice que já exista, mas diferente de qualquer um dos utilizados na construção da função escalar de P5. Isto significa que aqueles três vértices não são definitivamente adjacentes; logo, não formam uma Zona de Desnível de Dualidade.

Os passos essenciais para determinar as soluções do Contorno Convexo, encontram-se descritos detalhadamente em Algoritmo 9.

Passo 1. Determinar os vértices que optimizam separadamente cada um dos objectivos, utilizando o algoritmo MPS para cada um deles.

Passo 2. Se não existe qualquer vértice ou os vértices são o mesmo então terminar (apenas existe uma solução não dominada).

Passo 3. Se os três vértices são todos diferentes então passar ao Passo 7.

Passo 4. Procurar um terceiro vértice utilizando combinações de pesos aleatórias no problema P5. Se não se encontrar qualquer vértice então terminar (só dois vértices).

Passo 5. Se o AD não pretende determinar mais vértices, então terminar ou iniciar (ou reiniciar) a análise duma Zona de Desnível de Dualidade já conhecida (ver secção 6.2).

Passo 6. Questionar o AD acerca dos vértices que pretende utilizar para determinar a direcção de pesquisa de novos vértices (indicar três que sejam diferentes e adjacentes).

Passo 7. Determinar a função escalar cujos pesos correspondem ao gradiente do plano que contém os três vértices indicados.

Passo 8. Resolver o problema P5, em que a função objectivo é a função escalar construída no passo anterior, utilizando o algoritmo de Dijkstra.

Passo 9. Se o vértice encontrado é novo então acrescenta-se ao conjunto das soluções não dominadas. Voltar ao Passo 6.

Algoritmo 9. Tri-objectivo : determinar soluções do Contorno Convexo.

No Passo 7 quando uma das componentes do gradiente do plano que contém os três vértices indicados é negativa, terá que haver um relaxamento na direcção de pesquisa, para que a combinação de pesos possam ser aplicados à resolução do problema P5, pois $\lambda_i \geq 0$ (i = 1, 2, 3). Uma das alternativas é anular esta componente ou atribuir um valor muito próximo de zero. Note-se que se pode considerar, sem perda de generalidade, que qualquer gradiente tem no máximo uma componente negativa, pois se existirem duas ou três, basta multiplicar todas as componentes por (-1) — passando a ter uma ou zero, respectivamente.

6. Método de procura em Zonas de Desníveis de Dualidade

Para determinar as soluções que pertencem a uma dada Zona de Desnível de Dualidade, resolve-se um problema de determinar os k caminhos mais curtos, cuja função objectivo é construída à custa do gradiente do hiperplano definido pelas soluções que caracterizam a Zona de Desnível de Dualidade a analisar. O algoritmo utilizado na resolução deste problema

é o denominado por MPS (ver secção 4.3 do Capítulo 3). Em cada iteração do algoritmo, o AD pode decidir pela continuidade da pesquisa de soluções dentro daquela Zona de Desnível de Dualidade, ou então, por terminar essa pesquisa. A pesquisa de soluções termina mediante três situações :

a) uma das soluções encontradas satisfaz o AD, de tal modo que é escolhida como "melhor" solução de compromisso (solução final);

b) determinaram-se todas as soluções não dominadas da Zona de Desnível de Dualidade;

c) o AD considera que, apesar de ainda poderem existir soluções não dominadas na Zona de Desnível de Dualidade, elas não o irão satisfazer, decidindo, assim, analisar outra.

6.1. Problema bi-objectivo

Os principais passos para determinar as soluções não dominadas de uma determinada Zona de Desnível de Dualidade, encontram-se descritos com detalhe em Algoritmo 10.

Passo 1. Indicar os dois vértices que definem a Zona de Desnível de Dualidade a analisar (têm que ser diferentes e definitivamente adjacentes).

Passo 2. Questionar o AD se pretende impor restrições nos valores das funções objectivo. Tomando estas restrições e os valores máximos não dominados de cada objectivo nesta zona constrói-se a região de interesse.

Passo 3. Construir a função escalar soma ponderada, cujos pesos correspondem ao gradiente da recta que passa pelos vértices indicados no Passo 1.

Passo 4. Determinar a primeira solução não dominada, resolvendo o problema P4, utilizando o algoritmo MPS. Saltar para o Passo 6.

Passo 5. Determinar a próxima solução (k-ésimo caminho mais curto) do problema P4, utilizando o algoritmo MPS.

Passo 6. Se não foi determinada qualquer solução ou toda a região de interesse ficou completamente analisada, então terminar.

Passo 7. Se a solução encontrada já existe ou é dominada então voltar ao Passo 5.

Passo 8. Se a solução encontrada está fora de um dos limites impostos então voltar ao Passo 5; caso contrário, acrescentar a solução ao conjunto das não dominadas.

Passo 9. Caso o AD pretende determinar mais soluções, voltar ao Passo 5.

Algoritmo 10. Bi-objectivo : pesquisa numa Zona de Desnível de Dualidade.

Sempre que se encontra uma nova solução na Zona de Desnível de Dualidade, a região onde se podem localizar soluções não dominadas (que são representadas por triângulos — Fig. 18) é actualizada. Desta forma, para se verificar se uma determinada solução pertence a essa Zona de Desnível de Dualidade (Passo 8), basta verificar se pertence a um dos vários triângulos que traduzem aquelas regiões.

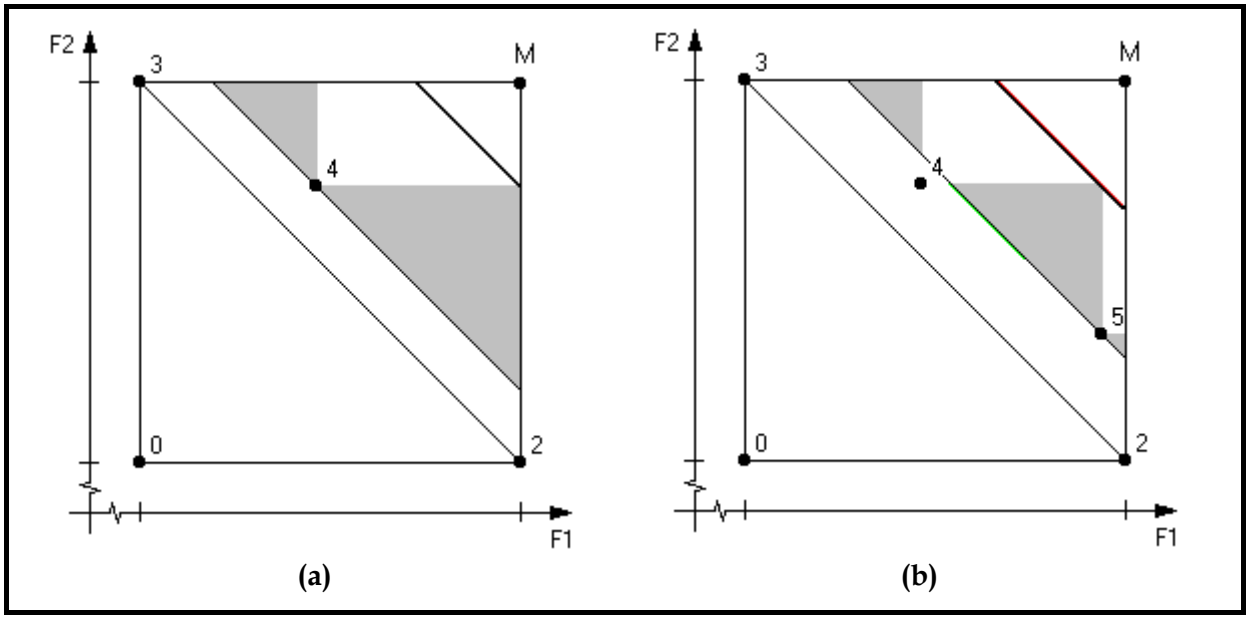


Fig. 18 – Bi-objectivo : pesquisa numa Zona de Desnível de Dualidade.

A Fig. 18 apresenta um possível desenvolvimento da pesquisa de soluções não dominadas na Zona de Desnível de Dualidade definida pelas soluções 2 e 3. Inicialmente, a região onde se podem localizar soluções não dominadas é representada apenas pelo triângulo cujos vértices são os pontos 2, 3 e M. Depois de ser determinada a solução 4, aquela região é substituída por duas (regiões a cheio na Fig. 18.(a)). Note-se que abaixo do segmento de recta que passa por 4 (e paralela a $\overline{23}$) não foram encontradas soluções e qualquer solução que se encontre no rectângulo que contém 4 e M como vértices é dominada pela solução 4. Depois de ter sido determinada a solução 5, as duas regiões são substituídas por três (regiões a cheio na Fig. 18.(b)).

A linha mais carregada que se encontra na Fig. 18.(a) e (b), que é paralela ao segmento $\overline{23}$ e passa pelo ponto mais afastado das regiões onde se podem localizar soluções não dominadas relativamente ao segmento $\overline{23}$, indica o limite a partir do qual não existem mais soluções não dominadas. Ou seja, como as soluções são determinadas segundo a direcção de pesquisa associada ao gradiente da recta que passa pelos vértices 2 e 3, se for determinada

uma solução que se encontre para além daquela linha, isso significa que toda a região onde é possível se encontrar soluções não dominadas ficou completamente analisada; logo, não existem mais soluções não dominadas nesta Zona de Desnível de Dualidade.

Refira-se que caso sejam impostas restrições nos valores das funções objectivo, esta linha também é determinada à custa dessas restrições, como se pode observar na Fig. 19.

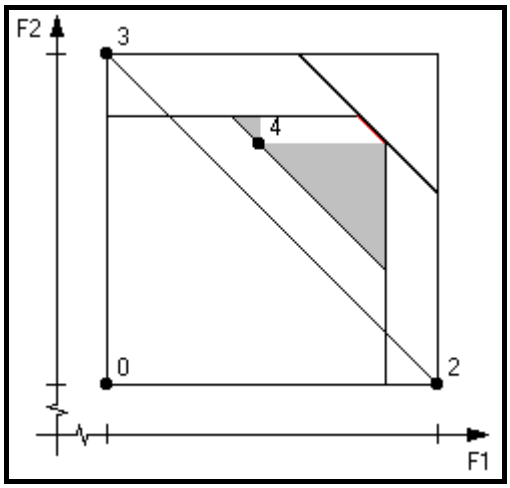


Fig. 19 – Bi-objectivo : definição da região de interesse.

Comparando a Fig. 18.(a) com a Fig. 19, verifica-se que a região de interesse sofreu alteração (diminuiu) com a introdução de restrições nos valores das funções objectivo.

6.2. Problema tri-objectivo

Os passos essenciais para determinar soluções não dominadas de uma determinada Zona de Desnível de Dualidade, encontram-se descritos detalhadamente em Algoritmo 11.

Passo 1. Indicar os três vértices que definem a Zona de Desnível de Dualidade a analisar (têm de ser diferentes e definitivamente adjacentes).

Passo 2. Questionar o AD se pretende impor restrições nos valores das funções objectivo. Tomando estas restrições constrói-se a região de interesse.

Passo 3. Construir a função escalar soma ponderada, cujos pesos correspondem ao gradiente do plano que contém os vértices indicados no Passo 1.

Passo 4. Determinar a primeira solução não dominada, resolvendo o problema P5, utilizando o algoritmo MPS. Saltar para o Passo 6.

Passo 5. Determinar a próxima solução (k-ésimo caminho mais curto) do problema P5, utilizando o algoritmo MPS.

Passo 6. Se não foi determinada qualquer solução ou toda a região de interesse ficou completamente analisada, então terminar.

Passo 7. Se a solução encontrada já existe ou é dominada, então voltar ao Passo 5.

Passo 8. Se a solução encontrada não pertence à região de interesse, então acrescentá-la ao conjunto das não dominadas que estão fora da região de interesse e voltar ao Passo 5. Caso contrário, acrescentar a solução ao conjunto das não dominadas

Passo 9. Caso o AD pretenda determinar mais soluções, voltar ao Passo 5.

Algoritmo 11. Tri-objectivo : pesquisa numa Zona de Desnível de Dualidade.

Para se verificar se uma solução é dominada (Passo 7), tem-se em conta todas as soluções não dominadas já encontradas : aquelas que pertencem e que não pertencem à Zona de Desnível de Dualidade a analisar.

Uma situação que se pode considerar muito particular, é aquela em que no Contorno Convexo apenas foram encontradas duas soluções. Nesta caso, apesar de não existir qualquer Zona de Desnível de Dualidade, é possível determinar mais soluções não dominadas. Para tal, a direcção de pesquisa corresponde a uma combinação de pesos onde dois dos valores são nulos.

7. Métodos interactivos para encontrar uma solução final

Nesta secção são descritos dois métodos interactivos associados aos dois processos apresentados na abordagem proposta. Estes métodos destinam-se a encontrar soluções não dominadas do problema, até que o AD decida terminar com esta pesquisa, visto ter encontrado uma solução que o satisfaz (melhor solução de compromisso).

No entanto, enquanto que no primeiro método as soluções são encontradas de acordo com uma determinada direcção de pesquisa (que é calculada), no segundo elas são determinadas em duas fases distintas, consoante pertençam ao Contorno Convexo (ver secções 5.1 e 5.2 deste capítulo) ou a Zonas de Desníveis de Dualidade (ver secções 6.1 e 6.2 deste capítulo), possibilitando, desta forma, orientar a procura de soluções não dominadas para regiões do espaço dos objectivos que o AD considere que poderão conter uma solução satisfatória, o que não acontece com o primeiro método, visto neste apenas existir uma região em análise, que é todo o espaço dos objectivos.

Por outro lado, o segundo método permite determinar, ora soluções de um tipo, ora do outro, não havendo obrigatoriedade de primeiro se determinarem todas as soluções do Contorno Convexo (vértices) e só depois analisar as Zonas de Desníveis de Dualidade pretendidas. Ou seja, depois de se analisar algumas Zonas de Desníveis de Dualidade, pode-se regressar à análise do Contorno Convexo, se ainda existirem soluções deste tipo por encontrar. No entanto, sempre que se queira analisar uma Zona de Desnível de Dualidade, as soluções que a caracterizam terão que estar identificadas (o que é feito ao analisar-se o Contorno Convexo).

Para melhor se perceber a abordagem apresentada, são desenvolvidos a seguir dois exemplos : um para o problema bi-objectivo e outro para o problema tri-objectivo. No desenvolvimento destes exemplos, colocamo-nos no papel de um AD hipotético no sentido de ilustrar o funcionamento dos métodos e dos utensílios gráficos colocados à sua disposição.

7.1. Problema bi-objectivo

Para ilustrar o funcionamento da abordagem apresentada, gerou-se aleatoriamente uma rede não orientada com 20 nós e 100 arcos, em que de cada nó saem 5 arcos e cada arco tem associado dois valores correspondentes a dois objectivos. Pretende-se determinar a "melhor" solução de compromisso, associada ao caminho entre os nós 1 e 19 (**s** = 1 e **t** = 19), que mais de perto corresponda às preferências do AD, em presença dos dois objectivos.

7.1.1. Procura em todo o espaço dos objectivos

Os valores obtidos no passo inicial deste método, com vista à determinação da direcção de pesquisa das soluções, foram os seguintes : 1037 (mínimo do objectivo 1) e 392 (mínimo do objectivo 2). Desta forma, a função escalar utilizada foi construída a partir dos seguintes pesos : λ_1 = 0.275 (= 392/(1037+392)) e λ_2 = 0.725 (= 1 – 0.275), como descrito na secção 4 deste capítulo.

A Fig. 20 apresenta o resultado da primeira pesquisa de soluções não dominadas, que corresponde à determinação da primeira solução, assim como a solução ideal :

Solução 1 = [1, 11, 16, 19] ==> (1956, 392)

Solução 0 = [ ] ==> (1037, 392) — solução ideal.

A região a cheio indica as zonas onde ainda é possível encontrar soluções não dominadas.

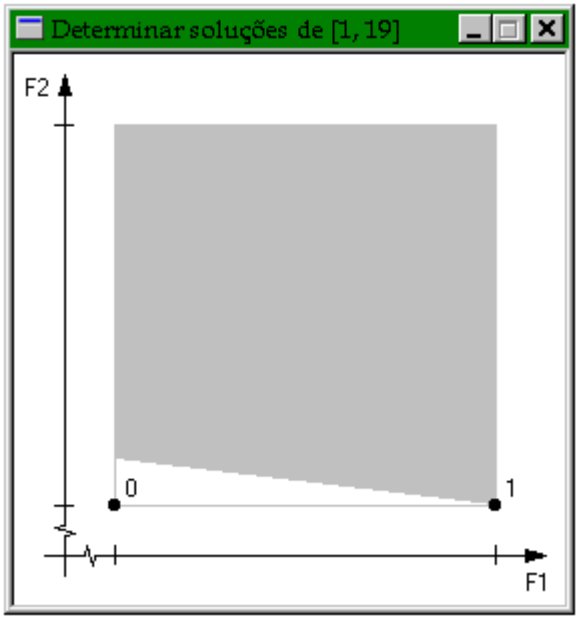


Fig. 20 – Bi-objectivo : primeira pesquisa no Espaço Total.

Como a solução obtida não satisfaz o AD, este decidiu realizar uma nova pesquisa, da qual resultou a solução 2 (Fig. 21) :

Solução 2 = [1, 8, 19] ==> (1671, 532).

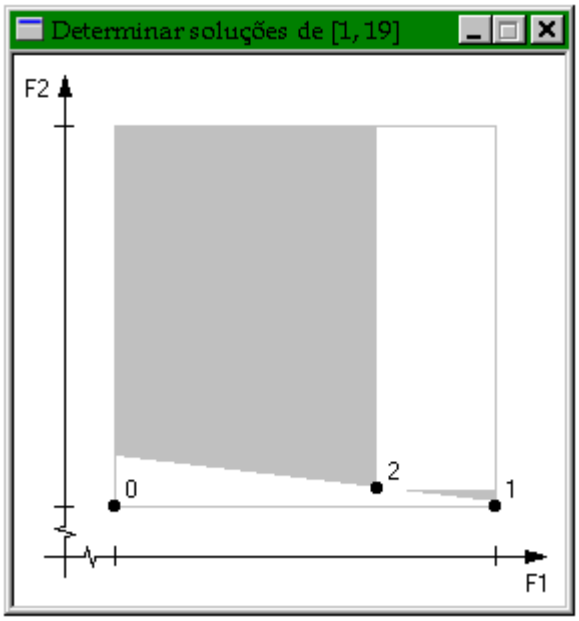


Fig. 21 – Bi-objectivo : segunda pesquisa no Espaço Total.

Como o AD não se sente satisfeito com as soluções conhecidas, resolveu efectuar mais uma pesquisa, da qual resultou a solução 3 (Fig. 22) :

Solução 3 = [1, 10, 14, 19] ==> (1488, 1373).

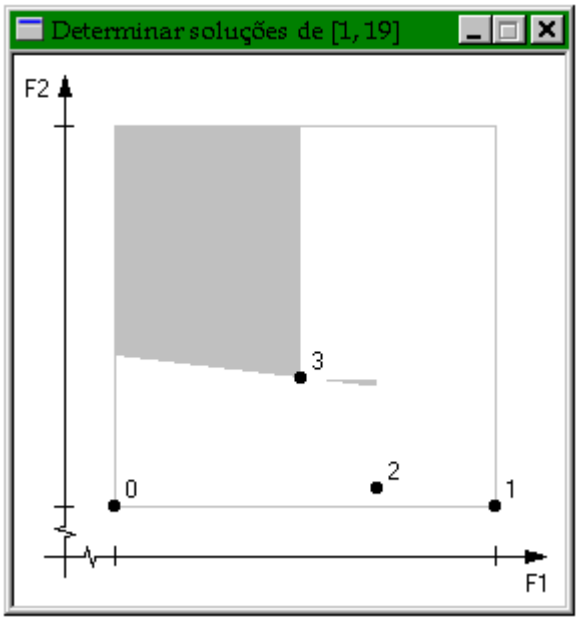


Fig. 22 – Bi-objectivo : terceira pesquisa no Espaço Total.

Apesar desta última solução o satisfazer razoavelmente, o AD decidiu efectuar mais uma pesquisa, da qual foi encontrada a solução 4 (Fig. 23) :

Solução 4 = [1, 6, 3, 4, 19] ==> (1360, 2170).

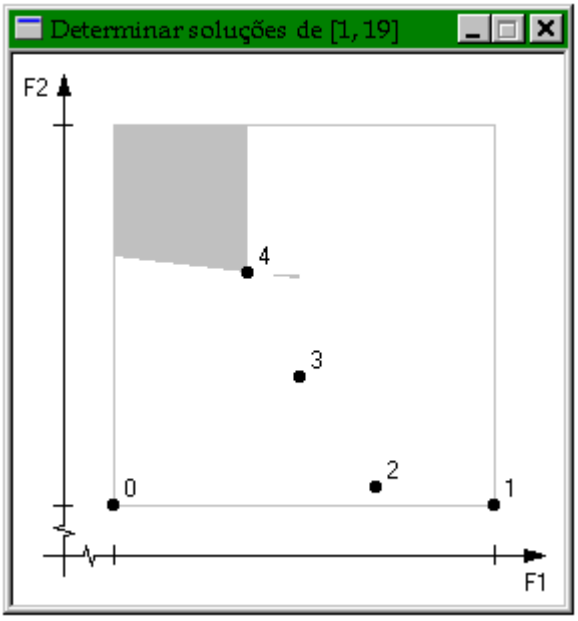


Fig. 23 – Bi-objectivo : quarta pesquisa no Espaço Total.

Analisando esta última solução e a região onde é possível se encontrar soluções não dominadas, o AD decidiu terminar com a pesquisa de soluções não dominadas, escolhendo como "melhor" solução de compromisso, a solução 3 : [1, 10, 14, 19] ==> (1488, 1373).

7.1.2. Procura no Contorno Convexo e em Zonas de Desníveis de Dualidade

A Fig. 24 apresenta as duas primeiras soluções do Contorno Convexo (vértices), que optimizam cada uma das funções objectivo separadamente e a solução ideal :

Solução 1 = [1, 5, 13, 4, 19] ==> (1037, 3296) — optimiza o objectivo 1

Solução 2 = [1, 11, 16, 19] ==> (1956, 392) — optimiza o objectivo 2

Solução 0 = [ ] ==> (1037, 392) — solução ideal.

Analisando as soluções 1 e 2, conclui-se que qualquer solução não dominada terá que estar situada no rectângulo cujos vértices são estas duas soluções, a solução ideal e o ponto formado pelos valores máximos correspondentes aquelas duas soluções, que neste caso terá os valores de 1956 (máximo de F1) e 3296 (máximo de F2). Por outro lado, qualquer solução do Contorno Convexo, se existir, encontra-se abaixo do segmento $\overline{12}$, mais precisamente no interior do triângulo formado pelas soluções 0, 1 e 2 (região a cheio na Fig. 24).

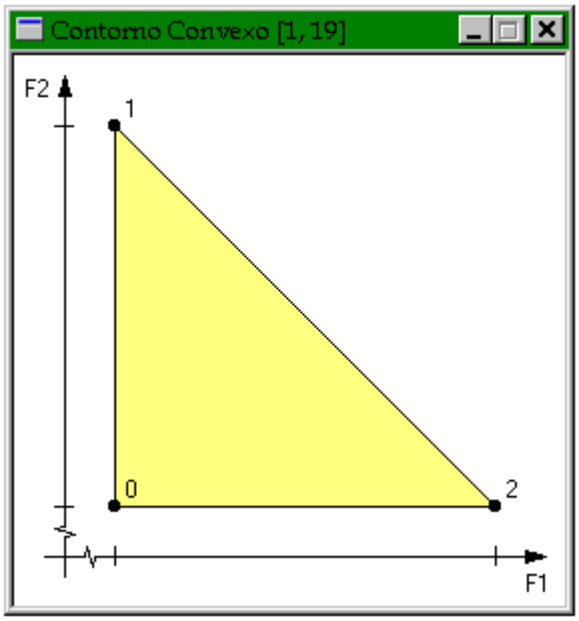


Fig. 24 – Bi-objectivo : primeira pesquisa no Contorno Convexo.

