**Manual de computação evolutiva e metaheurística**

**Autor(es):** Cunha, António Gaspar, coord.; Takahashi, Ricardo, coord.; Antunes, Carlos Henggeler, coord.

**Publicado por:** Imprensa da Universidade de Coimbra; Editora da Universidade Federal de Minas Gerais

**URL**

**persistente:** URI:http://hdl.handle.net/10316.2/5655

**DOI:** DOI:http://dx.doi.org/10.14195/978-989-26-0583-8

**Accessed :** 13-Apr-2020 12:10:04

A navegação consulta e descarregamento dos títulos inseridos nas Bibliotecas Digitais UC Digitalis, UC Pombalina e UC Impactum, pressupõem a aceitação plena e sem reservas dos Termos e Condições de Uso destas Bibliotecas Digitais, disponíveis em https://digitalis.uc.pt/pt-pt/termos.

Conforme exposto nos referidos Termos e Condições de Uso, o descarregamento de títulos de acesso restrito requer uma licença válida de autorização devendo o utilizador aceder ao(s) documento(s) a partir de um endereço de IP da instituição detentora da supramencionada licença.

Ao utilizador é apenas permitido o descarregamento para uso pessoal, pelo que o emprego do(s) título(s) descarregado(s) para outro fim, designadamente comercial, carece de autorização do respetivo autor ou editor da obra.

Na medida em que todas as obras da UC Digitalis se encontram protegidas pelo Código do Direito de Autor e Direitos Conexos e demais legislação aplicável, toda a cópia, parcial ou total, deste documento, nos casos em que é legalmente admitida, deverá conter ou fazer-se acompanhar por este aviso.



pombalina.uc.pt

digitalis.uc.pt

COMPUTAÇÃEVOLUTIVE MET HEURÍTICANTÓNIO GASPAR-CUNHA

MANUAL DE

RICARDO TAKAHASHI

CARLOS HENGGELER ANTUNES

COORDENADORES

IMPRENSA DA

UNIVERSIDADE

DE COIMBRA

COIMBRA

UNIVERSITY

PRESS

(Página deixada propositadamente em branco)

1

E N S I N O

2 **CO-EDIÇÃO**

Imprensa da Universidade de Coimbra

Email: imprensauc@ci.uc.pt

URL: http://www.uc.pt/imprensa\_uc

Vendas online http://www.livrariadaimprensa.com

Editora da Universidade Federal de Minas Gerais

URL: http://www.editoraufmg.com.br/

**CONCEPÇÃO GRÁFICA**

António Barros

**INFOGRAFIA DA CAPA**

Carlos Costa

**EXECUÇÃO GRÁFICA**

Sersilito • Maia

**ISBN**

978-989-26-0150-2 (IUC)

978-85-7041-950-7 (EDITORAufmg)

**ISBN Digital**

978-989-26-0583-8

**DOI**

http://dx.doi.org/10.14195/978-989-26-0583-8

**DEPÓSITO LEGAL**

344533/12

A edição desta obra contou com o apoio da Universidade do Minho e do Instituto de Engenharia de Sistemas e Computadores de Coimbra - INESC Coimbra.

**©JUNHO 2012, IMPRENSA DA UNIVERSIDADE DE COIMBRA**

3

COMPUTAÇÃO

MANUAL DE

EVOLUTIVA

ANTÓNIO GASPAR-CUNHA RICARDO TAKAHASHI

CARLOS HENGGELER ANTUNES COORDENADORES

E META HEURÍS TICA

IMPRENSA DA

IMPRENSA DA

UNIVERSIDADE

UNIVERSIDADE

DE COIMBRA

DE COIMBRA

COIMBRA

COIMBRA

UNIVERSITY

UNIVERSITY

PRESS

PRESS

(Página deixada propositadamente em branco)

v

Sum´ario

Pref´acio x 1 Introdu¸c˜ao 1

1. Otimiza¸c˜ao 1 2. Heur´ıstica 8 3. Computa¸c˜ao Evolutiva 13 4. Premissa: Localidade Fraca 16 5. Conclus˜oes 20

I M´etodos Bio-Inspirados

2 Algoritmos Gen´eticos 25

1. A Inspira¸c˜ao Biol´ogica 25 2. Estrutura de um Algoritmo Gen´etico 26 3. Exemplo: Aplica¸c˜ao de um AG ao Problema da Mochila 28 4. Propriedades dos Algoritmos Gen´eticos 36 5. Extens˜oes ao Algoritmo Gen´etico Simples 38 6. Aplica¸c˜oes Pr´aticas 45 7. Conclus˜oes 47

3 Estrat´egias Evolutivas 49

1. Optimiza¸c˜ao em Espa¸cos Cont´ınuos 50 2. Caracter´ısticas Gerais 51 3. Nomenclatura 52 4. Estrat´egia Evolutiva (1+1) 54 5. Estrat´egias Evolutivas Multimembros 58 6. Tratamento das Restri¸c˜oes 63 7. Estrat´egias Evolutivas Avan¸cadas 65

4 Programa¸c˜ao Gen´etica 67

1. Descri¸c˜ao da Programa¸c˜ao Gen´etica 70 2. Algoritmo Prot´otipo 79 3. Exemplo de Aplica¸c˜ao: Regress˜ao Simb´olica 81 4. Conclus˜oes 85

5 Colˆonia de Formigas 87

1. Aprendendo com as Formigas Reais 87 2. Construindo Formigas Artificiais 88 3. Otimiza¸c˜ao por Colˆonia de Formigas 90 4. Hist´orico dos Algoritmos ACO 93 5. ACO Aplicada a Problemas com Restri¸c˜oes 96 6. ACO Aplicada a Problemas Multiobjetivo 100 7. ACO Aplicada a Problemas com Vari´aveis Cont´ınuas 104 8. Conclus˜oes 105

6 Algoritmos Imunoinspirados 107

1. O Sistema Imunol´ogico 108 2. Engenharia Imunol´ogica 116 3. Algoritmos Imunol´ogicos 121 4. Exemplo de Aplica¸c˜ao 128 5. Sistemas Imunol´ogicos Artificiais e Computa¸c˜ao Evolutiva 137

II M´etodos N˜ao Bio-Inspirados

7 Evolu¸c˜ao Diferencial 141

1. Introdu¸c˜ao 141 2. Evolu¸c˜ao Diferencial 143 3. Comportamento da Muta¸c˜ao Diferencial 145 4. Aspectos Avan¸cados 152 5. Conclus˜oes 160

8 Recozimento Simulado 163

1. Implementa¸c˜ao do Recozimento Simulado 165 2. Aplica¸c˜oes 167 3. Recozimento Simulado Multiobjetivo 171 4. Conclus˜oes 174

9 Busca Tabu 177

1. Funcionamento de um algoritmo BT 178 2. O Algoritmo Busca Tabu 183 3. Exemplos de regras de proibi¸c˜ao 183 4. Lista de Candidatos 186 5. Implementa¸c˜ao eficiente da lista tabu 187 6. Tamanho da lista tabu 189 7. Crit´erios de aspira¸c˜ao 191 8. Mem´oria de Longo Prazo 192 9. Oscila¸c˜ao estrat´egica 200

10 GRASP: Busca Gulosa Aleatorizada e Adaptativa 203

1. Introdu¸c˜ao 204 2. Esquemas de Busca Local 204 3. Procedimentos de Busca Gulosos Aleatorizados Adaptativos 205 4. M´etodo de Constru¸c˜ao Guloso Aleat´orio 206 5. Hibridiza¸c˜oes com Religamento de Caminhos 210 6. Conclus˜oes 213

11 Algoritmos de Estima¸c˜ao de Distribui¸c˜ao 215

1. Introdu¸c˜ao 216 2. Blocos Construtivos 216 3. Algoritmos de Estima¸c˜ao de Distribui¸c˜ao 218 4. Limita¸c˜oes dos AEDs 223 5. Modelos Gr´aficos Probabil´ısticos 224 6. Aplica¸c˜oes de AEDs 228 7. Novas Perspectivas em AEDs 234 8. Material Adicional 235 9. Conclus˜oes 236

12 Pesquisa Local Iterativa e em Vizinhan¸ca Vari´avel 237

1. Fundamentos de Pesquisa Local Iterativa 238 2. Componentes de Pesquisa Local Iterativa 240 3. Um caso de estudo - O Problema do Caixeiro Viajante 242 4. Pesquisa de Vizinhan¸ca Vari´avel 244 5. Conclus˜oes 245

III T´opicos Avan¸cados

13 Distˆancias Generalizadas: Espa¸cos Combinat´orios 249

1. Panorama de Aptid˜ao 250 2. Distˆancia em Problemas Combinat´orios 255 3. Operadores Topol´ogicos / Geom´etricos 262 4. Otimiza¸c˜ao de Redes em Arvore ´ 264 5. Estudo de Caso 270

14 Tratamento de Restri¸c˜oes em Algoritmos Evolutivos 275

1. M´etodos de Penalidade 277 2. M´etodos de Preserva¸c˜ao da Factibilidade 282 3. M´etodos Baseados em Representa¸c˜oes e Decodificadores 283 4. M´etodos Baseados em Otimiza¸c˜ao Multiobjetivo 285 5. M´etodos H´ıbridos 286 6. Tratamento de Restri¸c˜oes em Problemas Multiobjetivo 288 7. Conclus˜oes 290

15 Otimiza¸c˜ao em Ambientes Incertos 291

1. Introdu¸c˜ao 292 2. A incerteza presente em problemas reais de otimiza¸c˜ao 292 3. Problemas Ruidosos 295 4. Problemas Robustos 300 5. Aproxima¸c˜ao de Fun¸c˜ao 304 6. Problemas Dinˆamicos 310 7. Conclus˜oes 321

16 Tomada de Decis˜ao em Ambiente Multiobjectivo 323

1. Programa¸c˜ao Linear Multiobjectivo - Formula¸c˜ao e Conceitos Fundamentais 326 2. Programa¸c˜ao Linear Inteira e Programa¸c˜ao N˜ao Linear Multiobjectivo 329 3. Processos de C´alculo de Solu¸c˜oes Eficientes em Programa¸c˜ao Linear Multiobjectivo 331 4. M´etodos multiobjectivo de apoio `a decis˜ao 342 5. O M´etodo TRIMAP 347 6. Integra¸c˜ao entre M´etodos em Sistemas de Apoio `a Decis˜ao 353 7. Breve referˆencia a estudos de aplica¸c˜ao 355

17 Algoritmos Evolutivos Multi-Objectivo 357

1. Otimiza¸c˜ao Multiobjetivo 361 2. Algoritmos Evolutivos Multiobjetivo 367 3. Desenvolvimentos Recentes e Futuros 375

18 Algoritmos Gen´eticos em Problemas de Classifica¸c˜ao 381 1. Ajuste de Parˆametros de Classificadores por AGs 383

2. Indu¸c˜ao de Arvores de Decis˜ ´ ao com Algoritmos Gen´eticos 388 3. Decomposi¸c˜ao de Problemas Multiclasses 398 4. Conclus˜oes 405

Referˆencias Bibliogr´aficas 407 ´Indice Remissivo 447

Pref´acio

As t´ecnicas computacionais que s˜ao hoje denominadas por Computa¸c˜ao Evolutiva e por Me taheur´ısticas se desenvolveram, de maneira relativamente independente, durante os ´ultimos 40 anos do s´eculo XX, no seio de duas comunidades cient´ıficas que mantiveram relativamente pouco contato ao longo desse per´ıodo. Durante esse tempo, ambos os conjuntos de t´ecnicas se consolidaram, sendo hoje reconhecidos como parte integrante do repert´orio fundamental de ferramentas da Computa¸c˜ao e da Engenharia que possibilitam a s´ıntese de muitos dos sistemas tecnol´ogicos hoje existentes. Apenas no decorrer da ´ultima d´ecada do s´eculo XX se formou, nas respectivas comunidades cient´ıficas, uma consciˆencia das conex˜oes existentes entre esses dois corpos de conhecimento, que partilham muitos dos seus princ´ıpios e fundamentos. Em 2008, um grupo de colegas Portugueses e Brasileiros que tra balham nas ´areas de algoritmos evolutivos e metaheur´ısticas decidiu promover a organiza¸c˜ao de uma Escola Luso-Brasileira de Computa¸c˜ao Evolutiva (ELBCE). O primeiro evento ocorreu em Outubro de 2009 na Universidade Federal de Minas Gerais, em Belo Horizonte, e a segunda escola em Junho de 2010 na Universidade do Minho, em Guimar˜aes. O terceiro evento ocorreu na Universidade de S˜ao Paulo, no Campus I da cidade de S˜ao Carlos, em Abril de 2012. Os temas cobertos nesses eventos apresentaram um panorama abrangente dessas ´areas do conhecimento. Al´em disso, ficou claro o efeito de uma apresenta¸c˜ao articulada desses temas para potencializar a compreens˜ao dos seus aspectos mais fundamentais.

O grande interesse manifestado pelos pesquisadores envolvidos na leciona¸c˜ao das aulas, bem como o grande n´umero de estudantes que aderiu `as trˆes edi¸c˜oes do evento, sugeriram a pertinˆencia da escrita deste livro que inclui um vasto leque de temas relacionados com a computa¸c˜ao evolutiva e metaheur´ısticas. O presente livro foi escrito com o objetivo de constituir uma obra de referˆencia em L´ıngua Portuguesa, abrangendo os n´ıveis de gradua¸c˜ao e p´os-gradua¸c˜ao do nosso ensino universit´ario e polit´ecnico. Al´em disso, mesmo na literatura t´ecnica em geral, n˜ao temos conhecimento de obra similar, escrita em qualquer l´ıngua, que proponha tal recorte tem´atico. Gostar´ıamos de agradecer aos autores dos v´arios cap´ıtulos pela rapidez na sua resposta `a solicita¸c˜ao feita bem como pelo rigor cient´ıfico colocado na sua escrita. Gostar´ıamos de agradecer tamb´em aos revisores an´onimos que permitiram melhorar a qualidade deste trabalho e `a Imprensa da Universidade de Coimbra que se dispˆos a investir neste projeto.

Abril de 2012

Ant´onio Gaspar-Cunha

Ricardo H. C. Takahashi

Carlos Henggeler Antunes

1

CAP´ITULO 1

Introdu¸c˜ao

*Ricardo H. C. Takahashi ∗ Ant´onio Gaspar-Cunha ∗∗*

*∗Departamento de Matem´atica*

*Universidade Federal de Minas Gerais*

*∗∗Instituto de Pol´ımeros e Comp´ositos / I3N*

*Universidade do Minho*

Este livro trata dos temas da Computa¸c˜ao Evolutiva e da Metaheur´ıstica, considerando principalmente o contexto da Otimiza¸c˜ao. Neste cap´ıtulo, uma breve discuss˜ao introdut´oria a esses trˆes assuntos ´e apresentada, procurando primeiro dar uma no¸c˜ao sobre o que ´e cada um desses trˆes campos cient´ıficos. A seguir, procura-se mostrar como esses assuntos se relacionam, e quais raz˜oes tˆem levado a uma tendˆencia de crescente convergˆencia entre eles. Ao longo da discuss˜ao, este cap´ıtulo procura apresentar um panorama dos problemas de fundo que ser˜ao abordados nos demais cap´ıtulos deste livro.

1. Otimiza¸c˜ao



A Otimiza¸c˜ao ´e o campo de conhecimentos cujas t´ecnicas visam determinar os extremos (m´aximos ou m´ınimos) de fun¸c˜oes, em dom´ınios determinados1. De maneira concreta, podemos pensar que uma

1 O leitor deve estar atento para o fato de que esta defini¸c˜ao pretende dar uma no¸c˜ao geral do que significa Otimiza¸c˜ao, mas n˜ao pretende conter, de maneira exaustiva, toda a diversidade e complexidade desse campo do conhecimento.

2 MANUAL DE COMPUTAC¸AO EVOLUTIVA E METAHEUR ˜ ´ISTICA

Figura 1.1: Representa¸c˜ao esquem´atica de uma antena de seis elementos. Na pr´atica, uma antena real corresponde a uma estrutura met´alica com geometria igual `a do diagrama.

fun¸c˜ao cujo extremo se quer descobrir representa algum fator de m´erito relacionado com um sistema que se deseja construir ou analisar. Se o sistema for, por exemplo, um dispositivo tecnol´ogico que se pretende construir, a fun¸c˜ao pode representar seu custo de constru¸c˜ao, ou sua eficiˆencia energ´etica, ou sua taxa de falhas esperada, ou outros fatores de m´erito ainda. Deve ficar claro qual ´e o motivo pelo qual se deseja determinar o extremo de uma fun¸c˜ao com tal significado: o extremo (m´aximo ou m´ınimo) representa a melhor maneira poss´ıvel de construir o sistema que est´a sendo projetado – no sentido de ser o projeto de m´ınimo custo, ou de m´axima eficiˆencia energ´etica, ou de m´ınima taxa de falhas, etc. O dom´ınio no qual ´e executada a otimiza¸c˜ao2, ou seja, a regi˜ao do espa¸co de vari´aveis da fun¸c˜ao na qual se procura determinar seu extremo, corresponde ao conjunto das alternativas diferentes dispon´ıveis para construir o sistema em quest˜ao. Desenvolveremos essas id´eias a partir de dois exemplos.

Exemplo: otimiza¸c˜ao com vari´aveis cont´ınuas

Como primeiro exemplo, considere-se o problema de projetar uma antena, como a da Figura 1.1. Projetar essa antena significa escolher os tamanhos dos seis elementos da antena e as cinco distˆancias que separam esses elementos entre si. S˜ao portanto onze valores a serem representados pelo vetor x,

o qual seria dado, nesse caso, por:

x =

⎡

⎢⎢⎢⎣*x*1

*x*2...

*x*11

⎤

⎥⎥⎥⎦*,* (1.1)

sendo as vari´aveis *x*1 a *x*6 correspondentes aos comprimentos dos elementos, e as vari´aveis *x*7 a *x*11 correspondentes `as distˆancias entre eles.

O dispositivo representado na Figura 1.1 pode ser uma antena para transmiss˜ao de sinais que deva enviar dados em uma dire¸c˜ao espec´ıfica. Um dos crit´erios de m´erito que frequentemente s˜ao empregados para avaliar o desempenho de antenas de transmiss˜ao desse tipo ´e a sua *raz˜ao frente costas*. Essa fun¸c˜ao representa a raz˜ao entre a energia que a antena transmite na dire¸c˜ao desejada e a energia transmitida para outras dire¸c˜oes do espa¸co. Essa fun¸c˜ao, portanto, deve assumir valores os maiores poss´ıveis, para que a antena apresente bom desempenho. O dom´ınio no qual ser´a realizada a otimiza¸c˜ao corresponde `a regi˜ao do espa¸co das vari´aveis de decis˜ao (as vari´aveis constituintes do vetor x) na qual essas vari´aveis assumem valores considerados pratic´aveis. No caso de uma antena dessas, para trabalhar em frequˆencias da ordem das centenas de MHz, as vari´aveis que representam os comprimentos dos elementos e as distˆancias entre esses elementos usualmente poder˜ao variar entre zero 2 A palavra *Otimiza¸c˜ao*, com inicial mai´uscula, diz respeito ao campo do conhecimento, enquanto a palavra *otimiza¸c˜ao*, com inicial min´uscula, diz respeito ao procedimento de determina¸c˜ao do extremo de uma fun¸c˜ao.

CAP´ITULO 1 

INTRODUC¸AO˜ 3

e algumas dezenas de cent´ımetros. Portanto, um processo de otimiza¸c˜ao que ir´a procurar a melhor antena dentre todas as poss´ıveis poder´a restringir sua busca, por exemplo, `a regi˜ao do espa¸co de vari´aveis na qual todas essas vari´aveis tenham valor menor que 0,5 metro – dessa forma estabelecendo se o dom´ınio no qual ser´a realizada a otimiza¸c˜ao. Dessa forma aparecem as seguintes express˜oes que definem esse dom´ınio:

0 *≤ x*1 *≤* 0*,* 5

0 *≤ x*2 *≤* 0*,* 5 ...

0 *≤ x*11 *≤* 0*,* 5

(1.2)

Al´em da condi¸c˜ao de que cada uma das vari´aveis que representam as dimens˜oes da antena deva ter valor menor que 1 metro, ´e poss´ıvel que haja outro tipo de condi¸c˜ao sobre as vari´aveis decorrente da situa¸c˜ao pr´atica em cada caso. Por exemplo, pode ser necess´ario que a antena, uma vez montada, caiba dentro de uma determinada caixa na qual ela dever´a ser transportada. Assim, al´em de se restringir o tamanho de cada elemento, ´e necess´ario tamb´em restringir a soma das distˆancias entre os elementos, de forma que a antena montada n˜ao exceda o tamanho m´aximo permitido, por exemplo 1,5 metro. Nesse caso, aparece uma outra desigualdade que deve ser obedecida, do tipo:

*x*7 + *x*8 + *x*9 + *x*10 + *x*11 *≤* 1*,* 5 (1.3)

Chamemos de *f*(x) a fun¸c˜ao que atribui a uma antena cujas vari´aveis de decis˜ao estejam especifica dos no vetor x o valor de sua raz˜ao frente-costas. Essa ´e a chamada *fun¸c˜ao-objetivo* do problema. Na pr´atica, os valores dessa fun¸c˜ao para os diferentes valores que o vetor x pode assumir ser˜ao calculados por meio de um programa que ir´a simular a antena, dados os parˆametros x, sendo esse programa baseado na teoria eletromagn´etica e em m´etodos num´ericos diversos de interpola¸c˜ao, integra¸c˜ao, etc. O problema de otimiza¸c˜ao ´e representado, de maneira sint´etica, por:

x*∗* = arg max *f*(x)

sujeito a:

⎧⎪⎪⎪⎪⎪⎨

*x*7 + *x*8 + *x*9 + *x*10 + *x*11 *≤* 1*,* 5 0 *≤ x*1 *≤* 0*,* 5

0 *≤ x*2 *≤* 0*,* 5

⎪⎪⎪⎪⎪⎩

...

0 *≤ x*11 *≤* 0*,* 5

(1.4)

Pode-se ler essa express˜ao da seguinte forma: x*∗*, que ´e o vetor de vari´aveis de decis˜ao que soluciona o problema, ´e o argumento que maximiza a fun¸c˜ao *f*(x) (ou seja, ´e o conjunto de valores das vari´aveis de decis˜ao que faz a fun¸c˜ao assumir seu m´aximo valor poss´ıvel), considerando os valores das vari´aveis de decis˜ao que satisfa¸cam ao conjunto de desigualdades `as quais o problema encontra-se sujeito. Essas desigualdades que especificam os valores permitidos para as vari´aveis de decis˜ao s˜ao denominadas *res tri¸c˜oes* do problema de otimiza¸c˜ao, e a regi˜ao do espa¸co de vari´aveis de decis˜ao cujos pontos satisfazem as restri¸c˜oes ´e chamada *regi˜ao fact´ıvel*, ou *conjunto fact´ıvel* do problema.

Algumas caracter´ısticas desse problema de otimiza¸c˜ao merecem ser destacadas:

*•* As vari´aveis deste problema s˜ao de natureza cont´ınua, ou seja, podem assumir quaisquer valores dentro de intervalos. A fun¸c˜ao-objetivo *f*(*·*) ´e uma fun¸c˜ao n˜ao-linear das vari´aveis presentes no vetor de vari´aveis de decis˜ao, x. Para cada instˆancia (ou seja, conjunto espec´ıfico de valores nas coordenadas) do vetor x, a fun¸c˜ao-objetivo assume um valor correspondente, provavelmente mantendo uma certa continuidade. Isso significa que valores muito pr´oximos de x devem produzir valores parecidos de *f*(x).

4 MANUAL DE COMPUTAC¸AO EVOLUTIVA E METAHEUR ˜ ´ISTICA

Figura 1.2: Problema de projeto de uma rede de comunica¸c˜oes. Na sub-figura da esquerda, s˜ao mostradas todas as poss´ıveis conex˜oes ligando pares de n´os. A sub-figura da direita mostra uma rede contendo um conjunto de conex˜oes que liga todos os n´os entre si, contendo ainda um caminho alternativo para o caso de falha do arco por onde passa o maior fluxo de dados. Esse arco ´e representado com tra¸co mais grosso que os demais.

*•* N˜ao encontra-se dispon´ıvel uma express˜ao expl´ıcita para a fun¸c˜ao *f*(x). O c´alculo dessa fun¸c˜ao ´e feito por uma simula¸c˜ao computacional das leis f´ısicas envolvidas no problema. Dessa forma, qualquer que seja o algoritmo de otimiza¸c˜ao que v´a tentar determinar um extremo dessa fun¸c˜ao, tal algoritmo n˜ao poder´a pressupor a utiliza¸c˜ao de uma express˜ao da fun¸c˜ao. A ´unica informa¸c˜ao que se pode empregar ´e a avalia¸c˜ao da fun¸c˜ao (ou seja o valor que ela assume) em determinados pontos do espa¸co de vari´aveis, para os quais se executou o algoritmo de c´alculo da fun¸c˜ao.

*•* Cada c´alculo de fun¸c˜ao-objetivo, para cada instˆancia do vetor de vari´aveis de decis˜ao, envolve um processamento computacional relativamente complexo, que provavelmente ir´a consumir tempo de execu¸c˜ao n˜ao desprez´ıvel. O processo de otimiza¸c˜ao, portanto, deve usar com parcimˆonia a possibilidade de avaliar essa fun¸c˜ao. Isso significa que est´a fora de quest˜ao, por exemplo, variar cada uma das vari´aveis de cent´ımetro em cent´ımetro, e avaliar a fun¸c˜ao-objetivo para cada uma das combina¸c˜oes resultantes, pois isso conduziria a executar o algoritmo de c´alculo da fun¸c˜ao nada menos que 5111 vezes3. Mesmo ap´os tal esfor¸co computacional gigantesco, o resultado ainda seria muito pobre sob o ponto de vista do projeto da antena, pois a precis˜ao necess´aria para que a qualidade da antena fosse garantida deveria ser da ordem dos d´ecimos de mil´ımetro, e n˜ao apenas dos cent´ımetros.

Exemplo: otimiza¸c˜ao com vari´aveis discretas

Para ilustrar o conceito de Otimiza¸c˜ao, conv´em apresentar mais um exemplo, com caracter´ısticas diferentes. Considere-se agora o problema de projeto de uma rede de comunica¸c˜ao de dados. O problema, representado na Figura 1.2, consiste em escolher, dentre todas as poss´ıveis conex˜oes que podem ligar um conjunto de n´os, aquelas conex˜oes a serem efetivamente constru´ıdas para constituir uma rede de comunica¸c˜oes. Pode-se imaginar um contexto em que tal sistema significasse, por exemplo, o conjunto de conex˜oes de fibra ´otica usado para interligar as redes de telefonia de cidades diversas (nesse caso, cada cidade seria um n´o).

O problema de projeto que se deseja resolver ´e definido como: projetar uma rede de baixo custo de constru¸c˜ao, que seja capaz de interligar todos os n´os, garantindo ainda que exista uma rota alternativa, em caso de falha da conex˜ao por onde passa o maior fluxo de dados.

3 Para se ter uma id´eia do que isto significa, caso cada avalia¸c˜ao de fun¸c˜ao objetivo durasse 1 segundo, a avalia¸c˜ao de todos esses valores demoraria aproximadamente 193 bilh˜oes de anos — muito mais do que qualquer fabricante de antenas estaria disposto a esperar...

CAP´ITULO 1 

INTRODUC¸AO˜ 5

Como no exemplo anterior, o problema de projeto pode ser formulado como um procedimento de otimiza¸c˜ao. Agora, cada vari´avel de decis˜ao corresponde `a inclus˜ao ou n˜ao de cada um dos *N* = *n*(*n −* 1)*/*2 arcos que podem ligar os *n* n´os. Adota-se por conven¸c˜ao que uma vari´avel *xi,j* , que representa a conex˜ao entre o n´o *i* e o n´o *j*, assume o valor 1 se tal conex˜ao for colocada, e o valor 0 se n˜ao for colocada na rede, e o vetor de vari´aveis de decis˜ao fica:

x =

⎡

⎢⎢⎢⎢⎢⎢⎢⎢⎢⎢⎢⎢⎢⎣*x*1*,*2

...

*x*1*,n*

*x*2*,*3

...

*x*2*,n*

...

*x*(*n−*1)*,n*

⎤

⎥⎥⎥⎥⎥⎥⎥⎥⎥⎥⎥⎥⎥⎦*.* (1.5)

O custo de instala¸c˜ao do arco que conecta o n´o *i* ao n´o *j* ´e definido como o valor *ci,j* . Dessa forma, a fun¸c˜ao-objetivo que representa o custo de constru¸c˜ao da rede ´e expresso por:

*f*(x) = *c*1*,*2*x*1*,*2 + *c*1*,*3*x*1*,*3 + *...* + *c*(*n−*1)*,nx*(*n−*1)*,n* = �*n i*=1

�*n*

*j*=1 *j�*=*i*

*cijxij ,* (1.6)

ou seja, ´e a soma dos custos dos arcos que forem instalados na rede (aqueles para os quais *xi,j* = 1). As restri¸c˜oes do problema, que como no caso anterior v˜ao dizer quais vetores de vari´aveis de decis˜ao s˜ao admiss´ıveis, desta vez n˜ao s˜ao representadas por express˜oes simples: s˜ao admiss´ıveis os vetores que representarem grafos conexos (redes em que exista um caminho entre todos os n´os) mesmo com a retirada do arco de maior fluxo. Essas restri¸c˜oes, portanto, n˜ao ser˜ao verificadas pela avalia¸c˜ao do valor de uma express˜ao matem´atica, mas sim pela execu¸c˜ao de algoritmos de teste que ir˜ao de alguma forma “percorrer” a rede, examinando o atendimento de tais condicionantes. Defina-se a fun¸c˜ao *g*(x) de maneira a expressar o resultado dessa verifica¸c˜ao, da seguinte forma:

*g*(x) =

⎧⎪⎪⎪⎪⎨

1 *,* se a rede descrita pelo vetor x ficar desconexa ap´os a retirada do arco de maior fluxo

⎪⎪⎪⎪⎩

*−*1 *,* se a rede descrita pelo vetor x permanecer conexa ap´os a retirada do arco de maior fluxo

*.* (1.7)

Com essa representa¸c˜ao para as restri¸c˜oes, o problema de otimiza¸c˜ao pode ser agora formulado da

seguinte maneira:

x*∗* = arg min *f*(x)

sujeito a: *g*(x) *<* 0 (1.8)

Novamente, x*∗* corresponde `a instˆancia do vetor de vari´aveis de decis˜ao que produz o valor extremo da fun¸c˜ao-objetivo (agora, ao contr´ario do exemplo anterior, o extremo procurado ´e o m´ınimo, e n˜ao o m´aximo), ou seja, ´e o *argumento* que minimiza a fun¸c˜ao *f*(x) garantindo o atendimento da restri¸c˜ao *g*(x) *<* 0.

