Universidade de Lisboa
Faculdade de Ciências
Departamento de Estatística e Investigação Operacional

Heurísticas para localização de sensores

Diogo Filipe Martins Ferreira da Silva

Dissertação
Mestrado em Investigação Operacional
2013

Universidade de Lisboa
Faculdade de Ciências
Departamento de Estatística e Investigação Operacional

Heurísticas para localização de sensores

Diogo Filipe Martins Ferreira da Silva

Dissertação orientada por:

António José Rodrigues
Maria Eugénia Captivo

Mestrado em Investigação Operacional

2013

Resumo

Neste trabalho é desenvolvido o algoritmo de atração, uma heurística inspirada em algoritmos evolutivos, tendo como objetivo a resolução de problemas contínuos de localização e cobertura. Numa primeira parte o problema é abordado tendo em conta uma cobertura binária, tornando o problema mais simples e a explicação das ideias chave da heurística mais clara. Depois de compreendida a natureza do problema binário será introduzida a cobertura contínua, bem como a explicação do funcionamento da heurística. É apresentada a comparação de resultados entre esta heurística, um algoritmo genético e uma meta-heurística baseada em enxame de partículas. Finalmente são apresentadas conclusões acerca das vantagens e desvantagens de cada um dos métodos estudados, bem como possíveis melhoramentos futuros.

Abstract

In this work we developed the attraction algorithm, a heuristic inspired by evolutionary algorithms that solves sensor location problems in a continuous environment. First, the problem is addressed assuming sensor coverage is binary, making the problem easier and facilitating the explanation of the key ideas behind the heuristic. After understanding the nature of the binary problem, continuous coverage will be introduced, followed by a full explanation of the algorithm. A comparison is made between results gathered from this heuristic, a genetic algorithm and a metaheuristic based on particle swarms. Finally, conclusions are presented about the advantages and disadvantages of each method studied, as well as possible future improvements.

Agradecimentos

Quero agradecer aos meus orientadores pela paciência e dedicação,
à minha família e amigos pelo apoio incondicional,
particularmente ao meu irmão, o meu oráculo de LaTeX
e finalmente à minha namorada pela inspiração e pelos bolinhos.

Não teria conseguido sem a vossa ajuda.
Obrigado.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

Capítulo 1

Introdução

Desde a escolha do local onde pousar uma caneta até decisões de grande escala como onde construir uma nova rede de escolas, o que varia é a complexidade do problema e as ferramentas que cada um tem ao seu dispor para o resolver.

Hoje em dia, na nossa sociedade, a solução de um problema de localização pode ser responsável pelo investimento de muito dinheiro. Sabendo isto, não é de estranhar que já tenham sido desenvolvidos algoritmos que resolvem alguns problemas desta natureza, dando como solução uma localização otimal. Mas o uso de algoritmos na resolução de problemas complexos suscita sempre uma pergunta. Quanto tempo demora a sua execução?

Infelizmente, na grande maioria dos casos, a relação entre a complexidade do problema e o tempo de execução do programa não é linear. Por outras palavras, demoraria demasiado tempo a obter a solução ótima de certos problemas de complexidade média. Uma forma de contornar esta questão é fazer uma troca: diminuir o tempo de execução sacrificando a qualidade da solução obtida. Foi com base nesta ideia que surgiram as primeiras heurísticas, algoritmos que rapidamente obtêm uma solução, mas que não garantem que esta seja ótima.

Esta forma de abordar a resolução dos problemas, tempo versus qualidade, fez com que alguns investigadores procurassem novas fontes de inspiração para os seus algoritmos. Como tal, não demorou muito até que aparecessem algoritmos inspirados em comportamentos animais e processos biológicos. Neste trabalho é apresentada uma heurística inspirada nos raciocínios básicos de um ser humano, quando deparado

com um problema de localização de sensores.

Independentemente da zona de interesse real que se tenciona cobrir com os sensores, no contexto do problema esta será reduzida ao quadrado unitário $[0, 1] \times [0, 1]$. A quantidade de sensores a usar é fixa à partida, bem como as suas probabilidades de deteção em função da distância. O objetivo é maximizar a probabilidade de deteção na zona de interesse, através da colocação dos sensores no seu interior.

Na realidade a probabilidade de deteção de um sensor depende de muitos mais fatores, como as condições ambientais, linha de visão, o tipo de ameaça que se está a tentar detetar, etc... Mas uma vez que neste trabalho a probabilidade de deteção depende apenas da distância entre a ameaça e o sensor, cada um deles terá uma área de deteção radialmente simétrica. A ameaça em si depende fortemente do contexto real do problema, podendo ser um incêndio florestal (Baby Vennila et al., 2012), poluição (Santini et al., 2008), ataque terrorista (Liu et al., 2011), etc...

Numa primeira fase deste trabalho, são explicados os conceitos do problema e apresentada a sua formulação. De seguida é feita uma pequena introdução aos algoritmos estudados, explicando as suas origens e funcionamento. Otimizam-se os parâmetros de cada um dos algoritmos, sendo os valores obtidos usados em testes comparativos dos seus desempenhos. São apresentados os resultados obtidos e são apresentadas as conclusões, bem como possíveis melhoramentos futuros.

Foi também criado um interface gráfico (Ver anexo) de forma a facilitar a interação com os programas desenvolvidos que permite:

- Gerar instâncias aleatórias às quais aplicar os algoritmos
- Alterar o número e tipo de sensores disponíveis
- Alterar os valores dos parâmetros de cada algoritmo
- Alterar os valores dos critérios de paragem
- Visualizar as soluções obtidas e o risco associado

Capítulo 2

Descrição do problema

2.1 Problema binário

Existem inúmeras variações de problemas de cobertura (Akyildiz et al., 2002; Cardei e Wu, 2004; Wang et al., 2008). Começaremos por abordar uma versão simplificada do problema, em que a cobertura dos sensores é binária.

A zona de interesse é o interior de um quadrado de lado um e define a área a ser coberta pelos sensores. Todos os sensores têm uma zona de cobertura circular, sendo definidos apenas pelo seu raio de ação. O objetivo é colocar os sensores de forma a maximizar a área da zona de interesse coberta por pelo menos um sensor (Figura 2.1).

Uma vez que a zona de interesse é quadrada e as zonas de cobertura dos sensores são círculos, pode ser complicado obter coberturas elevadas sem sobreposição das mesmas. Apesar de não existir nenhuma restrição no problema acerca de sobreposições, uma vez que a cobertura é binária, não há vantagem em ter dois ou mais

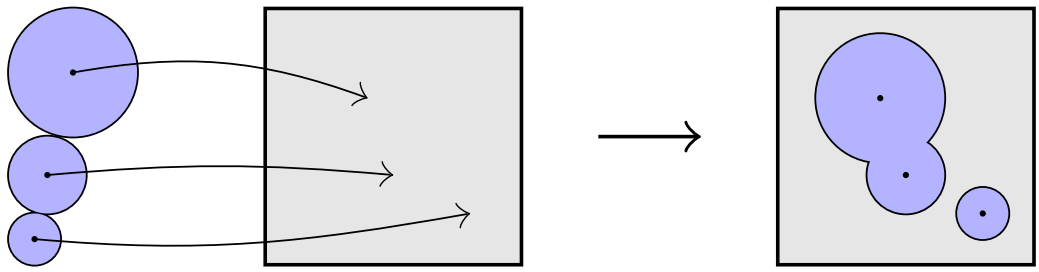


Figura 2.1: Colocação de sensores na zona de interesse

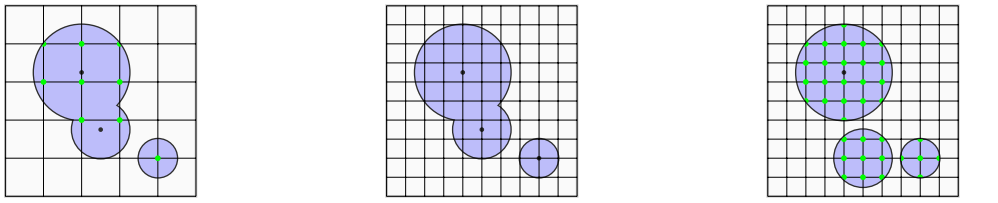


Figura 2.2: Cálculo da área coberta

sensores a cobrir a mesma zona.

Para se poder calcular uma aproximação da área coberta por uma solução, sobrepõe-se uma grelha de pontos à zona de interesse. De seguida, contam-se os pontos que estão cobertos por pelo menos um sensor e divide-se pelo número total de pontos. Multiplicando o resultado por 100 temos uma aproximação à percentagem de área coberta:

$$\frac{\text{número de pontos cobertos}}{\text{número total de pontos da grelha}} \times 100$$

Quanto maior a resolução de pontos na grelha, melhor será a aproximação obtida (Figura 2.2).

2.2 Problema probabilístico

Na vida real um sensor não é perfeito, não podendo garantir a deteção de um corpo apenas porque este invadiu o seu raio de ação. Existem inúmeras variáveis que condicionam a probabilidade de deteção de um sensor. Temperatura, luminosidade, humidade, ruído ambiente, etc... tudo isto pode afetar o desempenho de um sensor (Figura 2.3).

Claro que, ao contrário do que se passava com a cobertura binária, com cobertura probabilística existem vantagens em haver sobreposição de duas ou mais áreas de sensores. Suponhamos que temos dois sensores, s_1 e s_2, cuja probabilidade de deteção num dado ponto do espaço é de p_1 e p_2 respetivamente. O facto de um sensor detetar ou não o corpo não influencia a probabilidade de deteção do outro sensor, logo são acontecimentos independentes. Temos então que a probabilidade

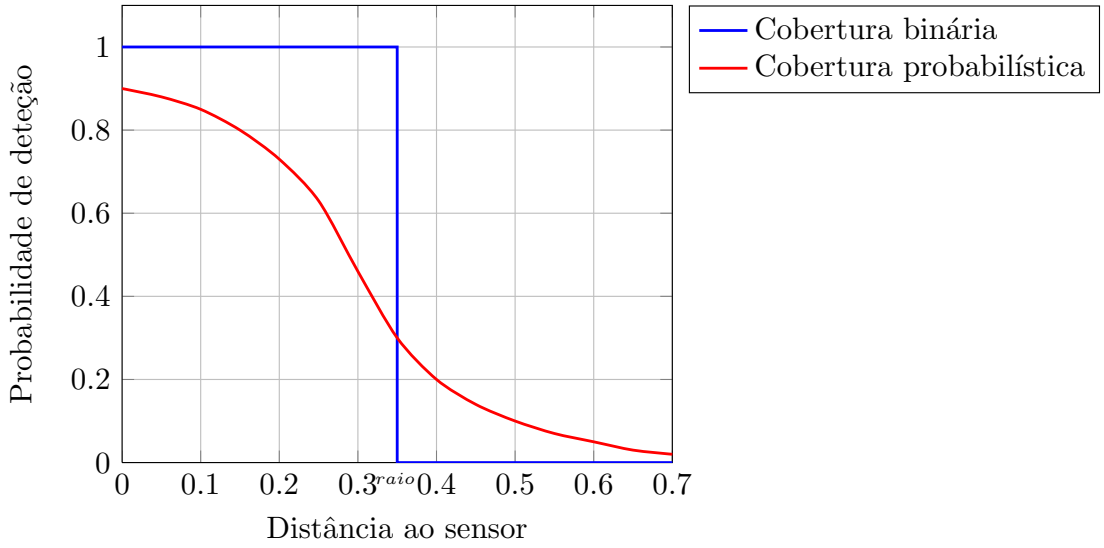


Figura 2.3: Comparação entre os dois tipos de cobertura

de ambos os sensores falharem na deteção do corpo é igual a $(1 - p_1) \times (1 - p_2)$. O que significa que a probabilidade de deteção por pelo menos 1 de n sensores, de um objeto num dado ponto do espaço x é $(1 - \prod_{i=1}^{n}(1 - p_i(x)))$ (Figura 2.4).

Quando os sensores são perfeitos e a sua cobertura binária, faz sentido que o objectivo seja maximizar a cobertura da zona de interesse. Mas quando os sensores são falíveis, cada ponto da zona de interesse deixa de ter dois estados, coberto ou descoberto, passando apenas a ter associada uma probabilidade de deteção. Isto significa que é necessário redefinir a forma como uma solução é avaliada.

2.3 Risco

Não existe uma definição de risco universalmente aceite, pois pode ter diferentes significados conforme o contexto do problema em que está inserido. Infelizmente, mesmo dentro do mesmo contexto muitas vezes é difícil chegar a um consenso, pois existem diferentes ideias sobre que fatores devem ser incluídos na sua definição, bem como na forma como estes devem ser medidos.

No contexto da defesa e segurança, o conceito de risco (de segurança) já teve várias definições ao longo dos anos (Wisner et al., 2003), mas presentemente uma das mais usadas (Willis et al., 2005; Willis, 2006) é da forma:

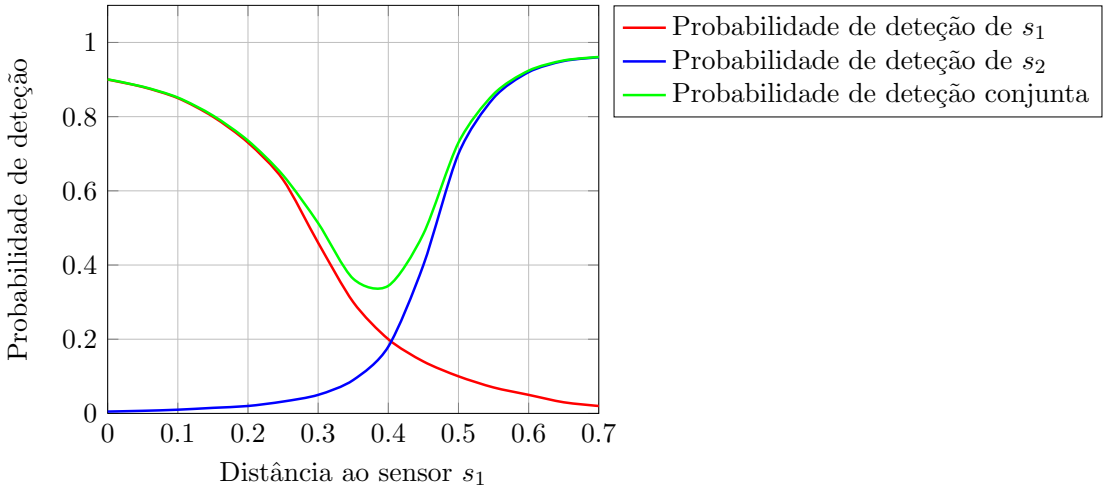


Figura 2.4: Probabilidade de deteção conjunta de dois sensores

$$\text{Risco} = \text{Ameaça} \times \text{Vulnerabilidade} \times \text{Consequência}$$

em que:

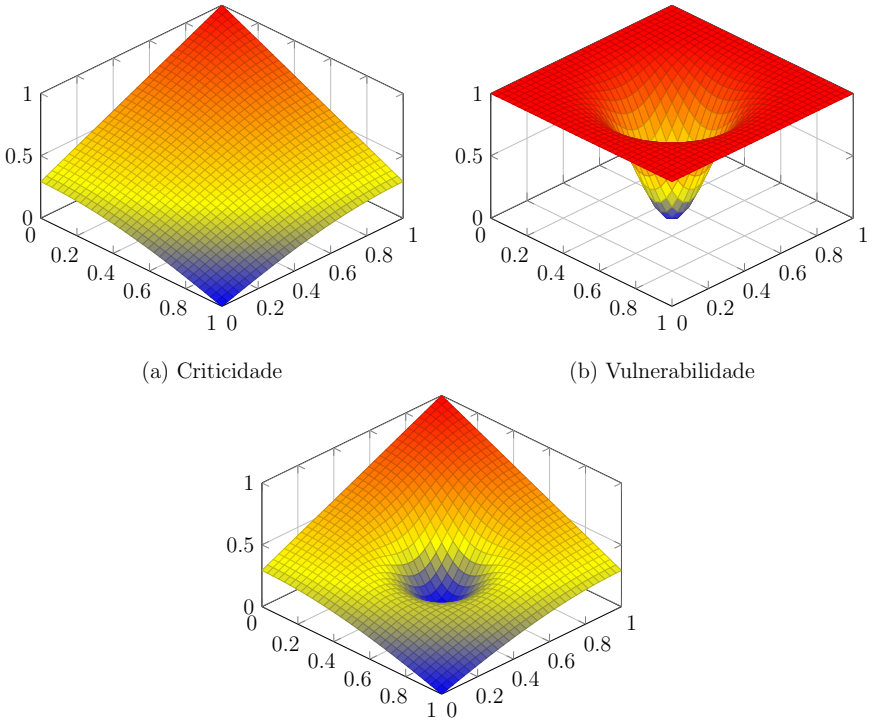
- Ameaça → Probabilidade de um ataque ocorrer.
- Vulnerabilidade → Probabilidade de um ataque causar danos, caso este ocorra.
- Consequência → Danos causados por um ataque (mortos, feridos, perdas económicas, danos morais, etc).

Um vez que no contexto deste trabalho não existem vários tipos de ameaça nem probabilidades associadas à ocorrência de cada uma delas, a definição de risco que vai ser usada é bastante mais simples:

$$\text{Risco} = \text{Criticidade} \times \text{Vulnerabilidade}$$

Ou seja, vai ser necessário calcular o risco existente em cada ponto da grelha de avaliação, mas para tal é necessário definir os conceitos de criticidade e vulnerabilidade.

Cada ponto terá três atributos associados: criticidade, vulnerabilidade e risco, todos com valores entre 0 e 1. O primeiro atributo simboliza a gravidade da presença de uma ameaça nesse mesmo ponto e está relacionado com a proximidade da ameaça a infraestruturas críticas. O segundo atributo é a probabilidade de uma ameaça



(a) Criticidade

(b) Vulnerabilidade

(c) Risco

Figura 2.5: Exemplo de criticidade, vulnerabilidade e risco resultante

passar por esse mesmo ponto sem ser detetada e simboliza a existência de uma outra rede de sensores já em uso, tendo um significado diferente do conceito de vulnerabilidade na primeira expressão de risco enunciada. O risco existente em cada ponto da grelha será calculado multiplicando os dois atributos anteriores a ele associados (Figura 2.5).

Mas como é o risco afetado pela localização dos sensores? A posição de um sensor diminui a vulnerabilidade associada aos pontos que estão ao seu alcance, o que por sua vez faz diminuir o risco. É apenas natural que o novo objectivo do problema seja minimizar o risco.

Assim, adaptando a fórmula usada no problema de cobertura binária, a qualidade de uma solução é calculada da seguinte forma:

$$\frac{\sum \text{Risco residual}}{\sum \text{Risco inicial}} \times 100$$

Em que o risco residual representa o risco presente nos pontos da grelha após a colocação dos sensores.