Como qualquer uma destas duas soluções não satisfaz o AD, este decidiu procurar um outro vértice, utilizando como direcção de pesquisa o gradiente da recta que passa pelos vértices 1 e 2 (que são adjacentes). Desta pesquisa, resultou o vértice 3 (Fig. 25) :

Solução 3 = [1, 8, 19] ==> (1671, 532).

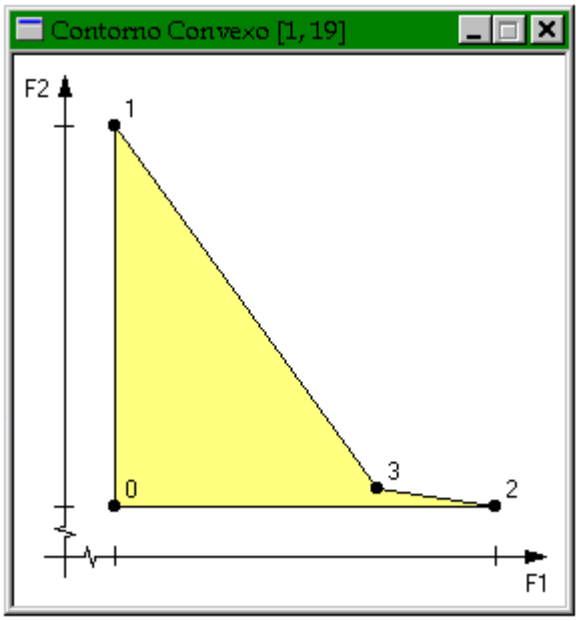


Fig. 25 – Bi-objectivo : primeira pesquisa no Contorno Convexo.

Com a determinação da solução 3, os vértices 1 e 2 deixaram de ser adjacentes entre si, passando a existir 2 combinações de vértices adjacentes : (1, 3) e (3, 2) — Fig. 25.

Como as soluções existentes ainda não satisfazem o AD e as Zonas de Desníveis de Dualidade que possam ser construídas a partir dos vértices 2 e 3 também não lhe agradam, decidiu-se realizar mais uma pesquisa de vértices, utilizando como direcção o gradiente da recta que passa pelos vértices 1 e 3. Desta pesquisa foi encontrado o vértice 4 (Fig. 26) :

Solução 4 = [1, 11, 3, 4, 19] ==> (1136, 2372).

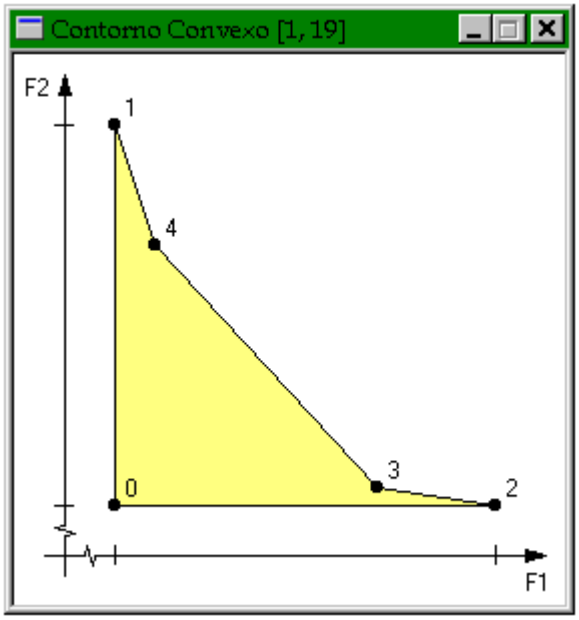


Fig. 26 – Bi-objectivo : segunda pesquisa no Contorno Convexo.

O AD ao analisar as soluções já encontradas, decidiu eliminar de futuras pesquisas as regiões acima do vértice 4 e à direita do vértice 3. Ou seja, apenas continuar a pesquisa entre as soluções 3 e 4, identificada como a região onde uma solução final assume valores interessantes para ambos os objectivos.

Desta forma, continuou a pesquisa de vértices entre as soluções 3 e 4, da qual não resultou qualquer nova solução (Fig. 27), o que significa que estes dois vértices são definitivamente adjacentes, e portanto, constituem uma Zona de Desnível de Dualidade.

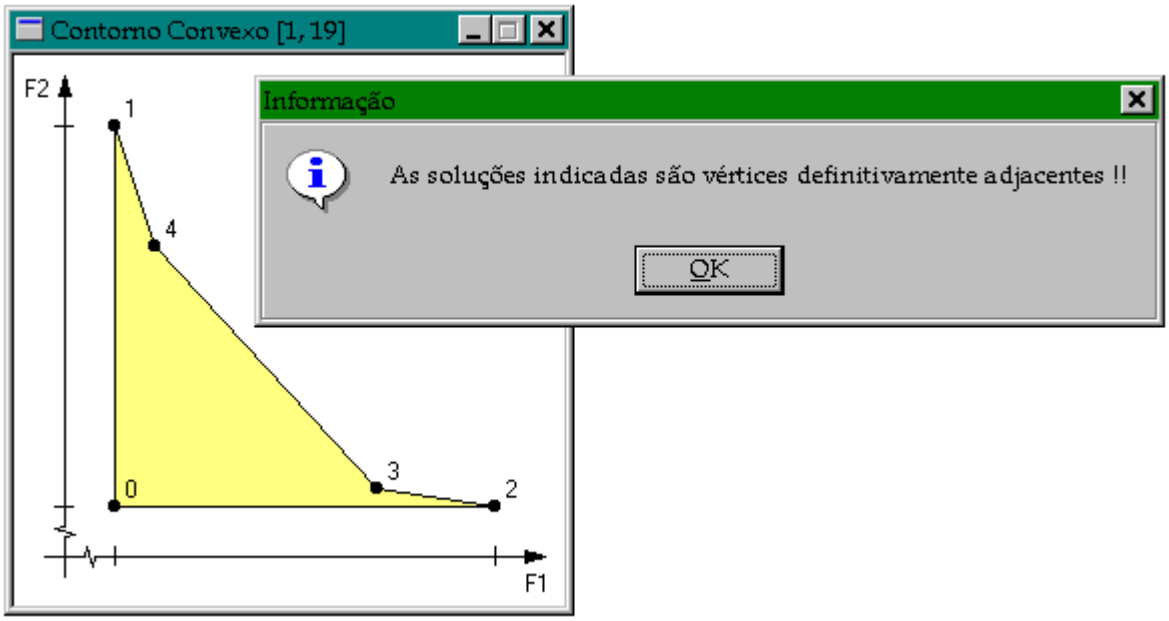


Fig. 27 – Bi-objectivo : quarta pesquisa no Contorno Convexo.

Como nesta região já não existem mais vértices, o AD decidiu pesquisar mais soluções não dominadas, mas agora na Zona de Desnível de Dualidade definida pelos vértices 3 e 4. Inicialmente foi encontrada a solução 5 (Fig. 28) :

Solução 5 = [1, 8, 12, 4, 19] ==> (1203, 2349).

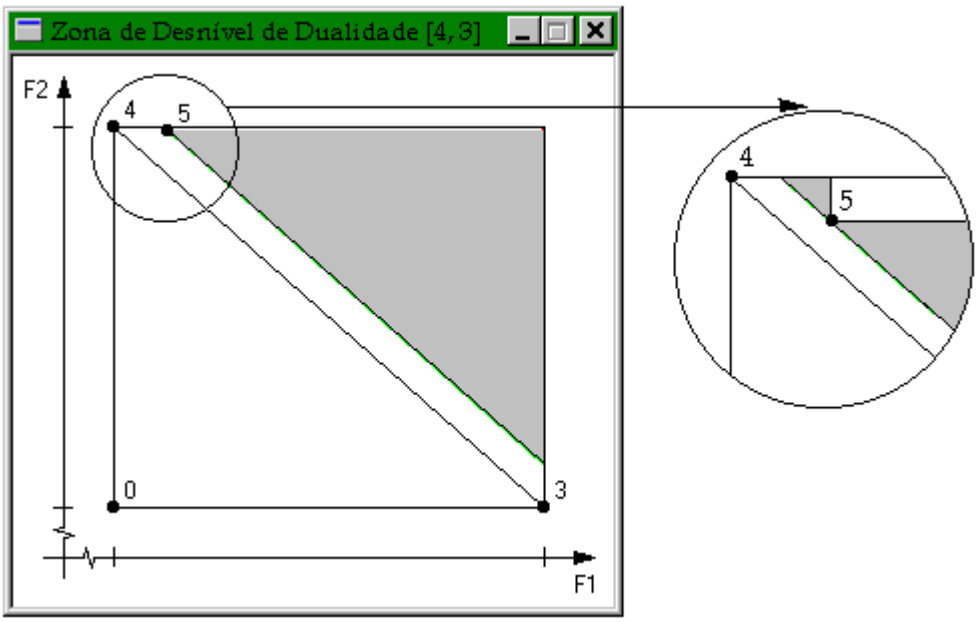


Fig. 28 – Bi-objectivo : primeira pesquisa numa Zona de Desnível de Dualidade.

Como a região onde é possível encontrar soluções não dominadas foi alterada, pois inicialmente era representada pelo triângulo cujos vértices eram as soluções 3 e 4 e o ponto definido pelos valores máximos de cada objectivo relativos às soluções 3 e 4 (as que definem a Zona de Desnível de Dualidade), que tem os valores de 1671 (objectivo 1) e 2372 (objectivo 2), é necessário actualizar esta região. Desta forma, existem agora duas regiões (Fig. 28 – "Zoom"), pois abaixo da linha que passa pela solução 5, paralela ao segmento $\overline{34}$ não foi encontrada qualquer solução e o rectângulo a branco que se encontra acima e à direita da solução 5, corresponde a uma zona onde qualquer solução que ali se localize é dominada pela solução 5.

Suponhamos que o AD não se encontra ainda satisfeito com esta solução, pelo que indicou uma nova pesquisa nesta Zona de Desnível de Dualidade. O resultado foi a determinação da solução 6 (Fig. 29) :

Solução 6 = [1, 10, 14, 19] ==> (1488, 1373).

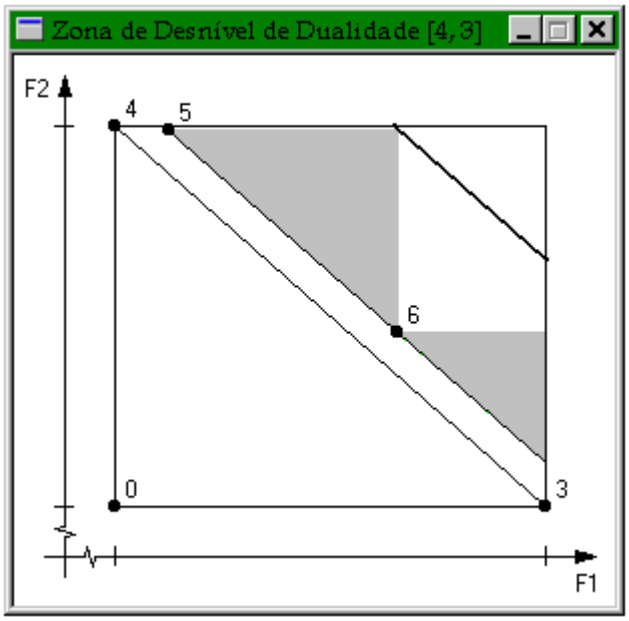


Fig. 29 – Bi-objectivo : segunda pesquisa numa Zona de Desnível de Dualidade.

Com a determinação da solução 6, as regiões onde é possível calcular novas soluções não dominadas sofreram alteração, como mostra o gráfico da Fig. 29. A linha mais carregada, que é paralela à linha que passa pelas soluções 3 e 4, corresponde ao limite dessas regiões, tal que se uma solução estiver para além dessa linha, isso significa que todas as soluções não dominadas desta Zona de Desnível de Dualidade foram encontradas. Inicialmente esta linha passava pelo ponto formado pelos valores máximos de cada objectivo (1671, 2372),

correspondentes às duas soluções que definem a Zona de Desnível de Dualidade (3 e 4), sendo depois sucessivamente actualizada, à medida que se calculam novas soluções não dominadas.

Se se prosseguir a pesquisa de soluções nesta Zona de Desnível de Dualidade, obtém-se a solução 7 (Fig. 30) :

Solução 7 = [1, 6, 3, 4, 19] ==> (1360, 2170).

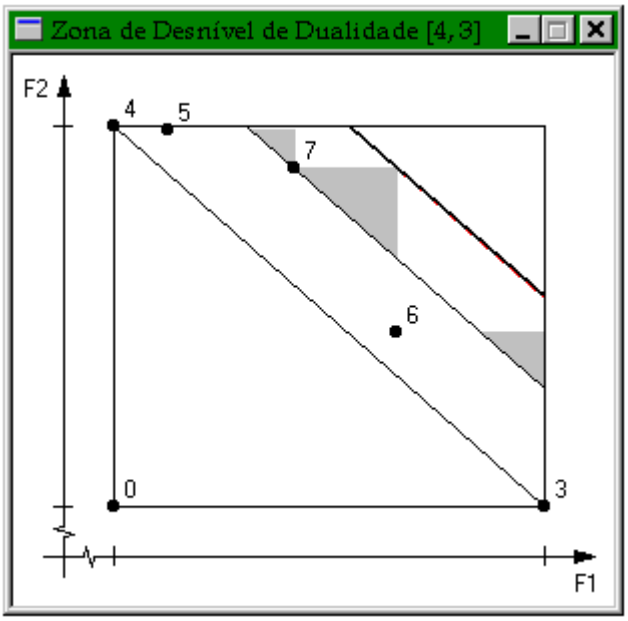


Fig. 30 – Bi-objectivo : terceira pesquisa numa Zona de Desnível de Dualidade.

Ao analisar a solução encontrada e as novas regiões onde se podem localizar novas soluções não dominadas, o AD decidiu escolher a solução 6 como a "melhor" solução de compromisso, aceitável como solução final.

No entanto, se o AD tivesse dúvidas relativamente a outras soluções encontradas noutras Zonas de Desníveis de Dualidade (o que não acontece) ou no Contorno Convexo, o AD tem a possibilidade de visualizar todas as soluções num único gráfico (Fig. 31.(a)) ou numa tabela (Fig. 31.(b)).

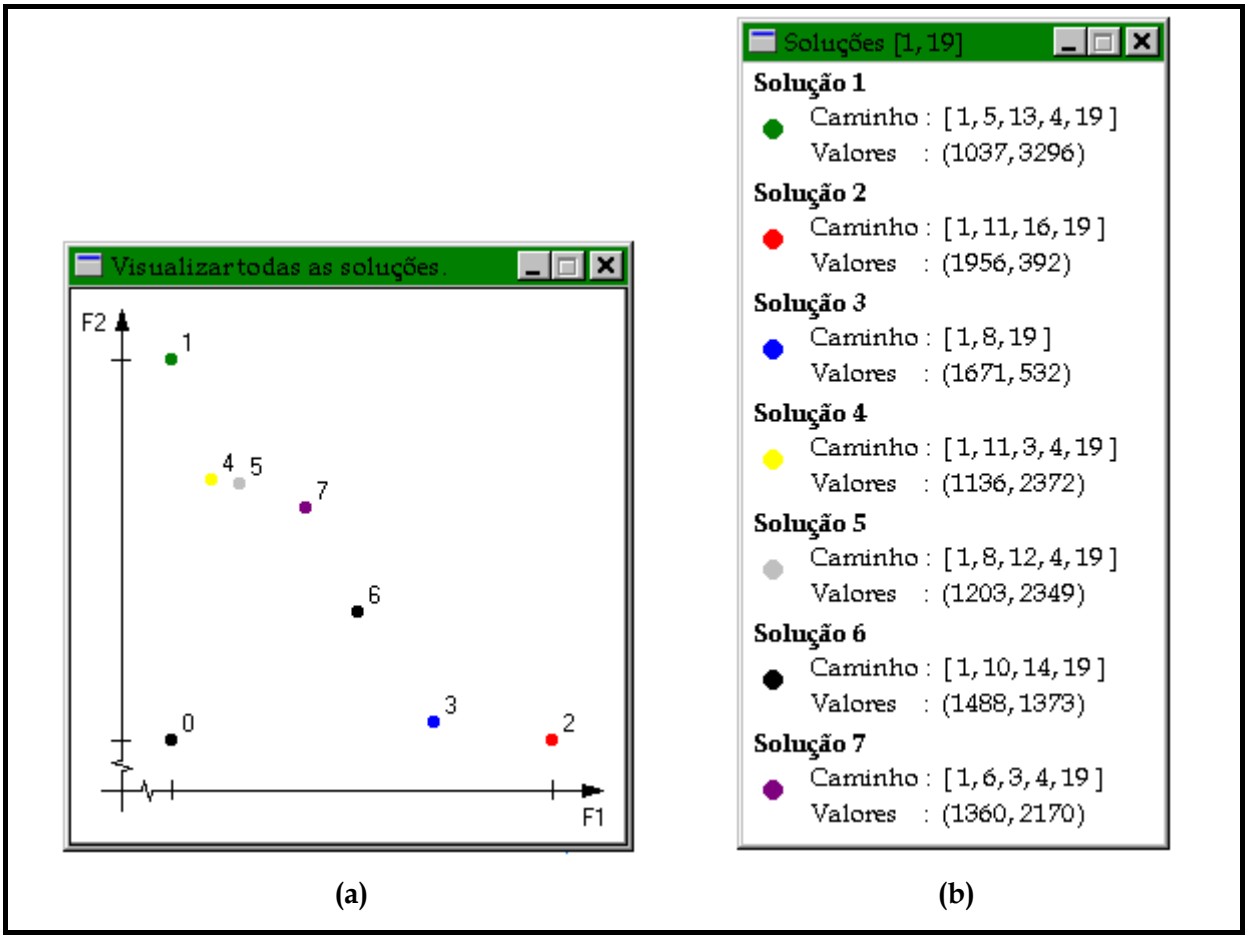


Fig. 31 – Bi-objectivo : **(a)** gráfico e **(b)** tabela com todas as soluções.

Refira-se ainda que, caso o AD não ficasse satisfeito com as soluções encontradas, podia regressar à pesquisa de mais soluções do Contorno Convexo, uma vez que é desconhecida qualquer outra Zona de Desnível de Dualidade que não seja a [4, 3], para posteriormente se poder analisar outras Zonas de Desníveis de Dualidade.

7.2. Problema tri-objectivo

Para ilustração foi gerada aleatoriamente uma rede não orientada com 50 nós e 125 arcos, em que a cada nó incidem 5 arcos e cada arco tem associado três valores correspondentes a três funções objectivo. Pretende-se então determinar uma solução de compromisso, correspondente a um caminho entre os nós 1 e 50 (**s** = 1 e **t** = 50).

7.2.1. Procura em todo o espaço dos objectivos

Os valores obtidos no passo inicial deste método, com vista à determinação da direcção de pesquisa das soluções, foram os seguintes : 1260 (mínimo do objectivo 1), 884 (mínimo do

objectivo 2) e 527 (mínimo do objectivo 3). A função escalar utilizada foi construída a partir dos seguintes pesos : λ_1 = 0.2076, λ_2 = 0.2960 e λ_3 = 0.4964, como descrito na secção 4 deste capítulo.

A Fig. 32 apresenta o resultado da primeira pesquisa de soluções não dominadas em todo o espaço dos objectivos do problema :

Solução 1 = [1, 34, 24, 50] ==> (1260, 1734, 1733).

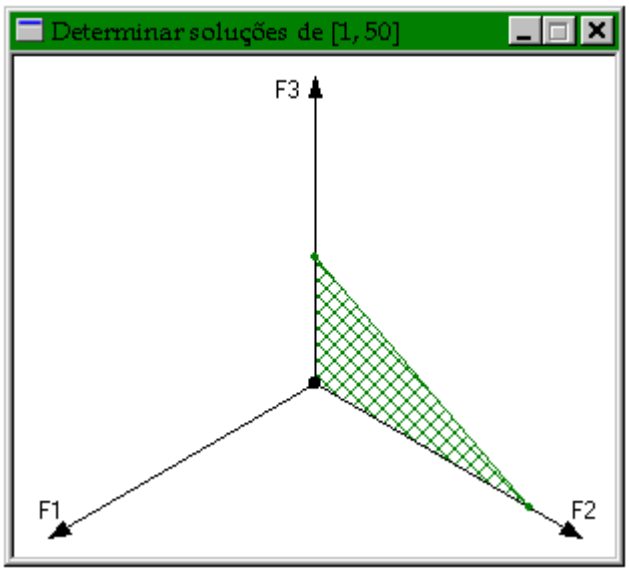


Fig. 32 – Tri-objectivo : primeira pesquisa no Espaço Total.

Como a solução encontrada não agrada o AD, este decidiu efectuar mais uma pesquisa de soluções não dominadas no espaço dos objectivos, a qual resultou na determinação de uma outra solução (Fig. 33) :

Solução 2 (vermelha) = [1, 28, 29, 2, 50] ==> (2013, 1371, 1790).

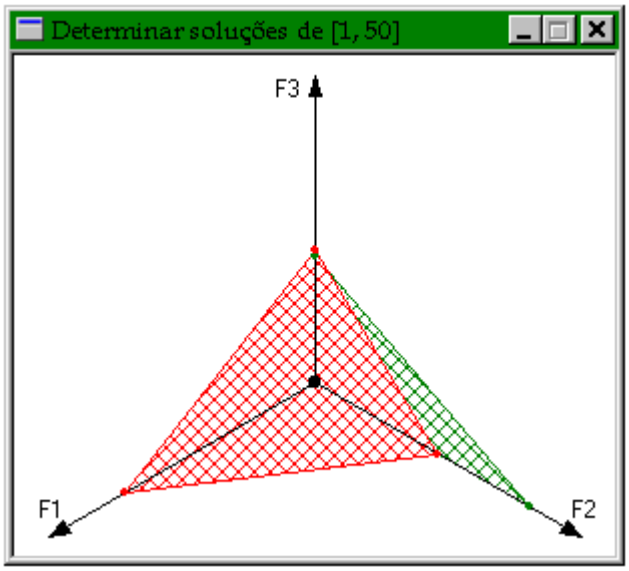


Fig. 33 – Tri-objectivo : segunda pesquisa no Espaço Total.

Também esta última solução não satisfaz o AD, pelo que decidiu efectuar mais uma pesquisa, da qual resultou mais uma solução (Fig. 34) :

Solução 3 (azul) = [1, 28, 29, 17, 50] ==> (2092, 903, 2203).