O problema de otimiza¸c˜ao deste exemplo possui as seguintes caracter´ısticas:

*•* As vari´aveis de decis˜ao agora s˜ao de natureza discreta, assumindo apenas os valores 0 ou 1. Devido a isso, n˜ao ´e poss´ıvel utilizar nenhuma propriedade caracter´ıstica dos espa¸cos cont´ınuos (tais como a existˆencia de vizinhan¸cas arbitrariamente pequenas, a continuidade de fun¸c˜oes, o conceito de “dire¸c˜ao”, etc) no processo de otimiza¸c˜ao.

6 MANUAL DE COMPUTAC¸AO EVOLUTIVA E METAHEUR ˜ ´ISTICA

*•* O c´alculo do valor da fun¸c˜ao *f*(x), no caso deste exemplo, ´e bastante simples (trata-se de uma soma de valores). Isso significa que, no curso de um processo de otimiza¸c˜ao, provavelmente o tempo computacional despendido com a execu¸c˜ao de instru¸c˜oes do algoritmo de otimiza¸c˜ao ser´a dominante, comparado com o tempo despendido com as avalia¸c˜oes da fun¸c˜ao-objetivo. Note-se que esta situa¸c˜ao corresponde ao oposto do que ocorria no exemplo anterior. A avalia¸c˜ao da fun¸c˜ao de restri¸c˜ao agora tamb´em tende a ser muito mais dispendiosa que a avalia¸c˜ao da fun¸c˜ao objetivo.

*•* Embora exista um n´umero finito de poss´ıveis redes, normalmente tamb´em n˜ao existe a alter nativa de avaliar todas as possibilidades, de maneira exaustiva, como procedimento para deter mina¸c˜ao do ´otimo, pois o n´umero de alternativas ´e muito grande. Deve-se notar que, para *n* n´os, existem 2*N* diferentes redes (nem todas elas fact´ıveis). Assim, por exemplo, num problema com *n* = 15 cidades a serem interligadas, o n´umero de arcos ligando cidades seria de *N* = 105, e haveria 2105 redes cujas fun¸c˜oes objetivo e de restri¸c˜ao teriam de ser avaliadas4.

Formula¸c˜ao geral dos problemas de otimiza¸c˜ao

Grande parte dos problemas de otimiza¸c˜ao podem ser formulados como:

x*∗* = arg min *f*(x)

sujeito a:

⎧⎪⎪⎪⎨ ⎪⎪⎪⎩

*g*1(x) *≤* 0 *g*2(x) *≤* 0 ...

*gm*(x) *≤* 0

(1.9)

Neste livro, a maioria dos problemas a serem tratados ter˜ao essa estrutura. O problema ´e formulado, ent˜ao, como a quest˜ao de determinar o vetor x*∗* que minimiza o valor de uma fun¸c˜ao-objetivo *f*(x). Note-se que mesmo no caso de problemas em que a interpreta¸c˜ao natural da fun¸c˜ao-objetivo seria a de algo que se deseja maximizar, sempre ´e poss´ıvel fazer uma transforma¸c˜ao para uma fun¸c˜ao que se deseja minimizar (por exemplo multiplicando a fun¸c˜ao original por *−*1). Assim, neste livro adotaremos a conven¸c˜ao de sempre procurar a minimiza¸c˜ao de fun¸c˜oes. Tamb´em por conven¸c˜ao, as fun¸c˜oes de restri¸c˜ao *g*1(x) a *gm*(x) ser˜ao definidas como fun¸c˜oes que s˜ao satisfeitas quando s˜ao menores ou iguais a zero. Dessa forma, uma restri¸c˜ao do tipo 0 *≤ x*1 *≤* 0*.*5, que apareceu no primeiro exemplo, seria desdobrada em *g*1(x) = *−x*1 *≤* 0 e *g*2(x) = *x*1 *−* 0*.*5 *≤* 0.

Como j´a sugerem os exemplos anteriormente mostrados, o vetor de vari´aveis de decis˜ao pode ser constitu´ıdo por vari´aveis cont´ınuas ou por vari´aveis discretas, ou mesmo por uma combina¸c˜ao de vari´aveis dos dois tipos. A fun¸c˜ao-objetivo pode ser desde uma express˜ao aritm´etica simples envolvendo as vari´aveis de decis˜ao, at´e uma rela¸c˜ao de elevada complexidade cuja f´ormula expl´ıcita n˜ao se conhe¸ca, e cujo c´alculo seria feito a partir de algoritmos num´ericos possivelmente custosos computacionalmente. O mesmo vale para as fun¸c˜oes de restri¸c˜ao *g*1(x) a *gm*(x).

Por fim, cabe mencionar que grande parte dos estudos realizados no campo da otimiza¸c˜ao con sideram a fun¸c˜ao-objetivo *f*(x) como uma fun¸c˜ao de v´arias vari´aveis (as vari´aveis de decis˜ao) que retorna um ´unico valor real. No entanto, em situa¸c˜oes pr´aticas, ´e muito frequente que o projeto de um determinado sistema deva considerar mais de um objetivo. Por exemplo, ´e frequente que o projeto de um dispositivo considere a minimiza¸c˜ao do custo e a maximiza¸c˜ao de um ´ındice de desempenho 4 Supondo uma avalia¸c˜ao a cada microsegundo (ou um milh˜ao de avalia¸c˜oes por segundo), o tempo total a ser gasto

com o processo exaustivo seria de aproximadamente 1*,* 28 *×* 1024 anos (1*,* 28 trilh˜oes de trilh˜oes de anos) – tempo que um consumidor de servi¸cos telefˆonicos hoje em dia dificilmente estaria disposto a esperar para poder fazer liga¸c˜oes interurbanas.

CAP´ITULO 1 

INTRODUC¸AO˜ 7

como metas a serem alcan¸cadas. A generaliza¸c˜ao do problema de otimiza¸c˜ao para tais situa¸c˜oes ´e denominada *otimiza¸c˜ao multiobjetivo*. Este assunto ´e tratado no cap´ıtulo 17.

Otimiza¸c˜ao com convergˆencia garantida

V´arias t´ecnicas cl´assicas de otimiza¸c˜ao foram desenvolvidas para casos especiais do problema (1.9), de forma a garantidamente obter a solu¸c˜ao exata. No caso de vari´aveis cont´ınuas, alguns desses casos especiais importantes s˜ao:

*•* A fun¸c˜ao objetivo e as restri¸c˜oes s˜ao fun¸c˜oes afins (ou seja, s˜ao constru´ıdas como combina¸c˜oes lineares das vari´aveis de decis˜ao). T´ecnicas tais como o m´etodo SIMPLEX e os m´etodos de pontos interiores para programa¸c˜ao linear s˜ao capazes de resolver eficientemente problemas dessa classe, com determina¸c˜ao garantida do ´otimo exato. Essa classe de problemas chega a definir um ramo da Otimiza¸c˜ao, que ´e chamado de Programa¸c˜ao Linear.

*•* A fun¸c˜ao objetivo ´e diferenci´avel (possui gradiente em todos os pontos) e unimodal (as curvas de n´ıvel correspondentes a um dado valor *f*(x) = *α*1 delimitam regi˜oes conexas que contˆem inteiramente as curvas de n´ıvel *f*(x) = *α*2, sempre que *α*1 *> α*2). Se o problema n˜ao possui restri¸c˜oes (recaindo na chamada otimiza¸c˜ao irrestrita), os pontos de ´otimo desses problemas podem ser determidados de maneira eficiente e com garantia de convergˆencia por m´etodos tais como os quasi-Newton ou os de regi˜oes de confian¸ca. Caso haja restri¸c˜oes lineares, m´etodos de programa¸c˜ao quadr´atica sequencial podem ser empregados tamb´em com garantia de con vergˆencia. Esses problemas e seus respectivos m´etodos de solu¸c˜ao fazem parte de outro ramo da Otimiza¸c˜ao, que se denomina Programa¸c˜ao N˜ao-Linear.

*•* A fun¸c˜ao objetivo ´e convexa e todas as restri¸c˜oes s˜ao convexas. Nesse caso, m´etodos de pon tos interiores para programa¸c˜ao convexa s˜ao capazes de determinar de maneira exata o ´otimo do problema, de maneira computacionalmente eficiente. Tamb´em esses problemas com seus respectivos m´etodos de solu¸c˜ao definem uma ´area da Otimiza¸c˜ao: a Programa¸c˜ao Convexa.

Muitas vezes ´e poss´ıvel aplicar um desses m´etodos mesmo que a fun¸c˜ao objetivo e as restri¸c˜oes do problema n˜ao tenham as caracter´ısticas que fariam com que a aplica¸c˜ao do m´etodo levasse garan tidamente ao ´otimo exato. Por exemplo, se um m´etodo quasi-Newton for empregado para otimizar uma fun¸c˜ao-objetivo que n˜ao seja unimodal, seu resultado poder´a ser um ´otimo apenas local, e n˜ao mais necessariamente o ´otimo global do problema. Outras vezes, o m´etodo n˜ao pode ser aplicado a problemas fora da classe para a qual foi projetado; um exemplo disso ´e o m´etodo SIMPLEX, que s´o se aplica a problemas de Programa¸c˜ao Linear.

Quando todas as vari´aveis de decis˜ao s˜ao discretas, define-se uma sub-´area da Otimiza¸c˜ao de nominada Otimiza¸c˜ao Combinat´oria. No jarg˜ao peculiar dessa ´area, ´e chamado de *algoritmo* um procedimento computacional para resolver problemas dotado da garantia de que o ´otimo exato da fun¸c˜ao-objetivo ser´a atingido em um n´umero finito de passos5. J´a uma *aproxima¸c˜ao* ´e um procedi mento dotado da garantia de que, ap´os um n´umero finito de passos, ser´a obtida uma solu¸c˜ao cujo valor de fun¸c˜ao-objetivo ser´a “pr´oximo” do valor do ´otimo exato, n˜ao divergindo deste mais que uma quantidade teoricamente assegurada. Por exemplo, se o valor ´otimo da fun¸c˜ao-objetivo a ser minimi zada for igual a *J∗*, um procedimento de aproxima¸c˜ao poder´a gerar uma solu¸c˜ao cuja fun¸c˜ao objetivo seja garantidamente menor que (1 + *�*)*J∗*. Por fim, uma *heur´ıstica* ´e um procedimento de busca de “boas solu¸c˜oes” para o problema que n˜ao ´e dotado de garantias quanto `a qualidade relativa da solu¸c˜ao encontrada em compara¸c˜ao com o ´otimo exato.

5 No restante deste livro, entretanto, o termo *algoritmo* ser´a utilizado em um sentido mais fraco, designando um proce dimento computacional constru´ıdo para desempenhar determinada fun¸c˜ao.

8 MANUAL DE COMPUTAC¸AO EVOLUTIVA E METAHEUR ˜ ´ISTICA

J´a deve estar claro que o exame de todas as poss´ıveis combina¸c˜oes de valores das vari´aveis de decis˜ao, no caso da otimiza¸c˜ao combinat´oria, sempre ´e um *algoritmo*, no sentido de que tal proce dimento, se executado, conduz necessariamente `a determina¸c˜ao do ´otimo exato do problema. No entanto, devido ao r´apido crescimento do n´umero de alternativas a serem examinadas `a medida em que o n´umero de vari´aveis cresce, tal tipo de procedimento normalmente n˜ao ´e aplic´avel exceto em problemas muito pequenos, com poucas vari´aveis. Em grande parte dos problemas combinat´orios, o n´umero de alternativas cresce exponencialmente com o n´umero de vari´aveis de decis˜ao. No exemplo de projeto de uma rede de comunica¸c˜oes visto anteriormente, o n´umero de alternativas era igual a 2*N* , ou seja, um crescimento exponencial com o n´umero *N* de poss´ıveis arcos que poderiam compor a rede. Dessa forma, a quest˜ao relevante em Otimiza¸c˜ao Combinat´oria normalmente n˜ao diz respeito a se existe algum algoritmo que teoricamente sempre resolva o problema (pois pelo menos o algoritmo “exaustivo” sempre existe), mas se existe algum procedimento computacional que resolva o problema em “tempo razo´avel” (alguns minutos, ou algumas horas, ou at´e alguns dias, normalmente ´e aceit´avel).

Uma propriedade que frequentemente est´a associada `a no¸c˜ao de “tempo razo´avel” de um pro cedimento ´e sua classe de complexidade. Os procedimentos cuja execu¸c˜ao demanda um n´umero de opera¸c˜oes proporcional a uma express˜ao polinomial envolvendo o n´umero de vari´aveis do problema pertencem `a chamada classe *P*. Formalmente, a classe *P* cont´em todos os problemas de decis˜ao que podem ser resolvidos por uma m´aquina de Turing determin´ıstica usando uma quantidade polinomial de tempo de computa¸c˜ao. Note-se que um algoritmo exaustivo, como mencionado acima, n˜ao per tence `a classe *P*, pois necessariamente envolve um n´umero de opera¸c˜oes que cresce exponencialmente (portanto mais rapidamente que qualquer polinˆomio) com o n´umero de vari´aveis do problema.

H´a diversos tipos de problemas combinat´orios para os quais foram desenvolvidos *algoritmos* de classe de complexidade *P*. Um desses ´e o problema de constru¸c˜ao da ´arvore geradora m´ınima: dado um conjunto de n´os, deseja-se determinar o conjunto de arcos que produz um grafo conexo (ou seja, um grafo no qual exista algum caminho que possa ser percorrido saindo de qualquer n´o e chegando a qualquer outro n´o), com a m´ınima soma de custos dos arcos. Para esse problema, s˜ao hoje corri queiramente empregados algoritmos6 cuja ordem de complexidade ´e *O*(*N* log(*N*)), ou seja, o esfor¸co computacional para a determina¸c˜ao da solu¸c˜ao ´otima ´e proporcional a *N* log(*N*), sendo *N* o n´umero de vari´aveis (arcos que podem ser instalados) do problema7.

O problema de constru¸c˜ao da rede de comunica¸c˜oes descrito no exemplo anteriormente mostrado, em particular, pode ser resolvido por meio de uma adapta¸c˜ao simples de algoritmos para ´arvore gera dora m´ınima. Usando tais algoritmos, a solu¸c˜ao ´otima para o problema de interligar 15 cidades seria obtida em alguns milisegundos, n˜ao sendo portanto necess´aria a enumera¸c˜ao exaustiva de alternativas que levaria trilh˜oes de anos.

2. Heur´ıstica



Nesta se¸c˜ao, ´e apresentada a motiva¸c˜ao para o desenvolvimento de *heur´ısticas*. Conforme foi visto na se¸c˜ao anterior, heur´ısticas s˜ao procedimentos que tratam problemas de otimiza¸c˜ao sem dispor de garantias te´oricas nem de que a solu¸c˜ao ´otima exata seja obtida, e nem tampouco de garantias de que a solu¸c˜ao obtida tenha algum tipo de proximidade em rela¸c˜ao `a solu¸c˜ao ´otima exata. A l´ogica de uma heur´ıstica ´e, ao contr´ario, a de garantir que a solu¸c˜ao obtida seja melhor que a vasta maioria das alternativas dispon´ıveis, sendo nesse sentido uma “boa solu¸c˜ao”. Com tal sentido, pode-se falar de heur´ısticas tanto no contexto da otimiza¸c˜ao com vari´aveis discretas quanto no contexto da otimiza¸c˜ao com vari´aveis cont´ınuas. A quest˜ao pertinente para motivar o desenvolvimento de heur´ısticas ´e: por

6 Os algoritmos mais comumente empregados para o problema da ´arvore geradora m´ınima s˜ao o algoritmo de Kruskal e o algoritmo de Prim.

7 A express˜ao *N* log(*N*) n˜ao ´e um polinˆomio, mas cresce mais devagar que o polinˆomio *N*2, o que faz com que algoritmos dessa ordem de complexidade sejam *P*.

CAP´ITULO 1 

INTRODUC¸AO˜ 9

quˆe n˜ao empregar m´etodos exatos de otimiza¸c˜ao (ou, no caso de vari´avels discretas, algoritmos)? Conforme foi visto na se¸c˜ao anterior, diversos ramos da Otimiza¸c˜ao vieram se dedicando a tratar diferentes classes de problemas, ao longo do s´eculo XX, de forma que hoje encontra-se dispon´ıvel um repert´orio relativamente vasto de m´etodos de otimiza¸c˜ao capazes de determinar, forma eficiente e garantida, a solu¸c˜ao ´otima exata de problemas dessas classes. No entanto, a variedade de problemas de interesse pr´atico, para os quais seria desej´avel que se conhecessem m´etodos exatos e eficientes de resolu¸c˜ao ´e muito mais vasta do que a dos problemas para os quais tais m´etodos encontram-se j´a desenvolvidos.

Para explicar esse ponto, considerem-se os seguintes problemas:

*•* Dado um conjunto de n´os, determinar um conjunto de arcos que ligam todos os n´os, formando um grafo conexo com a m´ınima soma dos custos dos arcos.

*•* Dado um conjunto de n´os, determinar um conjunto de arcos que ligam todos os n´os, formando um grafo conexo com a m´ınima soma dos custos dos arcos, com a restri¸c˜ao adicional de que nenhum n´o possa ter um n´umero de arcos incidindo nele (grau) maior que uma constante *R*.

O primeiro desses problemas ´e o chamado problema da *´arvore geradora m´ınima*, que constitui um desses problemas “b´asicos”, que j´a mereceram estudos extensos, e para os quais j´a se conhecem boas solu¸c˜oes (algoritmos de classe *P*). O segundo problema, por outro lado, ´e enunciado como uma pequena varia¸c˜ao em torno do primeiro problema, e com tal enunciado ele ´e conhecido como o problema da *´arvore geradora m´ınima com restri¸c˜ao de grau*. No que diz respeito `a complexidade computacional do problema, a modifica¸c˜ao causa um impacto profundo: o problema passa a integrar a classe dos problemas *N P*-dif´ıceis, para os quais n˜ao se conhecem hoje algoritmos capazes de determinar seu ´otimo garantidamente em tempo polinomial.

N˜ao ´e dif´ıcil intuir que problemas pr´aticos, correspondentes a situa¸c˜oes do mundo real, normal mente devem atender a certos requisitos que s˜ao impostos pela situa¸c˜ao pr´atica. A restri¸c˜ao de grau em uma rede de computadores, por exemplo, pode ser origin´aria da caracter´ıstica dos equipamentos da rede, que possivelmente teriam um n´umero de sa´ıdas m´aximo igual a *R*. O racioc´ınio que desenvol vemos aqui ´e o seguinte: (i) Mesmo que problemas pr´aticos tenham uma estrutura pr´oxima daquela dos problemas especiais, para os quais s˜ao conhecidos procedimentos exatos e eficientes, normalmente ser´a necess´aria alguma adapta¸c˜ao para que cada situa¸c˜ao particular seja representada adequadamente. (ii) Mesmo que a adapta¸c˜ao seja pequena, ´e grande a chance de que os m´etodos de solu¸c˜ao conhecidos para o problema especial n˜ao sejam mais aplic´aveis ao problema modificado. Tende a haver, portanto, uma profus˜ao de problemas que n˜ao se sabe resolver de maneira exata.

Na verdade, a quest˜ao de por quˆe desenvolver heur´ısticas vai bem al´em da coloca¸c˜ao de fundo pragm´atico que afirmasse que elas seriam necess´arias at´e que se desenvolvam algoritmos exatos de tempo polinomial para substitu´ı-las. Uma das quest˜oes fundamentais em aberto na teoria da com puta¸c˜ao, hoje em dia, diz respeito exatamente `a possibilidade ou impossibilidade de se constru´ırem tais algoritmos, para uma vasta classe de problemas conhecidos como *N P* (a sigla significa “n˜ao deterministicamente polinomial”). Formalmente, esses s˜ao os problemas em que se procura responder “sim” ou “n˜ao” a uma pergunta que diga respeito, por exemplo, `a existˆencia de um tipo de elemento em um conjunto, sendo que para a verifica¸c˜ao de cada elemento do conjunto encontra-se dispon´ıvel um algoritmo de tempo polinomial.

Exemplifiquemos isto no contexto de um problema de otimiza¸c˜ao com vari´aveis discretas. Suponha se que se deseja saber se existe algum vetor de vari´aveis de decis˜ao x cujo valor de fun¸c˜ao objetivo seja menor que um certo valor *�*. Normalmente, fixando-se um certo vetor de vari´aveis x*a*, ´e poss´ıvel em tempo polinomial responder `a pergunta quanto a se a express˜ao *f*(x*a*) *< �* ´e verdadeira ou n˜ao. Se um procedimento for executado de forma a que sejam testadas diversas alternativas aleatoriamente combinadas de valores das vari´aveis de decis˜ao, existe a possibilidade de que seja encontrado em tempo

10 MANUAL DE COMPUTAC¸AO EVOLUTIVA E METAHEUR ˜ ´ISTICA

polinomial algum vetor que atenda a *f*(x) *< �*. Um problema desse tipo ´e definido como pertencente `a classe *N P*.

Os quadros a seguir real¸cam as defini¸c˜oes de problema *P* e de problema *N P*.

Problema *P •* Dados de entrada: *f*(x) e *�*. 

*•* Pergunta: Existe x tal que *f*(x) *< �*?

*•* O problema ser´a da classe *P* se existir um algoritmo determin´ıstico que responda, em tempo polinomial, `a pergunta.



Problema *N P •* Dados de entrada: *f*(x) e *�*. 

*•* Pergunta 1: se for dado um certo x, esse vetor satisfaz *f*(x) *< �*?

*•* Pergunta 2: existe algum vetor x que satisfaz *f*(x) *< �*?

*•* O problema de responder `a Pergunta 2 ser´a da classe *N P* se existir um algoritmo determin´ıstico que responda, em tempo polinomial, `a Pergunta 1.



A motiva¸c˜ao que leva os problemas *N P* a serem designados por este nome pode ser explicada a partir do seguinte procedimento para tratar a Pergunta 2, cuja constru¸c˜ao ´e feita a partir de uma sequˆencia de respostas `a Pergunta 1:

Algoritmo 1 Pergunta 1 - procedimento n˜ao-determin´ıstico

1: Entradas: A fun¸c˜ao *f*(*·*) e o escalar *�*.

2: Sa´ıda: A resposta *R* `a pergunta: existe *x* tal que *f*(*x*) *< �*?

3: *R ←* FALSO

4: enquanto *R* =FALSO fa¸ca

5: Gerar aleatoriamente uma instˆancia do vetor *x*

6: se *f*(*x*) *< �* ent˜ao

7: *R ←* VERDADEIRO

8: fim se

9: fim enquanto



Para entender a denomina¸c˜ao *N P*, suponha-se que existam *m* instˆancias do vetor x que satisfazem *f*(*x*) *< �*, de um total de *M* poss´ıveis x diferentes. O passo 5 do procedimento que gera uma instˆancia aleat´oria de x tem uma probabilidade *mM* de gerar um x tal que *f*(*x*) *< �*. Note-se que para cada valor de *�* diferente, ocorrer´a um n´umero *m* de instˆancias do vetor x que ir˜ao satisfazer *f*(*x*) *< �*. Assim, dado um n´umero *�*, em m´edia ser˜ao necess´arias *Mm* tentativas aleat´orias at´e que se gere um x tal que *f*(*x*) *< �*. O n´umero de tentativas, nesse sentido, n˜ao depende do n´umero *n* de vari´aveis do problema. Assim, o problema de determina¸c˜ao de algum x que satisfa¸ca *f*(*x*) *< �* ser´a resolvido, em m´edia, em tempo polinomial, caso exista algum x que satisfa¸ca a essa rela¸c˜ao. Caso n˜ao exista tal x, o procedimento nunca ir´a terminar.

A pergunta considerada fundamental ´e: ser´a que existe, para todo problema *N P*, um algoritmo determin´ıstico de tempo polinomial que o resolva? Para alguns problemas dessa classe, que constituem o subconjunto conhecido como *P*, sabemos que isso ´e poss´ıvel, e s˜ao conhecidos tais algoritmos. Um exemplo desse tipo de problema pode ser definido a partir da fun¸c˜ao que mede o comprimento total de

CAP´ITULO 1 

INTRODUC¸AO˜ 11

uma ´arvore geradora que conecta um conjunto de n´os. Seja x o vetor que codifica os arcos que podem ligar esse conjunto de n´os, como na express˜ao (1.5), e seja *f*(x) a fun¸c˜ao que representa o comprimento total da ´arvore formada por esses arcos, como na express˜ao (1.6). Considere-se a pergunta: existe algum x tal que *f*(x) *< �*? Dado um vetor x, ´e poss´ıvel saber, em tempo polinomial, se *f*(x) *< �*; basta calcular o valor da express˜ao (1.6) para esse x. Portanto, essa pergunta ´e um problema *N P*. No entanto, como j´a foi discutido, existem procedimentos – os algoritmos de ´arvore geradora m´ınima – que s˜ao capazes de determinar, em tempo polinomial, de maneira determin´ıstica, o vetor x*∗* que minimiza *f*(x). Portanto, tais algoritmos podem ser empregados para responder `a pergunta quanto a se existe algum x que ir´a satisfazer *f*(*x*) *< �*. Basta determinar *f*(x*∗*): se *f*(x*∗*) *< �*, ent˜ao a resposta ´e sim, pois existe pelo menos um vetor que satisfaz a express˜ao, e esse vetor ´e x*∗*. Do contr´ario, a resposta ser´a n˜ao, pois n˜ao pode existir nenhum vetor que produza uma ´arvore com comprimento ainda menor que o m´ınimo. Assim, alguns problemas *N P* s˜ao tamb´em *P*.

No entanto, h´a uma sub-classe dos problemas *N P*, que s˜ao denominados problemas *N P*-completos, para os quais n˜ao apenas n˜ao se conhecem algoritmos de tempo polinomial: h´a hoje a suspeita de que n˜ao possam existir algoritmos de tempo polinomial para tais problemas. Se isso for verdade, haver´a uma impossibilidade te´orica impedindo o desenvolvimento de algoritmos de tempo polinomial que garantidamente conduzam `a solu¸c˜ao ´otima do problema.

H´a ainda os chamados problemas *N P*-dif´ıceis, que s˜ao definidos como problemas pelo menos t˜ao dif´ıceis quanto problemas *N P*-completos relacionados com eles. Problemas de otimiza¸c˜ao frequente mente s˜ao *N P*-dif´ıceis. Suponha-se que o problema de determinar se existe x tal que *f*(x) *< �* seja pertencente `a classe *N P*-completa. Nesse caso, o problema de determinar o vetor x*∗* que minimiza a fun¸c˜ao *f*(x) ser´a *N P*-dif´ıcil, pois determinar esse vetor ´otimo pode envolver um procedimento de executar v´arias vezes um algoritmo que determina um vetor x que satisfa¸ca *f*(x) *< �*, para valores cada vez menores de *�*. Deve-se notar que se colocam dois problemas n˜ao triviais: (i) se n˜ao existir x que satisfa¸ca *f*(x) *< �*, o procedimento que procura tal vetor pode nunca parar; e (ii) mesmo que exista x tal que *f*(x) *< �*, fica cada vez mais dif´ıcil encontrar esse vetor, `a medida em que *�* diminui, pois diminui a propor¸c˜ao *mM* dos vetores que satisfazem a desigualdade, em rela¸c˜ao ao conjunto de todos os poss´ıveis vetores. Dessa forma, ´e poss´ıvel que o problema de minimizar *f*(x) seja de tempo exponencial (e portanto n˜ao seja *N P*) mesmo que o problema de determinar um x que satisfa¸ca *f*(x) *< �* seja *N P*.

Retornamos ent˜ao `a justificativa para o desenvolvimento de heur´ısticas: elas s˜ao necess´arias n˜ao apenas porque n˜ao s˜ao conhecidos hoje algoritmos eficientes para resolver diversos problemas de im portˆancia pr´atica, mas principalmente porque ´e poss´ıvel que tais algoritmos simplesmente n˜ao existam. Dada a diferen¸ca de escala de tempo entre a dura¸c˜ao da execu¸c˜ao de um algoritmo exato com com plexidade exponencial (que, no caso do exemplo discutido anteriormente seria da ordem de trilh˜oes de trilh˜oes de anos) e a de uma heur´ıstica capaz de encontrar uma “boa solu¸c˜ao” para o mesmo problema em alguns minutos, fica clara a necessidade de se desenvolverem heur´ısticas.

Embora a discuss˜ao nesta se¸c˜ao, at´e este ponto, tenha se concentrado em problemas de vari´aveis discretas, as mesmas id´eias gerais se aplicam a problemas com vari´aveis cont´ınuas. Os m´etodos de otimiza¸c˜ao para tais problemas que tˆem garantia de convergˆencia para a solu¸c˜ao exata sempre se baseiam em premissas fortes sobre a estrutura da fun¸c˜ao a ser otimizada. Quando a fun¸c˜ao atende a premissas tais como a de convexidade, ou de unimodalidade e diferenciabilidade, m´etodos especial mente desenvolvidos para a classe de fun¸c˜oes em quest˜ao permitem determinar o ´otimo do problema de otimiza¸c˜ao com eficiˆencia computacional e garantia de convergˆencia. A Figura 1.3 mostra o gr´afico de uma fun¸c˜ao unimodal diferenci´avel, cujo m´ınimo pode ser determinado facilmente por m´etodos tais como os quasi-Newton.

Problemas reais, no entanto, n˜ao se restringem a ocorrer apenas nessas formas especiais: pelo contr´ario, fun¸c˜oes com estrutura bastante arbitr´aria podem ocorrer em problemas pr´aticos. A Figura 1.4 mostra o gr´afico de uma fun¸c˜ao multimodal que possui um grande n´umero de bacias de atra¸c˜ao,

12 MANUAL DE COMPUTAC¸AO EVOLUTIVA E METAHEUR ˜ ´ISTICA

−5

−10

−15

−20

−25

−30

4 −35

3.5

3

9.5

9

2.5

8.5

2

8

Figura 1.3: Gr´afico de uma fun¸c˜ao diferenci´avel e unimodal.

60 40 20

|  |
| --- |

|  |
| --- |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |

|  |
| --- |

0

−20

−40

10

5

0

−5

10

−10

|  |
| --- |

−10

5

0

−5

Figura 1.4: Gr´afico de uma fun¸c˜ao multimodal.

cada uma com um m´ınimo local. Para tais fun¸c˜oes, os m´etodos cl´assicos de otimiza¸c˜ao n˜ao iriam apenas demorar a convergir para o ´otimo: em geral, a convergˆencia simplesmente n˜ao ocorreria. Se se aplicar um m´etodo “de descida” como tentativa de determina¸c˜ao do ponto de m´ınimo da fun¸c˜ao representada na Figura 1.4, o m´etodo ir´a convergir para o m´ınimo local em cuja bacia de atra¸c˜ao o ponto de partida do m´etodo estiver localizado. Esse tipo de m´etodo simplesmente n˜ao possui qualquer mecanismo capaz de gui´a-lo, em meio a m´ultiplos m´ınimos locais, at´e o m´ınimo global. Na verdade, esse tipo de m´etodo n˜ao tem sequer um mecanismo capaz de conduz´ı-lo a outro m´ınimo local melhor, caso ele convirja para um m´ınimo local qualquer.

