versão impressa ISSN 0101-7438 / versão online ISSN 1678-5142

Seção de *Software*

Virgílio José Martins Ferreira Filho
Departamento de Engenharia Industrial
Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro – RJ
virgilio@ufrj.br

SABILOC – UM SISTEMA DE APOIO À DECISÃO PARA ANÁLISE DE PROBLEMAS DE LOCALIZAÇÃO BICRITÉRIO

Sérgio Fernandes*
Departamento de Matemática
Escola Superior de Tecnologia – Instituto Politécnico de Setúbal
Setúbal – Portugal
sergiof@est.ips.pt

M. Eugénia Captivo
Centro de Investigação Operacional
Faculdade de Ciências
Universidade de Lisboa
Lisboa – Portugal
mecaptivo@fc.ul.pt

João Clímaco
Faculdade de Economia e INESC – Coimbra
Universidade de Coimbra
Coimbra – Portugal
jclimaco@inescc.pt

* *Corresponding author* / autor para quem as correspondências devem ser encaminhadas

1. Introdução

Num problema de localização, pretende decidir-se da melhor forma possível onde localizar um certo número de serviços, de forma a servir um conjunto de comunidades cujas localizações e, por vezes, outras características, tais como a procura ou o peso, são conhecidas.

Essencialmente, os problemas de localização são classificados de acordo com o objectivo a optimizar e com as características do espaço das soluções. Podemos ter problemas de localização no plano Euclideano ou em rede. Os modelos de localização no plano, geralmente, envolvem problemas em que as possibilidades de localização de serviços são infinitas. Os problemas em rede são contínuos (localização de serviços nos vértices ou em qualquer ponto dos arcos) ou discretos (localização de serviços unicamente nos vértices). Normalmente, os locais potenciais de instalação de serviços não existem em qualquer ponto do espaço, o que faz com que os modelos discretos sejam os mais utilizados. Particularmente quando se trata da instalação de serviços indesejáveis. Por outro lado, ao contrário dos serviços indesejáveis que se querem longe das comunidades, os serviços desejáveis, que geralmente envolvem deslocações, querem-se nas proximidades. Como consequência, é de todo o interesse instalar os serviços de modo a que as comunidades estejam o mais próximo possível deles (em termos de distância dada pelo comprimento do caminho mais curto), ou seja, minimizar uma função distância entre serviços e comunidades (tal como, a soma ponderada das distâncias entre as comunidades e os serviços, ou a maior distância entre qualquer comunidade e o serviço mais próximo).

Durante as duas últimas décadas, tem havido um crescente aumento de interesse em problemas relacionados com o ambiente. A sociedade em geral tem-se preocupado com a manutenção da qualidade de vida em regiões onde poderá proceder-se à instalação de determinados serviços com efeitos desagradáveis, nomeadamente fábricas poluentes, incineradoras, aterros sanitários, antenas de telecomunicações, instalações militares, *etc*. No entanto, apesar da contestação, estes serviços são indispensáveis ao ser humano, e como tal, precisam de ser instalados. Enquanto que os primeiros modelos a serem estudados consideravam a localização de serviços desejados, apareceu nos últimos anos outro tipo de serviços designados por indesejáveis, que, pelo seu efeito desagradável, ou mesmo nocivo, diminuem a qualidade de vida das populações mais próximas. Assim, a determinação dos locais de instalação de serviços deste tipo deve ser feita de modo a que estes estejam o mais longe possível das comunidades (em termos de distância Euclideana), de forma a minimizar o efeito desagradável exercido sobre as comunidades, e simultaneamente, maximizar a acessibilidade destas aos serviços instalados mais próximos. Consideramos que estes objectivos contraditórios que caracterizam o problema de localização de serviços indesejáveis devem ser encarados numa modelação sob a perspectiva de duas funções objectivo. Naturalmente, a modelação dos aspectos ambientais sob a forma de objectivos, como alternativa à utilização das restrições, produzirá mais informação (Current *et al*., 1990).

Foi nesta perspectiva, e tendo em conta outras vantagens dos modelos bicritério, como veremos adiante, que decidimos implementar um Sistema de Apoio à Decisão (SAD), denominado SABILOC que tem como objectivo apoiar os decisores na localização de serviços considerando dois objectivos. Os modelos a que se destina o sistema, neste momento, são o de localização simples bicritério (PLSB) e o de *p*-localização bicritério (PPLB). O primeiro permite uma base de trabalho mais alargada. Nem a interacção entre os serviços, nem o número ou tipo de

serviços a instalar é predeterminado. Além disso, devido à simplicidade da formulação, os resultados obtidos interpretam-se muito facilmente. Pelo mesmo motivo, também podem fazer-se facilmente modificações, de modo a torná-lo mais adequado à realidade.

Neste trabalho, definem-se os modelos em questão e ilustra-se a utilização do SABILOC desenvolvido através de uma instância do PLSB.

O protótipo do sistema poderá ser descarregado a partir do endereço electrónico http://cio.fc.ul.pt/.

2. Definição dos modelos em estudo e considerações preliminares

Consideremos um conjunto de clientes $I = \{1,2,...,m\}$ e um conjunto de locais possíveis para a instalação de serviços, $J = \{1,2,...,n\}$. Sejam h_j e g_j os valores fixos (custos, lucros, …) associados à instalação do serviço *j* relativamente a cada função objectivo e assuma-se que l_{ij} e d_{ij} são os valores (custos, lucros, …) associados à afectação do cliente *i* ao serviço *j*, também para cada uma das funções objectivo. Podemos considerar, por exemplo, que uma função objectivo mede o custo e a outra o risco associado à instalação de serviços indesejáveis, e respectiva afectação das comunidades ou clientes.

Sejam as variáveis:

$$x_{ij} = \begin{cases} 1 & \text{se o cliente } i \text{ é afecto ao serviço } j \\ 0 & \text{caso contrário} \end{cases}$$

e

$$y_j = \begin{cases} 1 & \text{se o serviço } j \text{ é instalado} \\ 0 & \text{caso contrário} \end{cases}.$$

Assim, o problema de localização simples com dois objectivos pode ser formulado em Programação Linear Inteira Bicritério da seguinte forma:

(PLSB)
$$\min \sum_{j\in J}\sum_{i\in I} l_{ij}x_{ij} + \sum_{j\in J} h_j y_j \tag{1}$$

$$\min \sum_{j\in J}\sum_{i\in I} d_{ij}x_{ij} + \sum_{j\in J} g_j y_j \tag{2}$$

s. a:
$$\sum_{j\in J} x_{ij} = 1 \qquad \forall i \in I \tag{3}$$

$$x_{ij} \le y_j \qquad \forall i \in I,\ \forall j \in J \tag{4}$$

$$y_j \in \{0,1\} \qquad \forall j \in J \tag{5}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in I,\ \forall j \in J \tag{6}$$

Neste modelo, as funções objectivo minimizam os valores associados à afectação dos clientes aos serviços e à instalação de serviços. As restrições (3) garantem que todos os clientes são afectos a exactamente um serviço, enquanto que as restrições (4) garantem que os clientes só serão afectos a serviços instalados.

No problema de p-localização bicritério a única diferença consiste na existência de uma restrição adicional que fixa o número de serviços instalados em p.

$$\sum_{j \in J} y_j = p \tag{7}$$

De entre os problemas multiobjectivo, os que consideram dois objectivos têm sido extensivamente estudados e apresentam algumas vantagens. Entre elas, a possibilidade de desenvolver técnicas especializadas para a sua resolução eficiente. Uma outra vantagem corresponde à eficácia com que se comunica o conjunto de soluções ao decisor, quer seja na forma gráfica ou de tabela.

Tradicionalmente, os métodos de optimização multicritério classificam-se em três categorias, de acordo com o processo utilizado para agregar/articular as preferências do decisor, com vista a seleccionar uma solução de compromisso. A saber:

- métodos em que não há agregação ou articulação de preferências (geradores de todas as soluções eficientes);
- métodos em que é feita uma agregação *a priori* de preferências (de que se destacam os que usam uma função utilidade);
- métodos em que é feita uma articulação progressiva de preferências (interactivos).

Os métodos interactivos surgiram pela necessidade de tirar partido da intervenção do decisor para reduzir progressivamente o âmbito da pesquisa e minimizar quer o esforço computacional, quer o esforço cognitivo que é requerido ao decisor.

Duas perspectivas têm dominado a implementação de métodos interactivos: "orientada para a pesquisa" e "orientada para a aprendizagem" (Gardiner & Vanderpooten, 1997). A perspectiva "orientada para a procura" parte do princípio que o decisor age de acordo com uma estrutura de preferências pré-existente e inalterável (representada por uma função utilidade implícita), que pode ser eliciada sem contradições. O objectivo da interacção é a descoberta de uma solução óptima da função utilidade implícita (que não depende da evolução do processo interactivo), face à estrutura de preferências. A outra perspectiva, "orientada para a aprendizagem", não contempla a existência de uma estrutura de preferências bem definida. As preferências do decisor podem ser parcialmente alteradas e contraditórias. O objectivo da interacção é a aprendizagem das preferências, ou seja, a clarificação do que, na perspectiva do decisor, possa ser uma solução interessante. Compete ao decisor terminar o processo quando se sentir satisfeito com uma das soluções encontradas.