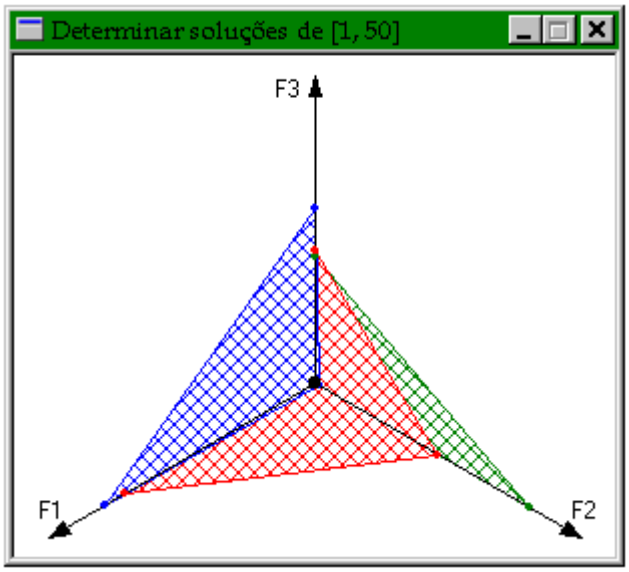


Fig. 34 – Tri-objectivo : terceira pesquisa no Espaço Total.

Pela análise das três soluções não dominadas já encontradas, o AD resolveu efectuar mais uma pesquisa, pois qualquer daquelas soluções não o satisfaz. O resultado desta pesquisa, foi a determinação de uma outra solução (Fig. 35) :

Solução 4 (amarela) = [1, 13, 47, 48, 24, 50] ==> (4381, 2160, 527).

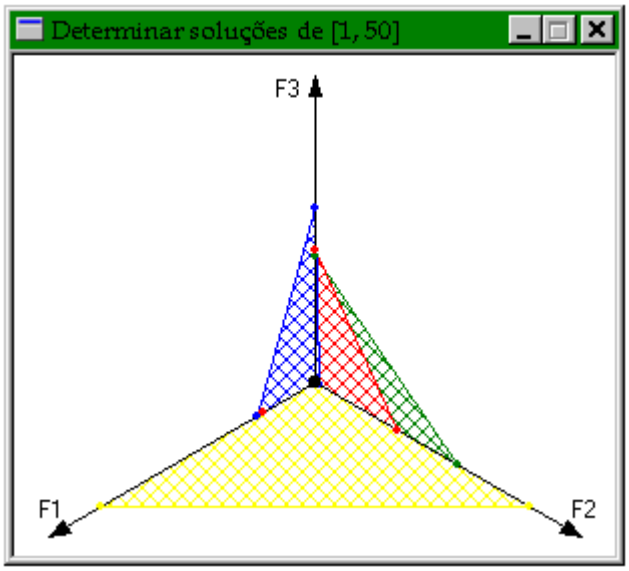


Fig. 35 – Tri-objectivo : quarta pesquisa no Espaço Total.

Como também esta última solução não agrada ao AD, este decidiu realizar mais uma pesquisa de soluções não dominadas, na qual foi encontrada mais uma solução (Fig. 36) :

Solução 5 (prateada) = [1, 34, 11, 17, 50] ==> (2455, 1322, 2011).

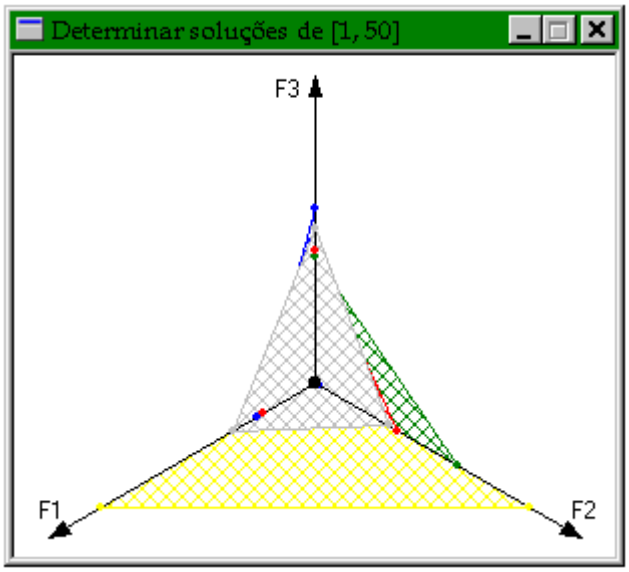


Fig. 36 – Tri-objectivo : quinta pesquisa no Espaço Total.

Numa primeira análise à solução encontrada, o AD sente-se razoavelmente satisfeito com ela. No entanto, para se escolher uma solução final, geralmente não basta apenas analisar cada solução individualmente, mas é necessário compará-la com as restantes. Para tal, as soluções devem ser analisadas em gráficos onde estejam todas representadas, e onde seja

possível verificar os compromissos entre os objectivos. Desta forma, e apesar de no gráfico da Fig. 36 estarem representadas todas as soluções não dominadas já encontradas, outras formas de visualizar a informação estão representadas na Fig. 37.

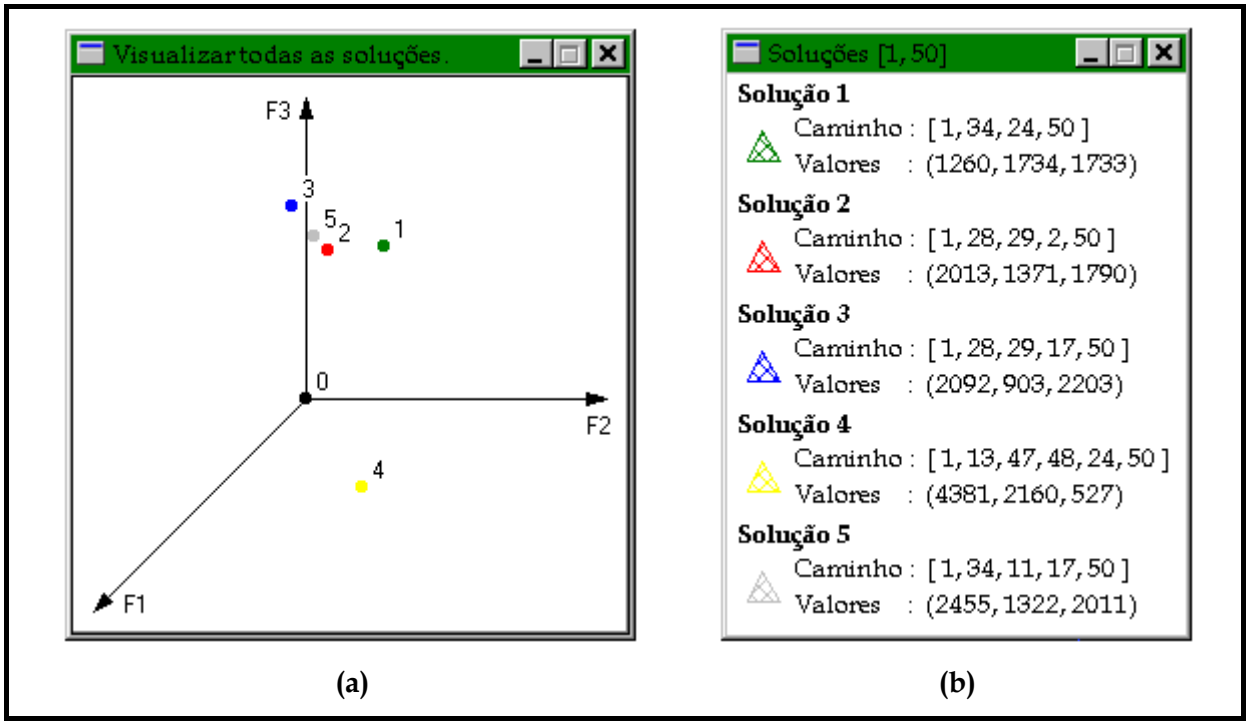


Fig. 37 – Tri-objectivo : **(a)** gráfico e **(b)** tabela com todas as soluções.

Também é possível analisar-se as soluções numa outra perspectiva, como por exemplo em projecções nos três planos ortogonais possíveis, como se pode ver na Fig. 38.

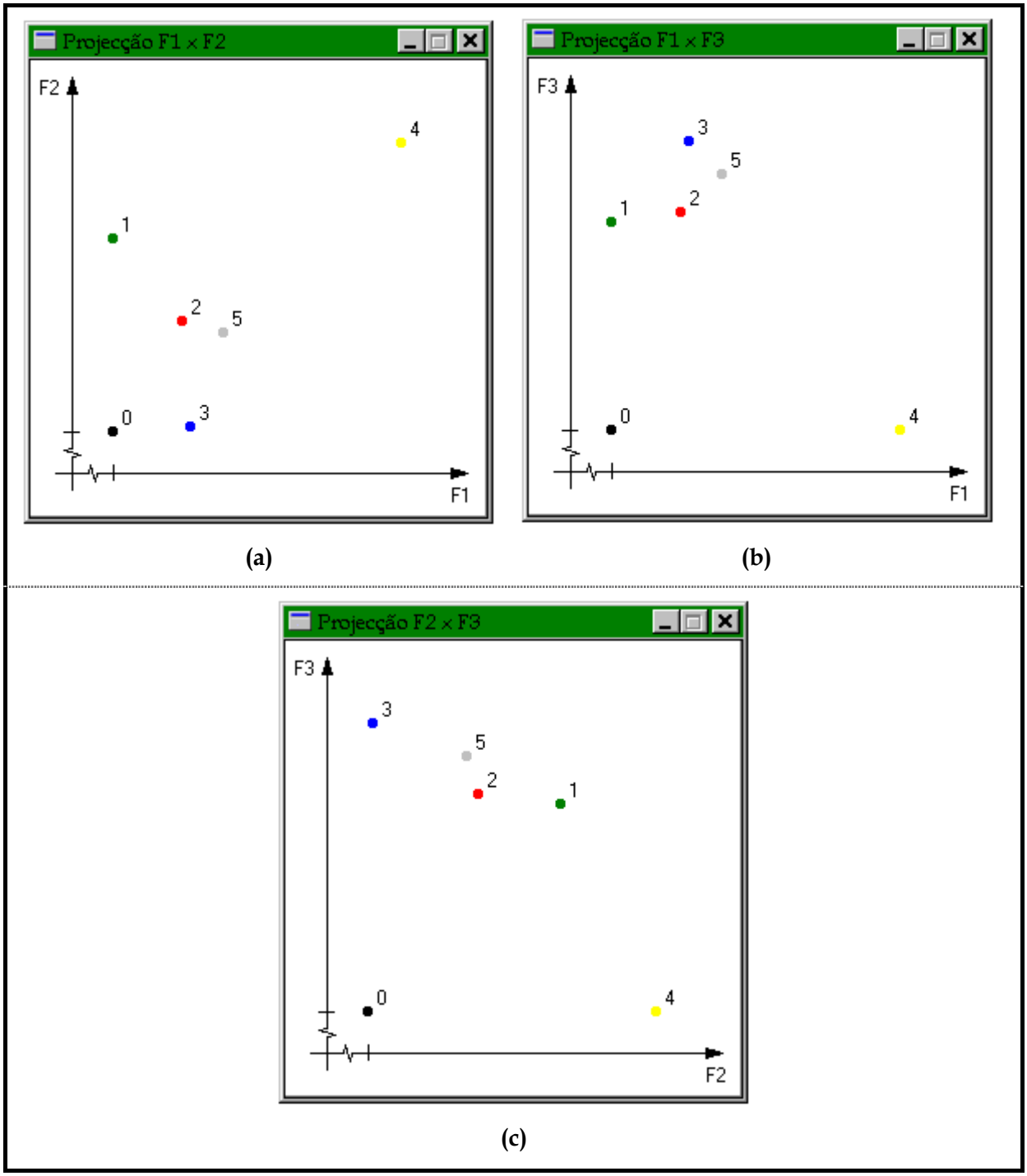


Fig. 38 – Tri-objectivo : projecção em **(a)** F1×F2, **(b)** F1×F3 e **(c)** F2×F3.

Pela análise das cinco soluções não dominadas já encontradas, o AD resolveu efectuar mais uma pesquisa. O resultado foi a determinação de uma outra solução (Fig. 39) :

Solução 6 (preta) = [1, 28, 36, 7, 50] ==> (1725, 1516, 2220).

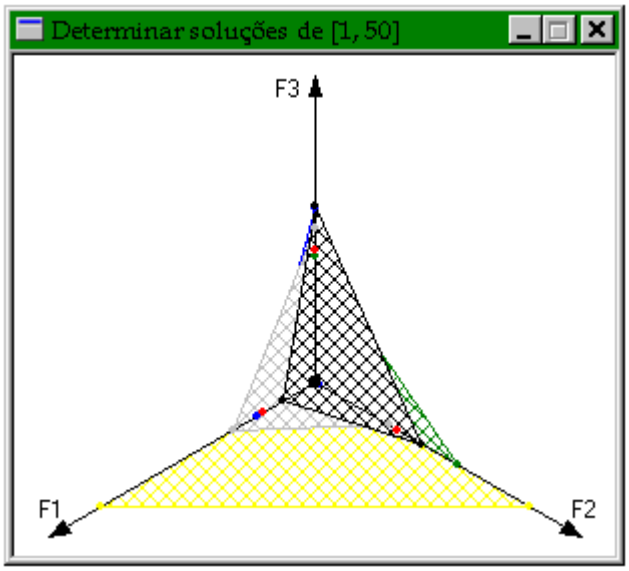


Fig. 39 – Tri-objectivo : sexta pesquisa no Espaço Total.

Com este resultado, o AD resolveu terminar com a pesquisa de soluções não dominadas em todo o espaço dos objectivos, uma vez que esta última solução o satisfaz, não havendo necessidade de prosseguir com a pesquisa. Desta forma, a solução que melhor reflecte as suas preferências é a solução 6.

7.2.2. Procura no Contorno Convexo e em Zonas de Desníveis de Dualidade

A Fig. 40 apresenta as três primeiras soluções do Contorno Convexo (vértices), que optimizam cada um dos objectivos separadamente e a solução ideal :

Solução 1 = [1, 34, 24, 50] ==> (1260, 1734, 1733)
Solução 2 = [1, 21, 8, 2, 50] ==> (2106, 884, 2885)
Solução 3 = [1, 13, 47, 48, 24, 50] ==> (4381, 2160, 527)
Solução 0 = [ ] ==> (1260, 884, 527).

O ponto central do gráfico corresponde à solução ideal, a verde está representada a solução 1 (óptimo do objectivo 1), a vermelho a solução 2 (óptimo do objectivo 2) e a azul a solução 3 (óptimo do objectivo 3). Note-se que qualquer destas soluções ocupa apenas um dos planos do gráfico, pois a outra coordenada é igual a um dos valores do ponto central (solução ideal).

Ao contrário do problema bi-objectivo, neste caso nada se pode concluir a partir dos três primeiros vértices (1, 2 e 3), relativamente à gama de valores que as soluções não dominadas podem atingir. De facto, pela análise destas três soluções, apenas se conhecem os valores mínimos que qualquer função pode atingir, mas em relação aos valores máximos (não dominados), isso já não é possível.

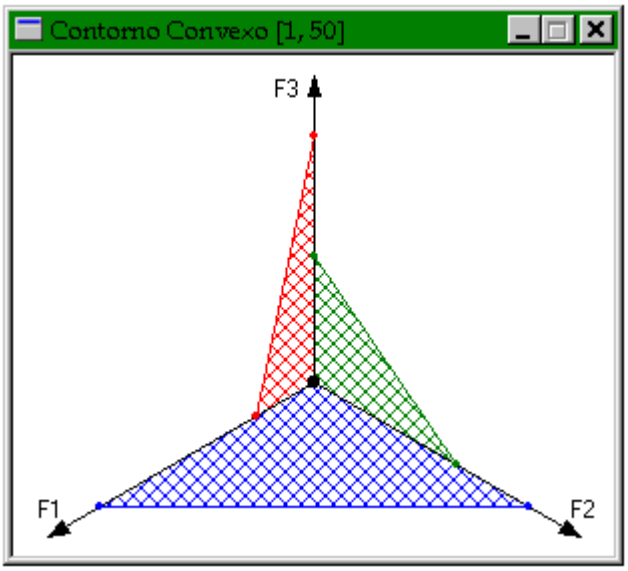


Fig. 40 – Tri-objectivo : primeira pesquisa no Contorno Convexo.

Como qualquer uma destas três soluções não satisfaz o AD, decidiu-se procurar um outro vértice, utilizando como direcção de pesquisa o gradiente do plano que contém os vértices 1, 2 e 3 (são adjacentes). Desta pesquisa resultou um outro vértice (Fig. 41) :

Solução 4 (amarelo) = [1, 28, 29, 17, 50] ==> (2092, 903, 2203).

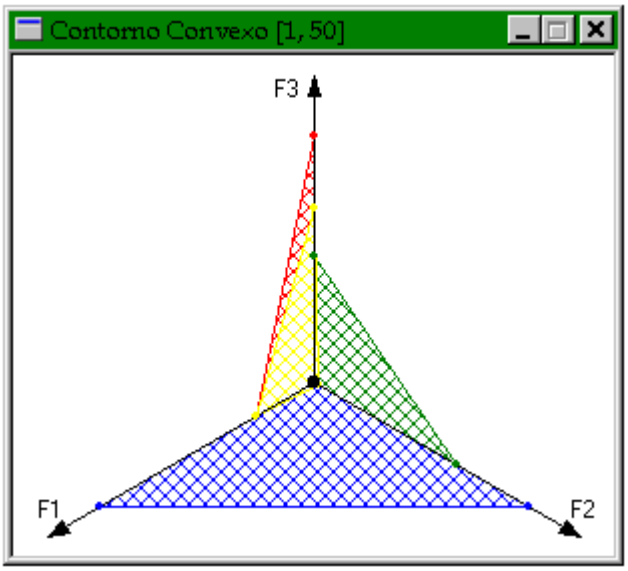


Fig. 41 – Tri-objectivo : segunda pesquisa no Contorno Convexo.

Com a determinação do novo vértice, 1, 2 e 3 deixaram de ser adjacentes entre si, passando a existir três combinações de vértices adjacentes : (1, 2, 4), (1, 3, 4) e (2, 3, 4).

Como as soluções existentes ainda não satisfazem o AD, decidiu-se pesquisar um outro vértice, utilizando o gradiente do plano que contém os vértices 1, 2 e 4 como direcção de pesquisa, da qual não resultou qualquer solução. Desta forma, aqueles três vértices são definitivamente adjacentes entre si, formando uma Zona de Desnível de Dualidade : [1, 2, 4].

Em consequência do resultado anterior, o AD decidiu procurar outro vértice, escolhendo como direcção de pesquisa o gradiente formado pelo plano que contém os vértices 1, 3 e 4. Desta pesquisa (Fig. 42), resultou o vértice 5 (prateado) :

Solução 5 = [1, 28, 29, 2, 50] ==> (2013, 1371, 1790).

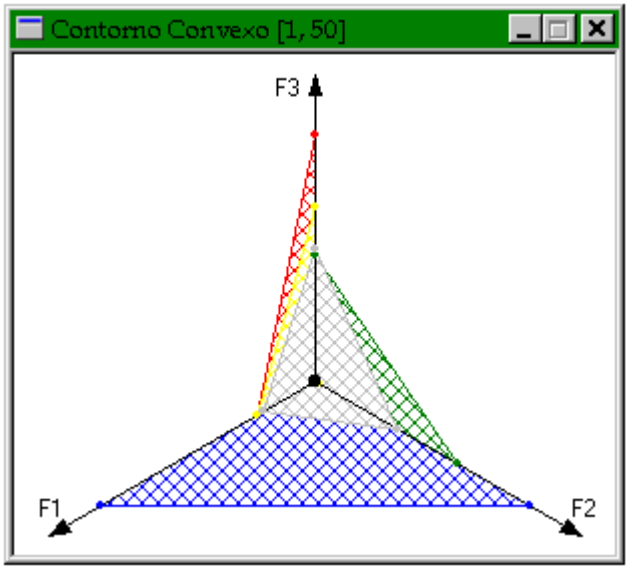


Fig. 42 – Tri-objectivo : terceira pesquisa no Contorno Convexo.

Como foi encontrado um vértice, os vértices 1, 3 e 4 deixaram de ser adjacentes entre si, passando a existir sete combinações de vértices adjacentes : (2, 3, 4), (1, 2, 5), (1, 3, 5), (1, 4, 5), (2, 3, 5), (2, 4, 5) e (3, 4, 5), e uma definitivamente adjacente : (1, 2, 4).

O AD, após analisar as soluções já encontradas, decidiu procurar mais soluções do Contorno Convexo. Utilizando o gradiente do plano que contém os vértices 1, 2 e 5, como direcção de pesquisa de mais soluções, foi encontrado um vértice já existente, mas diferente dos que definem o referido plano. Isto implica que aqueles vértices não são definitivamente adjacentes, não definindo, desta forma, uma Zona de Desnível de Dualidade. O mesmo resultado foi obtido quando se utilizaram os gradientes dos planos que contêm os vértices 2, 3 e 5, e 2, 4 e 5. Portanto, estas combinações de vértices também não constituem Zonas de Desníveis de Dualidade.

Resultado sensivelmente diferente foi obtido quando se utilizaram, como direcções de pesquisas, os gradientes dos planos que contêm os vértices 1, 3 e 5; 1, 4 e 5; 2, 3 e 4; 3, 4 e 5. De

facto, com qualquer uma daquelas direcções, apesar de não se determinar qualquer novo vértice, foi encontrado um dos vértices que constitui o respectivo plano. Desta forma, e como as componentes do gradiente são todas positivas, os vértices que pertencem a cada uma daquelas combinações são definitivamente adjacentes entre si, constituindo, assim, uma Zona de Desnível de Dualidade. Portanto, para além da já conhecida Zona de Desnível de Dualidade [1, 2, 4], existem mais quatro : [1, 3, 5], [1, 4, 5], [2, 3, 4] e [3, 4, 5].

A Fig. 43 mostra as soluções do Contorno Convexo (vértices) do problema que foram encontradas, as quais são apresentadas num gráfico (a) e numa tabela (b).

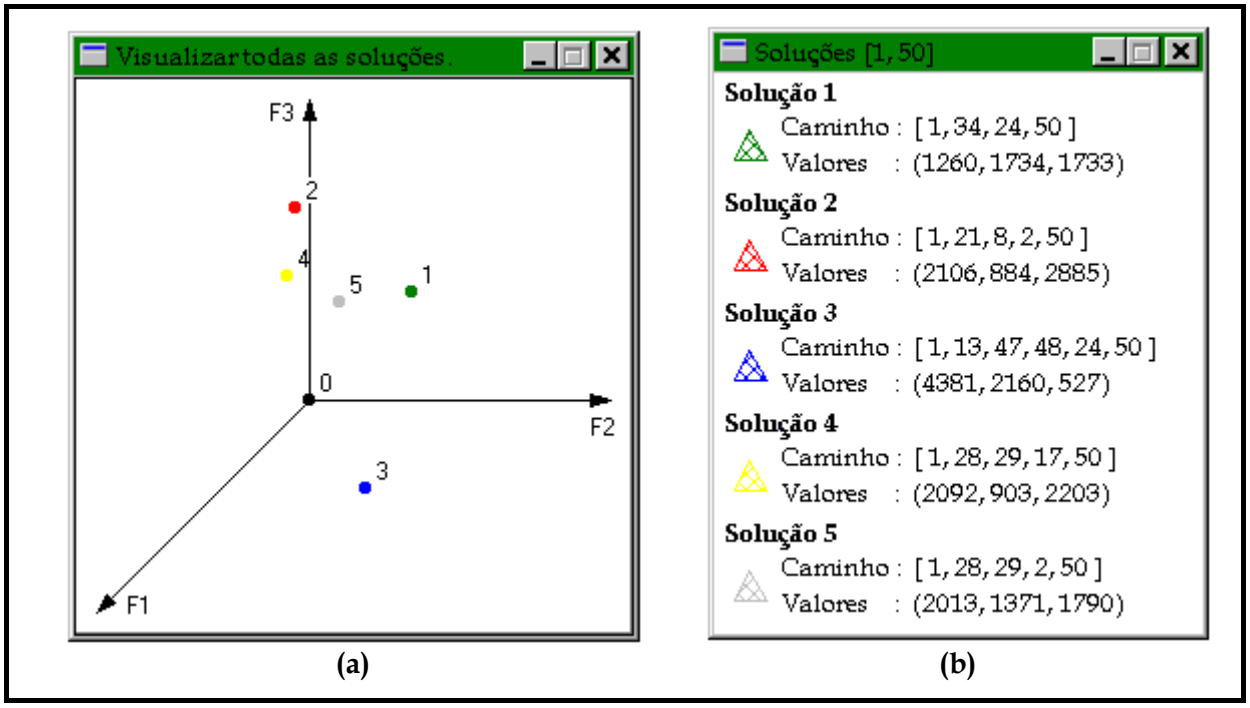


Fig. 43 – Tri-objectivo : **(a)** gráfico e **(b)** tabela com todos os vértices.