A necessidade de otimizar fun¸c˜oes desse tipo tem sido invocada como a principal motiva¸c˜ao para o desenvolvimento de m´etodos de otimiza¸c˜ao baseados em heur´ıstica, no caso de problemas de otimiza¸c˜ao com vari´aveis cont´ınuas. A heur´ıstica teria o papel de guiar uma busca no dom´ınio desejado, de

CAP´ITULO 1 

INTRODUC¸AO˜ 13

forma a examinar a “paisagem” geral da fun¸c˜ao, assim realizando uma busca em v´arias regi˜oes candidatas promissoras, com a progressiva concentra¸c˜ao da busca nas melhores regi˜oes encontradas, na tentativa de se determinar o ´otimo global do problema. Heur´ısticas, por defini¸c˜ao, n˜ao ser˜ao dotadas de garantias de convergˆencia para o ´otimo global exato. No entanto, no caso de problemas com vari´aveis cont´ınuas, elas constituem uma forma efetiva (frequentemente a ´unica forma) de se determinarem “boas solu¸c˜oes”, no sentido de serem solu¸c˜oes significativamente melhores que aquelas que seriam normalmente encontradas por m´etodos de busca apenas local8.

3. Computa¸c˜ao Evolutiva



Nos ´ultimos 40 anos do s´eculo XX, foi progressivamente se firmando a id´eia de se constru´ırem heur´ısticas inspiradas em mecanismos de adapta¸c˜ao dos seres vivos, conforme observados na natu reza. A justificativa geral para esse programa de trabalho ´e que tais mecanismos de adapta¸c˜ao, na natureza, devem produzir respostas adequadas para problemas de grande complexidade. Esse efeito ainda deve ocorrer como consequˆencia da execu¸c˜ao de a¸c˜oes simples por m´ultiplos agentes individu ais que interagem entre si e com o ambiente, produzindo coletivamente a solu¸c˜ao do problema de adapta¸c˜ao. Transposta para o contexto da computa¸c˜ao, tal id´eia torna-se bastante atraente, pois implica que agentes que executam tarefas computacionalmente muito simples podem resolver proble mas de grande complexidade quando agrupados em coletividades, com a possibilidade adicional de se explorar o paralelismo computacional que a estrutura de m´ultiplos agentes sugere.

A adapta¸c˜ao das esp´ecies ao ambiente natural, por meio do processo evolutivo, ou a busca de alimentos efetuada de maneira coordenada por colˆonias de formigas ou bandos de p´assaros, ou ainda a resposta do sistema imunol´ogico de mam´ıferos `a invas˜ao de agentes pat´ogenos, envolvem essencialmente a realiza¸c˜ao de buscas em espa¸cos de alternativas de elevada dimens˜ao, sobre fun¸c˜oes que n˜ao apenas s˜ao de elevada complexidade, mas que tamb´em mudam com o passar do tempo, exigindo uma cont´ınua execu¸c˜ao desses mecanismos de adapta¸c˜ao. As solu¸c˜oes para tais problemas de adapta¸c˜ao exibidas pelos atuais seres biol´ogicos existentes no planeta na verdade tratam-se de mecanismos bastante bem sucedidos, pois permitiram a essas esp´ecies evoluir e estar hoje presentes, ap´os bilh˜oes de anos de evolu¸c˜ao da vida e de luta pela sobrevivˆencia. Por esses motivos, acredita-se que a estrutura de tais mecanismos naturais de adapta¸c˜ao9 possa ser transposta com sucesso para a constru¸c˜ao de mecanismos computacionais de adapta¸c˜ao destinados a tratar de problemas de otimiza¸c˜ao de alta complexidade.

A primeira grande met´afora a ser sugerida, nesse contexto, foi a da evolu¸c˜ao natural dos seres vivos. O mecanismo heur´ıstico de adapta¸c˜ao iria tratar simultaneamente um conjunto de diferentes vetores de vari´aveis de decis˜ao (esse conjunto seria o an´alogo computacional de uma “popula¸c˜ao” de seres vivos). Cada um desses vetores de vari´aveis de decis˜ao, levando adiante a analogia, seria um “indiv´ıduo”. Os indiv´ıduos de uma popula¸c˜ao deveriam sofrer varia¸c˜oes aleat´orias que, por um lado, representassem novos pontos a serem analisados, mas que por outro lado tivessem algum tipo de semelhan¸ca (ou proximidade) com os pontos anteriores, seguindo a l´ogica de que mais pontos fossem gerados perto dos melhores pontos anteriores, e menos pontos fossem gerados perto dos piores, assim se criando uma “mem´oria” dentro do mecanismo de otimiza¸c˜ao. Por fim, fazendo analogia com o mecanismo de sele¸c˜ao natural darwiniana, os indiv´ıduos pertencentes a uma popula¸c˜ao s˜ao selecionados para passar para a gera¸c˜ao seguinte de acordo com um processo que pode ser aleat´orio mas que pelo menos garante maiores probabilidades de serem selecionados aos indiv´ıduos com melhores valores de fun¸c˜ao-objetivo – assim gerando uma press˜ao que conduz o processo `a convergˆencia para valores sempre melhores de fun¸c˜ao.

8 Deve-se notar que os m´etodos cl´assicos de otimiza¸c˜ao, quando aplicados a problemas com fun¸c˜oes arbitr´arias, podem simplesmente ficar inaplic´aveis, ou ainda podem passar a ter comportamento imprevis´ıvel (convergindo para pontos que n˜ao s˜ao sequer m´ınimos locais), ou por fim podem executar uma busca local, convergindo para o m´ınimo local da bacia de atra¸c˜ao na qual estiver localizado o ponto inicial da busca. 9 Um caso particular de problema de adapta¸c˜ao ´e o problema de otimiza¸c˜ao, do qual trata este livro.

14 MANUAL DE COMPUTAC¸AO EVOLUTIVA E METAHEUR ˜ ´ISTICA

10

8

6

4

2

0

−2

−4

−6

−8

~~−10 −5 0 5 10~~ −10

10

8

6

4

2

0

−2

−4

−6

−8

~~−10 −5 0 5 10~~ −10

(a) (b) 40

30 20 10

|  |
| --- |

|  |
| --- |
|  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

0

−10

−20

−30

−40

10

0

|  |
| --- |
|  |

|  |
| --- |

|  |  |
| --- | --- |
|  |  |
|  |  |

|  |
| --- |
|  |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |
|  |  |
|  |  |  |  |

−10 −10 −5 0 5 10

(c)

Figura 1.5: Resultados da execu¸c˜ao de um Algoritmo Gen´etico para a determina¸c˜ao do m´ınimo da fun¸c˜ao representada na Figura 1.4: (a) Popula¸c˜ao inicial gerada aleatoriamente. (b) V´arios pontos candidatos a m´ınimos locais determinados pelo AG, sendo o ponto de m´ınimo global destacado por um quadrado. (c) O ponto de ´otimo, tanto no mapa das curvas de n´ıvel quanto no gr´afico da fun¸c˜ao.

Essa met´afora inspirou o desenvolvimento das *Estrat´egias Evolutivas*, dos *Algoritmos Gen´eticos*, e da *Programa¸c˜ao Gen´etica*, cada qual interpretando de forma diferente os diversos elementos desse mecanismo geral de adapta¸c˜ao. Esses algoritmos s˜ao apresentados respectivamente nos cap´ıtulos 3, 2 e 4 deste livro.

A execu¸c˜ao de um algoritmo gen´etico sobre a fun¸c˜ao multimodal representada na Figura 1.4, leva aos resultados ilustrados na Figura 1.5. Na Figura 1.5(a), observa-se a distribui¸c˜ao dos pontos que comp˜oem a popula¸c˜ao inicial, os quais foram gerados aleatoriamente com distribui¸c˜ao uniforme sobre o dom´ınio de interesse. Esses pontos encontram-se sobrepostos ao mapa de curvas de n´ıvel da fun¸c˜ao. Na Figura 1.5(b), s˜ao mostrados pontos pertencentes `a popula¸c˜ao final, ap´os 150 gera¸c˜oes, os quais se localizam nas proximidades de diferentes m´ınimos locais da fun¸c˜ao. O ponto de ´otimo que foi encontrado (aquele cujo valor de fun¸c˜ao foi o menor dentre todos) se encontra representado em destaque. A Figura 1.5(c), por fim, mostra o ponto de ´otimo encontrado, que de fato coincide com

CAP´ITULO 1 

INTRODUC¸AO˜ 15

o ponto no qual ocorre o m´ınimo valor da fun¸c˜ao dentro do dom´ınio, referenciado tanto ao mapa de curvas de n´ıvel quanto ao gr´afico da fun¸c˜ao. Deve-se notar no gr´afico que o ponto de ´otimo global ´e localizado, de fato, um pouco abaixo dos demais m´ınimos locais.

Com o avan¸co das pesquisas, verificou-se que heur´ısticas dotadas de mecanismos distintos de funci onamento podem ser mais adequadas para problemas com determinada estrutura, e outras heur´ısticas podem funcionar melhor em outras classes de problemas. Isso motivou a que outros processos da na tureza diferentes do da evolu¸c˜ao das esp´ecies servissem de inspira¸c˜ao para o desenvolvimento de novas heur´ısticas, dotadas de propriedades que as tornam peculiarmente adequadas para tratar problemas com caracter´ısticas espec´ıficas. Alguns exemplos disso s˜ao:

*•* Um tipo de heur´ıstica, o *sistema de colˆonia de formigas*, cuja motiva¸c˜ao biol´ogica ´e a forma como as colˆonias de formigas encontram caminhos ´otimos que as levem at´e suas fontes de alimento, veio sendo desenvolvida nos ´ultimos 15 anos principalmente para tratar de problemas de determina¸c˜ao de caminhos ´otimos. Essa classe de m´etodos ´e tratada no Cap´ıtulo 5 deste livro.

*•* Outro tipo de heur´ıstica, o *sistema imunol´ogico artificial*, motivada em mecanismos dos sis temas imunol´ogicos dos mam´ıferos, acaba herdando de seu mecanismo natural inspirador a caracter´ıstica de privilegiar a preserva¸c˜ao da diversidade de solu¸c˜oes – isso ´e necess´ario, no caso dos seres vivos, para que estes estejam sempre preparados para combater diferentes organismos invasores, e no caso de problemas de otimiza¸c˜ao, essa caracter´ıstica pode ser tamb´em valiosa em diversos contextos pr´aticos. Esses algoritmos s˜ao abordados no Cap´ıtulo 6 deste livro.

*•* Algumas fun¸c˜oes-objetivo s˜ao particularmente irregulares, com uma dificuldade extra de que essa irregularidade se manifesta em diferentes escalas. Isso pode ser imaginado como se as bacias de atra¸c˜ao da fun¸c˜ao contivessem outras bacias menores, e essas ainda outras, e assim por diante. Para tratar desse tipo de fun¸c˜ao, uma heur´ıstica que se inspirou no fenˆomeno f´ısico do recozimento de metais10, o *recozimento simulado* veio se mostrando bastante adequada. O Cap´ıtulo 8 deste livro apresenta esses algoritmos.

O crescente entendimento dos mecanismos que fundamentam os algoritmos da computa¸c˜ao evolu tiva tem permitido, de certa forma, um afastamento dos novos algoritmos que vˆem sendo desenvolvidos em rela¸c˜ao `a estrita inspira¸c˜ao biol´ogica. De meados da d´ecada dos 1990 para c´a, diversos algoritmos foram desenvolvidos sem que houvesse a preocupa¸c˜ao de que de alguma forma fosse imitado algum mecanismo de adapta¸c˜ao observado na natureza. Hoje podem ser citados algoritmos tais como o de *evolu¸c˜ao diferencial*, ou o de *estima¸c˜ao da distribui¸c˜ao* que, pelos crit´erios: (1) de terem sido propostos de certa forma em continuidade com a tradi¸c˜ao dos algoritmos bio-inspirados; e (2) de serem baseados em busca com aleatoriedade e envolvendo uma popula¸c˜ao de pontos – pode-se dizer que perten¸cam ao campo da Computa¸c˜ao Evolutiva. Esses algoritmos s˜ao descritos respectivamente no Cap´ıtulo 7 e no Cap´ıtulo 11 deste livro. Por outro lado, ao contr´ario dos primeiros algoritmos desse campo, estes dois se fundamentam em opera¸c˜oes que n˜ao reivindicam qualquer semelhan¸ca com nenhum processo natural, e que fazem parte do algoritmo simplesmente porque parece computacionalmente razo´avel que tais opera¸c˜oes contribuam para a determina¸c˜ao do ´otimo. Os novos algoritmos que se encontram em desenvolvimento, na linha de frente da pesquisa em Computa¸c˜ao Evolutiva, tendem a aprofundar essa tendˆencia de incorpora¸c˜ao de opera¸c˜oes e mecanismos que n˜ao sejam bio-inspirados, mas sim inspirados em argumentos matem´aticos ou computacionais.

Metaheur´ısticas e Computa¸c˜ao Evolutiva

O fio tem´atico deste livro s˜ao as heur´ısticas originalmente derivadas do campo da Computa¸c˜ao Evo lutiva. Acabamos de discutir que, embora a inspira¸c˜ao biol´ogica ainda seja uma fonte de id´eias para 10 Na metalurgia, o recozimento serve para que a estrutura cristalina de metais atinja n´ıveis de menor energia, melhorando as propriedades mecˆanicas do material.

16 MANUAL DE COMPUTAC¸AO EVOLUTIVA E METAHEUR ˜ ´ISTICA

as pesquisas nesse campo, ele j´a n˜ao pode ser definido a partir dessa inspira¸c˜ao, pois j´a se tornou frequente o uso de mecanismos cuja justificativa ´e essencialmente matem´atica e computacional. Al´em dessa inflex˜ao conceitual dentro da pr´opria Computa¸c˜ao Evolutiva, h´a um outro processo que tamb´em tem causado um realinhamento das fronteiras da ´area: tem havido, nos ´ultimos anos, uma convergˆencia entre a Computa¸c˜ao Evolutiva e linhas de pesquisa provenientes do campo da Pesquisa Operacional, que h´a muito tˆem se dedicado ao estudo de t´ecnicas heur´ısticas estoc´asticas de prop´osito geral para problemas de otimiza¸c˜ao combinat´oria – as *metaheur´ısticas*.

Diferente das heur´ısticas mais cl´assicas, que eram desenvolvidas para classes de problemas bastante particulares, as metaheur´ısticas s˜ao heur´ısticas que foram desenvolvidas com o objetivo de constitu´ırem t´ecnicas baseadas em opera¸c˜oes suficientemente gerais para serem aplic´aveis `a maior parte dos pro blemas combinat´orios. Dessa forma, as metaheur´ısticas em geral se fundamentaram em conceitos relativamente gen´ericos, tais como o de vizinhan¸cas de pontos no espa¸co de solu¸c˜oes discretas, ou de trajet´orias conectando dois pontos, ou ainda na organiza¸c˜ao da informa¸c˜ao adquirida a respeito de regi˜oes desse espa¸co de solu¸c˜oes, permitindo a constru¸c˜ao de algoritmos eficientes com caracter´ısticas tanto de busca local quanto de busca global.

Dentre as t´ecnicas originariamente classificadas como metaheur´ısticas que tˆem concorrido para essa fus˜ao com os algoritmos da Computa¸c˜ao Evolutiva, podem ser mencionadas como exemplos a *busca tabu*, as t´ecnicas de *busca gulosa randomizada adaptativa*, a *busca local iterada*, a *busca por vizinhan¸ca vari´avel*, e outras. Essas t´ecnicas constituem os temas dos cap´ıtulos 9, 10 e 12 deste livro.

4. Premissa: Localidade Fraca



Conforme j´a foi discutido, as t´ecnicas cl´assicas de otimiza¸c˜ao dependem de que sejam satisfeitas premissas relativamente fortes sobre as fun¸c˜oes a serem otimizadas, para funcionarem corretamente. Assim, no caso de vari´aveis cont´ınuas, h´a m´etodos que dependem de premissas tais como a convexidade, ou a diferenciabilidade, ou a unimodalidade da fun¸c˜ao-objetivo. No caso de vari´aveis discretas, os problemas para os quais s˜ao conhecidos algoritmos exatos de tempo polinomial usualmente s˜ao aqueles para os quais a realiza¸c˜ao de uma busca local gulosa11 ´e suficiente para garantir que se atinja o ´otimo – esse tipo de propriedade da fun¸c˜ao objetivo se parece muito com a propriedade de unimodalidade das fun¸c˜oes de vari´aveis cont´ınuas, que tamb´em garante que uma busca local conduza ao m´ınimo global da fun¸c˜ao-objetivo. Problemas de otimiza¸c˜ao combinat´oria que n˜ao s˜ao suscept´ıveis de serem resolvidos de maneira exata por nenhuma estrat´egia gulosa costumam ser classificados na classe *N P*-dif´ıcil, n˜ao sendo hoje conhecidos para esses problemas algoritmos exatos de tempo polinomial.

Pois bem, para atingirem o ´otimo, os m´etodos da Computa¸c˜ao Evolutiva n˜ao dependem de pre missas dessa ordem. A quest˜ao a ser aqui discutida ´e: esses m´etodos seriam ent˜ao capazes de produzir boas solu¸c˜oes aproximadas para problemas de qualquer natureza, independente da suposi¸c˜ao de qual quer estrutura para a fun¸c˜ao-objetivo? E previs´ ´ ıvel que a resposta a essa pergunta seja negativa. Mas qual seria ent˜ao a premissa da qual depende o funcionamento dos algoritmos evolutivos? Que tipo de estrutura deve possuir uma fun¸c˜ao objetivo para que seja poss´ıvel aplicar um algoritmo desse tipo?

Essa premissa tem sido identificada como sendo a *localidade fraca* da fun¸c˜ao. Para explicar esse conceito, recorremos `a Figura 1.6. Essa figura procura ilustrar a ideia de que mesmo uma fun¸c˜ao com grande grau de aleatoriedade (portanto insuficientemente dotada de estrutura, pelo menos no sentido que normalmente seria exigido pelos m´etodos cl´assicos de otimiza¸c˜ao) ainda pode ser otimizada, pelo

11 Uma busca “gulosa” ´e aquela feita procurando modificar solu¸c˜oes previamente conhecidas de forma a melhor´a-las imediatamente (com uma ´unica mudan¸ca em uma vari´avel, ou com um pequeno n´umero predefinido de mudan¸cas em veri´aveis). No caso mais geral, esse tipo de busca costuma levar a ´otimos locais dos problemas combinat´orios, ou seja, ´otimos que n˜ao podem ser melhorados considerando apenas a pequena modifica¸c˜ao permitida pelo m´etodo, mas que ainda n˜ao atingem o ´otimo global, que s´o seria atingido por meio de uma modifica¸c˜ao envolvendo um maior n´umero de vari´aveis.

30 25 20 15 10 5

CAP´ITULO 1 

INTRODUC¸AO˜ 17

30

25

20

15

10

5

−1~~0 −5 0 5 10 15 20~~ 0

−1~~0 −5 0 5 10 15 20~~ 0

(a) (b)

30

25

20

15

10

5

0

−1~~0 −5 0 5 10 15 20~~ −5

(c)

Figura 1.6: As subfiguras mostram fun¸c˜oes localmente aleat´orias. As subfiguras (a) e (b) representam uma fun¸c˜ao na qual ocorre a localidade fraca, ou seja, que apresentam algum tipo de tendˆencia global. Na subfigura (a) uma popula¸c˜ao de amostras gerada aleatoriamente ´e representada por pontos sobre a curva, e o ponto de m´ınimo exato ´e representado por um quadrado. A subfigura (b) mostra uma nova popula¸c˜ao, gerada a partir de perturba¸c˜oes dos melhores pontos da popula¸c˜ao da subfigura (a). A subfigura (c) mostra uma fun¸c˜ao de caracter´ıstica globalmente aleat´oria, sem nenhum tipo de tendˆencia global. Uma popula¸c˜ao inicial gerada aleatoriamente, representada por pontos sobre o gr´afico, n˜ao cont´em informa¸c˜ao capaz de direcionar a busca para as imedia¸c˜oes do ponto de m´ınimo, representado por um quadrado.

18 MANUAL DE COMPUTAC¸AO EVOLUTIVA E METAHEUR ˜ ´ISTICA

menos em um sentido aproximado, por meio de m´etodos caracter´ısticos da Computa¸c˜ao Evolutiva. O que se exige, para que isso seja poss´ıvel, ´e que a fun¸c˜ao possua pelo menos a caracter´ıstica de *localidade fraca*, que pode ser entendida como possuir uma esp´ecie de tendˆencia global que se superponha `a aleatoriedade local, de forma a permitir que informa¸c˜oes coletadas em diferentes regi˜oes do dom´ınio da fun¸c˜ao possam ser comparadas de forma a discriminar as melhores regi˜oes. Isso ´e ilutrado pelas subfiguras 1.6(a) e 1.6(b), que mostram uma fun¸c˜ao com aleatoriedade local, mas que exibe uma tendˆencia global. Uma popula¸c˜ao inicial, mostrada na subfigura 1.6(a), ´e gerada de forma a amostrar aleatoriamente alguns valores da fun¸c˜ao, e a compara¸c˜ao entre os valores de fun¸c˜ao obtidos permite que alguns pontos sejam identificados como sendo os melhores. Uma nova amostragem realizada nas proximidades desses pontos, mostrada na subfigura 1.6(b), j´a permite a obten¸c˜ao de boas solu¸c˜oes aproximadas para o ´otimo da fun¸c˜ao.

Contrastando com a situa¸c˜ao em que existe a localidade fraca, a subfigura 1.6(c) exibe uma fun¸c˜ao que n˜ao disp˜oe dessa propriedade. Seu gr´afico, em toda a extens˜ao do dom´ınimo de interesse, se assemelha a uma linha passando por uma sequˆencia de pontos gerados aleatoriamente. Em uma si tua¸c˜ao assim, o conhecimento do valor da fun¸c˜ao em alguns pontos do dom´ınio n˜ao traz informa¸c˜ao a respeito de como a fun¸c˜ao deve se comportar em regi˜oes pr´oximas desses pontos12. Assim, uma fun¸c˜ao desprovida de localidade fraca tende a n˜ao ser otimiz´avel, no sentido de que nenhum procedimento sistem´atico ser´a em m´edia melhor que uma busca aleat´oria (feita com o simples sorteio sucessivo de pontos para serem examinados), e esta tamb´em n˜ao ser´a em m´edia melhor que nenhum procedi mento organizado da forma que for. Independente do que se fa¸ca, deparar com o ´otimo depender´a exclusivamente do acaso.

A propriedade de localidade fraca pode ent˜ao ser definida, de maneira mais precisa, como a presen¸ca de *correla¸c˜ao* entre os valores assumidos pela fun¸c˜ao-objetivo, sendo tal correla¸c˜ao tanto maior quanto menor for a *distˆancia* entre os pontos do espa¸co de vari´aveis de decis˜ao nos quais se avalia a fun¸c˜ao objetivo. A ideia ´e que, quanto mais pr´oximos estiverem dois pontos, mais parecidos dever˜ao ser seus valores de fun¸c˜ao-objetivo, pelo menos em um sentido probabil´ıstico. Fun¸c˜oes que tenham tal propriedade ser˜ao otimiz´aveis por meio de t´ecnicas da Computa¸c˜ao Evolutiva.

Essa formula¸c˜ao evidencia um componente essencial para que fa¸ca sentido o conceito de localidade fraca: ´e necess´ario que esteja bem definida uma no¸c˜ao de *distˆancia* no espa¸co de vari´aveis de decis˜ao. Somente o uso de tal no¸c˜ao ´e que permite que um algoritmo fa¸ca buscas em regi˜oes *perto* de um ponto, ou ainda que o algoritmo evite regi˜oes *pr´oximas* de outros pontos. Sem alguma medida de distˆancia, seria imposs´ıvel construir algoritmos evolutivos. No caso de fun¸c˜oes de vari´aveis cont´ınuas, essa no¸c˜ao decorre diretamente das propriedades do espa¸co vetorial normado R*n*, que j´a incluem automaticamente uma defini¸c˜ao de distˆancia (usualmente, mas n˜ao necessariamente, a distˆancia entre dois pontos ´e a norma euclidiana do vetor diferen¸ca entre esses pontos).

No caso de problemas combinat´orios, a situa¸c˜ao ´e bastante diferente. O conceito que seria mais ´obvio para definir a distˆancia entre solu¸c˜oes, nesse caso, seria a contagem do n´umero de vari´aveis cujos valores n˜ao coincidem, de uma solu¸c˜ao para a outra. Tal conceito, entretanto, frequentemente se revela inadequado para medir distˆancias de forma ´util. A Figura 1.7 ilustra essa inadequa¸c˜ao. Para conduzir a discuss˜ao aqui, imagine-se que o grafo da Figura 1.7(a) represente uma rede de distribui¸c˜ao de ´agua, que sai de um reservat´orio representado por um quadrado, e que percorre a rede at´e chegar nos consumidores, representados pelos n´os circulares. As figuras 1.7(b) e 1.7(c) representam modifica¸c˜oes na primeira rede. Deve-se notar que a rede da Figura 1.7(b) ´e formada pela m´ınima perturba¸c˜ao poss´ıvel sobre a rede da Figura 1.7(a): apenas um arco ´e retirado, sendo inclu´ıdo outro arco para

12 O leitor mais atento talvez fique incomodado com um detalhe que est´a sendo apresentado sem muito rigor: uma fun¸c˜ao cont´ınua, como a da subfigura 1.6(c), necessariamente tem de possuir alguma localidade, pois pelo menos a uma distˆancia muito pequena, os valores que a fun¸c˜ao assume devem ser parecidos, pela pr´opria defini¸c˜ao de continuidade. Se o leitor admitir que as amostras devem estar espa¸cadas de uma distˆancia m´ınima *δ* entre si (essa distˆancia pode representar, por exemplo, a resolu¸c˜ao da representa¸c˜ao num´erica em um sistema computacional), a argumenta¸c˜ao aqui desenvolvida torna-se precisa.

CAP´ITULO 1 

INTRODUC¸AO˜ 19

(a)

(b) (c)

Figura 1.7: (a) Uma rede de distribui¸c˜ao de ´agua, ligando o reservat´orio, representado por um quadrado, aos n´os consumidores, representados por c´ırculos. (b) Outra rede de distribui¸c˜ao de ´agua, obtida retirando um arco da primeira rede, e acrescentando outro arco nessa rede. (c) Ainda outra rede de distribui¸c˜ao de ´agua, obtida agora retirando dois arcos da primeira rede, e acrescentando outros dois arcos.

20 MANUAL DE COMPUTAC¸AO EVOLUTIVA E METAHEUR ˜ ´ISTICA

substitu´ı-lo. J´a a rede da Figura 1.7(c) ´e formada por uma perturba¸c˜ao maior: dois arcos s˜ao retirados, sendo inclu´ıdos dois outros arcos n˜ao existentes na configura¸c˜ao da rede da Figura 1.7(a). O ponto relevante ´e: a rede da situa¸c˜ao (b) ´e muito mais diferente da rede da situa¸c˜ao (a) que a rede da situa¸c˜ao (c). Se o leitor analisar as trˆes redes, ir´a notar que o fluxo de ´agua do n´o-raiz at´e os n´os consumidores muda completamente da situa¸c˜ao (a) para a situa¸c˜ao (b). Em muitos arcos da situa¸c˜ao (b), o fluxo aumenta muito, em outros a dire¸c˜ao do fluxo se inverte, em compara¸c˜ao com a rede da situa¸c˜ao (a). J´a na rede da situa¸c˜ao (c), a mudan¸ca n˜ao ´e t˜ao dr´astica: a maior parte dos fluxos de ´agua, ap´os a mudan¸ca de configura¸c˜ao, permanecem parecidos com os fluxos da situa¸c˜ao (a). Os fluxos em alguns poucos ramos se alteram, mesmo assim n˜ao t˜ao drasticamente quanto no caso anterior. A maioria dos fluxos permanece constante. Ou seja, ao contr´ario do que seria previsto pela m´etrica do n´umero de perturba¸c˜oes, a rede (a) est´a mais distante da rede (b) que da rede (c). Parece, portanto, que a modifica¸c˜ao de algumas vari´aveis em um problema combinat´orio deveria significar uma altera¸c˜ao na distˆancia maior que a modifica¸c˜ao em outras vari´aveis, talvez `a semelhan¸ca daquilo que ocorre quando perturbamos os algarismos de um n´umero: a perturba¸c˜ao nos algarismos das unidades representa mudan¸ca menor que a perturba¸c˜ao nos algarismos das centenas ou dos milhares. Uma discuss˜ao a esse respeito ´e apresentada no Cap´ıtulo 13 deste livro.

O problema geral que se coloca pode ser enunciado dessa forma: a propriedade de localidade fraca ´e necess´aria para que seja poss´ıvel a constru¸c˜ao de algoritmos de otimiza¸c˜ao via Computa¸c˜ao Evolutiva. No entanto, para que essa propriedade se manifeste em um problema, ´e preciso que a no¸c˜ao de distˆancia esteja adequadamente constru´ıda, de forma que os algoritmos possam explor´a la. Em s´ıntese, um algoritmo precisa “saber” quando uma solu¸c˜ao ´e pr´oxima de outra, para poder tomar uma decis˜ao quanto a se uma solu¸c˜ao que ainda n˜ao foi examinada deva ser ou n˜ao avaliada. Essa quest˜ao de definir o que ´e distˆancia entre duas solu¸c˜oes, e consequentemente definir o que ´e a vizinhan¸ca de uma solu¸c˜ao, ou ainda o que ´e uma trajet´oria que liga duas solu¸c˜oes, ou outros conceitos relacionados que servem de base para definir as opera¸c˜oes realizadas pelos algoritmos, de uma forma ou de outra perpassa toda a discuss˜ao a respeito de algoritmos evolutivos e metaheur´ısticas para problemas combinat´orios. Dessa maneira, os cap´ıtulos 10 e 12 deste livro tamb´em tˆem conex˜ao com essa discuss˜ao.

5. Conclus˜oes



Este livro ir´a tratar do conjunto de t´ecnicas que vˆem sendo desenvolvidas para tratar dos problemas de otimiza¸c˜ao dotados de pouca estrutura, ou seja, daqueles problemas sobre os quais se sup˜oe haver uma localidade do tipo fraca em seu espa¸co de solu¸c˜oes. O livro est´a organizado em trˆes partes. A Parte I, que compreende os cap´ıtulos 2 ao 6, agrega os m´etodos chamados de bio-inspirados, que foram desenvolvidos por analogia com algum processo biol´ogico de adapta¸c˜ao. A Parte II, que inclui os cap´ıtulos 7 a 12, apresenta os m´etodos que foram desenvolvidos sem estarem ancorados nesse tipo de analogia.