Para o tipo de problema que nos propomos estudar, o problema de localização bicritério, o número de soluções eficientes pode ser elevado. Não é realista apresentar ao decisor todas as soluções e esperar que ele tenha capacidade de escolher a que mais lhe agrada. De facto, o decisor revela-se incapaz de fazer uma escolha fundamentada quando é confrontado com uma grande quantidade de informação, isto é, um grande número de soluções eficientes, principalmente quando muitas delas apresentam apenas variações ligeiras no que se refere aos valores das funções objectivo (Clímaco *et al*., 2003). Acreditamos que os métodos interactivos são a melhor escolha, especialmente se forem concebidos como orientados para a aprendizagem (melhorando o conhecimento sobre o problema) e não como métodos que procuram uma solução óptima de uma qualquer função utilidade implícita.

Torna-se assim necessário resolver, em cada interacção, um problema de optimização com uma só função objectivo, de forma a obtermos uma solução eficiente do problema de

localização simples bicritério apresentado no início desta secção. Um dos procedimentos mais utilizados na obtenção de soluções eficientes para o problema de programação linear multiobjectivo é a optimização da soma ponderada das funções objectivo. No entanto, a utilização directa desta metodologia para o problema de programação linear inteira multiobjectivo não garante a obtenção das soluções eficientes na sua totalidade, devido à existência de soluções não dominadas nos *gaps* de dualidade.

Para fazer face a esta situação, recorremos a um resultado, de Ross & Soland (1980), de grande aplicabilidade para problemas bicritério de programação linear inteira. Este resultado mostra que para determinar uma solução eficiente para o problema de programação linear inteira bicritério

$$\begin{aligned} \max \quad & f_1(x) \\ \max \quad & f_2(x) \\ \text{s. a:} \quad & Ax = b \\ & x \geq 0 \text{ e inteiro} \end{aligned}$$

é suficiente calcular a solução óptima do seguinte problema, para $\alpha_1, \alpha_2 \in \mathbb{R}$:

$$\begin{aligned} \max \quad & \lambda_1 f_1(x) + \lambda_2 f_2(x) \\ \text{s. a:} \quad & Ax = b \\ & x \geq 0 \text{ e inteiro} \\ & f_1(x) \geq \alpha_1 \\ & f_2(x) \geq \alpha_2 \end{aligned}$$

em que λ_1 e $\lambda_2 \in \mathbb{R}$ representam pesos positivos e que satisfazem $\lambda_1 + \lambda_2 = 1$.

Assim, utilizando o resultado anterior, é possível gerar todas as soluções não dominadas (suportadas e não suportadas) do problema. Para tal, terá de ser resolvido o seguinte problema monocritério com duas restrições adicionais para diferentes valores de α_1 e α_2:

(PLSB-M)
$$\min \sum_{i\in I}\sum_{j\in J}\left(\lambda_1 l_{ij} + \lambda_2 d_{ij}\right)x_{ij} + \sum_{j\in J}\left(\lambda_1 h_j + \lambda_2 g_j\right)y_j \qquad (8)$$

s. a:
$$\sum_{j\in J} x_{ij} = 1 \qquad \forall i \in I \qquad (3)$$

$$x_{ij} \leq y_j \qquad \forall i \in I,\ \forall j \in J \qquad (4)$$

$$y_j \in \{0,1\} \qquad \forall j \in J \qquad (5)$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in I,\ \forall j \in J \qquad (6)$$

$$\sum_{j\in J}\sum_{i\in I} l_{ij}x_{ij} + \sum_{j\in J} h_j y_j \leq \alpha_1 \qquad (9)$$

$$\sum_{j\in J}\sum_{i\in I} d_{ij}x_{ij} + \sum_{j\in J} g_j y_j \leq \alpha_2 \qquad (10)$$

$$\lambda_1 + \lambda_2 = 1,\ \lambda_1, \lambda_2 > 0 \qquad (11)$$

Dias, Captivo & Clímaco (2003), fazendo um uso adequado deste modelo, desenvolveram um algoritmo eficiente para a resolução do PLSB-M, baseado no conhecido algoritmo *Dualoc* de Erlenkotter (1978) para o problema de localização simples. O algoritmo proposto aproveita o mais possível a estrutura do problema de localização simples que o problema PLSB-M apresenta, e trabalha com as duas restrições adicionais através de um processo de pesquisa em árvore (*branch and bound*).

Este algoritmo foi entretanto adaptado para a resolução do problema de *p*-localização bicritério.

3. SABILOC

As arquitecturas abertas, a modularidade, a flexibilidade, a reutilização e outras capacidades que permitam futuras alterações são conceitos a considerar durante o desenvolvimento e implementação dos SAD.

À medida que as capacidades dos sistemas informáticos e a gama de utilização que lhes é dada aumentam, as interfaces vão necessitando de acomodar o aumento de complexidade. É através da interface que o utilizador entra em contacto com o método interactivo, logo, a qualidade desta é um dos principais factores de aceitação do método. Se um decisor encontrar uma interface com a qual sinta que é recompensador trabalhar, cujo funcionamento seja bastante intuitivo, então é muito provável que aceite o método interactivo subjacente. Se, por outro lado, encontrar uma interface incompatível com a sua maneira de agir e pensar, certamente que recusará o método.

A programação orientada por objectos (POO) proporciona as capacidades para tornar as interfaces flexíveis de forma a que parte do código possa ser alterada sem afectar o restante. Estas linguagens são especialmente adequadas para reutilizar *software* existente, com ausência de erros, numa nova aplicação que exija o aumento das funcionalidades (Solanki & Gorti, 1997), ou seja, caso se pretenda construir um sistema modular. Um sistema modular pode ser construído de forma a que a implementação de uma parte seja largamente independente da implementação de outras. As vantagens dos sistemas modulares são:

- É mais fácil manter um sistema construído de uma forma modular, já que é muito menos provável que as alterações num subsistema tenham efeito no restante sistema.
- Pela mesma razão, é mais fácil actualizar um sistema modular. Desde que os módulos de substituição verifiquem as especificações da interface dos seus antecessores, as outras partes do sistema não são afectadas.
- É mais fácil construir um sistema cuja utilização seja fiável. Isto porque os subsistemas podem ser completamente testados quando isolados, deixando menos problemas para mais tarde, quando todo o sistema for integrado.
- Um sistema modular pode ser implementado com pequenos incrementos. Desde que cada módulo seja concebido para proporcionar um pacote útil e coerente de funcionalidades, os incrementos podem ser introduzidos progressivamente.

Todo o programa foi desenvolvido na linguagem de POO Microsoft Visual C++, versão 6.0 e foi criado a pensar em futuras inclusões de outros modelos para além do de localização simples bicritério. As bibliotecas a utilizar como ferramentas externas podem ser programadas em qualquer linguagem, orientada por objectos ou não, desde que satisfaçam os requisitos de comunicação.

Tendo em conta todas as vantagens enunciadas anteriormente quanto aos sistemas modulares, obviamente que decidimos construir um SAD modular, facilitando assim a inclusão de novos modelos, bem como, de novos algoritmos que permitam resolver de forma mais eficiente os problemas bicritério. Para desenvolver o SABILOC de forma modular, houve uma atenção especial com os ficheiros de biblioteca.

Procurou também desenvolver-se e implementar-se uma interface de qualidade, de fácil compreensão e intuitiva na sua utilização, com que o utilizador menos experiente se sinta à vontade para explorar os diversos campos do problema.

A implementação do método interactivo para uso no SABILOC é orientada para a aprendizagem progressiva e selectiva do conjunto das soluções eficientes. O objectivo não é convergir para uma solução óptima de qualquer função utilidade implícita, mas antes apoiar o decisor na eliminação do subconjunto das soluções eficientes que se revelem sem interesse prático. Não há decisões irrevogáveis durante o processo interactivo, sendo permitido ao decisor rever decisões anteriores. Em cada interacção, na fase de diálogo, o decisor apenas terá que dar indicações sobre a área a pesquisar. O processo termina quando o decisor considera já conhecer o suficiente sobre o conjunto das soluções não dominadas. O método interactivo a implementar é baseado no método de Ferreira *et al*. (1994). Procurou explorar-se adequadamente a estrutura do modelo bicritério, de forma a obter soluções não dominadas e a visualizá-las no espaço dos objectivos, facilitando a compreensão do decisor. O facto de estarmos perante problemas bicritério permite que na fase de diálogo sejam apresentadas as informações de uma forma bastante perceptível, como veremos na apresentação do SABILOC desenvolvido. Na fase de cálculo, para gerar soluções eficientes, resolve-se o PLSB-M para diferentes valores de α_1 e α_2, e aplica-se o resultado de Ross e Soland permitindo, como já foi visto, determinar todas as soluções não dominadas do problema, inclusive as que se encontram nos desvios de dualidade.

4. Funcionamento do SABILOC

Descreve-se agora uma sequência específica de acções que o utilizador pode realizar de forma a obter, através do SABILOC proposto, resultados satisfatórios correspondentes a uma instância do problema de localização simples bicritério. Uma das possibilidades será ler um ficheiro de dados criado anteriormente, referente a uma instância do PLSB que se pretenda resolver. O ficheiro de dados terá que ter extensão "md1" e verificar o seguinte formato:

m (n.º de serviços) n (n.º de comunidades)
l_{ij} (matriz correspondente aos custos de afectação da 1ª função objectivo)
h_j (custos fixos da 1ª função objectivo)
d_{ij} (matriz correspondente aos custos de afectação da 2ª função objectivo)
g_j (custos fixos da 2ª função objectivo)

Todos os valores introduzidos têm que ser inteiros.