Observando a Fig. 43.(a) pode-se concluir que se as soluções ali representadas fossem projectadas no plano F2 = 0, talvez se percebesse melhor a sua amplitude. Para tal, analise-se a Fig. 44, onde os vértices estão projectados no plano F1×F3 (F2=0).

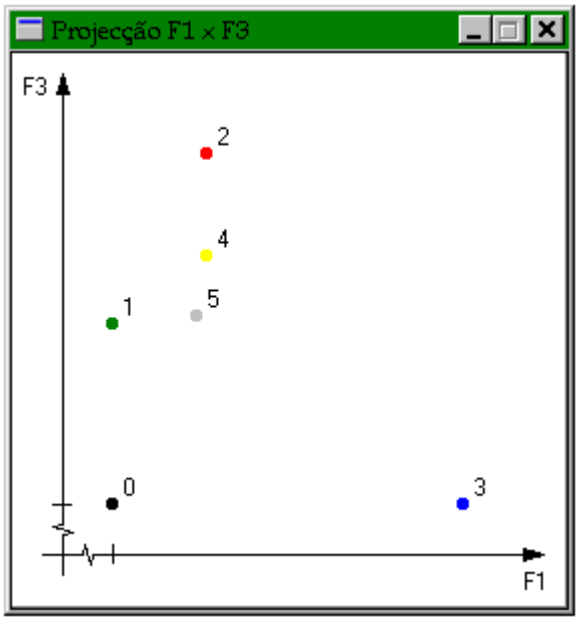


Fig. 44 – Tri-objectivo : projecção dos vértices no plano F2 = 0.

Ao analisar as soluções já encontradas, o AD decidiu pesquisar mais soluções não dominadas, mas agora numa das Zonas de Desníveis de Dualidade existentes (uma vez que não é possível determinar mais soluções do Contorno Convexo), que traduza a região que mais lhe interessa. Desta forma, resolveu analisar a Zona de Desnível de Dualidade definida pelos vértices 1, 2 e 4. Na primeira pesquisa de soluções não dominadas foi encontrada uma nova solução (Fig. 45) :

Solução 6 (preta) = [1, 28, 36, 7, 50] ==> (1725, 1516, 2220).

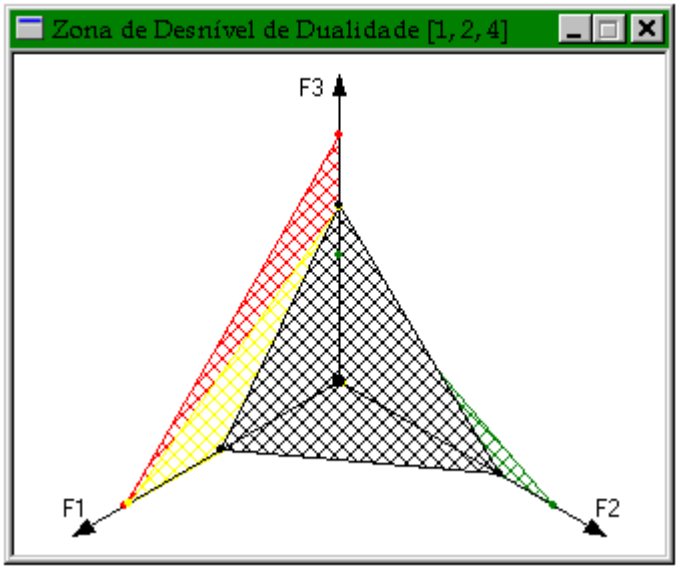


Fig. 45 – Tri-objectivo : primeira pesquisa numa Zona de Desnível de Dualidade.

Apesar desta solução satisfazer razoavelmente o AD, este indicou uma nova pesquisa nesta Zona de Desnível de Dualidade, a qual não teve êxito, uma vez que não existem mais soluções não dominadas nesta Zona de Desnível de Dualidade (Fig. 46).

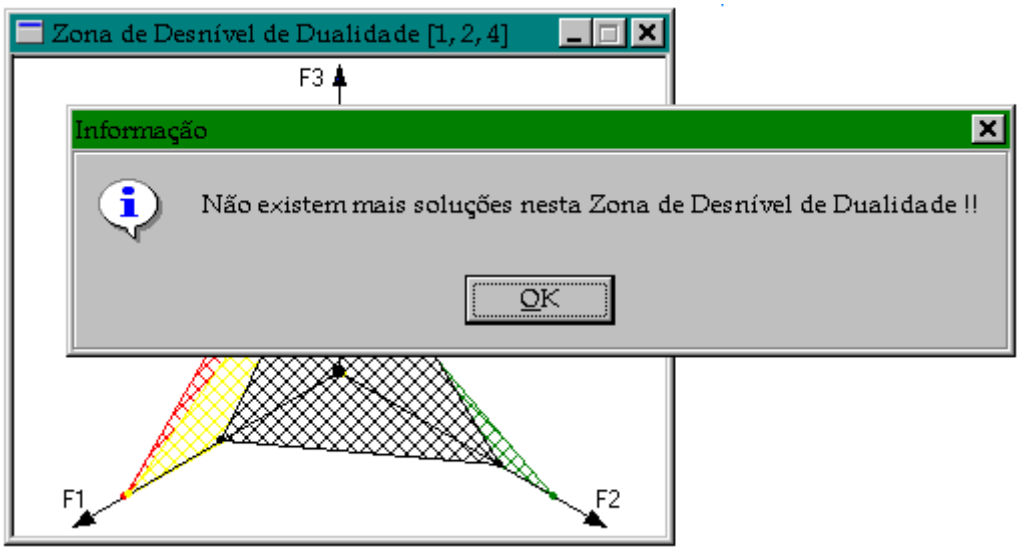


Fig. 46 – Tri-objectivo : segunda pesquisa numa Zona de Desnível de Dualidade.

Apesar da solução 6 satisfazer o AD, podendo desta forma escolhê-la como solução final, suponhamos que ele decidiu fazer uma nova pesquisa de soluções numa outra Zona de Desnível de Dualidade, esperando que possa ser encontrada uma solução que lhe agrade ainda mais. Desta forma, resolveu analisar a Zona de Desnível de Dualidade definida pelos vértices 3, 4 e 5. O resultando foi a determinação de um nova solução (Fig. 47) :

Solução 7 (lilás) = [1, 34, 11, 17, 50] ==> (2455, 1322, 2011).

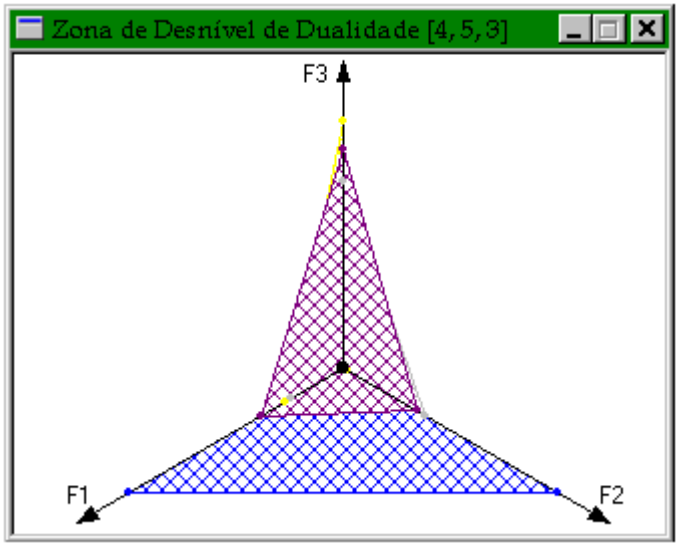


Fig. 47 – Tri-objectivo : primeira pesquisa numa Zona de Desnível de Dualidade.

Ao analisar a solução encontrada, o AD decidiu escolher a solução 7 como solução final, ou seja, de acordo com as suas preferências, esta é a melhor solução de compromisso encontrada.

No entanto, se tivesse dúvidas relativamente a outras soluções encontradas noutras Zonas de Desníveis de Dualidade ou no Contorno Convexo, o AD tinha a possibilidade de visualizar todas as soluções num único gráfico (Fig. 48.(a)) ou numa tabela (Fig. 48.(b)).

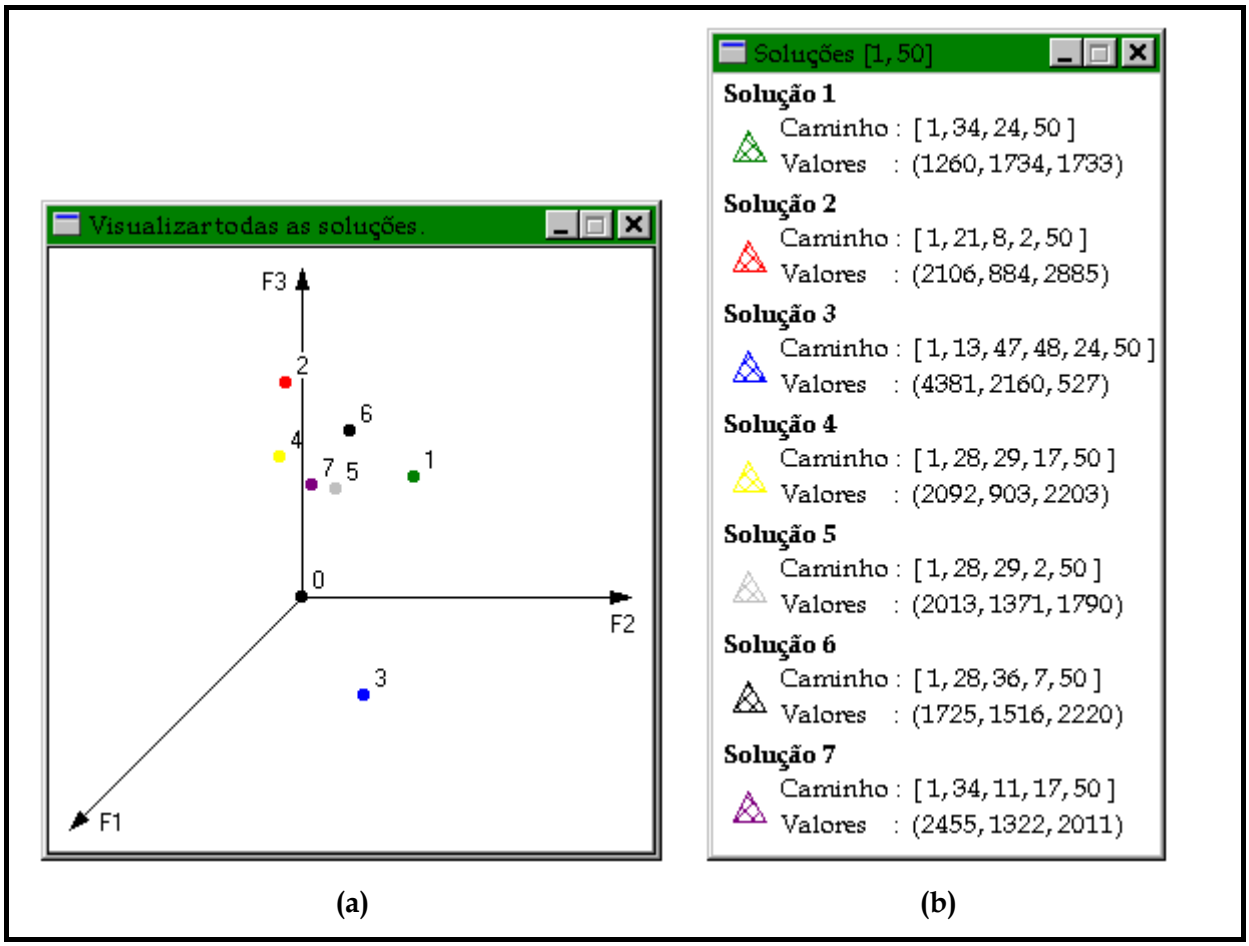


Fig. 48 – Tri-objectivo : **(a)** gráfico e **(b)** tabela com todas as soluções.

Ou então, podia ainda analisar as soluções não dominadas encontradas vistas segundo um outro prisma, como é o caso das projecções. Ou seja, pode-se analisar as soluções mediante apenas os valores de quaisquer dois objectivos : considerando os objectivos 1 e 2 (Fig. 49.(a)), os objectivos 1 e 3 (Fig. 49.(b)) e os objectivos 2 e 3 (Fig. 49.(c)).

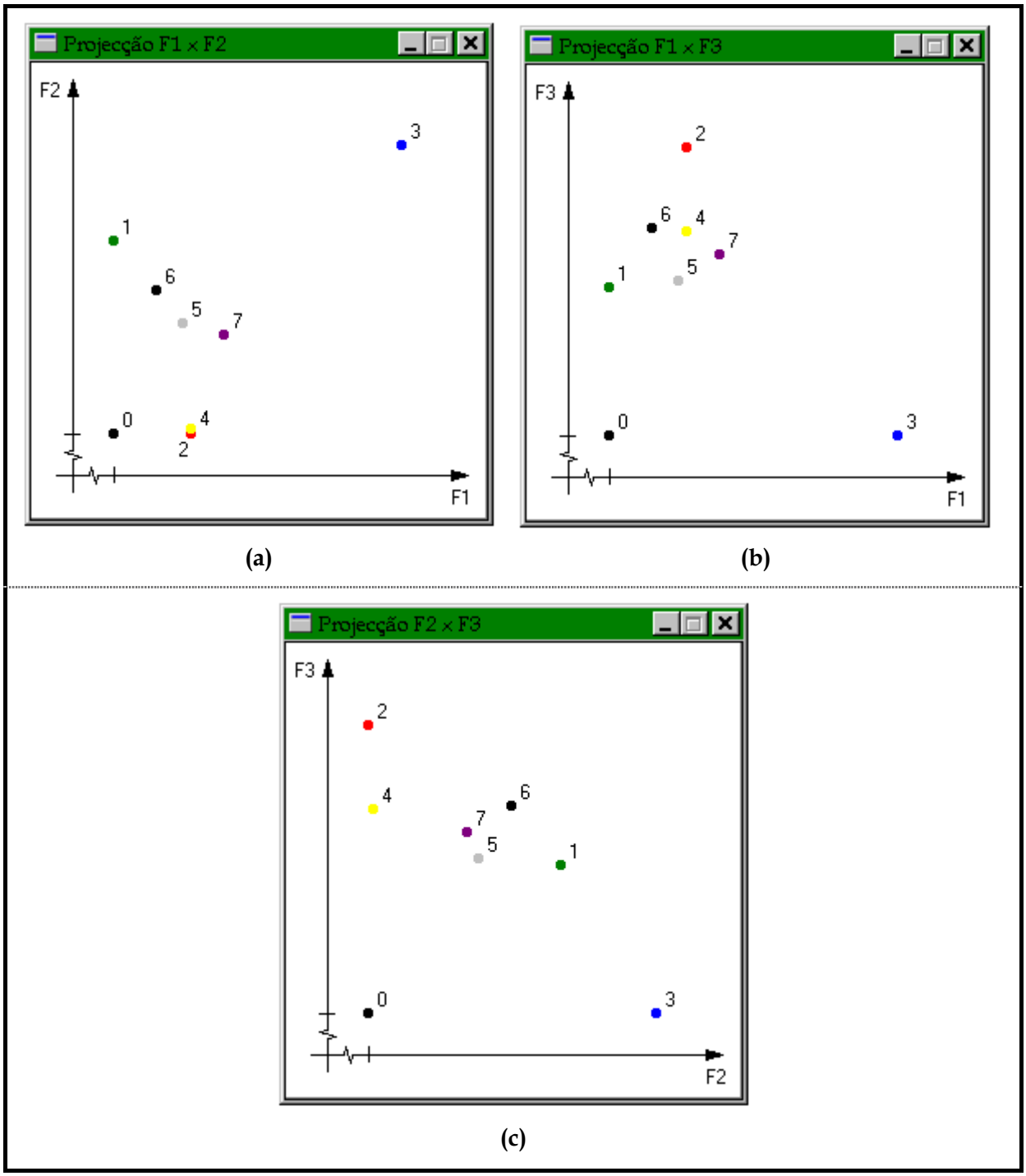


Fig. 49 – Tri-objectivo : projecção em **(a)** F1×F2, **(b)** F1×F3 e **(c)** F2×F3.

Por outro lado, refira-se que, caso o AD não ficasse satisfeito com as soluções que foram encontradas, podia continuar com a pesquisa na Zona de Desnível de Dualidade ([3, 4, 5]), ou então analisar noutras Zonas de Desníveis de Dualidade.

C A P Í T U L O 6

Aplicação ao Problema de Encaminhamento em Redes Integradas de Comunicações

1. Introdução

As tradicionais arquitecturas de redes de comunicações, foram estruturadas para suportar utilizadores com requisitos de qualidade de serviço ("Quality of Service" — QoS) simples e homogéneos [16]. No entanto, com o aumento da procura dos diversos serviços, as tecnologias de rede que floresceram num ambiente de economia de escala (isto é, quanto maior a dimensão, menor o custo unitário) são, gradualmente, substituídas por novas tecnologias que oferecem economia de alcance (isto é, abrangendo muitas áreas). As modernas redes de comunicações devem poder adaptar-se a utilizadores com diversos requisitos de QoS, com elevado grau de granularidade e subjectivos. Esta tarefa é realizada através de um conjunto de mecanismos de controlo da rede, que actuam em várias escalas de tempo.

O desempenho de uma rede depende de vários factores, tais como a configuração, o número de dados que pode ser transferido e os métodos de gestão da rede. Numa rede de telecomunicações, as funções da rede são responsáveis pela transferência de informação entre dois utilizadores finais, seleccionando e fixando os recursos de rede ao longo de um caminho. Uma **chamada** é a associação lógica entre a transmissão e os utilizadores finais. Uma **ligação** é a "cadeia" de recursos de rede que suporta uma chamada [16].

Considerem-se as operações sobre uma rede em tempo real. Sempre que uma chamada chega a um ponto da rede, é necessário determinar se existe um caminho através do qual a chamada pode ser encaminhada para o seu destino. Se for determinado um caminho, é então

necessário decidir se é ou não utilizado; se não existe nenhum caminho disponível, tem que se decidir o que fazer com a chamada. Cada passo destes está relacionado com um aspecto diferente do encaminhamento.

O **encaminhamento** é um elemento muito importante na gestão de redes, que consiste num conjunto de regras de decisão (geralmente denominada por técnica de encaminhamento) para ligar as chamadas, quando chegam a um ponto da rede. Desta forma, o encaminhamento está relacionado com operações sobre a rede, em tempo real, e com um mecanismo de controlo ao nível das chamadas, através do qual é escolhido um dos caminhos disponíveis (se existir mais do que um) para estabelecer a comunicação entre dois pontos (origem e destino) da rede.

Alguns dos tradicionais algoritmos de encaminhamento consistem em resolver problemas de caminho mais curto com um só objectivo, em que a função objectivo se baseia numa métrica simples, tal como custo, atraso, número de arcos, probabilidade de perda, etc., ou numa única função de diferentes métricas. No entanto, o problema torna-se mais complexo quando se introduzem restrições para satisfazer os vários requisitos de QoS.

Geralmente, estas restrições pertencem a duas categorias, consoante a entidade onde são impostas : nos arcos e nos caminhos. As **restrições nos arcos** são limitações na utilização dos arcos do caminho escolhido (por exemplo, a capacidade disponível num arco deve ser maior ou igual ao requerido pela chamada). As **restrições nos caminhos** são limites predefinidos de uma métrica de desempenho ao longo do caminho escolhido (por exemplo, o atraso ponto a ponto na rede não pode exceder o que a chamada pode tolerar) [16].

Devido à subjectividade dos actuais requisitos de QoS e aos complexos compromissos entre eles, tornou-se mais difícil definir uma adequada métrica de encaminhamento. Além disso, como as características dos vários tipos de tráfego são distintas, a mesma métrica não é aplicável a todos os casos.

Se, por um lado, não existe grande vantagem em ter um caminho em que a QoS associada seja melhor do que a especificada pelo utilizador, por outro, é muito difícil determinar um caminho com a QoS especificada pelo utilizador. Por esta razão, é que para redes integradas de comunicações, é necessário um novo paradigma de encaminhamento, que realce a procura de um caminho aceitável e que satisfaça os vários requisitos de QoS.

Devido à crescente procura de novos e mais sofisticados serviços de telecomunicações, entra em jogo a heterogeneidade dos requisitos de QoS, nas redes integradas de comunicações. Consequentemente, os operadores de redes não podem continuar a confiar nas

técnicas de encaminhamento que se baseiam numa métrica simples, sendo necessário considerar explicitamente métricas distintas nos algoritmos [16].

Geralmente, o problema da escolha do caminho é formulado como um problema de caminho mais curto com uma única função objectivo (uma única métrica ou uma função envolvendo diferentes métricas), cujos requisitos de QoS podem ser incorporados nestes modelos matemáticos, através da utilização de restrições adicionais. No entanto, e uma vez que os modelos matemáticos têm inerentemente uma estrutura de rede (podendo, desta forma, utilizar eficazmente algoritmos específicos e eficientes), a introdução de restrições adicionais destrói algumas propriedades interessantes, o que implica um elevado esforço computacional.

2. Tecnologias de encaminhamento

O encaminhamento alternativo é muito utilizado em redes telefónicas, para fornecer serviços fiáveis. Existem duas classes de algoritmos de encaminhamento alternativo : determinístico e aleatório.

"**D**ynamic **N**on-**H**ierarchical **R**outing" (DNHR) é um algoritmo determinístico, cuja atribuição de caminhos alternativos é estática e sujeita a alterações com as horas do dia [16].

No algoritmo denominado por "**D**ynamic **T**raffic **M**anagement" (DTM), também determinístico, as decisões de encaminhamento são produzidas em grupos de chamadas, em vez de chamada a chamada, permitindo-lhes testar um caminho directo e um caminho alternativo recomendado, com o maior número de circuitos livres [16].

Nos algoritmos de encaminhamento alternativo aleatórios, cada caminho é escolhido com uma dada probabilidade por chamada. A probabilidade atribuída a cada caminho é constantemente actualizada, de forma a reflectir a possibilidade do caminho ser testado com sucesso. Quanto maior for a possibilidade de um caminho alternativo ser testado com sucesso, maior é a probabilidade de ser escolhido. Na versão conhecida por "**D**ynamic **A**lternate **R**outing" (DAR) os encaminhamentos escolhidos são restabelecidos sempre que uma chamada falha.

Os protocolos de encaminhamento entre domínios, tal como o "**O**pen **S**hortest **P**ath **F**irst" (OSPF), suportam métricas de encaminhamento de diferentes tipos de serviço, baseadas em diferentes combinações de atraso, débito e segurança.