Por fim a Parte III, intitulada de T´opicos Avan¸cados, apresenta um apanhado de quest˜oes es pec´ıficas que s˜ao transversais a todos os m´etodos. Nos cap´ıtulos 13, 14 e 15 e s˜ao abordadas quest˜oes que s˜ao comuns a algoritmos diversos, e que surgem em contextos bastante diversificados: a quest˜ao da codifica¸c˜ao das solu¸c˜oes, o tratamento de restri¸c˜oes e o tratamento de incertezas. Os cap´ıtulos 16 e 17 abordam uma classe de problemas que hoje ´e central na Teoria de Otimiza¸c˜ao: os Problemas Multicrit´erio. O cap´ıtulo 16 discute a quest˜ao da tomada de decis˜ao nesse contexto, no qual faz-se necess´ario escolher uma dentre as m´ultiplas solu¸c˜oes eficientes dispon´ıveis, sendo a escolha feita por meio da intera¸c˜ao com um tomador de decis˜ao, enquanto o cap´ıtulo 17 discute os algoritmos evolutivos empregados para se fazer a determina¸c˜ao do chamado conjunto de solu¸c˜oes eficientes nessa classe de problemas.

A Computa¸c˜ao Evolutiva ´e usualmente classificada, hoje em dia, como um corpo de t´ecnicas per-

CAP´ITULO 1 

INTRODUC¸AO˜ 21

tencentes a um campo do conhecimento um pouco maior, denominado *Inteligˆencia Computacional*. Outras t´ecnicas inclu´ıdas na Inteligˆencia Computacional seriam, por exemplo, as *Redes Neurais Ar tificiais* e os *Sistemas Fuzzy*. O cap´ıtulo 18 aborda a aplica¸c˜ao de algoritmos evolutivos a problemas de classifica¸c˜ao. Essa aplica¸c˜ao al´em de ser de grande relevˆancia *per si*, cumpre neste livro o papel de ilustrar a conex˜ao da tem´atica aqui tratada com outros ramos da Inteligˆencia Computacional – uma vez que esse tipo de problema ´e mais comumente abordado por meio das chamadas t´ecnicas de *Aprendizado de M´aquina*, nas quais se incluem, por exemplo, as Redes Neurais Artificiais.

(Página deixada propositadamente em branco)

23

PARTE I

M´etodos Bio-Inspirados

(Página deixada propositadamente em branco)

25

CAP´ITULO 2

Algoritmos Gen´eticos

*Francisco J. B. Pereira*

*Departamento de Engenharia Inform´atica e de Sistemas*

*Instituto Superior de Engenharia de Coimbra*

Os Algoritmos Gen´eticos (AGs) s˜ao m´etodos de pesquisa probabil´ısticos inspirados nos princ´ıpios da selec¸c˜ao natural e da gen´etica. Foram desenvolvidos por John Holland e pelos seus alunos da Universidade de Michigan durante os anos 1960 (Holland, 1975). De acordo com Goldberg (1989), a investiga¸c˜ao tinha um duplo objectivo: efectuar um estudo rigoroso dos mecanismos de adapta¸c˜ao existentes na natureza e desenvolver modelos computacionais que retivessem os princ´ıpios b´asicos identificados nos sistemas naturais.

1. A Inspira¸c˜ao Biol´ogica



A evolu¸c˜ao biol´ogica efectua uma pesquisa num espa¸co de grande dimens˜ao e complexidade, cons titu´ıdo por todas as poss´ıveis combina¸c˜oes gen´eticas que podem ser geradas. Alguns dos elementos pertencentes a este espa¸co de procura poder˜ao originar organismos vi´aveis. O processo evolutivo con tribui para a obten¸c˜ao de um conjunto de indiv´ıduos bem adaptados ao meio ambiente em que se encontram.

De acordo com os pressupostos apresentados por Darwin, o processo de evolu¸c˜ao natural recorre a dois mecanismos b´asicos: a selec¸c˜ao e a reprodu¸c˜ao com varia¸c˜ao (Darwin, 1859). A selec¸c˜ao

26 MANUAL DE COMPUTAC¸AO EVOLUTIVA E METAHEUR ˜ ´ISTICA

garante que os indiv´ıduos mais aptos (os que melhor se adaptam ao meio ambiente) tˆem maior pro babilidade de sobreviver. Gra¸cas a esta situa¸c˜ao, tˆem a possibilidade de gerar um maior n´umero de descendentes, propagando assim as suas caracter´ısticas ao longo das gera¸c˜oes. A existˆencia de um me canismo de varia¸c˜ao associado `a reprodu¸c˜ao garante que os descendentes gerados n˜ao s˜ao uma c´opia fiel dos progenitores. A combina¸c˜ao destas duas for¸cas permite que, ao longo de sucessivas gera¸c˜oes, a popula¸c˜ao de indiv´ıduos evolua de forma gradual.

A formula¸c˜ao, no in´ıcio do s´eculo XX, da teoria gen´etica da hereditariedade permitiu integrar o conceito de gene no processo evolutivo, tornando mais claro de que forma ´e que as altera¸c˜oes s˜ao introduzidas na popula¸c˜ao. A variabilidade das esp´ecies passa a ser explicada atrav´es da aplica¸c˜ao de um conjunto de operadores gen´eticos, como o operador de recombina¸c˜ao1 ou de muta¸c˜ao. A moderna teoria da evolu¸c˜ao natural, resultante da fus˜ao das ideias originais de Darwin com a gen´etica Mendeliana ´e, normalmente, designada por Neodarwinismo.

Os princ´ıpios b´asicos da evolu¸c˜ao biol´ogica serviram de inspira¸c˜ao para o desenvolvimento de al goritmos de pesquisa e adapta¸c˜ao em ambientes computacionais, tendo em vista aplica¸c˜oes, tanto no campo da engenharia, como no da biologia. A quest˜ao central associada ao surgimento dos AGs ´e colocada de forma simples por John Holland: “Como transformar as ideias de Darwin em algorit mos?”(Holland, 2000).

2. Estrutura de um Algoritmo Gen´etico



Adoptando a inspira¸c˜ao natural, os AGs processam conjuntos de elementos do espa¸co de procura (potenciais solu¸c˜oes para o problema). Estes conjuntos, usualmente denominados popula¸c˜oes, s˜ao transformados (evolu´ıdos) ao longo de sucessivas itera¸c˜oes (gera¸c˜oes). O objectivo ´e encontrar uma solu¸c˜ao de qualidade elevada (idealmente a solu¸c˜ao ´optima) para o problema que est´a a ser resolvido. No in´ıcio s˜ao escolhidas aleatoriamente v´arias solu¸c˜oes. A partir deste conjunto inicial, e de acordo com os dois mecanismos b´asicos utilizados pela evolu¸c˜ao natural referidos anteriormente, as transforma¸c˜oes na popula¸c˜ao de solu¸c˜oes s˜ao efectuadas da seguinte forma:

*•* Os elementos mais aptos de uma determinada gera¸c˜ao s˜ao seleccionados para servirem de proge nitores das solu¸c˜oes que ir˜ao aparecer na gera¸c˜ao seguinte. Na maior parte dos casos, o processo de selec¸c˜ao ´e probabil´ıstico;

*•* Operadores de transforma¸c˜ao, designados operadores gen´eticos, actuam sobre os elementos se leccionadas originando as novas solu¸c˜oes.

Existem duas decis˜oes cruciais que ´e necess´ario tomar quando se pretende aplicar um AG na resolu¸c˜ao de um problema:

*•* Escolha da representa¸c˜ao para as solu¸c˜oes que fazem parte do espa¸co de procura: a proposta original de Holland defendia a utiliza¸c˜ao de um c´odigo bin´ario para efectuar a representa¸c˜ao das poss´ıveis solu¸c˜oes de um problema. Desta forma, os indiv´ıduos processados por um AG cl´assico s˜ao simplesmente sequˆencias de 0’s e 1’s codificando a informa¸c˜ao necess´aria para representar um ponto do espa¸co de procura;

*•* Defini¸c˜ao da fun¸c˜ao de aptid˜ao que associe a cada solu¸c˜ao uma medida de qualidade, represen tando a sua capacidade para resolver o problema em causa.

Para al´em da inspira¸c˜ao biol´ogica, os AGs adoptaram um conjunto de designa¸c˜oes para quali ficar situa¸c˜oes e entidades relacionadas com o seu modo de funcionamento. Assim, o conjunto das 1 O operador de *recombina¸c˜ao* tamb´em ´e chamado, por vezes, de operador de *cruzamento*.

CAP´ITULO 2 

ALGORITMOS GENETICOS ´ 27

solu¸c˜oes que v˜ao sendo processadas ´e usualmente designado por popula¸c˜ao. Cada uma das itera¸c˜oes do processo de optimiza¸c˜ao ´e considerada uma gera¸c˜ao e a sequˆencia das v´arias popula¸c˜oes reflecte a evolu¸c˜ao ao longo do tempo. Os elementos que constituem uma determinada popula¸c˜ao s˜ao desig nados por indiv´ıduos ou cromossomas. Os genes s˜ao os constituintes b´asicos dos cromossomas, podendo ser encarados como cada uma das caracter´ısticas elementares de uma solu¸c˜ao. Est˜ao localiza dos numa determinada posi¸c˜ao ou locus e cada um deles pode tomar um valor de entre um conjunto de possibilidades, denominadas alelos.

A estrutura gen´erica de um AG ´e ilustrada no Algoritmo 1. Nele pode ser consultado o processo iterativo que vai gerando sucessivas popula¸c˜oes (passos 4 a 9).

Algoritmo 1 Estrutura Gen´erica de um Algoritmo Gen´etico

1: *t ←* 0

2: Gerar a popula¸c˜ao inicial *P*(*t*)

3: Avaliar os indiv´ıduos de *P*(*t*)

4: repita

5: Seleccionar progenitores *P*(*t*) a partir de *P*(*t*)

6: Aplicar operadores gen´eticos a *P*(*t*) obtendo a nova popula¸c˜ao *P*(*t* + 1)

7: Avaliar P(t+1)

8: *t ← t* + 1

9: at´e (crit´erio de termina¸c˜ao atingido)

10: Devolver resultado final da optimiza¸c˜ao



Devido ao mecanismo de selec¸c˜ao, a qualidade m´edia dos elementos que constituem a popula¸c˜ao tem tendˆencia para aumentar ao longo do tempo. Os operadores gen´eticos s˜ao os respons´aveis pela obten¸c˜ao de novas solu¸c˜oes, ao mesmo tempo que tentam garantir que o processo mantenha um n´ıvel adequado de diversidade. Um AG possui dois tipos de operadores gen´eticos fundamentais: recombina¸c˜ao e muta¸c˜ao. Os detalhes de implementa¸c˜ao de cada um destes operadores dependem da representa¸c˜ao adoptada. Tal como as duas quest˜oes previamente identificadas (representa¸c˜ao e aptid˜ao), este ´e outro ponto fulcral e deve ser abordado com grande cuidado, de modo a maximizar a efic´acia do processo de optimiza¸c˜ao. Existem, no entanto, princ´ıpios gen´ericos associados ao funcionamento destes operadores que s˜ao descritos a seguir:

*•* Recombina¸c˜ao: Operador estoc´astico que efectua a troca de material gen´etico entre dois pro genitores, conduzindo `a cria¸c˜ao de dois novos indiv´ıduos (os descendentes). Estes ser˜ao formados por sequˆencias gen´eticas parciais de cada um dos elementos originais. O operador de recom bina¸c˜ao ´e considerado o principal respons´avel pela pesquisa efectuada por um AG. O objectivo central da sua aplica¸c˜ao ´e simples de compreender: ele actua sobre indiv´ıduos que foram se leccionados na popula¸c˜ao actual, ou seja, s˜ao solu¸c˜oes com algum potencial. A combina¸c˜ao de caracter´ısticas complementares de indiv´ıduos promissores poder´a original novas solu¸c˜oes que integrem as vantagens de ambos e que possuam uma aptid˜ao acrescida para o problema que est´a a ser resolvido.

*•* Muta¸c˜ao: Operador un´ario que actua sobre as solu¸c˜oes resultantes da recombina¸c˜ao e que altera ligeiramente algumas das suas caracter´ısticas. Num AG, a muta¸c˜ao ´e um processo completa mente aleat´orio e tem como objectivo manter um n´ıvel de diversidade adequado na popula¸c˜ao, garantindo que alelos que eventualmente desapare¸cam tenham a possibilidade de reaparecer. Existem variantes de algoritmos evolutivos descritos em outros cap´ıtulos deste manual, nos quais a muta¸c˜ao desempenha um papel diferente.

Na proposta original de Holland (1975), ´e descrito um terceiro operador gen´etico denominado invers˜ao. Este operador un´ario tem a capacidade de proceder a um reordenamento parcial dos genes

28 MANUAL DE COMPUTAC¸AO EVOLUTIVA E METAHEUR ˜ ´ISTICA

existentes no cromossoma com o objectivo de tornar a aplica¸c˜ao do operador de recombina¸c˜ao mais eficaz. Embora fa¸ca parte da descri¸c˜ao original de um AG, a invers˜ao ´e muito menos utilizada do que os dois outros operadores referenciados2. O pr´oprio Holland, num trabalho recente (Holland, 2000), refere-se `a recombina¸c˜ao e `a muta¸c˜ao como os dois principais operadores gen´eticos de um AG. Por esse motivo, apenas estes dois operadores ser˜ao apresentados neste texto.

A medida que o n´ ` umero de gera¸c˜oes aumenta, um AG converge gradualmente para regi˜oes do espa¸co de procura onde se encontram solu¸c˜oes promissoras. A optimiza¸c˜ao termina quando um determinado crit´erio de termina¸c˜ao ´e atingido. Os crit´erios mais comuns s˜ao:

*•* N´umero limite de gera¸c˜oes atingido;

*•* Descoberta de uma solu¸c˜ao com qualidade pretendida;

*•* Inexistˆencia de melhoria durante um determinado per´ıodo de tempo.

Ao atingir um crit´erio de termina¸c˜ao, o AG devolve o resultado final da optimiza¸c˜ao. Existem duas alternativas para apresenta¸c˜ao de resultados: devolver a melhor solu¸c˜ao encontrada ao longo da optimiza¸c˜ao ou devolver um conjunto de indiv´ıduos de qualidade elevada (por exemplo, devolver todos os indiv´ıduos que fazem parte da ´ultima gera¸c˜ao).

Bibliografia B´asica

John Holland apresentou a sua ideia original sobre AGs no livro *Adaptation in Natural and Artificial Systems*, originalmente publicado em 1975 (Holland, 1975). E uma obra rigorosa, onde os principais ´ conceitos relacionados com o funcionamento destes algoritmos s˜ao apresentados e analisados. E de lei- ´ tura essencial para quem pretenda compreender os fundamentos te´oricos associados ao funcionamento de um AG.

Ao longo dos anos tˆem sido publicados outros textos introdut´orios que s˜ao uma excelente porta de entrada para quem pretende perceber os principais conceitos relacionados com o funcionamento de um AG. Todos estes livros descrevem exemplos pr´aticos de aplica¸c˜ao, o que facilita a compreens˜ao dos conceitos apresentados. O livro de David Goldberg *Genetic Algorithms in Search, Optimization, and Machine Learning*, apesar de ter sido publicado em 1989, mant´em a sua actualidade (Goldberg, 1989). Alternativas mais recentes e igualmente relevantes s˜ao os livros *Genetic Algorithms + Data Structures = Evolution Programs* de Michalewicz (1992), *An Introduction to Genetic Algorithms* de Mitchell (1996) ou *An Introduction to Evolutionary Computing* de Eiben e Smith (2003).

A obra *Handbook of Evolutionary Computation* editada por B¨ack et al. (1997) agrupa um conjunto muito abrangente de contribui¸c˜oes relativas a toda a ´area da computa¸c˜ao evolutiva. E de consulta ´ obrigat´oria sempre que se pretenda aprofundar ou procurar referˆencias adicionais sobre um assunto espec´ıfico. Por sua vez, o livro *Inteligˆencia Artificial: Fundamentos e Aplica¸c˜oes* de Costa e Simoes (2008) merece uma referˆencia. Embora seja um livro gen´erico de introdu¸c˜ao `a ´area da inteligˆencia artificial possui uma sec¸c˜ao importante dedicada a algoritmos evolutivos. Para alguns leitores poder´a ter a vantagem adicional de ser escrito em portuguˆes. Finalmente, ao longo do texto apresentamos diversas referˆencias bibliogr´aficas que podem servir de introdu¸c˜ao aos diferentes t´opicos abordados.

3. Exemplo: Aplica¸c˜ao de um AG ao Problema da Mochila



Em termos pr´aticos, para aplicar um AG a um determinado problema ´e necess´ario definir qua tro componentes essenciais: representa¸c˜ao para as solu¸c˜oes, fun¸c˜ao de aptid˜ao, m´etodo de selec¸c˜ao 2 A raz˜ao para este facto est´a, provavelmente, relacionada com o surgimento de representa¸c˜oes e de operadores de recombina¸c˜ao alternativos que tornam a sua aplica¸c˜ao menos importante.

CAP´ITULO 2 

ALGORITMOS GENETICOS ´ 29

Tabela 2.1: Propriedades dos objectos da instˆancia do KSP a optimizar

*Obj*1 *Obj*2 *Obj*3 *Obj*4 *Obj*5 *Obj*6 *Obj*7 *Obj*8





Peso 10 18 12 14 13 11 8 6

Valor 5 8 7 6 9 5 4 3





e operadores gen´eticos. E ainda necess´ ´ ario escolher valores para alguns parˆametros que controlam o funcionamento do AG, tais como o tamanho da popula¸c˜ao, o n´umero m´aximo de gera¸c˜oes a processar (assumindo que este ´e o crit´erio de termina¸c˜ao adoptado) ou a probabilidade de aplica¸c˜ao dos opera dores gen´eticos. Para al´em destes, podem existir outros parˆametros espec´ıficos das componentes que constituem o AG.

O Problema da Mochila

O problema da mochila3 (KSP) ´e um exemplo cl´assico de uma situa¸c˜ao de optimiza¸c˜ao combinat´oria e pode ser enunciado da seguinte forma: dado um conjunto de objectos, cada um deles com um peso e um valor espec´ıfico, determinar o subconjunto mais valioso cujo peso total n˜ao ultrapasse a capacidade de uma mochila.

O problema pode ser descrito mais formalmente: existe um conjunto *S* com *N* objectos, sendo cada objecto *Obji* caracterizado por um peso *Pi* e por um valor *Vi*, *i* = 1*, ..., N* (todos os pesos e todos os valores s˜ao positivos). Existe, al´em disso, uma mochila com capacidade *C >* 0. O objectivo ´e encontrar um subconjunto de objectos que satisfa¸ca as seguintes condi¸c˜oes:

max *N*

*i*=1

sujeito a  *N i*=1

*xi × Vi* (2.1) *xi × Pi ≤ C* (2.2)

Nas equa¸c˜oes 2.1 e 2.2, *xi ∈ {*0*,* 1*}* (*xi* = 1 se o objecto *Obji* se encontra na mochila, *xi* = 0 caso contr´ario).

Implementa¸c˜ao do AG

As caracter´ısticas da instˆancia do KSP que vai ser utilizada para ilustrar o funcionamento de um AG s˜ao as seguintes4:

*•* N´umero de objectos: 8

*•* Capacidade da mochila: 35

*•* O peso e o valor de cada um dos objectos podem ser consultados na tabela 2.1.

3 Do inglˆes, *Knapsack Problem*. Na literatura, o problema descrito nesta sec¸c˜ao ´e conhecido como 0-1 Knapsack Problem. Existem diversas variantes do KSP como, por exemplo, o Multidimensional Knapsack Problem. 4 A instˆancia escolhida ´e extremamente simples e serve apenas para exemplificar o funcionamento de um AG.

30 MANUAL DE COMPUTAC¸AO EVOLUTIVA E METAHEUR ˜ ´ISTICA

Representa¸c˜ao

Uma representa¸c˜ao comum para este problema ´e uma sequˆencia bin´aria com N posi¸c˜oes (em que N ´e o n´umero de objectos existentes). Cada posi¸c˜ao est´a associada a um objecto: se tiver o valor 1 significa que foi escolhido para ser inclu´ıdo na mochila. Em rela¸c˜ao ao exemplo que estamos a considerar, a sequˆencia *{*10100000*}* ´e uma solu¸c˜ao poss´ıvel que especifica que os objectos *Obj*1 e *Obj*3 se encontram na mochila.

A representa¸c˜ao proposta ´e a mais natural para o problema do KSP. Para al´em disso, ´e uma representa¸c˜ao bin´aria, o que a torna especialmente apropriada para aplica¸c˜ao de um AG simples. Possui, no entanto, uma limita¸c˜ao importante, uma vez que permite a existˆencia de muitas solu¸c˜oes inv´alidas no espa¸co de procura. Uma solu¸c˜ao inv´alida no KSP ocorre quando o peso total do conjunto de objectos escolhidos excede a capacidade total da mochila. Uma vez que a representa¸c˜ao n˜ao inclui nenhum mecanismo que limite a quantidade de objectos a escolher (por exemplo, a solu¸c˜ao *{*11111111*}* faz parte do espa¸co de procura definido por esta representa¸c˜ao), o processamento de solu¸c˜oes inv´alidas por parte do AG ´e inevit´avel. Existem v´arias alternativas para lidar com esta situa¸c˜ao. A lista seguinte enuncia as trˆes mais relevantes:

*•* Penalizar: A op¸c˜ao mais simples ´e tratar das solu¸c˜oes inv´alidas durante a avalia¸c˜ao. De acordo com este princ´ıpio, as solu¸c˜oes ilegais s˜ao penalizadas. A amplitude da penaliza¸c˜ao ´e proporcional ao grau de viola¸c˜ao das restri¸c˜oes.

*•* Reparar: A repara¸c˜ao de uma solu¸c˜ao inv´alida ´e uma possibilidade interessante, uma vez que permite manter apenas indiv´ıduos legais na popula¸c˜ao. Esta abordagem obriga a desenvolver um algoritmo de repara¸c˜ao simples e r´apido, espec´ıfico para o problema a optimizar. O principal ponto fraco desta estrat´egia ´e a eventual dificuldade em encontrar um algoritmo de repara¸c˜ao. Em alguns casos, a correc¸c˜ao de uma solu¸c˜ao ilegal pode ser uma tarefa extremamente d´ıficil de resolver.

*•* Alterar Representa¸c˜ao: Existem sempre v´arias representa¸c˜oes poss´ıveis para o mesmo pro blema. A decis˜ao de qual a representa¸c˜ao mais apropriada para uma determinada situa¸c˜ao deve levar em considera¸c˜ao as propriedades do espa¸co de procura associado a cada uma delas (por exemplo, se admitem ou n˜ao solu¸c˜oes ilegais). No entanto, ´e importante referir que este n˜ao deve ser o ´unico crit´erio a considerar quando se procede `a escolha da representa¸c˜ao. E essencial ´ considerar as vantagens e desvantagens associadas a cada hip´otese e optar pela que oferece mais garantias em termos de efic´acia.

Neste ponto optamos pela abordagem baseada em penaliza¸c˜ao. Na sec¸c˜ao 5 ser˜ao descritas e analisadas abordagens alternativas.

Aptid˜ao

A fun¸c˜ao de aptid˜ao atribui a qualidade a uma solu¸c˜ao, especificando a sua adequa¸c˜ao para o problema em causa. A qualidade ´e estabelecida de forma quantitativa, o que facilita a compara¸c˜ao entre o m´erito relativo de diferentes solu¸c˜oes.

No caso do KSP, o crit´erio essencial para determinar a qualidade de uma solu¸c˜ao ´e o valor total proporcionado pelos objectos escolhidos. E formalizado como um problema de maximiza¸c˜ ´ ao, pelo que a fun¸c˜ao de aptid˜ao deve atribuir valores mais elevados aos melhores indiv´ıduos. No entanto, e dada a representa¸c˜ao adoptada, ´e necess´ario incluir na fun¸c˜ao de aptid˜ao um mecanismo que penalize as solu¸c˜oes inv´alidas. Se esta componente n˜ao fosse considerada, ´e trivial verificar que a melhor solu¸c˜ao para o problema seria a que escolhe todos os objectos para serem inclu´ıdos na mochila. Claramente esta ´e uma solu¸c˜ao inv´alida, uma vez que o peso total dos objectos excede a capacidade da mochila.

CAP´ITULO 2 

ALGORITMOS GENETICOS ´ 31

A avalia¸c˜ao da aptid˜ao de uma solu¸c˜ao *S* ´e feita recorrendo `a equa¸c˜ao 2.3:

*Qualidade*(*S*) =  *N i*=1

*xi × Vi − P en*(*S*) (2.3)

em que *xi ∈ {*0*,* 1*}* (*xi* = 1 se o objecto *Obji* se encontra na mochila, *xi* = 0 caso contr´ario) e *P en*(*S*) ´e a penaliza¸c˜ao associada `a solu¸c˜ao S. O valor da penaliza¸c˜ao pode ser calculado de diversas formas. Deve, no entanto, obedecer a trˆes princ´ıpios b´asicos: i) a penaliza¸c˜ao ´e proporcional ao grau de viola¸c˜ao da restri¸c˜ao de capacidade; ii) nunca deve compensar exceder a capacidade da mochila; iii) a penaliza¸c˜ao de um ´ındiv´ıduo v´alido ´e 0. Neste texto adoptamos uma penaliza¸c˜ao linear, calculada de acordo com a express˜ao apresentada na equa¸c˜ao 2.4. Para uma descri¸c˜ao de outros tipos de penaliza¸c˜ao, consultar (Costa e Simoes, 2008; Michalewicz, 1992), e o cap´ıtulo 14 deste livro.

*P en*(*S*) =

0*,* se S legal;

*ρ ×* ( *Ni*=1 *xi × Pi − C*)*,* caso contr´ario (2.4)

O valor de *ρ* ´e obtido atrav´es da seguinte express˜ao:

*ρ* = *maxi*=1*,...,NVi* 

*Pi*(2.5)

AG: Simula¸c˜ao de uma Itera¸c˜ao

Estabelecidas a representa¸c˜ao e a aptid˜ao, o AG pode iniciar a optimiza¸c˜ao. Ser´a ainda necess´ario especificar alguns valores para parˆametros, mas eles ser˜ao apresentados `a medida que forem necess´arios. Nesta sec¸c˜ao apresentamos detalhadamente todas as etapas que constituem um ciclo iterativo completo do AG. De acordo com o algoritmo 1, a cria¸c˜ao da popula¸c˜ao inicial ´e a primeira tarefa a ser realizada. Existe um parˆametro, o tamanho da popula¸c˜ao, que indica quantos indiv´ıduos fazem parte de cada gera¸c˜ao. Num AG simples, o seu valor mant´em-se constante ao longo da optimiza¸c˜ao. A dimens˜ao da popula¸c˜ao a escolher para um caso concreto depende de v´arios factores, mas os valores padr˜ao utilizados na maioria das situa¸c˜oes pertencem ao intervalo [100, 500]. No exemplo descrito nesta sec¸c˜ao adoptamos uma popula¸c˜ao de apenas 10 indiv´ıduos para manter a explica¸c˜ao o mais simples poss´ıvel.

A popula¸c˜ao inicial ´e gerada aleatoriamente. Considerando a representa¸c˜ao adoptada e a instˆancia do KSP que est´a a ser optimizada, ´e necess´ario apenas gerar 10 sequˆencias bin´arias de tamanho 8. Ap´os serem gerados os 10 indiv´ıduos da popula¸c˜ao inicial, aplica-se a equa¸c˜ao 2.3 para calcular a qualidade de cada um deles. A tabela 2.2 apresenta um resumo dos individuos gerados e avaliados. Como se pode verificar, na popula¸c˜ao inicial existem 3 indiv´ıduos inv´alidos. A melhor solu¸c˜ao (*I*5) tem qualidade 17.0, correspondendo `a escolha dos objectos *Obj*2 e *Obj*5.

O processo que conduz `a cria¸c˜ao da popula¸c˜ao da nova gera¸c˜ao inicia-se com a selec¸c˜ao dos proge nitores. Neste passo, os indiv´ıduos de melhor qualidade da gera¸c˜ao actual tˆem maior probabilidade de serem seleccionados para dar origem `as novas solu¸c˜oes. V´arios m´etodos de selec¸c˜ao tˆem sido apresenta dos por diversos autores. No trabalho original de Holland ´e proposta a selec¸c˜ao por roleta (ou selec¸c˜ao proporcional) (Holland, 1975). Com este m´etodo, a probabilidade de um indiv´ıduo ser seleccionado baseia-se na raz˜ao entre a sua qualidade e a soma das qualidades de todos os indiv´ıduos da popula¸c˜ao. Diversas desvantagens tˆem sido associadas `a selec¸c˜ao por roleta (Eiben e Smith, 2003). A principal ´e a dificuldade em controlar a press˜ao selectiva, i.e., em estabecer uma rela¸c˜ao apropriada entre o m´erito relativo de um indiv´ıduo e a probabilidade de ser seleccionado como progenitor. Situa¸c˜oes em que a press˜ao selectiva ´e demasiado elevada, i.e., quando apenas um pequeno grupo dos melhores indiv´ıduos conseguem ser escolhidos como progenitores, podem levar o algoritmo a convergir prematuramente.

32 MANUAL DE COMPUTAC¸AO EVOLUTIVA E METAHEUR ˜ ´ISTICA

Tabela 2.2: Popula¸c˜ao Inicial

Indiv´ıduos *Vi**Pi* Penaliza¸c˜ao Qualidade





*I*1 = *{*10100000*}* 12 22 0 12.0

*I*2 = *{*10100010*}* 16 30 0 16.0

*I*3 = *{*00010011*}* 13 38 2.1 10.9

*I*4 = *{*00010000*}* 6 14 0 6.0

*I*5 = *{*01001000*}* 17 31 0 17.0

*I*6 = *{*00000010*}* 4 8 0 4.0

*I*7 = *{*01110110*}* 30 63 19.4 10.6

*I*8 = *{*01000100*}* 13 29 0 13.0

*I*9 = *{*10101101*}* 29 62 18.7 10.3

*I*10 = *{*00011000*}* 15 27 0 15.0





Por outro lado, se a press˜ao selectiva for demasiado baixa, i.e., se todos os indiv´ıduos tiverem aproxi madamente a mesma probabilidade de serem escolhidos como progenitores, o algoritmo deixa de ter a capacidade de identificar e convergir para ´areas promissoras no espa¸co de procura.