2.4 Formulação

Seja:

- $n \rightarrow$ Número de sensores a usar.
- $x_s \rightarrow$ Abcissa do local onde o sensor s está posicionado.
- $y_s \rightarrow$ Ordenada do local onde o sensor s está posicionado.
- $f_s(d) \rightarrow$ Função que indica a probabilidade de deteção do sensor s num ponto que se encontre a uma distância d.
- $r \rightarrow$ Resolução da grelha de pontos ($r \times r$) a sobrepor à zona de interesse.
- $C_{r \times r} \rightarrow$ Matriz imutável que em cada posição contém a criticidade do ponto da grelha correspondente.
- $V_{r \times r} \rightarrow$ Matriz imutável que em cada posição contém a vulnerabilidade do ponto da grelha correspondente.
- $D_{r \times r} \rightarrow$ Matriz que em cada posição contém a probabilidade de deteção conjunta dos sensores no ponto da grelha correspondente.

Função objetivo:

- $min \sum_{i=1}^{r} \sum_{j=1}^{r} C_{ij} \times V_{ij} \times (1 - D_{ij})$

Sujeito a:

- $0 \leq x_s \leq 1, \forall s \in 1..n$
- $0 \leq y_s \leq 1, \forall s \in 1..n$
- $D_{ij} = 1 - \prod_{s=1}^{n} [1 - f_s(\sqrt{(x_s - i)^2 + (y_s - j)^2})]$

Capítulo 3

Enquadramento

Vários algoritmos têm como inspiração comportamentos de seres vivos relativamente simples que, embora individualmente não demonstrem grande inteligência, em grupo aparentam ter uma inteligência superior. Isto deve-se a mecanismos de entreajuda e comunicação, que tornam certos comportamentos possíveis apenas quando inseridos num grupo.

Como exemplo, consideremos um formigueiro. Certas formigas têm como principal tarefa encontrar comida e trazê-la de volta para o formigueiro, sendo o comportamento de cada formiga bastante simples. Procurar comida dando preferência a caminhos marcados por feromonas e libertar feromonas se estiver a transportar comida de volta para o formigueiro. Desta forma, cada formiga que encontre comida estará, durante o seu retorno, a tornar o seu percurso mais aliciante a outras formigas.

Sem este mecanismo cada formiga não teria forma de comunicar às restantes se encontrou comida, não tirando assim partido do facto de pertencer a um grupo. É por este motivo que, embora inicialmente lhes seja desconhecida a localização de comida, rapidamente se formam carreiros de formigas. Por outras palavras, mesmo partindo duma solução inicial aleatória, acabam por convergir para uma solução melhor.

	s_1	s_2	s_3	s_4	s_5
	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
Abcissas $\rightarrow$	x_1	x_2	x_3	x_4	x_5
Ordenadas $\rightarrow$	y_1	y_2	y_3	y_4	y_5

Figura 3.1: Estrutura de uma solução com cinco sensores

3.1 Algoritmo genético

Inspirado em princípios genéticos de seleção natural, o algoritmo genético foi desenvolvido em 1960 por John Holland e os seus alunos da Universidade de Michigan (Gaspar-Cunha et al., 2012; Holland, 1962).

Este algoritmo tem como base a seguinte ideia. Indivíduos mais bem adaptados ao seu meio ambiente têm uma maior probabilidade de se reproduzir, tornando mais provável a propagação das suas características genéticas para gerações futuras. Esta propagação não se dá tal que a descendência é uma cópia dos progenitores, mas sim uma mistura de ambos.

Voltando ao problema de cobertura, se encararmos cada geração como uma iteração e cada indivíduo como uma solução do problema, temos que, soluções com melhor cobertura têm uma maior probabilidade de passar as suas características para soluções de iterações seguintes. Existem inúmeras formas de combinar a informação existente em duas soluções de forma a originar duas soluções descendentes. Suponhamos que uma solução do problema de cobertura com cinco sensores tem a estrutura mostrada na Figura 3.1.

Uma das formas mais simples de fazer o cruzamento de duas soluções consiste em definir um ponto de corte nas soluções progenitoras, atribuindo a uma solução descendente a primeira parte de um progenitor e a segunda parte do outro. A segunda solução descendente fica com as partes restantes de ambas as soluções progenitoras (Figura 3.2).

Na natureza é raro ocorrerem erros durante este tipo de processos genéticos, mas quando ocorrem, podem causar mutações na descendência. Uma mutação pode ser

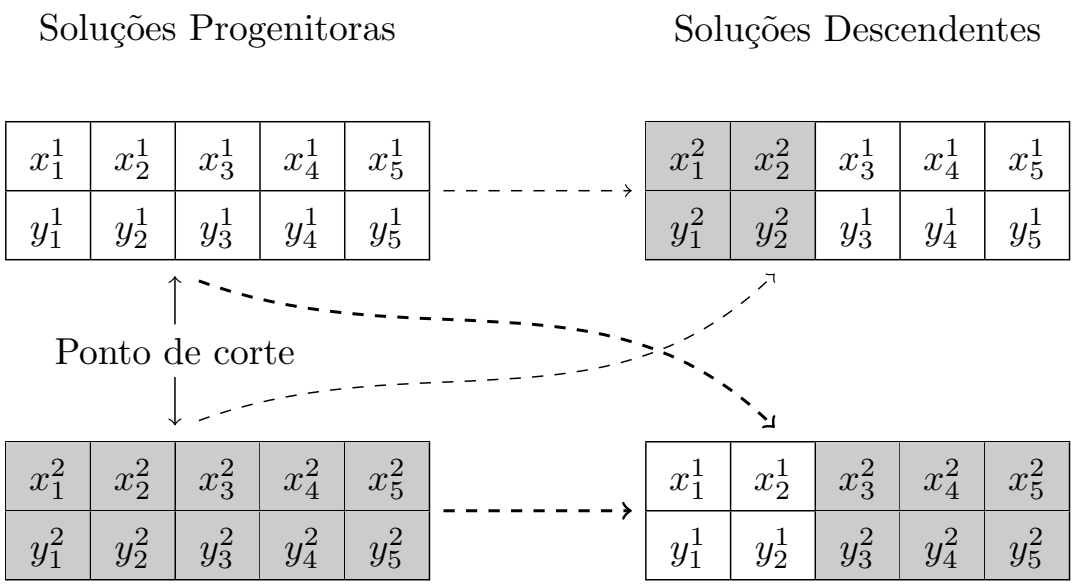


Figura 3.2: Exemplo de um tipo de cruzamento

Algoritmo 1: Cruzamento básico de duas soluções, $S1$ e $S2$

input : Soluções $S1$ e $S2$

$p \leftarrow$ ponto de corte

para *cada sensor* **faça**

  se $s <= p$ **então**

    $D1[s] \leftarrow S1[s]$

    $D2[s] \leftarrow S2[s]$

  senão

    $D1[s] \leftarrow S2[s]$

    $D2[s] \leftarrow S1[s]$

output: Soluções $D1$ e $D2$

praticamente impercetível e não ter qualquer impacto num indivíduo ou ter uma enorme influência na sua adaptabilidade ao seu meio ambiente. Mutações são uma parte importante do processo evolutivo, pois por vezes permitem o aparecimento de novas características que muito dificilmente teriam aparecido através dos métodos normais de cruzamento.

Se um indivíduo possuir uma mutação desfavorável à sua sobrevivência, este passa a ter uma menor probabilidade de reprodução, tornando menos provável a propagação dessa mutação para gerações futuras. Se por outro lado a mutação lhe conferir alguma vantagem, tornando-o mais bem adaptado ao seu meio ambiente, este terá uma maior probabilidade de reprodução e de propagar a sua mutação para gerações futuras.

Não é de espantar que, tendo sido inspirado por estes processos, também o algoritmo genético tenha um mecanismo que simule o acontecimento destas mutações. Sempre que é gerada nova descendência, há a probabilidade de alguma das suas características ser alterada. Ou seja, voltando ao problema de cobertura binária com cinco sensores, sempre que se dá um cruzamento e uma nova solução é gerada, há a probabilidade de um ou mais sensores terem as suas coordenadas modificadas. A probabilidade de mutação deve ser escolhida com cuidado, pois se for demasiado alta pode não permitir que as soluções evoluam naturalmente, derrotando a ideia base do algoritmo.

Claro que, tal como com os cruzamentos, existem inúmeras outras formas de definir o operador mutação num algoritmo genético, esta é apenas uma delas (Figura 3.3). Quanto à probabilidade de ocorrer uma mutação, tanto pode ser um parâmetro imutável definido no início, como ir variando ao longo das iterações de acordo com alguma fórmula.

É necessário definir mais três parâmetros, o número de soluções geradas por cruzamento em cada iteração ($ncruz$), quantas soluções aleatórias são acrescentadas à população em cada iteração ($nrand$) e se a melhor solução encontrada até ao momento faz sempre parte da população ($elitismo$). Estes três fatores devem ser definidos de forma a que a dimensão da população seja sempre a mesma no inicio de cada iteração.

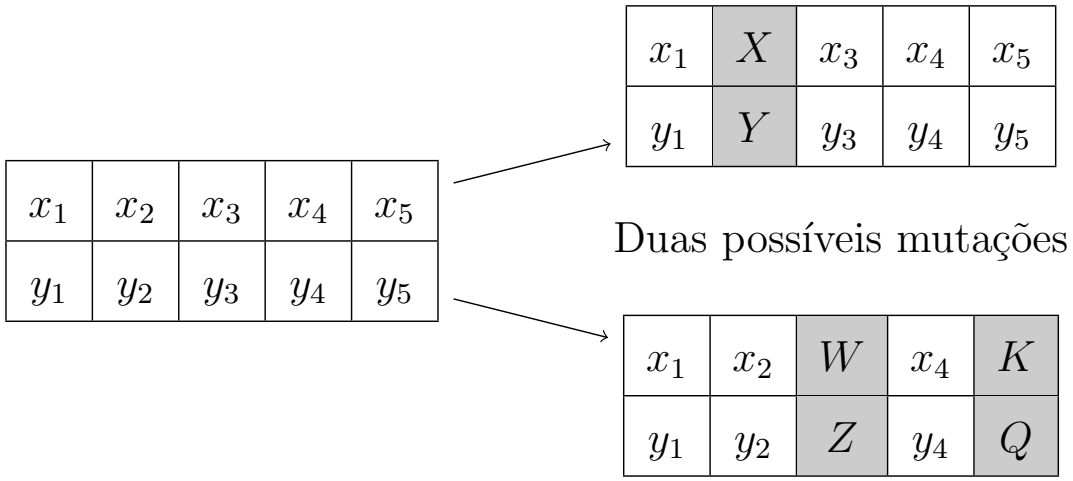


Figura 3.3: Exemplo de uma mutação

Algoritmo 2: Mutação dos sensores de uma solução

input : $SOL \leftarrow$ Uma solução do problema

para *cada sensor* **faça**

  $r \leftarrow$ valor aleatório entre 0 e 1

  se $r <=$*probabilidade de mutação* **então**

    $SOL[s] \leftarrow$ Novas coordenadas do sensor s

output: SOL

No algoritmo genético utilizado neste trabalho utiliza-se sempre o elitismo, sendo uma das soluções aleatórias substituída pela melhor solução encontrada até ao momento. Os outros dois parâmetros são definidos da seguinte forma:

- $ncruz \leftarrow 75\%$ da dimensão da população ($\frac{3}{4} \times npop$)
- $nrand \leftarrow 25\%$ da dimensão da população ($\frac{1}{4} \times npop$)

Claro que, para que isto funcione, *npop* tem de ser divisível por 4 e *ncruz* divisível por 2 de forma a tornar possíveis os cruzamentos exemplificados. Isto implica que *npop* seja multiplo de 8.

Assim, em cada iteração tem-se uma população de dimensão *npop* pois:

$$npop = ncruz + nrand$$

Um pseudo-código detalhado do algoritmo genético usado encontra-se em anexo.

Algoritmo 3: Algoritmo genético genérico

input : Parâmetros do algoritmo

$POP \leftarrow$ População de soluções iniciais

$MELHOR \leftarrow$ Melhor solução encontrada até ao momento

enquanto *Critério de paragem não for atingido* **faça**

- $CRUZ \leftarrow$ Soluções geradas por cruzamento das soluções em POP
- $RAND \leftarrow$ Soluções aleatórias adicionais (opcional)
- $RAND \leftarrow$ Elitismo, substituição de uma das solução aleatórias pela melhor encontrada até ao momento (opcional)
- $POP \leftarrow CRUZ \cup RAND$
- $MELHOR \leftarrow$ Melhor solução até ao momento

output: $MELHOR$

3.2 Enxame de partículas

Uma outra meta-heurística bem mais recente, chamada enxame de partículas, foi inventada por Kennedy e Eberhart (1995), tendo como inspiração as interações e comunicações entre elementos de um bando de aves ou cardume de peixes.

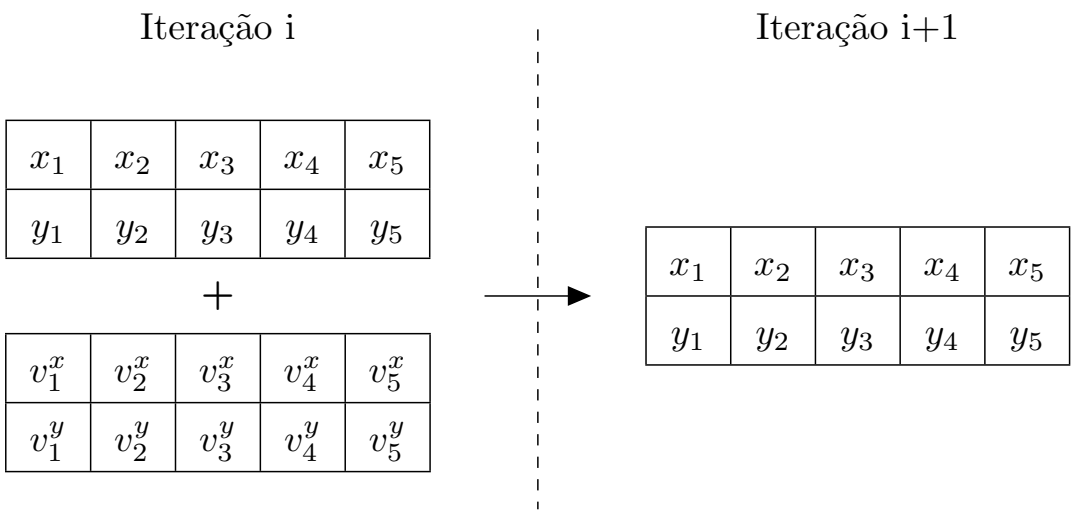


Figura 3.4: Evolução da posição de uma partícula

Imaginemos uma ave a voar numa dada direção e com uma dada velocidade. Se esta estiver sozinha, pode alterar a sua rota e movimentar-se sem perigo de chocar com outra ave. Mas se por outro lado, esta pertencer a um bando em voo, o mesmo já não é verdade. Por esta razão, os movimentos de cada um dos elementos de um bando não dependem só de si, mas também dos movimentos dos restantes elementos. Esta forma de interdependência é tão simples como cada elemento tentar manter a distância aos elementos do bando mais próximos.

Claro que existem coisas mais importantes do que manter a formação, uma delas é continuar vivo. O que acontece então, quando um fator externo ao grupo, como por exemplo um predador, obriga um ou mais elementos do bando a fazer uma mudança brusca na direção? Neste caso, os elementos que vêm a sua vida ameaçada dão um maior peso à sua decisão individual de evitar o perigo, passando para segundo plano tudo o resto. Mas como serve este comportamento de inspiração para um algorimo de enxame de partículas?

Suponhamos que cada elemento do bando é uma solução do nosso problema de cobertura binária, sendo a sua posição definida no espaço de busca pela matriz com as coordenadas dos sensores. Suponhamos agora que a velocidade de cada elemento é uma matriz de igual dimensão, que quando somada à sua matriz posição, origina a posição da iteração seguinte (Figura 3.4).

Claro que isto significa que na primeira iteração é necessário atribuir uma matriz

posição e uma matriz velocidade a cada partícula, o que pode ser feito de forma aleatória desde que se respeitem os limites do espaço de busca. Mas o ponto principal deste tipo de algoritmos está na forma como a matriz velocidade evolui ao longo das iterações.

Em cada iteração, a matriz velocidade de uma partícula é calculada através de uma combinação linear convexa de três componentes.

$$V_p^i = \alpha V_p^{i-1} + r\beta(S_p - P_p^i) + r\gamma(S - P_p^i)$$

em que:

- $V_p^i \rightarrow$ Matriz velocidade da partícula p na iteração i.
- $P_p^i \rightarrow$ Matriz posição da partícula p na iteração i.
- $S_p \rightarrow$ Matriz posição da melhor solução encontrada pela partícula p.
- $S \rightarrow$ Matriz posição da melhor solução encontrada por todas as partículas.
- $\alpha, \beta, \gamma \rightarrow$ Pesos associados a cada componente, em que $\alpha + \beta + \gamma = 1$
- $r \rightarrow$ Componente aleatória compreendida entre 0 e 1.

A componente $(S_p - P_p^i)$ é um vetor no espaço de busca que tem início na posição atual da partícula p e fim na posição da melhor solução encontrada pela partícula p. De forma análoga, a componente $(S - P_p^i)$ é um vetor no espaço de busca que tem início na posição atual da partícula p e fim na posição da melhor solução encontrada por todas as partículas.

Assim, torna-se mais claro que o papel das componentes α, β e γ é regular a importância dada a três ações: manter a velocidade original, ir na direção da melhor solução encontrada pela própria partícula e ir na direção da melhor solução global. O peso atribuído a cada uma destas componentes determina a forma como se pretende abordar o problema, por exemplo: A atribuição de uma maior importância à velocidade original de cada partícula (α) favorece a exploração do espaço de busca (*exploration*), por outro lado, a atribuição de um maior peso à melhor solução global (γ) favorece a busca local (*exploitation*) em torno da melhor solução encontrada.

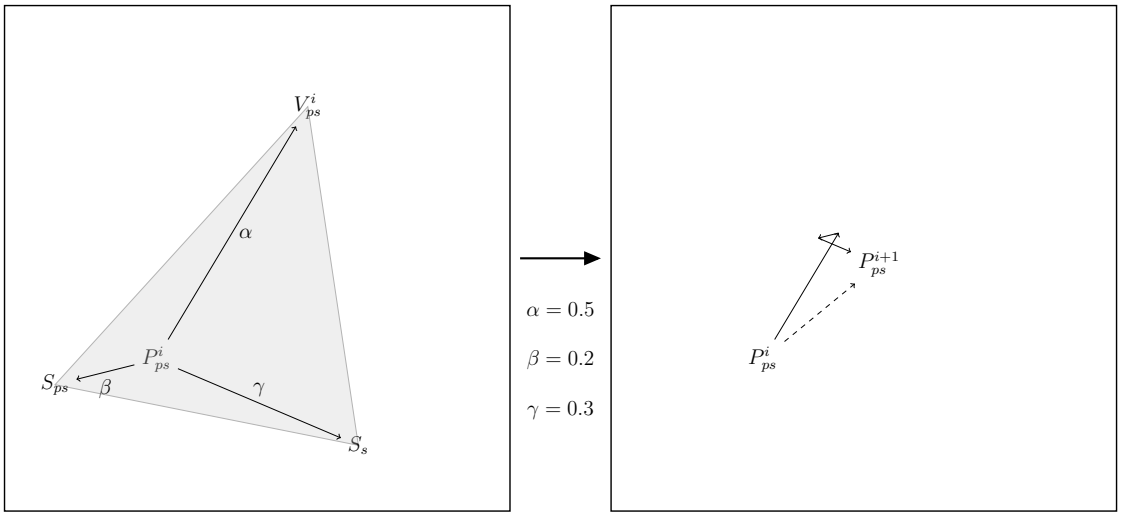


Figura 3.5: Influência de α, β e γ na deslocação de um sensor

Consideremos agora apenas o sensor s pertencente à partícula p de um problema de cobertura binário e vejamos que influência têm estas componentes na sua posição (Figura 3.5). A área a cinzento resulta do facto de se estar a fazer uma combinação linear convexa das três componentes ($\alpha, \beta, \gamma > 0 \wedge \alpha + \beta + \gamma = 1$) e representa todas as possíveis posições que a partícula pode assumir na próxima iteração com base nos três vetores V_{ps}^{i}, S_{ps} e S_s, sendo a sua posição determinada pelos pesos associados a cada uma das três componentes. Desta forma tem-se a garantia de que as coordenadas escolhidas para o sensor em cada iteração se encontram dentro da zona de interesse.