Nas actuais redes integradas, o encaminhamento sujeito a várias restrições nos caminhos (por exemplo, restrições no custo e no atraso), apesar de ser intratável, tem uma característica apetecível. É possível formular um problema de encaminhamento sujeito a várias restrições nos caminhos, como um problema de caminho mais curto multicritério, onde cada métrica de caminho com restrições é tomada como um objectivo do encaminhamento. Uma abordagem simples de encaminhamento multicritério, é supor que o caminho "óptimo" é não dominado, tal que qualquer outro caminho tem pelo menos para uma das funções objectivo, um valor pior do que o correspondente valor daquele caminho. Com esta suposição, pode-se gerar o conjunto de caminhos não dominados, e através de uma abordagem baseada em função utilidade determinar o caminho desejado.

3. Restrições de qualidade de serviço

A QoS pode ser considerada como um grau de adaptação para os critérios de serviço especificados pelo utilizador [16]. Muitas das estruturas de QoS existentes, tanto concentram os critérios de desempenho na gestão do tráfego, como fornecem requisitos de qualidade orientados pelo utilizador com elevado grau de granularidade. Um elemento importante na arquitectura de redes integradas é a capacidade de oferecer uma diversidade de requisitos de QoS às diferentes aplicações que utilizam a rede, e assim, tornar eficiente o uso dos recursos da rede.

Em [16] é proposto uma estrutura de QoS ao nível da chamada, na qual a QoS é especificada em termos de três classes de restrições, que podem depender do tipo de serviço da chamada : *restrições de desempenho* (por exemplo, atraso), *restrições de recursos* (por exemplo, transferência média, canal de segurança) e *restrições de prioridade* (por exemplo, estabelecimento de prioridades, retenção das prioridades).

Uma **restrição de desempenho** é uma restrição sobre uma medida de qualidade de informação completamente perceptível, transferida durante uma ligação [16]. Uma restrição de desempenho pode, ou não, ser negociada entre a rede e os utilizadores finais. Uma restrição negociável é especificada em termos de um conjunto de valores limitados por um valor requerido e um aceitável, em que o valor requerido é o nível de desempenho mais desejado, que o utilizador gostava de alcançar se os recursos estivessem logo disponíveis, e o valor aceitável é o nível de desempenho menos desejado, mas que o utilizador pode tolerar. Uma restrição não negociável é especificada em termos de apenas um valor aceitável. Apesar

de cada restrição de desempenho ser basicamente uma restrição de QoS dependente do caminho, ela pode ser implementada como um atributo de arco (por exemplo, débito) ou como um atributo de caminho (por exemplo, atraso). Um *atributo de arco* é um parâmetro de arco, considerado individualmente, servindo para determinar se um dado arco é aceitável para transportar uma dada ligação. Um *atributo de caminho* é o somatório de um parâmetro de arco ao longo de um dado caminho, servindo para verificar se o caminho é aceitável para transportar uma dada ligação [16].

Uma **restrição de recursos** é uma limitação na utilização de um dado tipo de recurso da rede, com um conjunto de características específicas [16]. As restrições de recurso, que estão sujeitas à definição do utilizador, podem estar directa ou indirectamente relacionadas com a QoS (por exemplo, segurança e escolha do transportador, respectivamente). Os recursos podem referir-se a elementos básicos da rede (por exemplo, arcos), ou aglomerados de elementos da rede (por exemplo, domínios administradores). A eficiência da topologia utilizada na escolha do caminho é determinada pela disponibilidade e aceitabilidade dos recursos da rede com vários atributos de recursos. Cada *atributo de recursos* (por exemplo, transmissão média) está associado a um conjunto de possíveis valores de atributos discretos (por exemplo, satélite, microondas). Uma restrição de recurso pode ser especificada em termos de um subconjunto deste conjunto. Um recurso da rede não é aceitável para uma chamada, a não ser que cada um dos valores dos atributos de recurso pertença ao correspondente conjunto de restrições de recurso. As restrições de recurso mais simples, que são predeterminados e não dependem do estado da rede, determinam, ou não, se um dado recurso é aceitável para o encaminhamento de uma chamada.

Uma **restrição de prioridade** é uma condição imposta na distribuição dos recursos da rede, para fornecer diferentes probabilidades de bloqueio ao tráfego das diferentes classes de prioridade. As restrições de prioridade podem ser implementadas em uma ou duas abordagens numa arquitectura de encaminhamento : encaminhamento que provoca a preempção e encaminhamento que não provoca a preempção. No *encaminhamento que provoca a preempção*, os recursos da rede que já foram atribuídos a chamadas existentes, podem ser recuperados e usados para acomodar novas chamadas de grande importância [16]. Assim, a eficácia do bloqueio das chamadas de alta prioridade, é melhorada à custa da interrupção das chamadas de baixa prioridade. No *encaminhamento que não provoca a preempção*, as chamadas de alta prioridade têm acesso preferencial e garantido aos recursos da rede, sem interferir com as

chamadas existentes fora dela. Desta forma, a eficácia do bloqueio das chamadas de alta prioridade é melhorada à custa da eficácia do bloqueio das chamadas de baixa prioridade.

4. Arquitectura de uma chamada

A arquitectura de uma chamada, que pode ser utilizada para suportar a adaptação da QoS, é composta por **processamento** e **encaminhamento da chamada**. Estes processos são duas componentes chave do controlo da rede, localizados em cada nó e representando um sistema de comutação numa rede. Estes processos são utilizados em simultâneo, para fornecer a cada chamada uma ligação que satisfaça todas as restrições de QoS e para manter um nível aceitável de QoS durante a chamada. A Fig. 50 apresenta o fluxo de controlo de sinais entre um par de utilizadores finais durante a instalação de uma chamada.

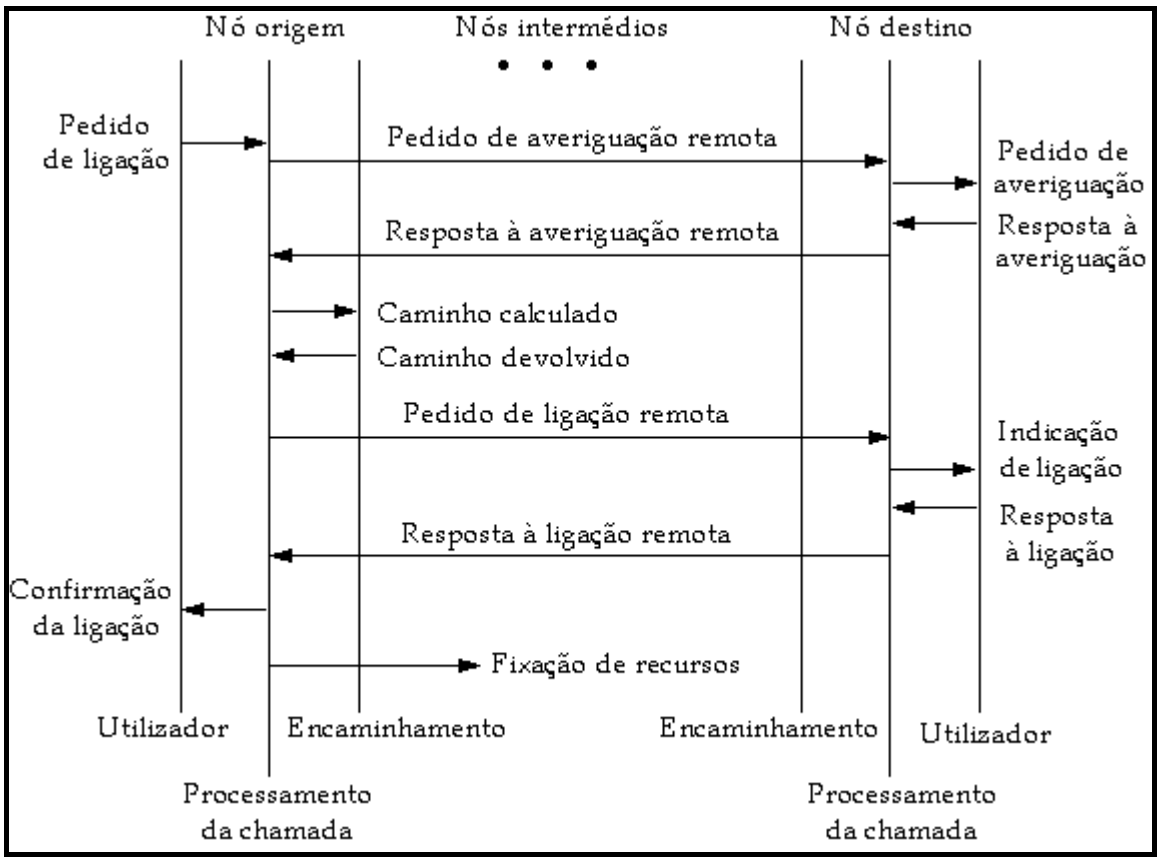


Fig. 50 – Arquitectura de uma chamada [16].

Quando o utilizador na origem faz um pedido de ligação, o *processamento da chamada* da origem transfere o controlo para o *processamento da chamada* do destino, o qual averigua a QoS

ali especificada. Depois de averiguar sucessivamente o utilizador, o *processamento da chamada* do destino devolve a informação obtida ao *processamento da chamada* da origem, para que as restrições de QoS obtidas a partir dos requisitos de QoS dos utilizadores finais sejam consolidadas num conjunto consistente de restrições, as quais são atribuídas à ligação a ser estabelecida. Note-se que um conjunto de restrições de QoS é aceitável para a instalação de uma chamada, se for aceitável para ambos os utilizadores. Consequentemente, o *processamento da chamada* da origem obtém, a partir do encaminhamento, um caminho aceitável que satisfaça as restrições de QoS consolidadas. A QoS correspondente ao caminho seleccionado é referido como a QoS disponível [16].

A negociação de QoS é um processo que envolve a rede e os utilizadores finais, e que ocorre durante a instalação da chamada, para determinar o nível de QoS que foi estipulado pelos utilizadores finais e que a rede pode suportar. Logo que seja concluída a negociação de QoS, um *pedido de ligação remota* é enviado através da rede e indicado ao utilizador no destino. A resposta é transmitida ao *processamento da chamada* do nó origem, que notifica o utilizador na origem da conclusão da negociação de QoS. Por fim, o *processamento da chamada* percorre o caminho, fixando os recursos desde a origem até ao destino.

5. Gestão da ligação

A **gestão da ligação** é um mecanismo de controlo da rede ao nível da ligação, responsável pelo estabelecimento, continuação e libertação das ligações. A Fig. 51 mostra as transições entre quatro estados distintos de uma ligação : *estabelecimento*, *restabelecimento*, *transferência de informação* e *libertação*.

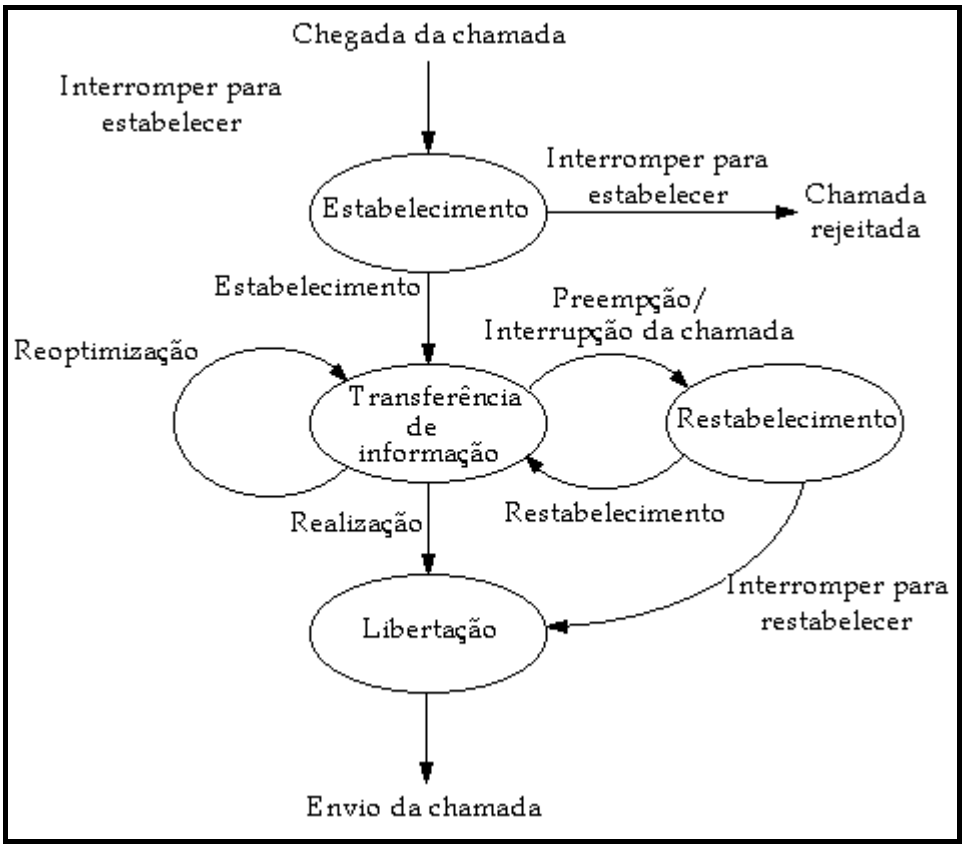


Fig. 51 – Estados de uma ligação [16].

O **estabelecimento** está associado à instalação da ligação.

O **restabelecimento** é necessário após a ligação existente ter sido interrompida, devido a uma falha da rede ou por provocar a preempção. A *reoptimização* é executada pela rede para conservar os recursos utilizados para estabelecer as ligações.

Quando a chamada chega a um nó da rede, regista o estado de *estabelecimento*. Se não existem recursos suficientes para suportar a chamada, esta é rejeitada; caso contrário, ao estabelecer com êxito a ligação (isto é, utilizar um caminho em que são satisfeitas todas as restrições de QoS), a chamada regista o estado de **transferência de informação**.

Quando a chamada no estado de *transferência de informação* é concluída, regista o estado de **libertação** e a ligação é imediatamente libertada. Quando uma chamada está no estado de *transferência de informação*, a sua ligação pode ser interrompida ou ficar sujeita a preempção. Se isto acontecer, a chamada regista o estado de *restabelecimento* até que a rede possa tentar, automaticamente, restabelecer a ligação, determinando um novo caminho aceitável. Durante o restabelecimento, os requisitos de QoS associados à chamada afectada, são ajustados de acordo com as regras de actualização, que têm em consideração o valor previamente estabelecido. Caso o *restabelecimento* tenha êxito, a chamada volta ao estado de *transferência de*

informação. O intervalo de tempo no qual a tentativa de *restabelecimento* de uma ligação pode ser repetido, é limitado pelo atraso do restabelecimento da chamada. Além deste atraso, o processo de *restabelecimento* é abortado e a chamada entra no estado de *libertação*.

A *reoptimização* é concluída pela rede sem o envolvimento directo do utilizador, determinando um caminho mais económico ou satisfazendo as restrições de QoS mais rigorosas. O processo da *reoptimização* é semelhante ao do *restabelecimento*.

Durante a preempção da chamada, a chamada que provoca a preempção deve estar no estado de *estabelecimento* ou de *restabelecimento* da ligação. Durante o cálculo do caminho, a chamada apenas pode provocar a preempção nas chamadas de mais baixa prioridade que estejam no estado de *transferência de informação*. No entanto, para provocar a preempção devido a conflitos nos recursos, quando estes são atribuídos à nova ligação, a chamada sujeita a preempção pode estar em qualquer estado.

6. Encaminhamento sujeito a restrições de QoS

Nesta secção são examinados os problemas de encaminhamento sujeitos a restrições de desempenho, de recursos e de prioridade.

No **encaminhamento sujeito a restrições de desempenho**, uma restrição no caminho é obtida a partir do valor aceitável de cada parâmetro de desempenho. Cada métrica associada a um caminho com restrições, pode ser ela própria um critério de minimização para aumentar a possibilidade de encontrar um caminho que satisfaça essa restrição específica no caminho. No entanto, segundo [16] "um problema de caminho mais curto multicritério não é um problema de optimização bem definido, a não ser que todos os critérios estejam envolvidos numa única função utilidade, utilizada como a única função objectivo do problema".

No **encaminhamento sujeito a restrições de recursos**, é prática corrente atribuir pesos aos arcos e incorporá-los na função objectivo do encaminhamento, de forma a adaptar a selecção preferencial dos recursos da rede. Neste caso, contudo, pode não ser possível satisfazer as preferências utilizador, atendendo aos coeficientes de ponderação que atribui aos recursos, dado que a minimização de uma função objectivo soma ponderada ao longo do caminho, não garante a minimização do uso dos recursos num dado arco. Além disso, o desempenho do encaminhamento é muito sensível aos pesos atribuídos aos arcos [16].

No **encaminhamento sujeito a restrições de prioridade**, os algoritmos de encaminhamento que provoca a preempção, que contam com o transbordo para determinar os

caminhos desejáveis, não são muito eficientes na utilização dos recursos da rede [16]. Os algoritmos do encaminhamento que não provoca a preempção, que reservam os recursos da rede nas chamadas de alta prioridade, não são muito eficientes, uma vez que os recursos reservados não reclamados não podem ser utilizados pelas chamadas de baixa prioridade.

Na abordagem que provoca a preempção, pode ser evitado o transbordo, se toda a informação relevante para o encaminhamento ficar disponível em cada nó, através de um protocolo eficiente de distribuição da topologia. As *restrições de prioridade* podem ser especificadas em relação aos estados das ligações, se forem implementadas como restrições nos arcos. Especificamente, a cada chamada é atribuído um nível de prioridade para os estados de estabelecimento da ligação (*prioridade de estabelecimento*), de restabelecimento da ligação (*prioridade de restabelecimento*) e de transferência de informação (*prioridade de retenção*) [16]. Apenas é permitida a preempção, quando a prioridade da chamada que provoca a preempção é mais elevada do que a da chamada que fica sujeita a preempção. O nível de prioridade adequado utilizado na comparação, é aquele que está associado ao estado de *ligação* de uma dada chamada. Assim, a admissibilidade da chamada num dado arco, depende do nível adequado de prioridade da chamada. Os valores da prioridade são também utilizados para resolver conflitos dos recursos, quando várias chamadas tentam, simultaneamente, utilizar os mesmos recursos da rede. Neste caso, a chamada com a mais elevada prioridade é processada em primeiro.

Ocorre um efeito cíclico indesejado, quando uma chamada com elevada prioridade de restabelecimento, provoca a preempção a uma chamada existente com prioridade de retenção muito baixa, e posteriormente, fica sujeita a preempção por esta última, porque a prioridade de retenção da primeira é menor do que a prioridade de restabelecimento da última. Consequentemente, as duas chamadas podem, alternativamente, comutar entre os estados de restabelecimento da ligação e o de transferência de informação. O efeito cíclico pode ser evitado, se a prioridade de retenção requerida por cada chamada for, pelo menos, tão elevada como a sua prioridade de restabelecimento [16].

7. Estruturas de encaminhamento

Nesta secção, faz-se uma breve referência a algumas estruturas de encaminhamento, comparando-se as suas capacidades em redes integradas de comunicações. Para tal, consideram-se cinco aspectos específicos : economia de escala, economia de alcance, elevada

granularidade, optimalidade do caminho e capacidade adaptativa. As comparações estão sumariadas na seguinte tabela [16] :

Estrutura do encaminhamento	economia de alcance	economia de escala	elevada granularidade	optimalidade do caminho	capacidade adaptativa
chamada a chamada	×		×	×	×
controlado por tabela		×			
na origem	×		×	×	
passo a passo		×			×
com "fallback"	×		×	×	×
alternativo		×			×

O **encaminhamento chamada a chamada** tem flexibilidade para determinar um caminho com as características e os requisitos de QoS de cada chamada [16]. É permitido o controlo da rede com elevada granularidade, já que o cálculo do caminho é feito à chegada de cada chamada. Como o cálculo total é independente do número de tipos de chamada, então existe economia de alcance. Desde que as redes integradas actuais estejam bem definidas (de forma a suportar as relações requeridas de QoS), o encaminhamento chamada a chamada apresenta grande potencial para o encaminhamento sensível à QoS.

No **encaminhamento controlado por tabela predefinida**, todos os caminhos são predeterminados e guardados em tabelas de encaminhamento [16]. Existe economia de escala, porque muitas chamadas são encaminhadas pelo mesmo caminho. Os caminhos escolhidos não podem ser óptimos para todos os tipos de tráfego, a não ser que tabelas diferentes sejam utilizadas em cada tipo de tráfego. As tabelas de encaminhamento têm que se adaptar às alterações topológicas.

No **encaminhamento na origem**, as decisões para encaminhar uma chamada são totalmente produzidas na origem, baseadas na configuração geral e no estado da rede, as quais são actualizadas através de um protocolo de distribuição da topologia, o que faz com que as funções de encaminhamento nos nós intermédios sejam relativamente simples [16]. Com suficiente informação da topologia disponível na origem, é possível ter um controlo da rede com elevada granularidade e optimizar os caminhos desejados. Existe economia de alcance, já que a mesma informação da topologia pode ser utilizada por muitos tipos de tráfego. Por isso, é que nas redes integradas modernas o encaminhamento na origem é muito

prometedor para o caminho escolhido onde existe um aumento crescente de muitos tipos de tráfego, devido às aplicações "multimedia".

No **encaminhamento passo a passo**, as decisões de encaminhamento são distribuídas e o cálculo do encaminhamento nos nós intermédios não é trivial [16]. Com a utilização de tabelas de encaminhamento, existe economia de escala, apesar destas tabelas exigirem armazenamento. Neste tipo de encaminhamento, são necessários passos para prevenir ciclos. Embora o processo possa responder rapidamente a interrupções, a recuperação é sub-óptima, porque retém o subcaminho da origem ao ponto de interrupção. É difícil utilizar o encaminhamento passo a passo para suportar a adaptação da QoS ao nível da chamada (por exemplo, chamada que provoca a preempção), porque não existe informação suficiente, relativa às chamadas específicas, de onde as decisões de encaminhamento são produzidas.

O **encaminhamento com "fallback"** (isto é, com possibilidade de voltar ao início do processo, calculando outro caminho) determina, sequencialmente, os caminhos baseados numa sequência de instâncias de encaminhamento, até uma delas estar disponível, ou a chamada ficar bloqueada ao realizar uma sequência predeterminada de "fallbacks" (Fig. 52). O encaminhamento com "fallback" adapta-se às alterações do estado da rede, permitindo a determinação de caminhos alternativos, para adaptar as restrições de recursos preferidas, às chamadas que provocam a preempção, e de outras marcas de encaminhamento que exijam o cálculo de vários caminhos por cada chamada instalada. Como o encaminhamento com "fallback" oferece grande economia de alcance, adapta utilizadores heterogéneos através do cálculo de caminhos alternativos quando necessário. A sequência de instâncias de encaminhamento com "fallback" é predeterminada ou seleccionada em tempo real, de acordo com as regras estabelecidas. As instâncias de encaminhamento podem ser especificadas para suportar a elevada granularidade, desde que a informação pertinente de QoS esteja disponível.