Neste texto optamos por aplicar a selec¸c˜ao por torneio (Eiben e Smith, 2003; Mitchell, 1996). Este m´etodo apresenta duas caracter´ısticas importantes que o ajudam a controlar a press˜ao selectiva: por um lado, n˜ao est´a excessivamente dependente dos valores concretos da qualidade dos indiv´ıduos. A ´unica informa¸c˜ao de que necessita para fazer a selec¸c˜ao ´e uma hierarquiza¸c˜ao das solu¸c˜oes (em termos de m´erito). Em segundo lugar, possui um parˆametro designado tamanho do torneio que ajuda a controlar explicitamente a press˜ao selectiva. O algoritmo 2 ilustra como a selec¸c˜ao por torneio escolhe um progenitor.

Num AG simples, o tamanho da popula¸c˜ao mant´em-se constante ao longo das gera¸c˜oes. Se a popula¸c˜ao for constitu´ıda por P indiv´ıduos, ´e necess´ario seleccionar P progenitores para dar origem aos P descendentes que formar˜ao a nova gera¸c˜ao. Deste modo, o processo descrito no algoritmo 2 ´e repetido P vezes. Num processo de selec¸c˜ao alguns elementos (os mais aptos) ser˜ao seleccionados v´arias vezes, enquanto que outros poder˜ao nunca ser escolhidos. E esta press˜ ´ ao evolutiva que possibilita a convergˆencia gradual do algoritmo para ´areas promissoras do espa¸co de procura. Tal como foi referido anteriormente, o tamanho de torneio ajuda a controlar a press˜ao selectiva. Em regra, este parˆametro assume um valor entre 2 e 5, sendo raro encontrar situa¸c˜oes em que seja superior. Numa situa¸c˜ao de torneio bin´ario (tamanho de torneio igual a 2), a press˜ao selectiva ´e m´ınima5. Pelo contr´ario, se o tamanho de torneio for 5, a press˜ao selectiva j´a ´e consider´avel6.

No exemplo do KSP que est´a a ser analisado, ´e adoptado um torneio bin´ario. A tabela 2.3 ilustra o processo de selec¸c˜ao e indica quais os indiv´ıduos escolhidos como progenitores. Cada linha mostra os dois indiv´ıduos envolvidos num torneio e apresenta o vencedor (o que tem melhor qualidade)

Ap´os a selec¸c˜ao, os progenitores s˜ao agrupados em pares, aos quais ´e aplicado o operador de recombina¸c˜ao. Neste exemplo recorremos ao operador original proposto por Holland (1975), designado recombina¸c˜ao com um ponto de corte. De acordo com este operador, os dois indiv´ıduos s˜ao alinhados, um ponto de corte ´e escolhido aleatoriamente ao longo dos cromossomas e os descendentes

5 O n´umero esperado de c´opias do melhor elemento da popula¸c˜ao actual no conjunto de progenitores ´e 2 e apenas a pior solu¸c˜ao est´a garantidamente afastada do processo reprodutivo. 6 Como ´e evidente, o tamanho da popula¸c˜ao tamb´em influencia a press˜ao selectiva.

CAP´ITULO 2 

ALGORITMOS GENETICOS ´ 33

Algoritmo 2 Selec¸c˜ao por torneio com tamanho de torneio k

1: Escolher da popula¸c˜ao actual *k* indiv´ıduos aleatoriamente

2: Seleccionar o melhor dos *k* indiv´ıduos como progenitor

Tabela 2.3: Selec¸c˜ao dos progenitores

Torneio Participantes Vencedor





1 *{I*8(13*.*0)*, I*3(10*.*9)*} I*8

2 *{I*1(12*.*0)*, I*5(17*.*0)*} I*5

3 *{I*4(6*.*0)*, I*9(10*.*3)*} I*9

4 *{I*4(6*.*0)*, I*10(15*.*0)*} I*10

5 *{I*8(13*.*0)*, I*3(10*.*9)*} I*8

6 *{I*9(10*.*3)*, I*10(15*.*0)*} I*10

7 *{I*7(10*.*6)*, I*3(10*.*9)*} I*3

8 *{I*5(17*.*0)*, I*2(16*.*0)*} I*5

9 *{I*7(10*.*6)*, I*5(17*.*0)*} I*5

10 *{I*7(10*.*6)*, I*2(16*.*0)*} I*2





s˜ao criados a partir de sequˆencias complementares dos dois progenitores. A figura 2.1 resume o modo de actua¸c˜ao do operador de recombina¸c˜ao com um ponto de corte. Existem outras propostas, como a recombina¸c˜ao com v´arios pontos de corte ou a recombina¸c˜ao uniforme, que generalizam o modo de actua¸c˜ao deste operador para representa¸c˜oes bin´arias. A descri¸c˜ao de seu funcionamento pode ser consultada em (Eiben e Smith, 2003; Booker et al., 1997). A probabilidade de aplica¸c˜ao da recombina¸c˜ao ´e um parˆametro do AG e define a probabilidade deste operador ser efectivamente aplicado a cada um dos pares de progenitores. O seu valor ´e usualmente elevado (valores t´ıpicos entre 0.7 e 1.0), uma vez que ´e o principal respons´avel pela constru¸c˜ao de boas solu¸c˜oes. Nos casos em que o operador de recombina¸c˜ao n˜ao ´e aplicado, os progenitores envolvidos passam para a fase seguinte sem altera¸c˜oes.

Na tabela 2.4 exemplifica-se o efeito da aplica¸c˜ao da recombina¸c˜ao com um ponto de corte aos elementos escolhidos anteriormente. Considera-se que os pares de progenitores s˜ao formados de acordo com a ordem pela qual foram seleccionados7. A barra vertical identifica o ponto de corte em cada

7 Deve evitar-se que o operador de recombina¸c˜ao seja aplicado a duas c´opias do mesmo indiv´ıduo.

P1

D1

P2

D2

Figura 2.1: Recombina¸c˜ao com um ponto de corte

34 MANUAL DE COMPUTAC¸AO EVOLUTIVA E METAHEUR ˜ ´ISTICA

Tabela 2.4: Aplica¸c˜ao do operador de recombina¸c˜ao com um ponto de corte aos indiv´ıduos seleccio nados. A barra vertical identifica o local de corte em cada par de progenitores.

Progenitores Descendentes





*{*010 *|* 00100*} {*01001000*}*

*{*010 *|* 01000*} {*01000100*}*

**

*{*1010 *|* 1101*} {*10101000*}*

*{*0001 *|* 1000*} {*00011101*}*

**

*{*01000100*} {*01000100*}*

*{*00011000*} {*00011000*}*

**

*{*0100010 *|* 0*} {*0100010 1*}*

*{*0001001 *|* 1*} {*0001001 0*}*

**

*{*010010 *|* 0 0*} {*01001010*}*

*{*101000 *|* 1 0*} {*10100000*}*

**

**

um dos pares. Neste exemplo assume-se que o operador de recombina¸c˜ao n˜ao ´e aplicado ao terceiro par, pelo que estes progenitores passam sem altera¸c˜ao para a fase seguinte. Pode ainda observar se um fen´omeno curioso na aplica¸c˜ao da recombina¸c˜ao ao primeiro par de solu¸c˜oes: embora os dois cromossomas sejam diferentes, a localiza¸c˜ao do ponto de corte faz com que os descendentes gerados sejam iguais aos progenitores.

Os indiv´ıduos que resultam da recombina¸c˜ao s˜ao sujeitos a muta¸c˜ao. No exemplo que estamos a relatar adoptamos a muta¸c˜ao bin´aria, o operador cl´assico para representa¸c˜oes bin´arias (Holland, 1975; Goldberg, 1989). O seu modo de actua¸c˜ao ´e extremamente simples: quando aplicada a um gene do cromossoma (i.e., a um constituinte b´asico da solu¸c˜ao que neste caso ´e um bit), troca o seu valor. A probabilidade de aplica¸c˜ao do operador de muta¸c˜ao ´e mais um parˆametro do AG, sendo definida em rela¸c˜ao a genes individuais, ou seja, define qual a probabilidade de um gene sofrer uma muta¸c˜ao. O seu valor ´e bastante baixo, sendo vulgar encontrar valores entre 0.001 e 0.01. Tal como foi referido anteriormente, num AG este operador tem como fun¸c˜ao principal agitar a pesquisa, reintroduzindo diversidade e evitando que esta fique presa em ´optimos locais de fraca qualidade.

Na tabela 2.5 exemplifica-se o efeito da aplica¸c˜ao da muta¸c˜ao bin´aria no exemplo que est´a a ser descrito. Na coluna da esquerda surgem os indiv´ıduos que resultam da recombina¸c˜ao. Os genes sublinhados s˜ao sujeitos a muta¸c˜ao, dando origem ao novo conjunto de indiv´ıduos que surge na coluna da direita. Depois da aplica¸c˜ao do operador de muta¸c˜ao, o processo de cria¸c˜ao da popula¸c˜ao da nova gera¸c˜ao est´a conclu´ıdo. As novas solu¸c˜oes s˜ao avaliadas (tabela 2.6) e, caso n˜ao tenha sido atingido nenhum dos crit´erios de termina¸c˜ao que estejam a ser considerados, pode iniciar-se imediatamente uma nova itera¸c˜ao do algoritmo.

Em resumo, a aplica¸c˜ao de um AG a uma instˆancia do KSP obrigou a definir as seguintes compo nentes:

*•* Representa¸c˜ao para as solu¸c˜oes;

*•* Fun¸c˜ao de aptid˜ao;

CAP´ITULO 2 

ALGORITMOS GENETICOS ´ 35

Tabela 2.5: Aplica¸c˜ao do operador de muta¸c˜ao bin´aria aos indiv´ıduos que resultam da recombina¸c˜ao. Os genes sublinhados foram escolhidos para ser sujeitos a muta¸c˜ao.

Antes da muta¸c˜ao Depois da muta¸c˜ao





*{*010 01000*} {*01101000*}*

**

*{*01000100*} {*01000100*}*

*{*10101 000*} {*10100000*}*

**

*{*00011101*} {*00011101*}*

*{*0 1 00010 0*} {*00000110*}*

**

*{*00011000*} {*00011000*}*

*{*01000101*} {*01000101*}*

*{*000100 1 0*} {*00010110*}*

**

*{*0 1 0 01010*} {*00101010*}*

**

*{*10100 000*} {*10101000*}*

**

**

**

Tabela 2.6: Avalia¸c˜ao dos descendentes

Indiv´ıduos *Vi**Pi* Penaliza¸c˜ao Qualidade





*D*1 = *{*01101000*}* 24 43 5.5 18.5

*D*2 = *{*01000100*}* 13 29 0 13.0

*D*3 = *{*10100000*}* 12 22 0 12.0

*D*4 = *{*00011101*}* 23 54 13.2 9.8

*D*5 = *{*00000110*}* 9 19 0 9.0

*D*6 = *{*00011000*}* 15 27 0 15.0

*D*7 = *{*01000101*}* 16 45 6.9 9.1

*D*8 = *{*00010110*}* 15 33 0 15.0

*D*9 = *{*00101010*}* 20 33 0 20.0

*D*10 = *{*10101000*}* 21 35 0 21.0





36 MANUAL DE COMPUTAC¸AO EVOLUTIVA E METAHEUR ˜ ´ISTICA

*•* Mecanismo de selec¸c˜ao;

*•* Operadores gen´eticos de recombina¸c˜ao e muta¸c˜ao;

*•* Crit´erio de termina¸c˜ao.

Para al´em destas componentes, foi ainda necess´ario especificar valores para os seguintes parˆametros: tamanho da popula¸c˜ao, tamanho de torneio, probabilidade de aplica¸c˜ao da recombina¸c˜ao, probabi lidade de aplica¸c˜ao da muta¸c˜ao e n´umero m´aximo de gera¸c˜oes (assumindo que ´e este o crit´erio de termina¸c˜ao adoptado).

Medidas de Desempenho

Quando se aplica um AG a um determinado problema ´e importante conseguir perceber se ele apresenta um bom desempenho. A avalia¸c˜ao pode ser feita em compara¸c˜ao com outro m´etodo de optimiza¸c˜ao ou pode servir para determinar qual a melhor configura¸c˜ao para o AG que est´a a ser analisado. Os AGs s˜ao m´etodos de pesquisa estoc´asticos pelo que as compara¸c˜oes devem ser efectuadas utilizando testes estat´ısticos8. Para que estes testes possam ser utilizados, a aplica¸c˜ao de um AG a um determinado problema deve ser repetida v´arias vezes. S´o assim poder˜ao ser obtidos dados suficientes para tornar a an´alise estat´ıstica significativa. O n´umero de repeti¸c˜oes de um AG ´e um parˆametro adicional que deve ser especificado9.

As medidas mais utilizadas para aferir o desempenho de um AG s˜ao a eficiˆencia e a efic´acia. A primeira est´a directamente relacionada com a sua rapidez, enquando que a segunda mede a qualidade das solu¸c˜oes. H´a trˆes crit´erios comuns para estimar a qualidade:

*•* Taxa de sucesso: Se a qualidade da solu¸c˜ao ´optima for conhecida, pode contabilizar-se quantas repeti¸c˜oes do AG a conseguem descobrir. A raz˜ao entre este valor e o total de repeti¸c˜oes corresponde `a taxa de sucesso do algoritmo.

*•* M´edia das melhores solu¸c˜oes10: Para cada uma das repeti¸c˜oes realizadas determina-se a qualidade da melhor solu¸c˜ao. Esta medida ´e obtida efectuando a m´edia destes valores.

*•* Melhor solu¸c˜ao encontrada11: Qualidade da melhor solu¸c˜ao encontrada pelo AG.

Os trˆes crit´erios descritos est˜ao evidentemente relacionados. Al´em disso, dependendo do problema concreto e do objectivo da optimiza¸c˜ao, pode fazer mais sentido utilizar uma ou outra medida. Em (Eiben e Smith, 2003, cap. 14) pode ser consultada uma descri¸c˜ao detalhada das vantagens e limita¸c˜oes de cada um destes crit´erios.

4. Propriedades dos Algoritmos Gen´eticos



Apesar da sua simplicidade, os AGs possuem um conjunto de propriedades que os tornam especi almente robustos para efectuar pesquisas em espa¸cos de grande dimens˜ao e complexidade. Trˆes das principais caracter´ısticas que contribuem para esta situa¸c˜ao s˜ao:

*•* Processamento simultˆaneo de uma popula¸c˜ao de potenciais solu¸c˜oes para o problema. Esta particularidade permite que diferentes ´areas do espa¸co de procura sejam analisadas em paralelo, reduzindo a probabilidade de convergˆencia prematura para um ´optimo local.

8 A apresenta¸c˜ao dos testes existentes e das condi¸c˜oes em que podem ser aplicados est´a fora do ˆambito deste cap´ıtulo. Consultar (Zar, 2009) para uma apresenta¸c˜ao detalhada deste t´opico. 9 Embora existam excep¸c˜oes, 30 repeti¸c˜oes ´e um valor padr˜ao normalmente aceite pela comunidade da computa¸c˜ao evolutiva.

10 Do inglˆes *mean best fitness*. 11 Do inglˆes, *best-ever fitness*.

CAP´ITULO 2 

ALGORITMOS GENETICOS ´ 37

*•* Possuem componentes probabil´ısticas e n˜ao recorrem a informa¸c˜ao espec´ıfica sobre o problema que est˜ao a resolver (a ´unica informa¸c˜ao de que necessitam ´e o resultado da aplica¸c˜ao da fun¸c˜ao de aptid˜ao). Estes dois factores, combinados com a existˆencia de uma popula¸c˜ao de solu¸c˜oes, tornam estes m´etodos particularmente vers´ateis e robustos.

*•* Equil´ıbrio entre explora¸c˜ao de novas regi˜oes do espa¸co e conserva¸c˜ao da informa¸c˜ao j´a adqui rida: a selec¸c˜ao actua como principal for¸ca de conserva¸c˜ao, permitindo manter e consolidar a informa¸c˜ao descoberta anteriormente. Por seu lado, atrav´es de transforma¸c˜oes efectuadas nos elementos seleccionados, ´e promovida a explora¸c˜ao de novas regi˜oes do espa¸co. Assumindo que este possui algum tipo de regularidade, as referidas transforma¸c˜oes poder˜ao conduzir `a desco berta de novas regi˜oes, onde se encontrem solu¸c˜oes de qualidade superior.

Fundamenta¸c˜ao Te´orica

Na sua proposta, Holland analisou teoricamente o desempenho dos AGs como estrat´egia de pesquisa. A descri¸c˜ao rigorosa da an´alise efectuada est´a fora do ˆambito deste texto, aconselhando-se a consulta das obras (Holland, 1975; Goldberg, 1989) para um estudo mais detalhado. Neste ponto limitamo-nos a descrever de forma simplificada algumas das ideias principais. Toda a an´alise te´orica definida por Holland ´e sustentada por dois conceitos b´asicos: a no¸c˜ao de esquema e a no¸c˜ao de blocos constru tores. Um esquema ´e um modelo representativo de um determinado subconjunto de cromossomas que partilham os mesmos valores em algumas posi¸c˜oes. Mais concretamente, e limitando a an´alise a alfabetos bin´arios, um esquema ´e uma sequˆencia constitu´ıda pelos s´ımbolos *{*0*,* 1*, ∗}*. Posi¸c˜oes con tendo 0 ou 1 s˜ao consideradas definidas, enquanto que as que contˆem o s´ımbolo *∗* s˜ao consideradas irrelevantes para a an´alise12. A utiliza¸c˜ao de esquemas permite representar um subconjunto do espa¸co de procura de forma muito eficiente. Um cromossoma *ch* pertence a um determinado esquema *E* se e s´o se as sequˆencias *ch* e *E* forem iguais em todas as posi¸c˜oes em que *E* n˜ao cont´em um *∗*. Considere-se o seguinte exemplo: o esquema *E* = *{*1 *∗∗∗* 1*}* representa todos os cromossomas bin´arios com 5 genes que possuam um bit com valor 1 na primeira e na ´ultima posi¸c˜ao. As sequˆencias que se encaixam no esquema (por exemplo, 10001 ou 10101) designam-se instˆancias de E. Dois atributos dos esquemas s˜ao a sua ordem (n´umero posi¸c˜oes definidas) e o seu tamanho definido (distˆancia entre a primeira e ultima posi¸ ´ c˜oes definidas). Analisando o esquema *E*, verifica-se que ´e de ordem 2 e que o seu tamanho definido ´e 4.

Cada indiv´ıduo da popula¸c˜ao representa um n´umero elevado de esquemas (exactamente 2*n*, sendo n o n´umero de genes do cromossoma). Como consequˆencia, numa determinada gera¸c˜ao, apesar de o AG apenas avaliar explicitamente os indiv´ıduos que pertencem `a popula¸c˜ao, est´a implicitamente a estimar a qualidade de um n´umero muito elevado de esquemas. A qualidade de um esquema ´e definida como a qualidade m´edia de todas as suas instˆancias que pertencem `a popula¸c˜ao. Segundo Holland (1975), ´e a existˆencia deste paralelismo impl´ıcito que permite a um AG efectuar de forma eficiente a pesquisa de um espa¸co de procura de grande dimens˜ao. O teorema fundamental dos AGs, tamb´em chamado teorema dos esquemas13, enuncia o modo como a pesquisa ´e efectuada: “O n´umero de instˆancias de esquemas de ordem baixa, com tamanho definido reduzido e com qualidade acima da m´edia aumenta de forma exponencial ao longo das sucessivas gera¸c˜oes de um AG”. Por sua vez, a hip´otese dos blocos construtores ´e uma consequˆencia l´ogica do teorema anterior: “Um AG efectua a pesquisa de solu¸c˜oes de boa qualidade atrav´es da combina¸c˜ao de esquemas de pequena dimens˜ao (i.e., ordem baixa e tamanho definido reduzido) e de qualidade acima da m´edia. Estes esquemas s˜ao denominados os blocos construtores da solu¸c˜ao”.

A generaliza¸c˜ao da utiliza¸c˜ao de AGs conduziu a novas tentativas no sentido de entender o seu modo de funcionamento. Uma das quest˜oes centrais associadas `a an´alise do desempenho de um AG 12 Do inglˆes, *don’t care symbol*. 13 Do inglˆes: *schemata theorem*.

38 MANUAL DE COMPUTAC¸AO EVOLUTIVA E METAHEUR ˜ ´ISTICA

´e a seguinte: “Em quais problemas ´e suposto um AG ser eficaz?”. A consulta dos volumes da s´erie *Foundations of Genetic Algorithms* ´e uma boa introdu¸c˜ao para se obter uma ideia dos desenvolvimentos te´oricos recentes desta ´area.

5. Extens˜oes ao Algoritmo Gen´etico Simples



O AG descrito na sec¸c˜ao 3 possui as componentes b´asicas originalmente propostas por Holland (1975), sendo habitualmente identificado como AG simples ou AG can´onico. A ´unica excep¸c˜ao ´e o mecanismo de selec¸c˜ao adoptado. Uma consequˆencia l´ogica da dissemina¸c˜ao e populariza¸c˜ao desta t´ecnica ´e a introdu¸c˜ao de altera¸c˜oes `a proposta original. Ao longo dos anos, diversas modifica¸c˜oes foram introduzidas na sua estrutura, tanto ao n´ıvel da representa¸c˜ao e operadores utilizados, como ao n´ıvel da pr´opria arquitectura do AG. Actualmente, as op¸c˜oes de implementa¸c˜ao est˜ao, na maioria das situa¸c˜oes, dependentes do problema que se pretende resolver. Alguns exemplos de representa¸c˜oes alternativas incluem a utiliza¸c˜ao de vectores de n´umeros reais, permuta¸c˜oes, tabelas multidimensionais, sistemas de regras ou grafos. Uma das consequˆencias desta situa¸c˜ao ´e a necessidade de desenvolver operadores gen´eticos que se adaptem `as novas representa¸c˜oes (Eiben e Smith, 2003; Michalewicz, 1992). Nesta sec¸c˜ao apresentamos alguns exemplos de alternativas ao algoritmo can´onico.

Representa¸c˜ao e Operadores Gen´eticos

A selec¸c˜ao da representa¸c˜ao a adoptar para as solu¸c˜oes ´e crucial. Esta ´e provavelmente a decis˜ao mais importante a tomar quando se efectua o desenvolvimento de um AG para aplicar num problema de optimiza¸c˜ao, at´e porque v´arias outras componentes (como, por exemplo, os operadores gen´eticos ou a fun¸c˜ao de aptid˜ao) dependem desta decis˜ao.

Alguns factores que ´e necess´ario considerar s˜ao:

*•* O alfabeto deve ser bin´ario ou pode conter mais s´ımbolos?

*•* A representa¸c˜ao deve ser discreta ou cont´ınua?

*•* O ordenamento dos genes no cromossoma ´e relevante?

*•* O cromossoma pode conter informa¸c˜ao redundante?

*•* A representa¸c˜ao permite a existˆencia de solu¸c˜oes inv´alidas? Em caso afirmativo, como lidar com elas?

As decis˜oes a tomar em rela¸c˜ao aos t´opicos enunciados est˜ao directamente relacionadas com o problema a optimizar. Encontrar a representa¸c˜ao apropriada para um problema, ou conjunto de problemas, ´e uma tarefa dif´ıcil e ´e um t´opico importante de investiga¸c˜ao. Quando confrontados com um caso concreto, a experiˆencia acumulada na utiliza¸c˜ao de AGs e uma an´alise cuidada do problema a resolver ajudam a tomar uma decis˜ao sobre qual poder´a ser a melhor op¸c˜ao. Al´em disso, a realiza¸c˜ao de um conjunto preliminar de testes com v´arias alternativas pode ajudar a fundamentar melhor a escolha efectuada.

Para al´em da representa¸c˜ao bin´aria, a representa¸c˜ao real e a baseada em permuta¸c˜oes s˜ao as 2 alternativas que s˜ao adoptadas mais vezes por AGs. Nas pr´oximas sec¸c˜oes descrevemos brevemente as suas propriedades e apresentamos propostas de operadores gen´eticos para lidar com cada uma delas.

CAP´ITULO 2 

ALGORITMOS GENETICOS ´ 39

Representa¸c˜ao real

Os problemas nos quais as vari´aveis de decis˜ao tomam valores reais s˜ao muito comuns. Nestes casos, uma solu¸c˜ao ´e representada atrav´es de uma sequˆencia de n´umeros reais. Embora seja poss´ıvel recorrer a uma sequˆencia bin´aria para a codifica¸c˜ao14, ´e mais natural adoptar uma representa¸c˜ao real em que cada gene corresponde directamente uma vari´avel de decis˜ao. Cada um destes genes tem associado o seu pr´oprio dom´ınio. Quando se lida com problemas em dom´ınios cont´ınuos, esta representa¸c˜ao tem a vantagem de ser mais natural, ou seja, est´a mais pr´oxima das solu¸c˜oes que representa. Esta ´e uma caracter´ıstica importante que deve ser levada em considera¸c˜ao quando se est´a a decidir qual a representa¸c˜ao mais apropriada para um dado problema.

Considere que se pretende minimizar a fun¸c˜ao apresentada na equa¸c˜ao 2.6.

*f*(*x*1*, x*2) = 100 *×* (*x*21 *− x*2)2 + (1 *− x*1)2 *−* 2*.*048 *≤ x*1*, x*2 *≤* 2*.*048 (2.6)

Para este problema, a representa¸c˜ao real de uma solu¸c˜ao ´e feita recorrendo a 2 valores reais, respectivamente para as vari´aveis *x*1 e *x*2. Neste caso, uma solu¸c˜ao poss´ıvel ser´a: *{−*1*.*235*,* 0*.*023*}*. O n´umero de casas decimais a adoptar depende da precis˜ao que se pretende, embora esteja obviamente condicionada pelas limita¸c˜oes f´ısicas associadas `a implementa¸c˜ao computacional.

Os operadores de recombina¸c˜ao existentes para representa¸c˜oes reais dividem-se em dois grupos: recombina¸c˜ao discreta vs. recombina¸c˜ao aritm´etica. Os do primeiro grupo actuam de forma an´aloga aos operadores de recombina¸c˜ao bin´arios: determinam um ou mais pontos de corte entre genes e criam descendentes juntando sec¸c˜oes complementares dos progenitores. A principal desvantagem dos operadores discretos ´e a incapacidade que tˆem para introduzir novos valores na popula¸c˜ao, garantindo apenas novas combina¸c˜oes para alelos j´a existentes.

Os operadores de recombina¸c˜ao aritm´eticos permitem ultrapassar esta limita¸c˜ao. Com estes ope radores, o valor de cada um dos genes dos descendentes ´e encontrado dentro do intervalo definido pelos respectivos alelos que surgem nos progenitores15. Na equa¸c˜ao 2.7 apresentamos um exemplo de aplica¸c˜ao de um operador de recombina¸c˜ao aritm´etico. Dados dois progenitores *P*1 e *P*2, os descen dentes *D*1 e *D*2 s˜ao criados de acordo com as seguintes express˜oes:

*D*1 = *λ × P*1 + (1 *− λ*) *× P*2

*D*2 = *λ × P*2 + (1 *− λ*) *× P*1 (2.7)

Na equa¸c˜ao *λ ∈* [0*,* 1], o que significa que este operador efectua uma m´edia pesada dos valores dos progenitores para gerar os descendentes.

Em tra¸cos gerais existem dois tipos de operadores de muta¸c˜ao para representa¸c˜oes reais. Na muta¸c˜ao uniforme, o novo valor ´e escolhido aleatoriamente do dom´ınio do gene em quest˜ao. Pelo contr´ario, na muta¸c˜ao n˜ao-uniforme o valor actual do gene ´e tido em considera¸c˜ao e ´e alterado de uma forma mais limitada. A muta¸c˜ao gaussiana ´e o exemplo mais comum, sendo que neste caso o novo valor para um gene *Gi* ´e obtido atrav´es da seguinte express˜ao *Gi ← Gi* + *N*(0*,* 1).

Representa¸c˜ao em permuta¸c˜ao

Para apresentar as propriedades das representa¸c˜oes baseadas em permuta¸c˜oes regressamos ao exem plo do KSP. No que diz respeito a este problema, para al´em da representa¸c˜ao bin´aria, v´arias outras propostas tˆem sido apresentadas e analisadas. Em (Hinterding, 1999; Raidl e Gottlieb, 2005) pode

14 Em (Michalewicz, 1992) pode ser consultado um exemplo detalhado de todos os passos envolvidos neste processo. Ver ainda (Herrera et al., 1998) para um discuss˜ao das vantagens/desvantagens da utiliza¸c˜ao de representa¸c˜oes bin´arias em problemas cont´ınuos. 15 V´arios operadores de recombina¸c˜ao relaxam esta restri¸c˜ao e possibilitam que o novo valor saia ligeiramente do intervalo (Herrera et al., 1998). Esta modifica¸c˜ao pode ajudar a combater a convergˆencia prematura do algoritmo.

40 MANUAL DE COMPUTAC¸AO EVOLUTIVA E METAHEUR ˜ ´ISTICA

Tabela 2.7: Descodifica¸c˜ao da solu¸c˜ao *{*1*,* 8*,* 7*,* 4*,* 2*,* 6*,* 3*,* 5*}* utilizando a heur´ıstica *first-fit*. O s´ımbolo *√* indica que o objecto foi seleccionado para a mochila.

18742635



Mochila *√√√ X X √ X X*

Peso Acumulado 10 16 24 24 24 35 35 35





ser consultada uma lista abrangente das v´arias representa¸c˜oes. Uma das mais estudadas ´e a repre senta¸c˜ao baseada numa permuta¸c˜ao de valores, originalmente proposta por Hinterding (1999). As permuta¸c˜oes s˜ao representa¸c˜oes habitualmente utilizadas em problemas em que ´e necess´ario encontrar uma sequˆencia ideal para um conjunto de objectos. Entre os exemplos mais comuns, incluem-se o problema do caixeiro viajante, o problema do encaminhamento de ve´ıculos ou problemas de escalona mento.

As representa¸c˜oes em permuta¸c˜ao codificam uma sequˆencia ordenada de s´ımbolos, na qual n˜ao podem existir repeti¸c˜oes. A ordem pela qual os s´ımbolos surgem na representa¸c˜ao e/ou as rela¸c˜oes de adjacˆencia s˜ao as caracter´ısticas mais relevantes para definir a qualidade de uma solu¸c˜ao. No caso do KSP, a representa¸c˜ao consiste numa permuta¸c˜ao de todos objectos que fazem parte do problema que est´a a ser optimizado. Considerando o exemplo com 8 objectos apresentado na sec¸c˜ao 3, duas solu¸c˜oes poss´ıveis s˜ao *{*1*,* 4*,* 7*,* 8*,* 2*,* 6*,* 3*,* 5*}* ou *{*5*,* 2*,* 4*,* 1*,* 8*,* 7*,* 3*,* 6*}*16 .