No exemplo que iremos ver de seguida, a instância será gerada aleatoriamente no próprio SABILOC. A janela inicial contém uma pasta de seis páginas, em que a primeira é denominada "*Constants*" (Figura 1) onde deve ser introduzida a dimensão da instância, ou seja, o número de comunidades e de possíveis locais para instalação de serviços.

Consideremos uma instância com trinta comunidades e dez locais potenciais para a instalação de serviços.

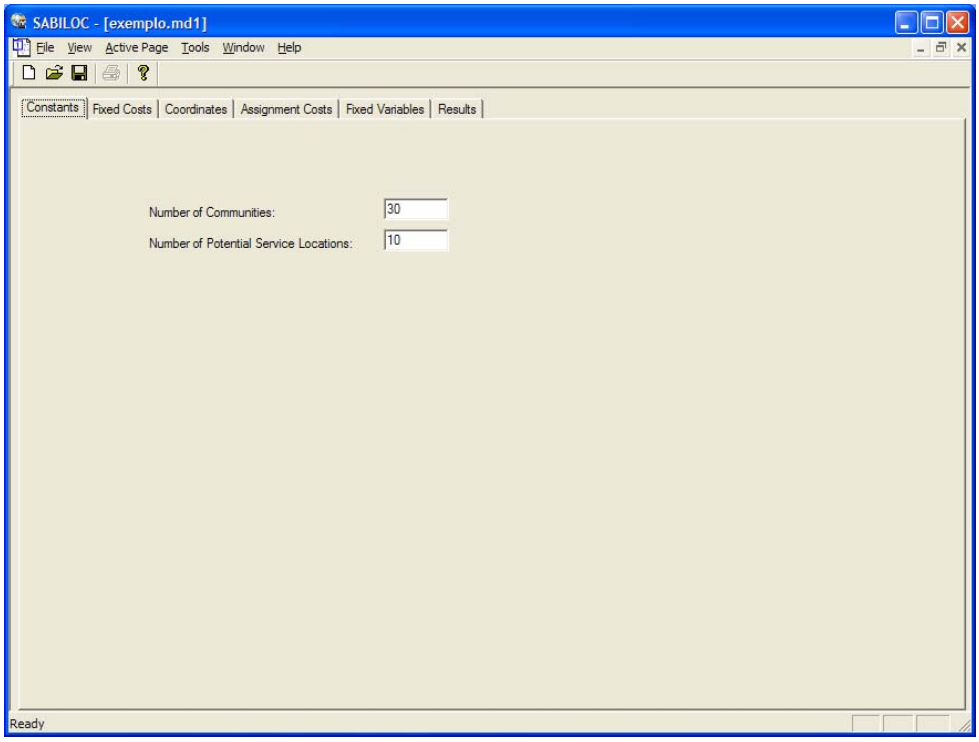

Figura 1 – Janela correspondente à página "*Constants*".

A página seguinte, "*Fixed Costs*" (Figura 2), tal como o nome indica, tem como intuito a inserção dos custos fixos de abertura dos serviços para cada função objectivo, correspondentes aos h_j e g_j do PLSB. Estes valores, tais como todos os outros necessários ao problema, podem ser lidos de um ficheiro de dados ou introduzidos directamente nas grelhas. Para gerar os custos de forma aleatória, seleccionamos o botão "*Randomize*" e aparecerá uma caixa de diálogo (Figura 3) para inserir informação relativa aos valores que se pretendem gerar. Para a instância em causa, foram gerados valores aleatórios, só para a primeira função objectivo, entre zero e quinhentos. Para a segunda função objectivo, os valores foram gerados através de uma outra opção desta página, seleccionando o botão "*Options*" (Figura 3). Para cada serviço, considerou-se a soma dos inversos das distâncias euclideanas às comunidades, multiplicada por uma constante de proporcionalidade que, nesta instância, optou-se por ser igual a 1000. Obviamente que este cálculo só foi possível após a definição das coordenadas dos serviços e das comunidades na página "*Coordinates*". Esta forma de definir os custos fixos associados à instalação de serviços faz sentido quando existem implicações ambientais. Efectivamente, o risco associado à instalação de um serviço indesejável é inversamente proporcional à soma das distâncias euclideanas às comunidades, dependendo também de outros factores como o relevo, a temperatura, o vento, *etc.* (Captivo & Clímaco, 2001).

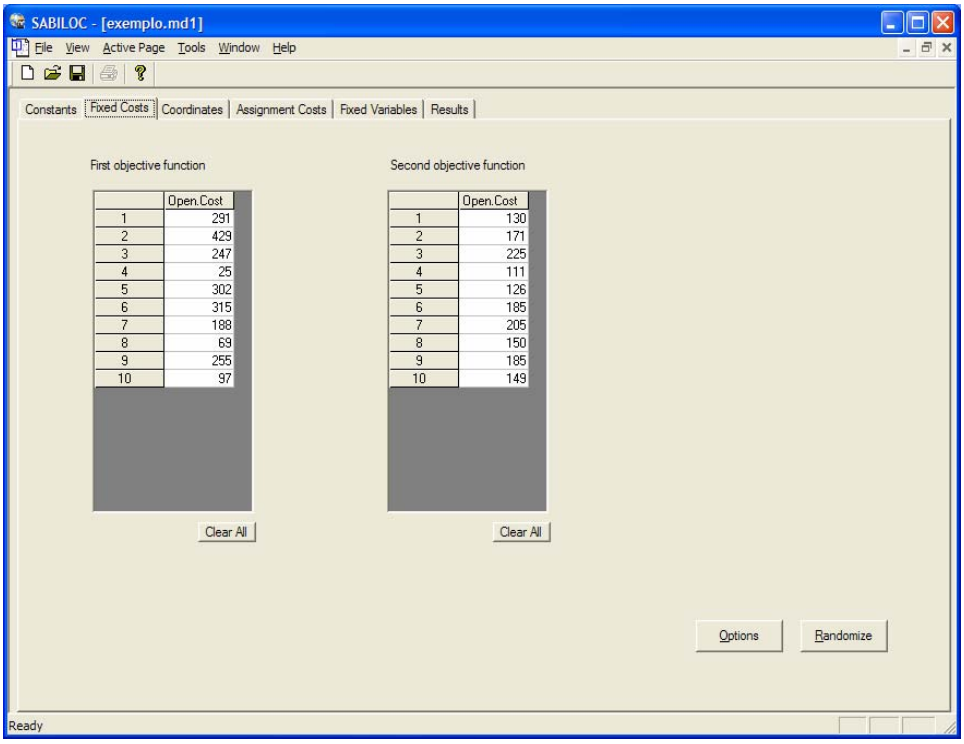

Figura 2 – Janela correspondente à página "*Fixed Costs*".

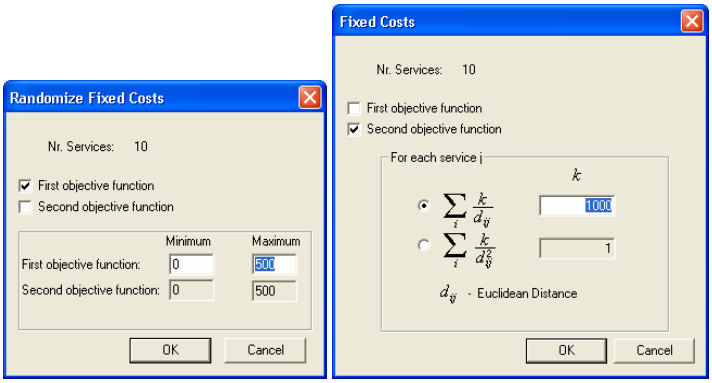

Figura 3 – Caixas de diálogo que permitem gerar os custos fixos.

Tomando em consideração o PLSB, para a definição completa da instância faltam definir as matrizes de custos de afectação para ambas as funções objectivo (l_{ij} e d_{ij}, $\forall i \in I$, $\forall j \in J$). Caso os dados estivessem disponíveis, deviam usar-se as grelhas da página "*Assignment Costs*" para introduzir directamente as matrizes, ou ler o ficheiro de dados correspondente. Como estamos a gerar aleatoriamente uma instância, não temos as matrizes de custos de afectação logo não podemos utilizar, de momento, a página "*Assignment Costs*". No entanto,

os custos de afectação são geralmente proporcionais às distâncias. Em muitas situações reais, a única informação disponível consiste nas coordenadas e/ou nas distâncias entre as diversas localidades, e neste caso, deve usar-se a página "*Coordinates*" (Figura 4).