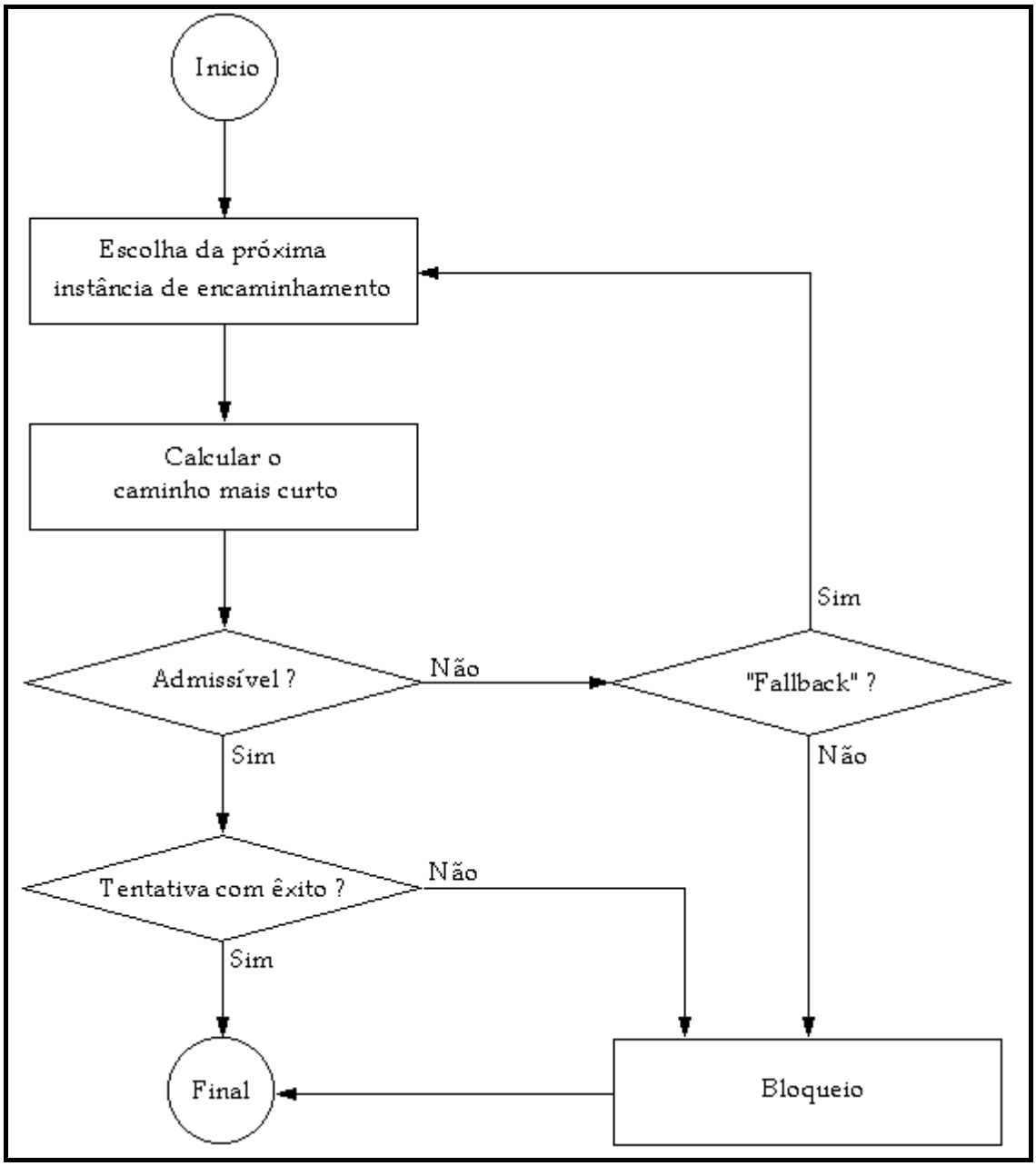


Fig. 52 – Encaminhamento com "fallback" [16].

No **encaminhamento alternativo**, um conjunto de caminhos predeterminados e armazenados em tabelas de encaminhamento é testado sequencialmente, durante a instalação de cada chamada para fixar os recursos, até uma tentativa ter sucesso ou a chamada ficar bloqueada na realização de uma sequência de tentativas [16]. Os caminhos alternativos podem depender das classes de tráfegos e da hora do dia. A utilização de tabelas de encaminhamento apresenta economia de escala, mas é necessário uma significativa armazenagem. As tabelas de encaminhamento podem ser actualizadas periodicamente, de modo a se adaptarem às condições existentes da rede. O modo como os caminhos são testados

depende das variantes de encaminhamento alternativo existentes. Enquanto sob cargas leves o encaminhamento alternativo minimiza a probabilidade de bloqueio, sob cargas intensas a probabilidade de bloqueio pode aumentar drasticamente, quando os caminhos alternativos utilizados tendem a consumir mais recursos da rede.

8. Encaminhamento chamada a chamada na origem baseado em regras

A estratégia proposta em [16] de encaminhamento chamada a chamada na origem com "fallbacks" baseado em regras, em redes de comunicações com tráfegos integrados sujeitos a diversos requisitos de QoS, consiste em determinar, eficientemente, um caminho aceitável para cada chamada, de acordo com o estado corrente da rede. Em oposição ao paradigma do encaminhamento tradicional, onde o passo inicial consiste em minimizar o valor de uma função objectivo, o paradigma de encaminhamento que se descreve a seguir realça a conjugação de várias restrições. Apesar disso, o algoritmo de caminho mais curto é utilizado habitualmente como uma forma de identificar caminhos aceitáveis.

8.1. Encaminhamento com "fallback" baseado em regras

A estratégia de encaminhamento baseado em regras utiliza toda a informação disponível para modificar, dinamicamente, a sequência de caminhos com "fallback", determinados de acordo com os estados da rede e da ligação. A estratégia começa por calcular o caminho com uma instância de encaminhamento inicial, o qual é determinado através do estado da ligação e dos requisitos de QoS utilizados. Se não for determinado qualquer caminho admissível, então são utilizadas regras de paragem para decidir se o "fallback" calculado está em ordem. No "fallback" escolhe-se uma nova instância de encaminhamento, com as restrições relaxadas de acordo com as regras de "fallback". Se for encontrado um caminho admissível, então a rede tentará fixar os seus recursos para a chamada, ao longo do caminho. Por vezes esta tentativa falha, porque há contenção dos recursos devido à latência na actualização da topologia, o que provoca a interrupção da chamada.

A alternativa mais sofisticada é permitir "**crankback**" (isto é, seleccionar mediante certas condições), até que a origem escolha uma nova instância de encaminhamento por "fallback", utilizando a última informação obtida a partir do teste sem sucesso. O encaminhamento com "**crankback**" é semelhante ao encaminhamento alternativo, excepto

que os caminhos alternativos não são predeterminados, mas calculados um de cada vez, após cada tentativa sem êxito para estabelecer uma ligação.

A estratégia proposta para o encaminhamento baseado em regras segue um modelo genérico que consiste em três módulos : **base de dados**, **base de conhecimento** e **mecanismo de inferência** [16]. A *base de dados* contém a informação da topologia, que é actualizada através de um protocolo de distribuição da topologia. A *base de conhecimento* contém regras que são utilizadas para gerar instâncias de encaminhamento, baseadas numa política de encaminhamento predeterminada e da QoS pedida pelos utilizadores. O *mecanismo de inferência* é basicamente um cálculo sequencial de caminhos mais curtos, a qual verifica também a admissibilidade do caminho.

"Fallback" com restrições de desempenho — Em vez de calcular um caminho óptimo para o problema de caminho mais curto com restrições, a estratégia de encaminhamento baseado em regras utiliza a seguinte heurística para as "fallbacks" [16] :

> Inicialmente, resolve-se o problema original de encaminhamento sem restrições nos caminhos, verificando-se, a seguir, a admissibilidade através da verificação do caminho escolhido e comparando as restrições nos caminhos. Se estas não forem satisfeitas para o caminho escolhido, a "fallback" permite que a chamada tenha uma ou mais oportunidades adicionais para procurar um outro caminho admissível.

Em vez de utilizar uma função utilidade, a estratégia do encaminhamento baseada em regras pressupõe que os critérios de encaminhamento são ordenados pelos utilizadores da rede, de acordo com as importâncias relativas. Estes são então utilizados para determinar as funções objectivo de caminho mais curto para o cálculo dos caminhos inicial e com "fallback".

"Fallback" com restrições de recursos — A estratégia de encaminhamento baseado em regras pode introduzir preferências nas restrições de recursos. As preferências utilizadas nos recursos podem-se traduzir por privilégios a atribuir a conjuntos de restrições de recursos. Cada um destes conjuntos pode ser escolhido para definir uma instância de encaminhamento, de acordo com algumas regras predeterminadas. No encaminhamento com "fallback", primeiro são calculados os caminhos que apenas utilizam os recursos que satisfazem as restrições (de recursos) requeridas, com um "fallback" para as restrições de recursos aceitáveis menos restritivas [16].

"Fallback" com restrições de prioridade — Nas redes com classes de prioridade, o encaminhamento que provoca a preempção pode ser interrompido, se o débito total não for aumentado, devido às chamadas que provocam a preempção. Apesar disso, e uma vez que

nenhum recurso está reservado para as chamadas de alta prioridade, não é provável que as chamadas de baixa prioridade sejam bloqueadas quando a rede está ligeiramente sobrecarregada. Muitas vezes, é possível relaxar as restrições de recurso de uma chamada, até que as suas restrições de desempenho possam ser encontradas, sem interrupções desnecessárias nas chamadas de mais baixa prioridade. Uma vez que a preempção está associada a interrupções, o encaminhamento que não provoca a preempção deverá ser testado antes da preempção ser considerada. Esta abordagem é referida como preempção de "look–around–first". A abordagem com "fallback" para o encaminhamento, permite a preempção de "look–around–first" para evitar a preempção desnecessária, na primeira tentativa para determinar um caminho admissível sem provocar preempção em qualquer chamada existente [16]. Se isto falhar, então na procura de um caminho admissível, a instância de encaminhamento com "fallback" apenas dá conta dos recursos consumidos pela mesma chamada ou pelas chamadas de mais alta prioridade.

8.2. Regras de "fallback" dependente do estado

Para facilitar a adaptação da QoS ao nível da chamada, as estratégias existentes de encaminhamento alternativo utilizam regras simples para seleccionar os caminhos alternativos predeterminados. Alguns protocolos de encaminhamento entre domínios, tal como OSPF, confiam nas instâncias predeterminadas de encaminhamento que dependem do tipo de serviço. A estratégia proposta de encaminhamento baseado em regras utiliza uma base de conhecimento, que contém um conjunto de regras para seleccionar, em tempo real, as instâncias de encaminhamento, quando necessárias para os cálculos dos caminhos inicial e com "fallback". Nas suas principais características incluem-se regras para seleccionar restrições nas instâncias de encaminhamento, regras de "fallback" de uma instância para outra e regras para determinar quando fazer "fallback" e quando parar.

9. Análise como problema multiobjectivo

Como se pode concluir das secções anteriores, é vantajoso considerar o problema de encaminhamento sujeito a restrições de qualidade de serviço como um problema multiobjectivo. Para tal, restrições nos caminhos, tais como custo, atraso (ou probabilidade de bloqueio) e largura de banda, são introduzidas explicitamente nos modelos matemáticos como funções objectivo de encaminhamento a optimizar.

Desta forma, os modelos de encaminhamento com múltiplos objectivos permitem compreender os compromissos entre os vários requisitos de QoS, os quais podem, eventualmente, estar dependentes das aplicações, racionalizando a comparação entre os diferentes encaminhamentos alternativos, de modo a confrontar os critérios conflituosos envolvidos no cálculo para seleccionar os caminhos aceitáveis. Assim, os modelos tornam-se mais realistas, assumindo a natureza multiobjectivo do problema, permitindo compreender os conflitos inerentes e os compromissos entre os vários objectivos, ao seleccionar o melhor plano de compromisso a partir do conjunto de soluções não dominadas [4].

10. Uma abordagem multiobjectivo ao problema do encaminhamento

Em redes integradas de comunicações, o problema de encaminhamento é modelado como um problema de caminho mais curto com múltiplos objectivos, no qual os vários aspectos do cálculo são explicitamente incorporados na formulação do problema, dando ênfase à procura de caminhos de compromisso satisfatórios, relacionados com os vários requisitos de QoS.

Para cada métrica, os requisitos de QoS são expressos como restrições adicionais "leves" (isto é, não incorporadas explicitamente na formulação matemática do problema) nos valores da função objectivo, em termos de limites requerido e aceitável, os quais definem regiões de preferência no espaço das funções objectivo. Estas regiões são então analisadas para determinar os caminhos (se existirem) que satisfaçam aqueles requisitos.

10.1. As métricas para os requisitos de QoS

Geralmente, as métricas de encaminhamento consideradas são o atraso, o custo, o número de arcos, a probabilidade de perda, a taxa de erro e a largura de banda. A função de agregação para calcular o valor de um caminho depende da métrica utilizada. Como as métricas são representações de uma rede no encaminhamento, então as maiores dificuldades encontram-se não só na complexidade do cálculo do caminho, mas também na cadeia de requisitos de QoS que se pode suportar. As características desejáveis de cada métrica são as seguintes [26] :

i) Para qualquer métrica escolhida, devem existir algoritmos eficientes para calcular os caminhos, para que o protocolo de encaminhamento seja capaz de abranger as grandes redes. A complexidade destes algoritmos deve ser, de preferência, comparável com a dos

algoritmos de encaminhamento actuais. Qualquer algoritmo deve também ser capaz de funcionar, tanto em ambientes centralizados, como em ambientes distribuídos.

ii) As métricas devem reflectir as características básicas da rede. A informação que elas contêm deve permitir que suportem os requisitos básicos de QoS. Note-se que qualquer requisito de QoS tem que ser planeado de acordo com restrições num caminho expresso em termos de métricas; portanto, as métricas determinam, até certo ponto, os tipos de QoS que a rede pode suportar.

iii) As métricas devem ser ortogonais entre si, de modo a não existir informação redundante entre elas. A informação redundante pode introduzir interdependência entre as métricas, tornando impossível o cálculo de cada métrica, separadamente. A avaliação recursiva entre métricas pode complicar substancialmente o cálculo dos caminhos.

Os algoritmos com apenas uma métrica para determinar caminhos, tais como atraso e número de arcos, são bem conhecidos e têm sido muito utilizados nas redes actuais. Desta forma, será que uma única métrica poderá suportar os requisitos de QoS utilizados ?

Uma possível abordagem consiste em definir uma função e gerar uma única métrica com vários parâmetros. A ideia é fixar vários fragmentos de informação numa única medida e utilizá-la como base para as decisões de encaminhamento.

No entanto, uma única métrica fixa pode apenas ser utilizada, na melhor das hipóteses, como um indicador, visto que ela não contém informação suficiente para estabelecer se os requisitos de QoS utilizados podem ser encontrados ou não. Outro problema é utilizar parâmetros fixos de diferentes regras de composição de métricas individuais.

Apesar de várias métricas poderem modelar uma rede com mais precisão, a dificuldade está em arranjar algoritmos eficazes para resolver problemas de caminho mais curto com várias restrições adicionais, uma vez que a introdução destas restrições destrói algumas propriedades fundamentais do modelo.

O problema no encaminhamento com QoS é muito mais complicado, uma vez que os requisitos dos recursos especificados são muito diversos e dependentes da aplicação. Inicialmente, a complexidade do cálculo é determinado pelas regras de composição das métricas. Basicamente, existem três tipos de métricas : aditiva, multiplicativa e côncava [26]. Seja $\mathbf{c_{ij}}$ a métrica para o arco (i, j) e **p** um caminho numa rede; então,

a) a métrica é aditiva se

$$V_p = \sum_{(i,j) \in p} c_{ij}$$

b) a métrica é multiplicativa se

$$V_p = \prod_{(i,j) \in p} c_{ij}$$

c) a métrica é côncava se

$$V_p = \min_{(i,j) \in p} \{ c_{ij} \}$$

Enquanto o atraso, o número de arcos e o custo seguem a função de agregação aditiva, a largura de banda segue a regra de agregação côncava.

As métricas de probabilidade de perda (e de taxa de erro) seguem a função de agregação

$$V_p = 1 - \prod_{(i,j) \in p} (1 - c_{ij})$$

No entanto, esta métrica pode ser transformada, primeiro numa métrica multiplicativa, e depois numa aditiva (podendo assim ser usada numa abordagem de caminho mais curto), da seguinte forma :

$$V_p = 1 - \prod_{(i,\, j) \in p} (1 - c_{ij})$$

$$1 - V_p = \prod_{(i,\, j) \in p} (1 - c_{ij})$$

Esta expressão dá a probabilidade de não bloqueio, que segue a função de agregação multiplicativa.

No entanto, aplicando logaritmos, obtém-se :

$$\log (1 - V_p) = \log \prod_{(i,\, j) \in p} (1 - c_{ij})$$

$$\log (1 - V_p) = \sum_{(i,\, j) \in p} \log\ (1 - c_{ij})$$

A maximização da probabilidade da transmissão ter êxito é então equivalente a

$$\max\ \log (1 - V_p) = \max \sum_{(i,\, j) \in p} \log\ (1 - c_{ij})$$

que é equivalente a

$$\min \sum_{(i,\, j) \in p} - \log\ (1 - c_{ij})$$

Uma vez que c_{ij} é uma probabilidade, o coeficiente associado ao arco (i, j), que é positivo, é

$$- \log (1 - c_{ij})$$

Qualquer métrica aditiva pode ser usada numa abordagem de caminho mais curto. No entanto, a abordagem proposta para o caso multiobjectivo pode ser utilizada sempre que

forem consideradas métricas aditivas, multiplicativas e probabilidade de perda, em que as duas últimas são transformadas no caso aditivo.

10.2. O algoritmo proposto

A metodologia geral para resolver problemas de encaminhamento quando modelados como problemas de caminho mais curto com **h** objectivos ($h \geq 2$), é a seguinte :

1. Determinar as soluções não dominadas que optimizam cada função objectivo individualmente, resolvendo **h** problemas de caminho mais curto. Isto produz informação relativa ao conjunto de valores de cada função objectivo, no conjunto de soluções não dominadas.
2. Indicar os requisitos de QoS para cada uma das **h** métricas (associadas às **h** funções objectivo), os quais são especificados através dos seguintes limites :

   valor requerido (nível de aspiração) : M_{req}

   valor aceitável (nível de reserva) : M_{ac} ($M_{req} < M_{ac}$).

O facto de acrescentar estes tipos de restrições "leves" define regiões de prioridade, nas quais as soluções não dominadas são pesquisadas de acordo com os requisitos de QoS. De forma a facilitar a ilustração gráfica deste método, a Fig. 53 mostra a forma de utilizar os requisitos de QoS para definir as regiões de prioridade de um problema com duas funções objectivo.

Esta abordagem tanto pode ser implementada de uma maneira interactiva (pedindo ao AD informação para conduzir a procura de soluções não dominadas), como de uma maneira automática (seguindo regras de procura predefinidas). O algoritmo consiste em optimizar funções objectivo de somas pesadas, para calcular as soluções não dominadas que pertencem ao contorno convexo, e em utilizar um algoritmo eficiente dos k caminhos mais curtos, para procurar as soluções não dominadas pertencentes às zonas do interior do respectivo invólucro convexo.

Neste algoritmo, os pesos associados à função escalar soma ponderada são calculados a partir das componentes, normalizadas, do gradiente do hiperplano que passa pelos pontos formados com os valores requeridos e com os que optimizam as funções objectivo separadamente. Com estes pesos, a região de maior interesse (a de primeira prioridade) é pesquisada de uma forma mais uniforme, uma vez que aquele gradiente traduz a relação existente entre os valores requeridos.

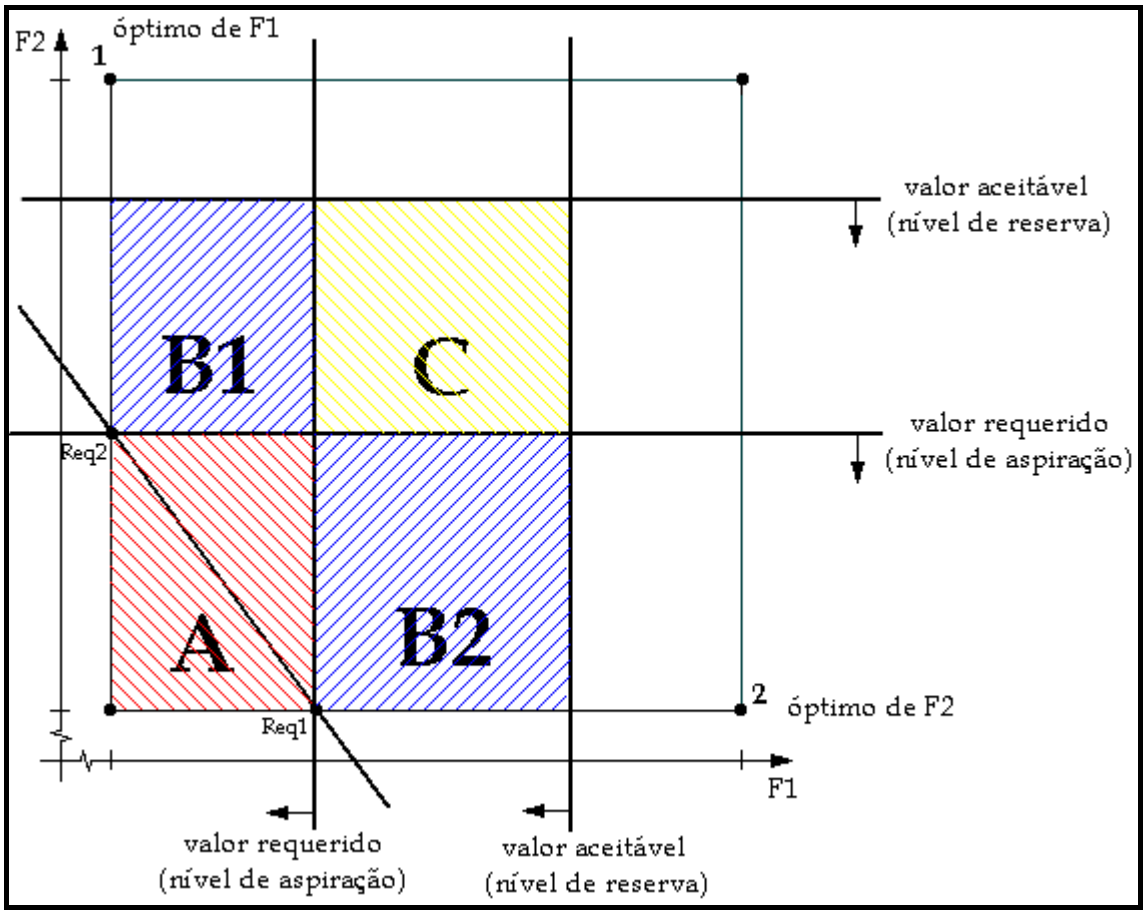


Fig. 53 – Os requisitos de QoS utilizados para definir as zonas de prioridade.

O passo principal deste algoritmo, cuja descrição pormenorizada se encontra em Algoritmo 12, consiste no seguinte :

1. determinar uma solução (não dominada) utilizando o algoritmo dos k caminhos mais curtos (MPS) aplicado à função escalar soma ponderada construída da forma descrita,
2. analisar a que região de prioridade pertence a solução,
3. se esta solução é melhor do que a melhor encontrada até ao momento (se o nível de prioridade desta nova solução for menor do que o da melhor solução encontrada até ao momento), então actualizar a melhor solução (que passa a ser esta última),
4. se todas as **r** regiões de maior prioridade ficaram totalmente analisadas, verificar se a melhor solução, encontrada até ao momento, pertence a uma dessas regiões ou à de prioridade **r+1** : se pertence, então o processo termina (sendo esta a solução escolhida), caso contrário continua a pesquisa da solução desejada.