A representa¸c˜ao em permuta¸c˜ao proposta para o KSP designa-se indirecta, uma vez que a con sulta de uma solu¸c˜ao n˜ao especifica imediatamente quais os objectos que se encontram na mochila. E necess´ario aplicar um algoritmo de descodifica¸c˜ ´ ao para interpretar a informa¸c˜ao armazenada no cromossoma e indicar qual a solu¸c˜ao final. A ordem pela qual os objectos surgem no cromossoma representa a sua prioridade no que diz respeito `a entrada na mochila. Uma heur´ıstica simples do tipo *first-fit* considera os objectos pela ordem em que se encontram na solu¸c˜ao e adiciona-os, se eles n˜ao provocarem a viola¸c˜ao da restri¸c˜ao de capacidade. Considerando a permuta¸c˜ao *{*1*,* 8*,* 7*,* 4*,* 2*,* 6*,* 3*,* 5*}* e as propriedades indicadas na tabela 2.1, na tabela 2.7 exemplifica-se o processo de descodifica¸c˜ao. A solu¸c˜ao final obtida indica que os objectos *{*1*,* 8*,* 7e6*}* foram seleccionados para entrar na mochila.

Existem vantagens e desvantagens associadas `a adopta¸c˜ao de uma representa¸c˜ao baseada em per muta¸c˜oes para resolver o problema do KSP. Ali´as, a um n´ıvel mais geral, existem sempre pontos fortes e pontos fracos associados a cada representa¸c˜ao que pode ser considerada para um determinado problema. Estes devem ser identificados e analisados, uma vez que isto ajudar´a a fazer uma escolha mais fundamentada pela representa¸c˜ao que se acredita ser mais adequada. Regressando `a proposta baseada em permuta¸c˜oes para o KSP, podem ser identificadas as seguintes vantagens/desvantagens17:

*•* Vantagens:

– A representa¸c˜ao em permuta¸c˜ao garante que as solu¸c˜oes processadas pelo AG s˜ao sem pre v´alidas. A existˆencia de um algoritmo de descodifica¸c˜ao assegura que a restri¸c˜ao de capacidade da mochila nunca ´e violada.

– Uma vez que o AG apenas processa solu¸c˜oes v´alidas, ´e mais simples implementar a fun¸c˜ao de aptid˜ao. N˜ao ´e necess´ario utilizar nenhum mecanismo de penaliza¸c˜ao para punir indiv´ıduos

16 Para simplificar a nota¸c˜ao, cada objecto ´e representado apenas pelo seu identificador num´erico. 17 Embora a an´alise seja feita para um problema particular, praticamente todos os t´opicos s˜ao v´alidos para situa¸c˜oes em que se opta por uma representa¸c˜ao indirecta

CAP´ITULO 2 

ALGORITMOS GENETICOS ´ 41

Tabela 2.8: Aplica¸c˜ao do operador de recombina¸c˜ao com um ponto de corte a permuta¸c˜oes. A barra vertical identifica o local de corte em cada par de progenitores.

Progenitores Descendentes





*{*3542 *|* 17986*} {*354243869*}*

*{*5127 *|* 43869*} {*512717986*}*

**

**

ilegais. A compara¸c˜ao entre solu¸c˜oes tamb´em se torna mais clara, uma vez que o ´unico crit´erio a considerar ´e o valor total proporcionado pelos objectos seleccionados.

*•* Desvantagens:

– A principal desvantagem associada `a utiliza¸c˜ao de uma representa¸c˜ao indirecta est´a direc tamente relacionada com a necessidade de desenvolver e aplicar um algoritmo de descodi fica¸c˜ao. Em muitos casos n˜ao ´e trivial encontrar um algoritmo que permita descodificar o cromossoma numa solu¸c˜ao final legal. Esta situa¸c˜ao n˜ao se verifica no problema do KSP que est´a a ser utilizado como exemplo, mas ´e comum em muitos problemas de optimiza¸c˜ao.

– O algoritmo de descodifica¸c˜ao ´e espec´ıfico do problema e da representa¸c˜ao, o que diminui a generalidade do AG. Al´em disso, e embora se pretenda que o algoritmo de descodifica¸c˜ao seja o mais simples poss´ıvel, existe sempre um tempo adicional de processamento que pode condicionar a eficiˆencia do processo de optimiza¸c˜ao.

O modo de actuar dos operadores gen´eticos depende da representa¸c˜ao adoptada. E f´ ´ acil verificar que a aplica¸c˜ao da recombina¸c˜ao com um ponto de corte e da muta¸c˜ao bin´aria a solu¸c˜oes codifica das como permuta¸c˜oes gera indiv´ıduos ilegais (ilegais, no sentido de n˜ao pertencerem ao espa¸co de procura)18. Na tabela 2.8 ilustra-se como a aplica¸c˜ao da recombina¸c˜ao com um ponto de corte pode originar solu¸c˜oes ilegais.

Na literatura existem v´arias propostas de operadores gen´eticos para permuta¸c˜oes. Tal como referi mos anteriormente, esta ´e uma representa¸c˜ao muito utilizada em alguns dos mais relevantes problemas de optimiza¸c˜ao que ocorrem no mundo real, o que naturalmente conduziu ao surgimento de diversas alternativas. Um operador de recombina¸c˜ao deve ter a capacidade de, dadas duas permuta¸c˜oes, gerar dois descendentes tais que: i) s˜ao permuta¸c˜oes legais (i.e., contˆem todos os s´ımbolos uma e uma s´o vez); ii) s˜ao constitu´ıdos por informa¸c˜ao relevante (i.e., blocos construtores) presente em cada um dos progenitores. No caso de uma permuta¸c˜ao, a informa¸c˜ao relevante inclui a posi¸c˜ao absoluta ocupada por cada um dos s´ımbolos e as rela¸c˜oes de precedˆencia e de adjacˆencia que se verificam. Dependendo do tipo de problema que est´a a ser resolvido, pode ser mais importante considerar a posi¸c˜ao absoluta dos s´ımbolos como, por exemplo, em problemas de escalonamento ou as rela¸c˜oes de adjacˆencia, como no caso do caixeiro viajante. Alguns dos exemplos mais conhecidos de operadores s˜ao a recombina¸c˜ao por ordem (*Order crossover*), recombina¸c˜ao por ciclo (*Cycle crossover*), recombina¸c˜ao por aresta (*Edge crossover*) ou o PMX (*Partially mapped crossover*). Em (Eiben e Smith, 2003; Michalewicz, 1992) pode ser consultado o funcionamento detalhado de cada um destes operadores e uma lista de problemas para os quais s˜ao mais apropriados.

Neste cap´ıtulo apresentamos o operador de recombina¸c˜ao por ordem, originalmente proposto por Davis (1991). Este operador ´e especialmente apropriado para lidar com situa¸c˜oes em que a informa¸c˜ao 18 Em sentido estrito, a muta¸c˜ao bin´aria n˜ao pode ser aplicada a uma representa¸c˜ao contendo s´ımbolos inteiros. No

entanto pode ser facilmente extendida para um operador que troque o valor de um gene para outro pertencente ao dom´ınio.

42 MANUAL DE COMPUTAC¸AO EVOLUTIVA E METAHEUR ˜ ´ISTICA

P1 3 5 4 2 1 7 9 8 6

a)

P2 5 1 2 7 4 3 8 6 9

D1 2 1 7 9 b)

D1 4 3 8 2 1 7 9 6 5 c)

Figura 2.2: Recombina¸c˜ao por ordem

mais importante a transmitir aos descendentes ´e a ordem pela qual os s´ımbolos aparecem nos pro genitores. No algoritmo 3 podem ser consultados os passos executados por este operador para criar dois descendentes D1 e D2 a partir dos progenitores P1 e P2. A figura 2.2 apresenta um exemplo de aplica¸c˜ao deste operador. O painel a) da figura apresenta os dois progenitores e os locais dos pontos de corte. No painel b) pode ver-se a sec¸c˜ao de P1 que ´e herdada por D1 e, finalmente, no painel c) surge o descendente j´a com os s´ımbolos herdados de P2. A gera¸c˜ao do descendente D2 fica como exerc´ıcio para o leitor.

Algoritmo 3 Operador de recombina¸c˜ao por ordem. Gera dois descendentes D1 e D2 a partir dos progenitores P1 e P2.

1: Alinhar as sequˆencias P1 e P2

2: Escolher aleatoriamente dois pontos de corte nas sequˆencias P1 e P2. Os pontos de corte situam-se entre dois s´ımbolos

3: Copiar a sec¸c˜ao entre os pontos de corte de P1 para D1

4: Com in´ıcio na posi¸c˜ao a seguir ao segundo ponto de corte, copiar os restantes s´ımbolos de P2 para D1. Os dois cromossomas P2 e D1 devem ser considerados circulares

5: Criar D2 repetindo os dois passos anteriores, mas trocando o papel desempenhado pelos progeni tores P1 e P2.



Devido `as propriedades das permuta¸c˜oes, a aplica¸c˜ao da muta¸c˜ao n˜ao pode considerar genes in dividualmente. A alternativa ´e escolher subconjuntos de genes e alterar as suas posi¸c˜oes. Existem v´arios operadores de muta¸c˜ao padronizados que s˜ao habitualmente aplicados a permuta¸c˜oes: muta¸c˜ao por troca (*swap mutation*), muta¸c˜ao por inser¸c˜ao (*insert mutation*), muta¸c˜ao por deslocamento (*dis placement mutation*) ou muta¸c˜ao por invers˜ao (*inversion mutation*). Para uma descri¸c˜ao completa e detalhada do modo de funcionamento destes operadores consultar (Eiben e Smith, 2003; Michalewicz, 1992). Neste cap´ıtulo apresentamos dois exemplos simples que ilustram o modo de funcionamento de dois destes operadores. No painel a) da figura 2.3 apresentamos um exemplo de aplica¸c˜ao da muta¸c˜ao por troca: dois genes s˜ao escolhidos aleatoriamente no cromossoma e os seus valores s˜ao trocados; no painel b) da mesma figura, a muta¸c˜ao por invers˜ao escolhe dois genes aleatoriamente e inverte a

CAP´ITULO 2 

ALGORITMOS GENETICOS ´ 43

3 5 4 2 1 7 9 8 6

a)

3 5 6 2 1 7 9 8 4

3 5 4 2 1 7 9 8 6

b)

3 5 4 2 8 9 7 1 6

Figura 2.3: Exemplos de operadores de muta¸c˜ao para permuta¸c˜oes: a) muta¸c˜ao por troca; b) muta¸c˜ao por invers˜ao

ordem dos valores que se encontram entre essas duas posi¸c˜oes.

Algoritmos de Repara¸c˜ao

Antes de propor novas formas de organiza¸c˜ao para o modo como os AGs processam solu¸c˜oes, ´e im portante regressar por um momento `a representa¸c˜ao bin´aria proposta para o KSP. Foram referidas na sec¸c˜ao 3, trˆes alternativas para lidar com solu¸c˜oes inv´alidas. Duas delas, penaliza¸c˜ao e desenvolvimento de representa¸c˜oes alternativas, foram j´a discutidas. E altura de abordar brevemente a repara¸c˜ ´ ao de solu¸c˜oes. Esta abordagem possui v´arios pontos de contacto com a utiliza¸c˜ao de representa¸c˜oes indirec tas descrita anteriormente. Tal como neste caso existe uma heur´ıstica simples, neste caso designada algoritmo de correc¸c˜ao, que actua sobre as solu¸c˜oes inv´alidas da popula¸c˜ao, corrigindo-as19. O ob jectivo ´e garantir que todos os indiv´ıduos avaliados s˜ao solu¸c˜oes legais. A defini¸c˜ao de um algoritmo de correc¸c˜ao pode ser uma tarefa complicada. Em v´arios casos, poder´a ser (quase) t˜ao dif´ıcil corrigir uma solu¸c˜ao inv´alida, como resolver o problema que se est´a a optimizar. Um caso paradigm´atico ´e o problema da constru¸c˜ao dos hor´arios de uma escola (Lewis, 2008). Devido a todas as restri¸c˜oes envolvidas, corrigir uma solu¸c˜ao ilegal pode ser extremamente dif´ıcil.

No caso do KSP ´e trivial corrigir um indiv´ıduo. Uma solu¸c˜ao ´e ilegal porque a capacidade da mochila foi ultrapassada, logo, a correc¸c˜ao deve ir retirando objectos at´e que o problema esteja ultra passado. O algoritmo 4 ilustra os passos a executar para corrigir a solu¸c˜ao S. Embora este algoritmo n˜ao especifique como devem ser escolhidos os objectos a retirar da mochila, existem duas possibili dades: retirar os objectos de acordo com uma heur´ıstica (por exemplo, o objecto mais pesado ou o objecto com pior *ratio* entre valor e peso) ou retir´a-los aleatoriamente. A primeira op¸c˜ao parece ser mais apropriada, mas a segunda hip´otese n˜ao deve ser imediatamente descartada. Proporciona maior robustez e evita introduzir componentes sˆofregas na pesquisa que poder˜ao comprometer a efic´acia global do algoritmo.

19 Em tra¸cos gerais, as vantagens e desvantagens associadas `a utiliza¸c˜ao de uma representa¸c˜ao indirecta mantˆem-se v´alidas.

44 MANUAL DE COMPUTAC¸AO EVOLUTIVA E METAHEUR ˜ ´ISTICA

Algoritmo 4 Repara¸c˜ao de uma representa¸c˜ao bin´aria para o KSP 1: enquanto ( *Ni*=1 *xi × Pi > C*) fa¸ca 2: Escolher uma posi¸c˜ao *xi* com valor 1

3: *xi ←* 0

4: fim enquanto



Arquitecturas alternativas

Num AG simples, as popula¸c˜oes sucedem-se itera¸c˜ao ap´os itera¸c˜ao sem que exista nenhum tipo de competi¸c˜ao entre progenitores e descendentes. De acordo com esta organiza¸c˜ao, designada geracional, assim que termina a cria¸c˜ao da nova popula¸c˜ao, a anterior desaparece. Ao longo dos anos, diversas alternativas de processamento das popula¸c˜oes foram apresentadas com o objectivo de aumentar a efic´acia dos AGs. Nas pr´oximas sec¸c˜oes descrevemos brevemente duas dessas propostas.

Elitismo

A qualidade m´edia dos indiv´ıduos que fazem parte das popula¸c˜oes processadas por um AG aumenta ao longo das gera¸c˜oes. Este aumento ´e mais pronunciado no in´ıcio da optimiza¸c˜ao e tem tendˆencia para diminuir ou estagnar `a medida que a popula¸c˜ao come¸ca a convergir.

Pelo contr´ario, ´e relativamente comum que a qualidade da melhor solu¸c˜ao diminua de uma gera¸c˜ao para outra. O desaparecimento do indiv´ıduo de melhor qualidade ocorre devido ao caracter proba bil´ıstico da selec¸c˜ao ou, mais provavelmente, devido `a ac¸c˜ao dos operadores gen´eticos. O elitismo evita que isto aconte¸ca. O funcionamento desta estrat´egia ´e muito simples de descrever: quando ter mina a avalia¸c˜ao da popula¸c˜ao de descendentes verifica-se qual ´e a qualidade do melhor indiv´ıduo. Se for inferior `a qualidade da melhor solu¸c˜ao da gera¸c˜ao anterior, esta ´e recuperada e inserida na nova popula¸c˜ao. Como a popula¸c˜ao tem tamanho fixo, a adi¸c˜ao de uma solu¸c˜ao obriga a eliminar um dos descendentes que acabou de ser criado. A escolha do elemento sacrificado pode ser feita de forma aleat´oria ou de acordo com um crit´erio pr´e-estabelecido (por exemplo, eliminar a solu¸c˜ao de pior qualidade).

Algoritmos *steady-state*

Nos AGs do tipo *regime permanente*20 deixa de existir a no¸c˜ao de gera¸c˜oes bem definidas em que os indiv´ıduos s˜ao completamente substitu´ıdos em cada itera¸c˜ao (Whitley, 1989). Pelo contr´ario, os progenitores e descendentes passam a coexistir e a competir por um lugar na popula¸c˜ao. O primeiro autor a estudar esta possibilidade em AGs foi De Jong em 1975 (De Jong, 1975). No seu estudo definiu um parˆametro *G* (designado *generation gap*) que estabelece a frac¸c˜ao da popula¸c˜ao que ´e substitu´ıda em cada gera¸c˜ao. O seu valor pode variar entre *G* = 1, correspondendo a uma organiza¸c˜ao geracional, at´e *G* = 1*/P* (em que P ´e o tamanho da popula¸c˜ao), no qual apenas um indiv´ıduo ´e substitu´ıdo em cada itera¸c˜ao.

O algoritmo 5 ilustra o modo de funcionamento de um AG *regime permanente* para *G* = 1*/P*. V´arios detalhes de implementa¸c˜ao devem ser clarificados em rela¸c˜ao ao funcionamento deste algoritmo (consultar (Lozano et al., 2004; Smith, 2007) para obter detalhes sobre as v´arias alternativas) :

*•* Selec¸c˜ao: o processo de selec¸c˜ao dos progenitores pode ser efectuado com um qualquer mecanismo de selec¸c˜ao (por exemplo, selec¸c˜ao por torneio), embora seja relativamente comum escolher as solu¸c˜oes aleatoriamente.

20 Do inglˆes *steady-state*.

CAP´ITULO 2 

ALGORITMOS GENETICOS ´ 45

*•* Operadores gen´eticos: embora a recombina¸c˜ao crie habitualmente duas solu¸c˜oes, o algoritmo apresentado lida apenas com um descendente. Neste passo pode assumir-se que ´e escolhido o descendente de melhor qualidade (ou mesmo que a escolha ´e feita de forma aleat´oria).

*•* Dom´ınio de substitui¸c˜ao: existem duas possibilidades para seleccionar a solu¸c˜ao *Z* que poder´a ser substitu´ıda pelo descendente. A primeira restringe a escolha aos progenitores. A segunda alarga a escolha a toda a popula¸c˜ao ou a um subconjunto da popula¸c˜ao.

*•* Estrat´egia de substitui¸c˜ao: Independentemente do dom´ınio considerado, a escolha da solu¸c˜ao *Z* a substituir pode ser feita de acordo com v´arios crit´erios. De entre as possibilidades, destacam-se a substitui¸c˜ao por qualidade, por idade (i.e., a solu¸c˜ao que est´a h´a mais tempo na popula¸c˜ao), por semelhan¸ca21 ou mesmo aleatoriamente. E poss´ ´ ıvel ainda considerar uma combina¸c˜ao de crit´erios para encontrar a solu¸c˜ao *Z* a substituir.

*•* Condi¸c˜ao de substitui¸c˜ao: No ´ultimo passo do algoritmo, a decis˜ao sobre a entrada do descen dente D na popula¸c˜ao est´a normalmente relacionada com a sua qualidade, ou seja, a solu¸c˜ao ´e aceite se tiver melhor qualidade do que o indiv´ıduo *Z* escolhido para ser substitu´ıdo. Uma alternativa a esta abordagem ´e efectuar a substitui¸c˜ao incondicional.

As regras gen´ericas enunciadas em cima permitem adoptar uma grande variedade de estrat´egias. Uma das mais conhecidas ´e a *replace worst*, proposta por Whitley no seu sistema GENITOR (Whitley, 1989) que advoga que a solu¸c˜ao *Z* ´e a de pior qualidade na popula¸c˜ao e a que a condi¸c˜ao de substitui¸c˜ao ´e baseada no m´erito, ou seja, o descendente entra popula¸c˜ao desde que tenha melhor qualidade do que o pior indiv´ıduo que l´a se encontra. Embora seja simples e intuitiva, esta estrat´egia provoca uma r´apida diminui¸c˜ao de diversidade, levando o algoritmo a convergir prematuramente. Por este motivo ´e normalmente prefer´ıvel adoptar um mecanismo que, para al´em da qualidade, considere medidas de semelhan¸ca entre indiv´ıduos, de modo a manter n´ıveis de diversidade adequados dentro da popula¸c˜ao.

Algoritmo 5 Algoritmo Gen´etico em Regime Permanente (Steady-State)

1: Seleccionar dois progenitores P1 e P2 da popula¸c˜ao actual

2: Criar um descendente D atrav´es da aplica¸c˜ao de operadores gen´eticos

3: Avaliar o desdendente D

4: Seleccionar uma solu¸c˜ao Z na popula¸c˜ao para substitui¸c˜ao

5: Decidir se o descendente D substitui a solu¸c˜ao Z



6. Aplica¸c˜oes Pr´aticas



Os AGs s˜ao actualmente um m´etodo de resolu¸c˜ao de problemas muito utilizado, tendo demonstrado a sua efic´acia nos mais diferentes dom´ınios, tanto cient´ıficos como em situa¸c˜oes do mundo real. A seguir apresentamos alguns exemplos de ´areas nas quais os AGs tˆem sido aplicados.

*•* Optimiza¸c˜ao num´erica: Esta categoria inclui muitos problemas de engenharia que podem ser modelados atrav´es de um conjunto de parˆametros reais para os quais ´e necess´ario encontrar a melhor configura¸c˜ao. O que caracteriza esta classe ´e o facto de os dom´ınios associados a cada parˆametro serem cont´ınuos. Dois exemplos de ´areas nas quais podem surgir este tipo de problemas s˜ao o controlo (Fleming e Purshouse, 2002) ou a qu´ımica te´orica e computacional (Johnston, 2004).

21 Existem diversas medidas que procuram aferir a semelhan¸ca entre duas solu¸c˜oes para um dado problema. S˜ao normalmente m´etricas espec´ıficas de uma dada representa¸c˜ao e/ou problema e procuram determinar at´e que ponto duas solu¸c˜oes partilham caracter´ısticas idˆenticas.

46 MANUAL DE COMPUTAC¸AO EVOLUTIVA E METAHEUR ˜ ´ISTICA

*•* Optimiza¸c˜ao combinat´oria: Os problemas desta classe s˜ao, provavelmente, as situa¸c˜oes de optimiza¸c˜ao `as quais os AGs s˜ao aplicados com maior frequˆencia. O que os distingue dos problemas do t´opico anterior ´e que, neste caso, as vari´aveis de decis˜ao s˜ao discretas. Existem in´umeras situa¸c˜oes do mundo real que podem ser aproximadas por problemas de optimiza¸c˜ao combinat´oria, o que justifica a aten¸c˜ao da comunidade cient´ıfica. Alguns dos exemplos mais relevantes s˜ao o problema do caixeiro viajante, o problema do encaminhamento de ve´ıculos ou problemas de escalonamento (Jensen, 2004; Lewis, 2008; Pereira e Tavares, 2009; Tsai et al., 2004).

*•* Rob´otica evolutiva: Nesta ´area recorre-se a algoritmos de inspira¸c˜ao biol´ogica para evoluir a morfologia e comportamentos de robots (Floreano e Nolfi, 2004).

*•* Simula¸c˜ao: A capacidade de proceder `a evolu¸c˜ao/adapta¸c˜ao de um conjunto de entidades permite que os AGs sejam usados no desenvolvimento e estudo de modelos de simula¸c˜ao perten centes a diversos dom´ınios. Alguns exemplos relevantes s˜ao a economia (estudo da emergˆencia de mercados econ´omicos), o sistema imunit´ario (estudo das muta¸c˜oes som´aticas), a ecologia (es tudo de fen´omenos do tipo hospedeiro/parasita) ou os sistemas sociais (estudo da evolu¸c˜ao do comportamento social em col´onias de insectos) (Mitchell, 1996).

A lista apresentada n˜ao pretende, de modo nenhum, ser exaustiva. A consulta de actas das con ferˆencias mais relevantes nesta ´area, a *Genetic and Evolutionary Computation Conference (GECCO)* 22 ou o *IEEE Congress on Evolutionary Computation (CEC)*23 entre outras, pode ser ´util para se obter uma ideia geral dos desenvolvimentos recentes e para ajudar a perceber quais as ´areas de aplica¸c˜ao preferenciais para os AGs.

Programa¸c˜ao Gen´etica

Uma das aplica¸c˜oes mais interessantes de algoritmos de inspira¸c˜ao biol´ogica ´e a possibilidade de evo luir programas de computador que resolvam um determinado problema. A programa¸c˜ao gen´etica pode ser considerada uma descendente directa dos AGs. Embora tenham existido algumas tentativas anteriores para desenvolver um sistema com estas caracter´ısticas, o nascimento da ´area ´e normalmente associado `a publica¸c˜ao, em 1992, do livro *Genetic Programming* de Koza (1992). A quest˜ao central para a qual este trabalho procura obter respostas ´e a seguinte: ‘De que forma pode um computador aprender a resolver um determinado problema sem ser explicitamente programado para isso?”

Um algoritmo de programa¸c˜ao gen´etica evolui um conjunto de programas de computador ao longo de v´arias gera¸c˜oes. O objectivo ´e encontrar uma solu¸c˜ao (i.e., um programa) que resolva um de terminado problema. Tal como num AG, ´e necess´ario definir uma representa¸c˜ao para as solu¸c˜oes, uma fun¸c˜ao de aptid˜ao, um mecanismo de sele¸cc˜ao e operadores gen´eticos. As abordagens iniciais recorreram a ´arvores para representar programas de computador. Actualmente existem diversas re presenta¸c˜oes alternativas como, por exemplo, as lineares ou baseadas em grafos. O livro *A Field Guide to Genetic Programming* de Poli et al. (2008b) apresenta uma introdu¸c˜ao clara e completa da ´area da programa¸c˜ao gen´etica, inclu´ındo uma descri¸c˜ao de aplica¸c˜oes pr´aticas. Por sua vez, na p´agina *Human Competitive Results*24 ´e poss´ıvel consultar um conjunto solu¸c˜oes obtidas por programa¸c˜ao gen´etica que, de acordo com um conjunto de crit´erios, s˜ao compar´aveis ou melhores do que solu¸c˜oes propostas por humanos.

22 http://www.sigevo.org/ 23 http://www.ieee-cis.org/ 24 http://www.genetic-programming.com/humancompetitive.html

CAP´ITULO 2 

ALGORITMOS GENETICOS ´ 47

7. Conclus˜oes



Os AGs s˜ao m´etodos de resolu¸c˜ao de problemas inspirados em modelos simplificados da evolu¸c˜ao natural. O seu modo de actua¸c˜ao ´e extremamente simples de enunciar: um conjunto inicial de poss´ıveis solu¸c˜oes para um determinado problema (a popula¸c˜ao inicial) vai sendo sucessivamente modificado atrav´es da aplica¸c˜ao de operadores inspirados nos mecanismos da selec¸c˜ao natural e da varia¸c˜ao gen´etica (i.e., o conjunto de solu¸c˜oes vai evoluindo ao longo de sucessivas gera¸c˜oes). Neste cap´ıtulo descrevemos as componentes b´asicas que constituem um AG, nomeadamente o m´etodo de selec¸c˜ao de progenitores e os operadores de transforma¸c˜ao que d˜ao origem a novas solu¸c˜oes. Identific´amos ainda a defini¸c˜ao da representa¸c˜ao e da fun¸c˜ao de aptid˜ao como duas decis˜oes cruciais para o sucesso da optimiza¸c˜ao. Para tornar a exposi¸c˜ao o mais clara poss´ıvel, descrevemos, passo a passo, a aplica¸c˜ao de um AG a uma instˆancia do KSP.

Uma das caracter´ısticas que torna os AGs especialmente apelativos ´e a simplicidade do seu funcio namento. Considerando a met´afora biol´ogica como ponto de partida, ´e trivial entender e implementar as etapas associadas ao processamento iterativo da popula¸c˜ao de solu¸c˜oes. Esta simplicidade est´a relacionada com o facto de um AG simples ser um m´etodo de optimiza¸c˜ao global. A aplica¸c˜ao a um determinado problema n˜ao exige conhecimento particular sobre a situa¸c˜ao em causa, sendo suficiente definir um crit´erio de qualidade que avalie o m´erito relativo das solu¸c˜oes que s˜ao geradas. Esta ca racter´ıstica torna os AGs especialmente robustos e com capacidade para lidar com espa¸cos de procura de grande dimens˜ao, irregulares, descont´ınuos e multimodais. E importante, no entanto, referir que ´ estas caracter´ısticas n˜ao inviabilizam que um AG possa ser combinado com um algoritmo que possua propriedades complementares. Pelo contr´ario, abordagens h´ıbridas que combinem as caracter´ısticas globais de um AG com propriedades mais locais e/ou especializadas de outros m´etodos podem ser ex tremamente eficazes na resolu¸c˜ao de problemas dif´ıceis. Este t´opico ser´a abordado num dos cap´ıtulos deste manual.

(Página deixada propositadamente em branco)

49

CAP´ITULO 3

Estrat´egias Evolutivas

*Lino Costa ∗ Pedro Oliveira ∗∗*

*∗Departamento de Produ¸c˜ao e Sistemas*

*Universidade do Minho*

*∗∗Instituto de Ciˆencias Biom´edicas Abel Salazar*

*Universidade do Porto*

Na natureza, de acordo com a Teoria da Evolu¸c˜ao de Darwin, a evolu¸c˜ao dos seres vivos ocorre devido `a selec¸c˜ao natural e `a adapta¸c˜ao ao ambiente. A selec¸c˜ao natural faz com que os seres vivos mais aptos, em rela¸c˜ao ao ambiente, tenham maior probabilidade de sobrevivˆencia. Por outro lado, para que seja poss´ıvel a adapta¸c˜ao dos seres vivos a um ambiente continuamente em mudan¸ca, surgiram mecanismos na natureza que tornam poss´ıvel a diversidade, isto ´e, a existˆencia de seres vivos com caracter´ısticas pr´oprias que, potencialmente, os podem tornar mais aptos. A cont´ınua evolu¸c˜ao das esp´ecies pode ser vista como um processo de optimiza¸c˜ao consistindo na adapta¸c˜ao dos seres vivos ao seu meio ambiente. Este modelo biol´ogico inspirou o desenvolvimento de diversos Algoritmos Evolucion´arios1, na d´ecada de 60 do s´eculo XX, nomeadamente, na Alemanha, as Estrat´egias Evolutivas (Rechenberg, 1973; Schwefel, 1995) e, nos Estados Unidos da Am´erica, os Algoritmos Gen´eticos (Holland, 1975;

1 O termo Algoritmo Evolucion´ario ´e utilizado, genericamente, na descri¸c˜ao de sistemas computacionais para a resolu¸c˜ao de problemas que utilizam como elementos chave na sua implementa¸c˜ao, modelos computacionais baseados em mecanis mos de evolu¸c˜ao. De referir que estes mecanismos de evolu¸c˜ao correspondem ou relacionam-se com processos evolutivos biol´ogicos.

50 MANUAL DE COMPUTAC¸AO EVOLUTIVA E METAHEUR ˜ ´ISTICA

Goldberg, 1989) para serem aplicados a problemas de optimiza¸c˜ao2.