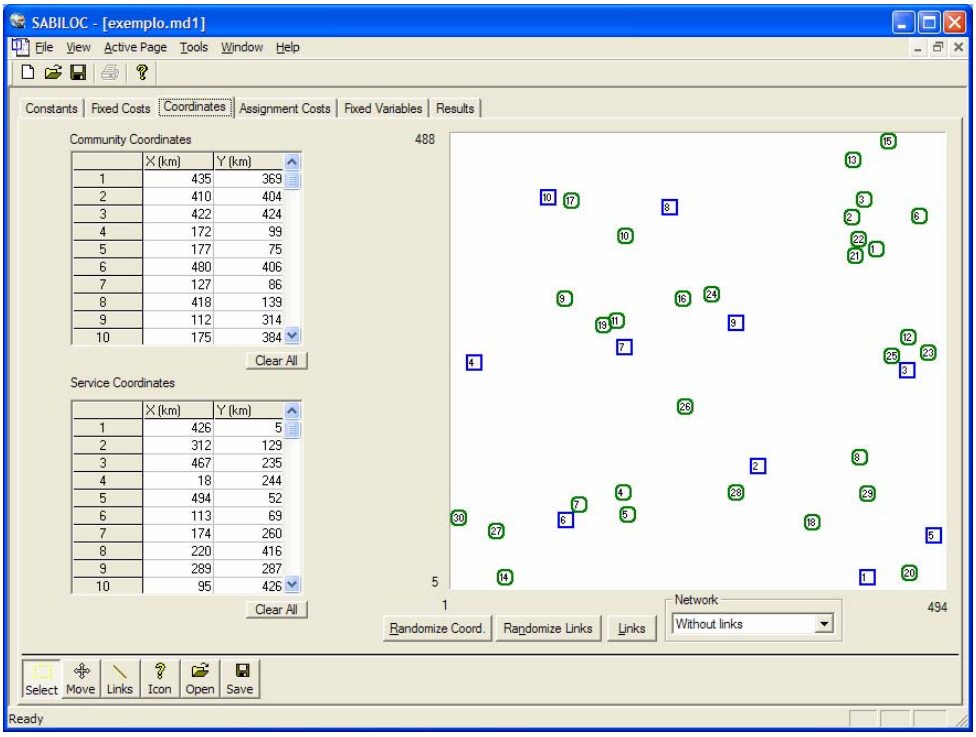

Figura 4 – Janela correspondente à página "*Coordinates*".

As coordenadas também poderão ser geradas de forma aleatória segundo uma distribuição uniforme, bastando para tal premir o botão "*Randomize Coordinates*". Surge então uma outra caixa de diálogo (Figura 5) que permite introduzir os valores mínimos e máximos das coordenadas para os serviços e comunidades. Por omissão, o sistema considera os valores mínimos iguais a zero e os máximos iguais a quinhentos. Consideremos estes valores para gerar as coordenadas.

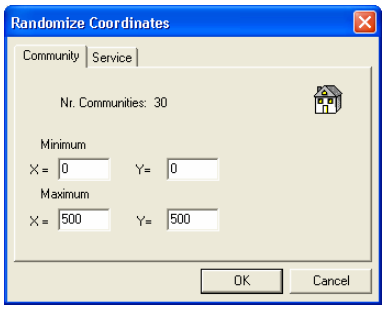

Figura 5 – Caixa de diálogo que permite gerar aleatoriamente as coordenadas.

A partir das coordenadas, podemos definir os arcos existentes entre os nodos da rede, bem como os seus custos (distâncias) para ambos os objectivos. Para tal, ao accionar o botão "*Links*" (Figura 4), abre-se uma caixa de diálogo que permite definir os arcos e o seu custo em cada função objectivo. No entanto, é possível definirmos as características dos arcos a gerar aleatoriamente através da caixa de diálogo que se abre com o premir do botão "*Randomize Links*" (Figura 6).

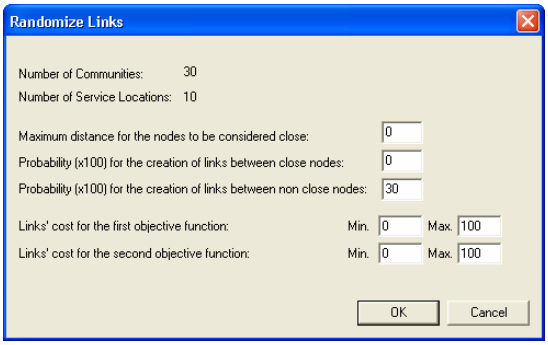

Figura 6 – Caixa de diálogo que permite gerar aleatoriamente os arcos.

Podemos agora visualizar a rede obtida através da Figura 7, onde para não sobrecarregar a figura se encontram apenas os arcos em que um dos extremos é o serviço 9.

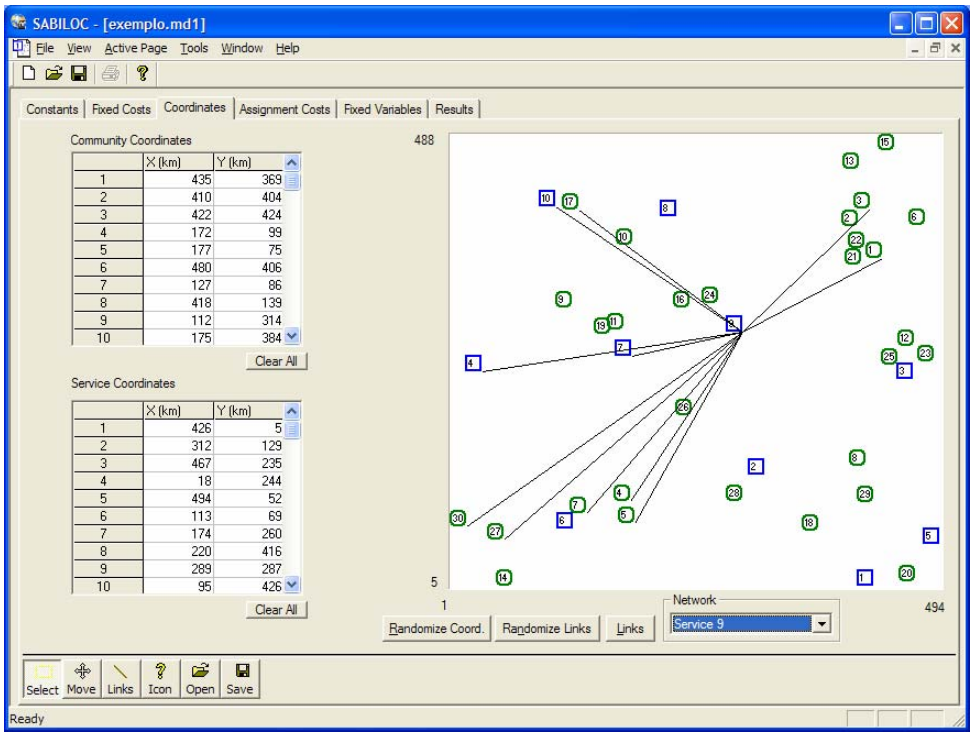

Figura 7 – Visualização dos arcos em que um dos extremos é o serviço 9.

Na página "*Assignment Costs*" (Figura 8) existem três botões para cada função objectivo que permitem o cálculo automático das matrizes de custos de afectação. Um deles tem como denominação "SPM (*shortest path matrix*) – *Link's Costs*" e permite calcular o custo do caminho mais curto entre cada comunidade e cada serviço, tendo em conta os custos dos arcos definidos na página "*Coordinates*".

Um outro botão é denominado "*Euclidean Distance*" e permite o cálculo das distâncias euclideanas (arredondadas às unidades) entre as comunidades e os serviços.

O terceiro botão, denominado "*k* / Euclidean Distance", permite o cálculo do inverso da distância euclideana entre cada comunidade e cada serviço multiplicado por uma constante de proporcionalidade *k*.

Para a instância em análise, optou-se pelo "SPM – *Link's Costs*" para a primeira função objectivo, e pelo "*Euclidean Distance*" para a segunda função objectivo.

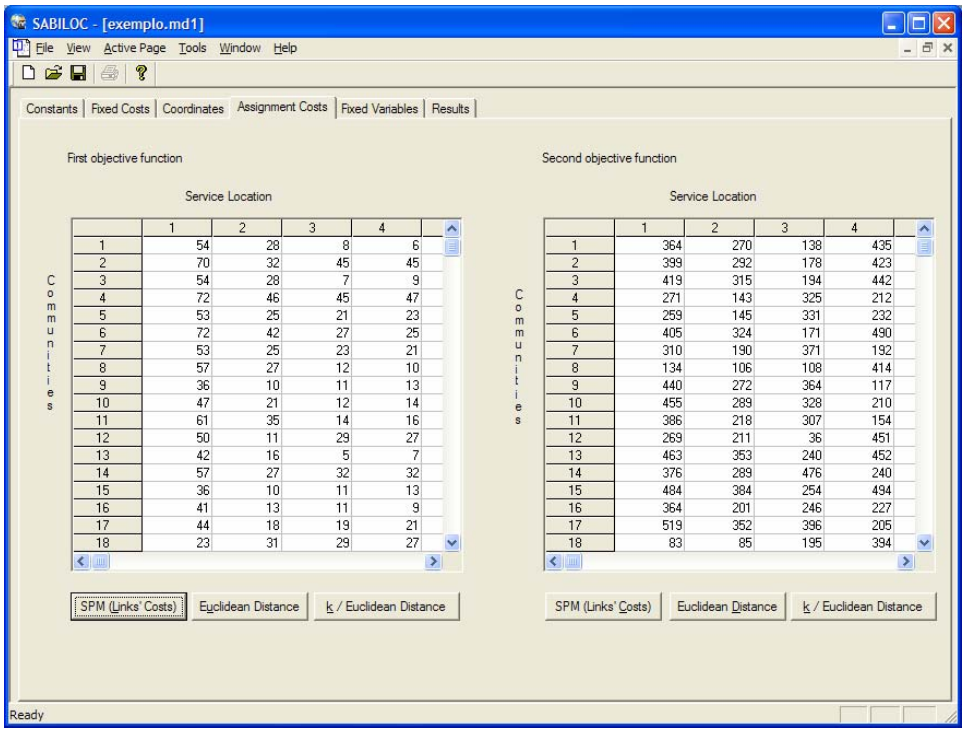

Figura 8 – Janela correspondente à página "*Assignment Costs*".