Na descrição de Algoritmo 12, considere-se a seguinte notação :

h ← número de funções objectivo (métricas) do problema;

NZonas ← número de regiões de prioridade;

Req(i), Ac(i) ← valores requerido e aceitável para o objectivo **i** (i = 1, ..., h);

Solução(i) ← solução que optimiza a função objectivo **i** (i = 1, ..., h)
Opt(i) ← valor da função objectivo **i** da Solução(i) (i = 1, ..., h)
Se não existe qualquer solução **Ou** existe apenas uma **Então**
    STOP
Indicar os requisitos de QoS associados a cada função objectivo : Req(i) e Ac(i)
Zona(j) ← região de prioridade **j** (j = 1, ..., NZonas) — utiliza Req(i) e Ac(i)
MelhorSolução ← ∅ (a melhor solução encontrada até ao momento)
MelhorNível ← ∞ (nível de prioridade da região a que pertence a MelhorSolução)
FObjPesada ← função escalar soma ponderada, construída à custa de Req e de Opt
Solução ← caminho mais curto relativo à função objectivo FObjPesada
REPETIR
    Prioridade ← nível de prioridade da região a que pertence a Solução
    Se Prioridade = 1 **Então**
        MelhorSolução ← Solução
        STOP *{ encontrou a solução desejada }*
    Se Prioridade **<** MelhorNível **Então**
        MelhorSolução ← Solução
        MelhorNível ← Prioridade
    Para j **desde** 1 **até** NZonas **Fazer**
        Se Zona(j) foi pesquisada na totalidade **e** MelhorNível = j + 1 **Então**
            STOP *{ encontrou a solução desejada }*
    Solução ← próximo caminho mais curto relativo à função objectivo FObjPesada
ATÉ encontrar a solução desejada

Algoritmo 12. Problema de encaminhamento : determinação da melhor solução.

10.3. A largura de banda como métrica

A largura de banda (ou débito) de um caminho é definida como a largura de banda residual mínima entre todos os arcos do caminho, ou a largura de banda segundo a regra do "funil" [26].

Se for considerada a largura de banda requerida, então, e uma vez que é uma métrica côncava (isto é, $b_{st} = \min \{ b_{ij} : (i, j) \in p \}$ em que p é um caminho), os requisitos de QoS para esta métrica, em termos dos mesmos limites

valor requerido (nível de aspiração) : b_{req}

valor aceitável (nível de reserva) : b_{ac} ($b_{ac} < b_{req}$)

pode ser tratada do seguinte modo :

- constrói-se uma nova rede, removendo os arcos (i, j) para os quais $b_{ij} < b_{req}$;
- se a rede resultante é conexa, então existem caminhos entre os nós origem e destino que satisfazem a largura de banda requerida (nível de aspiração);
- se a rede se torna não conexa, então removem-se apenas os arcos (i, j) da rede original, tais que $b_{ij} < b_{ac}$;
- se esta nova rede é conexa, então existe um caminho entre os nós origem e destino que satisfaz a largura de banda aceitável (nível de reserva);
- se a rede não é conexa, então não existe caminho para a largura de banda especificada pelos parâmetros de QoS.

10.4. Problema com duas funções objectivo

Para o problema com duas funções objectivo, podem definir-se quatro regiões associadas a três níveis de prioridade : A (nível 1), B1, B2 (nível 2) e C (nível 3) — Fig. 54. A cada função objectivo (métrica) estão associados dois requisitos de QoS : valor requerido e valor aceitável.

Na região A, de primeira prioridade, são satisfeitos os valores requeridos das duas métricas. Nas regiões B1 e B2, de segunda prioridade, é satisfeito o valor requerido de apenas uma das métricas e o valor aceitável da outra. Na região C, de terceira prioridade, o valor requerido das duas métricas não são satisfeitos, apenas são os seus valores aceitáveis.

Pode-se considerar uma outra região, formada por todos os restantes locais onde possam existir soluções não dominadas do problema com a menor prioridade (*última chance*). Nesta região, o valor aceitável (e o requerido) de pelo menos uma métrica não é satisfeito.

Por outro lado, se for indicada alguma preferência de um objectivo sobre o outro, isto pode ser traduzido fazendo uma distinção entre regiões com a mesma prioridade — neste caso apenas entre B1 e B2. Desta forma, se F1 ➢ F2 (F1 tem preferência em relação a F2) então B1 ➤ B2 (B1 é prioritária em relação a B2); se F2 ➢ F1 então B2 ➤ B1 (Fig. 54).

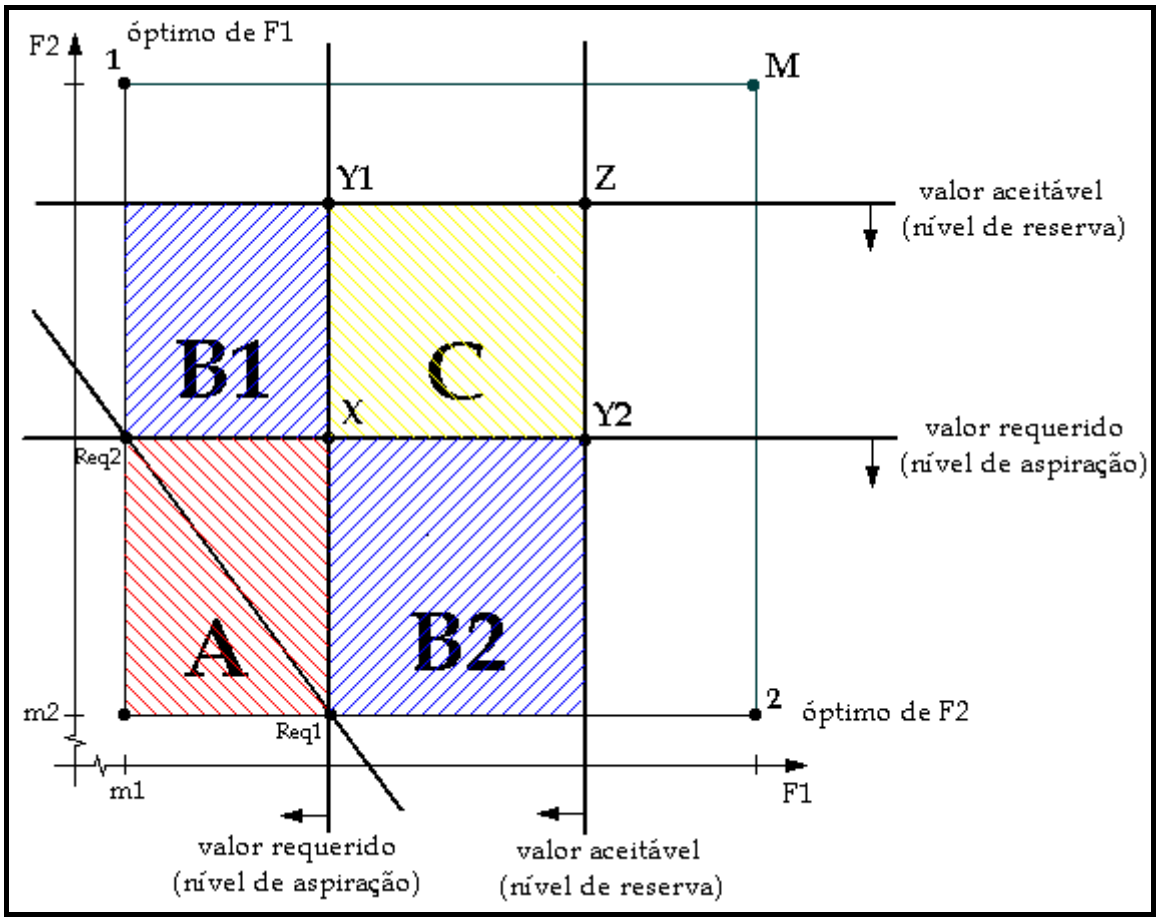


Fig. 54 – Encaminhamento (bi-objectivo) : construção das zonas de prioridade.

Note-se que qualquer solução de A não é dominada por soluções de B e C, e domina qualquer solução de C. Por outro lado, não existe solução de C que domine soluções de B. Da mesma maneira, se conclui que qualquer solução de *última chance* não domina as soluções das regiões A, B e C. Desta forma, fica justificada a não dominância da solução final.

Relativamente aos pesos envolvidos na construção da função escalar soma ponderada, estes são as componentes normalizadas do gradiente do hiperplano que passa pelos pontos Req1 e Req2, os quais são definidos através da conjugação dos valores requeridos e dos óptimos de F1 e F2, respectivamente m1 e m2, da seguinte forma :

$Req1 = (M_{req}(F1), m2)$,

$Req2 = (m1, M_{req}(F2))$

como se pode ver na Fig. 54. Com estes pesos, a região A (a de primeira prioridade) é pesquisada de uma forma mais uniforme, uma vez que aquele gradiente traduz a relação existente entre os valores $M_{req}(F1)$ e $M_{req}(F2)$.

Na descrição apresentada em Algoritmo 12, considera-se que B1 tem preferência em relação a B2, o que não acarreta perda de generalidade. Desta forma, no desenvolvimento deste algoritmo, a primeira solução que se encontrar dentro da região A é imediatamente seleccionada. No entanto, se a região A foi totalmente analisada sem determinar qualquer solução não dominada (o que acontece quando a recta de nível correspondente à função escalar soma ponderada passar o ponto X — Fig. 54), então verifica-se se já foi encontrada alguma solução na região B1. Da mesma maneira, se a região B1 foi totalmente analisada (recta de nível passou o ponto Y1), então verifica-se se já foi encontrada alguma solução em B2. O mesmo se passa em relação às regiões B2 (ponto Y2) e C (ponto Z).

No entanto, só se pode verificar se B2 foi totalmente analisada, quando se tiver a certeza que B1 já o foi, porque se isto não for garantido, pode acontecer que B2 seja totalmente analisada antes de B1 (o que acontece quando a recta de nível passar por Y2 sem ainda ter passado por Y1). Consequentemente, pode-se escolher, como solução final, uma que pertença a C, apesar de B1 ainda não ter sido totalmente analisada. Relativamente às regiões A e C, não existe qualquer problema, uma vez que a região A é a primeira a ser analisada na sua totalidade e a região C é a última (o ponto X é o primeiro a ser passado e o Z o último, qualquer que sejam os valores requeridos e aceitáveis indicados).

Desta forma, o processo termina apenas quando uma das seguintes situações ocorrer (analisar a Fig. 54) :

(1) foi encontrada uma solução pertencente à região A,

(2) foi analisada totalmente a região A e já foi encontrada uma solução em B1,

(3) foi analisada totalmente a região B1 e já foi encontrada uma solução em B2,

(4) foram analisadas totalmente as regiões B1 e B2 e já foi encontrada uma solução em C,

(5) foi analisada totalmente a região C, logo a solução escolhida é uma de *última chance*.

Por outro lado, é garantido que o processo termina, já que pelo menos uma solução é determinada : a solução 1 ou a 2 (Fig. 54). Se o problema não tiver qualquer solução, ou apenas uma, o algoritmo ao verificar esse facto termina imediatamente (Algoritmo 12).

10.5. Problema com três funções objectivo

Para o problema com três funções objectivo, podem ser definidas oito regiões correspondentes a quatro níveis de prioridade : A (nível 1), B1, B2 e B3 (nível 2), C1, C2 e C3 (nível 3) e D (nível 4), como mostram as figuras 55, 56 e 57.

A cada uma das três funções objectivo (métricas) estão associados dois requisitos de QoS : valor requerido e valor aceitável.

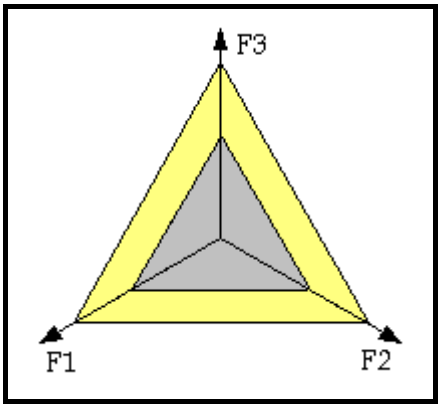


Fig. 55 – Encaminhamento (tri-objectivo) : zonas de primeira e quarta prioridade.

Na Fig. 55 encontram-se representadas a região A, de primeira prioridade (triângulo interior) e a região D, de quarta prioridade (coroa triangular exterior). Na região A são satisfeitos o valor requerido de todas as funções objectivo. Na região D, não é satisfeito o valor requerido de qualquer função objectivo, apenas é o valor aceitável de todas elas.

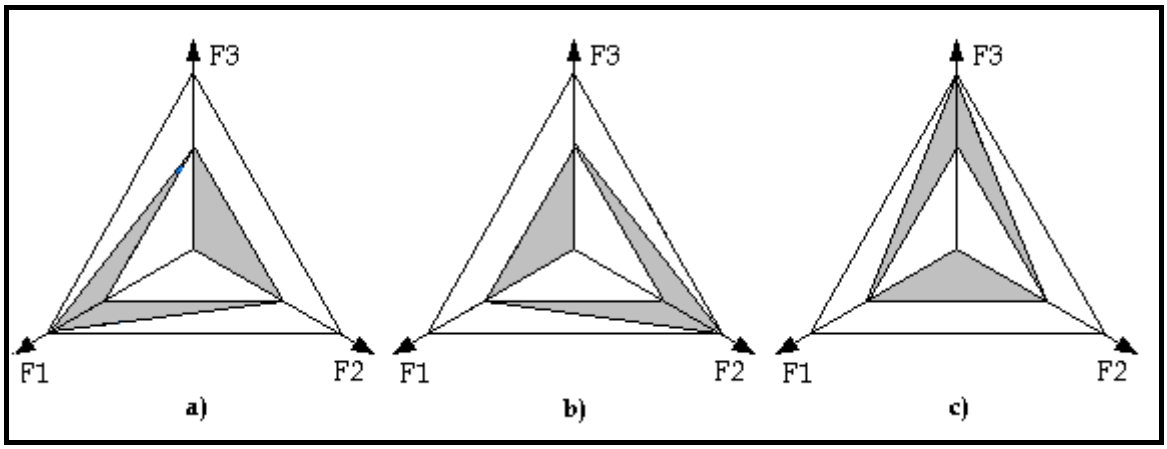


Fig. 56 – Encaminhamento (tri-objectivo) : zonas de segunda prioridade.

Nas regiões de segunda prioridade, B1, B2 e B3, são satisfeitos os valores requeridos de duas funções objectivo e apenas o aceitável da outra; ou seja, apenas o valor requerido de uma função objectivo não é satisfeito. Portanto, cada uma das três regiões formadas está relacionada com a função objectivo cujo valor requerido é satisfeito : B1 (Fig. 56.a)), B2 (Fig. 56.b)) e B3 (Fig. 56.c)), relacionado com a primeira, segunda e terceira função objectivo, respectivamente.

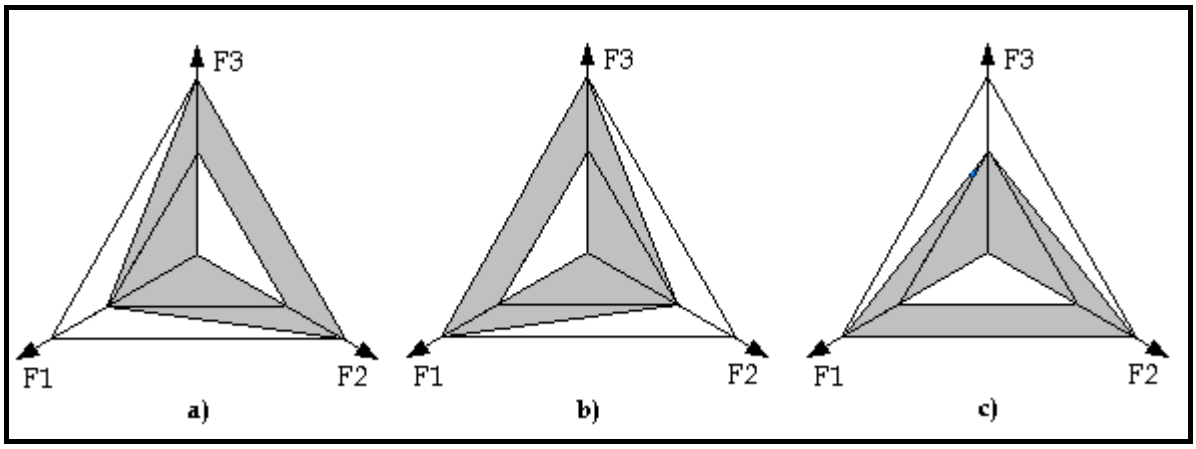


Fig. 57 – Encaminhamento (tri-objectivo) : zonas de terceira prioridade.

Nas regiões de terceira prioridade, C1, C2 e C3, são satisfeitos apenas o valor requerido de uma função objectivo e os aceitáveis das outras duas; ou seja, os valores requeridos de duas funções objectivo não são satisfeitos. Portanto, cada uma das três regiões formadas está relacionada com a função objectivo cujo valor requerido não é satisfeito : C1 (Fig. 57.a)), C2 (Fig. 57.b)) e C3 (Fig. 57.c)), relacionadas com a primeira, segunda e terceira função objectivo, respectivamente.

Pode-se considerar uma outra região, com a menor prioridade, associada às soluções não dominadas que não pertençam a qualquer das regiões descritas (*última chance*). Esta região está associada às soluções que não satisfazem os valores aceitáveis de pelo menos uma função objectivo. Isto é, o valor de pelo menos uma função objectivo é superior ao valor aceitável correspondente.

Também neste caso, a indicação de preferências entre os objectivos envolvidos, implica fazer-se distinção entre as regiões com a mesma prioridade. Por exemplo, se F1 ➢ F2 ➢ F3 então B1 ➤ B2 ➤ B3 e C1 ➤ C2 ➤ C3.

10.6. Exemplo ilustrativo

Para ilustrar o algoritmo proposto, atenda-se à análise de seis casos, que traduzem todas as situações genéricas possíveis de acontecer num problema com dois objectivos, de acordo com a Fig. 58.

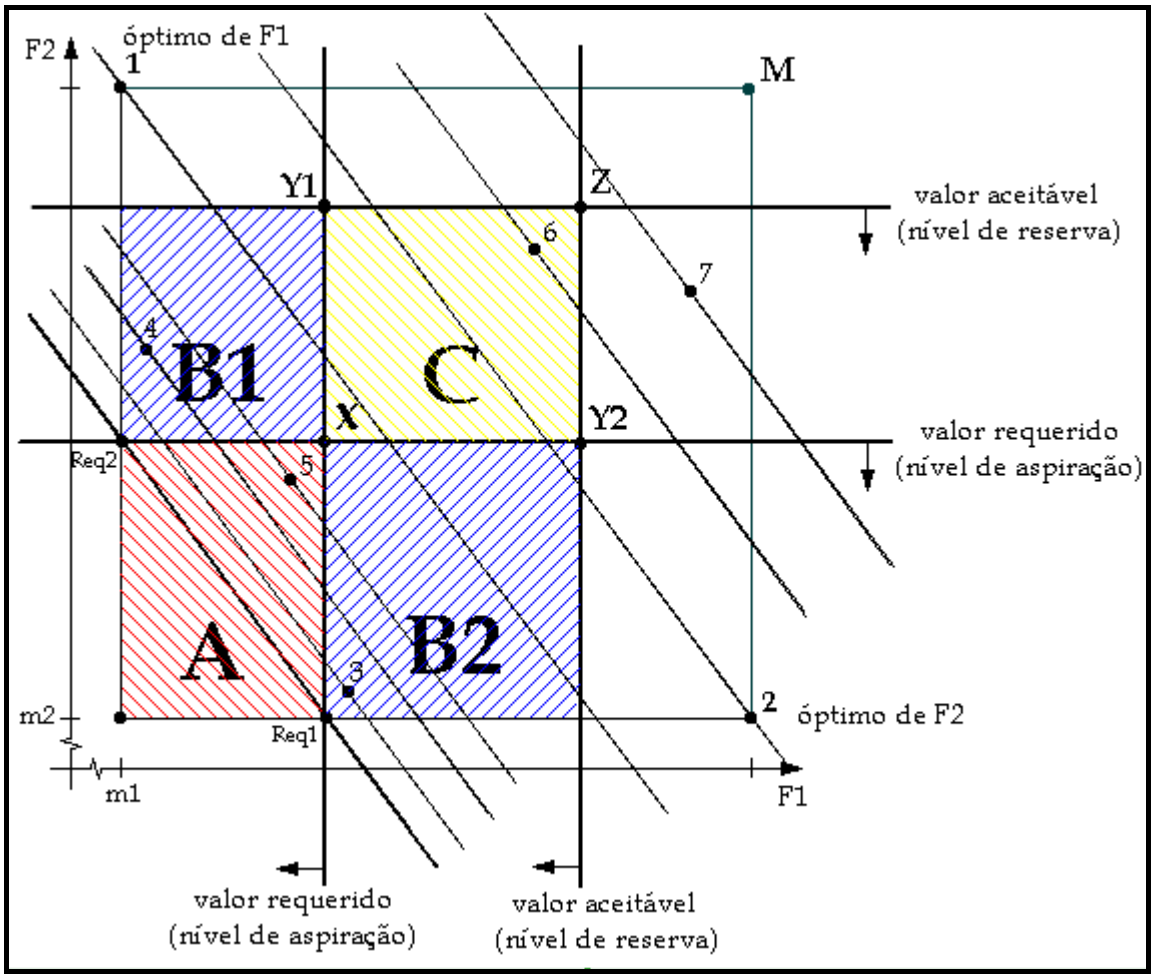


Fig. 58 – Determinação de soluções não dominadas nas várias zonas de prioridade.

Em primeiro lugar, são determinadas as soluções 1 e 2, que optimizam separadamente cada função objectivo. Depois, a partir dos requisitos associados a cada função objectivo indicados são construídas as regiões de prioridade. Finalmente, a primeira fase termina com a construção da função escalar soma ponderada, a partir dos requisitos de QoS e dos valores associados às soluções já determinadas, cujas rectas de nível são mostradas na Fig. 58.

1º Caso.

Usando aquela função escalar suponhamos que são alcançadas sucessivamente as soluções 3 e 4 (ambas fora da zona de primeira prioridade). A pesquisa prossegue e é obtida a solução 5. Desta forma, o processo termina com a escolha da solução 5, já que esta pertence à região de primeira prioridade e as soluções já encontradas (3 e 4) pertencem a uma zona de menor prioridade (B2 e B1, respectivamente).

2º Caso. Suponha-se que a solução 5 não existe.

Usando aquela função escalar são alcançadas sucessivamente as soluções 3 e 4 (ambas fora da zona de primeira prioridade). Prosseguindo com a pesquisa, obtém-se a solução 1. Desta forma, como a região de primeira prioridade foi totalmente pesquisada (a recta de nível associada à solução 1 passou X) e já foram determinadas soluções nas duas regiões de segunda prioridade, então o processo termina. Desta forma, se F1 tiver preferência relativamente a F2 (F1 ➢ F2), então a solução escolhida é a 4; se for F2 a preferencial, é a solução 3 a escolhida; se não existir qualquer preferência entre os objectivos, a solução escolhida é a 3, pois foi a determinada em primeiro lugar.

3º Caso. Suponha-se que as soluções 4 e 5 não existem e F1 ➢ F2 (B1 ➤ B2).

Usando aquela função escalar é alcançada a solução 3. Depois a pesquisa prossegue e é obtida a solução 1. Nesta altura, a região A ficou totalmente pesquisada (a recta de nível que passa pela solução 1 está para além de X), mas como não foi encontrada qualquer solução na região de segunda prioridade de maior preferência (B1) e esta ainda não está totalmente pesquisada, a solução 3 não pode ser escolhida como solução final. Continuando a pesquisa obtém-se a solução 2. Desta forma, como a região B1 ficou totalmente pesquisada (a recta de nível que passa pela solução 2 está para além de Y1) o processo termina, uma vez que se pode escolher a solução 3 que pertence à região B2.

4º Caso. Suponha-se que as soluções 3 e 5 não existem e F2 ➢ F1 (B2 ➤ B1).

Usando aquela função escalar são alcançadas as soluções 4 e 1, por esta ordem. No entanto, a solução 4 não pode ser escolhida, pois apesar da região A estar totalmente pesquisada, ainda não foi encontrada qualquer solução na região B2 e esta ainda não está totalmente pesquisada. Prosseguindo a pesquisa, são obtidas sucessivamente as soluções 2 e 6. Agora, como a região B2 ficou totalmente pesquisada (a recta de nível que passa pela solução 6 está para além de Y2), o processo termina escolhendo-se a solução 4 como solução final, uma vez que pertence à região B1.

5º Caso. Suponha-se que as soluções 3, 4 e 5 não existem.

Usando aquela função escalar, são determinadas as soluções 1 e 2, sucessivamente. Como estas duas soluções pertencem às regiões de menor prioridade (*última chance*) e ainda existem regiões com maior prioridade por pesquisar, o processo não pode terminar. A pesquisa prossegue e é obtida a solução 6. Desta forma, como as regiões de primeira e segunda prioridades estão totalmente pesquisadas (a recta de nível que passa pela solução 6 está para além de Y2) e a solução 6 pertence à região de terceira prioridade (é a única), então o processo termina com a escolha da solução 6 como solução final.

6º Caso. Suponha-se que as soluções 3, 4, 5 e 6 não existem.

Usando aquela função escalar são alcançadas sucessivamente as soluções 1 e 2. Ao prosseguir com a pesquisa, obtém-se a solução 7. Como as regiões de primeira, segunda e terceira prioridades estão totalmente pesquisadas (a recta de nível que passa pela solução 7 está para além de Z) e a solução 7 é de última chance, então o processo termina escolhendo a solução 1 (foi a primeira do seu tipo) como solução final.

10.7. Aplicação prática

Nesta secção são resolvidos dois problemas de encaminhamento : com duas e com três funções objectivo.

10.7.1. Caso bi-objectivo

Considere-se uma rede não orientada, gerada aleatoriamente, com 21 nós e 39 arcos, em que cada arco tem associados dois valores. Pretende-se determinar um caminho entre os nós 1 e 18 nesta rede. A Fig. 59 apresenta o resultado da resolução do problema especificado, para o qual se introduziram os seguintes dados :

- o objectivo 2 tem preferência em relação ao objectivo 1 (F2 ➢ F1);
- requisitos de QoS para a função objectivo 1 : 500 (requerido) e 600 (aceitável);
- requisitos de QoS para a função objectivo 2 : 300 (requerido) e 380 (aceitável).

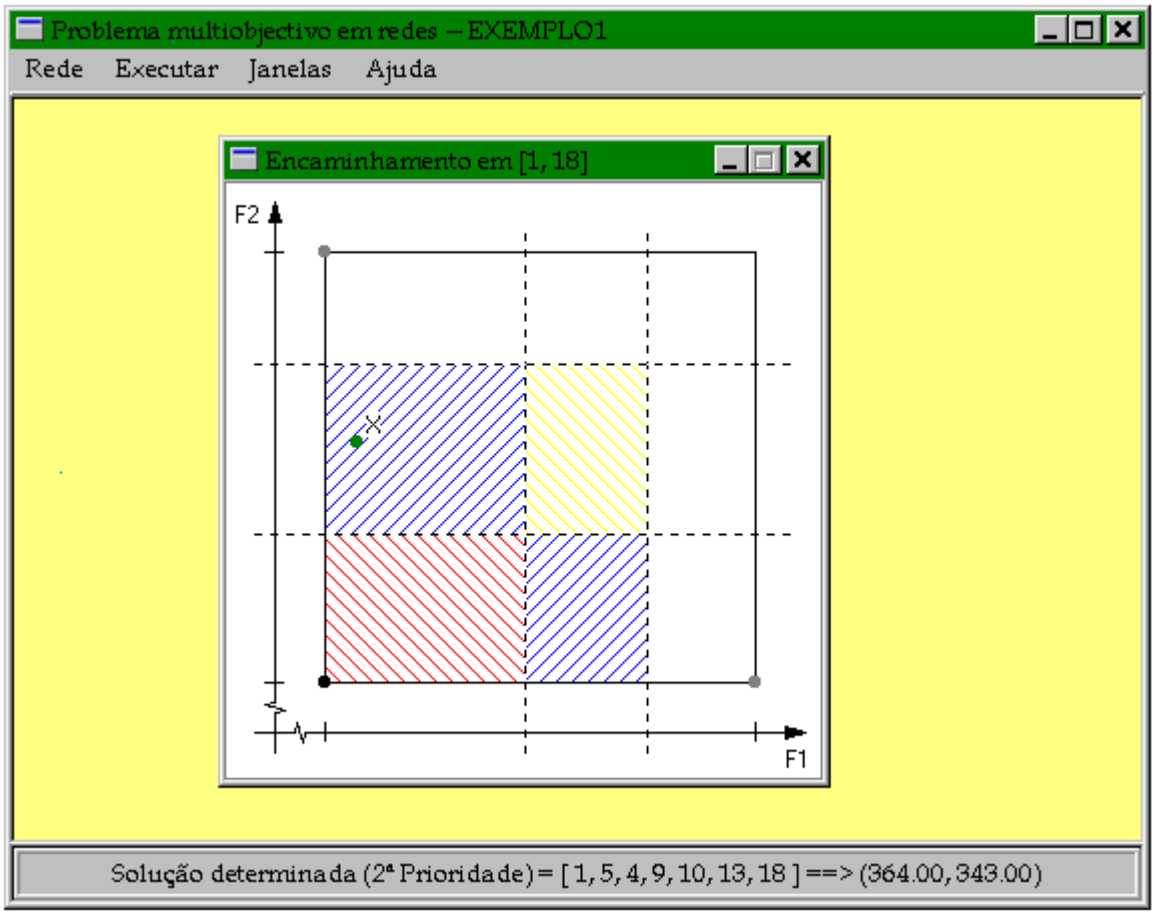


Fig. 59 – Problema de encaminhamento (bi-objectivo) : exemplo.

A solução determinada (ver Fig. 59) foi a seguinte :

p = [1, 5, 4, 9, 10, 13, 18] com um custo de c(p) = (364, 343),

que pertence à região de segunda prioridade de menor preferência (B1).

10.7.2. Caso tri-objectivo

Considere-se uma rede orientada, gerada aleatoriamente, com 50 nós e 250 arcos, em que cada arco tem associados três valores. Pretende-se determinar um caminho entre os nós 15 e 45 nesta rede. A Fig. 60 apresenta o resultado da resolução do problema especificado, para o qual se introduziram os seguintes dados :

- o objectivo 1 tem preferência em relação ao 2 e este em relação ao 3 (F1 ➢ F2 ➢ F3);
- requisitos de QoS para a função objectivo 1 : 1500 (requerido) e 2000 (aceitável);
- requisitos de QoS para a função objectivo 2 : 600 (requerido) e 1000 (aceitável);
- requisitos de QoS para a função objectivo 3 : 1400 (requerido) e 2000 (aceitável).

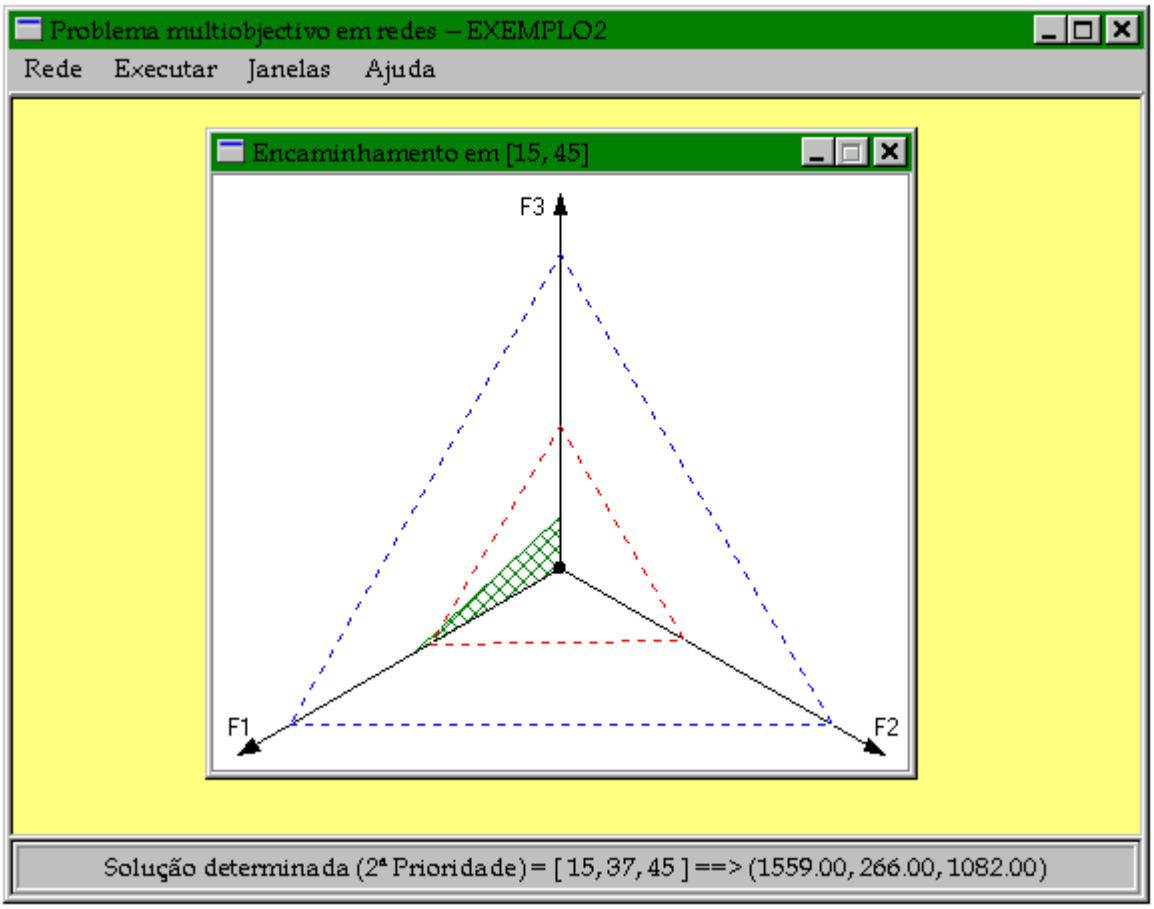


Fig. 60 – Problema de encaminhamento (tri-objectivo) : exemplo.

A solução determinada (ver Fig. 60) foi a seguinte :

p = [15, 37, 45] com um custo de c(p) = (1559, 266, 1082),

que pertence à região de segunda prioridade (apenas não foi satisfeito o valor requerido da função objectivo 1, e portanto foram satisfeitos os outros dois) de menor preferência.

C A P Í T U L O 7

Conclusões e Desenvolvimentos futuros

1. Conclusões

Muitos dos problemas reais que podem ser modelados através do recurso à teoria de optimização em redes, são de natureza multiobjectivo. Os objectivos geralmente envolvidos nestes problemas são os seguintes : custo, tempo, distância, acessibilidade, satisfação exigida, protecção do ambiente, minimização do risco, segurança, entre outros.

Na resolução de problemas com apenas um objectivo, procura-se encontrar a solução óptima, ou seja, a solução admissível que optimize a função objectivo, cujo valor é único.

Na resolução de problemas multiobjectivo, esse conceito não se aplica, uma vez que uma solução admissível que optimize um dos objectivos, não optimiza, em geral, os restantes objectivos, quando estes estão em conflito. Desta forma, a noção de solução óptima é substituída pela noção de solução não dominada, e o resultado pretendido é a determinação de uma solução de compromisso satisfatória entre o universo das soluções não dominadas.

De uma maneira geral, são de três tipos os métodos para resolver problemas de programação multiobjectivo (em particular os problemas de caminho mais curto), conforme as preferências do AD são introduzidas *a priori*, *a posteriori* ou *progressivamente*, no processo de decisão.

Nos métodos em que é feita uma agregação *a priori* das preferências do AD, o problema inicial é transformado num problema com um único objectivo, através, por exemplo, da construção de uma função utilidade, agregando os objectivos envolvidos a partir das preferências do AD. Desta forma, a solução óptima deste último problema é a solução do

problema multiobjectivo original. No entanto, a grande dificuldade está na construção da função utilidade, uma vez que é difícil obter os parâmetros que agreguem, numa única dimensão todos os objectivos envolvidos.

Nos métodos em que as preferências do AD são incorporadas *a posteriori*, determinam-se inicialmente todas as soluções não dominadas do problema, ou parte delas, as quais são posteriormente apresentadas ao AD para que este proceda à sua selecção. Apesar de existirem algoritmos que permitem, com maior ou menor esforço computacional, determinar todas as soluções não dominadas de vários tipos de problemas multiobjectivo, torna-se complicado escolher uma solução no conjunto das soluções não dominadas, que pode ser muito vasto, e onde muitas vezes as soluções possuem características muito semelhantes.

Os métodos em que as preferências do AD são incorporadas *progressivamente* são designados métodos interactivos. Em geral, determinam um pequeno número de soluções não dominadas (fase de cálculo), e depois o AD é chamado a indicar as suas preferências (fase de diálogo), de forma a utilizarem esta informação para gerar novas soluções não dominadas. Normalmente este processo é repetido até se obter uma solução satisfatória para o AD. Estes métodos revelam-se mais eficazes na procura de uma solução de compromisso satisfatória, uma vez que ao aproveitarem a intervenção do AD reduzem a zona de pesquisa, minimizando assim quer o esforço computacional, quer o esforço do AD no processamento da informação. Para tirar partido das capacidades de processamento de informação do AD, devem utilizar-se gráficos e meios de interacção adequados, de forma a ajudar o AD a analisar os compromissos entre os objectivos, que são inerentes às várias soluções.

Com este trabalho pretende-se apresentar uma abordagem ao problema de caminho mais curto bi e tri-objectivo, em redes orientadas e não orientadas. Esta abordagem foi implementada no quadro operacional de sistemas de apoio à decisão, os quais servem para ajudar o AD (utilizador) a identificar a "melhor" solução de compromisso, de acordo com as suas preferências.

Nos sistemas referidos foram desenvolvidos meios gráficos adequados quer à interpretação dos resultados (soluções) que vão sendo obtidos, quer ao diálogo com o AD, tendo como preocupação tornar os sistemas fáceis de compreender e de utilizar.

Relativamente à aplicação da abordagem algorítmica ao problema de encaminhamento em redes integradas de comunicações, conclui-se que é vantajoso considerar uma abordagem multiobjectivo na modelação deste problema, uma vez que desta forma é possível compreender os compromissos entre as diferentes QoS. Desta forma, os modelos tornam-se

mais realistas, assumindo a natureza multiobjectivo do problema, permitindo assim compreender os conflitos inerentes e os compromissos entre os diferentes objectivos, para seleccionar a "melhor" solução de compromisso do conjunto das soluções não dominadas.

2. Desenvolvimentos futuros

Com o desenvolvimento dos algoritmos para determinar os k caminhos mais curtos, os métodos existentes para resolver problemas de caminho mais curto com dois objectivos, que utilizam este processo na identificação de soluções não dominadas nas Zonas de Desníveis de Dualidade, já são suficientemente eficientes e rápidos, até porque os gráficos utilizados para apresentarem os compromissos entre os dois objectivos associados às soluções identificadas revelam relativamente bem esses compromissos.

Relativamente a métodos para resolver problemas deste tipo, apesar de já se ter um grau de desenvolvimento muito elevado (existem alguns trabalhos nesta área), qualquer melhoria nos algoritmos utilizados é vista com bons olhos. No que respeita aos interfaces, pode haver introdução de novas tecnologias no diálogo com o agente de decisão, tais como som, multimedia, etc..

No que respeita aos problemas com três e mais objectivos, existem poucos trabalhos nesta área. Apesar de os algoritmos utilizados nos problemas com dois objectivos poderem ser aplicados a este caso, a dificuldade está na determinação das soluções suportadas (que pertencem ao Contorno Convexo), na definição das Zonas de Desníveis de Dualidade, e ainda nos gráficos utilizados para se apresentar os compromissos entre os objectivos, inerentes às soluções não dominadas identificadas.

Relativamente ao cálculo de soluções do Contorno Convexo, pode não ser possível determinar todas as soluções deste tipo, pois pode ser difícil determinar certas combinações de pesos que conduzam ao cálculo de certas soluções do Contorno Convexo. De facto, a técnica de tomar três soluções deste tipo adjacentes pode não garantir a obtenção de uma solução não dominada, quando uma das componentes do gradiente formado a partir daquelas três soluções é negativa. Portanto, esta questão terá que ultrapassada em futuras investigações.

Em consequência do que se disse no parágrafo anterior, pode não ser possível determinar todas as Zonas de Desníveis de Dualidade. Por outro lado, podem existir soluções não dominadas que não pertençam a qualquer Zona de Desnível de Dualidade, segundo a

definição utilizada neste trabalho. Também esta questão terá que analisada com mais rigor em futuros desenvolvimentos desta área.

Relativamente aos gráficos utilizados na apresentação das soluções, dever-se-ão procurar formas cada vez mais sugestivas de traduzir os compromissos entre os objectivos, associados às soluções não dominadas identificadas. Portanto, também esta será uma área a considerar em desenvolvimentos futuros.

No que respeita à abordagem proposta para o problema de encaminhamento em redes integradas de comunicações, podem considerar-se duas possibilidades para a sua aplicação.

A primeira está associada à implementação de uma nova variante dum método do tipo Encaminhamento Dependente do Estado Periódico ("Periodic State Dependent Routing" — PSDR) para redes de comutação por circuitos. A versão original deste tipo de método de encaminhamento dinâmico baseia-se num tipo centralizado de controlo que fornece decisões de encaminhamento baseadas em actualizações periódicas do número de circuitos livres em cada linha da rede destino. A variante considerada, baseada numa abordagem multiobjectivo, utilizará uma versão automática do algoritmo para determinar soluções não dominadas correspondentes ao número de caminhos alternativos seleccionados para cada par de nós.

A segunda possibilidade está associada ao encaminhamento de fluxos contínuos (tais como audio e vídeo em redes de comutação por pacotes), onde o problema básico consiste em determinar para cada fluxo um caminho que satisfaça os vários objectivos conflituosos e as várias restrições, os quais foram a motivação original do paradigma do encaminhamento com QoS. Neste contexto, uma versão automática do algoritmo multiobjectivo será, em princípio, aplicada tendo em atenção a extrema eficiência do algoritmo para determinar os k caminhos mais curtos utilizado no método proposto.

Um outro aspecto a considerar em desenvolvimentos futuros é a possibilidade destas abordagens serem testadas com redes reais, uma vez que os testes aqui efectuados apenas utilizaram exemplos puramente académicos.

REFERÊNCIAS

[1] Ahuja, R. K., Magnanti, T. L. e Orlin, J. B., "Network Flows – Theory, Algoritms, and Applications", Prentice–Hall, 1993.

[2] Andriole, Stephen (editor), "Microcomputer Decision Support Systems : Design, Implementation and Evaluation", North Holland, 1986.

[3] Antunes, C. H., Alves, M. J., Silva, A. L. e Clímaco, J., "An integrated MOLP method base package — a guided tour of TOMMIX.", Computers and Operations Research 19, nº 7, 609–625, 1992.

[4] Antunes, C. H., Clímaco, J., Craveirinha, J. e Barrico, C., "Multiple Objective Routing in Integrated Communication Networks", Submetido para publicação, 1998.

[5] Clímaco J.C.N. e Martins, E.Q.V., "A bicriterion shortest path problem", European Journal of Operational Research 11, 399 – 404, 1982.

[6] Clímaco, J., "Programação Matemática com Objectivos Múltiplos — Alguns Problemas e uma Aplicação ao Planeamento de Centrais Produtoras de Energia Eléctrica", Dissertação de Doutoramento, Universidade de Coimbra, 1982.

[7] Clímaco, J. N., Antunes, C. H. e Alves, M. J., "Programação Linear Multiobjectivo. Métodos Interactivos, "Software" e Aplicações", Faculdade de Economia da Universidade de Coimbra, 1996.

[8] Clímaco, J. C. N. e Rodrigues, J. M. C., "Note on an Interactive Bicriteria Shortest Path Algoritm", Project Management and Scheduling Workshop, Lisbon, 1988.

[9] Cohon, J. L., "Multiobjective Programming and Planning", Academic Press, 1978.

[10] Current, J. R., Revelle, C. S. e Cohon, J. L., "An interactive approach to identify the best compromise solution for two objective shortest path problems", Computers and Operations Research, Vol. 17, 187 – 198, 1990.

[11] Denardo, E. V., "Dynamic Programming, Theory and Applications.", Prentice–Hall, Englewood Cliffs, NJ, 1980.

[12] Dreyfus, S., "An Appraisal of Some Shortest Path Algorithms", Operations Research 17, 395 – 412, 1969.

[13] Hansen, P., "Bicriterion path problems", em Lecture Notes in Economics and Mathematical Systems 177, Beckman, M. e Kunzi, H. P. (eds), 109 – 127, Springer Verlag, 1980.

[14] Hartley, R., "Finite, discounted, vector Markov decision process", Notes in Decision Theory (Note 85), University of Manchester, 1979.

[15] Henig, M. I., "The shortest path problem with two objectives functions", European Journal of Operational Research 25, 281 – 291, 1985.

[16] Lee, W. C., Hluchyj, M. G. e Humblet, P. A., "Routing Subject to Quality of Service Constraints in Integrated Communication Networks", IEEE Network, 46 – 55, 1995.

[17] Martins, E. Q. V., "Determinação de Caminhos Optimos em Redes Orientadas", Dissertação de Doutoramento, Universidade de Coimbra, 1984.

[18] Martins, E.Q.V., "On multicriteria shortest path problem algorithm", European Journal of Operational Research 16, 236 – 245, 1984.

[19] Martins, E. Q. V. e Santos, J. L. E., "A new shortest paths ranking algorithm", Submetido para publicação, 1996.

[20] Martins, E. Q. V., Pascoal, M. M. B. e Santos, J. L. E., "A new algorithm for ranking loopless paths", Submetido para publicação, 1997.

[21] Rodrigues, J. M. C., "Apoio à decisão na circulação em redes", Dissertação de Doutoramento, Universidade de Coimbra, 1993.

[22] Roy, B., "Méthodologie Multicritèrie d'Aide à la Decision", Economica, Paris, 1986.

[23] Sage, A., "An Overview of Contemporary Issues in Design and Development of Microcomputer Decision Support Systems" in Microcomputer Decision Support Systems : Design, Edit. S. Andriole, 3 - 46, Implementation and Evolution, North Holland, 1986.

[24] Shin, M., "Computational methods for Markov decision problems.", Dissertação de Doutoramento, University of British Columbia, Canadá, 1980.

[25] Steuer, R., "Multiple Criteria Optimization : Theory, Computation and Application.", Wiley, 1986.

[26] Wang, Z. e Crowcroft, J., "Quality-of-Service Routing for Supporting Multimedia Applications", IEEE Journal on Selected Areas in Communications, Vol 14, nº 7, 1228 – 1234, 1996.