As Estrat´egias Evolutivas (EEs) foram desenvolvidas com o objectivo de serem aplicadas em op timiza¸c˜ao num´erica, revelando-se como algoritmos de optimiza¸c˜ao robustos e eficientes. As EEs s˜ao algoritmos iterativos que pesquisam com base em popula¸c˜oes de indiv´ıduos, que representam potenci ais solu¸c˜oes para o problema de optimiza¸c˜ao. Cada itera¸c˜ao ´e, por analogia com os sistemas biol´ogicos, chamada de *gera¸c˜ao*. Em cada gera¸c˜ao, os operadores gen´eticos (*recombina¸c˜ao* e *muta¸c˜ao*) actuam sobre a popula¸c˜ao de indiv´ıduos permitindo a explora¸c˜ao do espa¸co de procura e controlar a diversi dade dos indiv´ıduos da popula¸c˜ao. O princ´ıpio de sobrevivˆencia do mais apto ´e implementado pela *selec¸c˜ao* que garante a sobrevivˆencia dos indiv´ıduos que representam solu¸c˜oes de maior qualidade para o problema. A condi¸c˜oes que se devem verificar para terminar o processo iterativo (e.g., a obten¸c˜ao de uma solu¸c˜ao de qualidade suficiente para a resolu¸c˜ao do problema) constituem os chamados crit´erios de paragem. Um boa introdu¸c˜ao `as EEs, focando, com alguma profundidade, os fundamentos te´oricos, ´e feita por Beyer e Schwefel (2002).

1. Optimiza¸c˜ao em Espa¸cos Cont´ınuos



Um grande n´umero de aplica¸c˜oes de EEs a problemas de optimiza¸c˜ao com espa¸cos de procura dis cretos e cont´ınuos encontram-se descritas na literatura (Rechenberg, 1964, 1994; Schwefel, 1995; B¨ack, 1996). No entanto, as EEs ser˜ao aqui apresentadas como algoritmos para a resolu¸c˜ao de problemas de programa¸c˜ao n˜ao linear em espa¸cos de procura cont´ınuos (Nocedal e Wright, 1999):

min *f*(x) onde x *∈* Ω (3.1)

sujeito a

*gj* (x) *≥* 0 com *j* = 1*,...,m*

*hi*(x) = 0 com *i* = *m* + 1*,...,m* + *p*

onde x ´e o vector das *vari´aveis de decis˜ao*; *f*(x) ´e a *fun¸c˜ao objectivo*, i.e., a fun¸c˜ao que se pretende minimizar3; g(x) ´e o vector das *restri¸c˜oes do tipo desigualdade* que devem ser satisfeitas; h(x) ´e o vector das *restri¸c˜oes do tipo igualdade* que devem ser satisfeitas.

Na formula¸c˜ao apresentada existem *n* vari´aveis reais, *m* restri¸c˜oes do tipo desigualdade e *p* res tri¸c˜oes do tipo igualdade (o n´umero total de restri¸c˜oes ´e *m* + *p*). O *espa¸co das vari´aveis*, Ω *⊆ �n*, corresponde ao conjunto de todos os valores poss´ıveis para as vari´aveis de decis˜ao. A procura da solu¸c˜ao de um problema de optimiza¸c˜ao, o ponto *´optimo* x*∗*, ´e feita no espa¸co das vari´aveis4. As res tri¸c˜oes de desigualdade s˜ao expressas em termos de desigualdades do tipo maior ou igual (*≥*)5. Muitas vezes, poder˜ao existir restri¸c˜oes de desigualdade especificando limites inferiores (*xk*) e superiores (*xk*) para as *n* vari´aveis *xk*, i.e., *xk ≤ xk ≤ xk*, para *k* = 1*,...,n*. 

Qualquer ponto x *∈* Ω diz-se satisfazer uma restri¸c˜ao se, para essa restri¸c˜ao, o lado esquerdo da express˜ao calculado nesse ponto est´a de acordo com o lado direito, em termos do operador relacional.

2 De notar que estas duas abordagens n˜ao s˜ao as ´unicas inspiradas na natureza para resolver problemas de optimiza¸c˜ao, existindo outras, tais como, a Programa¸c˜ao Evolucion´aria (Fogel et al., 1966), a Programa¸c˜ao Gen´etica (Koza, 1992), os algoritmos de Col´onia de Formigas (Dorigo et al., 1996) e os algoritmos de Enxames de Part´ıculas (Kennedy e Eberhart, 1995).

3 De notar que qualquer problema formulado em termos de maximiza¸c˜ao de uma fun¸c˜ao objectivo *f*(x) pode ser refor mulado da seguinte forma: max *f*(x) = *−* min(*−f*(x)). 4 Em geral, os problemas de optimiza¸c˜ao s˜ao abordados pressupondo-se que o ponto ´optimo x*∗* existe, ´e ´unico, e pode ser localizado utilizando um algoritmo de optimiza¸c˜ao. Apesar de muitas vezes este ser o caso, existem situa¸c˜oes em que tal n˜ao se verifica: se *f*(x) n˜ao ´e limitada inferiormente, ent˜ao x*∗* n˜ao existe ou, para certas fun¸c˜oes objectivo, x*∗* poder´a n˜ao ser ´unico. 5 Qualquer restri¸c˜ao expressa em termos da desigualdade menor ou igual pode ser facilmente transformada em termos da desigualdade maior ou igual bastando para isso multiplic´a-la por –1.

CAP´ITULO 3 

ESTRATEGIAS EVOLUTIVAS ´ 51

Um ponto ´e dito *ponto admiss´ıvel* se todas as restri¸c˜oes do tipo desigualdade e restri¸c˜oes do tipo igualdade forem satisfeitas nesse ponto. O conjunto de todos os pontos admiss´ıveis constitui a *regi˜ao admiss´ıvel* e pode ser definido da seguinte forma:

*A* = *{*x *∈* Ω : g(x) *≥* 0 *∧* h(x)=0*} .*

Todos os pontos que n˜ao pertencem ao conjunto *A* s˜ao pontos n˜ao admiss´ıveis. O ´optimo x*∗*, a solu¸c˜ao do problema, pertence necessariamente `a regi˜ao admiss´ıvel6.

2. Caracter´ısticas Gerais



Existem diferen¸cas importantes relativamente `a forma como as EEs e os algoritmos de optimiza¸c˜ao tradicionais (n˜ao evolucion´arios) resolvem os problemas de optimiza¸c˜ao cont´ınua formulados em (3.1). Tal como as EEs, os algoritmos de optimiza¸c˜ao tradicionais s˜ao iterativos. Em geral, iniciam a procura a partir de uma aproxima¸c˜ao inicial ao ´optimo que se pretende encontrar. A partir da aproxima¸c˜ao inicial s˜ao geradas, sucessivamente, novas estimativas do ´optimo. O processo iterativo ´e repetido at´e que os crit´erios de paragem sejam verificados. Os diversos algoritmos de optimiza¸c˜ao distinguem-se pela forma como s˜ao calculadas as novas estimativas ao longo da procura. Quase todas as abordagens utilizam os valores da fun¸c˜ao objectivo, das restri¸c˜oes e, muitas vezes, das primeiras e segundas derivadas destas fun¸c˜oes. Alguns algoritmos, para calcular uma nova aproxima¸c˜ao ao ´optimo, utilizam apenas a informa¸c˜ao relativa `a aproxima¸c˜ao actual, enquanto que outros consideram, tamb´em, a informa¸c˜ao recolhida durante as itera¸c˜oes passadas. Os algoritmos de optimiza¸c˜ao tradicionais podem ser caracterizados, em grande parte, pela descri¸c˜ao anterior. No entanto, conv´em separar os algoritmos tradicionais em dois grupos principais: os m´etodos directos e os m´etodos baseados em gradientes. Os m´etodos directos guiam a procura com base em informa¸c˜ao relacionada com a fun¸c˜ao objectivo e/ou restri¸c˜oes. Enquanto que os m´etodos baseados em gradientes, para al´em desta informa¸c˜ao, necessitam, tamb´em, das primeiras e/ou segundas derivadas. Por este motivo, em geral, os m´etodos baseados em gradientes n˜ao s˜ao eficientes quando, num problema, n˜ao se verificam as condi¸c˜oes de diferenciabilidade e/ou continuidade das fun¸c˜oes. De salientar que, nestes m´etodos, os resultados obtidos dependem grandemente das aproxima¸c˜oes ao ´optimo consideradas no in´ıcio do processo iterativo e n˜ao s˜ao eficientes na resolu¸c˜ao de problemas com espa¸cos de procura de natureza discreta.

Em contraste com estas abordagens ditas tradicionais, as EEs:

*•* iniciam a procura a partir de uma popula¸c˜ao de potenciais solu¸c˜oes geradas de forma aleat´oria (caso seja conhecida podem partir de uma aproxima¸c˜ao inicial);

*•* trabalham, ao longo das gera¸c˜oes, com popula¸c˜oes de potenciais solu¸c˜oes, em vez de uma ´unica aproxima¸c˜ao ao ´optimo por itera¸c˜ao;

*•* n˜ao utilizam nenhuma informa¸c˜ao relativa `as primeiras e/ou segundas derivadas da fun¸c˜ao ob jectivo e/ou restri¸c˜oes;

*•* n˜ao exigem nenhuma condi¸c˜ao relativa `a continuidade e convexidade do espa¸co de procura;

*•* podem utilizar mecanismos que permitem encontrar m´ultiplos ´optimos locais (se existirem) numa unica execu¸ ´ c˜ao;

6 Quando um ponto satisfaz uma restri¸c˜ao *j* do tipo desigualdade, duas situa¸c˜oes podem ocorrer: o ponto est´a no limite da regi˜ao admiss´ıvel da restri¸c˜ao, i.e., *gj* (x) = 0 e neste caso a restri¸c˜ao *j* diz-se *activa*; ou o ponto est´a no interior da regi˜ao admiss´ıvel da restri¸c˜ao, i.e., *gj* (x) *>* 0 e neste caso a restri¸c˜ao *j* diz-se *inactiva*. Para qualquer ponto admiss´ıvel x, todas as *p* restri¸c˜oes do tipo igualdade est˜ao activas.

52 MANUAL DE COMPUTAC¸AO EVOLUTIVA E METAHEUR ˜ ´ISTICA

*•* guiam a procura com base em mecanismos e/ou regras probabil´ısticas (estoc´asticas).

Estas caracter´ısticas das EEs fazem com que sejam particularmente eficientes na resolu¸c˜ao de problemas reais, onde os espa¸cos de procura podem n˜ao ser convexos e, por isso, muitas vezes contˆem um grande n´umero de ´optimos locais e/ou mais do que uma solu¸c˜ao ´optima. Por outro lado, muitos problemas reais apresentam espa¸cos de procura de natureza discreta onde n˜ao ´e poss´ıvel garantir as condi¸c˜oes de diferenciabilidade e continuidade desej´aveis para muitos dos algoritmos tradicionais.

3. Nomenclatura



As EEs foram desenvolvidas, inicialmente, por Rechenberg (1973), com o objectivo de resolver problemas de optimiza¸c˜ao, tendo como base as estruturas e os processos de optimiza¸c˜ao que ocorrem na natureza. Mais tarde, Schwefel (1981) desenvolveu novos esquemas evolutivos com base nos mesmos princ´ıpios.

Nas EEs ´e utilizada uma popula¸c˜ao com um determinado n´umero de indiv´ıduos. Em cada gera¸c˜ao, a partir dos indiv´ıduos presentes na popula¸c˜ao, os *progenitores*, s˜ao gerados novos indiv´ıduos, os *descendentes*. Desta forma, ao longo das gera¸c˜oes, novas popula¸c˜oes s˜ao sucessivamente geradas. As EEs trabalham directamente com a representa¸c˜ao real das vari´aveis de decis˜ao, pelo que cada indiv´ıduo ´e um vector de n´umeros reais representando uma potencial solu¸c˜ao para o problema de optimiza¸c˜ao.

Desde o seu aparecimento, tem vindo a ser utilizada uma nomenclatura pr´opria para designar as diferentes EEs. Esta nomenclatura ´e baseada no n´umero de progenitores, no n´umero de descendentes e no tipo de selec¸c˜ao considerado. O n´umero de progenitores ´e designado por *μ* e o n´umero de descendentes por *λ*. Por outro lado, dois tipos de selec¸c˜ao foram originalmente descritos e designados por selec¸c˜ao ’+’ e selec¸c˜ao ’,’. A primeira e mais simples EE, desenvolvida por Rechenberg (1973), onde a selec¸c˜ao era feita sobre uma popula¸c˜ao de dois membros, i.e., *μ* + *λ* = 1 + 1, ´e designada, na nomenclatura atr´as apresentada, por EE-(1 + 1). Posteriormente, o mesmo autor desenvolveu uma estrat´egia multimembros mais complexa, onde a selec¸c˜ao era feita sobre uma popula¸c˜ao de *μ >* 1 indiv´ıduos e um descendente; i.e., *μ* + *λ* = *μ* + 1, que ´e designada por EE-(*μ* + 1).

De uma forma mais gen´erica, numa EE-(*μ*+*λ*), numa determinada gera¸c˜ao, existe uma popula¸c˜ao de *μ* progenitores que gera *λ* descendentes por muta¸c˜ao (Figura 3.1). Em seguida, os *μ* + *λ* indiv´ıduos s˜ao avaliados e ordenados de acordo com os seus valores da fun¸c˜ao objectivo7. Finalmente, o processo de selec¸c˜ao faz com que apenas os *μ* melhores de todos os *μ*+*λ* indiv´ıduos se tornem os progenitores da gera¸c˜ao seguinte, i.e., a selec¸c˜ao ´e feita a partir dos *μ*+*λ* indiv´ıduos. De referir que este mecanismo de selec¸c˜ao ´e determin´ıstico, uma vez que apenas os melhores indiv´ıduos s˜ao seleccionados para formarem a popula¸c˜ao de progenitores da gera¸c˜ao seguinte.

Um outro modelo conceptual pode ser definido onde, numa determinada gera¸c˜ao, uma popula¸c˜ao de *μ* progenitores gera, por muta¸c˜ao, *λ* descendentes (assumindo que *λ>μ*). Em seguida, os *λ* descendentes s˜ao avaliados e ordenados de acordo com os seus valores da fun¸c˜ao objectivo. Ent˜ao, os *μ* melhores dos *λ* descendentes gerados tornam-se os progenitores da pr´oxima gera¸c˜ao, i.e., os *μ* progenitores n˜ao s˜ao inclu´ıdos no processo de selec¸c˜ao. Esta EE que utiliza a selec¸c˜ao ’,’ ´e designada por EE-(*μ, λ*) (Figura 3.2). Os dois tipos de EEs atr´as descritas, a EE-(*μ* + *λ*) e a EE-(*μ, λ*), diferem basicamente no procedimento de selec¸c˜ao.

Originalmente, as EEs baseavam-se num ´unico operador, a muta¸c˜ao, para gerar novos indiv´ıduos. Posteriormente, foi introduzido um outro operador, a recombina¸c˜ao, que era aplicado juntamente com a muta¸c˜ao (Schwefel, 1995). Mais adiante, nesta mesma sec¸c˜ao, descrever-se-´a detalhadamente este operador.

7 Assumindo que os indiv´ıduos representam pontos admiss´ıveis. Mais tarde, ser˜ao abordados diversos mecanismos de tratamento de restri¸c˜oes.

Geração

actual

μ *Progenitores* λ *Descendentes*

CAP´ITULO 3 

ESTRATEGIAS EVOLUTIVAS ´ 53

Geração

seguinte

μ*+*λ *Descendentes* μ *Progenitores *



Figura 3.1: Aspecto Geral da Estrat´egia Evolutiva (*μ* + *λ*)

Geração

actual

Geração seguinte

μ *Progenitores* λ *Descendentes*

λ *Descendentes* μ *Progenitores *



Figura 3.2: Aspecto Geral da Estrat´egia Evolutiva (*μ, λ*)

54 MANUAL DE COMPUTAC¸AO EVOLUTIVA E METAHEUR ˜ ´ISTICA

Em seguida, cada uma das EEs atr´as referidas ir´a ser descrita com maior detalhe. Todos os algoritmos apresentados s˜ao formulados para problemas de minimiza¸c˜ao sem restri¸c˜oes. Mais adiante, ir-se-´a focar diversos esquemas de tratamento de restri¸c˜oes quer do tipo desigualdade quer do tipo igualdade.

4. Estrat´egia Evolutiva (1+1)



Este esquema evolutivo com apenas 2 membros, designado por EE-(1 + 1), foi proposto por (Re chenberg, 1964) para a resolu¸c˜ao de problemas de optimiza¸c˜ao. Numa determinada gera¸c˜ao, existe apenas um progenitor (*μ* = 1) e um ´unico descendente (*λ* = 1), e a selec¸c˜ao tem lugar apenas entre estes dois membros.

O funcionamento da EE-(1 + 1) pode ser descrito pelo Algoritmo 1. A procura inicia-se a partir de um ponto inicial x0 (uma aproxima¸c˜ao ao ´optimo). Em seguida, um novo ponto x*N* ´e gerado por muta¸c˜ao atrav´es da adi¸c˜ao de uma quantidade aleat´oria normal com m´edia 0 e variˆancia *σ*2. O desvio padr˜ao *σ* est´a relacionado com o tamanho do passo e vai sendo adaptado ao longo da procura. Depois, os dois pontos s˜ao comparados e o melhor (com menor valor da fun¸c˜ao objectivo e satisfazendo todas as restri¸c˜oes) ´e seleccionado para progenitor da pr´oxima gera¸c˜ao. Este processo ´e repetido at´e que o crit´erio de paragem seja verificado.

De seguida, vai-se analisar com mais detalhe cada um dos passos envolvidos neste algoritmo.

Algoritmo 1 Estrat´egia Evolutiva (1 + 1)

Require: x0, Δ*x*

1: *t ←* 0

2: x(*t*) *←* x0 [ onde x(*t*) = (*x*1*, x*2*,...,xn*)(*t*) ]

3: *σ ← |*Δ*x|* 

*√n*

4: para *i* = 1 to 10*n* fa¸ca

5: *sucessos*[*i*] *←* 0

6: fim para

7: enquanto CP falso fa¸ca [ *//* ´e o resto da divis˜ao inteira ]

8: se (*t//n* = 0 *∧ t ≥* 10*n*) ent˜ao [ Regra de 1/5 sucessos ]

9: *Ps ←*

10 *n i*=1

*sucessos*[*i*] 10*n*

10: *σ ←* 11: fim se

⎧⎨ 

*σ/c* se *Ps >* 1*/*5

*cσ* se *Ps <* 1*/*5 com *c* = 0*.*85 ⎩

*σ* se *Ps* = 1*/*5

12: x*N ←* x(*t*) + N(0*, σ*2) [ Muta¸c˜ao ]

13: se (*f*(x*N* ) *< f*(x(*t*))) ent˜ao [ Selec¸c˜ao ]

14: x(*t*+1) *←* x*N*

15: *sucessos*[(*t* + 1)*//*10*n*] *←* 1

16: sen˜ao

17: x(*t*+1) *←* x(*t*)

18: *sucessos*[(*t* + 1)*//*10*n*] *←* 0

19: fim se

20: *t ← t* + 1

21: fim enquanto

22: x*f inal ←* x(*t*)

23: retorno x*f inal*

**

CAP´ITULO 3 

ESTRATEGIAS EVOLUTIVAS ´ 55

Aproxima¸c˜ao Inicial

Como j´a foi dito, para iniciar a procura ´e necess´aria uma aproxima¸c˜ao inicial ao ´optimo x0. Mas, para al´em disso, ´e necess´ario escolher os valores iniciais para o desvio padr˜ao *σ*. O valor inicial t´ıpico para o desvio padr˜ao *σ* pode ser expresso pela seguinte equa¸c˜ao:

*σ* = Δ*x √n,*

onde Δ*x* ´e uma medida aproximada da distˆancia esperada da aproxima¸c˜ao inicial x0 ao ´optimo e *n* ´e a dimens˜ao do problema (o n´umero de vari´aveis de decis˜ao).

Ao longo da procura, devem-se garantir as seguintes duas condi¸c˜oes para os valores dos tamanhos do passo *σ*:

1. *σ >* 0 para *i* = 1*,...,n*;

2. os valores de *σ* devem ser suficientemente grandes para que pelo menos o d´ıgito menos signifi cativo da vari´avel *xi* seja alterado.

As condi¸c˜oes anteriores podem ser escritas em termos dos seguintes limites inferiores para os tamanhos do passo: *σ ≥ ε*1 e *σ ≥ ε*2 *|xi|* para *i* = 1*,...,n*, onde *ε*1 e *ε*2 dependem da precis˜ao do computador utilizado, sendo *ε*1 *>* 0 e *ε*2 *>* 0.

Muta¸c˜ao

O operador de muta¸c˜ao consiste em gerar um novo ponto xN atrav´es da adi¸c˜ao de uma quantidade aleat´oria z. A quantidade aleat´oria z traduz o tamanho do passo (ou comprimento do deslocamento), e a sua escolha deve ser feita de tal forma que pequenos deslocamentos ocorram frequentemente e grandes deslocamentos ocorram raramente. Com este prop´osito, em geral, as quantidades aleat´orias z s˜ao geradas de acordo com uma distribui¸c˜ao Normal. Para al´em disso, as seguintes condi¸c˜oes devem ser impostas `a distribui¸c˜ao dos tamanhos do passo:

1. O valor esperado dos componentes *zi* de z, com *i* = 1*,...,n*, deve ser nulo, i.e., *E*[*zi*] = 0; 2. As variˆancias *σ*2, com *i* = 1*,...,n*, devem ter valores pequenos.

Logo, os componentes aleat´orios *zi* podem ser calculados de acordo com uma distribui¸c˜ao Normal com m´edia nula e variˆancia *σ*2, i.e., *zi ∼ N*(0*, σ*2). Para determinar n´umeros aleat´orios normais a partir de n´umero aleat´orios uniformes, podem ser utilizadas as regras de transforma¸c˜ao de Box e Muller (1958)8.

Controlo do Tamanho do Passo

Para que o processo de optimiza¸c˜ao seja eficiente, os deslocamentos devem ser continuamente modifi cados. Se os deslocamentos forem demasiado pequenos, o n´umero de itera¸c˜oes do processo de procura ´e desnecessariamente grande; pelo contr´ario, se forem demasiado grandes, poder´a n˜ao se obter uma boa aproxima¸c˜ao ao ´optimo, ou mesmo, o processo poder´a n˜ao convergir. Por isso, em todas as es trat´egias de optimiza¸c˜ao, o controlo do tamanho do passo ´e uma das componentes mais importantes no processo de procura.

8 De acordo com estas regras, dois n´umeros aleat´orios independentes normalmente distribu´ıdos com m´edia nula e variˆancia unit´aria podem ser calculados partir de quaisquer dois n´umeros aleat´orios gerados uniformemente no inter

valo [0*,* 1] da seguinte forma:*x*1 =  *−*2 ln(*y*1) sin(2*πy*2) e *x*2 =  *−*2 ln(*y*1) cos(2*πy*2) onde *y*1 e *y*2 s˜ao os dois n´umeros aleat´orios gerados uniformemente no intervalo [0*,* 1] e, *x*1 e *x*2 s˜ao os dois n´umeros aleat´orios normalmente distribu´ıdos com m´edia nula e variˆancia unit´aria. Para se obter dois n´umeros aleat´orios *z*1 e *z*2 normalmente distribu´ıdos com m´edia nula e variˆancia *σ*2, basta multiplicar *x*1 e *x*2 pelo desvio padr˜ao *σ*, i.e., *z*1 = *σx*1 e *z*2 = *σx*2.

56 MANUAL DE COMPUTAC¸AO EVOLUTIVA E METAHEUR ˜ ´ISTICA x2

)k( x

)k( ϕ

)1k( x +

\* x

)k( r

)1k( r +

x1

Figura 3.3: Raz˜ao de Progresso

Nas EEs, a grandeza das quantidades aleat´orias *zi* depende do desvio padr˜ao *σ*. Quanto maior for o desvio padr˜ao *σ*, maiores ser˜ao as quantidades aleat´orias *zi* e, consequentemente, os tamanhos do passo. Rechenberg (1973), baseando-se na aplica¸c˜ao da EE-(1 + 1) a alguns tipos de problemas de optimiza¸c˜ao, formulou a chamada ”Regra de 1*/*5 de Sucessos” para controlar os tamanhos do passo. Esta regra pode ser descrita da seguinte forma:

”De tempos a tempos, ao longo do processo de procura, calcule-se a frequˆencia de sucessos, i.e., a raz˜ao entre o n´umero de sucessos e o n´umero de itera¸c˜oes. Se a raz˜ao for maior que 1*/*5, aumente-se a variˆancia, se for menor que 1*/*5, diminua-se a variˆancia.”

Assumindo-se que esta regra ´e aplicada periodicamente todas as Δ*t* itera¸c˜oes, a sua express˜ao pode ser definida da seguinte forma para uma itera¸c˜ao *k*:

*σ*(*k*+1) =

⎧⎨

*cinc σ*(*k*) se *Ps*(Δ*t*) *>* 1*/*5 *cdec σ*(*k*) se *Ps*(Δ*t*) *<* 1*/*5

*σ*(*k*) se *Ps*(Δ*t*)=1*/*5*,* ⎩

onde *Ps*(Δ*t*) ´e a propor¸c˜ao de sucessos nas ´ultimas Δ*t* itera¸c˜oes e, *cdec <* 1 e *cinc >* 1 s˜ao, respectiva mente, os coeficientes de diminui¸c˜ao e de aumento do desvio padr˜ao *σ*.

Defina-se a raz˜ao de progresso como o valor esperado da diferen¸ca radial ap´os uma itera¸c˜ao *k*

(Figura 3.3):

*ϕ*(*k*) = *E*����x*∗ −* x(*k*)��� *−*���x*∗ −* x(*k*+1)����= *E*�*r*(*k*) *− r*(*k*+1)�*,*

onde x*∗* representa o ´optimo e, x(*k*) e *r*(*k*) representam, respectivamente, a aproxima¸c˜ao ao ´optimo e a distˆancia ao ´optimo na itera¸c˜ao *k*.

Verifica-se que, para muitos problemas, a regra de 1*/*5 de sucessos mostra-se extremamente eficiente na manuten¸c˜ao, aproximadamente, da maior raz˜ao de progresso poss´ıvel para o ´optimo. No entanto, interessa definir qual a frequˆencia com que o crit´erio de sucesso deve ser testado (o valor do parˆametro Δ*t*) e qual o factor de diminui¸c˜ao ou de aumento dos desvios padr˜ao mais eficiente (os valores dos coeficientes *cdec* e *cinc*). Para tentar responder a esta quest˜ao, podem-se utilizar os resultados obtidos por Rechenberg (1973). A m´axima raz˜ao de progresso ´e dada por:

*ϕ*max = *c*1*r*(*k*) 

*n* sendo *c*1 *∼*= 0*.*2025*,*

com variˆancia comum *σ*2, cujo valor *σ*2*opt* ´optimo ´e dado por:

�

*σ*2*opt* =

*c*2*r*(*k*) *n* 

�2

com *c*2 *∼*= 1*.*224*,*

CAP´ITULO 3 

ESTRATEGIAS EVOLUTIVAS ´ 57

para todos os componentes *z*(*k*)

*i* do vector aleat´orio z(*k*). Nestas express˜oes, *r*(*k*) ´e a distˆancia actual

ao ´optimo e *n* ´e o n´umero de vari´aveis. Destas equa¸c˜oes pode-se obter a rela¸c˜ao para as varia¸c˜oes nos comprimentos dos deslocamentos ap´os uma gera¸c˜ao, supondo a condi¸c˜ao de m´axima raz˜ao de convergˆencia:

= *r*(*k*+1) 

*r*(*k*) = *r*(*k*) *− ϕ*max

*r*(*k*) = 1 *− c*1*n ,*

ou ap´os *n* gera¸c˜oes:

*σ*(*k*+1)

*opt*

*σ*(*k*)

*opt*

*σ*(*k*+*n*)

*opt*

*σ*(*k*) *opt*

=

1 *− c*1*n* *n.*

Quando *n* for muito maior que 1, o factor do comprimento do deslocamento tende para uma

constante:

lim *n→*+*∞*

1 *− c*1*n* *n*= *e−c*1 *∼*= 0*.*817 *∼*= 1 1*.*224 *.* 

Este resultado aplica-se ao caso limite em que existem muitas vari´aveis (*n* grande) e expressa que a raz˜ao de progresso ´e inversamente proporcional ao n´umero de vari´aveis. O facto da raz˜ao de progresso perto do seu m´aximo ser bastante insens´ıvel a pequenas varia¸c˜oes das variˆancias, juntamente com o facto da probabilidade de sucesso s´o poder ser determinada pela m´edia de diversas gera¸c˜oes, conduz `a seguinte formula¸c˜ao, mais precisa, da regra de 1*/*5 de sucessos onde *n* ´e o n´umero de vari´aveis de decis˜ao do problema de optimiza¸c˜ao:

”Ap´os cada *n* gera¸c˜oes, verifique-se quantos sucessos ocorreram nas ´ultimas 10*n* gera¸c˜oes. Se este n´umero for inferior a 2*n*, multipliquem-se os comprimentos dos deslocamentos pelo factor 0*.*85; sen˜ao, dividam-se os comprimentos dos deslocamentos pelo factor 0*.*85.”

Esta regra permite que os comprimentos dos deslocamentos ou as variˆancias dos deslocamentos aleat´orios sejam controlados, fixando-se os seguintes valores *cdec* = 0*.*85, *cinc* = 1*/*0*.*85 e Δ*t* = 10*n*. No entanto, a probabilidade de sucesso n˜ao fornece nenhuma indica¸c˜ao de qu˜ao apropriadas s˜ao as raz˜oes das variˆancias *σ*2*i* relativamente umas `as outras (para cada vari´avel), pelo que os comprimentos dos deslocamentos s´o podem ser todos reduzidos em conjunto ou todos aumentados em conjunto. Por este motivo, em geral, considera-se que *σ*21 = *σ*22 = *...* = *σ*2*n* = *σ*2.

Crit´erio de Paragem

O crit´erio de paragem determina quando o processo de procura deve ser terminado. Em geral, s˜ao utilizadas as condi¸c˜oes:

*f*(x(*k−*Δ*k*)) *− f*(x(*k*)) *≤ ε*3 ou*f*(x(*k−*Δ*k*)) *− f*(x(*k*)) 

*f*(x(*k*))*≤ ε*4*,*

e Δ*k ≥* 20*n*, onde *ε*3 e *ε*4 dependem da precis˜ao do computador utilizado, sendo *ε*3 *>* 0 e *ε*4 *>* 0. A condi¸c˜ao Δ*k ≥* 20*n* assegura que, no caso extremo, entre testes da regra de 1*/*5 de sucessos, os comprimentos dos deslocamentos s˜ao reduzidos ou aumentados, pelo menos pelo factor (0*.*85)*±*20 *∼*= (25)*∓*1, o que evita que a procura seja terminada apenas porque os comprimentos dos deslocamentos s˜ao for¸cados a variar muito frequentemente. Por outro lado, ´e evidente que o crit´erio de convergˆencia n˜ao precisa de ser verificado em todas as gera¸c˜oes. O procedimento usual ´e test´a-lo apenas cada 20*n* gera¸c˜oes.