No exemplo, os pesos $\alpha = 0.2$, $\beta = 0.3$ e $\gamma = 0.5$ são multiplicados pelos vetores a que estão associados (figura do lado esquerdo) dando origem aos três vetores a cheio na figura do lado direito. Estes três vetores, quando encadeados, dão origem ao vetor a tracejado, de P_{ps}^{i} para P_{ps}^{i+1} representando a movimentação do sensor da iteração i para a iteração i+1.

Um pseudo-código detalhado do enxame de partículas usado encontra-se em anexo.

Algoritmo 4: Cálculo da nova velocidade de uma partícula

input : $PART \leftarrow$ Uma solução do problema (partícula)

$V \leftarrow$ Velocidade atual da partícula

$MELPART \leftarrow$ Melhor solução encontrada por esta partícula

$MELHOR \leftarrow$ Melhor solução global encontrada até ao momento

$V = \alpha \times V + \beta \times (MELPART - PART) + \gamma \times (MELHOR - PART)$

output: V

Algoritmo 5: Enxame de partículas genérico

input : Parâmetros do algoritmo

$PART \leftarrow$ População de soluções iniciais (partículas)

$V \leftarrow$ Inicialização da velocidade de cada partícula

$MELHOR \leftarrow$ Melhor solução até ao momento

enquanto *Critério de paragem não for atingido* **faça**

- $V \leftarrow$ Cálculo da nova velocidade de cada partícula
- $PART \leftarrow PART + V$
- $MELHOR \leftarrow$ Melhor solução até ao momento

output: $MELHOR$

Capítulo 4

Algoritmo de atração

Os algoritmos estão constantemente a ser melhorados ao longo dos anos e muitas vezes é através da análise de outros algoritmos que surgem ideias híbridas potencialmente inovadoras (Juang, 2004; Kaveh e Malakouti Rad, 2010). Sendo o algoritmo genético e o enxame de partículas dois dos algoritmos evolutivos mais conhecidos, é normal que já tenham sido comparados e analisados inúmeras vezes (Eberhart e Shi, 1998; Hassan et al., 2005).

Claro que este tipo de análise pode não dar frutos, mas pode também dar origem a novas versões de algoritmos já existentes, ou até mesmo algoritmos completamente novos.

Tendo feito uma pequena introdução ao funcionamento geral dos dois algoritmos que o inspiraram, podemos agora passar à explicação do algoritmo de atração. Da mesma forma que certos algoritmos tiveram como inspiração o comportamento de formigas, o algoritmo de atração é inspirado nos comportamentos básicos de um ser humano quando confrontado com um problema de localização. Consideremos novamente o problema de cobertura binária enunciado anteriormente.

4.1 Problema de cobertura binária

Tal como com os algoritmos apresentados anteriormente, as soluções são representadas por uma matriz contendo as coordenadas dos sensores que a compõem. Como primeiro passo é escolhida uma solução inicial, ou seja, uma posição inicial para os

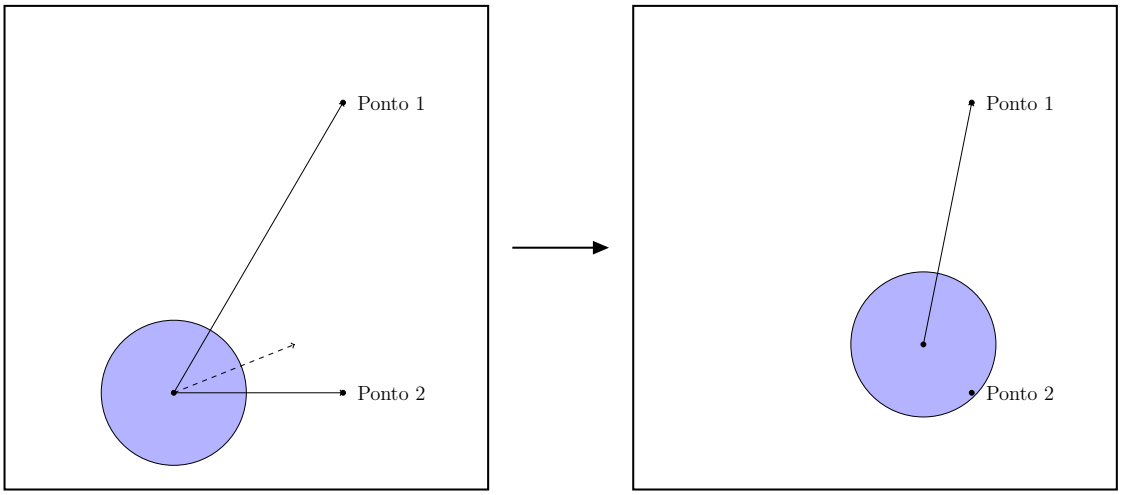


Figura 4.1: Influência de pontos não cobertos na atração exercida

sensores, aleatória ou não. De seguida, caso a solução não seja suficientemente boa, tenta-se melhorá-la. Para aumentar a cobertura da zona de interesse é necessário mover um ou mais sensores de forma a cobrir mais pontos da grelha de avaliação, mas como? O que deve condicionar e guiar esses movimentos?

Normalmente quando se tenta melhorar uma solução de um problema desta natureza, a primeira coisa que sobressai são as partes não cobertas da zona de interesse e de seguida o primeiro impulso seria mover o sensor mais próximo para cobrir essa parte, como que se este estivesse a ser atraído. Por este motivo, neste algoritmo, o principal fator que condiciona os movimentos dos sensores é a posição dos pontos não cobertos e tal como quando uma pessoa opta por cobrir um ponto escolhendo mover o sensor mais próximo, aqui também a distância entre o ponto e o sensor tem o seu papel. Quanto mais próximo um ponto não coberto estiver de um sensor, maior a atração exercida sobre ele.

Inspirado nos cálculos vetoriais dos algoritmos de enxame de partículas, em que em cada iteração é necessário pesar a atração entre vários pontos no espaço de pesquisa, aqui é utilizado cálculo vetorial para combinar a atração de vários pontos não cobertos, resultando num vetor final que traduz a movimentação do sensor (Figura 4.1).

Claro que este mecanismo usado isoladamente tem várias desvantagens.

O problema mais óbvio, uma vez que estamos a considerar um problema de

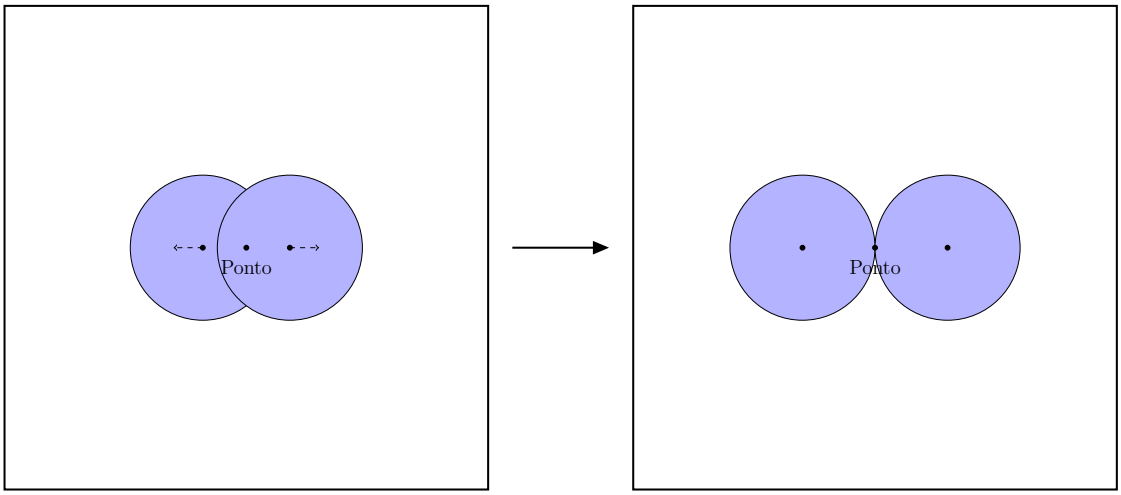


Figura 4.2: Repulsão exercida por um ponto duplamente coberto

cobertura binária, é a ausência de um método que desfavoreça a sobreposição de raios de deteção. Tal como quando nos deparamos com a área de cobertura de dois ou mais sensores sobrepostas o nosso primeiro impulso é afastá-los, esta questão pode ser resolvida através de um mecanismo de repulsão, que afaste um sensor de um ponto coberto por si e por pelo menos mais um sensor (Figura 4.2).

Uma outra desvantagem é que uma vez escolhida uma solução inicial, dificilmente a posição relativa entre sensores se altera muito. Uma vez que em cada iteração todos os sensores são atraídos pelo mesmo grupo de pontos, variando apenas a intensidade dessa atração em função da distância, um sensor que numa iteração esteja localizado à esquerda de um outro, dificilmente vai alterar muito a sua posição relativa em iterações futuras.

Esta dependência da solução inicial é uma característica péssima, pois faz com que a qualidade de soluções futuras dependa demasiado de uma escolha inicial. É necessário a existência de um mecanismo que permita ao algoritmo modificar mais facilmente a posição relativa dos sensores.

Um mecanismo de mutação, idêntico ao de um algoritmo genético, permite que em cada iteração, de acordo com uma certa probabilidade, coordenadas de um ou mais sensores sejam substituídas por valores aleatórios dentro do intervalo admissível. Desta forma as posições relativas entre os sensores da solução inicial perdem grande parte da sua influência em soluções futuras.

Tal como o mecanismo de mutação aleatoriza as coordenadas dos sensores, também os mecanismos de atração e repulsão possuem componentes aleatórias, para que o estado atual do sistema não determine por completo o estado seguinte.

Vejamos então um resumo das características principais do algoritmo de atração para o problema de cobertura binária:

- Mecanismo de atração (Maximiza a área coberta).
- Mecanismo de repulsão (Minimiza a área redundantemente coberta).
- Mecanismo de mutação (Diminui a dependência da solução inicial).
- Componentes de aleatorização (Torna o algoritmo não determinista).

Tendo introduzido o funcionamento do algoritmo para o problema simplificado, vejamos agora que alterações terá de sofrer para resolver o problema de cobertura probabilística original.

4.2 Problema de cobertura probabilística

Existem várias diferenças entre este problema e o anterior, o que obriga a certas mudanças no algoritmo.

Um vez que neste problema a cobertura dos sensores deixa de ser binária, passando a ser probabilistica, deixa de fazer sentido falar apenas de pontos cobertos e não cobertos, pois agora a probabilidade de deteção em cada ponto da grelha de avaliação pode tomar qualquer valor entre 0 e 1.

Este simples facto obriga a alterações no mecanismo de atração e faz com que o mecanismo de repulsão utilizado na versão anterior do algoritmo se torne contraprodutivo, pois pode ser necessária a sobreposição de zonas de deteção de vários sensores para atingir certos valores para a probabilidade de deteção. Felizmente, a própria natureza do problema torna este mecanismo desnecessário.

Tal como já foi explicado anteriormente, cada ponto da grelha tem três atributos, criticidade, vulnerabilidade e risco. Ao contrário do que se passava no problema anterior, em que todos os pontos da grelha de avaliação tinham igual importância,

agora um ponto é tão mais importante quanto maior o risco a ele associado e vice versa. O que significa que, de forma a refletir essa importância, quanto mais elevado for o risco associado a um ponto, maior a atração que este exerce nos sensores.

Se incorporarmos este novo fator no mecanismo de atração, temos que a magnitude da atração exercida por cada ponto da grelha de avaliação num sensor depende de:

- Posição do ponto relativamente ao sensor.
- Distância entre o ponto e o sensor.
- Risco associado ao ponto.
- Componente aleatória.

Algoritmo 6: Cálculo da velocidade no algoritmo de atração

input : $SOL \leftarrow$ Uma solução do problema

$RISCO \leftarrow$ Matriz com valores do risco em cada ponto da grelha

para *cada sensor* **faça**

  para *cada ponto da grelha* **faça**

    $rand \leftarrow$ Valor aleatório entre 0 e 1

    $pesorisco \leftarrow$ Peso baseado no risco neste ponto da grelha

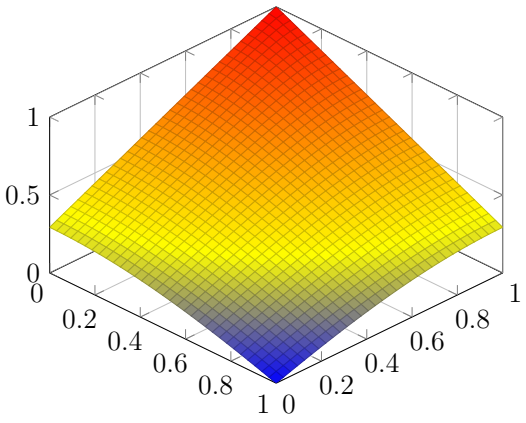
    $pesodist \leftarrow$ Peso baseado na distância entre o ponto e o sensor

    $vetor \leftarrow$ Vetor que vai do sensor até ao ponto

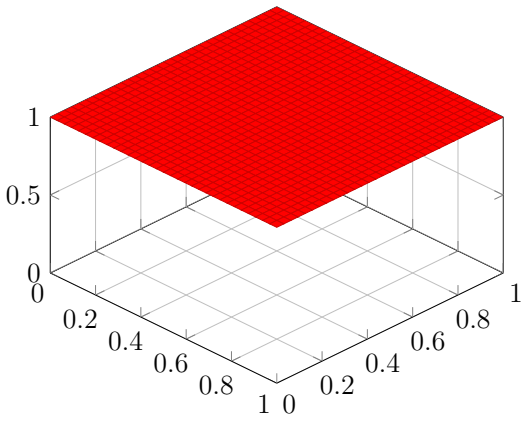
    $V = V + rand \times pesorisco \times pesodist \times vetor$

output: V

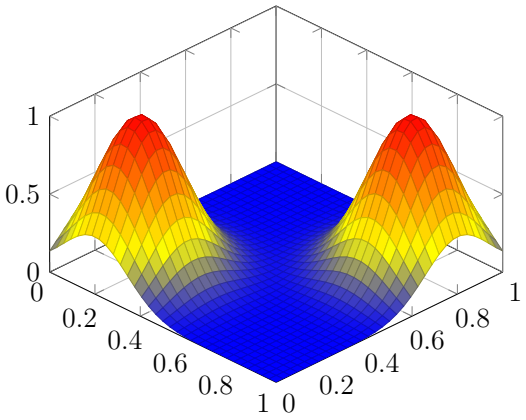
Desta forma, o mecanismo de repulsão torna-se desnecessário, pois quando dois sensores se aproximam, a probabilidade de deteção conjunta dos pontos que se encontram entre eles aumenta, o que por sua vez implica que o risco associado a esses pontos diminui. Ora, se pontos com um risco associado pequeno exercem uma menor atração sobre os sensores, permitem que estes sejam mais facilmente atraídos por outros pontos da grelha.



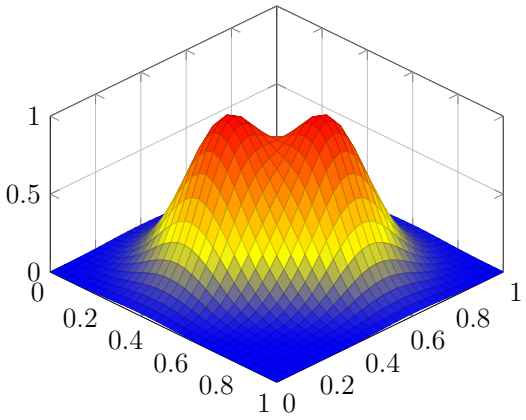
(a) Criticidade



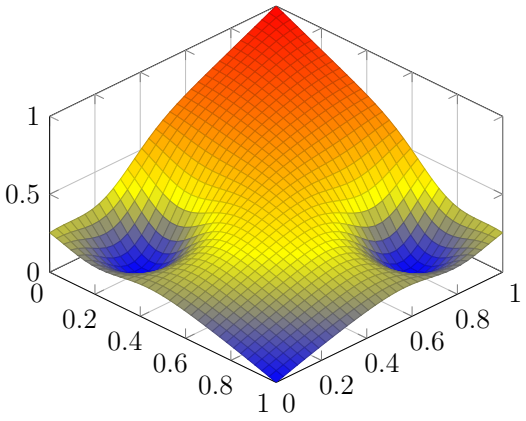
(b) Vulnerabilidade



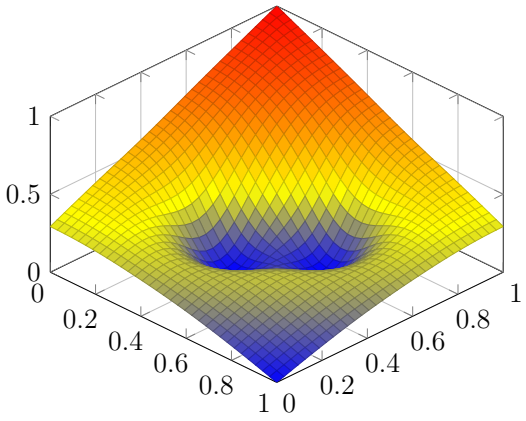
(c) Prob. de deteção conjunta, sensores afastados



(d) Prob. de deteção conjunta, sensores próximos



(e) Risco, sensores afastados



(f) Risco, sensores próximos

Figura 4.3: Mecanismo de repulsão torna-se desnecessário

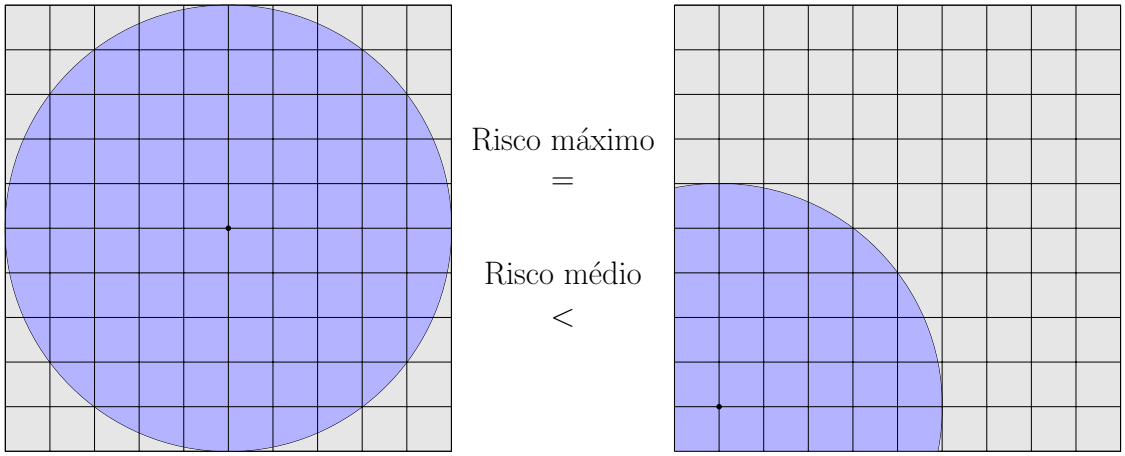


Figura 4.4: Exemplo de duas soluções

Como se pode verificar (Figura 4.3), o risco associado ao ponto $(\frac{1}{2}, \frac{1}{2})$ diminui bastante quando a distância entre os sensores diminui, o que resulta numa menor atração por parte do próprio.

Uma outra modificação consiste na forma de avaliação da qualidade de uma solução. Visto que os pontos da grelha de avaliação deixam de ter apenas dois estados, coberto e descoberto, deixa de fazer sentido calcular uma aproximação da área coberta da zona de interesse. Tendo em conta que agora o objetivo é minimizar o risco na zona de interesse, faz sentido que uma solução seja tanto melhor quanto maior for essa redução.

Mas que medida usar? Risco médio? Risco máximo? Outra medida baseada no risco?

Vamos assumir que todos os pontos da grelha de avaliação têm risco igual a 1. Como se pode ver (Figura 4.4) o risco máximo não avalia corretamente a qualidade da solução, pois apesar da diferença de qualidades ser óbvia, ambas têm risco máximo igual a 1. O mesmo não se passa com o risco médio, pois minimizar o risco médio é equivalente a minimizar o somatório do risco, o que significa que neste caso, é proporcional à quantidade de pontos não cobertos.

Para além disso, o máximo é uma medida mais susceptível a outliers e pequenas variações que a média, pois se o movimento de um sensor causar a diminuição do risco associado a vários pontos, mas simultaneamente aumentar muito o de um outro ponto, o valor do risco máximo pode aumentar muito e a solução ser de facto melhor.

Visto isto, uma vez que o valor de risco associado a cada ponto da grelha já tem em conta a criticidade e vulnerabilidade, é apenas natural que o objetivo seja minimizar o risco existente, ou seja o seu somatório ou risco médio, pois é uma medida intuitiva, robusta e que transmite bem a qualidade de uma solução.

Mas nem tudo necessita de ser modificado, o mecanismo de mutação pode manter-se inalterado uma vez que apenas altera a localização dos sensores, não tendo em conta o tipo de cobertura dos mesmos. Claro que neste caso, esta independência do resto vem do facto de ser um mecanismo bastante simples e pouco inteligente. Vejamos uma forma de modificar o mecanismo de mutação, para que as suas alterações nas soluções sejam um pouco mais inteligentes.

4.3 Utilidade e mutação probabilística

Consideremos um problema de cobertura probabilística em que apenas se tem de decidir a localização de um sensor. Suponhamos agora que existe um quarto atributo associado a cada ponto da grelha de avaliação, chamado utilidade. Ao colocarmos o sensor num ponto da zona admissível este vai ser mais ou menos útil conforme a redução no risco que provoca. O atributo utilidade pode tomar qualquer valor entre 0 e o risco inicial e corresponde à redução no risco do problema caso o sensor seja colocado nas coordenadas daquele ponto. Desta forma, sabemos o quão útil o sensor vai ser se for colocado naquele ponto.

O objetivo deste pré-processamento é permitir ao mecanismo de mutação decidir se um sensor está bem localizado e caso não esteja, mudar-lhe as coordenadas, tendo maior probabilidade de ser colocado num ponto com alto valor de utilidade.

Claro que se levantam várias questões:

- Significa que os sensores só podem ter como coordenadas os pontos da grelha?

  Não. Apesar do mecanismo de mutação só colocar sensores em coordenadas correspondentes a pontos da grelha de avaliação, como é usado sequencialmente com o mecanismo de atração, os sensores podem acabar por estar em qualquer local da zona de interesse (Figura 4.5).

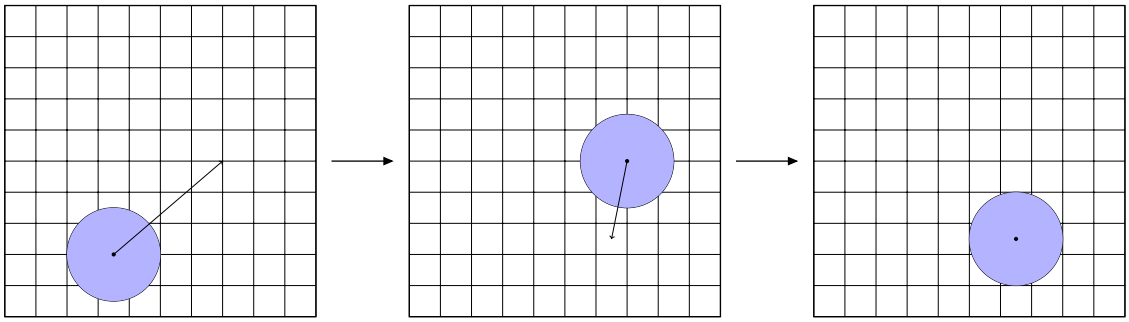


Figura 4.5: Efeito conjunto dos mecanismos de mutação e atração

- E se houver mais que um sensor, como é calculada a utilidade?

  São calculados valores de utilidade para cada tipo de sensor separadamente. Ou seja, para além dos atributos criticidade, vulnerabilidade e risco, cada ponto da grelha vai ter um atributo utilidade por cada tipo de sensor a usar. Isto porque se dois sensores do mesmo tipo forem colocados à vez num dado ponto duma zona de interesse vazia, o impacto no risco vai ser igual. Por outro lado, se os sensores tiverem especificações diferentes é pouco provável que o impacto no risco seja o mesmo.

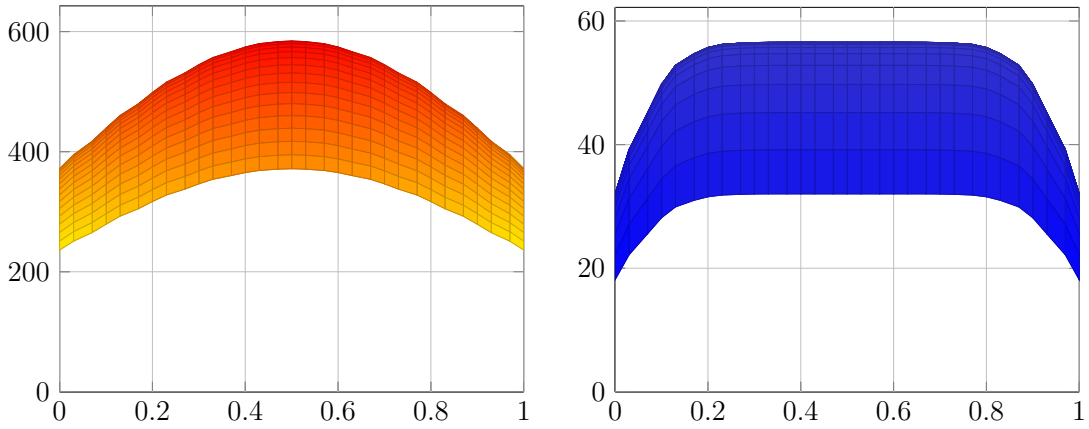
  Vejamos uma ilustração da utilidade associada a dois tipos de sensores diferentes, numa zona de interesse homogénea com risco igual a 1. Uma vez que neste caso vai haver uma simetria em relação ao ponto $(\frac{1}{2}, \frac{1}{2})$, a imagem mostra uma visão lateral da zona de interesse (Figura 4.6).

- Como é tomada a decisão de alterar as coordenadas de um sensor e como se escolhida a sua nova localização?

  A decisão de aplicar ou não a mutação a um sensor depende da sua localização atual e da matriz utilidade associada a esse tipo de sensor.

  Em primeiro lugar é calculada uma aproximação da utilidade do sensor na sua posição atual. Para tal é feito um arredondamento das suas coordenadas para o ponto da grelha de avaliação mais próximo, sendo-lhe atribuída a utilidade do ponto correspondente na matriz utilidade.

  Como se pode ver (Figura 4.7), a probabilidade de um sensor sofrer uma



(a) Sensor muito potente

(b) Sensor pouco potente

Figura 4.6: Utilidade de dois tipos de sensor diferentes

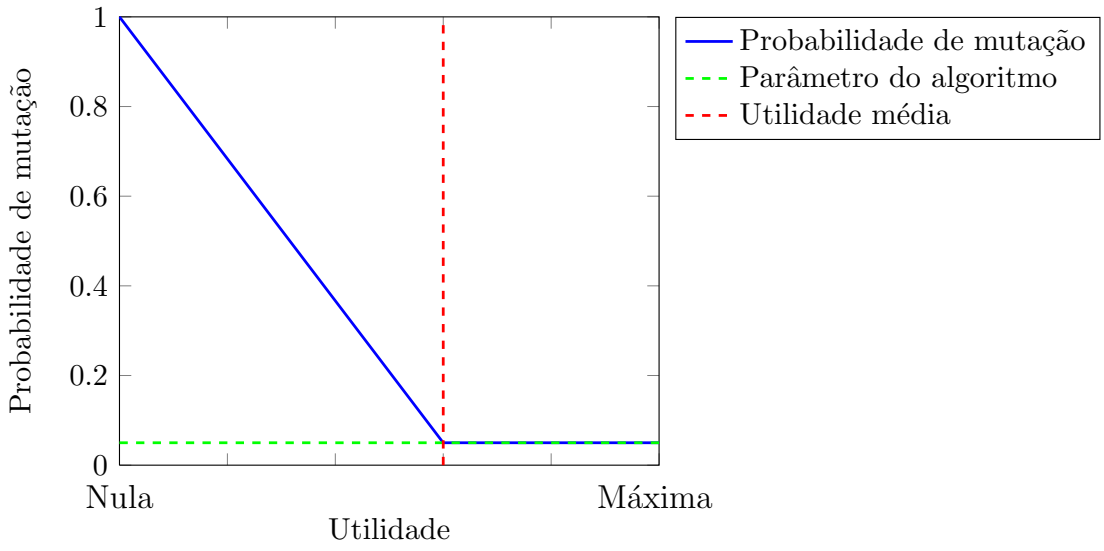


Figura 4.7: Probabilidade de mutação em função da utilidade

mutação é tanto maior quanto menor for a sua utilidade. No entanto, a relação entre a utilidade e a probabilidade de mutação muda quando a utilidade atual do sensor ultrapassa a linha da utilidade média. Desta forma, os sensores mal colocados tem uma maior probabilidade de sofrerem uma mutação, enquanto que os bem colocados tem uma maior probabilidade de continuarem a evoluir a sua posição naturalmente. A probabilidade de mutação no ponto de utilidade média é um parâmetro do algoritmo.

Algoritmo 7: Mutação probabilística do algoritmo de atração

input : $SOL \leftarrow$ Uma solução do problema

para *cada sensor* **faça**

- $utimedia \leftarrow$ Utilidade media deste tipo de sensor
- $uti \leftarrow$ Utilidade aproximada do sensor
- $prob \leftarrow$ Probabilidade de mutação baseada em uti e $utimedia$
- $rand \leftarrow$ Valor aleatório entre 0 e 1
- **se** $rand <= prob$ **então**
  - $SOL \leftarrow$ Nova posição do sensor baseada na sua matriz utilidade

output: SOL

Um pseudo-código detalhado do algoritmo de atração usado encontra-se em anexo.

Algoritmo 8: Algoritmo de atração genérico

input : Parâmetros do algoritmo

Cálculo da matriz utilidade de cada tipo de sensor

$SOL \leftarrow$ Solução inicial

enquanto *Critério de paragem for atingido* **faça**

- $V \leftarrow$ Cálculo da velocidade
- $SOL \leftarrow SOL + V$
- $SOL \leftarrow$ Aplicação de mutações se for o caso
- $MELHOR \leftarrow$ Melhor solução até ao momento

output: $MELHOR$

Capítulo 5

Otimização de parâmetros

Certos algoritmos necessitam que alguns parâmetros lhes sejam dados como input. E muitas vezes os valores escolhidos para esses parâmetros têm uma grande influência na qualidade das soluções apresentadas. Como tal, a não ser que seja conhecido o valor ótimo de um parâmetro para o problema em questão, é aconselhável fazer uma busca de forma a tentar descobri-lo.

5.1 Algoritmo genético

Parâmetros:

- Dimensão da população

  O primeiro passo num algoritmo genético consiste na criação da população inicial, mas para o poder fazer é necessário escolher a sua dimensão. Dependendo do problema a resolver, este parâmetro pode variar muito, mas normalmente encontra-se entre 100 e 500 (Gaspar-Cunha et al., 2012). Uma vez que o problema abordado neste trabalho é de pequena dimensão quando comparado com problemas da vida real, não se justifica assumir que tais dimensões sejam as mais apropriadas, como tal foram testados vários valores para este parâmetro.

  Foram geradas aleatoriamente 20 instâncias do problema e em cada uma, executou-se o algoritmo genético três vezes para cada valor do parâmetro $\in (8, 16, 24, ..., 120)$ e analisou-se a qualidade média da solução final. O valor

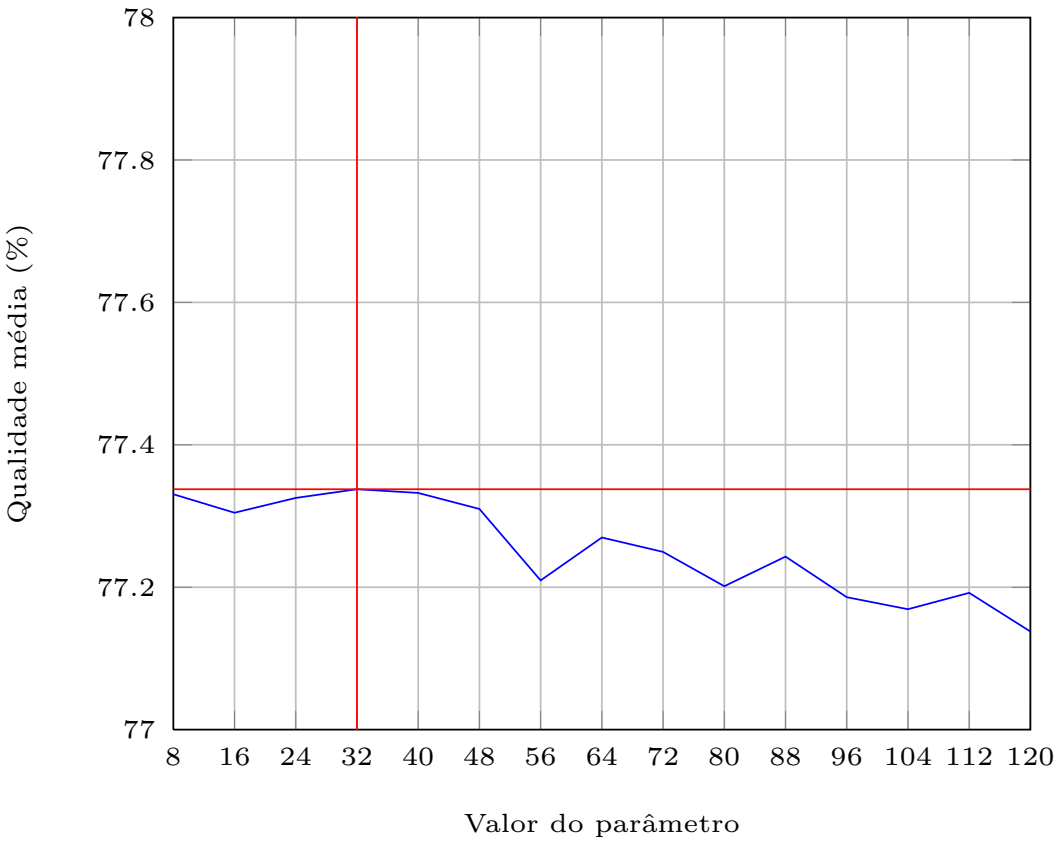


Figura 5.1: Valores testados para a dimensão da população

do parâmetro que obteve melhores resultados foi 32 (Figura 5.1).

- Probabilidade de mutação

  Este parâmetro determina a probabilidade de, em cada iteração do algoritmo, cada gene sofrer uma mutação. Naturalmente convém que esta probabilidade seja baixa, caso contrário, ocorrerão demasiadas mutações, destruindo a filosofia evolutiva do algoritmo. Os valores mais usados para este parâmetro encontram-se entre 0.001 e 0.01 (Gaspar-Cunha et al., 2012), ou seja, entre 0.1% e 1% de probabilidade de mutação.

  De forma a determinar o melhor valor para este parâmetro, fez-se o seguinte teste. Foram geradas aleatoriamente 20 instâncias do problema e em cada uma, executou-se o algoritmo genético três vezes para cada valor do parâmetro $\in (0, 0.001, 0.002, ..., 0.01)$ e analisou-se a qualidade média da solução final. O valor do parâmetro que obteve melhores resultados foi 0.007.

→ Os valores usados no capítulo 7 foram os seguintes:

Dimensão da população ← 32

Probabilidade de mutação ← 0.007

5.2 Enxame de partículas

Parâmetros:

- Número de partículas

  O número de partículas a usar depende muito da natureza do problema. Como tal, de forma a determinar o melhor valor para este parâmetro foram geradas aleatoriamente 20 instâncias do problema e em cada uma, executou-se o algoritmo genético três vezes para cada valor do parâmetro $\in (10, 15, 20, ..., 120)$ e analisou-se a qualidade média da solução final. O valor do parâmetro que obteve melhores resultados foi 50 (Figura 5.2).

- Alpha (α), beta (β) e gamma (γ)

  Para cada combinação de α, β e $\gamma \in (0, 0.1, ..., 1)$ tal que $\alpha + \beta + \gamma = 1$, executou-se o algoritmo de enxame de partículas 50 vezes e calculou-se a qualidade média das soluções. A combinação que originou melhores resultados foi $(\alpha, \beta, \gamma) = (0.8, 0.1, 0.1)$.

  De acordo com esta otimização, em cada iteração do algoritmo, a próxima posição de uma partícula deve depender 80% da velocidade atual da partícula, 10% da posição da melhor solução encontrada pela própria e 10% da posição da melhor solução encontrada até à iteração presente.

→ Os valores usados no capítulo 7 foram os seguintes:

Dimensão da população (número de partículas) ← 50

Alpha (α) ← 0.8

Beta (β) ← 0.1

Gamma (γ) ← 0.1

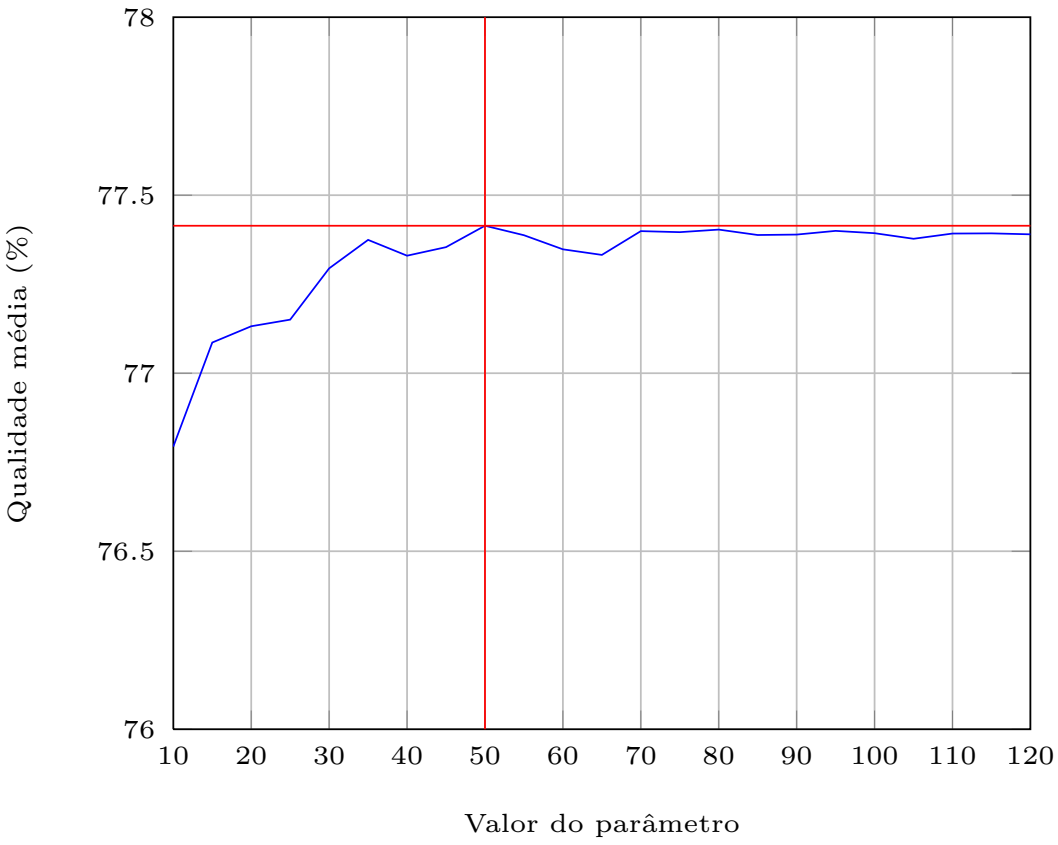


Figura 5.2: Valores testados para número de partículas

5.3 Algoritmo de atração

Parâmetros:

- Probabilidade de mutação de um sensor com utilidade acima da média

  De forma a determinar o melhor valor para este parâmetro, efetuou-se um teste similar ao que foi utilizado na determinação da probabilidade de mutação do algoritmo genético. Foram geradas aleatoriamente 20 instâncias do problema. Para cada combinação de instância e valor do parâmetro $\in (0, 0.005, 0.010, ..., 0.5)$, executou-se o algoritmo de atração três vezes e analisou-se a qualidade média da solução final. O valor do parâmetro que obteve melhores resultados foi 0.370 (Figura 5.3).

$\rightarrow$ O valor usado no capítulo 6 e 7 foi o seguinte:

Probabilidade de mutação para utilidade acima da média $\leftarrow$ 0.37

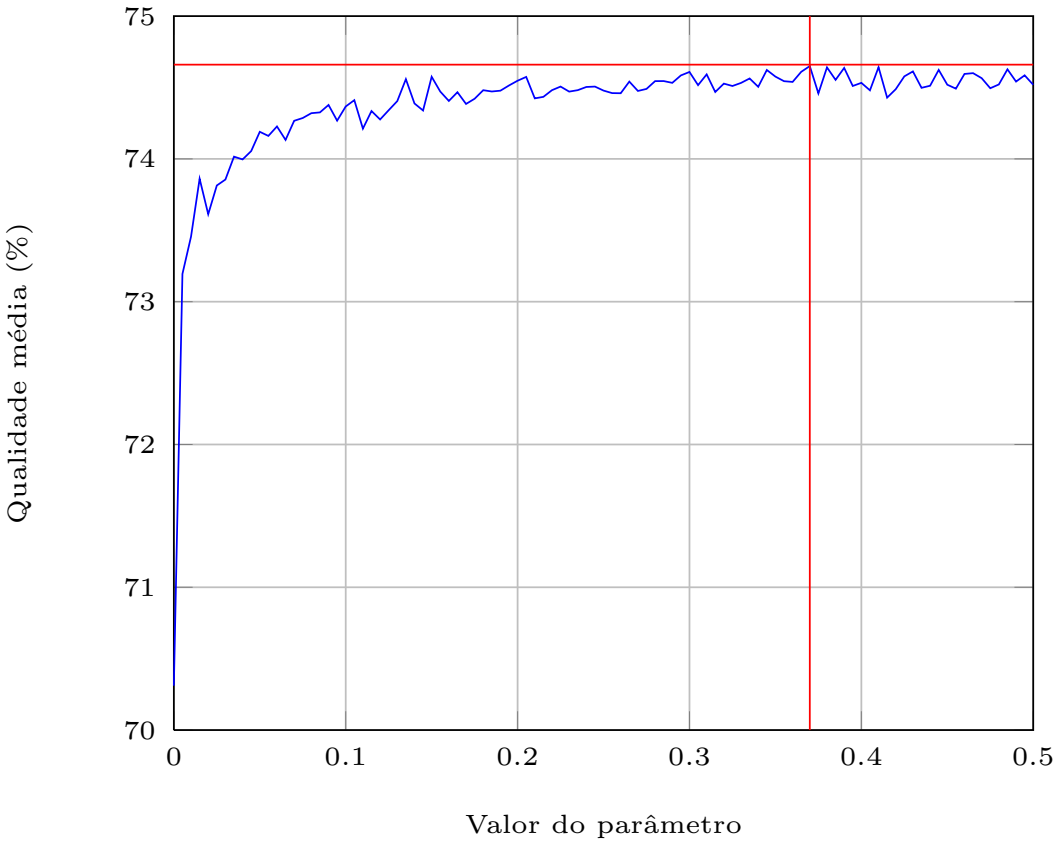


Figura 5.3: Valores testados para probabilidade de mutação

Capítulo 6

Discussão

Uma vez que uma heurística não garante a obtenção da solução ótima de um problema, é importante ter noção do quão longe as soluções obtidas estão do ótimo. Claro que para o fazer seria necessário saber a solução ótima à partida, o que significa que o problema já estaria resolvido.

Certas heurísticas, devido aos algoritmos que usam, conseguem garantir que as suas soluções estão a uma certa distância do ótimo. Infelizmente, esse não é o caso com o algoritmo de atração, como tal, para testar o seu desempenho, este foi aplicado a problemas cuja solução ótima já é conhecida.

6.1 Problema dos 4 sensores

Suponhamos que o risco tem valor 1 em todos os pontos da zona de interesse e que temos quatro sensores de cobertura binária com raio $\frac{\sqrt{2}}{4}$. Desta forma sabe-se que uma solução ótima é obtida colocando os sensores nas posições apresentadas na Figura 6.1.

Assim é possível avaliar o desempenho de um algoritmo, comparando as soluções obtidas com a solução ótima.

O algoritmo de atração foi aplicado a este problema 100 vezes, durando cada execução 30 segundos. No fim de cada uma das execuções foram guardadas a melhor solução encontrada e a sua qualidade, ou seja, a percentagem de risco anulado. De forma a avaliar o seu desempenho, coligiu-se os seguintes dados: a pior solução

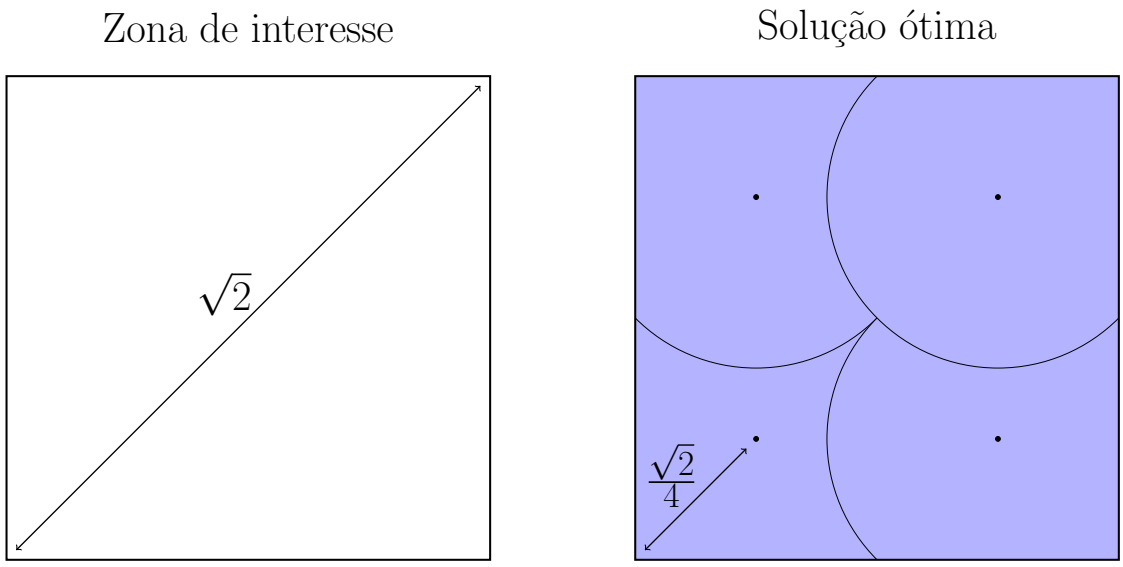


Figura 6.1: Solução ótima do problema com 4 sensores

obtida, a melhor solução obtida e a média da qualidade de todas as soluções (Figura 6.2).

6.2 Problema dos 9 sensores

Suponhamos agora que temos nove sensores de cobertura binária com raio $\frac{\sqrt{2}}{6}$, analogamente, obtém-se uma solução ótima através da disposição apresentada na Figura 6.3.

Repetindo o processo aplicado no problema anterior, foram obtidos os resultados apresentados na Figura 6.4.

Uma vez que se trata duma heurística, não há garantias de obter a solução ótima e, como era de esperar, a qualidade média das soluções piora à medida que o número de sensores aumenta e o problema se torna mais complexo.

Claro que nestes dois problemas os sensores foram definidos de forma a que a solução ótima só pudesse ser alcançada colocando os sensores em certos pontos específicos. Dependendo da quantidade e tipo de sensores disponíveis, pode tornar-se impossível ou trivial cobrir toda a zona de interesse.

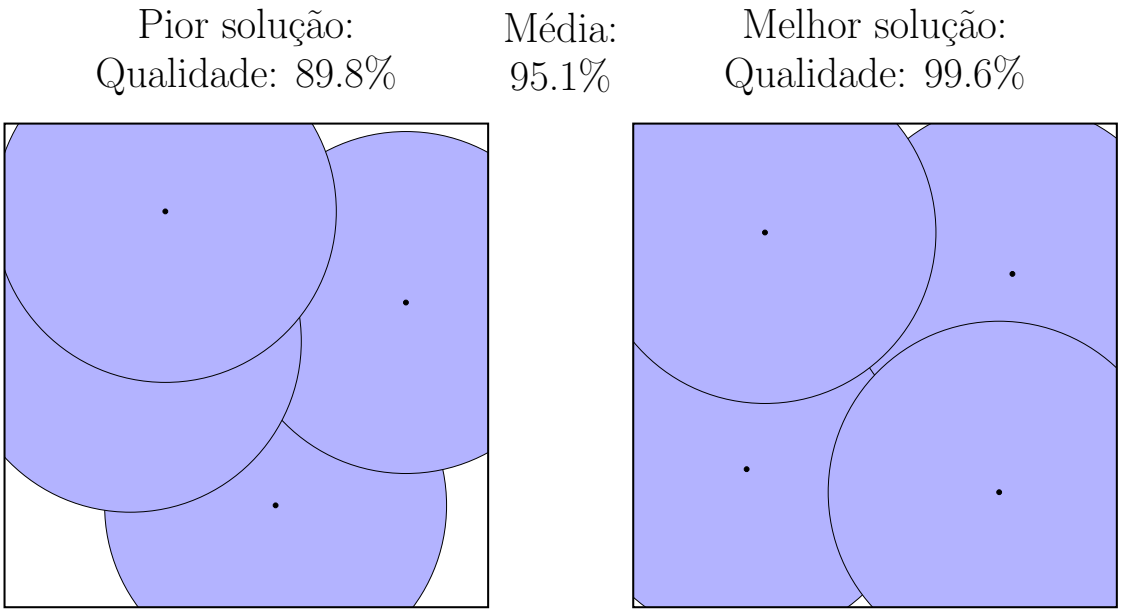


Figura 6.2: Resultados para o problema com 4 sensores

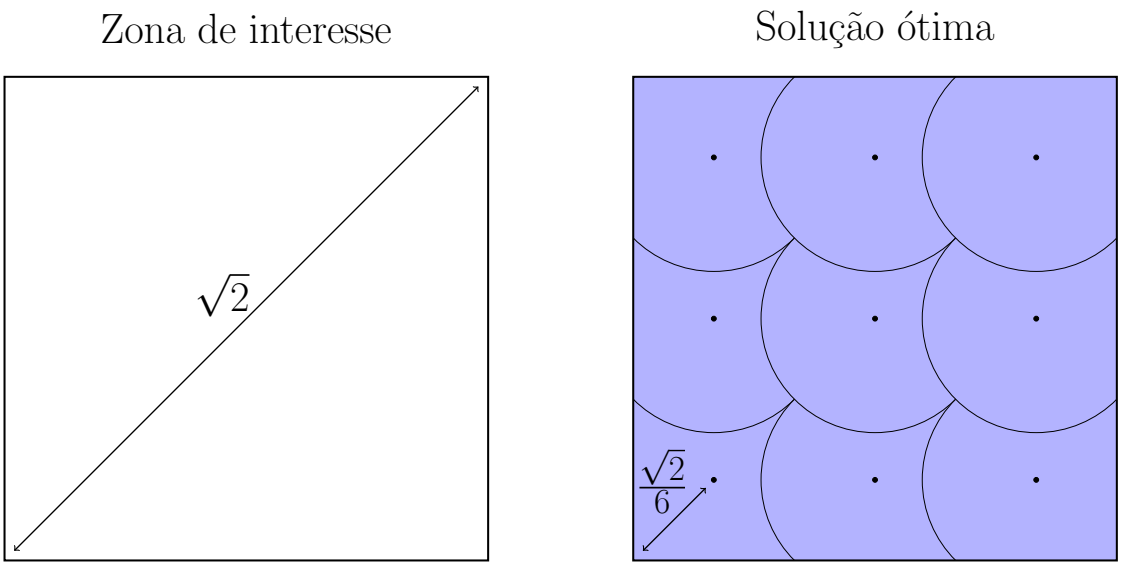


Figura 6.3: Solução ótima do problema com 9 sensores

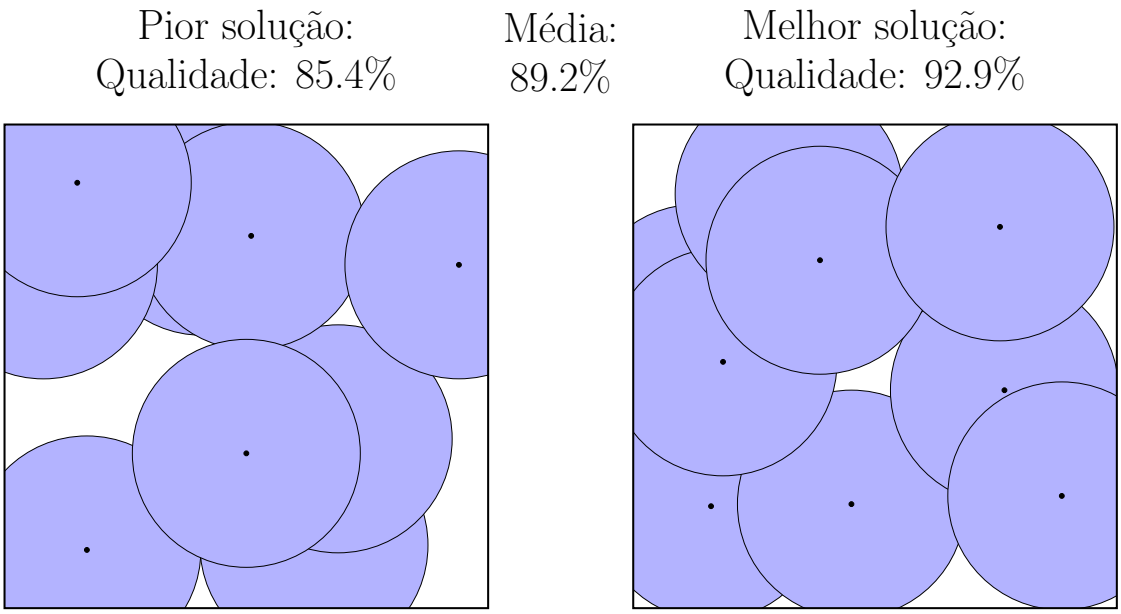


Figura 6.4: Resultados para o problema com 9 sensores

6.3 Problema combinatório

Como vimos nos dois problemas anteriores, é possível cobrir toda a zona de interesse usando 4 sensores de raio $\frac{\sqrt{2}}{4}$ ou 9 sensores de raio $\frac{\sqrt{2}}{6}$. Mas, e se ambos os tipos de sensor estiverem disponíveis simultaneamente em quantidades reduzidas?

Para se ter uma ideia do comportamento neste tipo de problemas, aplicou-se o algoritmo de atração a cada combinação possível dos dois tipos de sensor, guardando para cada uma delas a melhor solução encontrada.

Como é óbvio, se uma solução com uma certa quantidade de sensores de cada tipo tem uma certa qualidade ótima, qualquer solução construída a partir dessa através da adição de sensores, vai ter uma qualidade ótima maior ou igual à da solução original. Sabendo isto, faria sentido que na Figura 6.5, ao fixar a quantidade de sensores de um dos tipos, a qualidade tivesse um comportamento crescente ao longo dessa linha.

Claro que, uma vez que se usou uma heurística para determinar a qualidade de cada uma das soluções, não há garantias de que a qualidade associada a cada combinação possível de sensores seja a ótima. Isto justifica o facto de alguns valores não seguirem o comportamento esperado, como é o caso da solução (4,0) com qualidade

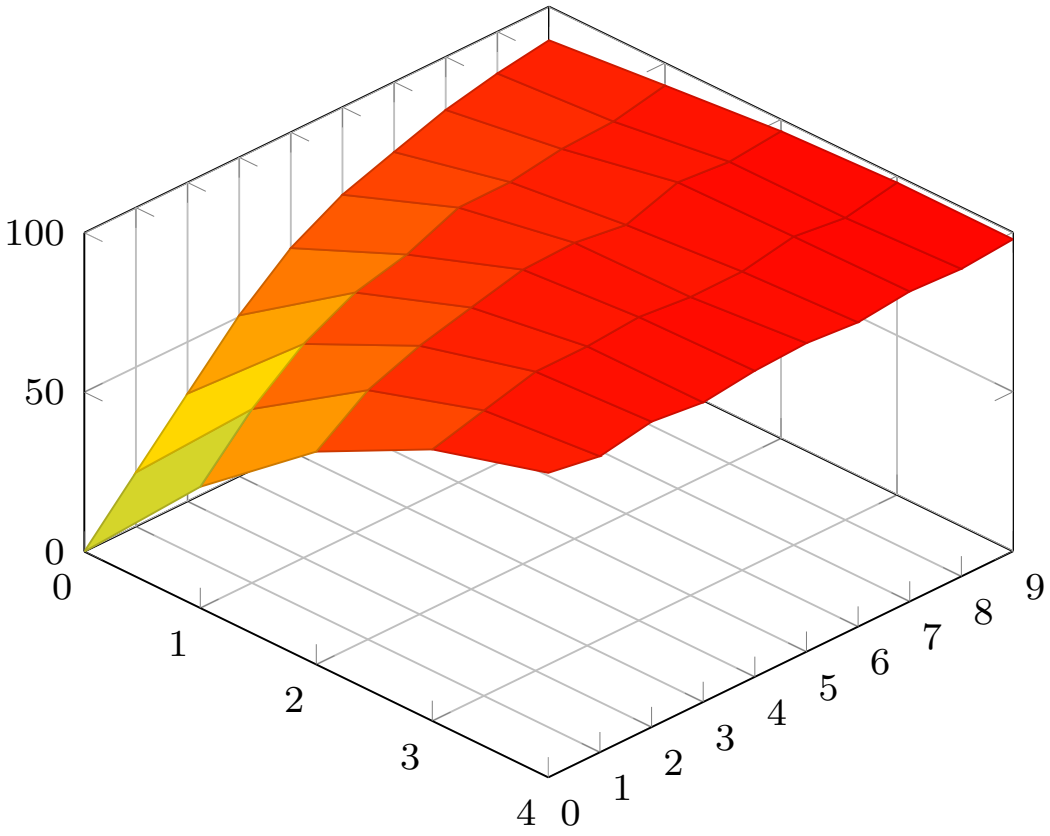


Figura 6.5: Qualidades obtidas para cada combinação de sensores

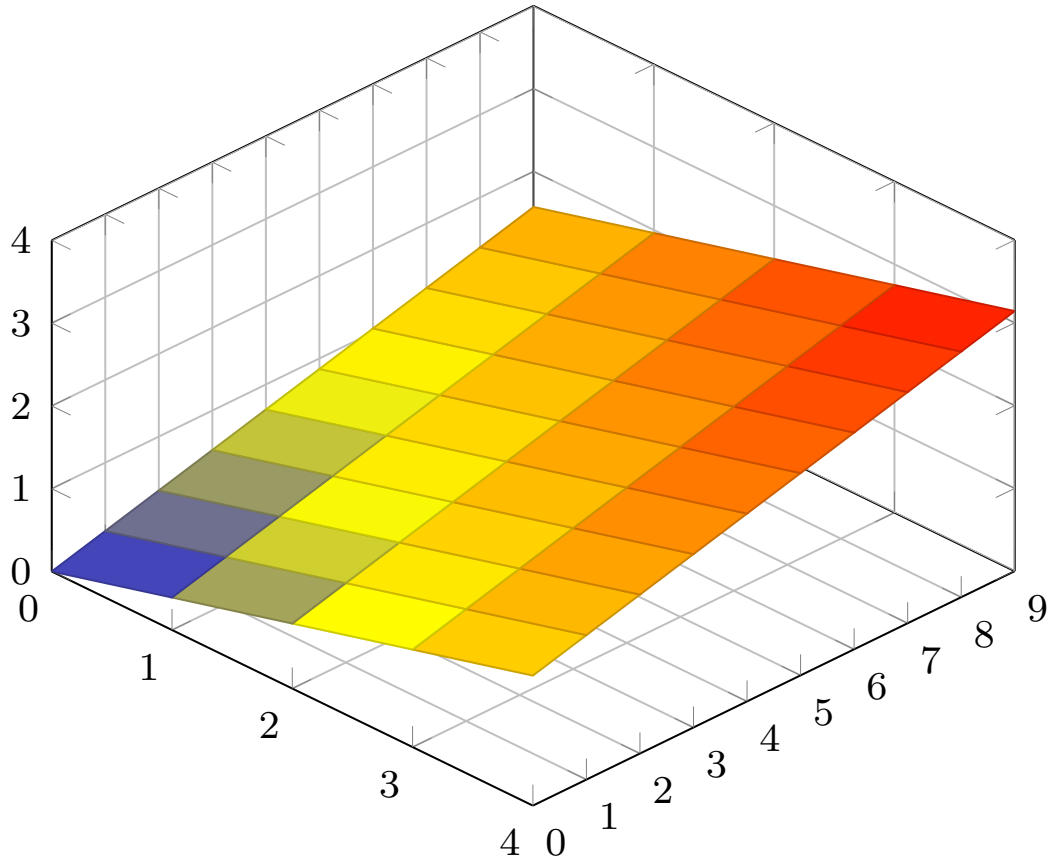


Figura 6.6: Custos de cada combinação possível de sensores

95.52% e a solução (4,1), que apesar de ter um sensor a mais, tem uma qualidade inferior, de 92.84%.

Neste tipo de problemas, faz sentido que, quanto maior o número de sensores que se está a usar, menor o ganho produzido por incluir mais um. Isto significa que se existissem custos associados ao uso dos sensores, a dado ponto, o aumento na qualidade duma solução proporcionado por usar mais um sensor não compensaria o custo adicional.

De forma a ilustrar esta ideia, suponhamos que o custo de um sensor equivale à área por si coberta. Como a área do círculo é dada por πr^2, temos então que, neste exemplo, um sensor com raio $\frac{\sqrt{2}}{4}$ tem custo $\frac{\pi}{8}$ e um com raio $\frac{\sqrt{2}}{6}$ tem custo $\frac{\pi}{18}$.

Suponhamos também que atribuímos igual importância aos dois objetivos, maximizar a cobertura e minimizar o custo. Uma vez que os dois objetivos são medidos em escalas diferentes, é aconselhável normalizá-los de forma a poderem ser combi-

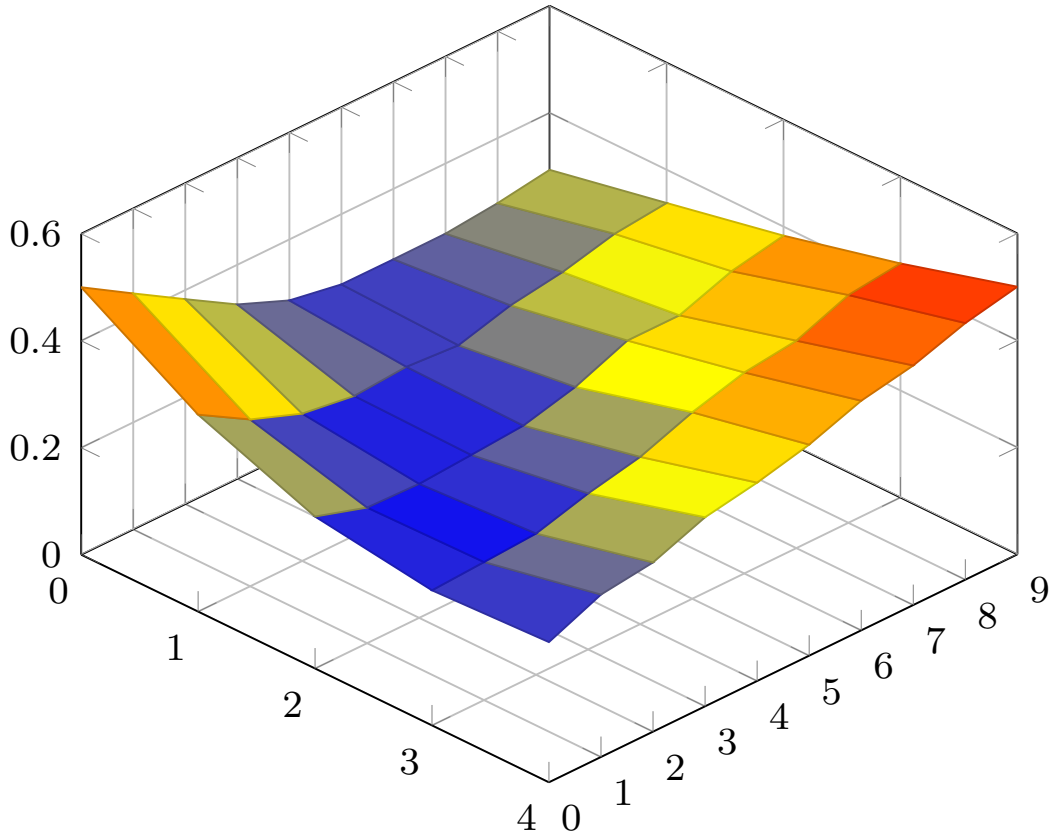


Figura 6.7: Análise das soluções tendo em conta os dois objetivos

nados.

Ao normalizar as qualidades (Figura 6.5), escalando todos os valores de forma a estarem entre 0 e 1 e não entre 0 e 100, o objetivo de maximizar a qualidade torna-se equivalente a minimizar (1-qualidade). Se de seguida normalizarmos também os custos (Figura 6.6), pode-se então combiná-los com as qualidades normalizadas da seguinte forma:

$$Peso_q \times (1 - Qualidade) + Peso_c \times Custos$$

sendo que, neste caso, estamos a atribuir igual peso a ambos os objetivos, temos $Peso_q = Peso_c = \frac{1}{2}$, por isso ficamos com a seguinte função objetivo:

$$min(\tfrac{1}{2} \times (1 - Qualidade) + \tfrac{1}{2} \times Custos)$$

Obtendo assim a figura 6.7.

Como se pode confirmar, ao combinar ambos os objetivos, certas soluções são postas de parte, quer por terem um número insuficiente de sensores, piorando a qualidade, quer por terem demasiados sensores, onde o acréscimo na qualidade não compensa o custo adicional. Assim, dependendo dos pesos atribuídos a cada objetivo, as melhores soluções são as que arranjam um compromisso entre uma boa cobertura e um baixo custo. Neste caso particular, as soluções mais interessantes seriam as representadas a azul escuro, nomeadamente as soluções (0,5), (1,4), (1,5), (2,1), (2,2), (2,3), (3,0), (3,1), (3,2) e (4,0).

Capítulo 7

Resultados

Gerámos 20 instâncias do problema definidas por:

- Valores de criticidade para cada ponto da grelha

  São gerados entre 1, 2 ou 3 pontos críticos dentro da zona de interesse representando infraestruturas a defender. Cada ponto crítico tem um desvio padrão associado que serve para definir como a sua criticidade se propaga em função da distância. De seguida são combinadas as várias criticidades de forma a criar a matriz que define a criticidade de cada ponto da grelha.

- Valores de vulnerabilidade para cada ponto da grelha

  São gerados entre 0, 1 ou 2 pontos dentro da zona de interesse, representando cada um deles um sensor com posição fixa dentro da zona de interesse. Cada um destes sensores tem um desvio padrão associado que serve para definir como a sua probabilidade de deteção se propaga em função da distância. De seguida são combinadas as várias probabilidades de deteção de forma a criar a matriz que define a vulnerabilidade de cada ponto da grelha.

- Número de sensores a usar

  Valor inteiro entre 1 e 4.

- Número de tipos de sensor

  Valor inteiro entre 1 e mínimo entre 3 e o número de sensores a usar.

- Desvio padrão associado a cada tipo de sensor

  Determina a potência do sensor, sendo gerados tantos valores quanto os tipos de sensor. Cada valor é número real entre 0.1 e 0.3.

O objetivo foi aplicar cada um dos três algoritmos a cada uma das instâncias e tabelar o seu desempenho.

Para avaliar o desempenho de uma heurística é necessário aplicá-la a um problema e analisar a qualidade da solução que apresenta. Mas uma vez que nenhuma das três heurísticas deste trabalho é determinista, os seus resultados podem ser diferentes quando aplicadas mais que uma vez à mesma instância.

Ou seja, para poder avaliar o desempenho de cada um destes algoritmos numa instância do problema é necessário aplicá-los mais que uma vez a essa mesma instância. Desta forma, não se corre um risco tão grande de atribuir uma medida de qualidade errada a uma heurística, pois esta é avaliada tendo em conta vários dos seus desempenhos.

Assim, cada heurística foi aplicada 5 vezes a cada uma das 20 instâncias, sendo depois tabelada a qualidade média de cada conjunto de 5 repetições.

Em condições normais seriam utilizados critérios de paragem diferentes (Capítulo 8.1), mas uma vez que neste trabalho o objetivo não é resolver o problema mas sim comparar desempenhos em tempo útil, foram utilizados os seguintes critérios de paragem:

- Qualidade da melhor solução = 100 (100% de redução do risco)
- Tempo de execução $\geq$ 120 segundos

Como se pode ver (Tabelas 7.1 e 7.2) a qualidade média das soluções obtidas pelos três algoritmos é bastante similar. Apesar do enxame de partículas ter obtido o melhor resultado médio em 13 das 20 instâncias, o algoritmo genético obteve o melhor desempenho global. Podemos também verificar que o algoritmo de atração teve o pior desempenho global, bem como em cada uma das 20 instâncias.

Uma vez que os três algoritmos são heurísticos, é interessante analisar a melhor solução encontrada em função do tempo de execução. Para este efeito foi gravada,

Instâncias	**Qualidades médias (% de risco anulado)**		
	Alg. Genético	**Enx. Partículas**	**Alg. Atração**
1ª	**43.1911**	43.1328	42.2001
2ª	**66.1353**	66.133	65.6554
3ª	77.7085	**77.7801**	77.0121
4ª	**48.8138**	48.3711	48.394
5ª	36.0626	**36.065**	36.0286
6ª	42.4746	**42.4805**	42.4646
7ª	64.6688	**64.7034**	64.2376
8ª	**78.7967**	77.9668	78.0108
9ª	33.8883	**33.898**	33.8864
10ª	41.5541	**41.5545**	41.4626
11ª	26.5013	**26.5014**	26.4982
12ª	38.2085	**38.2087**	38.1951
13ª	**71.885**	71.4013	71.0879
14ª	48.3927	**48.3932**	48.3859
15ª	27.906	**27.9065**	27.9013
16ª	**83.8192**	82.6512	80.7599
17ª	43.8339	**43.834**	43.8234
18ª	53.7502	**53.7504**	53.7262
19ª	41.6117	**41.6121**	41.608
20ª	**53.7671**	52.3449	53.6707
Médias:	**51.1485**	50.9344	50.7504

Tabela 7.1: Desempenho dos algoritmos

Instâncias	**Proporções (%) relativamente à melhor solução**		
	Alg. Genético	**Enx. Partículas**	**Alg. Atração**
1ª	**100**	99.8649	97.7055
2ª	**100**	99.9965	99.2744
3ª	99.9079	**100**	99.0126
4ª	**100**	99.0931	99.14
5ª	99.9933	**100**	99.8991
6ª	99.9861	**100**	99.9626
7ª	99.9465	**100**	99.2801
8ª	**100**	98.9468	99.0026
9ª	99.9714	**100**	99.9658
10ª	99.999	**100**	99.7788
11ª	99.9996	**100**	99.9879
12ª	99.9995	**100**	99.9644
13ª	**100**	99.3271	98.8911
14ª	99.999	**100**	99.9849
15ª	99.9982	**100**	99.9814
16ª	**100**	98.6065	96.3501
17ª	99.9998	**100**	99.9758
18ª	99.9996	**100**	99.955
19ª	99.999	**100**	99.9901
20ª	**100**	97.3549	99.8207
Médias:	**99.99**	99.6595	99.3961

Tabela 7.2: Proporções à melhor qualidade obtida por instância

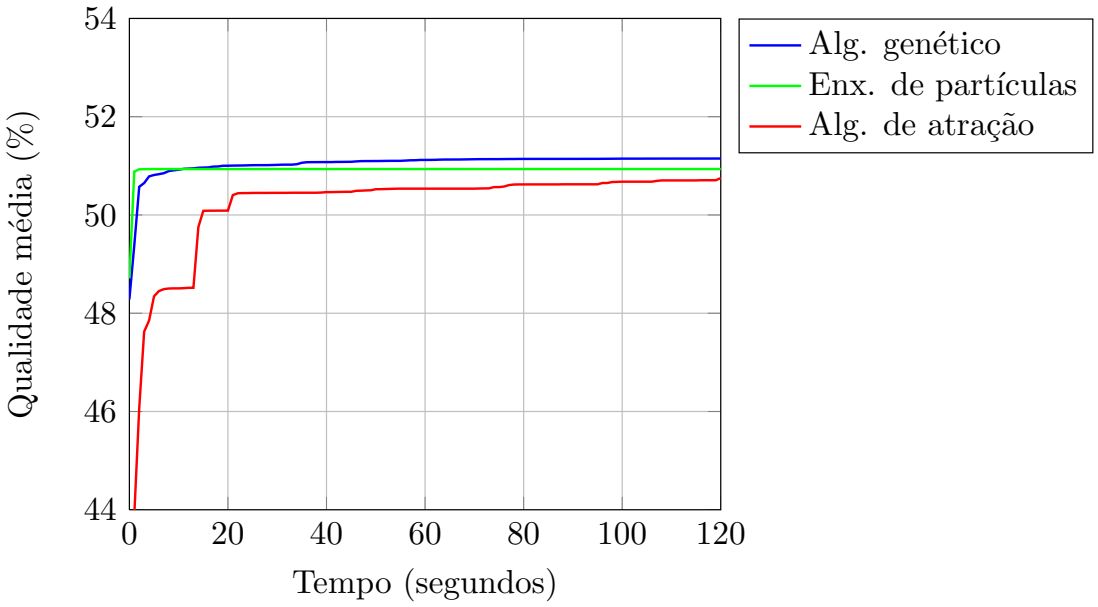


Figura 7.1: Média das qualidades das melhores soluções em função do tempo

a cada segundo de execução, a qualidade da melhor solução encontrada até ao momento, sendo de seguida os dados agrupados por algoritmo. Finalmente, fizeram-se as médias dos dados correspondentes a cada segundo de execução, obtendo assim um gráfico para cada algoritmo (Figura 7.1).

Como se pode verificar, o enxame de partículas obteve as melhores soluções nos primeiros segundos de execução, mas aparenta estagnar rapidamente, sendo ultrapassado pelo algoritmo genético perto de dez segundos após o inicio da execução. Assim, apesar de não evoluir tão depressa inicialmente, o algoritmo genético acaba por ultrapassar o enxame de partículas, conseguindo após esse momento manter sempre o primeiro lugar ao longo do resto do tempo de execução. Estes comportamentos podem significar que o algoritmo genético implementado é mais robusto que o enxame de partículas, ou que a falta de um mecanismo de mutação pode prejudicar o desempenho neste tipo de problemas, podendo levar a convergências prematuras.

No que toca ao algoritmo de atração, este apresenta as piores soluções durante a totalidade dos dois minutos de execução. Isto pode ser justificado através do facto de, ao contrário dos outros dois algoritmos, não ser utilizada uma população de soluções, fazendo com que a qualidade da solução inicial seja mais baixa.

Ou seja, no início de uma execução, tanto o algoritmo genético como o enxame de partículas geram várias soluções aleatórias de forma a criar a população inicial. Como tal, é escolhida como melhor solução até ao momento, a solução pertencente à população com a qualidade mais elevada. O mesmo não se passa com o algoritmo de atração pois este apenas tem uma solução inicial que, apesar de não ser simplesmente aleatória, não consegue competir com uma população de soluções.

Para além disso, a inexistência de uma população impossibilita buscas em paralelo e partilha de informação entre soluções como as presentes nos outros dois algoritmos.

Capítulo 8

Conclusão

Neste trabalho foi proposto um algoritmo para a resolução de problemas de localização de sensores. O Algoritmo proposto é inspirado em dois algoritmos evolutivos, o algoritmo genético e enxame de partículas e tenta imitar os raciocínios básicos de um ser humano quando confrontado com um problema de localização de sensores.

Fez-se uma otimização dos parâmetros de cada algoritmo, sendo de seguida efetuados testes comparativos entre os três algoritmos de forma a avaliar os seus desempenhos. Embora os três algoritmos alcancem soluções de qualidade similar durante o primeiro minuto de execução, os resultados mostram que o algoritmo genético tem claramente o melhor desempenho dos três. Apesar de o enxame de partículas obter soluções de qualidade superior às dos outros nos primeiros segundos de execução, aparenta estagnar, permitindo uma ultrapassagem da parte dos outros dois algoritmos.

Quanto ao algoritmo de atração, obtém as piores soluções durante os primeiros segundos de execução pois começa com uma solução inicial de qualidade inferior. Apesar deste facto, acaba por ultrapassar o enxame de partículas, conseguindo por vezes aproximar-se do algoritmo genético, mas sem nunca o alcançar.

Através destes resultados tornam-se claras algumas das vantagens em usar um algoritmo que tire partido de uma população de soluções. Não só a qualidade da solução inicial é bastante melhor pois é escolhida a melhor pertencente à população, como é possível tirar partido de interações entre soluções para guiar a evolução de soluções futuras.

É importante ter em mente que é sempre possível obter resultados mais fidedignos aumentando o número de instâncias e repetições corridas, tanto na otimização de parâmetros dos algoritmos como nas posteriores comparações de desempenho. Na otimização de parâmetros os valores obtidos como ótimos não se destacam muito em relação aos restantes, levantando dúvidas sobre se realmente serão os ótimos, ou se diferentes resultados teriam sido obtidos se tivessem sido corridas mais instâncias de teste.

Independentemente das incertezas que advém do facto dos três algoritmos serem heurísticas, os resultados obtidos permitiram fazer uma comparação interessante dos seus desempenhos.

8.1 Desenvolvimentos futuros

Existem inúmeras ideias que provavelmente melhorariam o algoritmo de atração:

- Novo critério de paragem

  Um critério de paragem dinâmico baseado na evolução da qualidade das soluções obtidas em cada iteração. Assim, é permitida a continuação de execuções que estejam a conseguir melhorar a qualidade das suas soluções e são terminadas aquelas que já não o estão a conseguir, poupando tempo.

- Cálculo da movimentação dos sensores

  A ideia por detrás da movimentação dos sensores faz sentido, mas pode ser melhorada. Depois de observar os resultados de dezenas de execuções, percebeu-se que a qualidade das soluções aumentava muitas vezes ao longo do tempo, mas muito pouco de cada vez. Como tal, uma revisão dos componentes que fazem parte deste cálculo, e da forma como cada um deles é por sua vez calculado, pode levar a melhorias na qualidade das soluções encontradas.

- Utilidade dinâmica

  Apesar do conceito de utilidade ser um dos pilares deste algoritmo, a versão aqui implementada está longe de perfeita. Em primeiro lugar, a utilidade é

calculada para cada tipo de sensor assumindo que só existe um sensor e que é daquele tipo, o que, exceto em casos triviais, não é verdade. Segundo, a utilidade é estática: depois de calculada nunca mais é alterada. O ideal seria esta evoluir à medida que o algoritmo ia aprendendo os locais onde os sensores de cada tipo tiveram melhor desempenho.

- População de soluções

  Uma população de soluções, mesmo que pequena, pode ser uma grande vantagem no que diz respeito a estratégias de evolução. O simples facto de existir mais que uma solução simultaneamente possibilita o uso de diferentes formas de criar novas soluções. Duas pesquisas paralelas poderiam ter objectivos diferentes, uma tentando melhorar as soluções já encontradas (*exploitation*), a outra procurando soluções radicalmente diferente dessas (*exploration*). Poderiam, por exemplo, ser aproveitadas sinergias entre o que cada solução aprendeu até ao momento através das suas matrizes dinâmicas de utilidade.

Em relação aos outros dois algoritmos, ao longo dos anos foram criadas inúmeras variantes na tentativa de os melhorar. As versões usadas neste trabalho são bastante simples e provavelmente podem ser melhoradas através da implementação de algumas ideias simples, por exemplo:

- Mecanismo de cruzamento mais inteligente - (Algoritmo genético)

  O mecanismo usado é bastante básico, escolhendo um ponto de corte de forma aleatória e impossibilitando o aparecimento de certas soluções descendentes que de outra forma poderiam ser benéficas. A implementação de um mecanismo de cruzamento inteligente, não tão aleatório poderia melhorar as qualidades das soluções descendentes.

- Revisão da restrição $\alpha + \beta + \gamma = 1$ - (Enxame de partículas)

  Apesar da restrição ter o papel de evitar que novas soluções não sejam admissíveis, também pode ser responsável pela convergência prematura do algoritmo. Uma revisão desta restrição poderia ser benéfica se conseguisse mini-

mizar o seu impacto negativo no que diz respeito à capacidade do algoritmo explorar o espaço de busca.

- Automatização da dimensão da população (Alg. genético e Enx. de partículas)

  No caso do algoritmo genético e do enxame de partículas, o problema da otimização da dimensão das populações podia ser evitado usando um sistema que determinasse o tamanho da população a usar baseado no número de dimensões do problema:

  $$n \text{ sensores} = (x_1, y_1 \ldots x_n, y_n) = 2 \times n \text{ coordenadas (dimensões)}$$

  Assim, para além de ter menos um parâmetro para otimizar, teria-se uma população de dimensão apropriada independentemente da instância do problema considerada.

Bibliografia

Ian Akyildiz, Weilian Su, Yogesh Sankarasubramaniam e Erdal Cayirci. Wireless sensor networks: A survey. *Computer Networks*, 38(4):393–422, 2002.

V. Baby Vennila, R.K. Gnanamurthy e B. Bhuvaneswari. Wireless sensor network for forest fire sensing and detection in tamilnadu. *Engineering Science and Technology: An International Journal*, 2(2):306–309, 2012.

Mihaela Cardei e Jie Wu. Coverage in wireless sensor networks. *Handbook of Sensor Networks*, páginas 422–433, 2004.

Russell Eberhart e Yuhui Shi. Comparison between genetic algorithms and particle swarm optimization. *Evolutionary Programming VII*, páginas 611–616, 1998.

António Gaspar-Cunha, Ricardo Takahashi e Carlos Antunes, editors. *Manual de Computação Evolutiva e Metaheurística*. Imprensa da Universidade de Coimbra, 2012.

Rania Hassan, Babak Cohanim, Olivier De Weck e Gerhard Venter. A comparison of particle swarm optimization and the genetic algorithm. In *Proceedings of the 1st AIAA Multidisciplinary Design Optimization Specialist Conference*, páginas 1–13, 2005.

John Holland. Outline for a logical theory of adaptive systems. *Journal of the Association for Computing Machinery*, 9:297–314, 1962.

Chia-Feng Juang. A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 34(2):997–1006, 2004.

A. Kaveh e S. Malakouti Rad. Hybrid genetic algorithm and particle swarm optimization for the force method-based simultaneous analysis and design. *Iranian Journal of Science and Technology, Transaction B: Engineering*, 34(B1):15–34, 2010.

James Kennedy e Russell Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, volume 4, páginas 1942–1948, 1995.

Annie Liu, Julian Bunn e K. Mani Chandy. Sensor networks for the detection and tracking of radiation and other threats in cities. In *Proceedings of the 10th International Conference on Information Processing in Sensor Networks*, páginas 1–12, 2011.

Silvia Santini, Benedikt Ostermaier e Andrea Vitaletti. First experiences using wireless sensor networks for noise pollution monitoring. In *Proceedings of the Workshop on Real-World Wireless Sensor Networks*, páginas 61–65, 2008.

Wei Wang, Vikram Srinivasan, Bang Wang e Kee-Chaing Chua. Coverage for target localization in wireless sensor networks. *IEEE Transactions on Wireless Communications*, 7(2):667–676, 2008.

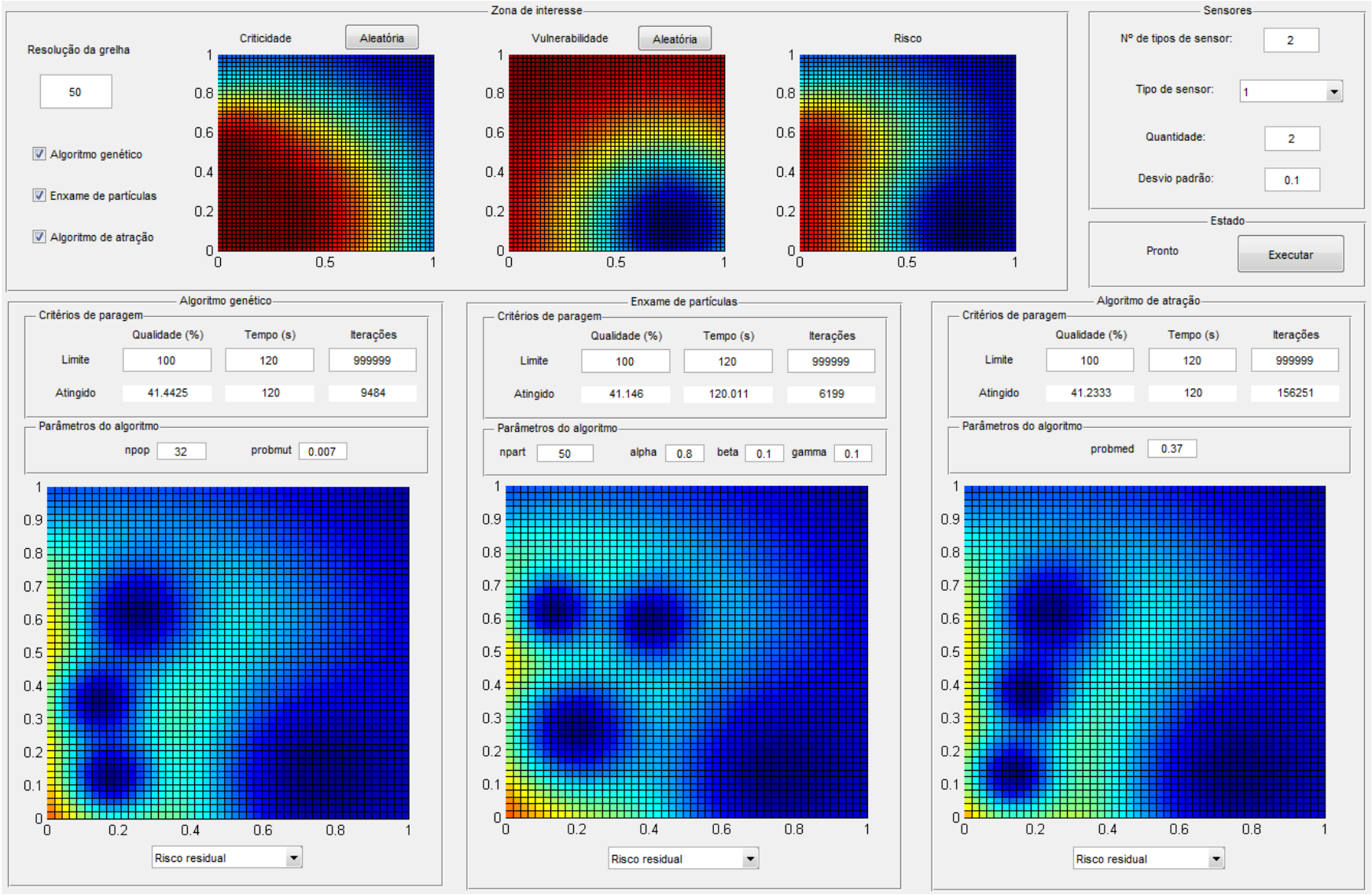
Henry Willis. Guiding resource allocations based on terrorism risk. RAND Corporation Working Papers, 2006.

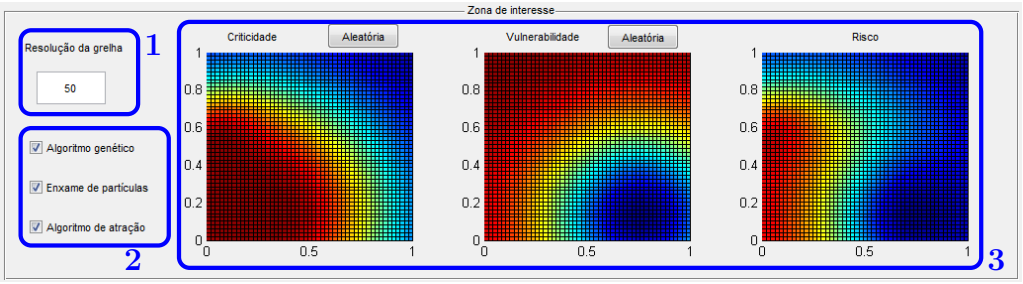
Henry Willis, Andrew Morral, Terrence Kelly e Jamison Medby. Estimating terrorism risk. Relatório técnico, RAND Corporation, 2005.

Ben Wisner, Piers Blaikie, Terry Cannon e Ian Davies. *At Risk: Natural Hazards, People's Vulnerability and Disasters*. Routledge, 2ª edição, 2003.

Apêndice A

Interface gráfico

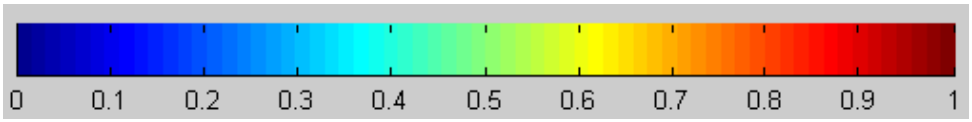

Zona de interesse
Resolução da grelha
50
Algoritmo genético
Enxame de partículas
Algoritmo de atração
Criticidade
Aleatória
Vulnerabilidade
Aleatória
Risco
Sensores
Nº de tipos de sensor:
2
Tipo de sensor:
1
Quantidade:
2
Desvio padrão:
0.1
Estado
Pronto
Executar
Algoritmo genético
Critérios de paragem
Qualidade (%)
Tempo (s)
Iterações
Limite
100
120
999999
Atingido
41.4425
120
9484
Parâmetros do algoritmo
npop
32
probmut
0.007
Risco residual
Enxame de partículas
Critérios de paragem
Qualidade (%)
Tempo (s)
Iterações
Limite
100
120
999999
Atingido
41.146
120.011
6199
Parâmetros do algoritmo
npart
50
alpha
0.8
beta
0.1
gamma
0.1
Risco residual
Algoritmo de atração
Critérios de paragem
Qualidade (%)
Tempo (s)
Iterações
Limite
100
120
999999
Atingido
41.2333
120
156251
Parâmetros do algoritmo
probmed
0.37
Risco residual

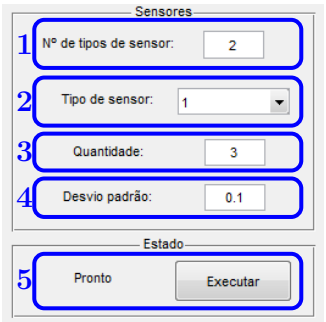


1. Resolução da grelha de pontos quadrada a sobrepor à zona de interesse. Se a resolução for n, a grelha terá $n \times n$ pontos.

2. Escolha dos algoritmos que vão ser executados.

3. Criação da zona de interesse. Aqui encontra-se a visualização da criticidade, vulnerabilidade e risco resultante da combinação de ambas. Tanto a criticidade como a vulnerabilidade podem ser geradas aleatoriamente através dos botões correspondentes. O código de cores utilizado é o seguinte:





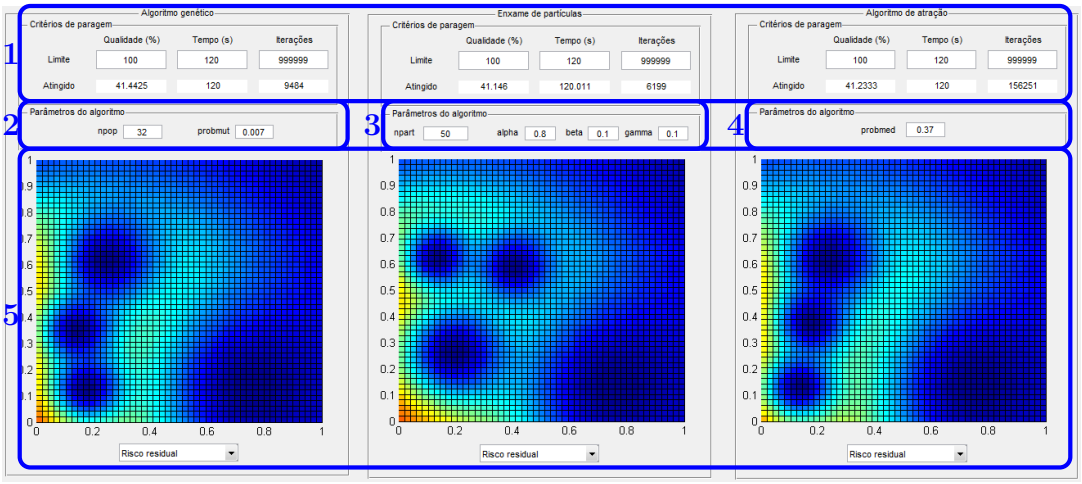
1. Número de tipos de sensor.

2. Tipo de sensor a editar em 3 e 4.

3. Número de sensores do tipo escolhido em 2.

4. Desvio padrão (> 0) do tipo de sensor escolhido em 2.

5. Estado do sistema e botão de execução.



1. Critérios de paragem (Qualidade da solução, tempo de execução e iterações) para cada um dos algoritmos, bem como os valores atingidos por cada critério no final da execução.

2. Parâmetros do algoritmo genético, tamanho da população (npop) e probabilidade de mutação (probmut).

3. Parâmetros do enxame de partículas, número de partículas (npart) e pesos das componentes vetoriais (alpha,beta e gamma).

4. Parâmetros do algoritmo de atração, probabilidade de mutação de um sensor com utilidade acima da média (probmed).

5. Visualização dos resultados obtidos por cada algoritmo. É possível mudar o tipo de visualização a partir da lista de opções por baixo de cada eixo.

Apêndice B

Funções e pseudo-códigos

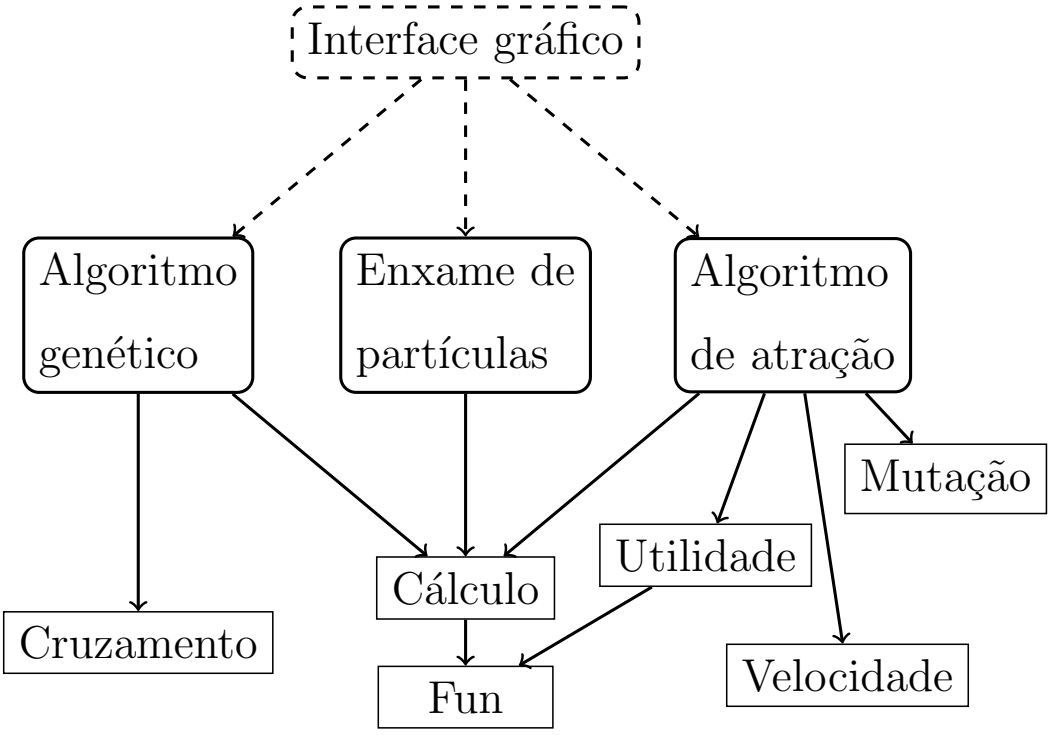


Figura B.1: Hierarquia de funções

B.1 Glossário

Nome	Descrição
$lado$	Resolução da grelha de pontos ($lado \times lado$)
$passo$	Distância mínima entre dois pontos da grelha ($\frac{1}{lado-1}$)
$nsen$	Número total de sensores
$ntipos$	Número de tipos de sensor
$rand$	Número real aleatório entre 0 e 1
$qualalvo$	Qualidade limite (critério de paragem)
$tempoalvo$	Tempo limite (critério de paragem)
$iteralvo$	Iteração limite (critério de paragem)
$Desvpad$	Vetor com o desvio padrão de cada sensor ($1 \times nsen$)
SOL	Matriz solução com as coordenadas dos sensores ($2 \times nsen$)
$SOLFINAL$	Matriz com a melhor solução até ao momento ($2 \times nsen$)
$CRIT$	Matriz com a criticidade associada a cada ponto ($lado \times lado$)
$VULN$	Matriz com a vulnerabilidade associada a cada ponto ($lado \times lado$)
$RISCO$	Matriz com o risco associado a cada ponto ($lado \times lado$)
$npop$	Dimensão da população do algoritmo genético
$ncruz$	Nº de soluções originadas por cruzamento por iteração ($\frac{3}{4} \times npop$)
$nrand$	Nº de soluções aleatórias acrescentar por iteração ($\frac{1}{4} \times npop$)
$probmut$	Probabilidade de mutação
$Qual$	Vetor com a qualidade de cada solução da população ($1 \times npop$)
POP	População de soluções do algoritmo genético ($2 \times nsen \times npop$)
$npart$	Dimensão da população do enxame de partículas (nº de partículas)
$Qual$	Vetor com a qualidade de cada solução da população ($1 \times npart$)
$PART$	População de partículas ($2 \times nsen \times npart$)
V	Matriz com as velocidades de cada partícula ($2 \times nsen \times npart$)
$MELPART$	As melhores soluções encontradas por cada partícula ($2 \times nsen \times npart$)
$probmed$	Probabilidade de mutação para sensor com utilidade acima da média
UTI	Utilidade associada a cada tipo de sensor ($lado \times lado \times ntipos$)
V	Velocidade da solução ($2 \times nsen$)

Tabela B.1: Glossário

B.2 Algoritmo genético

Algoritmo 9: Pseudo-código do algoritmo genético

input : qualalvo, tempoalvo, iteralvo, npop, probmut

$r_0 \leftarrow$ somatório do risco inicial de todos os pontos da grelha

POP $\leftarrow$ população com soluções iniciais aleatórias

para $s = 1, \ldots,$ npop **faça**

  SOL $\leftarrow$ s-ésima solução da população

  (RISCO, Qual[s]) $\leftarrow$ `Cálculo(SOL )`

qual $\leftarrow max($Qual$)$

SOLFINAL $\leftarrow$ solução com qualidade qual

[iter, tempo] $\leftarrow$ [0,tempo de execução até ao momento]

enquanto qual $<$ qualalvo & tempo $<$ tempoalvo & iter $+ 1 \leq$ iteralvo **faça**

  CRUZ $\leftarrow$ `Cruzamento(POP,Qual)`, cruzamento das soluções em POP

  para $n = 1, \ldots,$ ncruz **faça**

    para $s = 1, \ldots,$ nsen **faça**

      se $rand <$ probmut **então**

        CRUZ[*1,s,n*] $\leftarrow rand$

        CRUZ[*2,s,n*] $\leftarrow rand$

  RAND $\leftarrow$ soluções aleatórias adicionais, sendo a primeira substituída pela melhor solução encontrada até ao momento (elitismo)

  POP $\leftarrow$ CRUZ $\cup$ RAND

  para $s = 1, \ldots,$ npop **faça**

    SOL $\leftarrow$ s-ésima solução da população

    (RISCO, Qual[s]) $\leftarrow$ `Cálculo(SOL )`

  se $max($Qual$) >$ qual **então**

    qual $\leftarrow max($Qual$)$

    SOLFINAL $\leftarrow$ solução com qualidade qual

  [iter, tempo] $\leftarrow$ [iter $+ 1$,tempo de execução até ao momento]

output : SOLFINAL, qual

B.3 Enxame de partículas

Algoritmo 10: Pseudo-código do enxame de partículas

input : qualalvo, tempoalvo, iteralvo, npart, α, β, γ

$r_0 \leftarrow$ somatório do risco inicial de todos os pontos da grelha

PART $\leftarrow$ população com soluções iniciais aleatórias (partículas)

MELPART $\leftarrow$ PART

V $\leftarrow$ inicializar o vetor velocidade

para $p = 1, \ldots,$ npart **faça**

    SOL $\leftarrow$ s-ésima solução da população

    (RISCO, Qual[s]) $\leftarrow$ `Cálculo(SOL )`

[Melqual, qual] $\leftarrow$ [Qual, max(Melqual)]

SOLFINAL $\leftarrow$ solução com qualidade max(Melqual)

[iter, tempo] $\leftarrow$ [0,tempo de execução até ao momento]

enquanto qual $<$ qualalvo & tempo $<$ tempoalvo & iter $+ 1 \leq$ iteralvo **faça**

    $[brand, grand] \leftarrow [rand, rand]$

    para $p = 1, \ldots,$ npart **faça**

        para $s = 1, \ldots,$ nsen **faça**

            V[*1,s,p*] $\leftarrow \alpha \times$ V[*1,s,p*] $+ \beta \times brand \times$ (MELPART[*1,s,p*] $-$ PART[*1,s,p*]) $+ \gamma \times grand \times$ (SOLFINAL[*1,s*] $-$ PART[*1,s,p*])

            V[*2,s,p*] $\leftarrow \alpha \times$ V[*2,s,p*] $+ \beta \times brand \times$ (MELPART[*2,s,p*] $-$ PART[*2,s,p*]) $+ \gamma \times grand \times$ (SOLFINAL[*2,s*] $-$ PART[*2,s,p*])

    PART $\leftarrow$ PART $+$ V

    para $p = 1, \ldots,$ npart **faça**

        SOL $\leftarrow$ s-ésima solução da população

        (RISCO, Qual[s]) $\leftarrow$ `Cálculo(SOL )`

        se Qual[p] $>$ Melqual[p] **então**

            Melqual[p] $\leftarrow$ Qual[p]

            MELPART[p]) $\leftarrow$ SOL

    qual $\leftarrow$ max(Melqual)

    SOLFINAL $\leftarrow$ solução com qualidade qual

    [iter, tempo] $\leftarrow$ [iter $+ 1$,tempo de execução até ao momento]

output : SOLFINAL, qual

B.4 Algoritmo de atração

Algoritmo 11: Pseudo-código do algoritmo de atração

input : qualalvo, tempoalvo, iteralvo, probmed

$r_0 \leftarrow$ somatório do risco inicial de todos os pontos da grelha

UTI $\leftarrow$ `Utilidade(`CRIT,VULN `)`

SOL $\leftarrow \emptyset$

SOL $\leftarrow$ `Mutação(`SOL,UTI,probmed `)`

(RISCO, qual) $\leftarrow$ `Cálculo(`SOL `)`

SOLFINAL $\leftarrow SOL$

[iter, tempo] $\leftarrow$ [0,tempo de execução até ao momento]

enquanto qual $<$ qualalvo $\&$ tempo $<$ tempoalvo $\&$ iter $+ 1 \leq$ iteralvo **faça**

- V $\leftarrow$ `Velocidade(`SOL,RISCO `)`
- SOL $\leftarrow$ SOL $+$ V
- SOL $\leftarrow$ `Mutação(`SOL,UTI,probmed `)`
- (RISCO, novaqual) $\leftarrow$ `Cálculo(`SOL `)`
- **se** novaqual $>$ qual **então**
  - qual $\leftarrow$ novaqual
  - SOLFINAL $\leftarrow$ SOL
- [iter, tempo] $\leftarrow$ [iter $+ 1$,tempo de execução até ao momento]

output : SOLFINAL, qual

B.5 Função cruzamento

Algoritmo 12: Pseudo-código da função cruzamento

input : POP, Qual

ncasais $\leftarrow \frac{\text{ncruz}}{2}$

para $n = 1, \ldots,$ npop **faça**

  Qualcum[n] $\leftarrow \sum_{i=1}^{n}$ Qual[i]

para $c = 1, \ldots,$ ncasais **faça**

  $r \leftarrow$ número aleatório entre 0 e max(Qualcum)

  $n1 \leftarrow 1$

  enquanto $r \geq$ Qualcum[$n1$] **faça**

    $n1 \leftarrow n1 + 1$

  $n2 \leftarrow$ número inteiro aleatório entre 1 e npop, diferente de $n1$

  $p \leftarrow$ número inteiro aleatório entre 1 e nsen $- 1$

  para $n = 1, \ldots,$ ncasais **faça**

    para $s = 1, \ldots,$ nsen **faça**

      se $s > p$ **então**

        CRUZ[*1,s,2(n-1)*] $\leftarrow$ POP[*1,s,n1*]

        CRUZ[*2,s,2(n-1)*] $\leftarrow$ POP[*2,s,n1*]

        CRUZ[*1,s,2(n-1)+1*] $\leftarrow$ POP[*1,s,n2*]

        CRUZ[*2,s,2(n-1)+1*] $\leftarrow$ POP[*2,s,n2*]

      senão

        CRUZ[*1,s,2(n-1)*] $\leftarrow$ POP[*1,s,n2*]

        CRUZ[*2,s,2(n-1)*] $\leftarrow$ POP[*2,s,n2*]

        CRUZ[*1,s,2(n-1)+1*] $\leftarrow$ POP[*1,s,n1*]

        CRUZ[*2,s,2(n-1)+1*] $\leftarrow$ POP[*2,s,n1*]

output : CRUZ

B.6 Função cálculo

Algoritmo 13: Pseudo-código da função cálculo

input : SOL, CRIT, VULN

$r_0 \leftarrow$ somatório do risco inicial de todos os pontos da grelha

para $s = 1, \ldots,$ nsen **faça**

  para $i, j = 1, \ldots,$ lado **faça**

    $(x, y) \leftarrow ([i-1] \times \text{passo}, [j-1] \times \text{passo})$

    $\text{dist} \leftarrow \sqrt{(\text{SOL}[1,s] - x)^2 + (\text{SOL}[2,s] - y)^2}$

    $\text{COBER}[i,j,s] \leftarrow \texttt{Fun}(\text{dist}, \text{Desvpad}[s])$

TEMP $\leftarrow$ Matriz de 1s (lado $\times$ lado)

para $s = 1, \ldots,$ nsen **faça**

  para $i, j = 1, \ldots,$ lado **faça**

    $\text{TEMP}[i,j] \leftarrow \text{TEMP}[i,j] \times (1 - \text{COBER}[i,j,s])$

para $i, j = 1, \ldots,$ lado **faça**

  $\text{RISCO}[i,j] \leftarrow \text{CRIT}[i,j] \times \text{VULN}[i,j] \times \text{TEMP}[i,j]$

$\text{r} \leftarrow \sum_{i=1}^{\text{lado}} \sum_{j=1}^{\text{lado}} \text{RISCO}[i,j]$

$\text{qualidade} \leftarrow 100 \times \frac{(\text{r}_0 - \text{r})}{\text{r}_0}$

output : RISCO, qualidade

B.7 Função fun

Algoritmo 14: Pseudo-código da função fun

input : x, σ

$\text{y} \leftarrow \texttt{normpdf}(\text{x}, 0, \sigma) / \texttt{normpdf}(0, 0, \sigma)$

output: y

NOTA: $\texttt{normpdf}\ (x, \mu, \sigma) = \frac{1}{\sigma \times \sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$

B.8 Função utilidade

Algoritmo 15: Pseudo-código da função utilidade

input : CRIT, VULN

$r_0 \leftarrow$ somatório do risco inicial de todos os pontos da grelha

para $t = 1, \ldots,$ ntipos **faça**

  para $i, j = 1, \ldots,$ lado **faça**

    soma $\leftarrow 0$

    para $k, l = 1, \ldots,$ lado **faça**

      dist $\leftarrow \sqrt{([i-k] \times \text{passo})^2 + ([j-l] \times \text{passo})^2}$

      desvpad $\leftarrow$ desvio padrão associado ao tipo de sensor t

      soma $\leftarrow$ soma $+$ CRIT$[k,l] \times$ VULN$[k,l] \times (1 -$ `Fun(dist,desvpad )`$)$

    UTI$[i,j,t] \leftarrow r_0 -$ soma

output : UTI

B.9 Função velocidade

Algoritmo 16: Pseudo-código da função velocidade

input : SOL, RISCO

riscototal $\leftarrow \sum_{i=1}^{lado} \sum_{j=1}^{lado}$ RISCO[i,j]

para $i, j = 1, \ldots,$ lado **faça**

  PRISCO[i,j] $\leftarrow \frac{\text{RISCO}[i,j]}{riscototal}$

para $s = 1, \ldots,$ nsen **faça**

  para $i, j = 1, \ldots,$ lado **faça**

    $(x, y) \leftarrow ([i-1] \times \text{passo}, [j-1] \times \text{passo})$

    dist $\leftarrow \sqrt{(\text{SOL}[1,s] - x)^2 + (\text{SOL}[2,s] - y)^2}$

    se dist $<$ passo **então**

      PDIST[i,j,s] $\leftarrow \frac{1}{\text{passo}}$

    senão

      PDIST[i,j,s] $\leftarrow \frac{1}{\text{dist}}$

para $s = 1, \ldots,$ nsen **faça**

  soma $\leftarrow \sum_{i=1}^{lado} \sum_{j=1}^{lado}$ PDIST[i,j,s]

  para $i, j = 1, \ldots,$ lado **faça**

    PDIST[i,j,s] $\leftarrow \frac{\text{PDIST}[i,j,s]}{\text{soma}}$

para $s = 1, \ldots,$ nsen **faça**

  para $i, j = 1, \ldots,$ lado **faça**

    $r \leftarrow rand$

    V[1,s] $\leftarrow$ V[1,s] $+ r \times$ PRISCO[i,j] $\times$ PDIST[i,j,s] $\times (c_i -$ SOL[1,s]$)$

    V[2,s] $\leftarrow$ V[2,s] $+ r \times$ PRISCO[i,j] $\times$ PDIST[i,j,s] $\times (c_j -$ SOL[2,s]$)$

output : V

B.10 Função mutação

Algoritmo 17: Pseudo-código da função mutação

input : SOL, UTI, probmed

para $s = 1, \ldots,$ nsen **faça**

- $t \leftarrow$ tipo de sensor associado ao sensor s
- **se** SOL $= \emptyset$ **então**
  - prob $\leftarrow 1$
- **senão**
  - $(i, j) \leftarrow$ posição da matriz UTI associada ao ponto da grelha mais próximo do sensor s
  - utimedia $\leftarrow \frac{\sum_{i=1}^{\text{lado}} \sum_{j=1}^{\text{lado}} \text{UTI}[i,j,t]}{\text{lado} \times \text{lado}}$
  - **se** UTI$[i,j,t] <$ utimedia **então**
    - prob $\leftarrow 1 -$ UTI$[i,j,t] \times (\frac{1-\text{probmed}}{\text{utimedia}})$
  - **senão**
    - prob $\leftarrow$ probmed
- $r \leftarrow rand$
- **se** $r <$ prob **então**
  - utisoma $\leftarrow \sum_{i=1}^{\text{lado}} \sum_{j=1}^{\text{lado}}$ UTI$[i,j,t]$
  - $r \leftarrow$ número aleatório entre 0 e utisoma
  - soma $\leftarrow 0$
  - flag $\leftarrow 0$
  - **para** $i, j = 1, \ldots,$ lado **faça**
    - soma $\leftarrow$ soma $+$ UTI$[i,j,t]$
    - **se** flag $= 0$ *e soma* $> r$ **então**
      - $(x, y) \leftarrow (i, j)$
      - flag $\leftarrow 1$
  - NOVASOL$[1,s] \leftarrow (x - 1) \times$ passo
  - NOVASOL$[2,s] \leftarrow (y - 1) \times$ passo

output : SOL