Em situações reais, acontece que por vezes certos serviços têm obrigatoriamente de ser instalados em locais especificados à partida, quer por razões sociais, económicas e/ou políticas. Acontece também algumas comunidades terem de ficar afectas a serviços específicos, ou inversamente não poderem ficar associadas a determinados serviços. Assim, antes de se iniciar a pesquisa de soluções não dominadas, o SABILOC permite fixar variáveis através da página "*Fixed Variables*" (Figura 9). Isto é, os serviços podem ser definidos como fechados, abertos ou livres, e as afectações entre serviços e comunidades como activadas ou desactivadas. Para fixar as variáveis, como pretendido, temos então de

preencher as respectivas grelhas da página "*Fixed Variables*". Uma das grelhas corresponde às variáveis x_{ij} e a outra às variáveis y_j.

Suponhamos que na instância em causa se pretende que a comunidade 14 seja afecta ao serviço 1 e que o serviço 4 fique fechado nas soluções não dominadas a pesquisar. Ao fixar a afectação da comunidade 14 ao serviço 1 como activada, automaticamente o serviço 1 passa a ser fixo como aberto e todas as afectações dos restantes serviços para a comunidade 14 são desactivadas, pois cada comunidade é afecta a apenas um serviço. Ao fixar o serviço 4 como fechado, automaticamente todas as afectações a este são desactivadas (Figura 9).

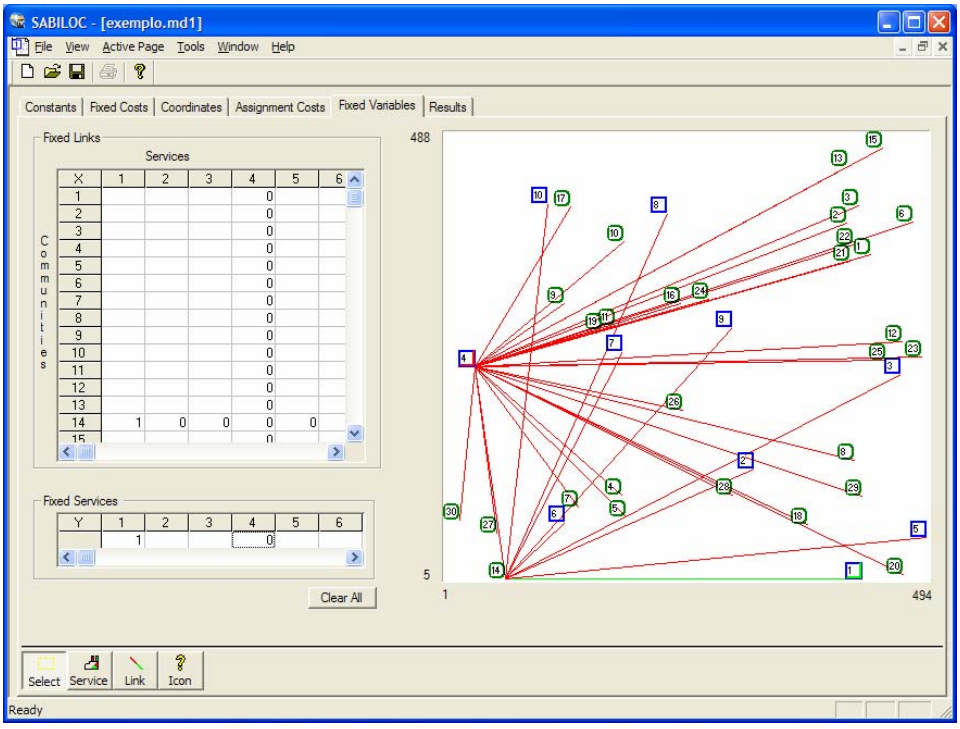

Figura 9 – Janela correspondente à página "*Fixed Variables*".

Saliente-se que a fixação de serviços e afectações tem de obedecer a certas regras de acordo com o modelo utilizado. O SABILOC está preparado para detectar as irregularidades que se pretendam realizar, como por exemplo fechar todos os serviços, afectar uma comunidade a mais que um serviço, ou ainda fechar um serviço, sendo este o único possível de ficar afecto a uma certa comunidade. Assim, vão surgindo várias mensagens de erro e de informação quando são detectadas acções incorrectas por parte do utilizador.

Após a introdução dos dados, podemos iniciar a procura interactiva de soluções eficientes através da página "*Results*". As soluções são calculadas progressivamente premindo o botão "*Execute*" ou seleccionando no menu "*Tools→Results→Execute*". Ao executar a primeira vez, caso a solução ideal não seja admissível, é apresentada a solução óptima para a primeira função objectivo. Na instância gerada, a primeira solução eficiente obtida, ou seja, a que minimiza a função objectivo 1, tem valor óptimo $F_1^* = 1231$ (Figura 10), obtendo-se como "melhor" valor para a outra função objectivo $F_2 = 7462$.

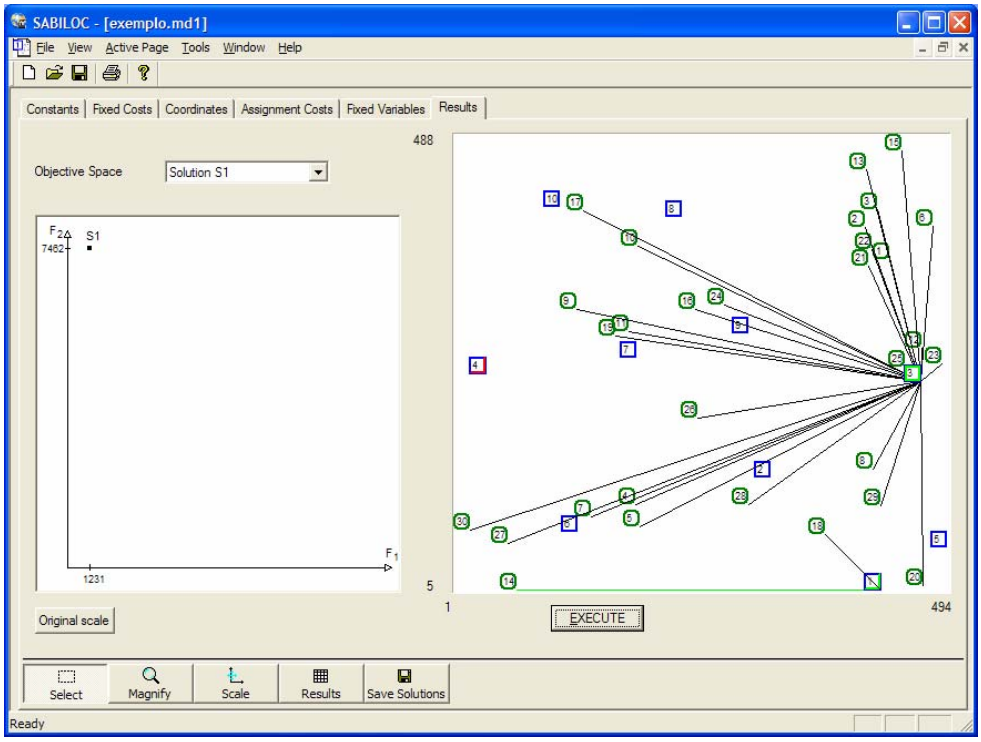

Figura 10 – Visualização no espaço dos objectivos da solução 1 que minimiza a função objectivo 1.

Na segunda iteração é apresentada a solução que minimiza a função objectivo 2, e que tem valor óptimo F_2^*= 4181, sendo o "melhor" valor obtido para função objectivo 1, F_1= 2034 (Figura 11). Como podemos ver, na segunda iteração, é também apresentada a solução ideal.

Mais uma vez, de forma a facilitar a visualização e compreensão da rede, foram definidas cores para caracterizar diversas situações. As cores dos serviços fixados na página "*Fixed Variables*" mantêm-se nesta página, ou seja, os serviços abertos são parcialmente verdes e os fechados parcialmente encarnados. As afectações activadas continuam a ser visualizadas a verde na rede desta página. Refira-se que, por motivos de simplificação da rede não são traçadas as afectações desactivadas. Após o cálculo de uma solução, os serviços abertos, excepto os que foram abertos por fixação, ficam quase totalmente verdes, mantendo um pouco da sua cor original. Note-se que podemos distinguir os serviços abertos por fixação e os abertos por força do cálculo de uma solução, apesar de possuírem as mesmas cores. As afectações obtidas pelo cálculo de uma solução, excepto as já fixas como activadas, são traçadas a preto.

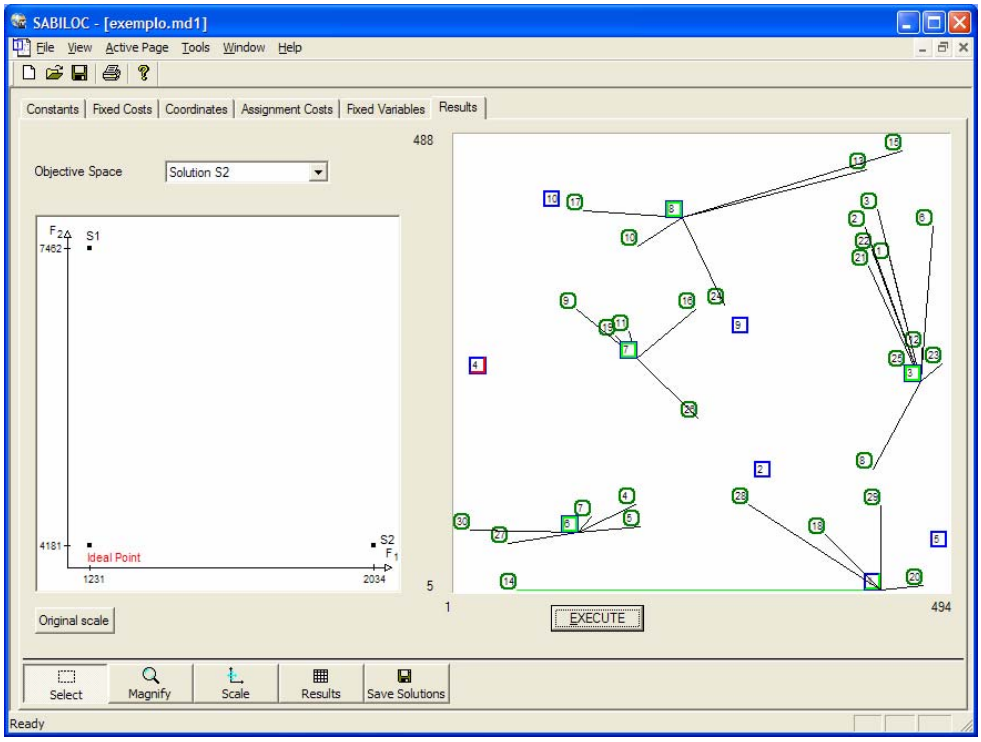

Figura 11 – Visualização no espaço dos objectivos da solução 2 que minimiza a função objectivo 2. Ponto ideal.

Analisemos, observando as Figuras 10 e 11, que enquanto a solução 1 corresponde à abertura de dois serviços no locais potenciais 1 e 3, a solução 2 corresponde à abertura de cinco serviços nos locais potenciais 1, 3, 6, 7 e 8. Se voltarmos às páginas "*Fixed Costs*" e "*Assignment Costs*", podemos ver que os valores da matriz de distâncias euclideanas usadas na função objectivo 2 são, de um modo geral, da mesma ordem de grandeza dos valores dos custos fixos para este mesmo objectivo, logo é compensador abrir vários serviços perdendo nos custos fixos, mas poupando nos custos de afectação. O mesmo já não se passa para a função objectivo 1 em que é preferível poupar nos custos fixos. Notemos também que as fixações foram respeitadas, isto é, o serviço 1 foi aberto ficando a comunidade 14 afecta a ele, e o serviço 4 ficou fechado tal como se pretendia.

O ícone "*Select*" da barra de ferramentas tem as mesmas funcionalidades em relação à rede que o da página "*Fixed Variables*". No entanto, nesta página, também permite que o utilizador insira os limites aceitáveis para ambos os objectivos, premindo o botão do rato na região do espaço dos objectivos onde pretende pesquisar soluções não dominadas. Esta opção só é válida após o cálculo dos mínimos lexicográficos e entre soluções não dominadas candidatas a adjacentes.

Os ícones "*Magnify*" e "*Scale*" têm como utilidade facilitar a visualização do espaço dos objectivos. A opção "*Magnify*" possibilita, caso o utilizador prima o botão esquerdo do rato sobre o espaço dos objectivos, obter uma ampliação da parte seleccionada, ou seja, funciona como uma lupa virtual. Premindo o botão direito do rato sobre a imagem, é possível definir

as características da lupa em termos de ampliação e tamanho. Com uma maior utilidade, temos a opção "*Scale*", que torna possível alterar a escala dos eixos dos objectivos (Figura 12). A opção de alterar a escala em alguns casos, é especialmente interessante a partir de um certo número de interacções, pois quando o número de soluções é considerável, a imagem torna-se confusa devido à proximidade das soluções. Por baixo do espaço dos objectivos, encontra-se o botão "*Original Scale*" que possibilita, tal como a sua denominação indica, voltar à escala original que é constituída pelos valores dos mínimos lexicográficos.

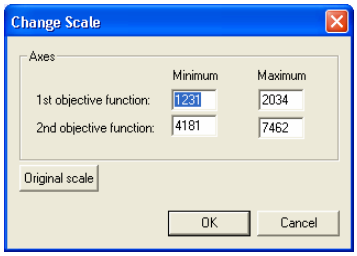

Figura 12 – Caixa de diálogo que permite alterar a escala dos eixos dos objectivos.

Os valores das variáveis x_{ij} e y_j do modelo podem ser vistos implicitamente a partir da imagem da rede ou de forma explícita através de grelhas (Figura 13). A opção é feita através do ícone "*Results*" da barra de ferramentas.

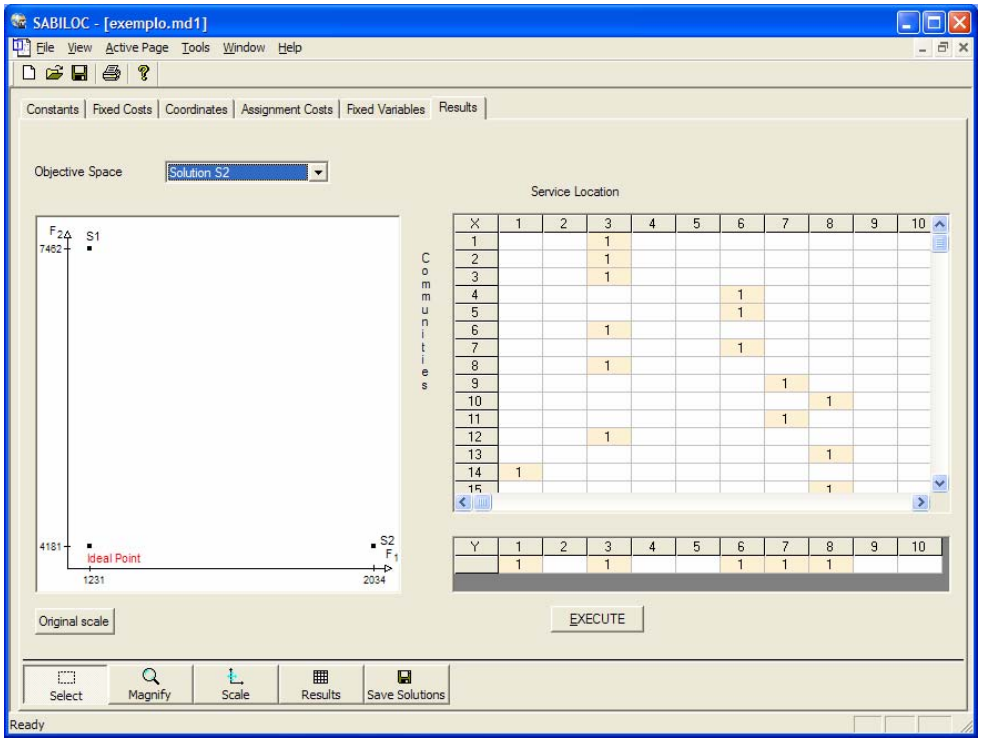

Figura 13 – Visualização da solução 2, em termos das variáveis, através de uma grelha de valores.

Supondo que queremos calcular outra solução eficiente, teremos de escolher novamente a opção "*Execute*". Para tal, uma possibilidade é premir o botão do rato nalgum ponto do rectângulo definido pelas soluções 1 e 2, que corresponde à região do espaço dos objectivos onde poderão existir soluções não dominadas. Deste modo, abre-se uma caixa de diálogo para definir os parâmetros da pesquisa que se pretende realizar. Vamos ser exigentes e tentar calcular uma solução não dominada próxima do ponto ideal. Para tal, premimos o botão do rato no espaço dos objectivos numa zona próxima do ponto ideal. Abre-se a caixa de diálogo da Figura 14 e aceita-se o que está definido por omissão: são dados os limites superiores escolhidos para as funções objectivo tal como se pretendia (ponto onde se premiu o botão do rato), o λ_1 é calculado automaticamente tendo em conta os limites superiores considerados e o algoritmo a utilizar na procura da solução é a adaptação do DUALOC (Dias *et al.*, 2003), com critério de paragem na ordem das 5 000 000 de iterações.

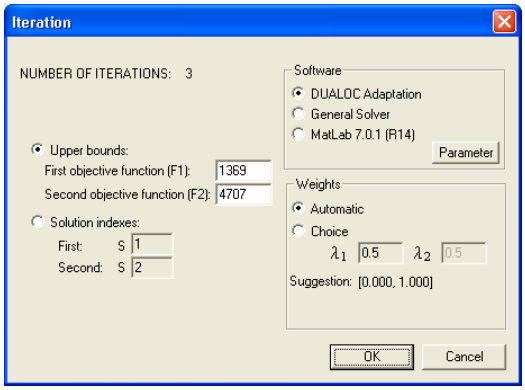

Figura 14 – Caixa de diálogo que permite a pesquisa de soluções.

Fomos demasiado exigentes com os limites para as funções objectivo e desta forma não existe qualquer solução não dominada na área escolhida (Figura 15).

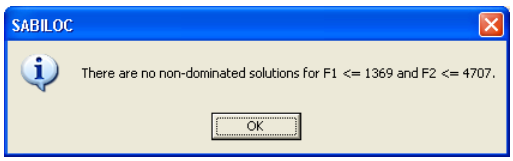

Figura 15 – Informação relativa à não existência de soluções não dominadas na área pesquisada.

Apesar de termos escolhido uma adaptação do DUALOC para a procura de uma solução não dominada, existem outras possibilidades que são o *general solver* de Programação Linear Inteira CPLEX e uma ferramenta do Matlab. O *general solver* não é fornecido juntamente com o SABILOC, logo a sua utilização só é possível se o computador possuir licença para tal. Por outro lado, a ferramenta do Matlab é sempre uma alternativa válida para a procura de soluções, desde que o utilizador instale no seu computador, previamente à utilização do SABILOC, um ficheiro executável que é fornecido – MCRInstaller. A obtenção de soluções não dominadas em cada iteração é notoriamente mais lenta, em especial na primeira iteração que se utiliza, quando comparada com as outras ferramentas. No entanto, é a única

alternativa à adaptação do DUALOC num computador que não possui licença de utilização do CPLEX.

Para o modelo de *p*-localização bicritério, são também dadas três hipóteses de escolha: novamente o *general solver* CPLEX, uma ferramenta do Matlab e um algoritmo baseado no anterior, adaptação do DUALOC, com as devidas alterações, de forma a que possa fixar-se o número de serviços que se pretendem abertos.

É oportuno aqui referir a possibilidade de alteração dos critérios de paragem dos algoritmos. Como podemos ver na Figura 14, existe um botão denominado "*Parameter*" na caixa de grupo *Software*. Dependendo do *software* escolhido para a obtenção de soluções não dominadas, assim é a caixa de diálogo que se abre para definir os parâmetros de paragem.
Uma outra forma de definir a região que se pretende pesquisar, para além dos limites superiores para os objectivos, é através de duas soluções não dominadas já determinadas e candidatas a serem adjacentes.

Com o intuito de facilitar a compreensão da instância bem como da aproximação interactiva bicritério, apresentamos ainda a visualização em ambos os espaços de outras soluções eficientes (Figuras 16 e 17).

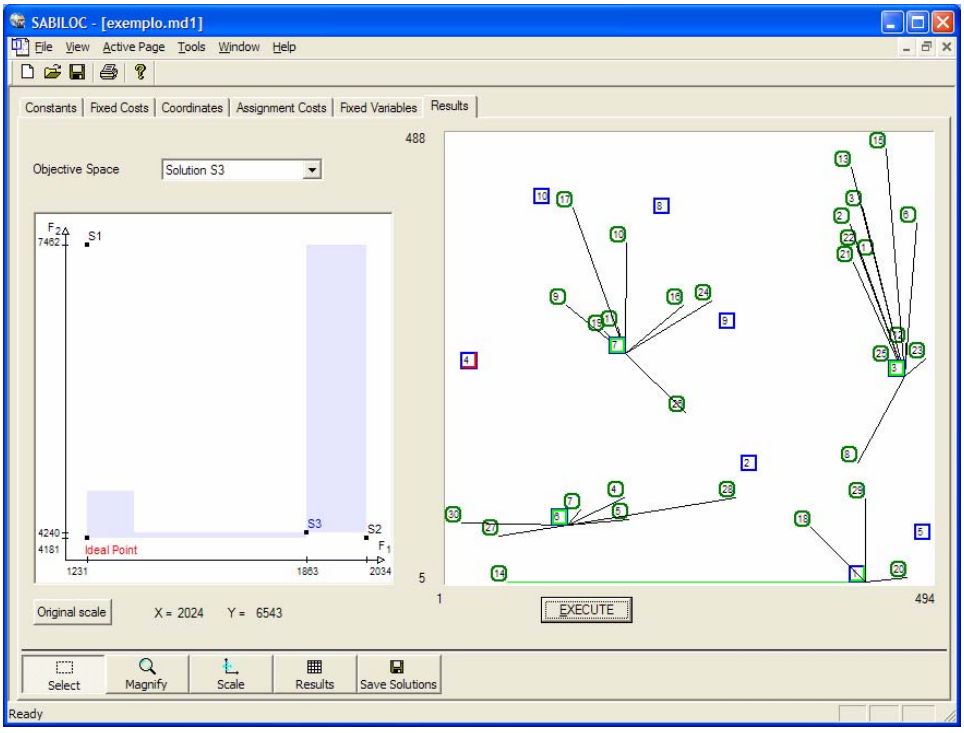

Figura 16 – Visualização no espaço dos objectivos das soluções 1, 2 e 3. Afectações da solução.

Por exemplo, a solução 3, obtida pesquisando a região entre as soluções não dominadas 1 e 2, em comparação com a 1, melhora em 3222 unidades de custo a função objectivo 2 e piora em 632 unidades de custo a função objectivo 1. Em comparação com a solução 2, melhora

em 171 unidades de custo a função objectivo 1 e piora em 59 unidades de custo a função objectivo 2.

Notemos que foi feita uma mudança de escala no espaço dos objectivos da Figura 17 com o intuito de facilitar a sua visualização, pois a imagem encontrava-se um pouco confusa devido à proximidade das soluções.

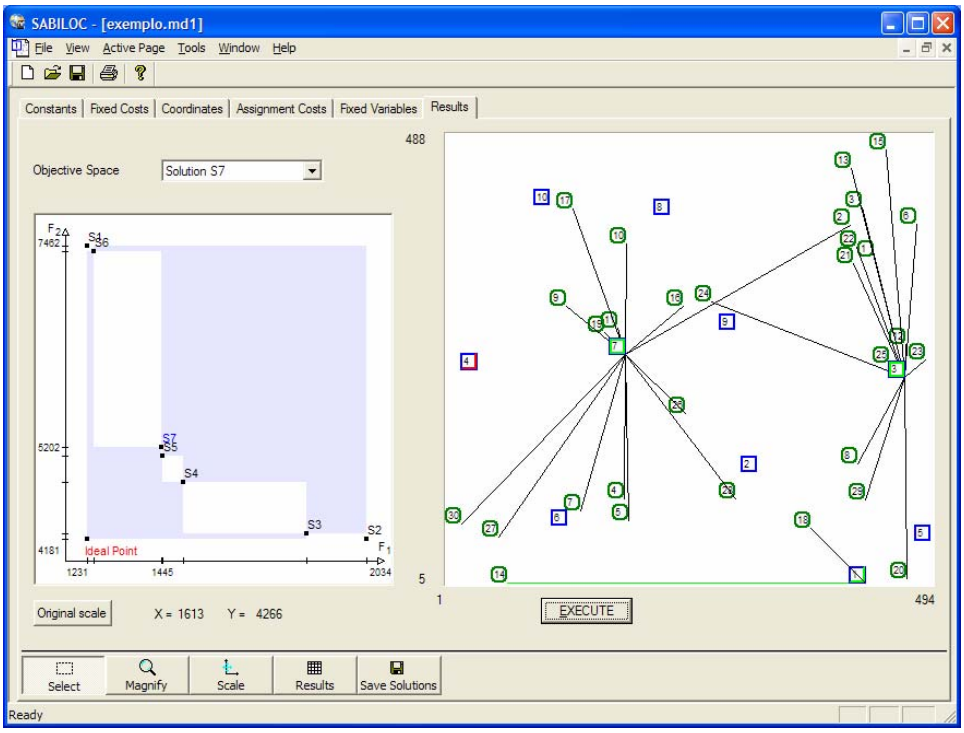

Figura 17 – Visualização no espaço dos objectivos (com mudança de escala) das soluções 1, 3, 4, 5, 6 e 7. Afectações da solução 7.

Refira-se que a caixa de listagem pendente que se encontra por cima da imagem do espaço dos objectivos permite visualizar novamente as soluções não dominadas já calculadas.

À medida que se vão encontrando soluções não dominadas, podemos ver no espaço dos objectivos, as áreas (a sombreado) onde não existem soluções eficientes quer por não admissibilidade, quer por dominância. Esta informação gráfica juntamente com a obtida, quando não são encontradas soluções numa certa região, auxilia o utilizador uma vez que, mostrando cumulativamente as regiões eliminadas, o processo de exploração é reduzido, evitando-se perda de tempo e de recursos correspondentes à exploração de áreas onde não existem, com certeza, outras soluções não dominadas.

O processo termina quando o utilizador considerar que possui informação suficiente sobre o conjunto das soluções não dominadas. Para esta instância, caso o utilizador tivesse continuado indefinidamente a pesquisa de soluções não dominadas teria encontrado 90. Apresenta-se de seguida a Figura 18, em que se podem ver todas as soluções não dominadas do problema.

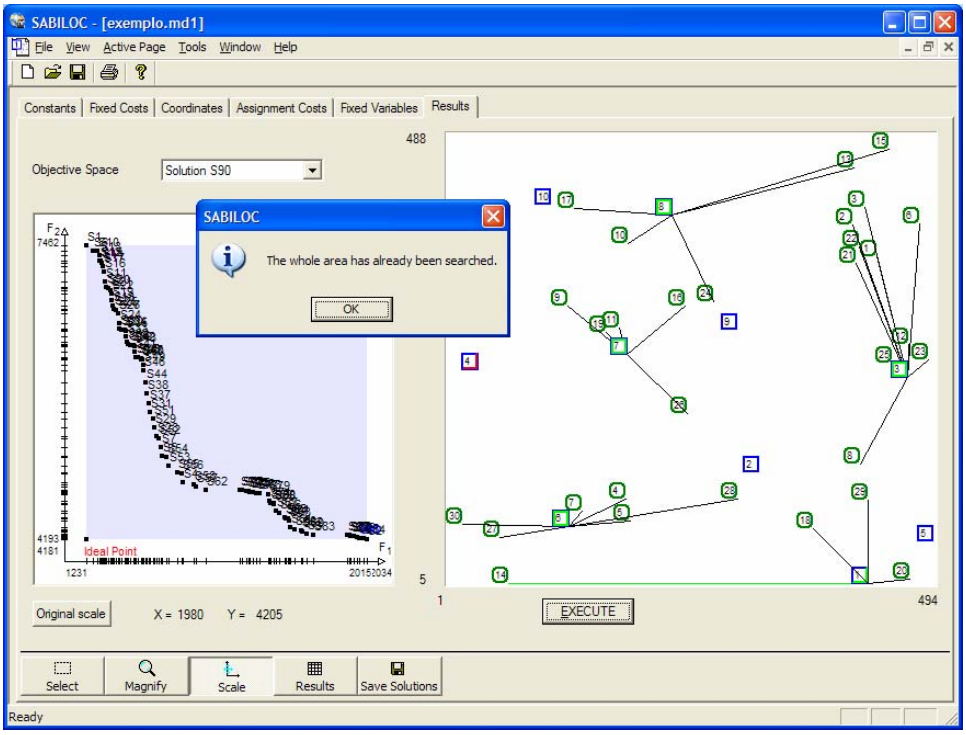

Figura 18 – Visualização no espaço dos objectivos de todas as soluções não dominadas. Afectações da solução 90.

Podem-se guardar num ficheiro de texto todas as soluções calculadas, accionando o botão correspondente que se encontra por baixo da imagem do espaço dos objectivos. Finalmente refira-se a possibilidade de impressão, bem como da sua pré-visualização, para todas as soluções determinadas. A informação a imprimir ou guardar para cada solução consiste nas afectações, nos serviços abertos e no valor de cada uma das funções objectivo. Para a solução que se encontra seleccionada na caixa de listagem pendente, é ainda impressa a imagem da rede.

5. Conclusões

Apesar de se encontrarem na literatura vários métodos interactivos com a possibilidade de aplicação a problemas de localização, não é usual existirem SAD que utilizem os métodos propostos. No entanto, é através dos sistemas que os utilizadores entram em contacto com os métodos interactivos. Obviamente, estes podem ser utilizados sem recurso a meios computacionais, mas a sua aplicabilidade é comprometida em muitas situações reais devido à dimensão dos problemas.

Surgiu então a necessidade de dispor de um sistema computacional que possibilite a utilização do método interactivo proposto, por forma a que o esforço computacional seja orientado para as regiões onde se localizam as soluções que melhor correspondem à estrutura de preferências do decisor. Foi com este objectivo que foi desenvolvido um sistema interactivo de apoio à decisão para problemas de localização com dois objectivos. O

SABILOC foi construído de forma modular, facilitando a inclusão de modelos e algoritmos que resolvam mais eficientemente os problemas bicritério. Os modelos que o sistema se propõe apoiar, neste momento, são o de localização simples bicritério e o de *p*-localização.

Em cada interacção do SABILOC, o decisor tem de inserir os limites aceitáveis para as funções objectivo, de forma directa, tendo em conta as áreas por explorar, ou então, implicitamente através de duas soluções não dominadas previamente calculadas e candidatas a serem adjacentes. Os valores para os pesos que servirão para ponderar as funções objectivo também têm de ser inseridos, havendo a possibilidade de se aceitar os que são propostos pelo sistema. O decisor terá ainda de escolher o algoritmo de resolução do problema auxiliar monocritério. Para o problema de localização simples bicritério é dada a possibilidade de escolha entre o *general solver* CPLEX, uma ferramenta do Matlab e um algoritmo desenvolvido por Dias, Captivo & Clímaco (2003). Para o problema de *p*-localização, o decisor poderá escolher entre o algoritmo anterior com as devidas alterações, de forma a possibilitar a utilização no modelo em questão e mais uma vez o *general solver* CPLEX e a ferramenta do Matlab.

Como em qualquer outro sistema computacional, há sempre melhorias a considerar, quer em termos da interface, quer em termos do código subjacente. Possivelmente, conseguem-se melhorias em termos de eficiência do sistema utilizando estruturas de dados mais adequadas. As estruturas de dados utilizadas num determinado programa são determinantes para o seu desempenho. Outra melhoria a considerar será fornecer ao utilizador mais meios de apoio relativamente à tomada de decisão, sempre com o intuito de apoiar e nunca de substituir o decisor, já que o objectivo não é convergir para uma solução de compromisso, mas apoiar o decisor na eliminação das soluções eficientes que se revelem sem interesse prático.

Com mais alguma programação, será possível fazer-se uma análise de pós-optimização relativamente às soluções não dominadas encontradas. Repare-se que pode ser uma ferramenta bastante útil de apoio ao decisor, caso este queira estudar uma solução que lhe pareça a mais conveniente para o fim em vista. Por exemplo, saber até que ponto poderão os custos fixos dos serviços abertos influenciar o facto de uma solução ser ou não ser dominada, ou até mesmo saber entre que valores poderão variar os custos de afectação entre um serviço aberto e as comunidades a ele afectas, de modo a que a solução se mantenha eficiente. Seria também interessante possibilitar uma análise de sensibilidade relativamente à introdução de variáveis adicionais, sem ter que recomeçar o problema. Por exemplo, saber se ao introduzir um novo potencial local de instalação de um serviço, uma determinada solução continua ou não a ser eficiente. Existe uma série de possibilidades em análise de sensibilidade que seria de todo o interesse oferecer ao utilizador de forma a apoiá-lo o mais possível nas suas decisões.

Figueira (1996) propõe que se dê ao utilizador a possibilidade de fazer uma análise *a posteriori* na vizinhança da solução preferida até ao momento, de forma a seleccionar a que realmente corresponde à "melhor" solução a ser implementada. Fazer uma análise *a posteriori* é analisar, com mais pormenor, soluções qualitativamente indiferentes, no que diz respeito aos modelos multicritério anteriormente utilizados. Consequentemente, algumas destas soluções podem ser "ligeiramente" dominadas no que respeita ao modelo matemático utilizado. É com facilidade que se consegue incorporar no SABILOC uma análise *a posteriori* em torno de uma solução não dominada.

O facto de o SABILOC ter sido desenvolvido de forma modular, permite a qualquer instante, e bastante rapidamente, a inclusão de novos modelos, bem como de algoritmos mais eficientes. Assim, será de todo o interesse o desenvolvimento e implementação de novos modelos mais realistas.

A expansão deste SAD, como de qualquer outro, para o campo dos modelos com mais que dois critérios, levanta alguns problemas no que diz respeito à apresentação da informação ao utilizador. A elaboração de uma interface gráfica simples e intuitiva deixa de ser, em grande parte, possível, devido à dificuldade em representar soluções não dominadas num espaço de objectivos com mais que duas dimensões.

Mas nem todas as atenções vão para a elaboração de novos modelos, pois o tempo que o decisor tem de esperar para obter uma resposta é extremamente importante para a sua aceitação. Assim, há também que desenvolver algoritmos eficientes de resolução, de forma a obter respostas em tempo útil. No caso dos modelos com mais que dois critérios, os tempos computacionais têm tendência a aumentar, pois o problema auxiliar a resolver em cada interacção apresenta mais restrições relativamente ao dos modelos bicritério.

Referências Bibliográficas

(1) Captivo, M.E. & Clímaco, J. (2001). Multicriteria Location Problems – Discussion on some models and algorithms. **In:** *AMCDA – Aide Multicritère à la Décision* [edited by A. Colorni, M. Parucini and B. Roy], Joint Research Centre, EUR Report, The European Commission, 2001, 47-58.

(2) Clímaco, J.N.; Antunes, C.H. & Alves, M.J. (2003). *Programação Linear Multiobjectivo*. Imprensa da Universidade, Coimbra.

(3) Current, J.; Min, H. & Schilling, D. (1990). Multiobjective Analysis of Facility Location Decisions. *European Journal of Operational Research*, **49**, 295-307.

(4) Dias, J.M.; Captivo, M.E. & Clímaco, J. (2003). An Interactive Procedure dedicated to a Bicriteria Plant Location Problem. *Computers & Operations Research*, **30**, 1977-2002.

(5) Erlenkotter, D. (1978). A Dual-Base Procedures for Uncapacitated Facility Location. *Operations Research*, **26**(6), 992-1009.

(6) Ferreira, C.; Clímaco, J. & Paixão, J. (1994). The Location-Covering Problem: a Bicriterion Interactive Approach. *Investigación Operativa*, **4**(2), 119-138.

(7) Figueira, J. (1996). L'Approche Interactive dans le cas des Problémes de flot Multicritéres. Thèse de Doctorat, Université Paris-Dauphine.

(8) Gardiner, L. & Vanderpooten, D. (1997). Interactive Multiple Criteria Procedures: Some Reflections. **In:** *Multicriteria Analysis* [edited by J. Clímaco], Springer, 290-301.

(9) Ross, G. & Soland, R. (1980). A Multicriteria Approach to the Location of Public Facilities. *European Journal of Operational Research*, **4**, 307-3321.

(10) Solanki, R. & Gorti, J. (1997). Implementation of an Integer Optimization Plataform Using Object Oriented Programming. *Computers & Operations Research*, **24**(6), 549-557.