58 MANUAL DE COMPUTAC¸AO EVOLUTIVA E METAHEUR ˜ ´ISTICA

5. Estrat´egias Evolutivas Multimembros



Nestas EEs multimembros existem, em cada gera¸c˜ao, *μ* progenitores e *λ* descendentes. A EE-(*μ, λ*) distingue-se da EE-(*μ*+*λ*) apenas na selec¸c˜ao que tem lugar, no primeiro caso, entre os *λ* descendentes enquanto que, no segundo caso, se faz a partir dos *μ* + *λ* indiv´ıduos (Schwefel, 1995).

Caso seja conhecida uma aproxima¸c˜ao ao ´optimo, a procura pode ser iniciada gerando *μ* pontos a partir desse ponto inicial. Caso contr´ario, a popula¸c˜ao inicial pode ser gerada de forma aleat´oria. Em cada gera¸c˜ao, *λ* pontos s˜ao gerados por muta¸c˜ao, como resultado da adi¸c˜ao de n´umeros aleat´orios normais z. O desvio padr˜ao da distribui¸c˜ao normal utilizada na gera¸c˜ao das quantidades aleat´orias normais tem um papel crucial numa vez que est´a relacionado com os deslocamentos ocorridos ao longo da procura. V´arias regras podem ser adoptadas com o prop´osito de adaptar estes desvios padr˜ao. Depois, os *λ* pontos gerados por muta¸c˜ao s˜ao ordenados de acordo com os seus valores da fun¸c˜ao objectivo. Finalmente, os *μ* melhores pontos s˜ao seleccionados para serem os progenitores na gera¸c˜ao seguinte. Este processo ´e repetido at´e que o crit´erio de paragem seja satisfeito.

De seguida, s˜ao apresentados mais detalhes relativos a cada um dos passos do procedimento atr´as descrito. Ser˜ao, tamb´em, apresentados v´arios algoritmos correspondentes a diferentes variantes de estrat´egias evolutivas multimembros.

Aproxima¸c˜ao inicial

De forma semelhante `a EE-(1+1), a procura pode iniciar-se a partir de uma aproxima¸c˜ao ao ´optimo x0, que permitir´a gerar os *μ* pontos da popula¸c˜ao inicial9. Caso, n˜ao seja conhecida nenhuma aproxima¸c˜ao ao ´optimo, a popula¸c˜ao inicial pode ser gerada de forma aleat´oria. Para al´em disso, ´e necess´ario escolher os valores iniciais para parˆametros associados ao esquema de auto-adapta¸c˜ao considerado10. Caso se conhe¸ca uma estimativa da distˆancia ao ´optimo, os valores iniciais t´ıpicos para o desvio ou os desvios padr˜ao podem ser expressos pela mesma equa¸c˜ao indicada para EE-(1 + 1).

Muta¸c˜ao

O operador de muta¸c˜ao consiste em gerar novos pontos pela adi¸c˜ao de quantidades aleat´orias, tal como acontecia na EE-(1 + 1). Contudo, nas EEs multimembros, cada progenitor produz, em m´edia, *λ/μ* descendentes, de tal forma que *λ* descendentes sejam gerados. As mesmas condi¸c˜oes acerca da normalidade das quantidades aleat´orias z s˜ao consideradas neste tipo de estrat´egias. No entanto, dependendo do tipo de controlo de passo usado para fazer a auto-adapta¸c˜ao, as quantidades aleat´orias normais s˜ao calculadas de diferentes formas como ´e descrito mais adiante.

Recombina¸c˜ao

Schwefel (1995) notou uma acelera¸c˜ao consider´avel na procura, bem como a facilita¸c˜ao da adapta¸c˜ao dos desvios padr˜ao pela introdu¸c˜ao do operador de recombina¸c˜ao nas EEs. Basicamente, a recom bina¸c˜ao consiste em, antes da muta¸c˜ao, recombinar um conjunto de progenitores por forma a encontrar uma nova solu¸c˜ao. Seja *ρ* o n´umero de progenitores que participam na recombina¸c˜ao (1 *≤ ρ ≤ μ*). Estes *ρ* progenitores s˜ao escolhidos aleatoriamente de entre os *μ* indiv´ıduos da popula¸c˜ao. Quando *ρ* = 1 ent˜ao n˜ao existe recombina¸c˜ao. Logo, a nomenclatura das EEs pode ser estendida, e as EEs com recombina¸c˜ao s˜ao, em geral, referidas por EE-(*μ/ρ* + *λ*) ou EE-(*μ/ρ, λ*). Existem dois tipos de recombina¸c˜ao principais:

9 Pode-se utilizar a aproxima¸c˜ao ao ´optimo como a m´edia da distribui¸c˜ao de probabilidade para gerar os restantes pontos da popula¸c˜ao inicial, e.g., a m´edia da distribui¸c˜ao normal. 10 Estes parˆametros poder˜ao ser o desvio padr˜ao inicial *σ* comum a todas as vari´aveis, ou os desvios padr˜ao iniciais *σi* distintos para cada vari´avel ou ainda as rota¸c˜oes de direc¸c˜oes de procura relacionadas com a matriz de covariˆancias.

CAP´ITULO 3 

ESTRATEGIAS EVOLUTIVAS ´ 59

*•* a Recombina¸c˜ao Interm´edia, e

*•* a Recombina¸c˜ao Discreta.

Estes tipos de recombina¸c˜ao podem ser aplicados quer `as vari´aveis de decis˜ao quer aos desvios padr˜ao *σ*. Em seguida, estes tipos de recombina¸c˜ao s˜ao descritos em termos das vari´aveis de decis˜ao. A sua aplica¸c˜ao aos desvios padr˜ao *σ* faz-se de forma similar.

Recombina¸c˜ao Interm´edia

Neste tipo de recombina¸c˜ao, os componentes dos descendentes s˜ao obtidos atrav´es da m´edia dos componentes dos progenitores correspondentes (escolhidos aleatoriamente a partir da popula¸c˜ao). Logo, considerando *ρ* progenitores escolhidos aleatoriamente, os descendentes x*D* ser˜ao dados por:

x*D* = 1*ρ* *ρ m*=1

x*m.* (3.2)

Se *ρ* = *μ*, este esquema tende a gerar descendentes pr´oximos do centr´oide da popula¸c˜ao.

Recombina¸c˜ao Discreta

Neste tipo de recombina¸c˜ao, cada componente dos descendentes ´e escolhido aleatoriamente a partir de um dos *ρ* progenitores. Logo, supondo *ρ* progenitores escolhidos aleatoriamente, os descendentes x*D* ser˜ao dados por:

x*D* = (*xu*1*,*1*,...,xun,n*) com *u*1 *∈ U*(0*, ρ*)*,...,un ∈ U*(0*, ρ*)*,* (3.3)

onde *ui*, para *i* = 1*,...,n*, s˜ao ´ındices gerados aleatoriamente (uniformemente), indicando a partir de que progenitor (dos *ρ* progenitores) o valor da vari´avel de decis˜ao ´e copiado para o descendente. Este procedimento permite obter diferentes combina¸c˜oes dos valores das vari´aveis de decis˜ao a partir das solu¸c˜oes existentes na popula¸c˜ao.

Controlo do Tamanho do Passo

Os desvios padr˜ao podem ser actualizados de diversas formas. Este processo consiste na adapta¸c˜ao dos parˆametros da estrat´egia durante a procura (Schwefel, 1995). As implementa¸c˜oes mais conhecidas s˜ao:

*•* a Adapta¸c˜ao Isotr´opica,

*•* a Adapta¸c˜ao N˜ao Isotr´opica, e

*•* a Adapta¸c˜ao N˜ao Isotr´opica com Rota¸c˜ao

Em seguida, descreve-se sucintamente cada um destes esquemas de actualiza¸c˜ao dos desvios padr˜ao.

Adapta¸c˜ao Isotr´opica

Neste esquema, tal como na EE-(1 + 1), ´e considerada uma variˆancia *σ*2 comum a todas as vari´aveis, i.e., *σ*21 = *σ*22 = *...* = *σ*2*n*. O desvio padr˜ao comum *σ* s˜ao agora actualizados pela equa¸c˜ao: *σ*(*k*+1) = *σ*(*k*)*ez* com *z ∼ N*(0*,* Δ2*σ*)*,* (3.4)

onde *z* ´e determinado de acordo com uma distribui¸c˜ao Normal com m´edia zero e variˆancia Δ2*σ*, onde Δ*σ* ´e um parˆametro do algoritmo.

60 MANUAL DE COMPUTAC¸AO EVOLUTIVA E METAHEUR ˜ ´ISTICA



Figura 3.4: Adapta¸c˜ao Isotr´opica

A Figura 3.4 ilustra o funcionamento deste esquema de adapta¸c˜ao para um problema com duas vari´aveis de decis˜ao. Nesta figura, todos os pontos pertencentes `a circunferˆencia centrada no ponto *x* correspondem a pontos com igual probabilidade de serem gerados por muta¸c˜ao.

Em geral, o valor de Δ*σ* ´e fixado em 1*/√n*.Uma vez que apenas existe uma ´unica variˆancia comum *σ*2, cada indiv´ıduo da popula¸c˜ao possui a seguinte estrutura:

(x; *σ*)=(*x*1*, x*2*,...,xn*; *σ*)*.*

Por sua vez, a muta¸c˜ao ´e dada por:

*xi* = *xi* + *zi* com *zi ∼ N*(0*, σ*)*.* (3.5)

Algoritmo 2 Estrat´egia Evolutiva (*μ/ρ* +*, λ*) com Adapta¸c˜ao Isotr´opica

Require: *μ*, *ρ*, *λ*, Δ*σ*

1: *t ←* 0

2: P(*t*) *←*(x(*t*)

*m* ; *σ*(*t*); *f*(x(*t*)

*m* )) : *m* = 1*,...,μ*

3: enquanto CP falso fa¸ca

4: para *l* = 1 to *λ* fa¸ca

5: g*l ← agrupar*(P(*t*)*, ρ*)

6: *σl ← recombinars*(g*l*)

7: x*l ← recombinarx*(g*l*)

8: *σl ← mutars*(*σl*)

9: x*l ← mutarx*(x*l, σl* )

10: fim para

[ onde x(*t*)

*m* = (*x*1*, x*2*,...,xn*)(*t*) ]

11: D(*t*) *←*(x(*t*) *m* ; *σ*(*t*) *m* ; *f*(x(*t*) *m* )) : *m* = 1*,...,μ*

12: P(*t*+1) *←* 13: *t ← t* + 1

selec¸c˜ao(D(*t*)) se *EE −* (*μ/ρ, λ*) selec¸c˜ao(D(*t*) *∪* P(*t*)) se *EE −* (*μ/ρ* + *λ*)

14: fim enquanto

15: x*f inal ← melhor*(P(*t*))

16: retorno x*f inal*

**

CAP´ITULO 3 

ESTRATEGIAS EVOLUTIVAS ´ 61



Figura 3.5: Adapta¸c˜ao N˜ao Isotr´opica

O Algoritmo 2 ´e o algoritmo da EE-(*μ/ρ* +*, λ*) com adapta¸c˜ao isotr´opica. Neste algoritmo, a fun¸c˜ao *agrupar* selecciona aleatoriamente *ρ* indiv´ıduos da popula¸c˜ao de progenitores P. As fun¸c˜oes *recombinars* (para os desvios padr˜ao) e *recombinarx* (para as vari´aveis de decis˜ao) recombinam, res pectivamente, os *ρ* indiv´ıduos de acordo com a equa¸c˜ao (3.2) ou a equa¸c˜ao (3.3). A fun¸c˜ao *mutars* muta os desvios padr˜ao aplicando a regra isotr´opica expressa na equa¸c˜ao (3.4). Finalmente, a fun¸c˜ao *mutarx* corresponde `a muta¸c˜ao das vari´aveis de decis˜ao como ´e descrita na equa¸c˜ao (3.5). O algoritmo devolve a solu¸c˜ao correspondente ao indiv´ıduo da popula¸c˜ao final com menor valor da fun¸c˜ao objectivo (a melhor aproxima¸c˜ao ao ´optimo).

Adapta¸c˜ao N˜ao Isotr´opica

Neste esquema, os desvios padr˜ao *σi* s˜ao actualizados (um para cada vari´avel de decis˜ao) de acordo

com a equa¸c˜ao:

*σ*(*k*+1)

*i* = *σ*(*k*)

*i eziez,* (3.6)

onde *zi ∼ N*(0*,* Δ2*σ*), z *∼ N*(0*,* Δ2*σ*‘) e, Δ*σ* e Δ*σ*‘ s˜ao parˆametros do algoritmo. Em geral, os valores

de Δ*σ* e Δ*σ*‘ s˜ao fixados, respectivamente, em 1*/* 2*√n* e 1*/√*2*n*. Este esquema n˜ao isotr´opico, ao contr´ario do anterior, permite adaptar os desvios padr˜ao independentemente para cada vari´avel de decis˜ao.

A Figura 3.5 ilustra o funcionamento deste esquema de adapta¸c˜ao para um problema com duas vari´aveis de decis˜ao. Neste figura, os pontos pertencentes `a elipse s˜ao os descendentes com igual probabilidade de serem gerados a partir de um ponto *x*.

Neste esquema existe agora uma variˆancia associada a cada vari´avel *σ*2*i* , pelo que cada indiv´ıduo da popula¸c˜ao passa a possuir a seguinte estrutura:

(x; *σ*)=(*x*1*,...,xn*; *σ*1*,...,σn*)*.*

Para fazer a muta¸c˜ao, aplica-se a seguinte equa¸c˜ao:

*xi* = *xi* + *zi* com *zi ∼ N*(0*, σi*)*.* (3.7)

O Algoritmo 3 ´e o algoritmo da EE-(*μ/ρ* +*, λ*) com adapta¸c˜ao n˜ao isotr´opica. Tal como ante riormente, neste algoritmo, as fun¸c˜oes *agrupar*, *recombinars* e *recombinarx* permitem seleccionar aleatoriamente *ρ* indiv´ıduos da popula¸c˜ao de progenitores P e recombin´a-los de acordo com a equa¸c˜ao

62 MANUAL DE COMPUTAC¸AO EVOLUTIVA E METAHEUR ˜ ´ISTICA

Algoritmo 3 Estrat´egia Evolutiva (*μ/ρ* +*, λ*) com Adapta¸c˜ao N˜ao Isotr´opica

Require: *μ*, *ρ*, *λ*, Δ*σ*, Δ*σ*

1: *t ←* 0

2: P(*t*) *←*(x(*t*)

*m* ; s(*t*)

*m* ; *f*(x(*t*)

*m* ) : *m* = 1*,...,μ*

]

3: enquanto CP falso fa¸ca

4: para *l* = 1 to *λ* fa¸ca

5: g*l ← agrupar*(P(*t*)*, ρ*)

6: s*l ← recombinars*(g*l*)

7: x*l ← recombinarx*(g*l*)

8: s*l ← mutars*(s*l*)

9: x*l ← mutarx*(x*l, σl* )

10: fim para

[ onde x(*t*)

*m* = (*x*1*, x*2*,...,xn*) e s(*t*)

*m* = (*σ*1*, σ*2*,...,σn*)

11: D(*t*) *←*(x(*t*) *m* ; s(*t*) *m* ; *f*(x(*t*) *m* )) : *m* = 1*,...,μ*

12: P(*t*+1) *←* 13: *t ← t* + 1

selec¸c˜ao(D(*t*)) se *EE −* (*μ/ρ, λ*) selec¸c˜ao(D(*t*) *∪* P(*t*)) se *EE −* (*μ/ρ* + *λ*)

14: fim enquanto

15: x*f inal ← melhor*(P(*t*))

16: retorno x*f inal*

**

(3.2) ou a equa¸c˜ao (3.3). Contudo, a fun¸c˜ao *mutars* muta os desvios padr˜ao aplicando agora a regra n˜ao isotr´opica expressa na equa¸c˜ao (3.6). Finalmente, a fun¸c˜ao *mutarx* corresponde `a muta¸c˜ao das vari´aveis de decis˜ao tal como ´e descrita na equa¸c˜ao (3.7) de forma independente para vari´avel de decis˜ao.

Adapta¸c˜ao N˜ao Isotr´opica com Rota¸c˜ao

Neste esquema, para modelar a poss´ıvel correla¸c˜ao entre as vari´aveis de decis˜ao s˜ao consideradas *n*(*n −* 1)*/*2 rota¸c˜oes. Os desvios padr˜ao *σi* e as rota¸c˜oes *αj* s˜ao actualizadas de acordo com a equa¸c˜ao:

*σ*(*k*+1)

*i* = *σ*(*k*)

*i eziez*

*α*(*k*+1)

*j* + *βN*(0*,* 1)*,* (3.8)

*j* = *α*(*k*)

onde *zi ∼ N*(0*,* Δ2*σ*), z *∼ N*(0*,* Δ2*σ*‘) e, Δ*σ* e Δ*σ*‘, e *β* s˜ao parˆametros do algoritmo. Em geral, os

valores de Δ*σ* e Δ*σ*‘ s˜ao fixados, respectivamente, em 1*/* 2*√n*, 1*/√*2*n* e *β* = 5. Este esquema n˜ao isotr´opico com rota¸c˜oes, ao contr´ario do anterior, permite adaptar os desvios padr˜ao tendo em conta a poss´ıvel existˆencia de correla¸c˜oes entre as vari´aveis de decis˜ao. Neste esquema existe agora uma variˆancia associada a cada vari´avel *σ*2*i* e rota¸c˜oes *αj* , pelo que cada indiv´ıduo da popula¸c˜ao passa a possuir a seguinte estrutura:

(x; *σ*; *α*)=(*x*1*,...,xn*; *σ*1*,...,σn*; *α*1*,...,αr*)

com *r* = *n*(*n −* 1)*/*2.

A Figura 3.6 ilustra o funcionamento deste esquema de adapta¸c˜ao para um problema com duas vari´aveis de decis˜ao. Neste figura, os pontos pertencentes `a elipse sujeita a rota¸c˜ao s˜ao os descendentes com igual probabilidade de serem gerados a partir de um ponto *x*.

CAP´ITULO 3 

ESTRATEGIAS EVOLUTIVAS ´ 63



Figura 3.6: Adapta¸c˜ao N˜ao Isotr´opica com Rota¸c˜ao

A muta¸c˜ao ´e feita de acordo com a seguinte equa¸c˜ao:

x= x + z com z *∼ N*(0*,* C)

C=

⎧⎨

*σ*2*i* se *i* = *j*

12 (*σ*2*i − σ*2*j* ) tan(2*αij* ) se *i* e *j* correlacionados ⎩ 

0 se *i* e *j* n˜ao correlacionados

*.* (3.9)

O Algoritmo 4 ´e o algoritmo da EE-(*μ/ρ* +*, λ*) com adapta¸c˜ao n˜ao isotr´opica e com rota¸c˜ao. As fun¸c˜oes *agrupar*, *recombinars* e *recombinarx* s˜ao semelhantes `as dos algoritmos anteriores. No en tanto, a fun¸c˜ao *mutars* muta os desvios padr˜ao aplicando agora a regra n˜ao isotr´opica com rota¸c˜oes expressa na equa¸c˜ao (3.8). Finalmente, a fun¸c˜ao *mutarx* corresponde `a muta¸c˜ao das vari´aveis de decis˜ao tal como ´e descrita na equa¸c˜ao (3.9) tendo em conta as covariˆancias, i.e., as associa¸c˜oes entre vari´aveis de decis˜ao.

Crit´erio de Paragem

O crit´erio de convergˆencia permite definir o t´ermino do processo de procura. Em geral, o crit´erio de convergˆencia adoptado para as estrat´egias evolutivas multimembros ´e terminar a procura quando as seguintes condi¸c˜oes s˜ao verificadas:

���*f*(*k*) max *− f*(*k*)

��� *≤ ε*3 ou

���

���*f*(*k*) max *− f*(*k*) min

min

���*f*(*k*)���*≤ ε*4*,*

min e *f*(*k*) s˜ao, respectivamente, o maior e o menor valor da fun¸c˜ao objectivo e a m´edia 

onde *f*(*k*) max, *f*(*k*)

dos valores da fun¸c˜ao objectivo da popula¸c˜ao de *μ* progenitores, *ε*3 e *ε*4 dependem da precis˜ao do computador utilizado, sendo *ε*3 *>* 0 e *ε*4 *>* 0.

6. Tratamento das Restri¸c˜oes



Os esquemas evolutivos, tal como foram apresentados, permitem resolver problemas de optimiza¸c˜ao sem restri¸c˜oes. E poss´ ´ ıvel tratar problemas restri¸c˜oes de desigualdade do tipo *gj* (x) *≥* 0 com *j* = 1*,...,m*, utilizando um mecanismo de elimina¸c˜ao das solu¸c˜oes n˜ao admiss´ıveis, i.e., em cada gera¸c˜ao,

64 MANUAL DE COMPUTAC¸AO EVOLUTIVA E METAHEUR ˜ ´ISTICA

Algoritmo 4 Estrat´egia Evolutiva (*μ/ρ* +*, λ*) com Adapta¸c˜ao N˜ao Isotr´opica com Rota¸c˜ao

Require: *μ*, *ρ*, *λ*, Δ*σ*, Δ*σ* , *β*

1: *t ←* 0

2: P(*t*) *←*(x(*t*)

*m* ; s(*t*)

*m* ; a(*t*)

*m* ; *f*(x(*t*)

*m* ) : *m* = 1*,...,μ*

(*σ*1*, σ*2*,...,σn*) e a(*t*)

[ onde x(*t*)

*m* = (*x*1*, x*2*,...,xn*), s(*t*)

*m* =

*m* = (*α*1*, α*2*,...,αk*) com *k* = *n*(*n −* 1)*/*2 ]

3: enquanto CP falso fa¸ca

4: para *l* = 1 to *λ* fa¸ca

5: g*l ← agrupar*(P(*t*)*, ρ*)

6: (s*l,* a*l*) *← recombinars*(g*l*)

7: x*l ← recombinarx*(g*l*)

8: (s*l,* a*l*) *← mutars*(s*l,* a*l*)

9: x*l ← mutarx*(x*l,* s*l,* a*l*)

10: fim para

11: D(*t*) *←*(x(*t*) *m* ; s(*t*) *m* ; a(*t*) *m* ; *f*(x(*t*) *m* )) : *m* = 1*,...,μ*

12: P(*t*+1) *←* 13: *t ← t* + 1

selec¸c˜ao(D(*t*)) se *EE −* (*μ/ρ, λ*) selec¸c˜ao(D(*t*) *∪* P(*t*)) se *EE −* (*μ/ρ* + *λ*)

14: fim enquanto

15: x*f inal ← melhor*(P(*t*))

16: retorno x*f inal*

**

se a muta¸c˜ao ou recombina¸c˜ao gerarem um ponto que n˜ao satisfaz as restri¸c˜oes, este n˜ao ´e aceite. Desta forma, a procura restringe-se ao interior da regi˜ao admiss´ıvel. No entanto, ´e muitas vezes dif´ıcil especificar um vector inicial x0 que seja uma aproxima¸c˜ao ao ´optimo e que satisfa¸ca todas as restri¸c˜oes (se for utilizado um vector inicial que n˜ao satisfa¸ca as restri¸c˜oes poder´a demorar muito tempo at´e que o processo de procura determine um ponto da regi˜ao admiss´ıvel). Um dos processos para a determina¸c˜ao de um vector inicial na regi˜ao admiss´ıvel ´e o descrito por Box (1965). Neste processo, uma fun¸c˜ao objectivo auxiliar *r*(x) representando a soma dos valores das fun¸c˜oes das restri¸c˜oes violadas

´e constru´ıda:

*r*(x) =  *m j*=1

*gj* (x)*δj* (x)*,*

onde

*δj* (x) =

*−*1 se *gj* (x) *<* 0 0 se *gj* (x) *≥* 0 *.*

Um decr´escimo no valor da fun¸c˜ao *r*(x) representa uma aproxima¸c˜ao `a regi˜ao admiss´ıvel. Assim, quando *r*(x) = 0 ent˜ao x satisfaz todas as restri¸c˜oes e pode ser utilizado como vector inicial. En quanto n˜ao ´e encontrado um ponto admiss´ıvel, a procura ´e feita com base na fun¸c˜ao *r*(x) definida anteriormente.

O esquema atr´as descrito permite tratar apenas restri¸c˜oes do tipo desigualdade. As restri¸c˜oes do tipo igualdade *hi*(x) = 0 podem ser reformuladas como restri¸c˜oes de desigualdade da seguinte forma:

*hi*(x) *≥* 0 *∧ −hi*(x) *≥* 0*.*

Para al´em disso, podem ser consideradas quantidades positivas e pequenas, *ε* e *ε* de tal forma que



*−ε ≤ hi*(*x*) *≤ ε*, i.e.:

*hi*(x) + *ε ≥* 0 *∧ −hi*(x) + *ε ≥* 0*.*

CAP´ITULO 3 

ESTRATEGIAS EVOLUTIVAS ´ 65

No entanto, o tratamento de restri¸c˜oes do tipo igualdade pode criar dificuldades `as EEs, dado que encontrar novos pontos que sejam admiss´ıveis pode traduzir-se num comportamento oscilat´orio. Outras t´ecnicas de tratamento de restri¸c˜oes podem ser implementadas. Coello Coello (2002) faz uma revis˜ao, no contexto dos algoritmos evolucion´arios, de diversas t´ecnicas para o tratamento de restri¸c˜oes do tipo desigualdade e igualdade.

7. Estrat´egias Evolutivas Avan¸cadas



Nesta sec¸c˜ao s˜ao referidas algumas abordagens baseadas em EEs que utilizam t´ecnicas avan¸cadas quer para a adapta¸c˜ao dos parˆametros de procura quer pela utiliza¸c˜ao de informa¸c˜ao de procura n˜ao local.

Uma dessas abordagens s˜ao as Meta-EEs propostas por Rechenberg (1994) (tamb´em, chamadas de *Nested Evolution Strategies*). As Meta-EEs utilizam informa¸c˜ao n˜ao local na procura. Para isso, a procura ´e hierarquicamente organizada em diversas EEs que fazem procura local por um per´ıodo de *γ* gera¸c˜oes. H´a uma EE externa cujas popula¸c˜oes s˜ao constitu´ıdas por diversas EEs internas. Assim, a nota¸c˜ao usada para descrever uma Meta-EE passa a ser EE-[*μ/ρ*+*, λ*(*μ/ρ* +*, λ*)*γ*] onde os parˆentesis rectos dizem respeito `a EE externa e os curvos `as EEs internas. Neste algoritmo, h´a *μ*popula¸c˜oes de progenitores de EE-(*μ/ρ* +*, λ*) que geram *λ*EE-(*μ/ρ* +*, λ*) descendentes. Ap´os *γ* gera¸c˜oes, a selec¸c˜ao correspondente `a EE externa ´e feita para determinar as melhores *μ*das EE-(*μ/ρ* +*, λ*) que servir˜ao de base para a pr´oxima popula¸c˜ao de *λ*EE-(*μ/ρ* +*, λ*) descendentes. As Meta-EEs podem ser utilizadas em optimiza¸c˜ao global, multi-local, problemas inteiros mistos e, tamb´em, na optimiza¸c˜ao dos pr´oprios parˆametros de procura das EEs internas.

Outras abordagens, s˜ao as propostas por Ostermeier et al. (1994) and Hansen e Ostermeier (2001), que utilizam informa¸c˜ao do progresso ao longo das gera¸c˜oes para fazer a auto-adapta¸c˜ao dos parˆametros de procura. No algoritmo proposto por Ostermeier et al. (1994), designado por CSA-ES (*Cumula tive Step-size Adaptation Evolution Strategy*), informa¸c˜ao relativa ao percurso evolutivo percorrido ao longo das gera¸c˜oes ´e utilizada para adaptar os tamanhos do passo. No algoritmo CMA-ES (*Covariance Matrix Adaptation Evolution Strategy*) desenvolvido por Hansen e Ostermeier (2001), o percurso evo lutivo cumulativo ´e tamb´em utilizado para adaptar a matriz de covariˆancias que permite a aplica¸c˜ao de muta¸c˜ao correlacionada. O estudo e a descri¸c˜ao aprofundada destas estrat´egias evolutivas avan¸cadas sai fora do ˆambito deste cap´ıtulo, podendo o leitor encontrar os detalhes destas propostas nas re ferˆencias bibliogr´aficas atr´as indicadas.

(Página deixada propositadamente em branco)

67

CAP´ITULO 4

Programa¸c˜ao Gen´etica

*Douglas A. Augusto Helio J. C. Barbosa*

*Laborat´orio Nacional de Computa¸c˜ao Cient´ıfica*

*Laboratório Nacional de Computação Científica, Petrópolis, RJ*

Este cap´ıtulo descreve a metaheur´ıstica evolucion´aria denominada *programa¸c˜ao gen´etica*, que visa a constru¸c˜ao autom´atica de programas de computador por meio de um processo iterativo inspirado na evolu¸c˜ao por sele¸c˜ao natural.

A programa¸c˜ao gen´etica (PG), cujo desenvolvimento ´e atribu´ıdo a Koza (1992), ´e uma me taheur´ıstica estoc´astica de otimiza¸c˜ao global baseada no princ´ıpio darwiniano de sele¸c˜ao natural, sendo, pois, uma abordagem da computa¸c˜ao evolucion´aria. A PG destina-se `a evolu¸c˜ao de programas de computador em linguagens arbitr´arias, em outras palavras, a PG otimiza estruturas funcionais ca pazes de realizar opera¸c˜oes, sejam elas l´ogicas, aritm´eticas, condicionais e de desvios, que normalmente mapeiam *entradas* em *sa´ıdas*.

Nesta concep¸c˜ao, submete-se uma popula¸c˜ao contendo um determinado n´umero de indiv´ıduos— programas candidatos criados aleatoriamente—ao processo simulado de evolu¸c˜ao segundo a sele¸c˜ao natural. Neste processo iterativo figuram a cada gera¸c˜ao: a sele¸c˜ao de indiv´ıduos promissores para procria¸c˜ao, a forma¸c˜ao de seus descendentes por meio de opera¸c˜oes gen´eticas como cruzamento e muta¸c˜ao, e finalmente a inser¸c˜ao destes novos indiv´ıduos na popula¸c˜ao, marcando-se o in´ıcio de uma nova gera¸c˜ao. Muito embora n˜ao existam garantias de progresso e tampouco de obten¸c˜ao de solu¸c˜oes ´otimas, como caracter´ıstico em qualquer metaheur´ıstica, essa dinˆamica tende, ao longo das gera¸c˜oes, a produzir solu¸c˜oes candidatas incrementalmente mais adaptadas.

Destacam-se como caracter´ısticas t´ıpicas da programa¸c˜ao gen´etica as seguintes qualidades: