Aprendizagem e Busca Local em Algoritmos Meméticos para Projeto Assistido por Computador

Frederico Gadelha Guimarães

Tese submetida à
Escola de Engenharia
Universidade Federal de Minas Gerais
para obtenção do título de Doutor em Engenharia Elétrica

Local Learning and Search in Memetic Algorithms for Computer Aided Design

Frederico Gadelha Guimarães

A thesis submitted to
Escola de Engenharia
Universidade Federal de Minas Gerais
for the degree of Doctor in Electrical Engineering

I would like to dedicate this thesis to my parents and my wife,
those whom I love so much,
and who have been loving me in return.

Aprendizagem e Busca Local em Algoritmos Meméticos para Projeto Assistido por Computador

Resumo

O projeto assistido por computador (PAC) é um processo de projeto automatizado, caracterizado pela associação de um modelo matemático e computacional do dispositivo a ser otimizado e uma técnica de busca automática, um método de otimização, adequada para encontrar os valores ótimos para os parâmetros de projeto. Entretanto, este processo de PAC automatizado frequentemente requer muitas consultas ao programa de análise até que um protótipo que satisfaça as especificações e requisitos definidos pelo projetista seja encontrado. Como o programa de análise geralmente utiliza um método numérico caro do ponto de vista computacional, é importante que o método de otimização realize sua tarefa com o mínimo de consultas a esse programa.

Nas últimas décadas, algoritmos evolucionários, uma classe especial de métodos de busca global, tem se estabelecido como ferramentas poderosas na solução de problemas de PAC. Porém, recentemente os algoritmos meméticos ou algoritmos híbridos têm recebido uma atenção crescente por parte dos pesquisadores devido ao seu potencial de superar o desempenho de algoritmos evolucionários em diversos contextos. Algoritmos meméticos em geral se beneficiam de operadores de busca local que são acrescidos aos operadores usuais de algoritmos evolucionários. Contudo, no contexto de otimização com variáveis contínuas em PAC, o custo computacional de operadores de busca local torna seu uso proibitivo. Este custo pode ser reduzido com o emprego de técnicas de aproximação de funções, dessa forma, tornando os algoritmos meméticos ferramentas de uso prático em PAC.

Os principais objetivos desta tese são investigar, implementar e testar o uso de aproximações locais em um arcabouço de algoritmos meméticos para a solução de problemas de PAC, em particular o projeto de dispositivos eletromagnéticos. Esta tese pretende contribuir para a investigação e desenvolvimento dos algoritmos meméticos, com foco especial em problemas cuja avaliação de funções envolve cálculos complexos e computacionalmente caros. Neste contexto, a complexidade adicional no algoritmo seria claramente justificável.

Com este propósito, este trabalho organiza, desenvolve e implementa um arcabouço para otimização evolucionária dedicada a problemas de PAC. Este arcabouço acomoda diversos algoritmos evolucionários disponíveis na literatura para otimização mono e multi-objetivo, incluindo é claro os algoritmos meméticos. Estes algoritmos são unificados em sua implementação através da identificação de uma estrutura comum. Cada algoritmo implementado pode utilizar a busca local baseada em aproximações proposta nesta tese. Este operador utiliza a técnica de interpolação multiquádrica para construir uma aproximação local para cada função objetivo e de restrição não linear do problema, e o método de programação quadrática sequencial para resolver a versão local do problema global. Para o operador multi-objetivo, um passo adicional deve ser feito, no qual o problema multi-objetivo é transformado em um problema mono-objetivo com restrições adicionais através da formulação de realização de metas.

Esta tese também apresenta uma análise formal de algoritmos meméticos. Esta análise é dividida em duas partes. A primeira parte investiga o efeito do operador de busca local nas propriedades de convergência global de algoritmos evolucionários típicos por meio da teoria de cadeias de Markov. A segunda parte estuda o custo computacional de algoritmos meméticos. Nesta parte, a complexidade computacional dos operadores de busca local é obtida e expressões para a carga extra da busca local são desenvolvidas. Esta análise conduziu aos seguintes resultados importantes sobre a metodologia de busca local proposta: (i) O algoritmo memético preserva as propriedades de convergência global de algoritmos evolucionários padrão. Quando o algoritmo memético emprega a busca local Lamarckiana e um operador de mutação irredutível, será globalmente convergente se o número de indivíduos selecionados para a busca local for menor do que o tamanho da população, i.e., se $\sigma < \mu$. (ii) O algoritmo memético preserva a complexidade polinomial de algoritmos evolucionários padrão. A complexidade da busca local é dominada por um termo que é quadrático com N_L, o número máximo de pontos no conjunto de dados local. (iii) a carga extra da busca local pode ser reduzida se a busca local não for usada em toda geração, o que permite que se empregue valores mais elevados para σ e n_{TR}, o

número de atualizações da região de confiança. Este é o compromisso entre a frequência e a intensidade da busca local.

No total seis problemas são discutidos. Três funções analíticas de teste sem restrições; um problema magnetostático irrestrito, o projeto da forma do pólo de um dispositivo magnetizador; e duas versões de um problema magnetostático restrito, o conhecido problema de teste 22. As três funções analíticas de teste são usadas para ilustrar o aumento na velocidade de convergência devido à busca local baseada na aproximação multiquádrica. O problema do magnetizador mostra que a carga extra imposta pela busca local baseada em aproximações é insignificante quando tratamos funções computacionalmente caras. As funções analíticas são rápidas de avaliar, logo o algoritmo memético consumiu mais tempo do que o algoritmo genético, mas esta situação se inverte claramente quando o tempo de avaliação aumenta. A versão mono-objetivo do problema 22 foi usada aqui como um exemplo representativo de problemas de PAC em eletromagnetismo aplicado. Doze algoritmos evolucionários diferentes são combinados com a busca local baseada na aproximação multiquádrica. O experimento mostra que todos os algoritmos evolucionários testados se beneficiaram do uso do operador de busca local. Esta é uma evidência empírica que, para a classe de problemas delimitada nesta tese, os algoritmos evolucionários em geral podem ser bastante melhorados com a sua associação com os operadores de busca local baseados em aproximações. A versão bi-objetivo do problema 22 foi usada para ilustrar o operador de busca local multi-objetivo baseado em aproximação multiquádrica. Os resultados também mostram que todos os três algoritmos meméticos multi-objetivo tiveram um desempenho melhor do que suas versões genéticas. A busca local melhorou não somente a velocidade de convergência medida pela métrica S-metric, mas melhorou também a qualidade dos conjuntos obtidos, considerando as métricas NDCSR e S-metric.

Local Learning and Search in Memetic Algorithms for Computer Aided Design

Abstract

The computer aided design (CAD) process is an automated system characterized by the association of a mathematical and computational model of the device being designed and an adequate automatic search technique, an optimization method, for finding the optimal values for the design variables. However, this automated CAD process often requires many calls to the analysis software until the optimized prototype satisfies the specifications and requirements defined by the designer. Since the analysis software usually employs a computationally intensive numerical method, it is desirable that the optimization method finds an acceptable solution while minimizing the number of calls to this software.

Evolutionary algorithms, a special class of global-search methods, have been recognized as powerful tools for the solution of CAD problems for quite some time, but in recent years, memetic algorithms or hybrid algorithms have received growing attention due to their potential to improve typical evolutionary algorithms in many different contexts. Memetic algorithms in general benefit from the association of local search operators with the usual reproductive operators of evolutionary algorithms. However, in the context of optimization with continuous variables within CAD problems, the computational cost of local search operators is not affordable. This cost can be greatly reduced with the employment of approximation techniques within these operators, making the use of memetic algorithms for CAD problems more practical.

The main objectives of this thesis are to investigate, implement and test the use of local approximations within a memetic optimization framework for the solution of CAD

problems, particularly the design of electromagnetic devices. This thesis aims to further advance the development and investigation of memetic algorithms, with particular attention to problems whose function evaluations involve complex and intensive calculations. In this context, the additional complexity in the algorithm would be justifiable.

This thesis organizes, develops and implements a framework for evolutionary optimization applied to CAD problems. This framework accomodates various evolutionary algorithms for both mono and multi-objective optimization, including memetic algorithms. These algorithms are unified in the framework thanks to the identification of a common structure for modeling them. Any evolutionary algorithm in this framework can employ the approximation-based local search proposed in this thesis. This operator utilizes the multiquadric interpolation technique for building a local approximation for each one of the nonlinear objective and constraint functions, and the sequential quadratic programming method to solve a local version of the global problem. For the multi-objective operator an additional step is taken to transform the multi-objective problem into a mono-objective problem with additional constraints via the goal attainment formulation.

This thesis also presents a formal analysis of memetic algorithms. This analysis is divided in two main parts. The first part investigates the effect of the local search operator on the global convergence properties of standard evolutionary algorithms via Markov chain theory. The second part studies the computational cost of memetic algorithms. In this second part, the computational complexity of the local search operators is derived and expressions for the overhead of the local search are developed. This analysis leads to the following important results about the proposed local search methodology: (i) The memetic algorithm preserves the global convergence properties of standard evolutionary algorithms. When the memetic algorithm employs Lamarckian local search and an irreducible mutation operator, it will be globally convergent if the number of individuals selected for local search is smaller than the population size, i.e., if $\sigma < \mu$. (ii) The memetic algorithm preserves the polynomial complexity of standard evolutionary algorithms. The complexity of the local search is dominated by a term that is quadratic with N_L, the maximum number of points in the local data set. (iii) The overhead of the local search can be reduced by not using the local search at every generation, which allows one to employ higher values for σ and n_{TR}, the number of trust region updates. That is the balance between frequency and intensity of the local search.

In total six problems are discussed. Three analytical test functions without constraints; one unconstrained magnetostatic problem, the design of the pole shape of a

magnetizer device; and two versions of a constrained magnetostatic problem, the well known benchmark problem 22. The three analytical test functions are used to illustrate the increase in convergence speed due to the multiquadric-based local search. The magnetizer problem shows the negligible overhead of the approximation-based local search when dealing with computationally intensive functions. The analytical functions are fast to evaluate, hence the memetic algorithm takes more time than the genetic algorithm, but this situation clearly inverts when the time to evaluate increases. The mono-objective version of the problem 22 is used here as a representative instance of CAD problems in electromagnetic design. Twelve different evolutionary algorithms from the framework are hybridized with the MQ-based local search. The experiment shows that all tested evolutionary algorithms benefit from the use of the local search operator. This is an empirical evidence that, for the class of problems delimited in this thesis, evolutionary algorithms in general can be greatly improved by their association with approximation-based local search operators. The bi-objective version of problem 22 is used to illustrate the multi-objective multiquadric-based local search operator. The results also show that all three multi-objective memetic algorithms perform better than their basic genetic counterparts. The local search not only improves the convergence speed measured by the S-metric, but also improves the quality of the outcome sets, regarding the metrics NDCSR and S-metric.

Acknowledgements

We have few opportunities in life to express our gratitude to those who were particularly important in our achievements. For sure, this is one of these opportunities, and this is the right place.

First and foremost, I thank to my supervisor, professor Jaime Arturo Ramírez. Six years ago, when I was still doing my undergraduation, I entered his room asking about the "scientific initiation" program. Since then, it has been a long and challenging road, but, at that time, I have never imagined that I would have arrived so far. During these years he motivated me and helped me a lot, even when I was in Canada facing some unexpected events... I would like to praise his patience and kindness with his students. I am also grateful to my co-supervisor, professor David Alister Lowther, who accepted me as his student at McGill University, during my stay in Montreal, thus allowing me to experience a year that will be always *dans mes memoirs*. He and his wife Irene were very kind and receptive with me and my wife. In summary, Ramírez and Lowther's guidance, experience and support were very important to my formation as a researcher.

I also thank to my professors in the Graduate Program in Electrical Engineering of the Universidade Federal de Minas Gerais, in particular, the professors Reinaldo Martinez Palhares, Renato Cardoso Mesquita, Ricardo Hiroshi Caldeira Takahashi, and Rodney Rezende Saldanha, who contributed to this work to some extent and to my education to a great extent. Since the beginning of my research, I have had one or more opportunities to work with them in subjects more or less related to the main subject of this thesis, and I have learned a lot from these opportunities. To the secretaries of the Graduate Program in Electrical Engineering, Anete Vieira and Arlete Vieira, who were always kind and helpful with me and others.

To my friends and colleagues in the Group of Optimization and Computer Aided Design (GOPAC), at Universidade Federal de Minas Gerais, especially Elizabeth Wanner,

with whom I had the satisfaction of having excellent discussions. I think we did fine pieces of work together and hopefully we will have other chances to do more! Also to my new friends and colleagues in the Computer Aided Design Laboratory (CADLab), at McGill University, especially Helder Pinheiro, Prakash Paul, Amir Hajiaboli, and Jun Ouyang, who have made a friendly and enjoyable environment to do research. Special thanks to my good friend (and fellow) Felipe Campelo at Hokkaido University, Japan. In spite of being half a world away, we always had interesting and productive discussions, resulting in a solid partnership. Thanks to Internet and the Yahoo! Instant Messenger program that shortened distances.

The construction of the knowledge enclosed in this text costs time, effort and, above all, demands patience from our beloved relatives and friends. Without their understanding and incentive, no project is possible. Considering the subject of this thesis, I could not help thanking my parents for the genes and memes they have given me, and of course, for their love and support. I do not have enough room here to adequately express my gratitude for everything they have done during so many years to enable me to be in the position of writing this text. My deepest gratitude to my wife Lívia Fares. Her love and friendship were paramount. I would say that she supported this thesis in all definitions of the term.

Finally, I must thank the support given by the National Council of Scientific and Technologic Development – CNPq – under grants 141731/2004-4 and 201041/2005-7.

Contents

"Thus the task is not so much to see what no one yet has seen,
but to think what nobody yet has thought about that which everybody sees."
— Arthur Schopenhauer, 1788–1860

List of Algorithms

List of Figures

List of Tables

Nomenclature

d	Number of optimization variables
m	Number of objective functions
n	Iteration (generation) counter
n_L	Number of consecutive iterations (generations) performed without local search
n_{TR}	Number of trust region updates
p	Number of inequality constraints
q	Number of equality constraints
x_k	Optimization variable
λ	Offspring population size
μ	Parent population size
ξ	Maximum archive size
N_L	Maximum number of points in the local data set
T	Number of selected competitors for tournament selection
$\mathbf{x}$	Vector of optimization variables
$\mathbf{f}(\,\cdot\,)$	Vector of objective functions
$\mathbf{g}(\,\cdot\,)$	Vector of inequality constraint functions
$\mathbf{h}(\,\cdot\,)$	Vector of equality constraint functions
$\mathbf{C}$	Transition matrix associated to the crossover step
$\mathbf{G}$	Transition matrix associated to the global search
$\mathbf{H}$	Transition matrix associated to the hybrid search
$\mathbf{L}$	Transition matrix associated to the local search

$\mathbf{M}$	Transition matrix associated to the mutation step
$\mathbf{S}$	Transition matrix associated to the selection step
$\mathbf{T}$	Transition matrix of a general Markov chain
s_i	State i in the state space
X_n	Stochastic process
$\mathscr{P}\{\cdot\}$	Probability of an event
$\mathscr{E}\{\cdot\}$	Mathematical Expectance
$\mathcal{A}$	Set of algorithms
$\mathcal{D}$	Global data set
$\mathcal{F}$	Set of all functions that map a given search space into a given objective space
$\mathcal{G}_i$	Set of points that satisfy the ith inequality constraint
$\mathcal{H}_j$	Set of points that satisfy the jth equality constraint
$\mathcal{L}$	Local data set
$\mathcal{S}$	State space of a Markov chain
$\mathcal{X}$	Search space defined by the lower and upper limits of each variable
$\mathcal{X}^*$	Set of optimal solutions in the search space
$\mathcal{Y}$	Objective space defined by the mapping of the search space by the vector of objective functions
$\mathcal{Y}^*$	Set of optimal solutions in the objective space
Ω	Set of feasible points in the search space
$A(n)$	Archive population of an evolutionary algorithm at generation n
$P(n)$	Population of an evolutionary algorithm at generation n
$\Phi_A(n)$	Fitness values for the population $A(n)$
$\Upsilon_A(n)$	Penalized values for the population $A(n)$
$\lvert\cdot\rvert$	Cardinality of a set, if the argument is a set; or absolute value if the argument is a scalar quantity
$O(\cdot)$	Order of the number of operations

$\langle T \rangle$ Averaged total time consumed by the algorithm to converge to the solution set

$\langle N \rangle$ Averaged number of generations required by the algorithm to converge to the solution set

Resumo expandido

Introdução

O projeto assistido por computador (PAC) é um processo de projeto automatizado que utiliza um modelo matemático e computacional do dispositivo ou sistema a ser projetado. O processo de PAC parte de um conjunto de especificações e requisitos, que levam à definição de um primeiro protótipo. Esses passos iniciais dependem do conhecimento e da intervenção humana. O processo é automatizado a partir da geração de um modelo computacional do protótipo, cuja descrição está intimamente relacionada com o método de análise a ser empregado, e sua associação com técnicas de busca automática, isto é, métodos de otimização adequados. Se o protótipo otimizado satisfaz as especificações definidas pelo projetista, então o processo se encerra e a solução final de projeto é alcançada.

Entretanto, este processo de PAC automatizado frequentemente requer muitas consultas ao programa de análise até que um protótipo que satisfaça as especificações e requisitos definidos pelo projetista seja encontrado. Como o programa de análise geralmente utiliza um método numérico caro do ponto de vista computacional, é importante que o método de otimização realize sua tarefa com o mínimo de consultas a esse programa. Esse é um objetivo importante ao se projetar técnicas de otimização para a classe de problemas em PAC.

Nas últimas décadas, algoritmos evolucionários, uma classe especial de métodos de busca global, têm se estabelecido como ferramentas poderosas na solução de problemas de PAC. Porém, recentemente os algoritmos meméticos ou algoritmos híbridos têm recebido uma atenção crescente por parte dos pesquisadores devido ao seu potencial de melhorar o desempenho de algoritmos evolucionários em diversos contextos. Algoritmos meméticos em geral se beneficiam de operadores de busca local que são acrescidos aos

operadores usuais de algoritmos evolucionários. Contudo, no contexto de otimização com variáveis contínuas em PAC, o custo computacional de operadores de busca local torna seu uso proibitivo. Este custo pode ser reduzido com o emprego de técnicas de aproximação de funções, dessa forma, tornando os algoritmos meméticos ferramentas de uso prático em PAC.

Objetivos da tese

Os principais objetivos desta tese são investigar, implementar e testar o uso de aproximações locais em um arcabouço de algoritmos meméticos para a solução de problemas de PAC, em particular o projeto de dispositivos eletromagnéticos. Esta tese pretende contribuir para a investigação e desenvolvimento dos algoritmos meméticos, com foco especial em problemas cuja avaliação de funções envolve cálculos complexos e computacionalmente caros. Neste contexto, a complexidade adicional no algoritmo seria claramente justificável.

1. Organização de um arcabouço para otimização evolucionária aplicada a problemas de PAC. Este arcabouço precisa acomodar vários algoritmos evolucionários para otimização mono e multi-objetivo, incluindo, é claro, os algoritmos meméticos. Esta tese identificará uma estrutura comum com o objetivo de unificar os vários algoritmos em um mesmo arcabouço.

2. Definição da metodologia para incorporar as aproximações locais na fase de busca local de algoritmos meméticos. Esta metodologia será definida como um operador independente do arcabouço de algoritmos desenvolvido na parte anterior.

3. Investigação do efeito dos operadores de busca local sobre as propriedades de convergência de algoritmos evolucionários típicos. O estudo de convergência será feito com a teoria de cadeias de Markov. Além disso, a complexidade computacional da busca local será analizada. O propósito aqui é entender como a busca local afeta a convergência global e a complexidade dos algoritmos meméticos.

4. Implementação, teste e validação da metodologia proposta em problemas práticos de otimização de dispositivos eletromagnéticos.

Definição do problema de otimização

O problema geral de otimização é apresentado a seguir:

$$\begin{aligned} &\min \mathbf{f}(\mathbf{x}) \in \mathbb{R}^m \\ &\text{sujeito a: } \mathbf{x} \in \Omega \end{aligned} \tag{1}$$

em que Ω é o conjunto factível, matematicamente definido por:

$$\Omega = \{\mathbf{x} \in \mathcal{X} : g_i(\mathbf{x}) \leq 0,\ i = 1, \ldots, p; \quad h_j(\mathbf{x}) = 0,\ j = 1, \ldots, q\} \tag{2}$$

As funções $g_i(\,\cdot\,)$ e $h_j(\,\cdot\,)$ são respectivamente as restrições de desigualdade e igualdade. O conjunto $\mathcal{X} \subset \mathbb{R}^d$ representa o espaço de busca:

$$\mathcal{X} = \{\mathbf{x} : l_k \leq x_k \leq u_k,\ k = 1, \ldots, d\} \tag{3}$$

com os limites inferiores e superiores de cada variável.

As funções $f_i(\,\cdot\,) : \mathcal{X} \mapsto \mathbb{R}$, $i = 1, \ldots, m$, são as funções objetivo do problema. Cada função objetivo mapeia o espaço de busca em um trecho da reta real. O vetor de funções objetivo $\mathbf{f}(\,\cdot\,) : \mathcal{X} \mapsto \mathcal{Y}$ mapeia o espaço de busca $\mathcal{X} \subset \mathbb{R}^d$ no espaço de objetivos $\mathcal{Y} \subset \mathbb{R}^m$.

Algoritmo evolucionário unificado

Nesta tese unificamos os vários algoritmos evolucionários existentes em um arcabouço genérico. O Algoritmo 0.1 apresenta os ingredientes genéricos para representar qualquer algoritmo evolucionário mono e multi-objetivo. O Algoritmo 0.1 apresenta três populações distintas: uma população de memória, denominada população de arquivo, que armazena as melhores soluções encontradas até então; uma população de indivíduos "pais" ou progenitores; e uma população de indivíduos "filhos" ou descendência. O algoritmo unificado apresenta duas etapas de seleção ou dois operadores que contribuem para a pressão seletiva: a seleção para reprodução, que escolhe indivíduos da população de "pais" para gerar a descendência, e a seleção para substituição, que determina quais indivíduos sobreviverão para a próxima geração.

O algoritmo evolucionário unificado em Algoritmo 0.1 é bastante prático para o estágio de implementação. Os demais algoritmos do arcabouço todos seguem este modelo principal e operadores de reprodução e seleção específicos podem ser facilmente acrescentados. Estes operadores são fornecidos como entrada para fazer o algoritmo unificado se comportar como uma estratégia evolutiva, um algoritmo genético, um MOGA ou NSGA2.

Algorithm 0.1: Algoritmo evolucionário unificado.

Data: population size μ, offspring size λ, maximum archive size ξ, search space $\mathcal{X}$, objective and constraint functions $\mathbf{f}(\cdot)$, $\mathbf{g}(\cdot)$, $\mathbf{h}(\cdot)$.
Result: Estimate(s) of $\mathcal{X}^*$ in the archive population $A(n)$.

$P(n=0) = \{\mathbf{p}^{(1)}, \ldots, \mathbf{p}^{(\mu)}\} \leftarrow$ `Initialize population`$(\mu, \mathcal{X})$;
$A(n=0) = \varnothing \leftarrow$ `Initialize archive /* Stores the best solution set */`
$\Upsilon_P(n) \leftarrow$ `Penalty`$(P(n), \mathbf{f}, \mathbf{g}, \mathbf{h})$;
$\Phi_P(n) \leftarrow$ `Fitness`$(\Upsilon_P(n))$;
$A(n=0) \leftarrow$ `Update`$(A(n), P(n), \xi)$;
`// Reproduction comprises selection and variation`
$Q(n=0) = \{\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(\lambda)}\} \leftarrow$ `Reproduction`$(P(n), A(n), \Phi_P(n))$;
while $\neg$ *stop criteria* **do**
  $\Upsilon_Q(n) \leftarrow$ `Penalty`$(Q(n), \mathbf{f}, \mathbf{g}, \mathbf{h})$;
  $\Phi_Q(n) \leftarrow$ `Fitness`$(\Upsilon_Q(n))$;
  $P(n+1) \leftarrow$ `Substitution`$(P(n), Q(n))$;
  $Q(n+1) \leftarrow$ `Reproduction`$(P(n+1))$;
  $A(n+1) \leftarrow$ `Update`$(A(n), P(n) \cup Q(n), \xi)$;
  $n \leftarrow n+1$;
end

A Figura 1 a seguir ilustra a arquitetura do arcabouço implementado nesta tese. Vários algoritmos evolucionários típicos estão imediatamente disponíveis, assim como muitos outros algoritmos não canônicos, dependendo da combinação de operadores utilizados.

Busca local baseada em aproximações

Métodos evolucionários podem ser vistos como técnicas de amostragem adaptativa, que realizam uma amostragem "inteligente" do espaço de busca. Esta percepção de inteligência é determinada pelos operadores do algoritmo que dirigem essa amostragem para as regiões mais promissoras. Estes operadores são projetados com a premissa de

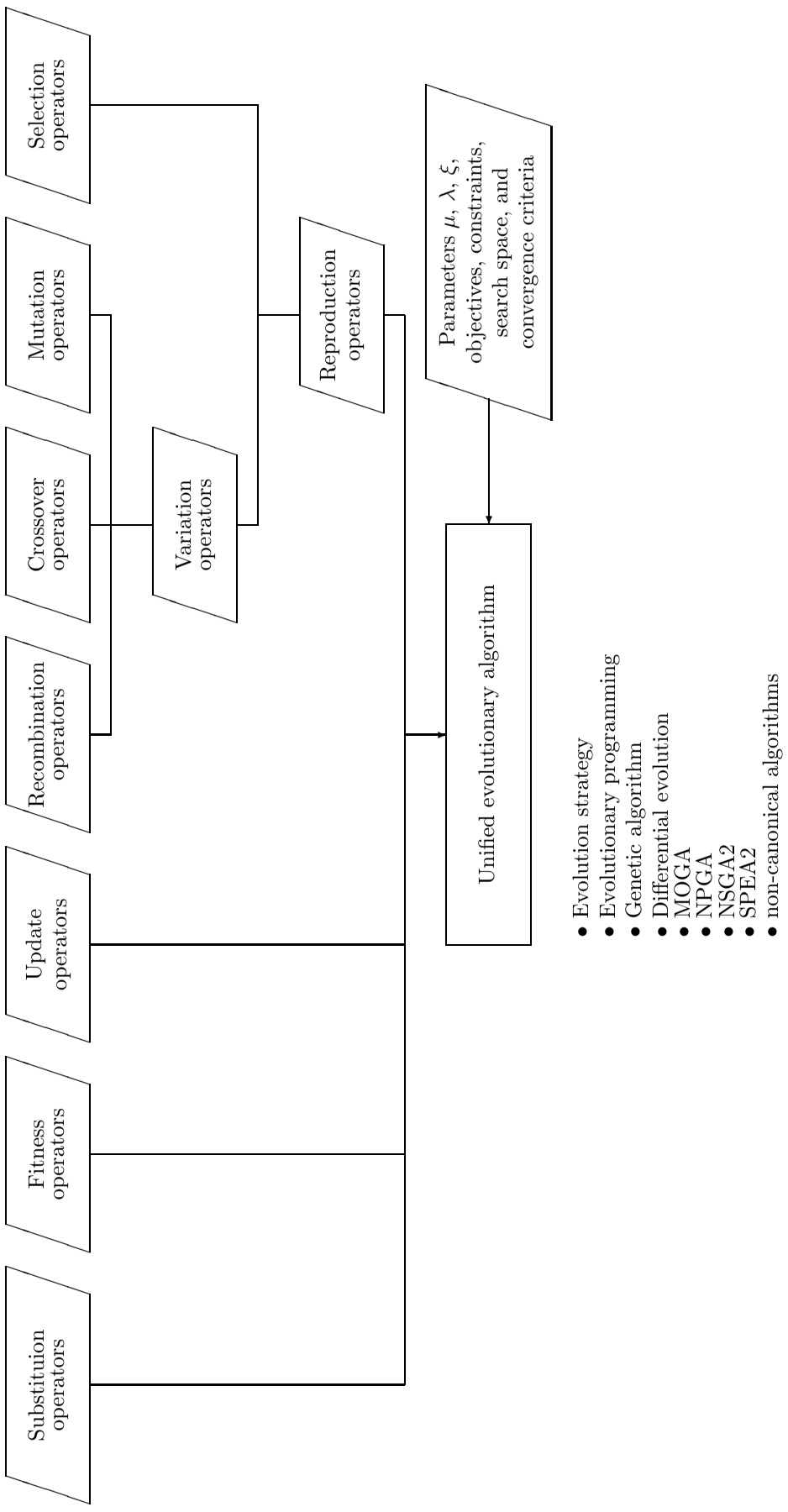

Figura 1: Fluxograma do arcabouço de algoritmos evolucionários baseado no modelo unificado.

que soluções ainda melhores podem ser encontradas nas vizinhanças das regiões mais promissoras.

A característica de amostragem dos métodos evolucionários pode ser usada para reduzir o custo computacional da fase de busca local em algoritmos meméticos, principalmente quando consideramos a classe de problemas em PAC. Os indivíduos da população podem utilizar parte da informação acumulada pelas consecutivas populações para construir modelos mais simples e menos custosos das funções envolvidas no problema de otimização. À medida que o processo de busca avança, podemos reconstruir o relacionamento entrada e saída das funções reais de forma progressivamente mais precisa, o que por sua vez auxilia a busca local.

Nesta seção descrevemos o operador de busca local baseado em aproximações multiquádricas.

Todas as avaliações invocadas pelo algoritmo evolucionário são armazenadas em um conjunto de dados global:

$$\mathcal{D}(n) = \left\{\mathbf{z}^{(i)}; f_1(\mathbf{z}^{(i)}), \ldots, f_m(\mathbf{z}^{(i)}); g_1(\mathbf{z}^{(i)}), \ldots, g_p(\mathbf{z}^{(i)}); h_1(\mathbf{z}^{(i)}), \ldots, h_q(\mathbf{z}^{(i)})\right\}_{i=1}^{N(n)} \quad (4)$$

em que $N(n)$ é o número total de amostras coletadas até a iteração n. Somente avaliações de funções não lineares são armazenadas e usadas na etapa de aproximação.

$\mathcal{D}(n)$ é o conjunto de dados global, mas somente parte dessa informação é usada na construção da aproximação local. Seja $\mathbf{x}^{(c)}$ a solução representada pelo indivíduo $\mathbf{p}^{(i)} \in P(n)$ selecionado para a busca local. A região de vizinhança é centrada em $\mathbf{x}^{(c)}$ e definida a seguir:

$$\mathcal{V}(\mathbf{x}^{(c)}, \epsilon) = \left\{\mathbf{z} : \|\mathbf{z} - \mathbf{x}^{(c)}\| \leq R(\epsilon)\right\} \quad (5)$$

em que $R(\epsilon)$ é uma região geométrica parametrizada por ϵ. Este parâmetro define o tamanho da vizinhança local.

Os pontos no conjunto de dados global que pertencem à vizinhança local formam o conjunto de dados local $\mathcal{L}$. Esse conjunto é usado para gerar as aproximações locais. Nesta tese utilizamos a técnica de interpolação multiquádrica como técnica de aproximação.

De posse das aproximações, define-se o problema de busca local como:

$$\begin{aligned} &\min \tilde{\mathbf{f}}(\mathbf{x}) \in \mathbb{R}^m \\ &\text{sujeito a: } \mathbf{x} \in \widetilde{\Omega} \cap \mathcal{V}(\mathbf{x}^{(c)}, \epsilon) \end{aligned} \tag{6}$$

em que $\mathcal{V}(\mathbf{x}^{(c)}, \epsilon)$ é a vizinhança local, $\tilde{\mathbf{f}}(\,\cdot\,)$ são as aproximações das funções objetivo, e $\widetilde{\Omega}$ é o conjunto factível determinado pelas aproximações das funções de restrição. O problem é portanto restrito à vizinhança local $\mathcal{V}(\mathbf{x}^{(c)}, \epsilon)$.

O caso mono-objetivo é facilmente resolvido pelo método de programação quadrática sequencial (SQP em inglês). A versão multi-objetivo necessita de uma etapa de transformação adicional, para transformar o problema multi-objetivo em uma versão mono-objetivo que possa ser resolvida pelo método SQP. Essa transformação é feita através da formulação de realização por metas.

O operador de busca local se baseia numa região de vizinhança local parametrizada por ϵ. Este parâmetro pode ser fixo ou ajustado de forma automática. Neste último caso, uma metodologia baseada nos métodos de regiões de confiança pode ser empregada para ajustar o parâmetro ϵ. O Algoritmo 0.2 a seguir apresenta o operador de busca local com regiões de confiança.

Hibridização com algoritmos evolucionários

Esta seção ilustra a hibridização da busca local baseada em aproximações com o arcabouço apresentado anteriormente.

Para isso, definimos:

- o número de gerações consecutivas em que a busca local não é aplicada $n_L \geq 0$.
- o número de indivíduos selecionados para a busca local $0 \leq \sigma \leq \mu$.

O Algoritmo 0.3 a seguir descreve como o operador de busca local é combinado com os algoritmos evolucionários. Observe que os Algoritmos 0.2 e 0.3 são válidos para o contexto mono e multi-objetivo.

Algorithm 0.2: Operador de busca local com regiões de confiança.

$\mathcal{L} \leftarrow$ `Build local data`$(\mathcal{V}(\mathbf{x}^{(c)}, \epsilon))$;
$\tilde{\mathbf{f}}, \tilde{\mathbf{g}}, \tilde{\mathbf{h}} \leftarrow$ `Generate local approximations`$(\mathcal{L})$;
Solve auxiliary problem starting from $\mathbf{x}^{(c)}$ and obtaining trial solution $\mathbf{x}^{(t)}$;
for $i = 1, \ldots, n_{TR}$ **do**
    Evaluate the real functions and the local approximations at $\mathbf{x}^{(t)}$;
    Compute the merit figure η;
    if $\eta < 0$ **then** `/* x(t) is rejected and ϵ decreases */`
        $\epsilon \leftarrow \omega_{down}\epsilon$;
    else if $0 < \eta < \mu_{low}$ **then** `/* x(t) is accepted and ϵ decreases */`
        $\epsilon \leftarrow \omega_{down}\epsilon$;
        $\mathbf{x}^{(c)} \leftarrow \mathbf{x}^{(t)}$;
    else if $\mu_{low} < \eta < \mu_{high}$ **then** `/* x(t) is accepted and ϵ remains the same */`
        $\mathbf{x}^{(c)} \leftarrow \mathbf{x}^{(t)}$;
    else `/* x(t) is accepted and ϵ may increase */`
        if $\mathbf{x}^{(t)} \in \partial\mathcal{V}(\mathbf{x}^{(c)}, \epsilon)$ **then** $\epsilon \leftarrow \omega_{up}\epsilon$;
        $\mathbf{x}^{(c)} \leftarrow \mathbf{x}^{(t)}$;
    $\mathcal{L} \leftarrow$ `Build local data`$(\mathcal{V}(\mathbf{x}^{(c)}, \epsilon))$;
    $\tilde{\mathbf{f}}, \tilde{\mathbf{g}}, \tilde{\mathbf{h}} \leftarrow$ `Generate local approximations`$(\mathcal{L})$;
    Solve auxiliary problem starting from $\mathbf{x}^{(c)}$ and obtaining trial solution $\mathbf{x}^{(t)}$;
end
return $\mathbf{x}^{(t)}$;

Algorithm 0.3: Algoritmo evolucionário unificado com busca local.

Data: population size μ, offspring size λ, maximum archive size ξ, search space $\mathcal{X}$, objective and constraint functions $\mathbf{f}(\cdot)$, $\mathbf{g}(\cdot)$, $\mathbf{h}(\cdot)$.
Result: Estimate(s) of $\mathcal{X}^*$ in the archive population $A(n)$.

$P(n=0) = \left\{\mathbf{p}^{(1)}, \ldots, \mathbf{p}^{(\mu)}\right\} \leftarrow$ `Initialize population`$(\mu, \mathcal{X})$;
$A(n=0) = \varnothing \leftarrow$ `Initialize archive /* Stores the best solution set */`
$\Upsilon_P(n) \leftarrow$ `Penalty`$(P(n), \mathbf{f}, \mathbf{g}, \mathbf{h})$;
$\Phi_P(n) \leftarrow$ `Fitness`$(\Upsilon_P(n))$;
$A(n=0) \leftarrow$ `Update`$(A(n), P(n), \xi)$;
`// Reproduction comprises selection and variation`
$Q(n=0) = \left\{\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(\lambda)}\right\} \leftarrow$ `Reproduction`$(P(n), A(n), \Phi_P(n))$;
while $\neg$ *stop criteria* **do**
  $\Upsilon_Q(n) \leftarrow$ `Penalty`$(Q(n), \mathbf{f}, \mathbf{g}, \mathbf{h})$;
  $\Phi_Q(n) \leftarrow$ `Fitness`$(\Upsilon_Q(n))$;
  if $\mod(n, n_L) = 0$ **then**
    for *each of the* σ *best individuals* **do**
      Local search operator `/* see Algorithm 3.2 */`
    end
  end
  $P(n+1) \leftarrow$ `Substitution`$(P(n), Q(n))$;
  $Q(n+1) \leftarrow$ `Reproduction`$(P(n+1))$;
  $A(n+1) \leftarrow$ `Update`$(A(n), P(n) \cup Q(n), \xi)$;
  $n \leftarrow n+1$;
end

Análise de convergência

Seja $\mathcal{X}^*$ o conjunto ótimo para o problema de otimização. Este pode representar:

1. todas as soluções globais, se existem mais de uma;
2. todas as soluções locais e globais, se o algoritmo é projetado para buscá-las;
3. ou todas as soluções Pareto-ótimas globais, em um contexto multi-objetivo.

Dizemos que o algoritmo é globalmente convergente se:

$$\lim_{n\to\infty} \mathscr{P}\{\mathsf{X}_n^A \subseteq \mathcal{X}^*\} = 1 \tag{7}$$

em outras palavras, se a probabilidade da população de arquivo $A(n)$ ter convergido para um subconjunto das soluções ótimas se aproxima de um quando o número de iterações vai para infinito. Dizemos que a população $P(n)$ *localiza* a solução ótima, enquanto a população $A(n)$ *converge* para a solução ótima.

Para provar a convergência de algoritmos evolucionários genéricos sob esse critério, pode-se utilizar o seguinte Lema da teoria de cadeias de Markov:

Lemma 0.1. *Uma cadeia de Markov homogênea com espaço de estados finito e matriz de transição irredutível visita todos os estados com probabilidade um, independente da distribuição inicial.*

Baseando-se nesse Lema, tudo que precisamos fazer é provar que a matriz de transição associada com o algoritmo evolucionário é irredutível. Dessa forma, garantimos que todos os estados serão visitados, inclusive aqueles relacionados a $\mathcal{X}^*$. Garantimos que o algoritmo localiza as soluções ótimas, e que portanto converge sob o critério acima se essas soluções são armazenadas em $A(n)$.

Vamos considerar um algoritmo genético padrão, usando seleção, cruzamento e mutação. Neste caso, a matriz de transição total da iteração é:

$$\mathbf{G} = \mathbf{SCM} \tag{8}$$

em que $\mathbf{S}$, $\mathbf{C}$, e $\mathbf{M}$ são respectivamente as matrizes de transição dos passos de seleção, cruzamento e mutação.

Analisando as propriedades estocásticas das matrizes de transição que os operadores determinam, é possível concluir se a matriz de transição total é irredutível ou não, e portanto, se o algoritmo é globalmente convergente ou não.

Analisando o operador de busca local

Nesta seção, analisamos o que acontece quando um operador de busca local é explicitamente usado no ciclo evolucionário. A busca local é usada antes da seleção, portanto a matriz de transição da iteração fica:

$$\mathbf{H} = \mathbf{LG} = \mathbf{L}(\mathbf{SCM}) \tag{9}$$

em que $\mathbf{L}$ é a matriz de transição associada à fase de busca local, $\mathbf{G}$ é a matriz de transição do algoritmo de busca global, e $\mathbf{H}$ é a matriz de transição do algoritmo híbrido, resultante da interação entre busca local-global.

Na abordagem Baldwiniana, que não muda a representação das soluções no espaço de busca, a matriz de transição $\mathbf{L}$ é uma matriz identidade, porque as populações antes e após a busca local são as mesmas. A abordagem Baldwiniana modifica as probabilidades de seleção, mas a matriz $\mathbf{S}$ preserva suas propriedades anteriores. Dessa forma, a busca local Baldwiniana não afeta a convergência global.

Na abordagem Lamarckiana, a população é modificada pela busca local. Se o operador é estocástico, então $\mathbf{L}$ é diagonal positiva. O algoritmo é portanto globalmente convergente como mostra o teorema a seguir.

Theorem 0.2. *Se* $\mathbf{SCM}$ *é positiva e a busca local é estocástica, então* $\mathbf{H}$ *é positiva e portanto irredutível. Logo a população de arquivo convergirá para o conjunto solução e o algoritmo híbrido é globalmente convergente.*

Se a busca local é determinística, então $\mathbf{L}$ é pelo menos linha-permissível. Nesse caso temos o seguinte resultado:

Theorem 0.3. *Se* $\mathbf{SCM}$ *é positiva e a busca local é determinística, então* $\mathbf{L}$ *é pelo menos linha-permissível, então* $\mathbf{H}$ *é também positiva e portanto irredutível. Logo a população de arquivo convergirá para o conjunto solução e o algoritmo híbrido é globalmente convergente.*

Contudo, quando $\mathbf{G} = \mathbf{SCM}$ é irredutível devido a um operador de mutação que produz uma matriz não positiva mas irredutível, e $\mathbf{L}$ é linha permissível, não podemos afirmar que $\mathbf{H}$ é irredutível. Portanto, não podemos provar a convergência global no caso geral.

Entretanto, é possível garantir a convergência global se o número de indivíduos selecionados para a busca local é menor do que o tamanho da população. Este resultado é expresso no teorema a seguir.

Theorem 0.4. *Se $\sigma < \mu$, $\mathbf{SCM}$ é irredutível e $\mathbf{L}$ é linha permissível, então $\mathbf{H}$ é também irredutível (mas não positiva em geral). Logo a população de arquivo convergirá para o conjunto solução e o algoritmo híbrido é globalmente convergente.*

Finalmente, vamos considerar o caso em que a busca local não é aplicada em toda geração mas apenas em um intervalo constante de gerações, digamos a cada n_L gerações. A matriz de transição torna-se:

$$\mathbf{H} = \mathbf{L}\underbrace{\mathbf{SCM} \times \mathbf{SCM} \times \cdots \times \mathbf{SCM}}_{n_L \text{ times}} \tag{10}$$

então:

$$\mathbf{H}^n = (\mathbf{L}\mathbf{G}^{n_L})^n \tag{11}$$

Para essa situação, temos o seguinte resultado:

Theorem 0.5. *Se $\sigma < \mu$, $\mathbf{G} = \mathbf{SCM}$ é irredutível, $\mathbf{L}$ é linha-permissível, e a busca local é aplicada a cada n_L gerações, então $\mathbf{H}$ é também irredutível (mas não positiva em geral). Logo a população de arquivo convergirá para o conjunto solução e o algoritmo híbrido é globalmente convergente.*

Portanto, dadas as considerações acima, o algoritmo híbrido usando busca local explícita preserva a convergência global desde que esta característica esteja presente no algoritmo evolucionário em questão.

Custo computacional

Nesta seção, analisamos o custo computacional do operador de busca local baseado em aproximações. Essa análise é importante porque, embora o algoritmo híbrido tenha o potencial de convergir em menos gerações em média, suas gerações são mais caras. Na prática, desejamos que o tempo total de otimização (em média) usando o algoritmo híbrido seja menor do que o tempo total usando o algoritmo convencional, e não apenas o número de gerações.

$\langle N \rangle$ representa o número médio de gerações para convergir. $\langle T \rangle$ representa o tempo médio total de otimização. Podemos estimar a complexidade computacional de um algoritmo evolucionário como

$$O(\mu, \lambda, \xi, d)\langle N \rangle = \left[O_r(\mu, \lambda, \xi, d) + O_e(\mu, \lambda, \xi, d)\right] \langle N \rangle \tag{12}$$

onde $O_r(\,\cdot\,)$ é o número de operações no passo de reprodução e $O_e(\,\cdot\,)$ é o número de operações no passo de avaliação. Estas quantidades estão relacionadas com os parâmetros do algoritmo e o número de variáveis d.

Analisando o operador de busca local baseado em interpolação multiquádrica, chega-se à seguinte expressão geral para seu custo computacional:

$$\begin{aligned} O_{LS} = (1 + n_{TR}) \big\{ & O((\mu + \lambda N_{\max}) N_L d) + O((m + p + q)(d + 1) N_L^2) + \\ & + O(\left[(m + p + q) + \text{step}(m - 1) m (1 + p + q)\right] d (d + 1) N_L) \big\} + n_{TR} O_e(d) \end{aligned}$$

em que $n_{TR} \geq 0$ é o número de atualizações da região de confiança, e $O_e(\,\cdot\,)$ é a complexidade da etapa de avaliação. A função degrau nessa expressão é definida como:

$$\text{step}(x) = \begin{cases} 0, & \text{if } x \leq 0 \\ 1, & \text{if } x > 0 \end{cases} \tag{13}$$

A expressão acima para O_{LS} é igualmente válida para os casos mono e multi-objetivo. Essa expressão indica que a complexidade do operador de busca local proposto é:

- Linear com o número de indivíduos;
- Linear com o número de objetivos e restrições;
- Linear com o número máximo de gerações;

- Linear com o número de atualizações da região de confiança;
- Quadrático com o número de variáveis;
- Quadrático com o número máximo de pontos no conjunto de dados local;

Como N_L é maior do que o número de variáveis d e o número de objetivos e restrições, nota-se que a complexidade da busca local é dominada pelo termo $O((m+p+q)(d+1)N_L^2)$. Porém, o termo $O((\mu + \lambda N_{\max})N_L d)$ também é importante, pois $N_{\max}$ e λ são números relativamente grandes.

A complexidade computacional de operadores em algoritmos evolucionários típicos é polinomial com relação a μ, λ, ξ, e d. A conclusão importante aqui é que o algoritmo memético preserva a complexidade polinomial do método original.

Resultados

Problemas analíticos

Neste primeiro experimento aplicamos o algoritmo genético típico nos seguintes problemas analíticos:

$$f_1(\mathbf{x}) = \sum_{i=1}^{d-1} 100 \left(x_i^2 - x_{i+1}\right)^2 + (1 - x_i)^2 \tag{14}$$

cujo espaço de busca é $\mathcal{X}_1 = \{\mathbf{x} : -2 \leq x_k \leq 2,\ k = 1, \ldots, d\}$.

$$f_2(\mathbf{x}) = 2.6164 + \frac{1}{d} \sum_{i=1}^{d} 0.01 \left[(x_i + 0.5)^4 - 30x_i^2 - 20x_i\right] \tag{15}$$

cujo espaço de busca é $\mathcal{X}_2 = \{\mathbf{x} : -6 \leq x_k \leq 6,\ k = 1, \ldots, d\}$.

$$f_3(\mathbf{x}) = 10d + \sum_{i=1}^{d} \left[x_i^2 - 10\cos(2\pi x_i)\right] \tag{16}$$

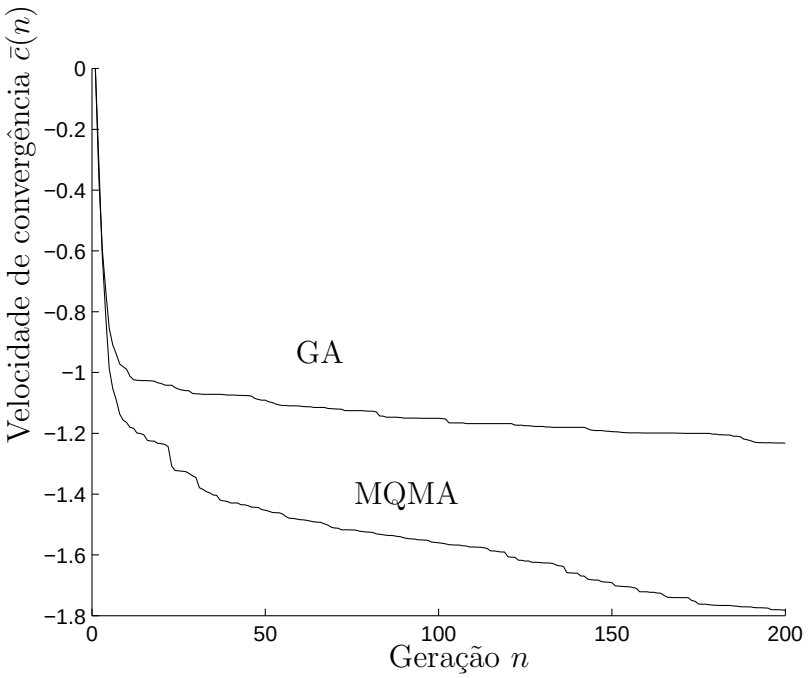

Figura 2: Velocidade média de convergência $\bar{c}(n)$ para a otimização de $f_1(\mathbf{x})$.

cujo espaço de busca é $\mathcal{X}_3 = \{\mathbf{x} : -5 \leq x_k \leq 5,\ k = 1, \ldots, d\}$.

A velocidade de convergência do algoritmo genético GA nesses problemas é comparada com a velocidade obtida por sua versão híbrida, denominada MQMA.

Os algoritmos GA e MQMA compartilham a seguinte configuração:

- $\mu = \lambda = 50$ e $\xi = 1$;
- ES(μ, μ) para a substituição;
- seleção por roleta;
- cruzamento real polarizado com $\rho_c = 0.8$, $\rho_{pol} = 0.8$, e extrapolação de 0.1;
- mutação Gaussiana com $\rho_m = 0.1$;
- executados por 200 gerações;

O MQMA utilizou busca local com vizinhança retangular com $\epsilon = 0.1$, $N_L = 200$, $\sigma = 1$ e $n_L = 1$.

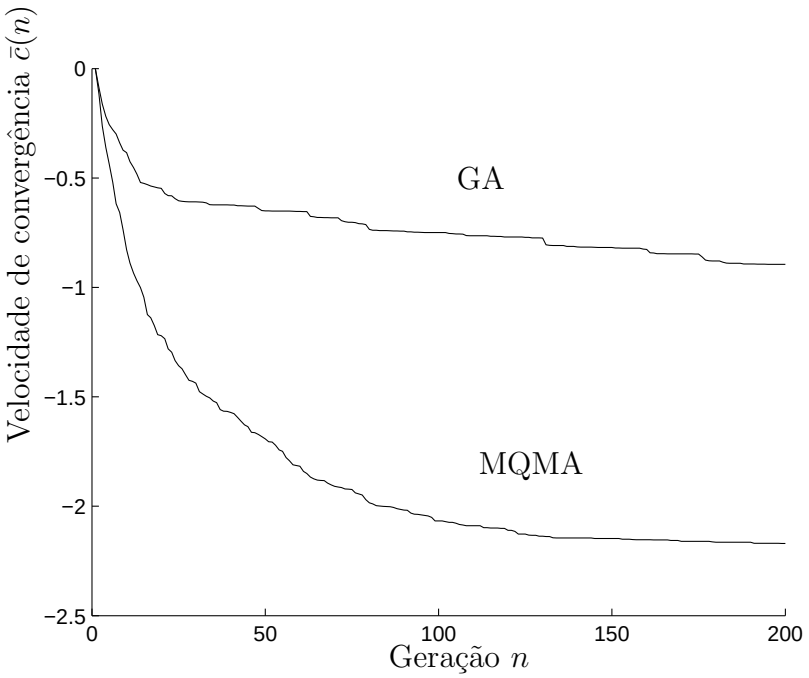

Figura 3: Velocidade média de convergência $\bar{c}(n)$ para a otimização de $f_2(\mathbf{x})$.

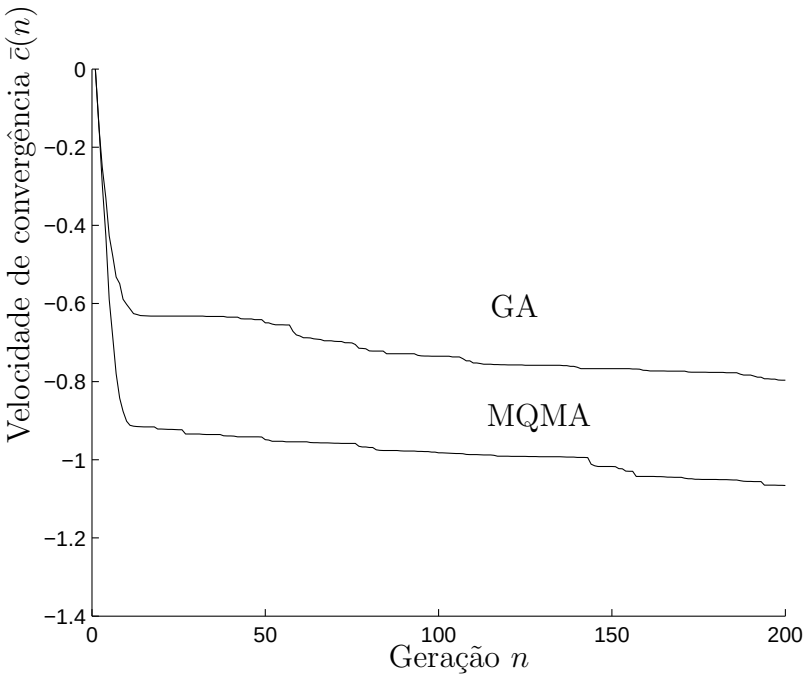

Figura 4: Velocidade média de convergência $\bar{c}(n)$ para a otimização de $f_3(\mathbf{x})$.

A curva de velocidade média de convergência foi levantada experimentalmente para

cada algoritmo. Podemos ver nas Figuras 2 a 4 que o algoritmo memético convergiu mais rapidamente em termos de número de gerações.

Projeto do magnetizador e custo computacional

O problema do magnetizador consiste na otimização da forma de seu pólo.

A geometria do dispositivo é mostrada na Figura 5. O objetivo é minimizar a diferença entre a densidade de fluxo magnético calculada ao longo da corda AB e um valor desejado:

$$D = \sum_{n=1}^{59} |B_{i,calc} - B_{i,ref}| \tag{17}$$

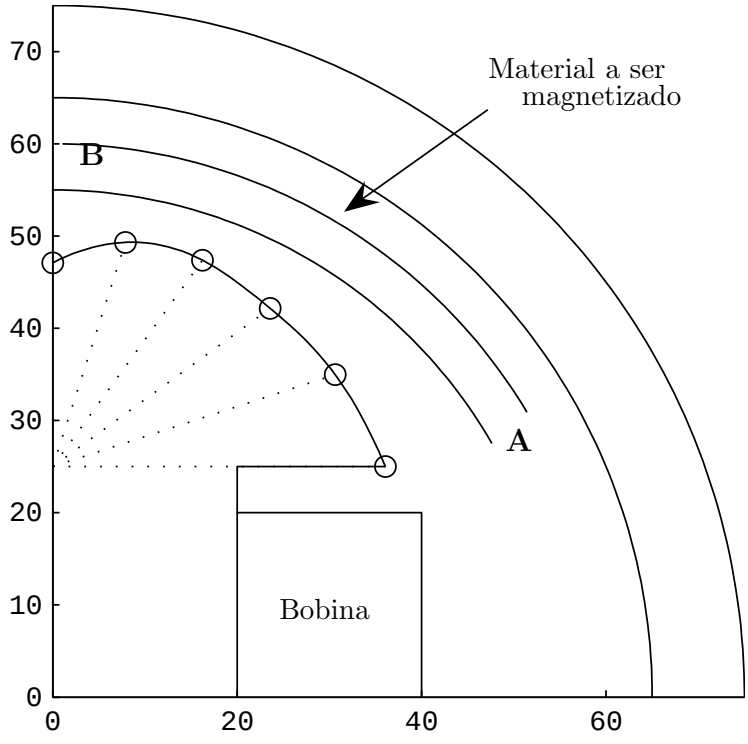

Figura 5: Geometria do problema do magnetizador em mm. O raio do estator é de $75mm$.

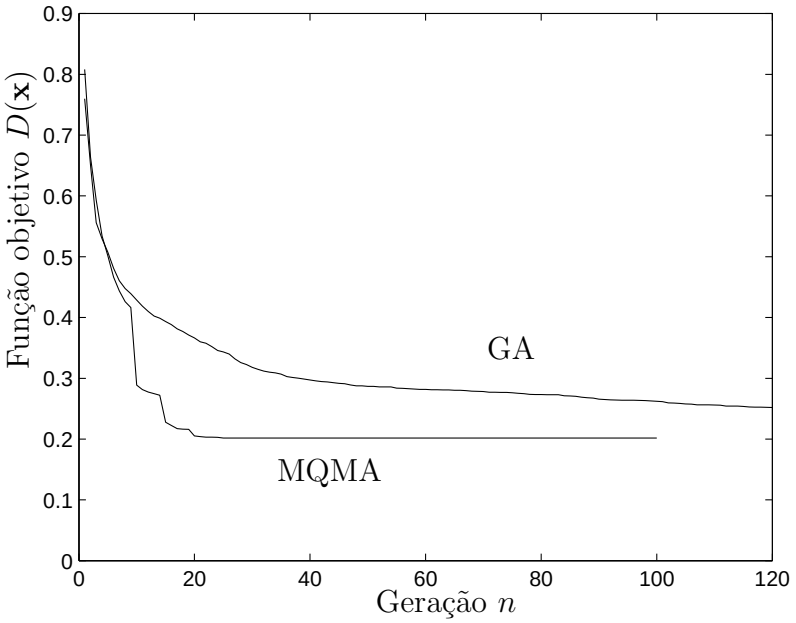

Figura 6: Curva média de convergência para o problema do magnetizador.

em que $B_{i,calc}$ e $B_{i,ref}$ são respectivamente a magnitude da densidade de fluxo calculada e o valor de referência no i-ésimo ponto ao longo da corda AB.

A metodologia adotada nesse problema é similar àquela utilizada nos problemas analíticos. O GA e o MQMA foram executados 30 vezes nesse problema e curvas médias de convergência foram levantadas. A Figura 6 mostra as curvas de convergência para cada algoritmo. O algoritmo híbrido convergiu em menos iterações, e também em menos tempo, mostrando que a metodologia de busca local com aproximações foi vantajosa nesse problema.

Nos experimentos anteriores com os problemas analíticos e com o problema do magnetizador, monitoramos os valores médios de: tempo de avaliação de um indivíduo (t_e), busca local de um indivíduo (t_{LS}), seleção (t_s), e variação (t_v). A Tabela 1 mostra os tempos médios para cada passo no problema do magnetizador e nos problemas f_1, f_2 e f_3.

Nos problemas analíticos, a busca local foi o passo mais computacionalmente caro. A carga extra da busca local baseada na interpolação multiquádrica é muito elevada, porque o custo de avaliação dessas funções é muito pequeno. Portanto, nos problemas

Passo	t_e	t_s	t_v	t_{LS}
	Função analítica f_1			
valores médios $[s]$	2.7×10^{-6}	1.0×10^{-3}	0.45×10^{-3}	2.24
normalizado	1	377	167	830,890
	Função analítica f_2			
valores médios $[s]$	2.2×10^{-6}	0.8×10^{-3}	0.37×10^{-3}	2.20
normalizado	1	380	166	999,160
	Função analítica f_3			
valores médios $[s]$	1.9×10^{-6}	0.7×10^{-3}	0.26×10^{-3}	1.89
normalizado	1	393	139	994,410
	problema do magnetizador			
valores médios $[s]$	1.23	0.65×10^{-3}	0.72×10^{-3}	2.35
normalizado	1	0.53×10^{-3}	0.59×10^{-3}	1.91

Tabela 1: Tempos médios

analíticos, embora o algoritmo memético convergiu em menos gerações, ele convergiu em mais tempo. Por outro lado, no problema do magnetizador, a situação inverte. O tempo para avaliar a população domina o processo de otimização. A carga extra da busca local torna-se menos importante nesse caso.

Sistema supercondutor de armazenamento de energia magnética

O problema 22 do TEAM Workshop é um conhecido problema de benchmark em otimização aplicada a dispositivos eletromagnéticos. Este problema em sua versão mono e multi-objetivo é usado como problema prático de teste para os algoritmos evolucionários e meméticos do arcabouço implementado nesta tese.

Primeiramente, aplicamos 12 algoritmos evolucionários e suas versões meméticas no problema 22 em sua versão mono-objetivo. Esses 12 algoritmos compreendem 6 GAs, 2 DEs, 2 ES e 2 EPs. Cada um desses algoritmos é hibridizado de duas formas diferentes:

- MA1 usa vizinhança retangular com $\epsilon = 0.1$, $N_L = 400$, $\sigma = 1$, $n_L = 4$, e $n_{TR} = 0$.
- MA2 usa vizinhança retangular com $\epsilon = 0.1$, $N_L = 400$, $\sigma = 10$, $n_L = 4$, e $n_{TR} = 1$.

MA2 tem uma busca local mais intensa que MA1, com a busca local sendo aplicada a 10 indivíduos da população e uma atualização da região de confiança.

Cada algoritmo foi executado 10 vezes e curvas de velocidade média de convergência foram levantadas.

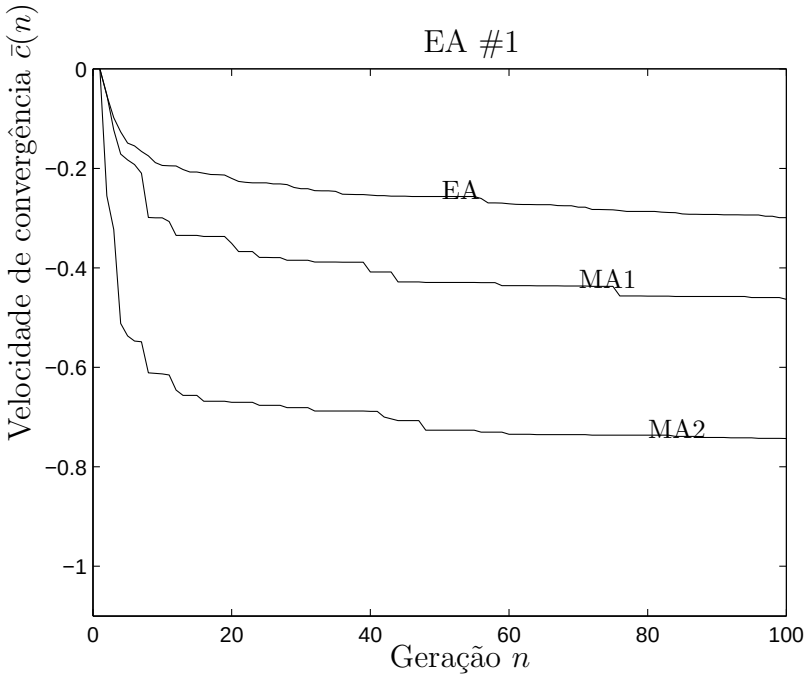

Figura 7: Velocidade de convergência para um dos GAs e suas versões meméticas.

Os 12 algoritmos evolucionários tiveram suas velocidades de convergência aumentadas quando hibridizados com a busca local. Os algoritmos meméticos convergiram mais rapidamente em todos os casos. As Figuras 7 e 8 ilustram os resultados para dois desses 12 algoritmos.

A versão multi-objetivo do problema 22 foi resolvida com o MOGA, NSGA2, SPEA2 e suas versões híbridas. neste experimento, usamos três diferentes estratégias para comparar a qualidade das soluções obtidas e a velocidade de convergência: (i) a função de aproveitamento é usada em um teste estatístico de significância; (ii) a métrica NDSCR é usada para comparar a qualidade das soluções alcançadas pelos algoritmos convencionais e híbridos; (iii) e a métrica de hipervolume é usada como base para levantar curvas de velocidade média de convergência.

A busca local multi-objetivo nas versões híbridas do MOGA, NSGA2 e SPEA2 utiliza

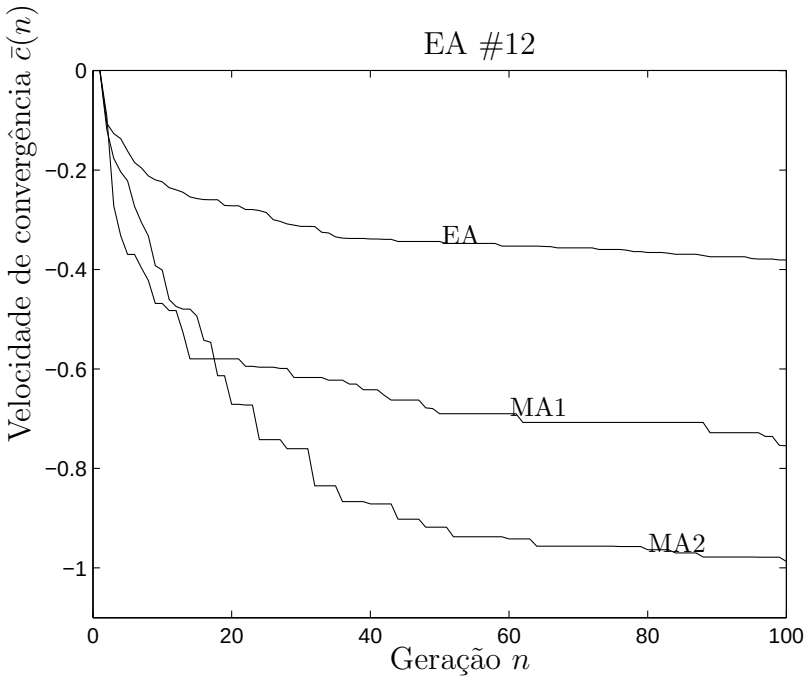

Figura 8: Velocidade de convergência para um dos EPs e suas versões meméticas.

$\sigma = 10$, $n_L = 4$, e $n_{TR} = 0$.

Os testes estatísticos com as funções de aproveitamento levam à conclusão de que a busca local produziu diferenças estatisticamente significativas entre os resultados gerados pelos algoritmos convencionais e híbridos.

A Figura 9 mostra os resultados para a métrica NDSCR. Os conjuntos não dominados gerados pelos algoritmos meméticos contribuíram mais para o conjunto de soluções não dominadas de ambos os algoritmos.

Finalmente, as Figuras 10, 11 e 12 mostram a velocidade de convergência da métrica de hipervolume para cada algoritmo. Pode-se perceber que as versões híbridas de MOGA, NSGA2 e SPEA2 convergiram mais rapidamente do que suas versões convencionais.

Esses resultados mostram que a metodologia de busca local proposta nesta tese melhora significativamente o desempenho de algoritmos evolucionários em geral.

No caso de problemas de PAC, a complexidade adicional de operadores de busca local baseada em aproximações é facilmente justificada nos algoritmos evolucionários híbridos pela pequena carga extra que esses operadores acarretam e pelo significativo

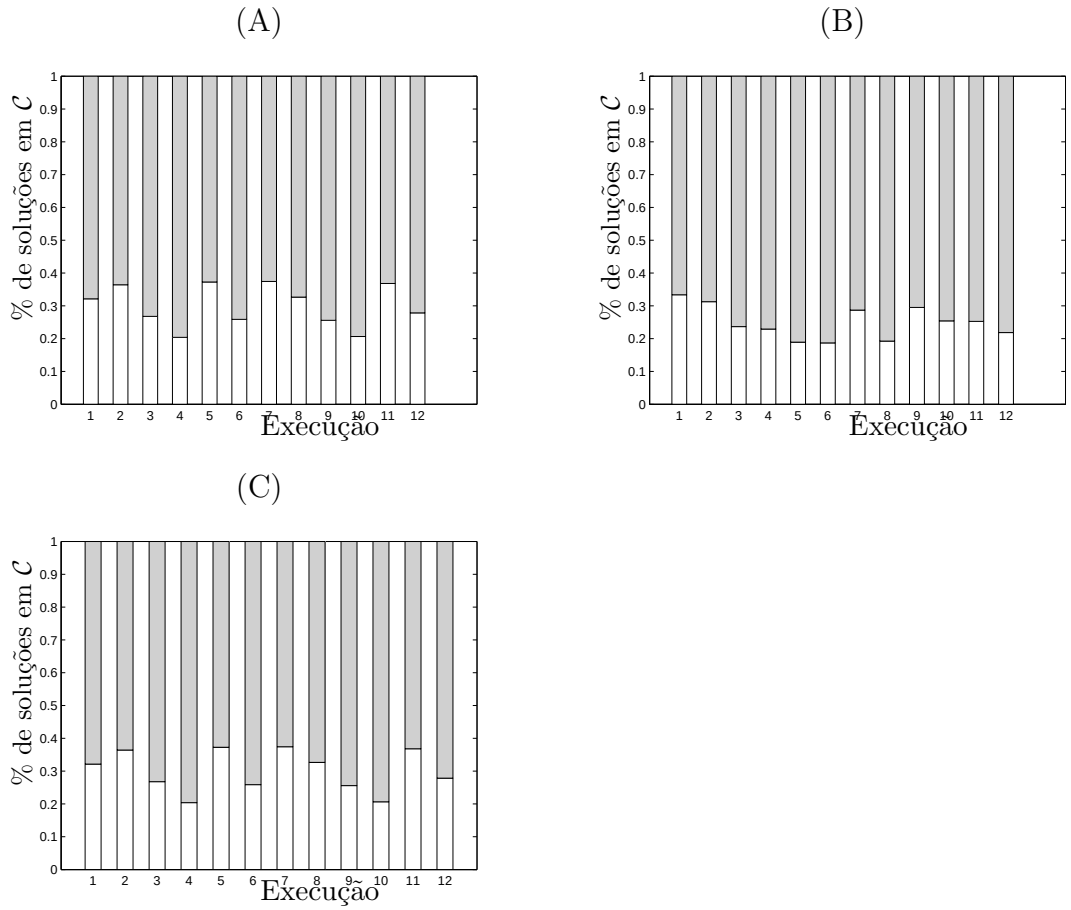

Figura 9: Métrica NDSCR para (A) MOGA e MQ-MOGA, (B) NSGA2 e MQ-NSGA2 (C) SPEA2 e MQ-SPEA2. As barras cinzas mostram a proporção de soluções provenientes do algoritmo memético.

ganho obtido em desempenho.

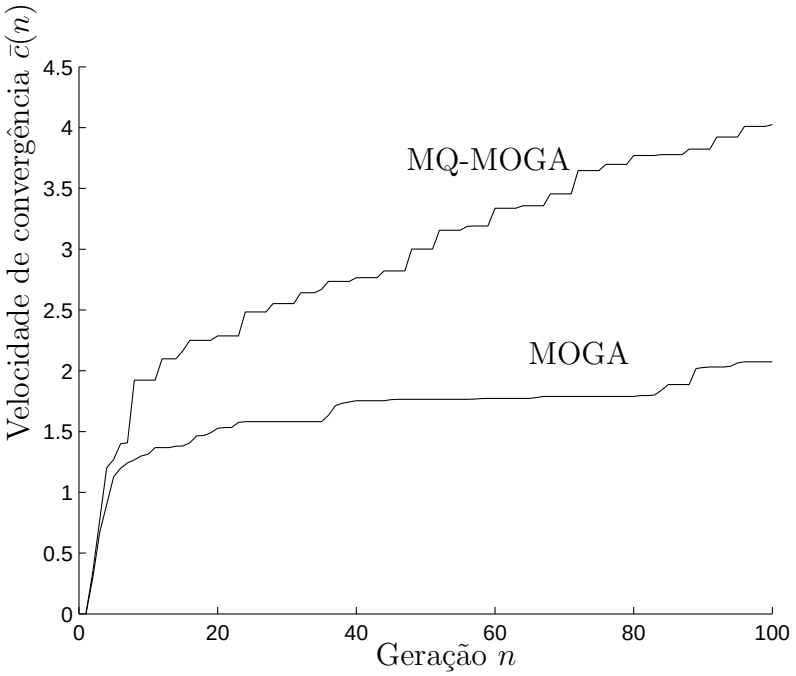

Figura 10: Velocidade de convergência para o MOGA.

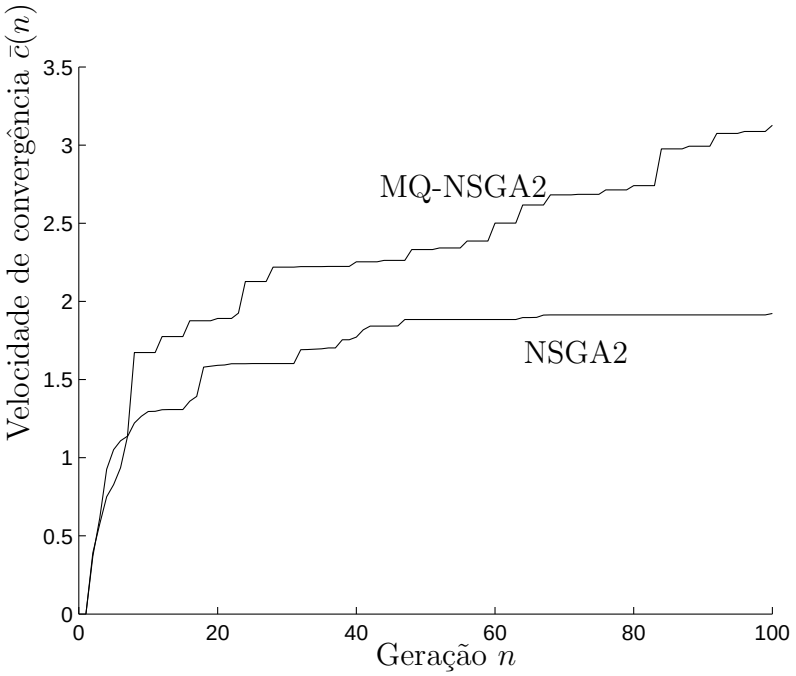

Figura 11: Velocidade de convergência para o NSGA2.

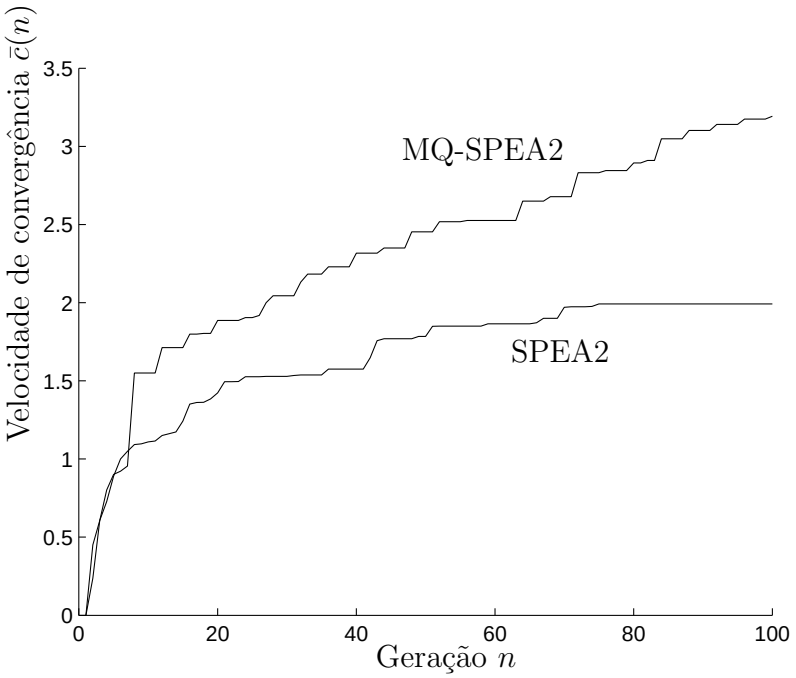

Figura 12: Velocidade de convergência para o SPEA2.

Chapter 1

Memetic Algorithms and Computer Aided Design

"The human race is particularly driven in his advancements
by the law that governs the evolution."
— Carlos Bernardo González Pecotche, 1901–1963

"When we die, there are two things we can leave behind us: genes and memes."
— Richard Dawkins, 1941–

1.1 Introduction

At some moment in our past, hundreds of thousands of years ago, a new evolution had its beginning. Since the appearance of life on Earth, natural evolution reigned alone and made possible the stage in which this new evolution develops, with new actors or units of evolution. Nature has created brains, and they were given learning ability. Thereby, cultural evolution developed in such a way that it may even interfere with the natural evolution. Some scientists even speculate that the moment of self-design in human history is not so far away (Hawking 2001).

Cultural evolution is a reality intensely experienced by humanity since the beginning

of what we term as civilization, in other words, since mankind abandoned the nomadic communal way of life and became organized in complex societies, mere ten to twelve thousand years ago. The parallel between the Darwin's evolutionary theory and the process of cultural development in mankind was proposed in 1976 by Richard Dawkins, in the last chapter of his book "The selfish gene" (Dawkins 1976). In a philosophical exercise, with the aim of illustrating that other kinds of replicators are possible, Dawkins proposes a new replicating unit in this new evolutionary scenario, similar to the gene as the replicating unit in biological evolution. He named it *meme*, a unit of cultural evolution (Dawkins 1976, chap. 11). The environment of memes is the brains, the minds of people, which form what he terms the ideosphere, in contrast with the biosphere. According to the definition given by Dawkins, a meme can be interpreted as an information pattern that propagates to other people's minds. In addition to genes, memes are also subject to mechanisms like selection, combination, and variation.

Memes related to arts, for instance, arose at a given moment in the past in many isolated groups. As generations passed away, such concepts have modified, adapted, and combined with other concepts, hence evolving into the diverse artistical concepts and trends that we see today. They are definitely more elaborated and complex than the original concepts that arose in the past. Other possible interpretations of memes are ideas, theories, music, trends, traditions, etc. All these concepts are better adapted to certain minds, that is, a given meme survives better in a specific mind than in others. The fittest memes, i.e., better adapted to a given cultural environment, are more capable of disseminating themselves more rapidly.

As the Dawkins' epigraph that begins this chapter suggests, the inheritance that we receive from our ancestors has not only a genetic component, but also something else. The other component in our inheritance corresponds to the memes, the replicating units of cultural evolution. Each culture is composed of the ensemble of all memes in a society. All cultures, in turn, form the human culture, the human knowledge.

This topic, though interesting, deserves to be explored by sociologists, psychologists, etc[1]. The main question here is: how does this relate to optimization algorithms? The answer are the memetic algorithms, which are evolutionary algorithms that employ local search operators.

Memetic algorithms are also known as genetic local search algorithms, or hybrid evolutionary algorithms. Although they have developed independently, nowadays there

[1]To know more about the Meme Theory, see the books (Blackmore 2000, Shennan 2003).

is a preference for terming them *memetic algorithms* (Moscato 1999, Krasnogor 2002, Hart, Krasnogor and Smith 2004). The common characteristic here is that an explicit local search is adopted within the standard evolutionary cycle of evolutionary algorithms.

The term *memetic algorithms* was coined by Pablo Moscato in 1989 (Moscato 1989). He was working with the utilization of local refinement operators in genetic algorithms for solving combinatorial optimization problems, specifically, travelling salesman problems (TSP). He used the expression memetic algorithms because the individuals in his algorithms were subject to local search operators, in addition to the traditional selection, combination, and mutation operators. An individual cannot modify his genes, but he can adapt his memes before transmitting them. That is exactly what local search operators do: they allow individuals to modify their inheritance before transmitting them to the next generations.

For similar reasons, memetic algorithms are given names as Lamarckian algorithms and Baldwinian algorithms. In Lamarck theory, for example, the experience acquired by the individuals is believed to pass to their descendants. With the use of local optimizers, the improvement achieved by a solution in a given generation could pass to the offspring.

In fact, the association of local and global search is the key to an efficient and successful optimization not only in combinatorial problems, as was originally proposed, but also in other difficult optimization problems in general. In this way, many studies have been produced in this field since Moscato's pioneering work. This thesis presents a study of memetic algorithms for nonlinear optimization in the context of computer aided design.

Generally, the basic cycle of a memetic algorithm consists simply of the standard cycle of an evolutionary algorithm combined with local search operators. The basic cycle of a memetic algorithm is shown in Algorithm 1.1. In this sketch, n represents the iteration counter or generation counter, the variables in capital letters represent populations, and Φ represents the fitness values. This nomenclature is adopted hereafter.

Observe that an explicit local search step is performed after the variation step. The variation step represents the usual operators of evolutionary algorithms - crossover and mutation. Actually, there are some variations in which the local search can be executed before the traditional genetic operators. In Baldwinian algorithms, for instance, the local search is performed before the selection step, in order to bias the selection of individuals. The local search can also be applied to all individuals in the population or to a fraction of them. These and other important aspects in the design of a memetic algorithm are

Algorithm 1.1: Basic cycle of a memetic algorithm.

Data: population size μ, search space, objective and constraint functions

```
1 P(n = 0) ← Initialize population;
2 while ¬ stop criteria do
3 |   Φ(n) ← Evaluate(P(n));
4 |   S(n) ← Selection(P(n), Φ(n));
5 |   V(n) ← Variation(S(n));
6 |   Q(n) ← Local Search(V(n)) ;          /* Explicit local search */
7 |   P(n + 1) ← Q(n);
8 |   n ← n + 1;
9 end
```

essential points for a good performance of the method (Krasnogor and Smith 2005)

The optimization by hybrid evolutionary algorithms is an important step in the process of computer aided design (CAD) in engineering. The context of CAD is briefly discussed in the next section.

1.2 The computer aided design process

In a computer aided design (CAD) problem, the design process is automated by using a mathematical model of the device or system to be designed. Figure 1.1 presents a flowchart description of the basic design process. In this flowchart, we can identify the fundamental blocks of a complete design process. The process starts with the establishment of specifications and requirements, which leads to the definition of a prototype, a first design model to the problem at hand. These initial steps depend on the knowledge and intervention of a human designer. The next step is to generate the computational geometric model of the prototype, whose description depends on the analysis method to be used. The main purpose of the prototype (and its associated computational model) is to provide a parameterized search space, and a set of objective and constraint functions for the optimization process. The goal of the optimization process is to find the solution, in the given search space, that moves the current design (current prototype) to the nearest possible prototype which will satisfy the requirements. Finally, if the optimized prototype satisfies the specifications and requirements defined by the designer, then the design process stops and the final solution is achieved.

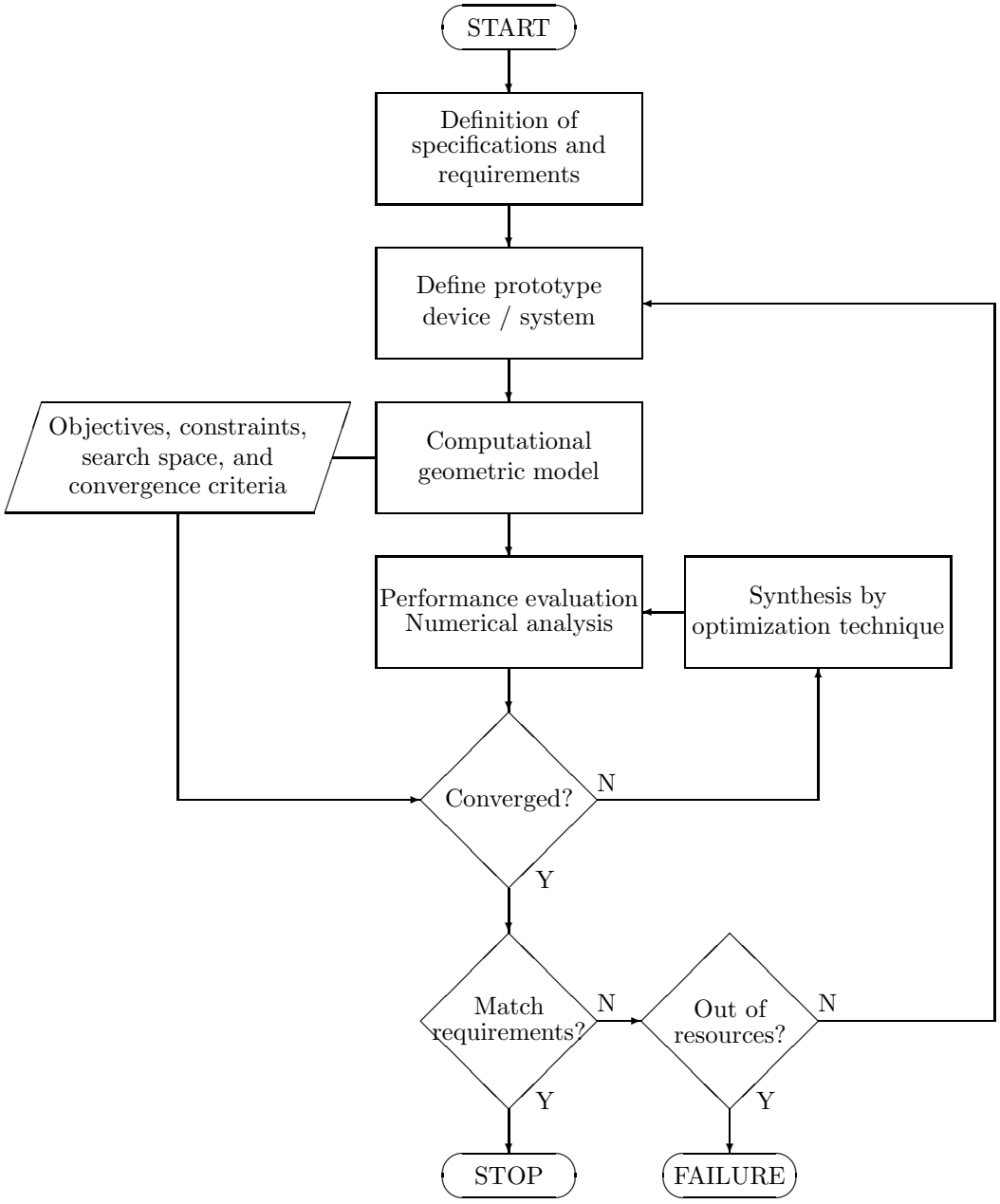

Figure 1.1: Flowchart description of the computer aided design process.

1.2.1 Where does the optimization fit?

From the prototype device and its computational model, the designer can produce a mathematical definition of the design problem, which consists of two fundamental elements: (i) the synthesis problem and (ii) the analysis problem.

The synthesis problem translates the product specifications and requirements into objective and constraint functions, and the design variables into optimization variables with their respective ranges. In other words, the synthesis problem is described by an optimization problem instance of the form:

$$\begin{aligned} &\min \mathbf{f}(\mathbf{x}) \in \mathbb{R}^m \\ &\text{subject to: } \mathbf{x} \in \Omega \end{aligned} \tag{1.1}$$

in which $\mathbf{x}$ are the optimization variables, $\mathbf{f}$ are the objective functions and Ω is the feasible set, mathematically defined by the constraint functions. In Chapter 2 we give a more precise statement of the optimization problem.

Therefore, the optimization process is an important step in any automated design process. This optimization problem requires an adequate optimization algorithm to search for the best solutions. During the search process, the evaluation of each possible design solution demands the solution of the analysis problem. The analysis problem consists of simulating the current prototype parameterized by the trial solution provided by the optimization algorithm in order to calculate a given quantity (like an intermediate variable) that is used in the definition of one or more of the objective and/or contraint functions. The simulation of the device or system means actually solving the mathematical equations that describe the laws that physically govern its behavior. In this situation, the evaluation of a single trial solution may involve the solution of one or more partial differential equations. When nonlinearities and accuracy requirements are incorporated, the computational cost becomes even more relevant. In coupled problems, the analysis step may even involve the association of different analysis softwares (Almaghrawi and Lowther 2007), which further increases the computational cost. This synthesis-analysis cycle is shown in the inner loop of the flowchart in Figure 1.1. When the prototype is optimized, its performance is evaluated to see if it meets the requirements defined in the beginning of the CAD process. If it does not meet the requirements, we can either give up due to lack of resources and rethink the design specifications or return to step 2 to modify the prototype, hence modifying the objective and contraint functions and the

search space. Notice that the outer loop leads to the definition of different optimization problem instances of the form in (1.1).

The software responsible for solving the analysis problem is treated as an expensive black-box. From the optimization point of view, it does not matter what is inside this black-box, as long as it provides consistent outputs[2]. In some problems, all objective and constraint functions are functions of the black-box output value. In this case, the time to evaluate one individual in a population-based evolutionary algorithm is simply the time consumed by the black-box. In other cases, only some of these functions depend on the black-box software, while the others are analytically defined and usually inexpensive. This is a slightly different interface, in which the total evaluation time is the sum of the black-box evaluation time, usually dominant, plus the time to evaluate the inexpensive functions.

1.3 Optimization strategies

In this section we briefly discuss available strategies that are suitable for the synthesis problem discussed before. Chapter 2 provides a deeper presentation of evolutionary techniques and a more rigourous definition of the optimization problem.

1.3.1 Local search methods

Local search methods are in general[3] iterative deterministic methods that aim at locally refining the current solution. Therefore, the next solution is given by:

$$\mathbf{x}(n+1) \leftarrow \mathbf{x}(n) + \eta(n)\mathbf{d}(n) \tag{1.2}$$

where $\mathbf{x}(n)$ represents the current solution, $\eta(n)$ is a scalar, and $\mathbf{d}(n)$ is an update vector. In gradient-based methods, this update vector is a direction in the search space that is a function of the gradient vectors of the objective and constraint functions.

The flowchart of this class of methods is shown in Figure 1.2. Observe that the search is restricted to a local vicinity around the initial solution.

[2]The term consistent is used here in the sense that the same input produces the same output whenever the black-box is evaluated.

[3]There are stochastic methods that under given circumstances behave as local search methods.

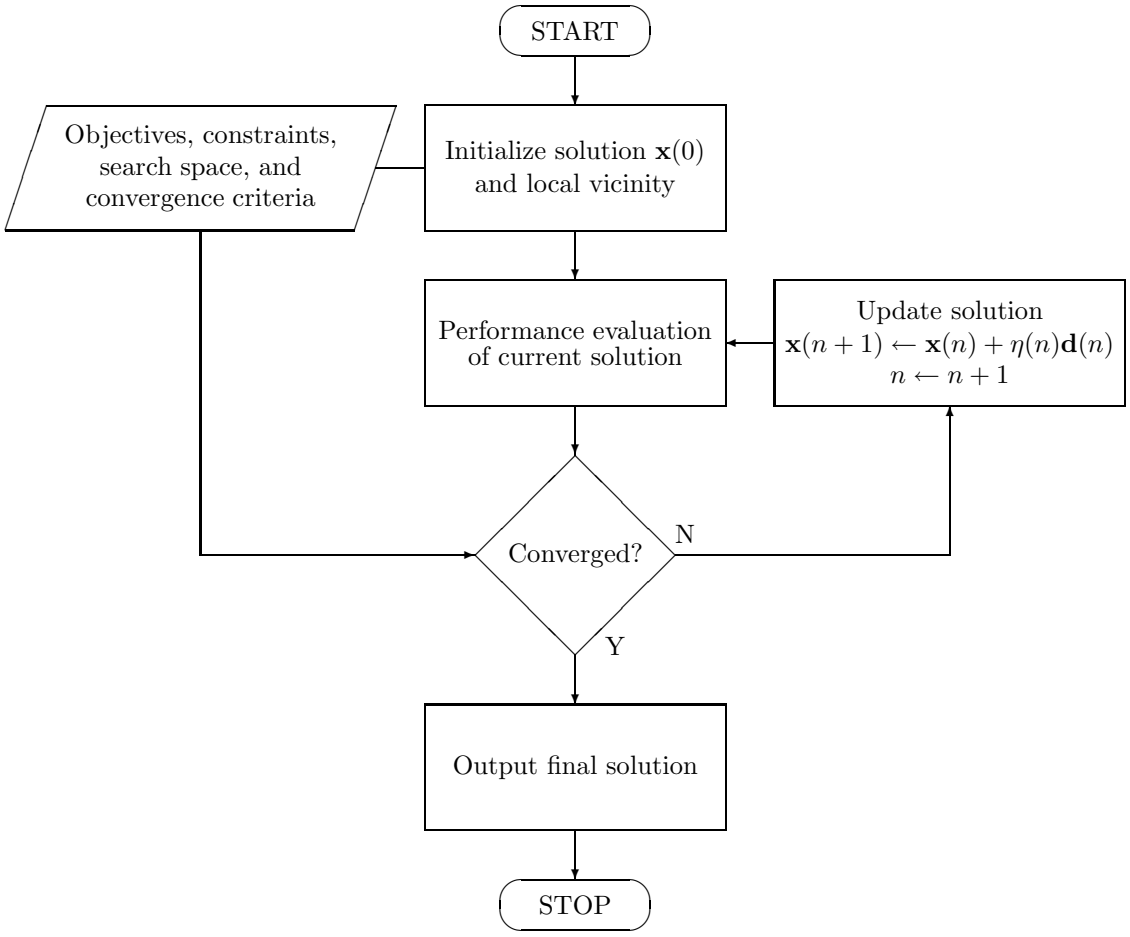

Figure 1.2: Flowchart description of local search methods.

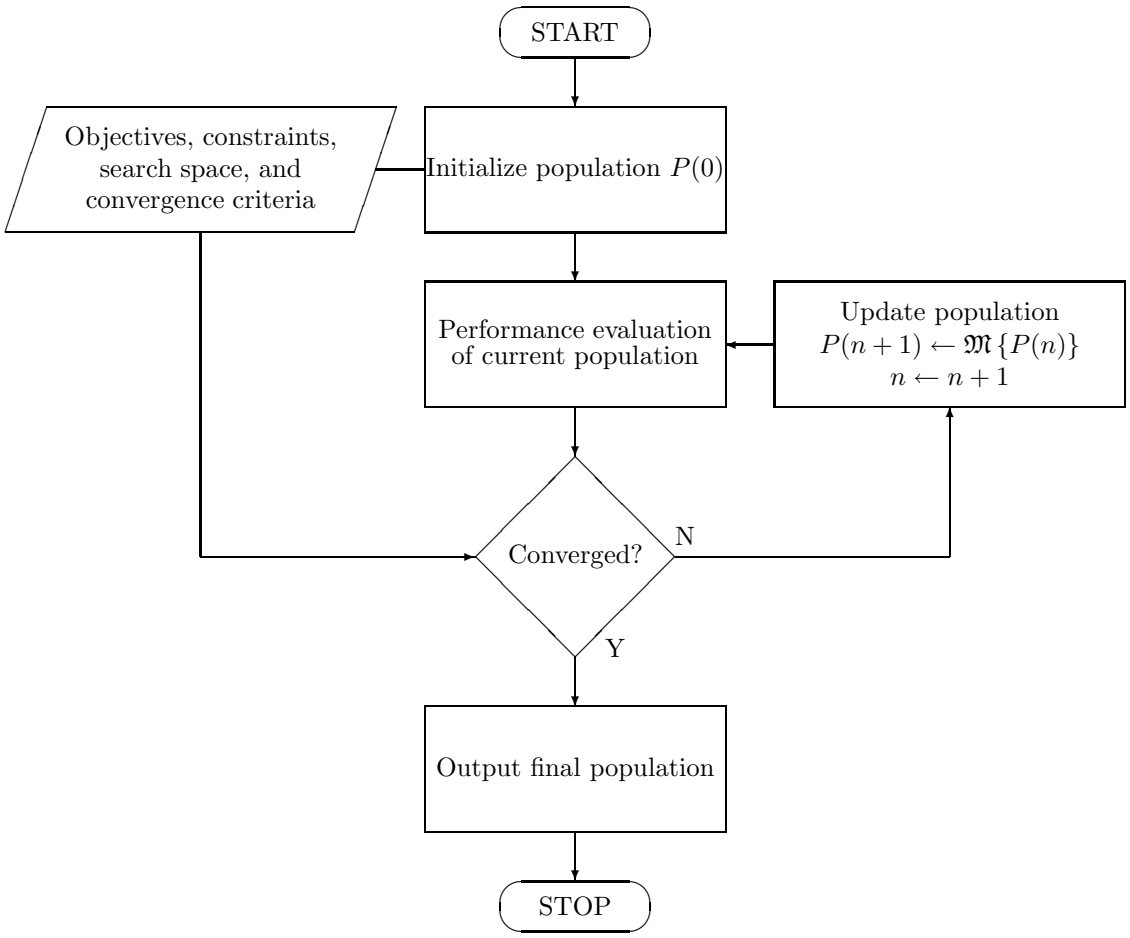

Figure 1.3: Flowchart description of evolutionary methods.

1.3.2 Evolutionary techniques

Evolutionary techniques are a special class of population-based global search methods, which are optimization methods that work with a population of solutions. This population is iteratively modified through the application of sequential heuristic operators with some random component. The current population is updated according to:

$$P(n+1) \leftarrow \mathfrak{M}\{P(n)\} \tag{1.3}$$

where $\mathfrak{M}\{\cdot\}$ represents the mapping performed by the operators of the algorithm from $P(n)$ to $P(n+1)$. The flowchart of this class of methods is shown in Figure 1.3.

Evolutionary algorithms, in the sense of optimization algorithms inspired by natural evolution, are neither the only class of population-based global search methods nor the

only class of nature-inspired optimization techniques. Other classes of methods include particle swarm optimization methods, artificial immune systems algorithms, simulated annealing methods, tabu search, and others.

1.3.3 Moving to memetic algorithms

As already mentioned, memetic algorithms employ local search methods as an additional operator in the mapping $\mathfrak{M}\{\cdot\}$. Memetic algorithms can be viewed as the association of the two paradigms described before.

In fact, the association of local search methods and global search methods has many advantages, due to the benefits that local searchers provide:

- Many local search methods are available today to the designer. These methods take advantage of characteristics such as convexity and differentiability of the functions;

- There are very specialized methods for treating constraints in optimization problems, including equality constraints (Takahashi, Saldanha, Dias-Filho and Ramírez 2003, Guimarães 2004, for instance). In general, these techniques rely on assumptions that favor their utilization as local search techniques. The handling of constraints in evolutionary algorithms is not trivial, and much research has been done on this subject (Coello 2002, Venkatraman and Yen 2005);

- In general, local search techniques are less sensitive to the increase in the dimensionality of the problem. Evolutionary algorithms, in contrast, suffer from the increase in the dimensionality of the problem, which is reflected in their poorer convergence and accuracy;

Therefore, hybrid strategies can provide beneficial tools to the individuals of a population-based method. Actually, any successful global search method, including evolutionary techniques, should find a good balance between the *exploitation* and *exploration* components. *Exploration* is important to guarantee that the algorithm has global search capability, giving confidence to the designer that it can potentially find the global optimum. The *exploitation* component means spending more searching effort in the most promising regions. The success of global search techniques is closely related to a good combination of these characteristics.

An algorithm such as random search has only the exploration component. It is the most reliable search method, in the sense that it guarantees that each solution (within a small vicinity) has an equal probability of being visited. At the other extreme, we find the deterministic methods, presenting only the exploitation component. They are the most efficient in the task of finding the closest optimum. They are able to find the best solution within a restricted local search area, but they cannot explore other regions. In a region between these two extrema, we find the evolutionary techniques, trying to find the best compromise between reliability and efficiency through the combination of global and local search components. However, the main difficulty is that the best compromise solution is strongly problem-dependent.

The success of evolutionary algorithms has always been related to a good balance between these components. Genetic algorithms, for instance, have always benefited from operators that provide some control over the global-local search balance, likewise the simulated binary crossover (Deb and Beyer 2001) and the real-biased crossover (Takahashi, Vasconcelos, Ramírez and Krahenbuhl 2003), amongst others (Lozano, Herrera, Krasnogor and Molina 2004, Ortiz-Boyer, Hervás-Martínez and García-Pedrajas 2005). More recently, algorithms based on the clonal selection principle[4] have presented great potential due to their clear combination of exploration and exploitation components (Campelo, Guimarães, Igarashi and Ramírez 2005, Campelo, Guimarães, Igarashi, Ramírez and Noguchi 2006). Memetic algorithms clearly follow this tendency, in which a good association of global and local search can favour the optimization process as a whole.

Assuming that we use a perfect local search method, in the sense that it always achieves the minimum point of the attraction basin to which the individual belongs, and also considering that all individuals undergo a local search, we observe that the original search space $\mathcal{X}$ (see detailed definition in the next chapter) reduces to the subset $\mathcal{X}^*$ of local optima and the subset $\mathcal{X}^\circ$ of points such that the local search fails (flat regions, for example). The search space to be explored by the global search side of the hybrid algorithm is then greatly reduced. However, this reduction has a price to be paid, which is the computational cost of the local search step, the local search side of the hybrid algorithm. Therefore, even if the convergence in terms of the number of generations is accelerated, the hybrid algorithm consumes more evaluations for each generation than the standard evolutionary algorithm, that is, each generation is more

[4]The clonal selection principle is a theory advanced by Frank MacFarlane Burnet in 1957 to explain the dynamics of the adaptive immune system in response to an antigen. This principle is largely used in the design of optimization algorithms based on artificial immune systems, a new paradigm in computational intelligence.

expensive. Accordingly, the local search is not applied to all individuals and sometimes, not in all generations. In other words, just a fraction of the population is selected for local search and only within predefined intervals. The specification of the local search intensity and frequency is referred as the balance between global and local search (Ishibuchi, Yoshida and Murata 2003). Therefore, the search space $\mathcal{X}$ does not reduce to $\mathcal{X}^* \cup \mathcal{X}^\circ$, but to something intermediate.

To conclude this section, we present in Figure 1.4 an overview of stochastic population-based global search methods. This diagram includes memetic algorithms as a Darwinian evolutionary technique. The next section provides the historical context of this thesis.

1.4 Historical context

In (Goldberg 1989, p. 202–204), the author already presented initial ideas about the association of local search with genetic algorithms, however, without going deeply into the topic. In fact, there was not much research on hybrid algorithms at that time. The work of (Moscato 1989) is considered the starting point for memetic algorithms. In his work, Moscato developed local search operators in genetic algorithms for the travelling salesman problem (TSP). At the beginning of 1990's, many hybrid genetic algorithms that appeared focused on combinatorial optimization problems. The reason for that is simple: since the popularization of genetic algorithms after Goldberg's work (Goldberg 1989), designers realized the need for additional or special operators in some applications, particularly, routing[5] and scheduling optimization problems, given that genetic operators generated bad or even infeasible solutions in these contexts. Hence, some operators were created to "fix" individuals produced by the genetic operators. Next, taking advantage of available heuristics for these problems, designers soon developed operators that also performed local refinements of the individual. In this way, many memetic algorithms arose in the context of combinatorial optimization (Moscato 1999, Merz and Freisleben 1997, Merz and Freisleben 1999, Burke and Smith 2000).

Recently, some researchers have studied the use of memetic algorithms for optimization with continuous variables (Hu, Huang and Wang 2003, Kimura and Konagaya 2003, Ong and Keane 2004, Lozano, Herrera, Krasnogor and Molina 2004). In (Ong and Keane 2004), an interesting memetic model is proposed: various local optimizers are available to the individuals of the population, which learn along with the optimization

[5]The travelling salesman problem and its variants.

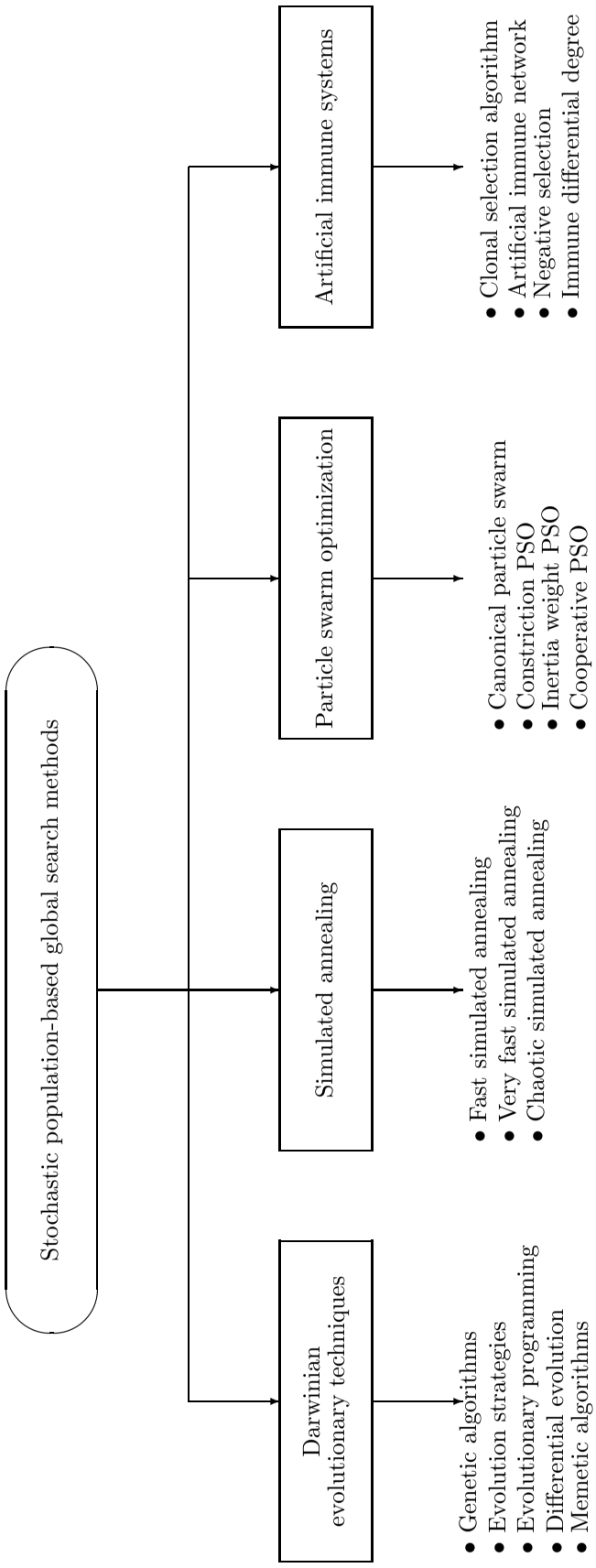

Figure 1.4: Overview of some population-based global search methods.

process which operator is the more efficient for the problem at hand.

In parallel with the development of memetic algorithms, there was the development of the Baldwinian approach (Mayley 1996, Turney 1996) and the Lamarckian approach (Grefenstette 1991, Ross 1999). In the first case, the local search is used to modify the fitness value of the individual, but without changing its genotype. In the second case, the local search does modify the genotype of the individual. Lamarckian and memetic algorithms are also covered in the book (Gen and Cheng 1997).

Also during the 1990's, the first memetic algorithms for multi-objective problems began to appear. The first algorithm combining local search in genetic algorithms was the Multi-objective Genetic Local Search – MOGLS – (Ishibuchi and Murata 1996). The MOGLS has been extensively employed in combinatorial problems, see for instance (Ishibuchi and Murata 1998, Ishibuchi and Murata 1999, Ishibuchi, Yoshida and Murata 2003), but in principle it can be adapted to real-valued multi-objective problems. Next, (Czyzak and Jaszkiewicz 1998) proposed a multi-objective simulated annealing algorithm as a local search method in hybrid algorithms. Finally, in (Knowles 2002, Knowles and Corne 2000b) the authors presented the M-PAES, a memetic algorithm that consists of a genetic algorithm coupled with the Pareto Archived Evolution Strategy – PAES – algorithm, described in (Knowles and Corne 1999, Knowles and Corne 2000a), as a multi-objective local optimizer. For a discussion on multi-objective memetic algorithms and a more detailed historical review, see (Knowles and Corne 2004). Adaptive memetic algorithms are discussed in (Ong, Lim, Zhu and Wong 2006). A special edition dedicated to memetic algorithms in diverse contexts can be found at the February 2007 issue of the journal IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics.

In the context of optimization applied to the design of electromagnetic devices, the hybridization of local optimizers with global search methods is well-established and widely used, however using a simpler approach. This classic approach consists of first executing a global search method, usually an evolutionary algorithm, and then running a local search algorithm using the best solution found by the global search as starting point (Yang, Park, Park and Ra 1995, Renders and Flasse 1996). The global search algorithm runs for a reduced number of iterations, just to identify a promising attraction basin, and the local search method is launched in that region to further refine the solution. Some work in this direction, concerned with the design of electromagnetic devices, can be found in (O. A. Mohammed 1997, Vasconcelos, Saldanha, Krahenbuhl and Nicolas 1997, Starzýnski and Wincenciak 1998, just to name a few). Notice that memetic algorithms adopt a distinct approach, since they employ this association during

the evolutionary cycle, not only after the entire evolutionary process. This is a significant difference: in the traditional approach the hybridization is done in two steps, while in memetic algorithms the local optimizer is part of the evolutionary cycle.

However, the local search step implies in an important additional cost, especially in optimization with continuous variables. When we consider practical problems, in which the fitness evaluation of one individual can demand considerable computational effort, this point becomes even more significant. Therefore, the cost of the local search is an important issue to be examined in the design of a memetic algorithm for this application domain. The work in this thesis circumvents this difficulty by employing local approximation techniques, with the goal of reducing the computational cost of the local search step.

The philosophy of utilizing function approximations in expensive optimization problems is already well understood, especially in applied optimization, an area in which this approach becomes even more useful (Brooker, Dennis, Frank, Serafini, Torczon and Trosset 1998). Traditionally, a global approximation is generated from a previous sampling of the expensive black-box function. We can find much work in this direction using different techniques such as artificial neural networks (Ishikawa, Tsukui and Matsunami 1996, Ebner, Magele, Brandstatter and Richter 1998), neuro-fuzzy networks (Rashid, Ramírez and Freeman 1999, Malik and Rashid 2000, Guimarães, Barros and Ramírez 2003), multi-dimensional interpolation techniques (Alotto, Caiti, Molinari and Repetto 1996, Alotto and Nervi 2001, Davey 2003), and also Kriging techniques (Ratle 2001, Lebensztajn, Marretto, Costa and Coulomb 2004). Yet another possible approach is to employ dynamic global approximations, which are updated and refined during the optimization process (Farina and Sykulski 2001, Tsao and Webb 2005, Guimarães and Ramírez 2006, to see some examples).

Also recently, researchers from the evolutionary optimization field have examined the use of approximations in problems with computationally intensive fitness functions (Bhattacharya and Guojun 2003, Bueche 2003, Jin, Olhofer and Sendhoff 2002). In (Jin, Olhofer and Sendhoff 2002), a combination of approximations and evaluations of the expensive function is implemented. The authors argue that it is difficult to build a consistent and accurate global model that guarantees the same global solution as the true functions. For this reason, some level of online sampling is needed to control and monitor the quality of the approximation. In (Knowles 2006), the author considers the use of a Gaussian process model to approximate the objectives in expensive multiobjective problems. The model is updated with new evaluations of the true objective

functions, hence characterizing a dynamic model. Another interesting direction is the fitness inheritance concept (Sastry, Goldberg and Pelikan 2001). In this approach, the fitness values of the offspring are estimated from the fitness values of the individuals in previous generations. For an overview of the use of approximations in evolutionary optimization, see (Jin 2005). Some recent work have investigated the use of surrogate-assisted local search in memetic algorithms (Zhou, Ong and Lim 2007), including its association with global approximations (Zhou, Ong, Nair, Keane and Lum 2007).

Although the dynamic approximation approaches gradually refine the models, they are still global models, aiming to be precise and reliable in the entire search space. When we consider more complex functions and more variables, the approximated model can become too complex and even produce unsatisfactory results. Moreover, it can be necessary to evaluate many samples to obtain a reasonable approximation, making the methodology not practical at all. This is why employing local approximations restricted to a small region of the search space can be a more interesting and worthwhile strategy. In this way, only samples in a close vicinity are used to generate a local approximation, which provides cheaper estimates of the true function values for points in that region. Such an approach has been investigated more recently in the literature. For example, in (Regis and Shoemaker 2004) the authors propose the use of local approximations to reduce the computational cost of evolutionary algorithms in expensive optimization problems. The local approximations provide estimates of the fitness values of the offspring. Only those individuals with good estimated fitness values are actually evaluated by the true objective function. With this procedure, not all individuals in the offspring are really evaluated, but only the most promising ones, thus reducing the number of expensive evaluations. A similar approach is adopted in (Guimarães, Campelo, Saldanha, Igarashi, Takahashi and Ramírez 2006), though this was not the focus of the work. The authors employ a multi-objective optimization algorithm based on the artificial immune system paradigm for the initial phase of the optimization, when all samples are stored. Then, local approximations provide the evaluation of the new solutions tried by the algorithm, relying on the exploration made in previous iterations and thus saving some computational effort.

This point seems appropriate to locate this thesis in the historical context presented here. As discussed, methodologies for local approximation are already well-established. Additionally, hybrid algorithms have presented interesting results in diverse applications, but the computational cost of local search techniques can not be neglected in problems such as the ones found in computer aided design. This thesis aims to further advance

the development and investigation of memetic algorithms, with particular attention to problems whose function evaluations involve complex and intensive calculations.

Approximation-based local search could also employ simpler approximations, such as the quadratic approximation. Very recent work in the literature has explored the use of local quadratic approximations within the local search (Wanner, Guimarães, Takahashi, Saldanha and Fleming 2005, Wanner, Guimarães, Takahashi, Lowther and Ramírez 2007, Wanner, Guimarães, Takahashi, Lowther and Ramírez 2008). When the local approximations are quadratic, the local search problem can be transformed into an LMI-based linear optimization problem, which is very efficient from a computational point of view. However, this methodology has some drawbacks: (i) the LMI-based local search operator cannot yet deal with problems with mixed equality and inequality constraints, or problems with more than one equality constraint, and (ii) all nonlinear functions in the optimization problem have to be approximated by quadratic functions. The methodology proposed in this thesis is more general for the following reasons:

- It can tackle problems with any number of inequality and equality constraints;
- It allows the combination of different approximation techniques, that is, each nonlinear function in the problem can be approximated by a specific method;
- The designer could decide which nonlinear functions in the problem are approximated and which are not. It is possible that one of the objective or constraint functions are analytical functions or fast-to-evaluate functions. Therefore, one does not need to generate an approximation for it. For example, the design problem of a given electromagnetic device may have among its objectives the minimization of the volume. This volume may be analytically expressed or calculated without much computational effort. One can use this precise calculation instead of an approximation. Situations like this are easily accomodated by the methodology in Chapter 3, which do not oblige that all nonlinear functions be approximated.

1.5 Objectives of the thesis

The main objectives of this thesis are to investigate, implement and test the use of local approximations within a memetic optimization framework for the solution of CAD problems, particularly the design of electromagnetic devices. The work will present the following main parts:

1. Organization of a framework for evolutionary optimization applied to CAD problems. This framework needs to accomodate various evolutionary algorithms for both mono and multi-objective optimization, including memetic algorithms. This thesis will identify a common structure and hence unify these algorithms into the framework.

2. Definition of the methodology for incorporating local approximations in the local search phase of memetic algorithms. This methodology will be defined as an operator for the framework of algorithms developed in the previous part.

3. Investigation of the effect of the local search operators on the convergence properties of standard evolutionary algorithms. The convergence study will be accomplished via Markov chain theory. Additionaly, we will analyze the computational complexity of the proposed approximation-based local search. This study has the goal of studying how the local search affects the global convergence and the complexity of the memetic algorithm.

4. Implementation, test and validation of the proposed methodology in practical problems comprising the optimization of electromagnetic devices.

1.6 Scope of the thesis

The alternative presented in this thesis is the employment of local approximations to reduce the computational burden of the local search step. In this approach, the local search operator uses approximations of the original functions in the problem, which are built with the information acquired by the current and previous populations. Using information from the history of past populations, the individuals generate local approximations to finally undergo their local improvement.

As discussed before, the use of approximations in optimization is not new, but traditionally global approximations have been adopted. The use of local approximations is relatively new. If many samples are needed to build a good global approximation, then this approach is not worth using in practice. On the other hand, local approximations are simpler to obtain and more precise, given that the functions become smoother and simpler when taken locally. The local operators can then be executed with no cost in terms of function evaluations.

Given the additional effort in generating the local approximations, the proposed strategy is useful in the context of optimization problems with computationally intensive objective and constraint functions, such as the ones found in some engineering problems. In this context, the additional complexity in the algorithm would be justifiable.

Therefore, it is necessary to clearly characterize the class of problems that are intended to be treated in this work. In this thesis we consider:

- nonlinear optimization problems with real variables;
- nonlinear objective functions that are possibly multimodal;
- nonlinear equality and inequality constraint functions that are possibly multimodal;
- the calculation involved in at least one of the objective and constraint functions is computationally expensive in such a way that the time required for the evaluation step is dominant in comparison with the time required by the other steps.

A great number of optimization problems of practical interest fall into this characterization. More specifically, the class of computer aided design (CAD) problems of engineering devices, which involve modelling of electromagnetic, mechanical, thermal, etc., phenomena, frequently fall into formulations of a similar type to that delineated above.

In industrial and practical applications, the designers often want (i) to minimize the time to achieve an adequate solution; or (ii) to maximize the quality of the solution in a given time budget. However, minimizing the optimization time (maximizing efficiency) and maximizing the quality (maximizing reliability) are generally conflicting objectives, and, as discussed before, a good trade-off between these extrema are search algorithms that present a good balance between the exploration and exploitation components. Hybrid algorithms are a good alternative in both cases since (i) they can converge faster, or (ii) if the time budget is limited, they guarantee that at least a local optimum is maintained during the search.

Evolutionary algorithms play an important role in the solution of complex CAD problems (Renner and Eckart 2003). With the help of local approximations, the memetic algorithms, which have been shown to be powerful optimization tools in various classes of problems, can also be applied to the class of CAD problems. The methodology

examined in this thesis allows that memetic algorithms provide important contributions in the solution of these problems. Observe that in the context of combinatorial problems, the evaluation time is not significant, though this is not always the case. Hence, the cost of the local search is not of much concern. Therefore, the great advancements achieved in evolutionary algorithms, deterministic algorithms, and approximation techniques can be put together to form a sophisticated framework for the class of problems identified before.

1.7 Outline of the thesis

The structure of the text is organized as follows:

Chapter 2 is dedicated to the characterization of mono and multi-objective optimization problems with continuous variables and the presentation of the main evolutionary techniques for solving these problems. After reviewing the extensive work available in the literature, the Chapter concludes with the identification of a common structure that can model all these algorithms. Particular attention is paid to the integration of the framewrok via this unified model, thus simplifying its implementation and use.

Chapter 3 starts with a classification and discussion of hybridization schemes. The local search problem for mono and multi-objective contexts is presented. An approximation-based methodology, consisting of the association of radial basis functions and the sequential programming method, is developed to solve this local search problem. The Chapter ends with the derivation of a trust region update formulation for the local search procedure.

Chapter 4 addresses the global convergence analysis of the evolutionary algorithm hybridized with local search operators. The analysis is done by viewing the population as a state and its evolution as a Markov process. By using properties from the Markov chain theory, one can draw conclusions about the effect of the local search on the global convergence properties. The second part of the Chapter is dedicated to the analysis of the computational complexity of the local search proposed in Chapter 3 and its computational overhead in optimization problems, with particular attention to expensive-to-evaluate black-box functions, commonly found in the context of CAD problems.

Chapter 5 presents the results of the optimization of practical electromagnetic devices

utilizing the framework developed in Chapters 2 and 3.

Chapter 6 concludes the thesis with an overall discussion of the work. The Chapter is closed with suggestions for future work.

1.8 Original contributions

The original contributions of this thesis are claimed to be:

- The organization and implementation of a framework of evolutionary techniques for CAD problems. The mono and multi-objective evolutionary algorithms are integrated into a unified algorithm. Therefore, all algorithms in the framework can be viewed as specific instances of this algorithm. Depending on the operators and parameters provided, this algorithm behaves as an evolution strategy, a genetic algorithm, a multi-objective genetic algorithm, etc. Having this unified algorithm is very useful for implementation purposes. It can model evolutionary algorithms such as genetic algorithms, evolution strategies, differential evolution, and also memetic algorithms.

- The proposition of the local search procedure that integrates radial basis function approximations and the sequential quadratic programming method for performing the local improvement of a given solution in the population of the evolutionary algorithm. The procedure is also extended to multi-objective problems. It is also derived an optional trust region update methodology for the local search.

- The derivation of new theorem results for the effect of the local search on the global convergence properties of standard algorithms. The analysis of the computational complexity of the proposed local search is also developed. The analysis leads to an overall expression for the computational complexity that accounts for any number of objective and constraint functions and for any number of trust region updates. An important result obtained from this analysis is that the memetic algorithms implemented in this work preserve the polynomial complexity of conventional evolutionary algorithms.

- The application of the implemented framework in the optimization of electromagnetic devices. The magnetizer problem illustrates the unimportance of the overhead of the proposed local search in CAD problems. The mono and multi-objective

formulations of the TEAM benchmark problem 22 are also used to test the framework and to verify the increase in the convergence speed of hybrid evolutionary algorithms.

Chapter 2

Evolutionary Optimization

"Charles Darwin formulated one of the most extraordinary ideas of the Western mind, the idea of evolution by means of natural selection."

— Donald Johanson, 1943–

"Imagine if, as a cosmic joke, humans have gradually evolved to leave many of us doubting evolution."

— Nicholas D. Kristof, 1959–

2.1 Preview

In Chapter 1 we provided an overview of the computer aided design process and discussed the role of optimization in this process. We presented an introduction of possible ways to approach the optimization problem, with special focus on evolutionary algorithms and memetic algorithms.

In this chapter we start with the mathematical formalization of the optimization problem in both mono and multi-objective scenarios. Real-world design solutions must meet diverse specifications and satisfy practical needs and restrictions that are translated into an optimization problem instance. The final product has to be "optimal" under many different aspects, sometimes conflicting ones. In the case of mono-objective

problems, we deal with a problem with only one performance criterion and a few mathematical constraints. Given these specifications, we search for the best solution. In some situations, we may be interested in finding not only the best solution, but also some "good" alternatives. In the context of multi-objective optimization, we search for a set of trade-off solutions. The optimization process becomes more complex since we need to redefine the concept of what an optimal solution is and, after finding some set of possible alternatives, we need to decide which one will be implemented.

Engineers, economists, and computer scientists soon realized the importance of developing powerful techniques for tackling these problems in different fields and applications. Although the work on evolutionary computation can be traced back to the 1950's, it was from the 1970's on that evolutionary techniques developed and became widely used. Starting from the 1990's, many multi-objective evolutionary techniques have appeared.

In order to organize and simplify the presentation and discussion of the various evolutionary algorithms in the literature, we begin with a very general evolutionary algorithm, identifying its basic features. Then, we proceed to the presentation of specific instances for mono and multi-objective problems. The concluding part of the chapter describes a framework of evolutionary algorithms that integrates all algorithms presented. In this way, the presentation and discussion of many evolutionary algorithms available in the literature converges to the definition of a unified evolutionary algorithm, which serves as the backbone of the framework proposed in this thesis. We conclude this chapter showing that this unified algorithm can model all the other algorithms presented for both mono and multi-objective contexts. In summary, in this chapter we move from the general to the particular, to finally extend our understanding of the general to a unified view of evolutionary algorithms.

2.2 Definition of the optimization problem

The general optimization problem can be stated as[1]:

$$\begin{aligned} &\min \mathbf{f}(\mathbf{x}) \in \mathbb{R}^m \\ &\text{subject to: } \mathbf{x} \in \Omega \end{aligned} \tag{2.1}$$

[1]The choice for a minimization problem instead of a maximization problem is arbitrary.

in which Ω is the feasible set, mathematically defined by:

$$\Omega = \{\mathbf{x} \in \mathcal{X} : g_i(\mathbf{x}) \leq 0,\ i = 1, \ldots, p; \quad h_j(\mathbf{x}) = 0,\ j = 1, \ldots, q\} \tag{2.2}$$

The functions $g_i(\,\cdot\,)$ and $h_j(\,\cdot\,)$ are respectively the inequality and equality constraints that define the sets:

$$\mathcal{G}_i = \{\mathbf{x} : g_i(\mathbf{x}) \leq 0\} \tag{2.3}$$

$$\mathcal{H}_j = \{\mathbf{x} : h_j(\mathbf{x}) = 0\} \tag{2.4}$$

The set $\mathcal{X} \subset \mathbb{R}^d$ represents the search domain, defined by a particular type of inequality constraints, namely, the box constraints, which define the lower and upper limits of the optimization variables:

$$\mathcal{X} = \{\mathbf{x} : l_k \leq x_k \leq u_k,\ k = 1, \ldots, d\} \tag{2.5}$$

or, written differently, the search domain $\mathcal{X}$ represents the cartesian product below:

$$\mathcal{X} = \prod_{k=1}^{d} [l_k, u_k] \tag{2.6}$$

Therefore, the feasible set can be written as:

$$\Omega = \mathcal{X} \bigcap \left(\bigcap_{i=1}^{p} \mathcal{G}_i \right) \bigcap \left(\bigcap_{j=1}^{q} \mathcal{H}_j \right) \tag{2.7}$$

The functions $f_i(\,\cdot\,) : \mathcal{X} \mapsto \mathbb{R},\ i = 1, \ldots, m$, are the objective functions of the problem. Each objective function maps the search domain into a subset of the Real number line. The vector of objective functions $\mathbf{f}(\,\cdot\,) : \mathcal{X} \mapsto \mathcal{Y}$ maps the search domain $\mathcal{X} \subset \mathbb{R}^d$ into a subset of the objective space $\mathcal{Y} \subset \mathbb{R}^m$.

The goal of the optimization process is to minimize the objective functions within the feasible set Ω. However, there is no unique solution for this problem, except when a utopian solution exists, i.e., the minima of all objectives coincide in the feasible space. Although this occurs often in single-objective problems, this is a very rare situation in practical multi-objective problems. Usually, in such cases, we have a set of trade-off solutions for the problem.

The first point to be addressed in multi-objective optimization is the concept of optimal solutions. First, we present the definitions related to each objective function individually. After that, we introduce the optimality concept when all functions are considered simultaneously.

Definition 2.1. (Locally optimal solution) A solution $\mathbf{x}^* \in \Omega$ is said to be locally optimal if there exists an $\epsilon > 0$ such that for all $\mathbf{x} \in \Omega$ satisfying $\|\mathbf{x} - \mathbf{x}^*\| \leq \epsilon$ we have $f(\mathbf{x}^*) \leq f(\mathbf{x})$.

In other words, the definition above states that it is not possible to find another solution better than $\mathbf{x}^*$ by locally perturbing this point within Ω. If we choose ϵ in such a way that all points belonging to the feasible set satisfy $\|\mathbf{x} - \mathbf{x}^*\| \leq \epsilon$ and $\mathbf{x}^*$ remains optimal, we can say that the solution is globally optimal. That leads to the following definition.

Definition 2.2. (Globally optimal solution) A solution $\mathbf{x}^* \in \Omega$ is said to be globally optimal if $\forall \mathbf{x} \in \Omega$ we have $f(\mathbf{x}^*) \leq f(\mathbf{x})$.

When we consider all objectives simultaneously, we need to define the comparison between vectors in order to extend the optimality definitions.

Definition 2.3. Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$. The following relations are defined as:

$$\mathbf{x} \leq \mathbf{y} \Rightarrow \{x_i \leq y_i,\ i = 1, \ldots, m\} \tag{2.8}$$

$$\mathbf{x} < \mathbf{y} \Rightarrow \{x_i < y_i,\ i = 1, \ldots, m\} \tag{2.9}$$

$$\mathbf{x} = \mathbf{y} \Rightarrow \{x_i = y_i,\ i = 1, \ldots, m\} \tag{2.10}$$

$$\mathbf{x} \neq \mathbf{y} \Rightarrow \{\exists i : x_i \neq y_i\} \tag{2.11}$$

Definition 2.4. (Dominance) A vector $\mathbf{x}$ dominates another vector $\mathbf{y}$ if and only if $\mathbf{x} \leq \mathbf{y}$ *and* $\mathbf{x} \neq \mathbf{y}$, that is, there is at least one index i such that $x_i < y_i$. This dominance relation is described as $\mathbf{x} \prec \mathbf{y}$ or $\mathbf{y} \succ \mathbf{x}$, which indicates, respectively, that $\mathbf{x}$ dominates $\mathbf{y}$ and $\mathbf{y}$ is dominated by $\mathbf{x}$.

Definition 2.5. Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$. If both relations $\mathbf{x} \prec \mathbf{y}$ and $\mathbf{y} \prec \mathbf{x}$ are not verified, then we say that both vectors are nondominated by each other, or incomparable.

In fact, when two vectors $\mathbf{x} \neq \mathbf{y}$ do not dominate each other, we necessarily observe $x_i < y_i$, $i \in \mathcal{I}$ and $x_j > y_j$, $j \in \mathcal{J}$, with $\mathcal{I} \cap \mathcal{J} = \varnothing$ and $\mathcal{I} \cup \mathcal{J} \subset \{1, 2, \ldots, m\}$.

In the context of optimization, we can employ dominance relations with solutions in the search domain:

Definition 2.6. Let $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}$. We say that $\mathbf{x}_1$ dominates $\mathbf{x}_2$ if $\mathbf{f}(\mathbf{x}_1) \prec \mathbf{f}(\mathbf{x}_2)$.

After presenting these definitions, we can finally state the optimality concepts for multi-objective optimization.

Definition 2.7. (Locally Pareto-optimal solution) A solution $\mathbf{x}^* \in \Omega$ is said to be locally Pareto-optimal if there exists an $\epsilon > 0$ such that for all $\mathbf{x} \in \Omega$ satisfying $\|\mathbf{x} - \mathbf{x}^*\| \leq \epsilon$ we do not verify $\mathbf{f}(\mathbf{x}) \prec \mathbf{f}(\mathbf{x}^*)$.

Putting it differently, this definition says that we cannot find another solution that dominates $\mathbf{x}^*$ by locally perturbing this solution.

Definition 2.8. (Globally Pareto-optimal solution) A solution $\mathbf{x}^*$ is said to be globally Pareto-optimal if $\nexists \mathbf{x} \in \Omega$ such that $\mathbf{f}(\mathbf{x}) \prec \mathbf{f}(\mathbf{x}^*)$.

As we said before, there may be many solutions in the search domain meeting this definition. Therefore, in multi-objective problems we have a set of solutions, whose definition is given below:

Definition 2.9. (Pareto-optimal set) Given a problem as in (2.1), its Pareto-optimal set is defined as:

$$\mathcal{X}^* = \{\mathbf{x} \in \Omega | \ \nexists \mathbf{z} \in \Omega : \mathbf{f}(\mathbf{z}) \prec \mathbf{f}(\mathbf{x})\} \tag{2.12}$$

Definition 2.10. (Pareto-optimal front) Given a problem as in (2.1) and its Pareto-optimal set $\mathcal{X}^*$, the Pareto-optimal front is:

$$\mathcal{Y}^* = \{\mathbf{y} = \mathbf{f}(\mathbf{x}) : \mathbf{x} \in \mathcal{X}^*\} \tag{2.13}$$

When problem (2.1) has only one objective, the optimal set $\mathcal{X}^*$ contains the global optima as delimited by Definition 2.2. In some applications, still considering one objective, finding some local optima may be helpful. In this case, the definition of the optimal set $\mathcal{X}^*$ must be modified to contain all local optima as delimited by Definition 2.1.

2.3 General structure of an evolutionary algorithm

Although the origins of evolutionary computation can be traced back to the 1950's, it was from the 1970's onwards that the research on evolutionary computation had its biggest growth, thanks to the work of many authors such as Holland, De Jong, Schwefel and Fogel (Bäck 1996, as commented herein). Following, the work in the 1980's and 1990's of authors such as Goldberg, Koza, Fogel, Mitchell, and others (Goldberg 1989, Koza 1992, Fogel 1995, Mitchell 1998, amongst others) contributed to the popularization and maturation of this computational methodology, nowadays an established area and under increasing development.

The relevance of evolutionary algorithms in problem-solving is due to advantages such as flexibility, adaptation, global search capability, and its suitability for parallel computation. Evolutionary optimization methods are stochastic algorithms that work with a population of candidate solutions (or alternatives). They follow a sequence of heuristic operations generally inspired from nature to modify the solutions in the population in such a way that they converge to the solution(s) of the problem, specifically, the optimal set $\mathcal{X}^*$. This sequence of steps changes the current population at each iteration, producing the next population, as described in the following diagram:

$$\underbrace{P(n) \xrightarrow{\text{Selection}} S(n) \xrightarrow{\text{Variation}} V(n) \longrightarrow P(n+1)}_{P(n+1) \leftarrow \mathfrak{M}\{P(n)\}} \tag{2.14}$$

where $\mathfrak{M}\{\cdot\}$ represents the mapping performed by the operators of the algorithm from $P(n)$ to $P(n+1)$.

The operations executed by the algorithm have some stochastic components, but the search process as a whole is far from a random search, since there is a mechanism of selective pressure that directs the search towards the most promising regions. As Dawkins points out[2]:

"Evolution is achieved by nonrandom survival of random hereditary changes".

This is equally appliable to evolutionary optimization methods, especially to emphasize the importance of the selective pressure in the search process.

[2]Dawkins referred to biological evolution when he stated this definition.

A general global search population-based evolutionary algorithm is presented in Algorithm 2.1. This algorithm, though simple, has the basic ingredients to generate a complex behavior similar to the evolutionary process itself: simple rules acting on agents of a population that interact with each other, receiving responses (fitness evaluation) and being subject to some perturbation (variation). With these elements, we can devise an algorithm that presents a complex behavior and produces interesting results.

In the algorithm 2.1, $P(n)$ is the online population, which is directly affected by the modifying operators, and $A(n)$ is the offline population or archive population, which stores the best solution set found by the algorithm. This archive population is updated every time after evaluating the solutions in $P(n)$.

Usually, in a mono-objective optimization problem, $A(n)$ has size $\xi = 1$, that is, only the best solution found by the algorithm is stored and maintained. When we are interested in finding not only one but also a set of optima, $A(n)$ may store more than one solution. This is the case when using niching evolutionary algorithms (Goldberg 1989, Sareni, Krahenbuhl and Nicolas 1998, Gallardo and Lowther 1999, Hui-Zhi, Fa-Chao and Cong-Man 2005) or multimodal immune-based algorithms (de Castro and Timmis 2002, Canova, Freschi and Repetto 2005, Campelo, Guimarães, Igarashi, Ramírez and Noguchi 2006). In some design situations, even in the case of only one objective, the designer may be interested in finding many alternative solutions for choosing *a posteriori* which one is more suitable for implementation purposes. The update method, in these cases, is a bit more complicated, since some diversity mechanism should be imposed on the solution set.

In multi-objective problems, the maximum size of $A(n)$ is greater than one, i.e., $\xi > 1$. The update method for the archive population is also more complicated, since we need to consider dominance relations and a good representation of the Pareto front. Actually, the adoption of an archive population $A(n)$ in multi-objective evolutionary algorithms is nowadays judged as an essential characteristic, being also considered as the dividing line between the first and second generations of evolutionary multi-objective techniques (Coello 2006). Thus, all algorithms that represent the state of the art in evolutionary multi-objective optimization employ such an explicit elitism method (Corne, Knowles and Oates 2000, Deb, Agarwal, Pratab and Meyarivan 2000, Zitzler, Laumanns and Thiele 2001).

The selection operator (see line 5 in Algorithm 2.1) can be applied to individuals of $P(n)$ or to the union $P(n) \cup A(n)$, which is an elitist selection. Once the fitness

Algorithm 2.1: General population-based evolutionary algorithm.

Data: population size μ, maximum archive size ξ, search space $\mathcal{X}$, objective and constraint functions $\mathbf{f}(\cdot)$, $\mathbf{g}(\cdot)$, $\mathbf{h}(\cdot)$.
Result: Estimate(s) of $\mathcal{X}^*$ in the archive population $A(n)$.

```
P(n = 0) = {p^(1), ..., p^(μ)} ← Initialize population(μ, X);
A(n = 0) = ∅ ← Initialize archive /* Stores the best solution set */
while ¬ stop criteria do
    Φ(n) ← Evaluate(P(n),f, g, h);
    S(n) ← Selection(P(n), A(n), Φ(n));
    V(n) ← Variation(S(n));
    P(n + 1) ← V(n);
    A(n + 1) ← Update(A(n),P(n),ξ);
    n ← n + 1;
end
```

values for $A(n)$ and $P(n)$ are determined, the selection works the same for mono and multi-objective algorithms.

As we can see, any population-based technique, either for mono-objective problems or multi-objective ones, can be adequately represented by the baseline Algorithm 2.1. The fundamental differences in all these algorithms found in the literature lie in:

1. the fitness calculation, particularly in multi-objective algorithms;
2. the mechanism for preserving diversity in $A(n)$, when its size is greater than one;
3. the selection mechanism, that can be either deterministic or stochastic;
4. the heuristic operators in the variation mechanism, that is, how the next population is generated based on the current one.

There are many possible ways of implementing these steps, which leads to the great diversity of algorithms related to the general Algorithm 2.1. Moreover, there is no best way of implementing these operations. That depends on the context, as we discuss in the next section.

2.3.1 No Free Lunch theorems

The *No Free Lunch* theorems were presented in (Wolpert and Macready 1997), with important implications in evolutionary computation. The implications of the standard

NFL theorem for multi-objective algorithms are discussed in (Corne and Knowles 2003). Let $\mathcal{A}$ be a set of algorithms and $\mathcal{F}$ be the set of functions that map $\mathcal{X}$ into $\mathcal{Y}$. The main result in (Wolpert and Macready 1997) is repeated here:

Theorem 2.1. *Let $a_1, a_2 \in \mathcal{A}$ be any pair of algorithms that generate a sequence $d_k = \{f(\mathbf{x}(1)), \ldots, f(\mathbf{x}(k))\}$ in k steps. The following result is valid:*

$$\sum_{f \in \mathcal{F}} \mathscr{P}\{d_k | f, k, a_1\} = \sum_{f \in \mathcal{F}} \mathscr{P}\{d_k | f, k, a_2\}$$

Proof. For a detailed proof, see (Wolpert and Macready 1997).

□

It means that when averaged over all possible functions in $\mathcal{F}$, the probabilities of obtaining a particular sequence d_k using a_1 amount to the same as that of using a_2. This theorem has some interesting consequences. For example, in a mono-objective minimization problem, the output of the algorithm is $\min d_k$. Thus:

$$\sum_{f \in \mathcal{F}} \mathscr{P}\{\min d_k | f, k, a_1\} = \sum_{f \in \mathcal{F}} \mathscr{P}\{\min d_k | f, k, a_2\} \tag{2.15}$$

or, more generally:

$$\sum_{f \in \mathcal{F}} \mathscr{P}\{\psi(d_k) | f, k, a_1\} = \sum_{f \in \mathcal{F}} \mathscr{P}\{\psi(d_k) | f, k, a_2\} \tag{2.16}$$

where ψ is a function that measures performance.

If a_1 is a random search algorithm and a_2 is a given evolutionary algorithm, we see from (2.16) that the evolutionary algorithm can perform better than random search in some problems but worse in others. The theorem is a representation of the intuitive idea that there is no tool that is optimal in all classes of problems. This result may sound in constrast with the common notion of robustness of evolutionary algorithms. Although they are applicable in a wide range of problems, it does not mean they are the most efficient for a given problem. In many problems, some level of specialization is needed to make the algorithm efficient. Nevertheless, in the absence of *a priori* knowledge, evolutionary algorithms are a good alternative due to their flexibility.

Theorem (2.1) still holds for multi-objective optimization, but, as discussed in (Corne and Knowles 2003), it is not as applicable in the multi-objective context as it is in the

mono-objective case. The main reason for that is that scalar performance measures of two multi-objective algorithms are not consistent: different performance metrics lead to different conclusions about which algorithm is better. As a consequence, a_1 is better than a_2 for some metrics, but it is worse for others. Additionally, two different multi-objective evolutionary algorithms can produce two different archives for the same sequence of visited points d_k, because the rules used in their update function may be different. As a consequence, the left and right hand sides of (2.16) may not be equal.

There is one more point about the NFL theorems that is worth commenting. The number of possible functions in $\mathcal{F}$ is[3]:

$$|\mathcal{F}| = |\mathcal{Y}|^{|\mathcal{X}|} \tag{2.17}$$

Most of these functions have no inner structure and are not found in real-world problems or CAD problems. Since some of them have no evident structure to be exploited, the random search can, in fact, perform better than another algorithm that does exploit some structure not present in the problem. Any optimization algorithm relies on some assumptions, some underlying structure. For example, gradient-based methods rely on convexity and differentiability premises. Evolutionary algorithms do not rely on these premises, but they base the search on the assumption of "weak locality". The weak locality concept means that the probability distribution of the objective function value at a given point is conditioned by the values already evaluated in the neighborhood of that point. Therefore, their operators are designed in such a way that better individuals have higher probability of being selected for the variation step, then generating more samples close to them. Of course, many problems in $\mathcal{F}$ do not follow this tendency, and are consequently deceptive functions for evolutionary methods.

The class of problems found in CAD optimization problems does have an inherent structure, of which evolutionary algorithms can take some advantage. Memetic algorithms are also based on a given assumption, namely, the premise that local search operators can improve the performance of standard evolutionary algorithms in a certain class of problems. If the local search operator employs a gradient-based method, we are introducing additional premises, specifically, that in a local vicinity the problem is continuous, differentiable, and becomes ultimatelly convex.

[3]In combinatorial optimization problems with finite domains, the search region is finite, then $|\mathcal{X}|, |\mathcal{Y}| < \infty$. In continuous optimization problems, which is the subject of this text, we have $|\mathcal{X}| = \infty$. Nonetheless, in digital computers we always have $|\mathcal{X}|, |\mathcal{Y}| < \infty$ and hence $|\mathcal{F}| < \infty$.

There is another important consequence of the NFL theorems: for a subset of functions of $\mathcal{F}$ with similar characteristics, one can design an algorithm specialized in these characteristics that performs better than other algorithms. Of course, when averaged over $\mathcal{F}$, Theorem 2.1 holds.

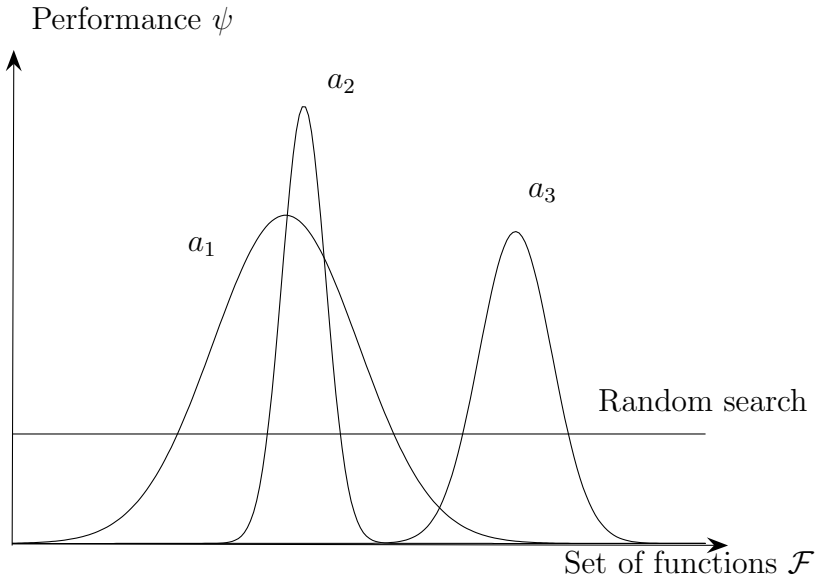

Figure 2.1: Illustration of the NFL theorem

Figure 2.1 illustrates this idea. Algorithm a_3 performs better than the random search and the algorithms a_1 and a_2 for a certain subset of functions in $\mathcal{F}$, but worse in others. For some functions a_2 performs even better than a_1. Nonetheless, the averaged performances (the area below the curves) over $\mathcal{F}$ is equal.

2.4 A framework of evolutionary algorithms

This section describes the evolutionary algorithms implemented in this work. These algorithms make part of a framework of evolutionary algorithms for optimization with real-valued functions.

In all the algorithms described in this section, the individual is directly represented

by its solution vector:

$$\mathbf{p}^{(i)} \triangleq \mathbf{x}^{(i)}, \quad \mathbf{p}^{(i)} \in P(n) \tag{2.18}$$

2.4.1 Fitness calculation

First, we describe the penalty calculation for constrained problems, which is the same for the mono and multi-objective algorithms.

Algorithm 2.2: Penalty calculation.

Input: $P(n)$, $\mathbf{f}(\cdot)$, $\mathbf{g}(\cdot)$, $\mathbf{h}(\cdot)$.
Output: $\Upsilon(n)$.

for $i = 1, \ldots, \mu$ **do**
  evaluate $\mathbf{f}(\mathbf{x}^{(i)})$, $\mathbf{g}(\mathbf{x}^{(i)})$, $\mathbf{h}(\mathbf{x}^{(i)})$;
end
if $\exists$ *equality constraints* **then**
  $\mathbf{f}(\mathbf{x}^{(i)}) \leftarrow \mathbf{f}(\mathbf{x}^{(i)}) + \gamma \sum_{j=1}^{q} \left| h_j(\mathbf{x}^{(i)}) \right|$;
end
if $\exists$ *inequality constraints* **then**
  for $k = 1, \ldots, m$ **do**
    $w_k \leftarrow \max f_k(\mathbf{x}^{(i)}), \quad \mathbf{x}^{(i)} \in \cap_{j=1}^{p} \mathcal{G}_j$;
  end
  if $\mathbf{x}^{(i)} \in \cap_{j=1}^{p} \mathcal{G}_j$ **then**
    $\mathbf{u} \leftarrow \mathbf{f}(\mathbf{x}^{(i)})$;
  else
    $\mathbf{u} \leftarrow \mathbf{w} + \gamma \sum_{j=1}^{p} \max \left[0, g_j(\mathbf{x}^{(i)})\right]$;
  end
end
return $\Upsilon(n) = \left\{\mathbf{u}(\mathbf{x}^{(1)}), \ldots, \mathbf{u}(\mathbf{x}^{(\mu)})\right\}$;

This penalty calculation scheme favours feasible individuals, considering inequality constraints only. All individuals that satisfy the inequality constraints are considered better than those that violate any of them. Since equality constraints are more restrictive than inequality ones, they are usually violated all the time, thus they are incorporated into the objective function values using the traditional penalty method (Bazaraa, Sherali and Shetty 1979). This kind of penalty calculation is based on the sophisticated tournament rules for constraint handling proposed in (Montes and Coello 2005).

The penalty calculation transforms the values in $\mathbf{f}(\cdot)$, $\mathbf{g}(\cdot)$, $\mathbf{h}(\cdot)$ into penalized values $\mathbf{u}(\cdot)$. The fitness function is a function that provides a scalar quality value

for the individuals of the population. The fitness function uses the penalized values calculated as in Algorithm 2.2, which become the objective function values if the problem is unconstrained.

In the mono-objective case, the values in $\Upsilon(n)$ are already scalar values, thus they are sorted in descending order, and ranked in such a way that the individual with rank one has the biggest value, and the one with rank μ has the smallest value. In the multi-objective case, an intermediate step is needed to scalarize and rank the population. We present some fitness scalarization methods later in this section. They allow the ranking of the population in descending order. After ranking the population, the rank values are then scaled as follows:

$$\bar{r}_i \leftarrow \frac{r_i - 1}{\mu - 1} \tag{2.19}$$

Finally, a nonlinear scaling is used to give the fitness values:

$$\Phi_i(n) \leftarrow \bar{r}_i^{\alpha} \tag{2.20}$$

in which $\bar{r}$ corresponds to the normalized value, $\alpha = 1$ gives linear scaling, while other values ($\alpha > 0$) give nonlinear scalings, as shown in Figure 2.2. Nonlinear scaling affects the selective pressure of genetic algorithms using stochastic selection. Scaling does not affect neither deterministic selection (as in evolution strategies) nor tournament selection.

2.4.2 Evolution Strategy

The first algorithm integrating the framework is based on evolution strategies, which appeared in the 1960's in Germany. The implementation described here follows the basic directives given in (Bäck 1996). The algorithm is presented in Algorithm 2.3.

The algorithm works with two distinct populations, $P(n)$, the parent population with size μ, and $Q(n)$, the offspring population with size λ. The selection operator is applied over $P(n)$ and $Q(n)$ to generate the next parent population $P(n+1)$. Albeit most evolution strategies work only with mutation, some recombination operators are available to the user.

Let $\mathbf{p}^{(i_1)}$ and $\mathbf{p}^{(i_2)} \in P(n)$, with $i_1, i_2 \in \mathcal{I} = \{1, \ldots, \mu\}$, be two individuals randomly

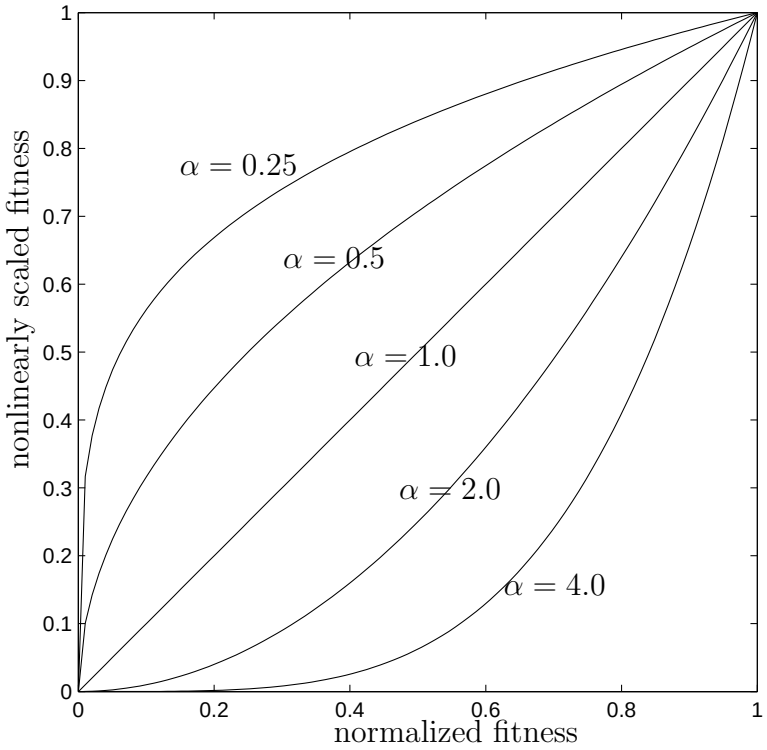

Figure 2.2: Nonlinear scaling for the fitness calculation.

selected from $P(n)$. The convex recombination generates a new individual $\mathbf{r}^{(j)}$, $j \in \mathcal{J} = \{1, \ldots, \lambda\}$ as follows:

$$\mathbf{r}^{(j)} = u\mathbf{p}^{(i_1)} + (1-u)\mathbf{p}^{(i_2)} \tag{2.21}$$

in which $u = U(0,1)$ is the sample of a random variable with uniform distribution within the interval $[0,1]$.

Three variants of the convex combination are presented in Algorithms 2.4, 2.5, 2.6. Figure 2.3 illustrates the differences among these three operators. These operators generate new solutions in the vicinity of the individuals in the population.

The mutation operator in ES is a Gaussian perturbation added to each individual. Each individual can have different standard deviations and different rotation angles for the covariance matrix, and these parameters can be adapted, see (Bäck 1996). Simpler implementations can use only one fixed standard deviation for all individuals. Here, we adopt one standard deviation for each variable with adaptation. The mutated individual

Algorithm 2.3: Evolution Strategy.

Data: population size μ, offspring size λ, search space $\mathcal{X}$, objective and constraint functions $f(\cdot)$, $\mathbf{g}(\cdot)$, $\mathbf{h}(\cdot)$.
Result: $A(n)$.

$P(n=0) = \{\mathbf{p}^{(1)}, \ldots, \mathbf{p}^{(\mu)}\} \leftarrow$ `Initialize population`$(\mu, \mathcal{X})$;
$A(n=0) = \varnothing \leftarrow$ `Initialize archive /* Stores the best solution set */`
while $\neg$ *stop criteria* **do**
  $R(n) \leftarrow$ `Recombination`$(P(n))$;
  $Q(n) \leftarrow$ `Mutation`$(R(n))$;
  $\Upsilon_P(n), \Upsilon_Q(n) \leftarrow$ `Penalty`$(P(n),Q(n),f(\cdot), \mathbf{g}(\cdot), \mathbf{h}(\cdot))$;
  $\Phi_P(n), \Phi_Q(n) \leftarrow$ `Fitness`$(\Upsilon_P(n),\Upsilon_Q(n))$;
  $S(n) \leftarrow$ `Selection`$(P(n), Q(n), \Phi_P(n), \Phi_Q(n))$;
  $P(n+1) \leftarrow S(n)$;
  $A(n+1) \leftarrow$ `Update`$(A(n),P(n))$;
  $n \leftarrow n+1$;
end

Algorithm 2.4: Convex recombination

$R(n) = \varnothing$;
for $j = 1, \ldots, \lambda$ **do**
  randomly select $i_1, i_2 \in \mathcal{I}$;
  $u \leftarrow U(0,1)$;
  $\mathbf{r}^{(j)} \leftarrow u\mathbf{p}^{(i_1)} + (1-u)\mathbf{p}^{(i_2)}$;
  put $\mathbf{r}^{(j)}$ in $R(n)$;
end
return $R(n)$;

Algorithm 2.5: Generalized convex recombination

$R(n) = \varnothing$;
for $j = 1, \ldots, \lambda$ **do**
  randomly select $i_1, i_2 \in \mathcal{I}$;
  for $k = 1, \ldots, d$ **do**
    $u \leftarrow U(0,1)$;
    $r_k^{(j)} \leftarrow up_k^{(i_1)} + (1-u)p_k^{(i_2)}$;
  end
  put $\mathbf{r}^{(j)}$ in $R(n)$;
end
return $R(n)$;

Algorithm 2.6: Panmitic recombination

```
R(n) = ∅;
for j = 1, ..., λ do
    randomly select i_1 ∈ I;
    for k = 1, ..., d do
        randomly select i_k ∈ I;
        u ← U(0, 1);
        r_k^(j) ← u p_k^(i_1) + (1 − u) p_k^(i_k);
    end
    put r^(j) in R(n);
end
return R(n);
```

is given by:

$$\mathbf{x} \leftarrow \mathbf{x} + \boldsymbol{\nu} \tag{2.22}$$

where $\boldsymbol{\nu}$ is a vector each component of which is $\nu_k = \sigma_k N(0,1)$. $N(0,1)$ gives a sample of a Gaussian random variable with zero mean and unitary standard deviation. The standard deviations σ_k are adapted with the modified 1/5-success rule, proposed in (Greenwood and Zhu 2001). Let ρ_s denote the percentage of successful mutations over the last $n_s > 10d$ trials. A successful mutation means that the offspring individual has a better fitness value when compared with its parent solution. This value can be monitored during the evolutionary process. The standard deviations are updated as follows:

$$\sigma_k(n+1) = \begin{cases} \min\left[\sigma_k(n)/c, u_k - l_k\right], & \text{if } \rho_s > 1/5 \\ \sigma_k(n), & \text{if } \rho_s = 1/5 \\ \sigma_k(n)c, & \text{if } 1/20 \leq \rho_s < 1/5 \\ \min\left[2\sigma_k(n), u_k - l_k\right], & \text{if } \rho_s < 1/20 \end{cases} \tag{2.23}$$

where $c = 0.85$. The idea behind this modified 1/5-success rule is that when the percentage of successful offspring is high, we can increase the mutation size; and when it is low, we should decrease the mutation size to refine the search. However, when the algorithm is stuck in a local minimum, the percentage of successful offspring drops significantly. When it is below 1/20, the standard deviation is doubled to broaden the search. Since the search space is limited, σ_k does not need to exceed $(u_k - l_k)$, the range for the kth variable, thus the $\min(\cdot, \cdot)$ function is used to avoid standard deviations of very high

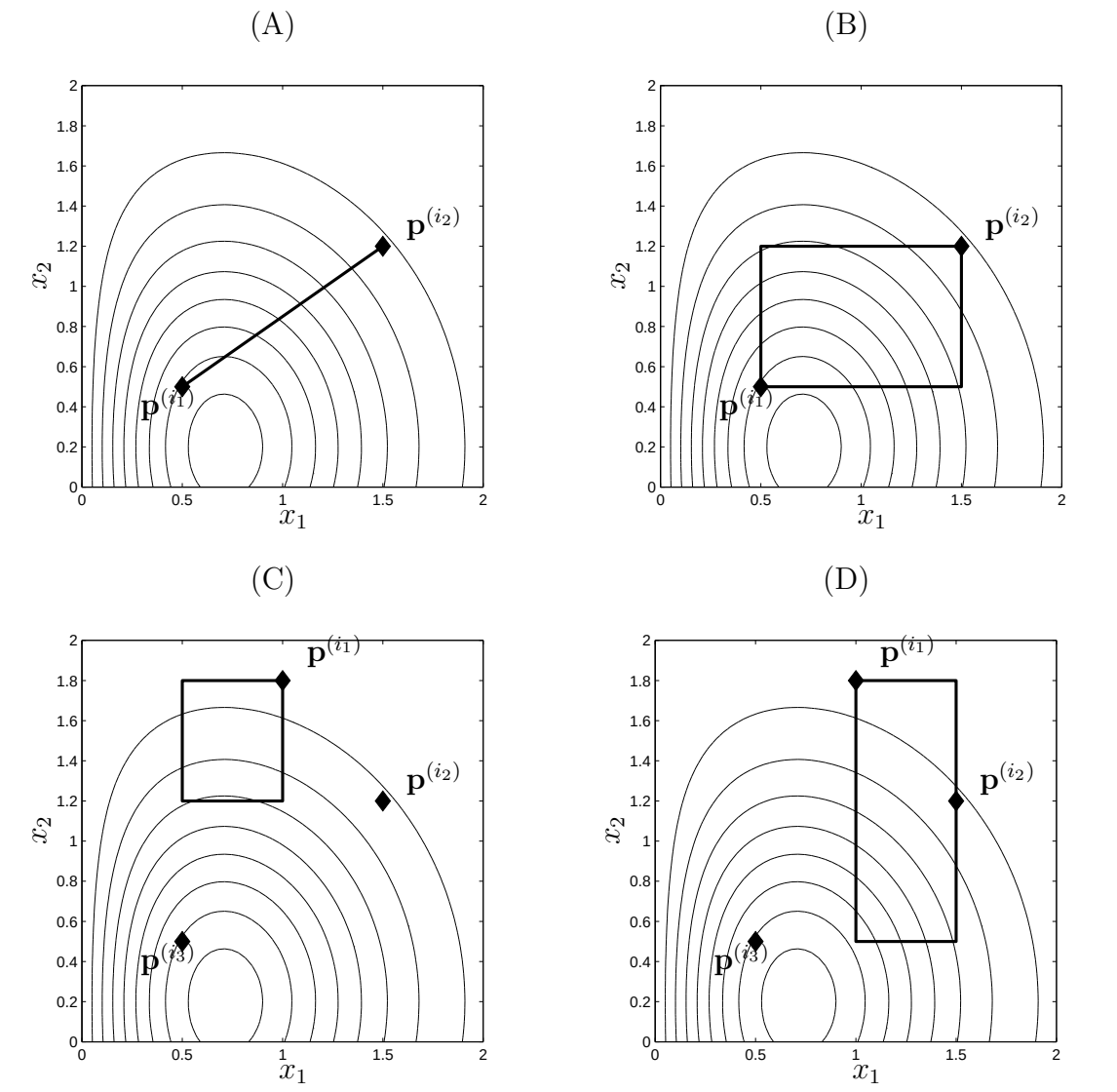

Figure 2.3: (A) In convex recombination, a new individual is generated along the line joining the two parent individuals; (B) in generalized convex recombination, a new individual is generated within the rectangle joining the parents; (C) panmitic recombination when the variable x_1 of $\mathbf{p}^{(i_1)}$ is combined with the variable x_1 of $\mathbf{p}^{(i_3)}$, and the variable x_2 of $\mathbf{p}^{(i_1)}$ is combined with the variable x_2 of $\mathbf{p}^{(i_2)}$; (D) panmitic recombination when the variable x_1 of $\mathbf{p}^{(i_1)}$ is combined with the variable x_1 of $\mathbf{p}^{(i_2)}$, and the variable x_2 of $\mathbf{p}^{(i_1)}$ is combined with the variable x_2 of $\mathbf{p}^{(i_3)}$.

magnitude. These heuristic adaptation rules aim at balancing global and local search.

Finally, we need to describe the selection mechanism. Evolution strategy algorithms

utilize two selection strategies, being both deterministic. They are briefly described in Algorithm 2.7.

Algorithm 2.7: ES($\mu + \lambda$) and ES(μ, λ) selection strategies

if *ES($\mu + \lambda$) selection* **then** `/* ES(`$\mu + \lambda$`) selection */`
  $P'(n) \leftarrow P(n) \cup Q(n)$;
  $\Phi'(n) \leftarrow \Phi_P(n) \cup \Phi_Q(n)$;
else `/* ES(`μ, λ`) selection */`
  $P'(n) \leftarrow Q(n)$;
  $\Phi'(n) \leftarrow \Phi_Q(n)$;
end
$S(n) \leftarrow$ select the μ best individuals from $P'(n)$ according to the fitness values in $\Phi'(n)$;

2.4.3 Evolutionary Programming

Another classic algorithm in evolutionary optimization is the evolutionary programming (Fogel 1995). It was initially developed to optimize finite state machines (FSM), using mutation operators for adding or deleting states and arcs in the FSM. Albeit originally applied to FSMs, it is easily extended to optimization with real parameters. The algorithm for the Evolutionary Programming is very simple. For each individual in the population, one new solution is produced by mutation. After that, the best μ individuals survive to the next generation, i.e., the selection mechanism is similar to ES($\mu + \mu$).

Algorithm 2.8: Evolutionary Programming

Data: population size μ, search space $\mathcal{X}$, objective and constraint functions $f(\cdot)$, $\mathbf{g}(\cdot)$, $\mathbf{h}(\cdot)$.
Result: $A(n)$.
$P(n=0) = \{\mathbf{p}^{(1)}, \ldots, \mathbf{p}^{(\mu)}\} \leftarrow$ `Initialize population(`μ, $\mathcal{X}$`)`;
$A(n=0) = \varnothing \leftarrow$ `Initialize archive /* Stores the best solution set */`
while $\neg$ *stop criteria* **do**
  $Q(n) \leftarrow$ `Mutation(`$P(n)$`)`;
  $\Upsilon_P(n), \Upsilon_Q(n) \leftarrow$ `Penalty(`$P(n)$,$Q(n)$,$f(\cdot)$, $\mathbf{g}(\cdot)$, $\mathbf{h}(\cdot)$`)`;
  $\Phi_P(n), \Phi_Q(n) \leftarrow$ `Fitness(`$\Upsilon_P(n)$,$\Upsilon_Q(n)$`)`;
  $S(n) \leftarrow$ `Selection(`$P(n), Q(n), \Phi_P(n), \Phi_Q(n)$`)` `/* ES(`$\mu + \mu$`) selection */`
  $P(n+1) \leftarrow S(n)$;
  $A(n+1) \leftarrow$ `Update(`$A(n)$,$P(n)$`)`;
  $n \leftarrow n + 1$;
end

There are some subtle differences between Evolutionary Programming and Evolution Strategies: (i) in Evolutionary Programming the offspring size is always equal to μ; (ii) in Evolution Strategies, each individual has an equal probability of generating an offspring, which means that each individual produces *on average* (λ/μ) offspring. If $\mu = \lambda$, each individual produces on average one offspring. In Evolutionary Programming conversely, each individual in $P(n)$ produces *always* one and only one offspring; (iii) The mutation operator in Evolutionary Programming is not adaptive in general, this idea is more commonly used in Evolution Strategies; (iv) The selection scheme in Evolutionary Programming is fixed and equivalent to ES$(\mu + \mu)$ selection of Evolution Strategy.

2.4.4 Genetic Algorithm

Traditionally, genetic algorithms employ binary coding to represent the candidate solutions of a given problem (Goldberg 1989). Nowadays, genetic algorithms also employ real coding, which is more interesting in optimization problems with real variables. This section describes the crossover, mutation and selection operators implemented in this work. It is possible to customize the genetic algorithm before using it. The general algorithm is presented in Algorithm 2.9.

Algorithm 2.9: Genetic Algorithm

Data: population size μ, search space $\mathcal{X}$, objective and constraint functions $f(\cdot)$, $\mathbf{g}(\cdot)$, $\mathbf{h}(\cdot)$.
Result: $A(n)$.

$P(n=0) = \{\mathbf{p}^{(1)}, \ldots, \mathbf{p}^{(\mu)}\} \leftarrow$ `Initialize population`$(\mu, \mathcal{X})$;
$A(n=0) = \varnothing \leftarrow$ `Initialize archive /* Stores the best solution set */`
while $\neg$ *stop criteria* **do**
  $\Upsilon(n) \leftarrow$ `Penalty`$(P(n), f(\cdot), \mathbf{g}(\cdot), \mathbf{h}(\cdot))$;
  $\Phi(n) \leftarrow$ `Fitness`$(\Upsilon(n))$;
  $S(n) \leftarrow$ `Selection`$(P(n), \Phi(n))$;
  $C(n) \leftarrow$ `Crossover`$(S(n))$;
  $M(n) \leftarrow$ `Mutation`$(C(n))$;
  $P(n+1) \leftarrow M(n)$;
  $A(n+1) \leftarrow$ `Update`$(A(n), P(n))$;
  $n \leftarrow n + 1$;
end

Before presenting the crossover operators, it is worth remarking on some subtle differences between recombination and crossover: in first generation genetic algorithms, the population obtained after selection is arbitrarily divided into two groups. Each indi-

vidual from the first group matches with another individual in the second group. Then, the crossover operator, which can be applied or not, depending on a random variable, generates two new individuals from each pair, which replace their parents. By contrast, in evolution strategies, one individual is produced from the recombination of two or more parents randomly selected from the parent population. The offspring size can be smaller or greater than the population size. Moreover, in an evolution strategy the substitution or not of the parents by their offspring is only determined by the selection step. The mutation has also some subtle differences: in evolution strategies, all individuals suffer mutation, while in genetic algorithms each individual is mutated only with probability ρ_m. The mutation parameters in evolution strategies are adapted along the optimization process. Although genetic algorithms can use adaptation of parameters, the usual versions adopt fixed crossover and mutation rates.

The simplest form of crossover with real coding is the convex crossover, in which the new individuals are generated along a line joining the two solutions selected for crossover. In order to extend the search of the operator, the generalized convex crossover can be used, generating a new individual in any point of a hyper-rectangle joining the original individuals. These two operators are presented in Algorithms 2.10 and 2.11. $\xi \geq 0$ is the extrapolation parameter, whose default value is zero.

Algorithm 2.10: Convex crossover

$C(n) = \varnothing$;
$\mathcal{I}_1 = \{1, \ldots, \mu/2\}$;
$\mathcal{I}_2 = \{\mu/2 + 1, \ldots, \mu\}$;
for $j = 1, \ldots, \mu/2$ **do**
  $i_1, i_2 \leftarrow j, j + \mu/2$;
  $u \leftarrow -\xi + (1 + 2\xi)U(0, 1)$;
  $\mathbf{c}^{(i_1)} \leftarrow u\mathbf{p}^{(i_1)} + (1 - u)\mathbf{p}^{(i_2)}$;
  $u \leftarrow -\xi + (1 + 2\xi)U(0, 1)$;
  $\mathbf{c}^{(i_2)} \leftarrow u\mathbf{p}^{(i_1)} + (1 - u)\mathbf{p}^{(i_2)}$;
  put $\mathbf{c}^{(i_1)}, \mathbf{c}^{(i_2)}$ in $C(n)$;
end
return $C(n)$;

Another interesting crossover for real coding is the real-biased crossover, analyzed in (Takahashi, Vasconcelos, Ramírez and Krahenbuhl 2003). Basically, one of the offspring is generated as in convex crossover, without bias. The other one has probability ρ_{pol} of being generated with bias, that is, with a higher probability of being closer to the best parent individual. Let the fitness value of $\mathbf{p}^{(i_2)}$ be higher than that of $\mathbf{p}^{(i_1)}$, i.e., $\mathbf{p}^{(i_2)}$ is

Algorithm 2.11: Generalized convex crossover

1 $C(n) = \varnothing$;
2 $\mathcal{I}_1 = \{1, \ldots, \mu/2\}$;
3 $\mathcal{I}_2 = \{\mu/2 + 1, \ldots, \mu\}$;
4 **for** $j = 1, \ldots, \mu/2$ **do**
5 $\quad i_1, i_2 \leftarrow j, j + \mu/2$;
6 $\quad$ **for** $k = 1, \ldots, d$ **do**
7 $\quad\quad u \leftarrow -\xi + (1 + 2\xi)U(0, 1)$;
8 $\quad\quad c_k^{(i_1)} \leftarrow up_k^{(i_1)} + (1 - u)p_k^{(i_2)}$;
9 $\quad$ **end**
10 $\quad$ **for** $k = 1, \ldots, d$ **do**
11 $\quad\quad u \leftarrow -\xi + (1 + 2\xi)U(0, 1)$;
12 $\quad\quad c_k^{(i_2)} \leftarrow up_k^{(i_1)} + (1 - u)p_k^{(i_2)}$;
13 $\quad$ **end**
14 $\quad$ put $\mathbf{c}^{(i_1)}, \mathbf{c}^{(i_2)}$ in $C(n)$;
15 **end**
16 **return** $C(n)$;

better. One offspring, say $\mathbf{c}^{(i_1)}$, is generated without bias. The other offspring $\mathbf{c}^{(i_2)}$ is generated with bias with probability ρ_{pol}. In order to obtain real-biased versions of the operators in Algorithms 2.10 and 2.11, we need to assure which one is better. After this, we need to change line 8 in Algorithm 2.10 and line 11 in Algorithm 2.11 to:

$$u \leftarrow -\xi + (1 + 2\xi)U(0, 1)U(0, 1)$$

also taking care of testing the condition $U(0, 1) \leq \rho_{pol}$.

In this way, $\mathbf{c}^{(i_2)}$ has higher chance of being closer to $\mathbf{p}^{(i_2)}$ than to $\mathbf{p}^{(i_1)}$. Figure 2.4 illustrates the effect of the real-biased convex crossover.

The simulated binary crossover was proposed in (Deb and Beyer 2001), being widely used today in real-coded genetic algorithms. Their basic idea was to create an operator with a behavior similar to that of the binary one-point crossover per variable. Using one-point crossover for the whole string, only one variable is actually changed, making orthogonal searches, i.e., the offspring is orthogonally distributed in the search space. Using one-point crossover per variable, it is possible to change all variables simultaneously, thus generating individuals over a volume centered in each parent individual. The simulated binary crossover aims at maintaining these characteristics for real-coded genetic algorithms. In addition, this operator presents some advantages over binary

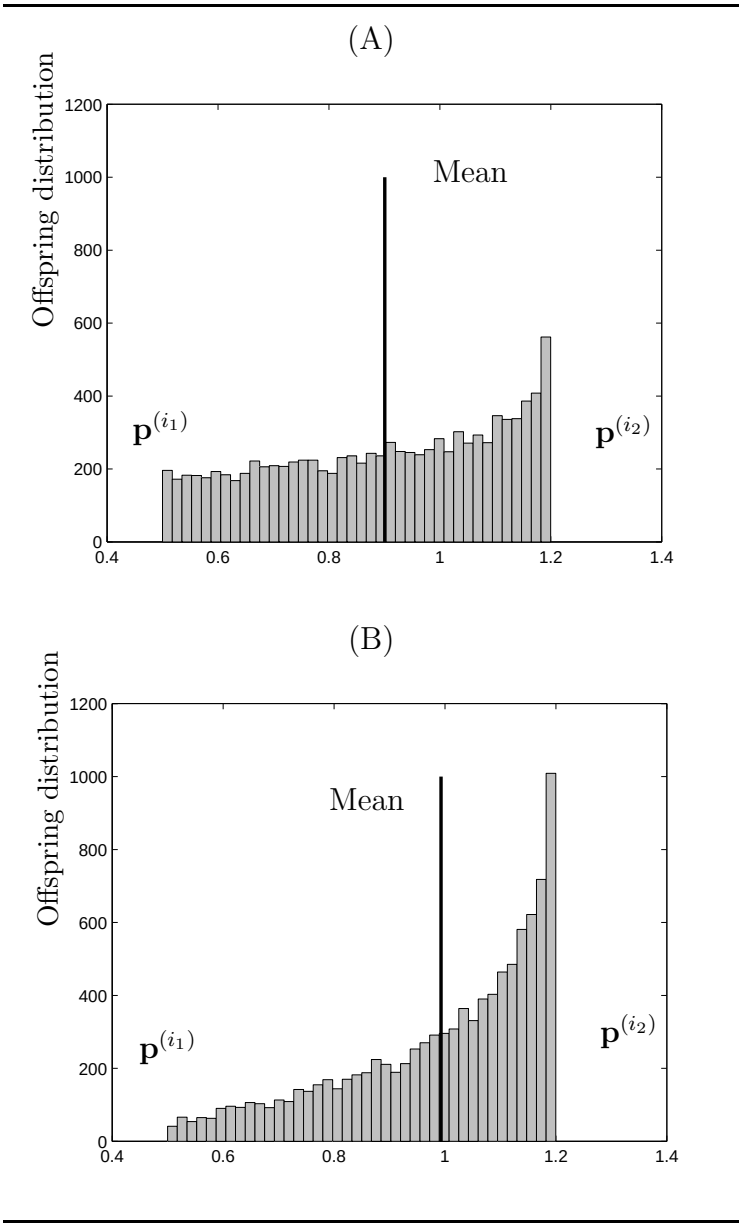

Figure 2.4: (A) Real-biased convex crossover with $\rho_{pol} = 0.3$ and (B) Real-biased convex crossover with $\rho_{pol} = 0.8$. The operator is applied 10,000 times with the same parents $\mathbf{p}^{(i_1)}$ and $\mathbf{p}^{(i_2)}$. The histogram of the children along the line joining the parents is shown. Increasing ρ_{pol}, the mean of the distribution gets closer to the best parent, which is $\mathbf{p}^{(i_2)}$ in this case.

operators, one of them is to provide some control over the local search.

The parameter $\eta > 0$ controls the distribution of new solutions. To illustrate the effect of this parameter, we apply the operator 10,000 times to the same pair of parent solutions for different values of η. The result is shown in Fig. 2.5. The default value is

Algorithm 2.12: Simulated binary crossover

$C(n) = \varnothing$;
$\mathcal{I}_1 = \{1, \ldots, \mu/2\}$;
$\mathcal{I}_2 = \{\mu/2 + 1, \ldots, \mu\}$;
for $j = 1, \ldots, \mu/2$ **do**
  $i_1, i_2 \leftarrow j, j + \mu/2$;
  for $k = 1, \ldots, d$ **do**
    $u_k \leftarrow U(0, 1)$;
    if $u_k \leq 0.5$ **then**
      $\beta_k \leftarrow (2u_k)^{1/(\eta+1)}$;
    else
      $[2(1 - u_k)]^{-1/(\eta+1)}$;
    end
    $c_k^{(i_1)} \leftarrow 0.5\left[(1 + \beta_k)p_k^{(i_1)} + (1 - \beta_k)p_k^{(i_2)}\right]$;
    $c_k^{(i_1)} \leftarrow 0.5\left[(1 - \beta_k)p_k^{(i_1)} + (1 + \beta_k)p_k^{(i_2)}\right]$;
  end
  put $\mathbf{c}^{(i_1)}, \mathbf{c}^{(i_2)}$ in $C(n)$;
end
return $C(n)$;

$\eta = 2$. Table 2.1 summarizes the parameters of each crossover operator.

Operator	Parameter	Default value
Convex	$0 < \xi < 1$	0
Generalized convex	$0 < \xi < 1$	0
Real-biased convex	$0 < \xi < 1; 0 < \rho_{pol} < 1$	0; 0.5
Real-biased gen. convex	$0 < \xi < 1; 0 < \rho_{pol} < 1$	0; 0.5
Simulated Binary	$\eta > 0$	2

Table 2.1: Summary of crossover operators and their parameters.

The next important step in genetic algorithms is the mutation, used to generate new solutions by perturbing some individuals in the population. This operator is responsible for the global search of the algorithm. For real-coded genetic algorithms, the mutation can be implemented as a perturbation vector added with probability ρ_m to each

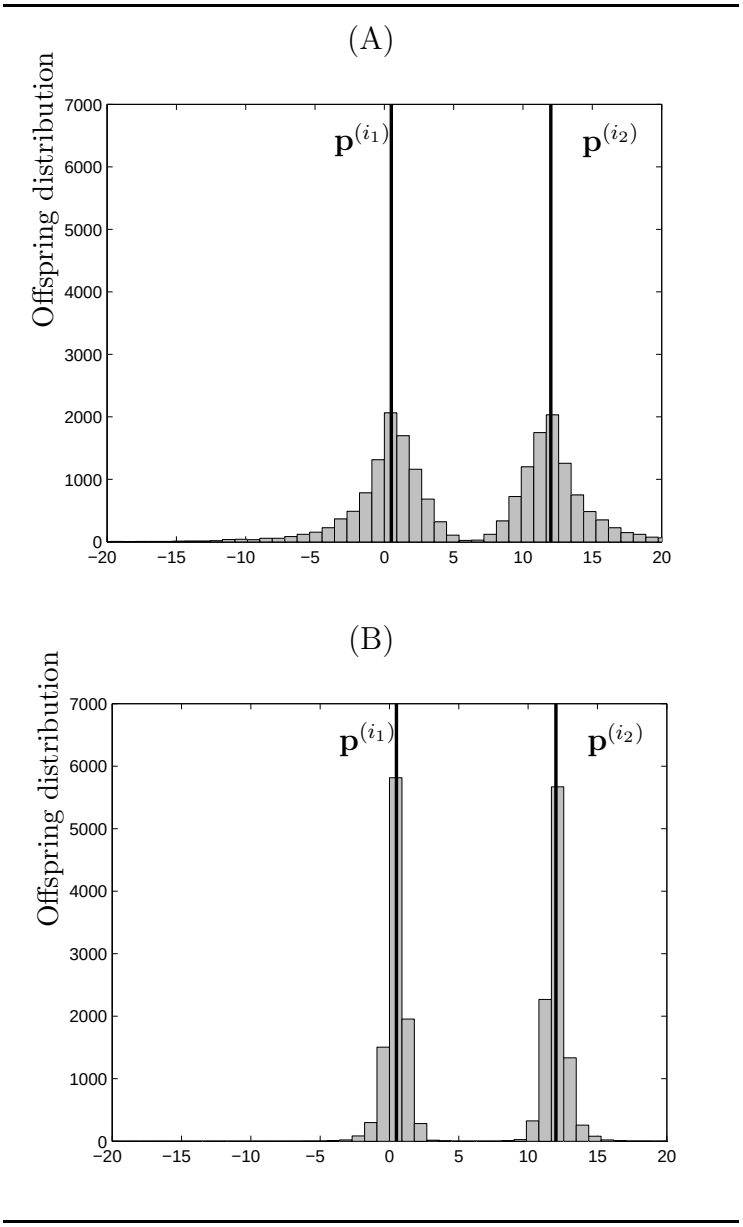

Figure 2.5: (A) Simulated binary crossover with $\eta = 2$ and (B) Simulated binary crossover with $\eta = 10$. The vertical lines show the positions of the parents $p^{(i_1)} = 0.5$ and $p^{(i_2)} = 12$. The operator is applied 10,000 times with the same parents and the histogram of the offspring is shown. Increasing η, the offspring has higher probability to be closer to their parents, characterizing a more intense local search.

individual in the population:

$$\mathbf{x} \leftarrow \mathbf{x} + \boldsymbol{\nu} \tag{2.24}$$

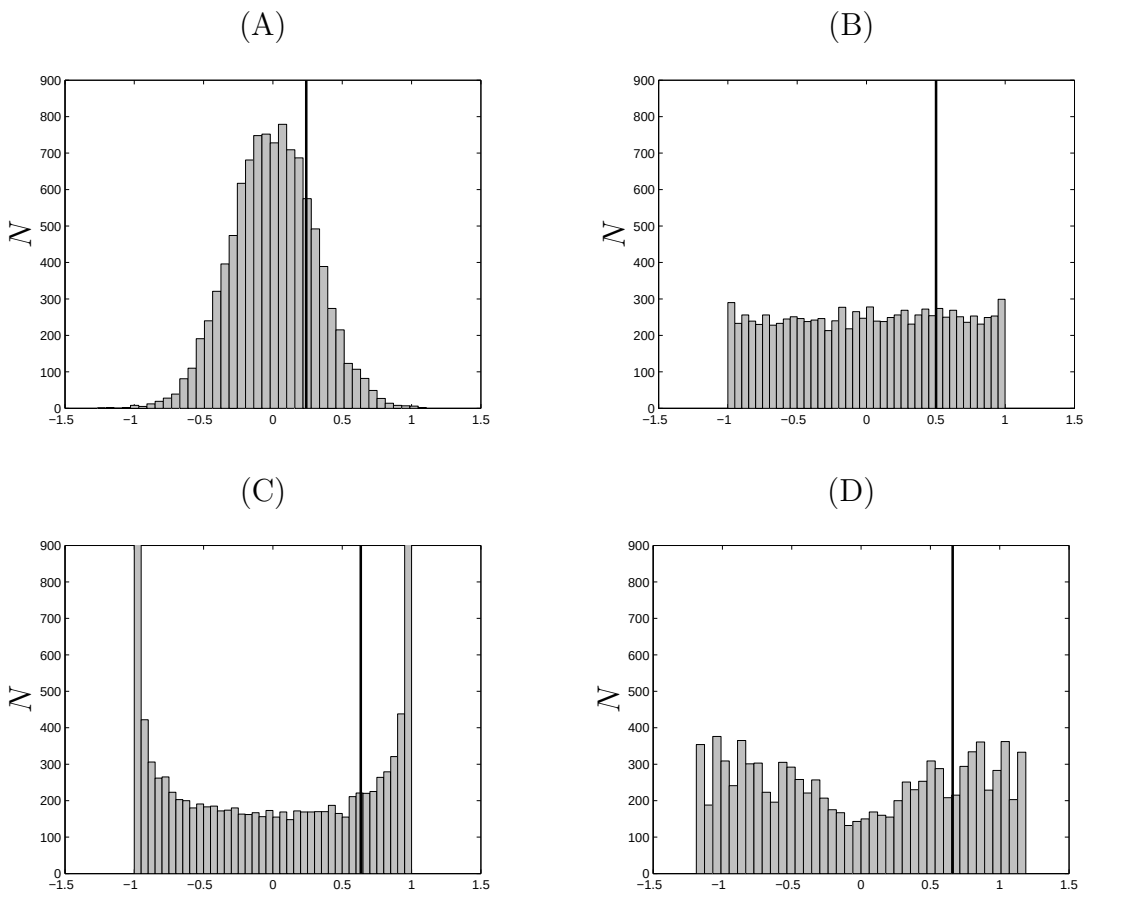

Figure 2.6: Histogram of the perturbations ν_k for each mutation operator: (A) Gaussian mutation, (B) uniform mutation, (C) chaotic mutation using logistic map, and (D) chaotic mutation using chaotic neuron. The vertical lines show the average magnitude of the perturbation for each distribution.

The perturbation vector can be characterized by different distributions. In Gaussian mutation, as in evolution strategies, the perturbation is:

$$\nu_k = \sigma(u_k - l_k)N(0,1) \tag{2.25}$$

where $\sigma > 0$ is a parameter whose default value is 0.2.

In uniform mutation we have:

$$\nu_k = \sigma(u_k - l_k)\left[2U(0,1) - 1\right] \tag{2.26}$$

and in chaotic mutation, we have:

$$\nu_k = \sigma(u_k - l_k)C(\pi) \tag{2.27}$$

in which $C(\pi)$ is a probability distribution obtained by a chaotic dynamic system with parameters π. A chaotic dynamic can be obtained with a nonlinear system as the logistic map:

$$z(n+1) = \pi z(n)\left[1 - z(n)\right] \tag{2.28}$$

Another system that presents chaotic behavior is given by the transfer function of a chaotic neuron:

$$z(n+1) = \pi_1 z(n) - 2\tanh\left[\pi_2 z(n)\right]\exp\left[-3z(n)^2\right] \tag{2.29}$$

Using for instance $\pi = 4$, the iterative application of the logistic map generates a distribution of values within the interval $[-1, 1]$. For the chaotic neuron, using $\pi_1 = 0.9$ and $\pi_2 = 5$, we get a distribution of values within the interval $[-1.2, 1.2]$, see Figure 2.6.

As we can observe in Figure 2.6, the chaotic mutation provides higher probabilities in the extrema and lower probabilities in the center. The converse is observed for the Gaussian mutation. In this way, there is more chance of generating new solutions farther from the original point. In addition, by calculating the mean of the absolute values we can find the average step size for each distribution, shown as vertical lines in Figure 2.6. The average step size is greater for the chaotic distributions.

An algorithm to generate the chaotic perturbation ν_k is presented in Algorithm 2.13. In this algorithm, $z(n) = \zeta(z(n-1), \pi)$ represents the nonlinear mapping with parameters π.

Finally, we need to describe the selection operators available. It is possible to choose between roulette wheel and tournament selection. These two selection methods are popular and well studied, and we indicate references (Goldberg 1989, Gen and Cheng 1997, Mitchell 1998) for more details. Since there are different forms of tournament selection in literature, we briefly describe the method used here:

- one competitor is randomly selected from the population until T competitors are selected for tournament;

Algorithm 2.13: Chaotic mutation

$z(0) \leftarrow U(0,1)$;
for $n = 1, \ldots, 100$ **do**
  $z(n) \leftarrow \zeta(z(n-1), \pi)$;
end
for $k = 1, \ldots, d$ **do**
  randomly select $i_k \in [1, 100]$;
  $\nu_k \leftarrow \sigma(u_k - l_k)z(i_k)$;
  $x_k \leftarrow x_k + \nu_k$;
end

- the one with higher fitness value wins the tournament and is selected. Ties are broken with random selection.

Although the usual tournament selection is without repetition, we adopt repetition just to guarantee the global convergence proof presented in Chapter 4.

The most common form of tournament is binary tournament, in which $T = 2$. This is the default value, if no value is defined by the user. The higher the value of T the higher the selective pressure, since the winner has to be better than many individuals in the population.

2.4.5 Differential Evolution

The last Darwinian evolutionary algorithm to be discussed before we advance to multi-objective problems is the Differential Evolution Algorithm. Differential Evolution appeared in the mid 1990s but has become more popular after the publication of (Storn and Price 1997) in the end of 1997, showing very good results on a variety of benchmark problems.

The main novelty in differential evolution is the scheme for generating new solutions from the current ones. In addition, the algorithm is very simple and has few parameters to adjust.

The search capability of the algorithm relies on the differential mutation, whereby a mutant vector is generated from three vectors in the population. The mutant vector is

Algorithm 2.14: Differential Evolution

Data: population size μ, search space $\mathcal{X}$, objective and constraint functions $f(\cdot)$, $\mathbf{g}(\cdot)$, $\mathbf{h}(\cdot)$.
Result: $A(n)$.

$P(n=0) = \{\mathbf{p}^{(1)}, \ldots, \mathbf{p}^{(\mu)}\} \leftarrow$ `Initialize population`$(\mu, \mathcal{X})$;
$A(n=0) = \varnothing \leftarrow$ `Initialize archive /* Stores the best solution set */`
while $\neg$ *stop criteria* **do**
  for $i = 1, \ldots, \mu$ **do**
    $\mathbf{v}_i \leftarrow$ `Generate Mutant Vector`();
    $\mathbf{q}^{(i)} \leftarrow$ `Crossover`$(\mathbf{p}^{(i)}, \mathbf{v}_i)$;
    Put $\mathbf{q}^{(i)}$ in $Q(n)$;
  end
  $\Upsilon_P(n), \Upsilon_Q(n) \leftarrow$ `Penalty`$(P(n), Q(n), f(\cdot), \mathbf{g}(\cdot), \mathbf{h}(\cdot))$;
  $\Phi_P(n), \Phi_Q(n) \leftarrow$ `Fitness`$(\Upsilon_P(n), \Upsilon_Q(n))$;
  $S(n) \leftarrow$ `Selection`$(P(n), Q(n), \Phi_P(n), \Phi_Q(n))$;
  $P(n+1) \leftarrow S(n)$;
  $A(n+1) \leftarrow$ `Update`$(A(n), P(n))$;
  $n \leftarrow n + 1$;
end

given by:

$$\mathbf{v}_i = \mathbf{x}_{r_1} + w(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}) \tag{2.30}$$

in which $r_1, r_2, r_3 \in \{1, \ldots, \mu\}$ with $r_1 \neq r_2 \neq r_3 \neq i$ and $0 < w \leq 2$. Recommended values in the literature are $w = 0.5$ and $w = 1.0$. This operation is illustrated in Figure 2.7.

Finally, the offspring of each individual is given by the recombination of its mutant vector and itself. Any of the real-coding recombination/crossover operators described before can be adopted with the Differential Evolution. In some variations of the algorithm, $\mathbf{x}_{r_1}$ is substituted for the best solution achieved. In this case, all mutant vectors can be seen as random perturbations around the best solution. Mutation operators can be added, especially in order to guarantee global convergence.

The selection mechanism is also very simple. The offspring substitutes the original solution only if it is better, otherwise, the original solution survives to the next generation. It is not exactly an ES($\mu + \mu$) selection, but instead an ES($1 + 1$) for each individual.

Further information on this class of algorithms can be found in (Price, Storn and

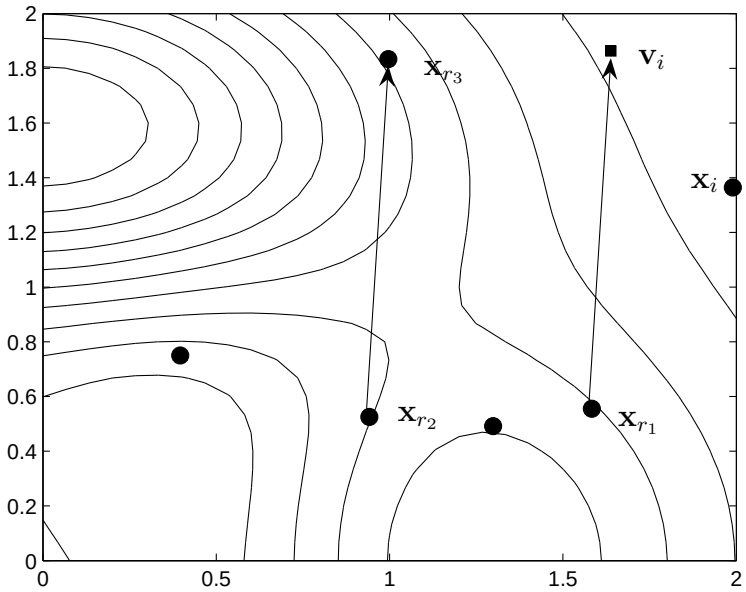

Figure 2.7: Illustration of the mutant vector in differential evolution for $w = 1.0$. The mutant vector $\mathbf{v}_i$ is obtained by perturbing a randomly selected individual ($\mathbf{x}_{r_1}$) in the population in the direction defined by two other individuals ($\mathbf{x}_{r_2}$ and $\mathbf{x}_{r_3}$). The offspring of $\mathbf{x}_i$ is given by the recombination of its mutant vector and itself.

Lampinen 2005), a book entirely dedicated to differential evolution.

2.4.6 Multi-objective problems

Genetic algorithms are characterized by crossover and mutation operators, and a probabilistic selection mechanism that favours the best solutions in the population. In the multi-objective context, the immediate problem to be solved is to define a scalar quality value for the individuals, i.e., the fitness value assessment. Once the fitness values are calculated, there is no difference in the selection, crossover and mutation operators. For this reason, the operators described before are still valid for multi-objective evolutionary algorithms.

Therefore, in the next sections we present some popular algorithms available in the literature, focusing on the fitness assignment method and the general structure of the algorithm. Although the algorithms reviewed in the literature guide the presentation given

here, we remark that with the various fitness assignment methods and genetic operators, a wide range of evolutionary algorithms are directly available. For a literature overview of multi-objective evolutionary algorithms, we recommend the books (Deb 2001, Coello, Veldhuizen and Lamont 2002) and the article (Coello 2006). The generic outline of a multi-objective genetic algorithm is similar to Algorithm 2.1. Many algorithms in the literature can be considered as particular instances of this general algorithm, with specific characteristics.

As already discussed, the archive population has a maximum size. For this reason, some techniques for archive reduction and diversity preservation should be implemented in the update function, making it more complicated than in the mono-objective case. Niching techiques are an example of diversity preservation in the archive population. The objective is to maximize the distribution of estimates of $\mathcal{Y}^*$.

2.4.7 Vector Evaluated Genetic Algorithm - VEGA

The first genetic algorithm for multi-objective problems was proposed by Schaffer, in the 1980's (Schaffer 1984). It is recognized as the first trial towards the utilization of genetic algorithms for multi-objective problems. VEGA is very simple and cited here just for historical reasons. Since there are more sophisticated algorithms available, there is no benefit in adopting VEGA in practice.

Basically, the fitness assignment scheme consists of dividing the population with μ individuals into m sub-populations, where m is the number of objectives. Individuals in each sub-population are evaluated considering only one objective, that is, individuals in the ith sub-population are evaluated using the ith objective function. After that, roulette wheel selection is applied together with the other genetic operators. VEGA's implementation is very simple, however its fitness assignment method tends to concetrate solutions on the extrema of the Pareto front, an effect termed as specialization. In the long term, individuals that are good for a given objective, and hence bad for the others, survive. The central part of the Pareto front, which corresponds to solutions with intermediate performance with respect to all objectives, is then badly represented.

2.4.8 Multiobjective Genetic Algorithm - MOGA

During the 1990's decade, many evolutionary techniques for multi-objective optimization have appeared. This period corresponds to the first generation of multi-objective evolutionary algorithms.

In 1993, Fonseca and Fleming proposed MOGA (Fonseca and Fleming 1993, Fonseca and Fleming 1998a, Fonseca and Fleming 1998b), which is the first algorithm basing the ranking of individuals on Pareto dominance relations. This criterion had been suggested in (Goldberg 1989), later implemented by Fonseca and Fleming. The classification given to an individual $\mathbf{p}^{(i)}$ is:

$$r(\mathbf{p}^{(i)}) = 1 + d_i \tag{2.31}$$

where d_i is the number of individuals in the population that dominate $\mathbf{p}^{(i)}$. Observe that nondominated individuals are given the value $r = 1$. The more dominated the individual is, the higher its ranking value.

After classifying the population, we need to assign fitness values to each individual according to its rank. Usually, linear scaling is used. The fitness calculation can employ niching techniques, in order to improve the distribution of solutions over the Pareto front. Niching techniques were used first to map sub-optimal solutions, but they showed to be useful for multi-objective genetic algorithms also (Horn, Nafpliotis and Goldberg 1994). The fitness value is degraded as follows:

$$\phi(\mathbf{p}^{(i)}) \leftarrow \frac{\phi(\mathbf{p}^{(i)})}{\mathrm{sh}_i} \tag{2.32}$$

where sh_i is an estimate of the density in the vicinity of $\mathbf{p}^{(i)}$. The sharing function sh_i is given by:

$$\mathrm{sh}_i = \sum_{j=1}^{N} s(d_{ij}) \tag{2.33}$$

with:

$$s(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma}\right)^{\alpha}, & d_{ij} \leq \sigma_{sh} \\ 0, & d_{ij} > \sigma_{sh} \end{cases} \tag{2.34}$$

The parameter σ_{sh} is the niche radius and controls the spread of individuals in the population. In multi-objective problems, the niche radius is defined in the objective space, thus d_{ij} is the Euclidian distance between two solutions in the objective space. Individuals in a dense region have their fitness values penalized by the sharing function. Usually, $\alpha = 1$, which gives the triangle sharing function. Nevertheless, the niche radius is an additional parameter for the user, and it is usually problem dependent. In the MOGA version implemented here, this parameter is automatically defined. The distances d_{ij} are calculated in a normalized objective space, that is, each objective is normalized to the interval $[0, 1]$. The biggest distance in this hypercube is $\sqrt{m}$, which is the biggest diagonal in the hypercube. It is possible to estimate the niche radius as:

$$\sigma_{sh} = \frac{\sqrt{m}}{\xi} \tag{2.35}$$

where ξ is the maximum size of $A(n)$.

Although the original MOGA does not use an offline population, the MOGA implemented here follows the scheme in Algorithm 2.1, adopting an offline population. The update function stores the nondominated solutions $P(n) \cup A(n)$. If the number of solutions in $A(n)$ exceeds the maximum number allowed in $A(n)$, then a reduction technique is used. This reduction technique is based on the solution counting in each niche, as presented in Algorithm 2.15.

Algorithm 2.15: Reduction by niche counting

```
1  s(A) ← size of A(n);
2  σ_sh ← √m/ξ;
3  while s(A) > ξ do
4  |   normalize the objective space;
5  |   calculate the distance matrix d_ij for solutions in A(n);
6  |   for i = 1, ..., s(A) do
7  |   |   c(i) ← number of times in which d_ij < σ_sh, j = 1, ..., s(A);
8  |   end
9  |   remove the individual a^(i) ∈ A(t) with the highest c(i);
10 |   s(A) ← size of A(n);
11 end
```

The implemented MOGA can employ all crossover and mutation operators presented before as well as roulette and tournament selection.

2.4.9 Niched Pareto Genetic Algorithm - NPGA

NPGA was originally proposed in (Horn, Nafpliotis and Goldberg 1994). The original algorithm employs tournament selection based on Pareto dominance. Two individuals were selected from the population and compared against a reference group, which is a fraction of the population selected at random. The one that dominated more individuals in the reference group was the winner. Any other situation was a tie, which was broken using the niche counting technique. NPGA eliminates the need for assigning scalar fitness values, making it computationally simple.

As well as MOGA and other algorithms of the first generation, the traditional implementation of NPGA does not use an archive population. We implemented an NPGA version that uses an archive population. Actually, the MOGA with tournament selection has a similar effect to NPGA.

2.4.10 Nondominated Sorting Genetic Algorithm - NSGA

Still during the 1990's decade, another interesting fitness assignment method appeared (Srinivas and Deb 1994): the nondominated sorting technique. The idea behind nondominated sorting is that the population can be sorted based on layers of nondominated fronts in the objective space. The classical NSGA adopts fitness sharing to degrade fitness values of individuals in the same Pareto front and very close to each other.

However, the classical NSGA had some practical problems, the main issue was the computational complexity of the original nondominated sorting algorithm, which is $O\left(m\mu^3\right)$. Additionally, the algorithm was not an elitist algorithm and needed the definition of the niche parameter σ_{sh}. For these reasons, in the beginning of 2000 the authors of NSGA proposed an improved version called NSGA2 (Deb, Agarwal, Pratab and Meyarivan 2000, Deb, Pratap, Agarwal and Meyarivan 2002). The main improvements in NSGA2 were the creation of a fast nondominated sorting algorithm, with computational complexity of $O\left(m\mu^2\right)$, lower than the original nondominated sorting technique, the adoption of elitism and the elimination of the niche parameter, through the use of another strategy for measuring the spread of solutions in the objective space.

The implementation discussed in this section is based on NSGA2, following the guidelines in (Deb, Agarwal, Pratab and Meyarivan 2000). NSGA2 maintains simultaneously the parent population $P(n)$ and the offspring population $Q(n)$. When generating

$P(n+1)$, the selection is applied to both $P(n)$ and $Q(n)$, like an ES$(\mu+\mu)$ approach. This is a way of implicit ellitism.

Algorithm 2.16: NSGA2

Data: population size μ, archive size ξ, search space $\mathcal{X}$, objective and constraint functions $f(\cdot)$, $\mathbf{g}(\cdot)$, $\mathbf{h}(\cdot)$.
Result: $A(n)$.

$P(n=0) = \{p^{(1)}, \ldots, p^{(\mu)}\} \leftarrow$ `Initialize population`$(\mu, \mathcal{X})$;
$A(n=0) = \varnothing \leftarrow$ `Initialize archive /* Stores the best solution set */`
$\Upsilon_P(n) \leftarrow$ `Penalty`$(P(n), \mathbf{f}(\cdot), \mathbf{g}(\cdot), \mathbf{h}(\cdot))$;
$\Phi_P(n) \leftarrow$ `Fitness`$(\Upsilon_P(n))$;
$Q(n) \leftarrow$ `Reproduction`$(P(n), \Phi_P(n))$;
while $\neg$ *stop criteria* **do**
  $\Phi_Q(n), \Phi_A(n) \leftarrow$ `Fitness`$(Q(n), A(n))$;
  $P(n+1), \Phi_P(n+1) \leftarrow$ `Reduction`$(P(n) \cup Q(n), \Phi_P(n) \cup \Phi_Q(n))$;
  $Q(n+1) \leftarrow$ `Reproduction`$(P(n+1), \Phi_P(n+1))$;
  $A(n+1) \leftarrow$ `Update`$(A(n), P(n) \cup Q(n))$;
  $n \leftarrow n+1$;
end

The reproduction routine consists of applying the selection, crossover and mutation operators in this sequence to generate an offspring population $Q(n)$ with μ individuals, the same size as $P(n)$. The fitness assignment method is based on the fast nondominated algorithm, see Algorithm 2.15, and the nonlinear ranking described before.

Algorithm 2.17: Fast nondominated sorting technique

$s(P) \leftarrow$ size of $P(t)$;
for $i = 1, \ldots, s(P)$ **do**
  $c(i) \leftarrow$ number of individuals in $P(n)$ that dominate $\mathbf{p}^{(i)}$;
  $L(i) \leftarrow$ set of solutions in $P(n)$ that $\mathbf{p}^{(i)}$ dominates;
end
$k \leftarrow 1$;
while $\neg$ *all individuals were classified* **do**
  $\mathcal{I} = \{i : c(i) = 0\}$;
  put nondominated solutions $\mathbf{p}^{(\mathcal{I})}$ in front $\mathcal{P}_k$;
  $\mathcal{J} = \{j : \mathbf{p}^{(j)} \in L(\mathcal{I})\}$;
  $c(\mathcal{J}) \leftarrow c(\mathcal{J}) - 1$;
  $k \leftarrow k+1$;
end

The result of the fast nondominated sorting is that each individual has a number associated with it that indicates to which front it belongs. These values are transformed

into fitness values using ranking and nonlinear scaling. The fitness values are available for the selection operator. The reduction routine creates a population $P(n+1)$ of size μ from the combined populations $P(n) \cup Q(n)$ of size 2μ. This routine adds individuals from the first fronts until the size of $P(n+1)$ is greater than μ. At this moment, a density measure in the objective space is used to eliminate individuals of the last added front until the size of $P(n+1)$ is equal to μ (Deb, Agarwal, Pratab and Meyarivan 2000). Finally, the update function routine is different from MOGA. In our implementation, see Algorithm 2.16, we use a strategy similar to that used in SPEA2, a multi-objective algorithm to be discussed further on.

Algorithm 2.18: Reduction by k-neighbour rule

```
s(A) ← size of A(n);
while s(A) > ξ do
    normalize the objective space;
    calculate the distance matrix d_ij for solutions in A(n);
    k ← round(√s(A));
    for i = 1, ..., s(A) do
        c(i) ← sum of the smallest k distances d_ij, j = 1, ..., s(A);
    end
    remove the individual a^(i) ∈ A(n) with the smallest c(i);
    s(A) ← size of A(n);
end
```

The reduction by k-neighbour rule calculates the normalized distance in the objective space from each individual to its k closest neighbours. That one with the smallest sum of these distances is within a densely populated region of the objective space. This individual is then eliminated from the archive.

2.4.11 Strength Pareto Evolutionary Algorithm - SPEA

SPEA is another famous multi-objective genetic algorithm in literature. It was proposed in the end of the 1990's (Zitzler and Thiele 1999) and is recognized as the first evolutionary algorithm employing an archive population with the best solutions (Coello 2006). SPEA can be seen as a milestone for the second generation of multi-objective evolutionary algorithms. Later, the authors identified some significant limitations in the first version, which motivated the development of a more sophisticated version, called SPEA2 (Zitzler, Laumanns and Thiele 2001). The version implemented in the evolutionary framework corresponds to SPEA2.

The general outline of SPEA2 is similar to Algorithm 2.1, with two important differences: First, the archive population has a fixed size, i.e. the size of $A(n)$ is equal to ξ for all n. Until now, the size of $A(n)$ was less than or equal to ξ. In SPEA2, the archive update function stores the nondominated solutions in $A(n) \cup P(n)$ into $A(n+1)$. If the size of $A(n+1)$ is less than ξ, the best solutions in $A(n) \cup P(n)$ according to their fitness values are stored into $A(n+1)$ until its size is equal to ξ. If the size of $A(n+1)$ is greater than ξ, the reduction by the k-neighbour rule is used. The second difference is that the selection operator acts only over individuals in $A(n)$. This is why the archive is completed with some nondominated solutions, in order to maintain some diverse alternatives for the selection operator.

Now we need only to detail the fitness assignment method in SPEA2. Let $P'(n) = P(n) \cup A(n)$ with size $\mu + \xi$. Each individual $\mathbf{p}^{(i)} \in P'(t)$ is given a strength value:

$$s(i) = \left|\{j : \mathbf{p}^{(j)} \in P'(n), \mathbf{p}^{(i)} \prec \mathbf{p}^{(j)}\}\right| \tag{2.36}$$

where $|\cdot|$ means the cardinality of the set in its argument.

The raw fitness is given by:

$$\varphi(i) = \sum_{\substack{\mathbf{p}^{(j)} \in P'(t) \\ \mathbf{p}^{(i)} \succ \mathbf{p}^{(j)}}} s(j) \tag{2.37}$$

putting it in words, the raw fitness of a given individual corresponds to the sum of the strength values of all solutions that dominate it. If the individual is not dominated by any other solution, $\varphi(i) = 0$. Thus, higher values for $\varphi(i)$ mean that the individual is dominated by many individuals which in turn dominate many other individuals.

To measure density, SPEA2 uses the following density value:

$$d(i) = \frac{1}{\sigma_i^k + 2} \tag{2.38}$$

where σ_i^k corresponds to the distance in the normalized objective space between $\mathbf{p}^{(i)}$ and its kth closest neighbour, with $k = \sqrt{\mu + \xi}$.

Finally, the fitness is:

$$\phi(\mathbf{p}^{(i)}) = \varphi(i) + d(i) \tag{2.39}$$

Next, these values can be ranked and scaled as described before.

2.5 Other evolutionary algorithms

In the previous section, we described real-coding implementations of some popular algorithms for global optimization. However, many other evolutionary techniques are available in the literature. In this section we briefly comment on other approaches, providing some references for additional information.

Another popular probabilistic optimization technique is the simulated annealing method (Kirkpatrick, Gelatt and Vecchi 1983). It is based on the annealing technique in metallurgy, a technique involving heating and controlled cooling of a given material to increase the size of its crystals and improve their quality. By analogy with this physical process, the simulated annealing algorithm works with one solution, which is randomly perturbed with a probability that depends on a temperature parameter. When the temperature is high, the random perturbations are big and can cover the entire search space. When the temperature is low, the random perturbations are small, resembling local search. Under some considerations, simulated annealing can be seen as ES$(1 + 1)$ or ES$(1 + \mu)$, with a specific annealing-based adaptation rule for the mutation. Today, there is a family of simulated annealing algorithms, like fast simulated annealing, chaotic simulated annealing, and others, used in a wide range of applications.

Particle swarm optimization - PSO - methods are based on the swarm intelligence concept (Engelbrecht 2005), which models the complex social behavior arising from the cooperation of simple particles. The first particle swarm optimization appeared in (Kennedy and Eberhart 1995). Since then, the PSO approach has been successfully applied in many different areas. It is also a population-based method, in which solutions are represented by particles in the search space $\mathcal{X}$. Individuals in the swarm are represented by their position (a point in the search space) and velocity. Their movement in the search space is influenced by other individuals in the swarm, but with some degree of independent exploration, which represents the exploration and exploitation trade-off in PSO. The velocity of each particle is updated according to its best visited point (cognitive component) and to the best point visited by the swarm (social component). Some variants are available for improving global search capability (Esquivel and Coello 2003) and constraint handling (Parsopoulos and Vrahatis 2002, Zielinski and Laur 2006).

Another interesting class of evolutionary algorithms nowadays is the class of immune algorithms. Optimization algorithms inspired by the immune system first appeared in the end of the 1990's decade (Chun, Kim, Jung and Hong 1997), but they were very similar to genetic algorithms. Only in the beginning of this decade was an immune algorithm with significant differences with respect to genetic algorithms proposed (de Castro and Von Zuben 2002), motivated by the clonal selection theory. Many immunology theories are used in artificial immune systems, but the clonal selection theory is by far the most used in designing optimization methods. Other immune algorithms for mono-objective optimization can be found in (Campelo, Guimarães, Igarashi and Ramírez 2005, Campelo, Guimarães, Igarashi, Ramírez and Noguchi 2006). Although immune algorithms are not inspired by the evolution theory, they can be considered as an evolutionary algorithm, given that the clonal selection principle models a micro-evolutionary process in the scope of the adaptive immune system.

Regarding multi-objective optimization algorithms, the literature is also very rich. The algorithms presented in this section are amongst the most famous algorithms and are recognized as the state of the art in multi-objective evolutionary optimization. Of course, it is not possible to state that they are the best ones, but they form a standard reference group of algorithms that everybody tries to defeat. Papers proposing new methods or new variants usually compare their results to those obtained by using these algorithms.

The Pareto Envelope-based Selection Algorithm - PESA - was proposed in (Corne, Knowles and Oates 2000). PESA is most recognized by its interesting grid-based approach for storing nondominated solutions, which can maintain a good diversity in the objective space. Moreover, PESA is very simple and the authors report satisfactory results in (Corne, Knowles and Oates 2000). It is worth mentioning that the success of PESA in many test problems when compared to the first SPEA also motivated the development of SPEA2.

The Pareto Archived Evolution Strategy - PAES - is a simple algorithm based on the ES$(1+1)$ (Knowles and Corne 1999, Knowles and Corne 2000a). Basically, one solution is randomly generated in the search space $\mathcal{X}$ and copied to the archive $A(n)$. Next, this individual is mutated and the archive is updated considering this new solution. The individual of the next generation is randomly selected from $A(n)$. The grid-based approach for diversity control used in PESA is also used in PAES. Observe that PAES can be seen as a local search mechanism in the archive population $A(n)$. Later, the authors explored this characteristic by proposing M-PAES, a multi-objective memetic

algorithm that consists of a genetic algorithm coupled with PAES as the local search tool (Knowles and Corne 2000b).

It is possible to find many other multi-objective techniques in the literature, based on a variety of paradigms. We remark simulated annealing methods (Czyzak and Jaszkiewicz 1998, Nam and Park 2000, Ho, Yang, Wong and Ni 2003), particle swarm optimization (Zhang, Zhou, Liu, Ma, Ma and Liang 2003, Baumgartner, Magele and Renhart 2004, Coello, Pulido and Lechuga 2004), differential evolution algorithms (Madavan 2002), evolution strategies (Costa and Oliveira 2002), artificial immune systems (Campelo, Guimarães and Igarashi 2007, for an overview), artificial life algorithms (Berry and Vamplew 2003), tabu search (Yang, Cardoso, Ho, Ni, Machado and Lo 2004, Ramirez-Rosado and Dominguez-Navarro 2006). The references indicated here are a good starting point for the reader interested in more information about these techniques.

2.6 Unified evolutionary algorithm

This chapter started with the general population-based evolutionary algorithm in Algorithm 2.1, which has the basic ingredients to produce an evolutionary optimization technique with interesting searching characteristics. Through the chapter, specific algorithms for both mono and multi-objective problems were reviewed and integrated into a computational framework for evolutionary optimization. However, due to some particularities of the evolution strategy and some multi-objective algorithms like the NSGA2, we notice some differences that make Algorithm 2.1 not that general.

The reason why Algorithm 2.1 is not that general is that NSGA2 and evolution strategies separate the offspring and the parent populations. Moreover, evolution strategies seem to use selection for survival while other evolutionary algorithms use selection for the variation step. The order in the sequence selection-variation does not match in all situations. The key to understand this apparent mismatch is to realize that all evolutionary algorithms implicitly have selection pressure before and after the variation step. The first selection pressure selects the individuals that will participate in the generation of the offspring and the second selection pressure selects those individuals that will survive to the next generation. From now on, let us call this second selection pressure a substitution operator. Although it is not explicitly mentioned in the algorithm for evolution strategies, see Algorithm 2.3, there is indeed an implicit selection for variation. Observe that the algorithms for the recombination operators, Algorithms 2.4-2.6,

randomly select the parent solutions that will generate the new offspring. Therefore, if λ offspring solutions have to be produced, every individual in the parent population has an equal probability of producing on average λ/μ offspring. The tradional selection in evolution strategies is not a selection for reproduction but really a selection for survival, that is, a substitution that preserves the population size constant. This observation was already made before in this chapter, when we compared the differences between evolution strategies and evolutionary programming. In the former, the selection for variation is stochastic with uniform probability and the substitution is deterministic, while in the latter, the selection is deterministic - each individual is deterministically selected to generate one offspring - and the substitution is ellitist and deterministic.

Having noticed that, we can now identify what needs to be done in order to make Algorithm 2.1 more general. We need to separate the parent and offspring populations and to explicitly add the substitution operator. This new unified model is presented in Algorithm 2.19. Having a unified algorithm such as Algorithm 2.19 is very practical for implementation purposes. The other algorithms follow this model and specific reproduction and update operators can be provided in the input data of this algorithm to make it perform as an evolution strategy, a genetic algorithm, a MOGA or an NSGA2. It also helps to identify similarities and differences among specific evolutionary algorithms, making the comparison more meaningful for instance. Next, we show that this algorithm can model all the other algorithms presented before.

2.6.1 One algorithm, multiple behaviors

All the algorithms discussed in this chapter can be considered as specific instances of Algorithm 2.19, which is more general than Algorithm 2.1. In this section we show how this single algorithm can present these multiple behaviors.

Let us start with the evolution strategy. Lines 4 and 5 in Algorithm 2.3 that generate the offspring are moved to outside the while loop in Algorithm 2.17. Therefore, before entering the while loop, Algorithm 2.19 first initializes the parent population $P(n = 0)$ and generates the first offspring $Q(n = 0)$ from this. The selection in line 8 of Algorithm 2.3 is in fact the substitution operator, which generates the next parent population. In evolution strategies, this substitution is done using either ES(μ, λ) or ES$(\mu+\lambda)$ schemes. The next offspring population in Algorithm 2.19 is produced via the reproduction step, which comprises selection (for reproduction) plus variation. In evolution strategy, the selection is not explicit but notice that an uniform roulette wheel selection would produce

Algorithm 2.19: Unified population-based evolutionary algorithm.

Data: population size μ, offspring size λ, maximum archive size ξ, search space $\mathcal{X}$, objective and constraint functions $\mathbf{f}(\cdot)$, $\mathbf{g}(\cdot)$, $\mathbf{h}(\cdot)$.
Result: Estimate(s) of $\mathcal{X}^*$ in the archive population $A(n)$.

1 $P(n=0) = \{\mathbf{p}^{(1)}, \ldots, \mathbf{p}^{(\mu)}\} \leftarrow$ `Initialize population`$(\mu, \mathcal{X})$;
2 $A(n=0) = \varnothing \leftarrow$ `Initialize archive /* Stores the best solution set */`
3 $\Upsilon_P(n) \leftarrow$ `Penalty`$(P(n), \mathbf{f}, \mathbf{g}, \mathbf{h})$;
4 $\Phi_P(n) \leftarrow$ `Fitness`$(\Upsilon_P(n))$;
5 $A(n=0) \leftarrow$ `Update`$(A(n), P(n), \xi)$;
`// Reproduction comprises selection and variation`
6 $Q(n=0) = \{\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(\lambda)}\} \leftarrow$ `Reproduction`$(P(n), A(n), \Phi_P(n))$;
7 **while** $\neg$ *stop criteria* **do**
8 $\quad \Upsilon_Q(n) \leftarrow$ `Penalty`$(Q(n), \mathbf{f}, \mathbf{g}, \mathbf{h})$;
9 $\quad \Phi_Q(n) \leftarrow$ `Fitness`$(\Upsilon_Q(n))$;
10 $\quad P(n+1) \leftarrow$ `Substitution`$(P(n), Q(n))$;
11 $\quad Q(n+1) \leftarrow$ `Reproduction`$(P(n+1))$;
12 $\quad A(n+1) \leftarrow$ `Update`$(A(n), P(n) \cup Q(n), \xi)$;
13 $\quad n \leftarrow n+1$;
14 **end**

the same effect of the implicit selection of standard evolution strategies.

In evolutionary programming, $\lambda = \mu$, and the selection is deterministic. Therefore, the selection operator of evolutionary programming could be implemented as a simple copy function, that is, the selected population is a copy of the parent population. The substitution operator is equivalent to the ES$(\mu + \mu)$ scheme, i.e., the best μ individuals survive to the next generation. Variation is performed only by mutation, without recombination.

In genetic algorithms, λ is also equal to μ. The offspring population in genetic algorithms is not evident, due to their implicit substitution method. The offspring population automatically substitutes for the parent population. Therefore, the substitution is not evident in Algorithm 2.9, but Algorithm 2.19 can behave as a genetic algorithm if we use an stochastic selection for reproduction, crossover and mutation for the variation step, and finally, if we adopt an ES(μ, μ) scheme for substitution, which is equivalent to the implicit substitution for survival in standard genetic algorithms.

In differential evolution, the selection for reproduction is also implicit, but notice in the `for` loop in lines 4 to 8 in Algorithm 2.14 that every individual generates one offspring. This is equivalent to what is done in evolutionary programming. Thus, the

selection in Algorithm 2.19 would be a simple copy function. Selection is ellitist for each pair of parent-offspring individuals. This scheme is equivalent to an ES$(1 + 1)$ for each corresponding pair in $P(n)$ and $Q(n)$.

The analysis is also valid for the multi-objective algorithms. The main difference resides in the fitness and update functions in lines 9 and 12 of Algorithm 2.19. In the NSGA2, see Algorithm 2.14, the reduction routine can be represented by the substitution step in Algorithm 2.19 and is similar to a ES$(\mu + \mu)$ strategy. Moreover, this algorithm explicitly maintains the parent and offspring population separate, which is already accomodated in Algorithm 2.19.

The flowchart in Figure 2.8 illustrates the architecture of the framework. Table 2.2 summarizes the specific configuration for Algorithm 2.19 in order to make it behave as a specific evolutionary algorithm presented before. It is worth noting that these algorithms are not the only possible alternatives in the framework. One can easily mix different selection, variation and substitution operators, hence producing non-standard algorithms.

2.7 Conclusion

This chapter was divided into two main parts. The first part was devoted to the precise statement of the optimization problem with one or more objectives. The second part provided an overview of evolutionary optimization techniques for solving the problems stated in the first part.

We began the presentation of evolutionary optimization with a fairly general algorithm, then we described some particular instances, and finally we concluded the chapter with a unified algorithm that accommodates all the algorithms presented before. This is the main body of the framework for evolutionary optimization developed in this work and it is adequate for mono and multi-objective problems. Therefore, the discussion moved from the general to the particular, converging in the end to an even more general and unifying view of evolutionary optimization techniques. As discussed before, this unified model presents a number of behaviors, including canonical and non-canonical algorithms.

In the next chapter we examine memetic algorithms. The evolutionary algorithms

Figure 2.8: Flowchart description of the framework based on the unified model.

Algorithm	Fitness function	Substitution	Selection for reproduction	Variation
ES	scalar	ES(μ, λ) or ES$(\mu + \lambda)$	uniform roulette	recombination (optional) plus mutation
EP	scalar	ES$(\mu + \mu)$	deterministic copy	mutation
GA	scalar	ES(μ, μ)	stochastic (roulette wheel or tournament)	crossover plus mutation
DE	scalar	ES$(1 + 1)$ for each parent-offspring pair	deterministic copy	mutant vector plus crossover
MOGA	ranking	ES(μ, μ)	stochastic (roulette wheel)	crossover plus mutation
NPGA	ranking	ES(μ, μ)	stochastic (tournament)	crossover plus mutation
NSGA2	nondominated sorting	ES$(\mu + \mu)$	stochastic (roulette wheel or tournament)	crossover plus mutation
SPEA2	strength value	ES(μ, μ)	stochastic (roulette wheel or tournament)	crossover plus mutation

Table 2.2: Configuration of the unified model to produce a specific evolutionary algorithm

presented here can be directly hybridized with the local search methodology that we will discuss in the next chapter. The local search operator will be part of the reproduction step. Therefore, Algorithm 2.19 is still valid as a model for memetic or hybrid algorithms. Algorithm 2.19 can behave as a memetic algorithm if an adequate local search operator is provided together with its variation operators.

Chapter 3

Memetic Optimization

"Order and simplification are the first steps towards the mastery of a subject."
— Thomas Mann, 1875–1955

3.1 Preview

The previous chapter presented the mathematical definition of the optimization problem with continuous variables and also an overview of evolutionary optimization methods suitable for solving it. Chapter 2 concluded with a general unifying model, presented in Algorithm 2.19. This algorithm can also model memetic algorithms, by just including a local search operator in the reproduction step.

Nonetheless, the connection between global and local search algorithms can be performed in a number of different ways. This chapter begins with a discussion on possible hybridization schemes. After that, we present the local learning approach for memetic algorithms, in which the local behavior of the nonlinear objective and constraint functions is learned via a radial basis function (RBF) approximation. We review the multiquadric interpolation method, a kind of radial basis function, which is used for local learning. We present the local search methodology for mono and multi-objective problems, which integrates a local search method with the RBF approximations. This methodology gives rise to specific operators to be used in the variation step of memetic algorithms. An optional trust region update methodology is also developed for this local search. The

chapter ends with a discussion on the specific hybridization adopted in this thesis.

3.2 Hybridization schemes

In general, a memetic algorithm can be defined as the combination of local search operators and traditional reproduction operators in evolutionary algorithms. In this section, we discuss approaches for this hybridization.

3.2.1 Classical hybridization

Classical hybridization techniques consist of performing a complete execution of the global search algorithm, with the objective of identifying a promising attraction basin. The solution obtained by the global search is then used to start the local search (Yang, Park, Park and Ra 1995, O. A. Mohammed 1997). The local search is used to fine tune the solution. We can recognize this approach as a classical one, because it is largely adopted in optimization and greatly explored in literature. However, this classical hybridization cannot be recognized as a kind of memetic algorithm, since the local search is not part of the reproduction step of the algorithm. It is presented here for historical reasons and for a better organization of the chapter. After the execution of an instance of Algorithm 2.19, a local search method is applied to the solutions returned in $A(n)$.

This approach is indeed simple and efficient in many applications, but the definition of the transition criteria is a key issue in its utilization. In general, some level of experience from the user is required, that is, the user defines how much time the global search should run until switching to the local search. Of course, budget time limits are also relevant in this decision. More sophisticated strategies for switching the global and local search were also proposed, see (Vasconcelos, Saldanha, Krahenbuhl and Nicolas 1997).

The classical hybridization can also be associated with approximation techniques. In (Rashid, Ramírez and Freeman 2000), for instance, the classical hybridization strategy is employed over a global approximation generated by a neuro-fuzzy network. The derivative information analytically extracted from the network was used to perform the gradient-based local search. Another possibility is using the global search method directly over the real functions. The samples obtained during the global search are then used to generate a local approximation around the best point. Finally, the local search

method is applied to the local approximation.

3.2.2 Baldwinian approach

In biology, the Baldwin effect is the effect of learning by individual animals over their lifetime on the species evolution. Put differently, the learning ability of individuals can guide the evolutionary process. In fact, the Baldwin effect opposes Lamarckism, by stating that acquired characteristics can be only indirectly inherited. This effect is called like this in honor of the american naturalist J. Mark Baldwin, who described the phenomenon in 1896. The first computational work on the Baldwin effect appeared in (Hinton and Nowlan 1987). Since then, some research has been done on the computational aspects of the Baldwin effect (Mayley 1996, Turney 1996).

In evolutionary algorithms, we can implement the Baldwin effect by considering that individual learning changes the fitness value of the individual without changing its genotype. The local search should be used immediately before the selection step, modifying the fitness values based on the quality of the locally improved solutions. The local search finds an improved solution whose fitness is used to represent the original solution. In this way, the fitness value of a given point in the search space is not directly related to the evaluation of this point, but to the best value in the vicinity of this point. This approach has the effect of smoothing the fitness surface, which in turn helps the evolutionary process. The Baldwin effect can be understood as a filtering of the original fitness surface, possibly with some important degree of multimodality.

The individual learning has an indirect effect on the evolution, which means that the representation of the solution in the search space does not change, i.e., the new improved solution is not genetically assimilated by the individual. The indirect effect is that individuals have their chances of being selected based on their potential to achieve good solutions. Therefore, individuals represent regions in the search space, not only points. In theory, assuming a "perfect" local optimizer, all individuals in the same attraction basin would have the same fitness value.

Figure 3.1 shows the fitness landscape of a given multimodal objective function and three individuals $p^{(1)}$, $p^{(2)}$, and $p^{(3)}$. With the Baldwinian approach, the fitness value of $p^{(3)}$ can be higher than the fitness value of $p^{(2)}$ after a Baldwinian local search, albeit the original fitness value at $p^{(3)}$ is lower than that at $p^{(2)}$. In this way, the evolutionary process can be biased in the direction of $p^{(3)}$, which is within a potentially better region.

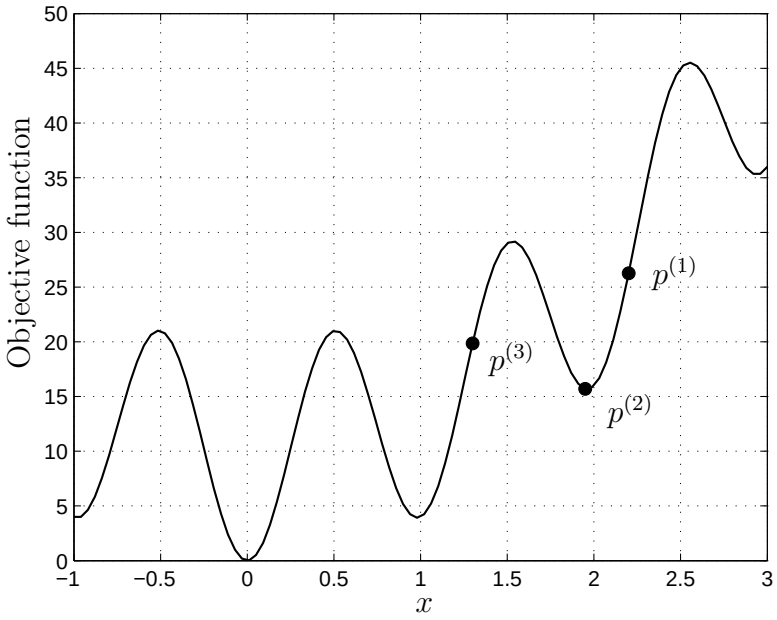

Figure 3.1: Landscape of a multimodal function $f(x)$. With the Baldwinian approach, $p^{(1)}$ and $p^{(2)}$ can have the same fitness value. Moreover, $p^{(3)}$ can have a fitness value higher than that of $p^{(2)}$.

With time, as the population converges in the direction of $p^{(3)}$, the best solution in this region can be geneticaly assimilated by the population, i.e., the improvement achieved by individual learning is indirectly assimilated. That is the Baldwin effect in an evolutionary optimization algorithm. The individuals $p^{(1)}$ and $p^{(2)}$ can have the same or very similar fitness values, after Baldwinian local search. Therefore, there will be no bias or very little bias between them, hence both will have similar chances of being selected by the stochastic selection operator. In practice, we do not have "perfect" optimizers and the local search requires an additional computational cost. By restricting the reach of the local search or the number of maximum evaluations allowed, we obtain different fitness values for $p^{(1)}$ and $p^{(2)}$.

3.2.3 Lamarckian approach

In 1801, Jean-Baptiste Lamarck proposed that individual learning would have an effect on the inherited characteristics. In this way, traits acquired during lifetime could be

inherited by descendants. Today, Lamarck's ideas have no ground in biology[1], but it is possible to apply them in the artificial world of computers with interesting results. In the context of hybrid algorithms, we can apply the Lamarckian approach by changing the genotype after local search. In contrast to the Baldwinian approach, in which only the phenotype is modified, in the Lamarckian approach the genotype is modified after execution of the local search. The individual can change his inheritance to its offspring. The initial solution, provided by the previous generation, is actually substituted by the new solution.

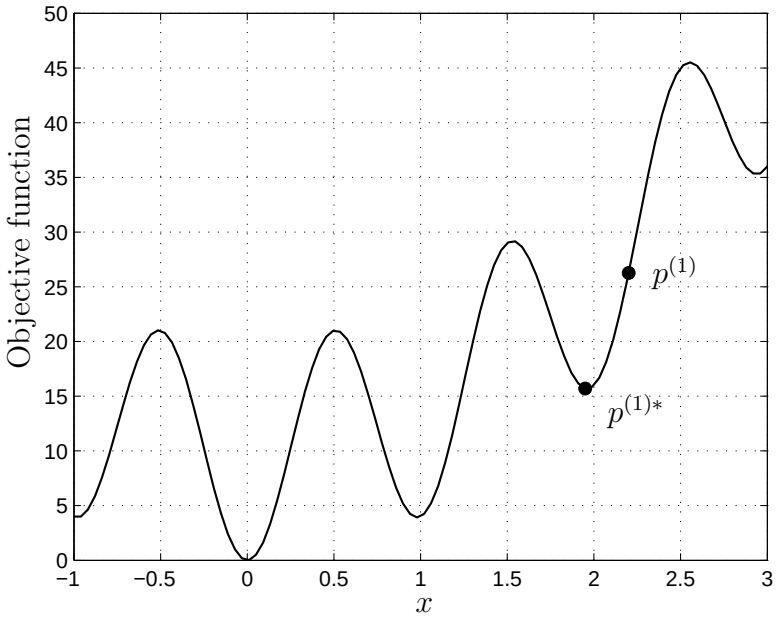

Figure 3.2: Landscape of a multimodal function $f(x)$. After Lamarckian local search, the solution represented by $p^{(1)}$ is actually modified, representing the improved solution $p^{(1)*}$ in the search space.

Figure 3.2 illustrates an individual $p^{(1)}$ in the search space that has its genotype modified by local search to represent another point in the search space.

Many variants for the association of global and local search methods are available in the literature, especially in the field of combinatorial optimization (Krasnogor and Smith 2005). In optimization with continuous variables, the Lamarckian approach is more used (Kimura and Konagaya 2003, Ong and Keane 2004).

[1]To be fair with Lamarck, he was right and ahead of his time for his belief that evolution had occurred. His mistake was only about the mechanism of evolution.

3.3 Approximation-based local search

Evolutionary methods can be regarded as adaptive sampling techniques, making an "intelligent" sampling of the search space. This perceived intelligence is determined by the operators of the algorithm that direct the sampling of the search space towards the most promising regions. As noted before, these heuristic operators are designed based on some general premises about the problem, namely, that better solutions can be found by generating more samples in the vicinity of the best points, and that some level of global exploration is needed to find better solutions by chance.

This sampling characteristic of evolutionary methods can be advantageously used to reduce the computational cost of the local search phase in memetic algorithms, especially when considering the class of problems of interest in this work. Individuals can employ part of the accumulated information acquired by the consecutive populations to build simpler and less expensive models of the functions involved in the optimization problem. Assuming that the optimization problem fits into the class of problems delineated in Chapter 1, the accumulated information represented by the samples gathered by the algorithm is very valuable. We can progressively reconstruct the black-box input-output relationship to an arbitrarily high precision, which in turn may be exploited to speed up the searching process via local search operators.

By viewing the evolutionary process as an intelligent sampling process, we note that there will be more samples in the most promising regions of the search space. In these regions, the evolutionary convergence is slow, with poor precision due to genetic drift caused by the variation operators. As the generations advance, we obtain more samples in these regions, making the approximations more accurate. It is reasonable to believe that the local approximations will improve as the search progresses, making the solutions achieved by indirect local search arbitrarily close to the ones that would be obtained by direct local search.

To conclude this section, we point out that:

- We assume that the time spent in the evaluation step is dominant in the whole process;
- We require that the total time spent in the optimization process when using the hybrid algorithm with approximation-based local search must be less than that spent when using the standard algorithm.

The first observation is often true in CAD problems. It implies that some additional complexity in the algorithm operations is justifiable. The time needed to generate and evaluate the approximations must be small in comparison to the time consumed in evaluating solutions directly. The second point, actually a requisite, means that the hybrid algorithm must converge with less evaluations than the standard algorithm, or at least provide a better solution for the same number of evaluations. We will return to these points in Chapter 4. For now, we detail how the approximations are generated and how the local search is performed in mono and multi-objective contexts.

3.3.1 Multiquadric interpolation method

The multiquadric interpolation (MQI) was originally proposed in (Hardy 1971) as a tool for interpolating level surface curves in topography. Since then, the method has been applied in many different contexts requiring interpolation of multidimensional data (Hardy 1990, for a review). The multiquadric and the inverse multiquadric functions are now regarded as types of radial basis functions, together with other functions based on radial distances that make RBF neural networks universal approximators, see (Park and Sandberg 1991) for a discussion on the subject. The MQI technique has also demonstrated its power and efficiency in electromagnetic optimization problems (Alotto, Caiti, Molinari and Repetto 1996, Canova, Gruosso and Repetto 2003, Coulomb, Kobetski, Costa, Maréchal and Jönsson 2003), although these works employ it in order to generate global approximations. In (Guimarães, Campelo, Saldanha, Igarashi, Takahashi and Ramírez 2006), the authors employ the multiquadric interpolation to obtain local approximations within a multi-objective artificial immune system optimization algorithm. The samples obtained in the first iterations are stored and used to generate local approximations, which substitute for the expensive eletromagnetic solver in last iterations.

The multiquadric model is given by the linear superposition of nonlinear quadric functions centered at each sample point:

$$m(\mathbf{x}) = \sum_{i=1}^{N} w_i q_i(\mathbf{x}) = \sum_{i=1}^{N} w_i \left[\|\mathbf{x} - \mathbf{c}_i\|^2 + s^2 \right]^{1/2} \tag{3.1}$$

where N is the number of input-output pair samples, $\mathbf{x}$ is the vector of optimization variables, $\mathbf{c}_i$ is the center of $q_i(\,\cdot\,)$ and s is a constant called the shape parameter.

When the multiquadric functions are used in radial basis function neural networks

with less neurons than data points, the centers and the shape parameter, which are the nonlinear parameters of the network, can be adjusted by specific training equations. The weights can be found by the least square estimator method, since there is only one layer and the output function is linear with respect to the weights (Jang, Sun and Mizutani 1997). In this case, the function built by the network is considered an approximation since there is no guarantee that it is exact on all data points.

In the interpolation usage of multiquadric functions, each input point in the data set is defined as a center, so the number of centers is equal to the number of data points. The s constant is defined a priori and is a critical aspect in the use of the method. The only parameters that need to be determined are the coefficients w_i. In this case the function obtained is considered an interpolation of the data, because it is exact on all data points under the machine precision.

The w coefficients are determined from the set of samples by assembling the system of N equations and N unknowns:

$$\begin{aligned} m_1 &= \sum_{i=1}^{N} w_i q_i(\mathbf{c}_1) = \sum_{i=1}^{N} w_i \left[\|\mathbf{c}_1 - \mathbf{c}_i\|^2 + s^2 \right]^{1/2} \\ &\vdots \\ m_N &= \sum_{i=1}^{N} w_i q_i(\mathbf{c}_N) = \sum_{i=1}^{N} w_i \left[\|\mathbf{c}_N - \mathbf{c}_i\|^2 + s^2 \right]^{1/2} \end{aligned}$$

where $m_i = m(\mathbf{c}_i)$.

Expressing in matricial form, we have:

$$\mathbf{m} = \mathbf{Q}\mathbf{w} \tag{3.2}$$

where $Q_{ij} = Q_{ji} = q_i(\mathbf{c}_j)$.

Sensitivity Information

Once the MQ model has been generated, we have an analytical description of the objective function. In this way, we may express the gradient vector and the Hessian matrix analytically, which are useful for deterministic optimization methods. The advantanges of deriving sensitivity information are the use of more accurate derivatives and less computational effort. For instance, the numerical estimation of the gradient using forward

differences would have a cost of the order of $(d+1)N$ evaluations of a single multiquadric function. The analytical evaluation of the gradient has a cost of the order of N evaluations of a single multiquadric function. If the values of the multiquadric functions are stored when evaluating $m(\mathbf{x})$ (raising the storage cost) the gradient is obtained almost inexpensively.

The derivatives of the MQ model are:

$$\frac{\partial m(\mathbf{x})}{\partial x_i} = \sum_{n=1}^{N} w_n (x_i - c_{ni}) q_n^{-1}(\mathbf{x}) \tag{3.3}$$

$$\frac{\partial^2 m(\mathbf{x})}{\partial x_i \partial x_j} = -\sum_{n=1}^{N} w_n (x_i - c_{ni})(x_j - c_{nj}) q_n^{-3}(\mathbf{x}) \tag{3.4}$$

$$\frac{\partial^2 m(\mathbf{x})}{\partial x_i^2} = \sum_{n=1}^{N} w_n \left[q_n^{-1}(\mathbf{x}) - (x_i - c_{ni})^2 q_n^{-3}(\mathbf{x}) \right] \tag{3.5}$$

which are very simple to implement. In fact, these expressions are simpler than the analytical derivatives of some neural network models or neuro-fuzzy models (Rashid, Ramírez and Freeman 2000). During the evaluation of $m(\mathbf{x})$, we may store the values $q_n(\mathbf{x}), n = 1, \ldots, N$. After that, the gradient and the Hessian at $\mathbf{x}$ are obtained with the expressions above.

When using quasi-Newton methods, only the gradient is needed, since the Hessian is approximated. Nonetheless, the Hessian may be helpful for sensitivity analysis, as illustrated in (Takahashi, Ramírez, Vasconcelos and Saldanha 2001).

The shape constant

In the multiquadric interpolation method, the value for s^2 is defined *a priori*. A suggested value found in literature is (Hardy 1990):

$$s^2 = \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} \|\mathbf{c}_i - \mathbf{c}_j\|^2}{N(N-1)} \tag{3.6}$$

However, this formula is not general and may produce undesirable performance in some problems. In fact, this value must be small, or it will dominate the quadric function if $\|\mathbf{x} - \mathbf{c}_n\|^2 << s^2$. On the other hand, this constant interferes in the smoothness of the function, which is a desirable characteristic. In order to get a more reasonable definition

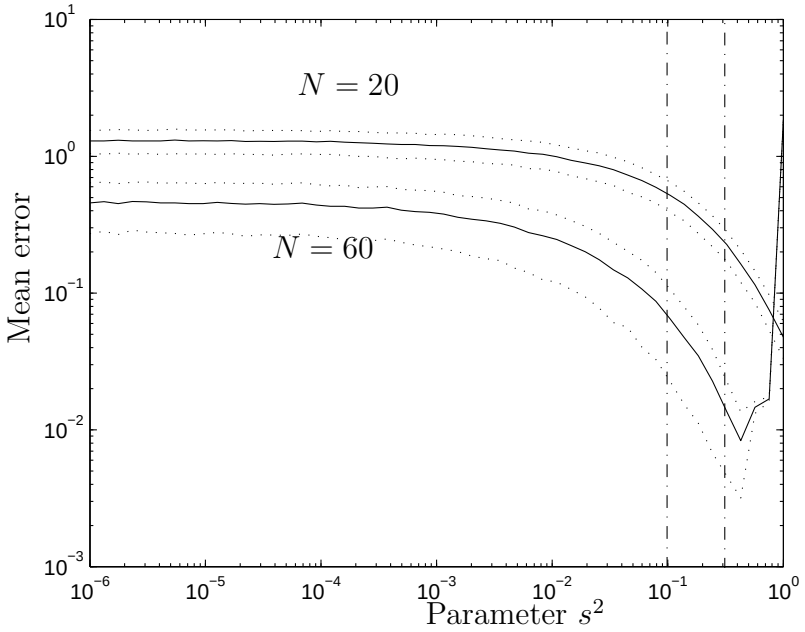

Figure 3.3: Mean curves for the MSE versus the s^2 parameter in logarithmic scales.

of a "small" value, the optimization variables are normalized to the interval $[0, 1]$.

A typical error curve with respect to s^2 is shown in Figure 3.3 for the function

$$f(\mathbf{x}) = 2(x_1 - 1)\sin(x_1 + x_2)\cos(x_1 - x_2) \tag{3.7}$$

in the interval $0 \leq x_{1,2} \leq 2$.

In Figure 3.3, we measure the error of the MQ model for various values for s^2. The error measure is the mean squared error, MSE, over N test points randomly distributed in the space of variables. The MSE is calculated 100 times for each value of s^2, considering different test points. In this way we obtain the mean and standard deviation of the error measure, which are shown respectively in solid and dotted lines. We can see from these curves that the error decreases when more sample points are used to build the approximation or when the value for s^2 increases. However, if s^2 is too big, it dominates the quadric function and the error increases because of ill-conditioning. A safe region for s^2 is from about 0.1 to 0.3.

It is not guaranteed that this value is optimal for all problems, but after normalizing the input space, we obtain a more meaningful value for s^2. Note that the expressions

for the derivatives must be scaled for considering the normalization of the data points. The scale factor is:

$$\sigma_k = \frac{1}{u_k - l_k} \tag{3.8}$$

where $[l_k, u_k]$ corresponds to the inferior and superior limits for the kth variable.

The ith element of the gradient must be multiplied by σ_i and the element ij of the Hessian must be multiplied by $\sigma_i \sigma_j$.

Many other approximation techniques available in the literature can be utilized to develop approximation-based local search operators. We refer to (Jin 2005) for an overview of different techniques for fitness approximation in evolutionary optimization. In this thesis, we focus on the multiquadric interpolation method, but the methodology for local search is very general. Approximations obtained by other techniques can be accomodated as well. As an example, in (Guimarães, Campelo, Igarashi, Lowther and Ramírez 2007) the proposed methodology is associated to typical multilayer perceptron neural networks with satisfactory results. However, neural networks require the specification of the topology, which is not an easy task. There are training methods designed to optimize topology, that is to say, to deal with the bias-variance dilemma, but they are more complicated. In the context of approximation-based memetic algorithms, the local approximations are generated many times during the evolutionary process, thus the generation of the local model has to be simple and fast. This is pointed out in the second requirement stated in the beginning of this section. Training algorithms for neural networks are also optimization procedures, whose cost may become important when used many times. RBF interpolation is relatively fast and provides efficient and satisfactory local approximations (Regis and Shoemaker 2004, Guimarães, Wanner, Campelo, Takahashi, Igarashi, Lowther and Ramírez 2006). Multiquadric interpolation, in particular, provides a simple and smooth interpolation between data points (Hardy 1990).

3.3.2 The local search operator

All evaluations performed by the evolutionary algorithm are stored in a global data set:

$$\mathcal{D}(n) = \left\{ \mathbf{z}^{(i)}; f_1(\mathbf{z}^{(i)}), \ldots, f_m(\mathbf{z}^{(i)}); g_1(\mathbf{z}^{(i)}), \ldots, g_p(\mathbf{z}^{(i)}); h_1(\mathbf{z}^{(i)}), \ldots, h_q(\mathbf{z}^{(i)}) \right\}_{i=1}^{N(n)} \tag{3.9}$$

where $N(n)$ is the total number of samples collected by the algorithm at iteration n. $\mathcal{D}(n)$ contains each solution $\mathbf{z}^{(i)}$ in the search space tested by the algorithm and its respective values for the objective and constraint functions. Only nonlinear functions are stored. Linear constraints are explicitly provided by the user and are not approximated.

$\mathcal{D}(n)$ is the global data set representing all information acquired by the algorithm until time n. Part of this information, the data within the neighborhood of the individual, is used for building the local approximations, specifically the multiquadric interpolation described before.

Define $\mathbf{x}^{(c)}$ as the solution represented by the individual $\mathbf{p}^{(i)} \in P(n)$ selected for local search. The neighborhood region is centered at $\mathbf{x}^{(c)}$ and is generaly defined as follows:

$$\mathcal{V}(\mathbf{x}^{(c)}, \epsilon) = \left\{\mathbf{z} : \|\mathbf{z} - \mathbf{x}^{(c)}\| \leq R(\epsilon)\right\} \tag{3.10}$$

where $R(\epsilon)$ is a region whose size is parameterized by ϵ.

Considering $\|\cdot\|_\infty$, we have a rectangular neighborhood:

$$\mathcal{V}(\mathbf{x}^{(c)}, \epsilon) = \left\{\mathbf{z} : x_k^{(c)} - \epsilon(u_k - l_k) \leq z_k \leq x_k^{(c)} + \epsilon(u_k - l_k),\ k = 1, \ldots, d\right\} \tag{3.11}$$

and considering $\|\cdot\|_2$, we have an ellipsoidal neighborhood

$$\mathcal{V}(\mathbf{x}^{(c)}, \epsilon) = \left\{\mathbf{z} : (\mathbf{z} - \mathbf{x}^{(c)})^T \mathbf{\Delta} (\mathbf{z} - \mathbf{x}^{(c)}) \leq 1\right\} \tag{3.12}$$

in which the matrix $\mathbf{\Delta}$ is given by:

$$\Delta_{ij} = \begin{cases} [\epsilon(u_k - l_k)]^{-1}, & i = j \\ 0, & i \neq j \end{cases} \tag{3.13}$$

The parameter $0 < \epsilon < 1$ defines the size of the local neighborhood with respect to the parameter range. The parameter is set to 0.1 by default.

Algorithm 3.1 shows how the local data set $\mathcal{L}$ is assembled from $\mathcal{D}$. Identical points and points closer than a threshold ζ do not enter the local data set. The elimination of very similar points is important, because they cause ill-conditioning of the matrix $\mathbf{Q}$ in the multiquadric interpolation and also in multilayer perceptron networks trained by second order methods (Levenberg-Marquadt backpropagation algorithm, for example).

Algorithm 3.1: Building local data set

```
1  L = x^(c) /* Stores local data set                                  */
2  for i = 1, ..., N(n) do
3  |  if z^(i) ∈ V(x^(c), ε) then
4  |  |  calculate the distance between z^(i) and each data point in L;
5  |  |  if z^(i) is at least an amount ζ distant from all points in L then
6  |  |  |  put z^(i) and their respective evaluations in L;
7  |  |  end
8  |  end
9  end
10 return L;
```

The approximations are generated to fit data in $\mathcal{L}$. With the approximations in hand, we can define the local search problem as[2]:

$$\begin{aligned} &\min \tilde{\mathbf{f}}(\mathbf{x}) \in \mathbb{R}^m \\ &\text{subject to: } \mathbf{x} \in \widetilde{\Omega} \cap \mathcal{V}(\mathbf{x}^{(c)}, \epsilon) \end{aligned} \tag{3.14}$$

in which $\mathcal{V}(\mathbf{x}^{(c)}, \epsilon)$ is the local neighborhood, $\tilde{\mathbf{f}}(\,\cdot\,)$ are the approximations for the objective functions, and $\widetilde{\Omega}$ is the approximated feasible set:

$$\widetilde{\Omega} = \mathcal{X} \bigcap \left(\bigcap_{i=1}^{p} \widetilde{\mathcal{G}}_i \right) \bigcap \left(\bigcap_{j=1}^{q} \widetilde{\mathcal{H}}_j \right) \tag{3.15}$$

and:

$$\widetilde{\mathcal{G}}_i = \{\mathbf{x} : \tilde{g}_i(\mathbf{x}) \leq 0\} \tag{3.16}$$

$$\widetilde{\mathcal{H}}_j = \{\mathbf{x} : \tilde{h}_j(\mathbf{x}) = 0\} \tag{3.17}$$

Observe that the local search problem has an additional constraint in comparison to the original optimization problem: the local problem is restricted to the region $\mathcal{V}(\mathbf{x}^{(c)}, \epsilon)$.

The mono-objective case is easily and directly solved by employing the Sequential Quadratic Programming - SQP - method. The multi-objective version needs some additional steps to transform the problem into a constrained mono-objective auxiliary

[2]The mono-objective problem is a particular case with $m = 1$.

problem.

Define the vector $\mathbf{z}_i^* \in \mathbb{R}^d$ as the point of minimum of the approximated objective function $\tilde{f}_i(\,\cdot\,)$ considered individually. For each such function, this vector is obtained after solving (using SQP method):

$$\begin{aligned} &\mathbf{z}_i^* = \arg\min \tilde{f}_i(\mathbf{x}) \\ &\text{subject to: } \mathbf{x} \in \widetilde{\Omega} \cap \mathcal{V}(\mathbf{x}^{(c)}, \epsilon) \end{aligned} \tag{3.18}$$

Let $\mathbf{v}_i \in \mathbb{R}^m$ be the objective vector associated with each single-objective optimum point $\mathbf{z}_i^*$:

$$\mathbf{v}_i = \tilde{\mathbf{f}}(\mathbf{z}_i^*) = \begin{bmatrix} \tilde{f}_1(\mathbf{z}_i^*) \\ \vdots \\ \tilde{f}_m(\mathbf{z}_i^*) \end{bmatrix} \tag{3.19}$$

Then define $\mathbf{o} \in \mathbb{R}^m$ as the vector, in the objective space, consisting of the minimal values of all approximated objective functions:

$$\mathbf{o} = \begin{bmatrix} \tilde{f}_1(\mathbf{z}_1^*) \\ \vdots \\ \tilde{f}_m(\mathbf{z}_m^*) \end{bmatrix} \tag{3.20}$$

Note that, by construction, except in the particular case in which all objective functions share the same optimum point, there is no point in the search space $\mathcal{X}$ that has, as its image, the point $\mathbf{o}$ in the objective space $\mathcal{Y}$. Due to this reason, this point is often called the *utopian solution* of the problem. Finally, define the positive convex cone $\mathcal{K}$ with generating vectors $\{\mathbf{v}_i - \mathbf{o}\}_{i=1}^m$:

$$\mathcal{K} = \left\{ \mathbf{k} \in \mathbb{R}^m : \mathbf{k} = \sum_{i=1}^{m} \alpha_i (\mathbf{v}_i - \mathbf{o}), \;\; \alpha_i \geq 0 \right\} \tag{3.21}$$

The goal attainment formulation becomes defined by the auxiliary problem (Gembicki and Haimes, 1975):

$$\mathbf{x}^*, \gamma^* = \arg\min_{\mathbf{x},\gamma} \gamma$$

$$\text{subject to: } \begin{cases} \mathbf{x} \in \widetilde{\Omega} \cap \mathcal{V}(\mathbf{x}^{(c)}) \\ \gamma > 0 \\ \tilde{\mathbf{f}}(\mathbf{x}) \leq \mathbf{o} + \gamma \mathbf{k} \end{cases} \tag{3.22}$$

in which the scalar γ is a relaxation variable.

This auxiliary problem is in a form that can be adequately solved using the SQP method. Using different vectors $\mathbf{k} \in \mathcal{K}$, different Pareto-optimal points are found in each run. Each point belonging to the Pareto-set can be found with this formulation, associated with at least one combination of weights $(\alpha_1, \ldots, \alpha_m)$.

A graphical interpretation of the goal attainment procedure is presented in Figure 3.4. A vector $\mathbf{k}$ in the cone $\mathcal{K}$ formed by the vectors $\mathbf{v}_1 - \mathbf{o}$ and $\mathbf{v}_2 - \mathbf{o}$ defines a direction in the objective space. The auxiliary problem defined by the goal attainment formulation performs the search along this direction until a locally Pareto-optimal point is reached. Each solution of the auxiliary problem (3.21) provides a new individual that enhances the local approximations of the problem functions, and therefore is expected to enhance the original functions too.

3.3.3 An illustrative example

We now turn to an example to illustrate the local search operator presented in the previous section.

Consider that the objective function associated with the optimization of a given electromagnetic device has the surface pattern shown in Figure 3.5. The search space $\mathcal{X}$ is defined as $0 \leq x_{1,2} \leq 5$.

The local neighborhood is arbitrarily defined as a rectangular neighborhood as in (3.10) with $\epsilon = 0.1$, which gives:

$$\mathcal{V}(\mathbf{x}^{(c)}, 0.1) = \left\{ \mathbf{z} : x_k^{(c)} - 0.5 \leq z_k \leq x_k^{(c)} + 0.5,\ k = 1, \ldots, 2 \right\} \tag{3.23}$$

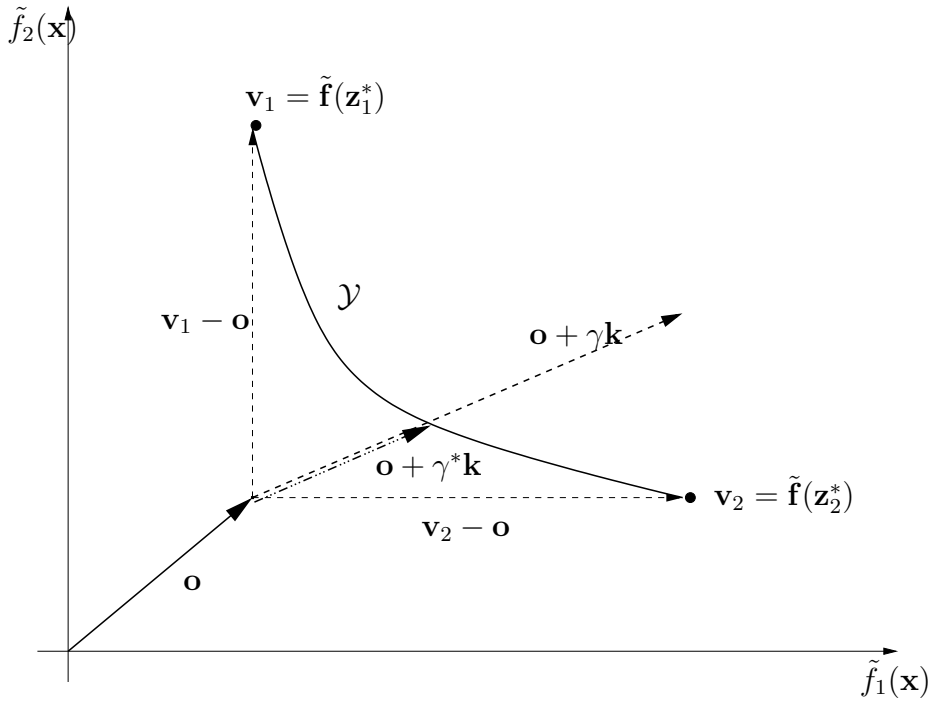

Figure 3.4: Geometric interpretation of the goal attainment procedure.

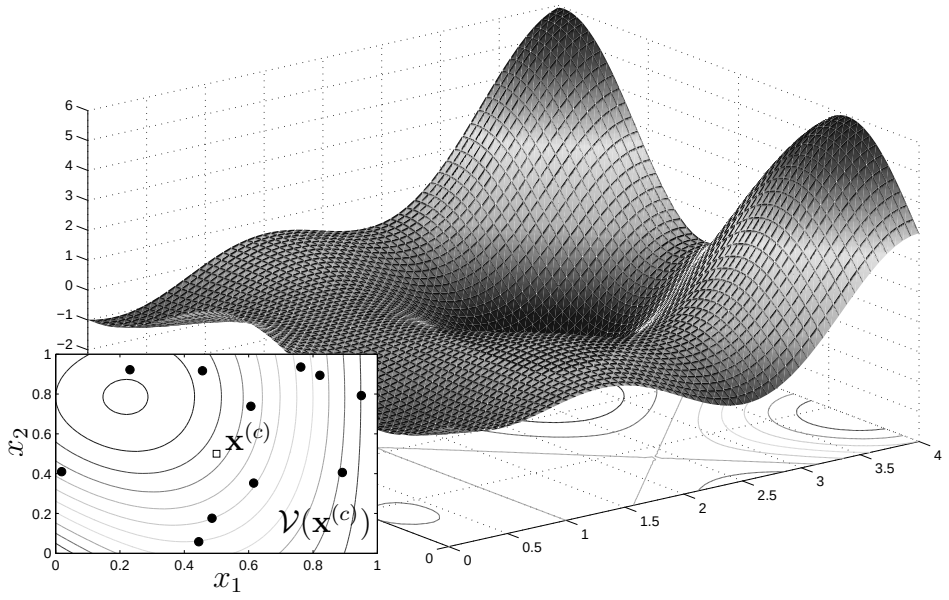

Figure 3.5: Illustration of the approximation-based local search.

At every generation, the evaluation of all offspring produced is stored in the global data set $\mathcal{D}$. Suppose that at a given generation, the point

$$\mathbf{x}^{(c)} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix} \tag{3.24}$$

is selected for local search. This point and its local neighborhood $\mathcal{V}(\mathbf{x}^{(c)}, 0.1)$ are shown in the small rectangle in Figure 3.5. The contour plot of the objective function inside the local neighborhood is also shown.

The first step is to assemble the local data set $\mathcal{L}$ from the samples in $\mathcal{D}$. These points are also shown in Figure 3.5. With these points, we build a multiquadric model for the objective function that is valid inside the region $\mathcal{V}(\mathbf{x}^{(c)}, 0.1)$. If this problem has nonlinear constraint functions, they are also approximated using the same data in $\mathcal{L}$.

The final step is to solve the local search problem using these approximations. The new point is restricted to belong to the region $\mathcal{V}(\mathbf{x}^{(c)}, 0.1)$. In the case of a rectangular neighborhood, this restriction leads to additional box constraints. In the case of an ellipsoidal neighborhood, this restriction leads to an additional quadratic constraint.

3.3.4 Local approximation and trust region update

In the previous section, the size of the local neighborhood is fixed and parameterized by ϵ as a fraction of the optimization parameter range. However, this fixed value can be small in some situations or big in others, meaning that some automatic adjustment procedure may be useful. Moreover, the local approximations can be used in an "open-loop" or "closed-loop" sense. In the former, the solution provided by the approximation-based local search replaces the initial solution (in a Lamarckian approach) and in the latter, the new solution is compared with the initial solution before replacement. The quality of the local approximations can be improved by decreasing the neighborhood size or, if its quality is already good, the quality of the solution achieved by the local search can be improved by increasing the neighborhood size. A procedure to automatically control the parameter ϵ can be devised based on ideas adopted in trust region methods.

Originally, trust region methods were developed to overcome difficulties faced by line-search methods in unconstrained second-order steepest descent algorithms (Kelley 1999). A trust region is defined as the region in which the local quadratic approximation built by

these deterministic methods can be trusted to accurately represent the original function. A basic trust region algorithm has two main steps:

1. Solve the approximated local search problem for the current initial solution $\mathbf{x}^{(c)}$ and trust region parameter ϵ to obtain new trial solution $\mathbf{x}^{(t)}$;

2. Test the trial point and trust region parameter to decide whether or not to accept the trial solution, the trust region parameter, or both. At least one of $\mathbf{x}^{(c)}$ or ϵ changes in this step.

Trust region algorithms differ in how these steps are implemented. The important point here is that we can adapt these rules to the context of the general approximation-based local search operator we are describing in this Chapter.

The test used in trust region update is centered on how well the local approximations represent the real functions inside the trust region. In measuring this, original trust region methods consider only the objective function with no constraints, because they were developed for unconstrained optimization. Nonetheless, we can extend this concept by using penalties to build an unconstrained cost function equivalent to the auxiliary constrained problem being solved by the SQP method. Remember that in the multi-objective case, the local search problem is transformed into the mono-objective constrained problem (3.21). This penalized cost function is defined as $\kappa(\mathbf{x})$ when the real function values are considered, and $\tilde{\kappa}(\mathbf{x})$ when the approximated values are considered.

The following merit figures are defined:

$$\eta = \frac{\kappa(\mathbf{x}^{(c)}) - \kappa(\mathbf{x}^{(t)})}{\tilde{\kappa}(\mathbf{x}^{(c)}) - \tilde{\kappa}(\mathbf{x}^{(t)})} \tag{3.25}$$

We introduce the following control parameters:

$$0 < \mu_{low} < \mu_{high} < 1 \text{ and } 0 < \omega_{down} < 1 < \omega_{up}$$

These parameters are used to define the control rules for the trust region update, as illustrated in Algorithm 3.2. Typical values for μ_{low} and μ_{high} are repectively 0.25 and 0.75. Typical values for ω_{down} and ω_{up} are respectively 0.5 and 2. These typical values are adopted in this work.

Algorithm 3.2: Local search operator with trust region update

$\mathcal{L} \leftarrow$ `Build local data`$(\mathcal{V}(\mathbf{x}^{(c)}, \epsilon))$;
$\tilde{\mathbf{f}}, \tilde{\mathbf{g}}, \tilde{\mathbf{h}} \leftarrow$ `Generate local approximations`$(\mathcal{L})$;
Solve auxiliary problem starting from $\mathbf{x}^{(c)}$ and obtaining trial solution $\mathbf{x}^{(t)}$;
for $i = 1, \ldots, n_{TR}$ **do**
  Evaluate the real functions and the local approximations at $\mathbf{x}^{(t)}$;
  Compute the merit figure η;
  if $\eta < 0$ **then** `/* x(t) is rejected and ϵ decreases */`
    $\epsilon \leftarrow \omega_{down}\epsilon$;
  else if $0 < \eta < \mu_{low}$ **then** `/* x(t) is accepted and ϵ decreases */`
    $\epsilon \leftarrow \omega_{down}\epsilon$;
    $\mathbf{x}^{(c)} \leftarrow \mathbf{x}^{(t)}$;
  else if $\mu_{low} < \eta < \mu_{high}$ **then** `/* x(t) is accepted and ϵ remains the same */`
    $\mathbf{x}^{(c)} \leftarrow \mathbf{x}^{(t)}$;
  else `/* x(t) is accepted and ϵ may increase */`
    if $\mathbf{x}^{(t)} \in \partial\mathcal{V}(\mathbf{x}^{(c)}, \epsilon)$ **then** $\epsilon \leftarrow \omega_{up}\epsilon$;
    $\mathbf{x}^{(c)} \leftarrow \mathbf{x}^{(t)}$;
  $\mathcal{L} \leftarrow$ `Build local data`$(\mathcal{V}(\mathbf{x}^{(c)}, \epsilon))$;
  $\tilde{\mathbf{f}}, \tilde{\mathbf{g}}, \tilde{\mathbf{h}} \leftarrow$ `Generate local approximations`$(\mathcal{L})$;
  Solve auxiliary problem starting from $\mathbf{x}^{(c)}$ and obtaining trial solution $\mathbf{x}^{(t)}$;
end
return $\mathbf{x}^{(t)}$;

Some remarks should be made concerning Algorithm 3.2:

The parameter n_{TR} determines the maximum number of trust region updates allowed. If this number is zero, the trial solution is accepted without evaluating the real functions and without any trust region update, because the loop **for** in line 4 is not entered. This approach is more economic, given that the local search operator will consume no additional evaluations. The neighborhood size is fixed during the entire evolutionary process. In the beginning, the approximations will not be very accurate and the local search can provide non-useful solutions. However, in subsequent generations, when more samples are available, the approximations will improve specially because it is more likely that the local approximations will be built in the same regions.

If $n_{TR} > 0$ at least one trust region update will occur. This local search is more accurate, but will consume n_{TR} evaluations of the real functions. If the accuracy of the approximations is poor, the trial solution is not accepted and the neighborhood size is reduced. The trial solution is accepted whenever $\eta > 0$, but the neighborhood size is reduced if $0 < \eta < \mu_{low}$, or stays the same if $\mu_{low} < \eta < \mu_{high}$. If $\eta > \mu_{high}$ and the trial solution happens to be in the border of the neighborhood region, it means that our local approximations are very accurate and the neighborhood size is restricting the improvement that we can obtain by local search. Thus, we can increase ϵ in this situation.

The trial solution is added to the data set, since it was evaluated anyway. New local approximations are generated with the local data centered at $\mathbf{x}^{(c)}$ and a new trial solution is obtained.

3.4 Hybridizing with evolutionary algorithms

This section addresses the hybridization of the proposed approximation-based local search and the framework presented at the end of Chapter 2. For this purpose, we use Algorithm 2.19, which is representative of the framework. We also define the following parameters:

- the number of consecutive generations in which the local search is not applied, denoted by $n_L \geq 0$. For example, if $n_L = 0$, the local search operator is applied at every generation. If $n_L = 4$, the local search operator is applied at every four generations.

- the number of individuals in the population that will be subject to local search, denoted by $0 \leq \sigma \leq \mu$.
- the maximum number of points in the local data set, denoted by N_L. This parameter will be used in the computational complexity analysis in Chapter 4.

Algorithm 3.3: Unified population-based evolutionary algorithm with local search.

Data: population size μ, offspring size λ, maximum archive size ξ, search space $\mathcal{X}$, objective and constraint functions $\mathbf{f}(\cdot)$, $\mathbf{g}(\cdot)$, $\mathbf{h}(\cdot)$.
Result: Estimate(s) of $\mathcal{X}^*$ in the archive population $A(n)$.

```
1  P(n=0) = {p^(1), ..., p^(μ)} ← Initialize population(μ, X);
2  A(n=0) = ∅ ← Initialize archive /* Stores the best solution set  */
3  ϒ_P(n) ← Penalty(P(n), f, g, h);
4  Φ_P(n) ← Fitness(ϒ_P(n));
5  A(n=0) ← Update(A(n),P(n),ξ);
   // Reproduction comprises selection and variation
6  Q(n=0) = {q^(1), ..., q^(λ)} ← Reproduction(P(n), A(n), Φ_P(n));
7  while ¬ stop criteria do
8  |  ϒ_Q(n) ← Penalty(Q(n), f, g, h);
9  |  Φ_Q(n) ← Fitness(ϒ_Q(n));
10 |  if  mod (n, n_L) = 0 then
11 |  |  for each of the σ best individuals do
12 |  |  |  Local search operator /* see Algorithm 3.2                    */
13 |  |  end
14 |  end
15 |  P(n+1) ← Substitution(P(n), Q(n));
16 |  Q(n+1) ← Reproduction(P(n+1));
17 |  A(n+1) ← Update(A(n),P(n) ∪ Q(n),ξ);
18 |  n ← n+1;
19 end
```

The local search is applied immediately after the evaluation step and before the reproduction step. Algorithm 3.3 shows the Algorithm 2.19 with the local search operator described in Algorithm 3.2. Lines 10 to 14 in Algorithm 3.3 use the parameters n_L and σ to decide whether or nor the local search will be applied. Of course, as discussed at the end of Chapter 2, the local search can be integrated within the reproduction step. In this case, lines 10 to 14 in Algorithm 3.3 could be embedded within the local search operator, which would be called before the selection and variation operators.

Observe that the scheme in Algorithm 3.3 as well as the local search operator in Algorithm 3.2 are valid for mono and multi-objective problems.

3.5 Conclusion

This Chapter presented the approaches for hybridizing local search operators and evolutionary algorithms in general. The approximation-based local search, the main proposition of this thesis, was then introduced. The traditional strategy of starting a local search method after a complete execution of the evolutionary algorithm was referred as the classical approach. The Lamarckian and Baldwinian approaches can be considered in memetic algorithms. In the Baldwinian approach, the individual evolution affects only the fitness value of the individual, while in the Lamarckian approach genetic assimilation occurs; that is, the new solution is incorporated by the individual in its representation. In the context of optimization with real variables, the Lamarckian approach is used more often. The modification of the fitness value biases the population to the best regions, and eventually, the locally optimal solutions will be genetically assimilated by the population. Although more reasonable from a biological perspective, the Baldwinian approach introduces some delay due to the genetic assimilation. Conversely, the Lamarckian approach does not present this delay, but may reduce diversity, especially if the local search is applied to many individuals. In Baldwinian hybrid algorithms, only the standard genetic operators influence diversity, particularly the selective pressure in the stochastic selection operator. From a practical point of view, and considering the problems of interest in this work, the local search operator is applied only to a small fraction of the population. Therefore, the effect in diversity is not as relevant and the Lamarckian approach is adopted.

The local evolution can be achieved in an indirect way, by using a local representation of the problem, based on the knowledge acquired by the algorithm along successive generations. This indirect local search via local approximations reduces the computational cost associated to the exploitation component of memetic algorithms. However, it is important to analyze the theoretical effects of the proposed local search in the global convergence properties of evolutionary algorithms. This issue is addressed in the next Chapter. We will also discuss the computional cost of the proposed methodology.

Chapter 4

Analysis of Memetic Algorithms

"Since the mathematicians have invaded the theory of relativity, I do not understand it myself anymore."

— Albert Einstein, 1879–1955

"Mathematics consists of proving the most obvious thing in the least obvious way."

— George Pólya, 1887–1985

4.1 Preview

Evolutionary algorithms rapidly evolved since their appearance, showing that they are interesting and powerful tools for problem-solving. Albeit successful in many practical applications and empirical experiments, the mathematical analysis of evolutionary algorithms is still limited. The formal analysis of evolutionary algorithms (and stochastic population-based algorithms in general) is still in its infancy. Many problems in this area are still open to the community, and reseachers recognize the need of more research effort in this direction.

Despite these limitations, an interesting tool for analyzing evolutionary algorithms is the Markov chain theory. It comes from the fact that evolutionary algorithms can be viewed as a stochastic process, in which the population represents the current state of

the process and the operations of the algorithm determine the transition probabilities among states.

Chapter 3 discussed memetic algorithms in general and presented the approximation-based local search methodology to be coupled with the evolutionary algorithms presented in Chapter 2. In this Chapter we employ the Markov chain theory to analyze memetic algorithms, with particular attention to the effect of the local search operators on global convergence properties, for instance. The Chapter concludes with the computational cost analysis of memetic algorithms, especially those employing the local search methodology proposed in the previous Chapter for the context of CAD problems.

4.2 Convergence analysis

Evolutionary algorithms work on a set of solutions, called a population, by iteratively applying a sequence of operators with stochastic characteristics. The population of the next generation depends only on the current one, thus we can state:

$$P(n+1) \leftarrow \mathfrak{M}\{P(n)\} \tag{4.1}$$

where $\mathfrak{M}\{\cdot\}$ represents the mapping performed by the algorithm.

Since the mapping performed by $\mathfrak{M}\{\cdot\}$ is a stochastic process and it depends only on the current state, we can use Markov Chains (Iosifescu 1980, Norris 1997) to analyze the algorithm. In the next section, we introduce some useful definitions that will be important for analyzing memetic algorithms.

4.2.1 Preliminary definitions

Let $\mathcal{S}$ be a finite set of states with cardinality $|\mathcal{S}|$ and $\{\mathsf{X}_n \in \mathcal{S} : n \in \mathbb{N}\}$ a random sequence or stochastic process. X_n is a Markov chain if it is a stochastic process with the following property:

$$\mathscr{P}\{\mathsf{X}_{n+1} = \mathsf{s}_{n+1} | \mathsf{X}_n = \mathsf{s}_n, \ldots, \mathsf{X}_0 = \mathsf{s}_0\} = \mathscr{P}\{\mathsf{X}_{n+1} = \mathsf{s}_{n+1} | \mathsf{X}_n = \mathsf{s}_n\} \tag{4.2}$$

that is, the future state depends only on the present state, and it is independent of the past states. Observe that the sequence of populations of an evolutionary algorithm falls

in this definition, since the transitions from one population to another is stochastic and independent of the previous populations.

Therefore, we can write:

$$\mathscr{P}\{\mathsf{X}_{n+1} = \mathsf{s}_j | \mathsf{X}_n = \mathsf{s}_i\} = \tau_{ij} \tag{4.3}$$

with $i, j \in \{1, \ldots, |\mathcal{S}|\}$ and $\mathsf{s}_i, \mathsf{s}_j \in \mathcal{S}$, where $|\mathcal{S}|$ is the number of states. Since the state space is finite, we can conveniently represent the transition probabilities in a $|\mathcal{S}| \times |\mathcal{S}|$ matrix:

$$\mathbf{T} = \begin{bmatrix} \tau_{11} & \cdots & \tau_{1|\mathcal{S}|} \\ \vdots & \ddots & \vdots \\ \tau_{|\mathcal{S}|1} & \cdots & \tau_{|\mathcal{S}||\mathcal{S}|} \end{bmatrix} \tag{4.4}$$

Each entry of the transition matrix $\mathbf{T}$ gives the probability that the next state is s_j given that the current state is s_i. In addition, $\mathbf{T}$ is a double stochastic matrix, since it has the following properties:

$$\sum_{i=1}^{|\mathcal{S}|} \tau_{ij} = \sum_{j=1}^{|\mathcal{S}|} \tau_{ij} = 1 \tag{4.5}$$

i.e., the sum of all elements in a row and the sum of all elements in a column is equal to one.

Given the probability distribution for the states at time step n, we can obtain the probability distribution for the next step by:

$$\boldsymbol{\pi}(n+1) = \boldsymbol{\pi}(n)\mathbf{T} \tag{4.6}$$

and

$$\boldsymbol{\pi}(n) = \boldsymbol{\pi}(0) \underbrace{\mathbf{T} \times \cdots \times \mathbf{T}}_{n \text{ times}} = \boldsymbol{\pi}(0)\mathbf{T}^n \tag{4.7}$$

where $\pi_i(n) = \mathscr{P}\{\mathsf{X}_n = \mathsf{s}_i\}$, $i \in \{1, \ldots, |\mathcal{S}|\}$.

Now, we need to define the state space for a general population-based algorithm. The state of an algorithm in the time step n represents the population at time n. The

operations in the mapping $\mathfrak{M}$ define its transition matrix $\mathbf{T}$. The state space $\mathcal{S}$ is the set of all possible populations with size μ for the search space $\mathcal{X}$.

If we adopt a binary representation, the search space is discrete and finite. The state is a binary string with μdL bits, in which d is the number of variables, and L is the number of bits used for each variable. With dL bits (each individual), we can represent 2^{dL} different elements. The total number of populations is the number of different combinations of 2^{dL} elements in groups of μ. Using simple combinatorial analysis, we find that the total number of populations is:

$$|\mathcal{S}| = \binom{2^{dL} + \mu - 1}{\mu} = \frac{(2^{dL} + \mu - 1)!}{(2^{dL} - 1)!\mu!} \tag{4.8}$$

If we use a real representation of the variables, the search space is continuous, infinite and limited (assuming that inferior and superior limits for the variables exist). In this case, the probability that a given individual $\mathbf{p}^{(i)}$ in the population is equal to any point $\mathbf{x} \in \mathcal{X}$ is zero, but the probability that it lies within a small region $\nu(\mathbf{x}, \epsilon)$ can be defined:

$$\mathscr{P}\{\mathbf{p}^{(i)} \in \nu(\mathbf{x}, \epsilon)\} \geq 0, \quad \mathbf{p}^{(i)} \in P(n) \tag{4.9}$$

We can discretize the search space by considering an arbitrary grid of small hypercubes with size ϵ, in such a way that the probability that an individual lies within each hypercube can be defined[1]. The number of states (total number of populations) for this case is:

$$|\mathcal{S}| = \frac{(r^d + \mu - 1)!}{(r^d - 1)!\mu!} \tag{4.10}$$

Thus, under these considerations, the analysis developed here can be equally applied to evolutionary algorithms using real coding.

The canonical algorithms presented in Chapter 2 also have an archive population $A(n)$, with size $\xi \geq 1$. Considering this population, the state representation changes to:

$$\mathsf{s}_i = \left(\mathsf{s}_i^A; \mathsf{s}_i^P\right) \tag{4.11}$$

where s_i^A is the part of the state associated to $A(n)$, and s_i^P is the part associated to

[1]Of course, in digital computers, where some 64 bit representation of the real numbers is used, the search space will always be finite.

$P(n)$.

The number of states also changes to:

$$|\mathcal{S}| = \frac{(2^{dL} + \mu + \xi - 1)!}{(2^{dL} - 1)!(\mu + \xi)!} \tag{4.12}$$

for binary coding, and to:

$$|\mathcal{S}| = \frac{(r^{d} + \mu + \xi - 1)!}{(r^{d} - 1)!(\mu + \xi)!} \tag{4.13}$$

for real coding.

After the archive update function, we can say that the individuals represented by s_i^A are the best individuals in the state s_i.

Before proceeding, we need to give more useful definitions:

Definition 4.1. A state s_j is said to be accessible from the state s_i if $\exists k < \infty$ such that: $\mathscr{P}\{\mathsf{X}_{n+k} = \mathsf{s}_j | \mathsf{X}_n = \mathsf{s}_i\} = \tau_{ij}^{(k)} > 0$, where $\tau_{ij}^{(k)}$ is the element ij of the matrix $\mathbf{T}^k$, and we write $\mathsf{s}_i \rightarrow \mathsf{s}_j$.

Definition 4.2. A state s_j is said to be a communicating state with the state s_i if $\mathsf{s}_i \rightarrow \mathsf{s}_j$ and $\mathsf{s}_j \rightarrow \mathsf{s}_i$, and we write $\mathsf{s}_i \leftrightarrow \mathsf{s}_j$.

Definition 4.3. A Markov chain $\{\mathsf{X}_n \in \mathcal{S} : n \in \mathbb{N}\}$ is irreducible if its state space is a communicating class, that is, all its states are communicating states.

The last definition means that it is possible to get to any state from any state in an irreducible Markov chain. Therefore, every state will be visited in finite time regardless of the initial state.

Due to the use of the archive population $A(n)$, some states are not communicating states, because of elitism. Thus, states containing solutions in $A(n)$ that are poorer than those represented by the current state are not accessible from it. These intermediate states in the search process are called transient states, since $\mathsf{s}_i \rightarrow \mathsf{s}_j$, but $\mathsf{s}_j \not\rightarrow \mathsf{s}_i$.

We will adopt the following notation:

Definition 4.4. Those states that represent an $A(n)$ whose elements belong to $\mathcal{X}^*$ are called essential states. They form the set $\mathcal{E}$. Those states that represent an $A(n)$ whose elements do not belong to $\mathcal{X}^*$ are called inessential states, hence they are intermediate states. They form the set $\mathcal{I}$.

4.2.2 Convergence proof of standard evolutionary algorithms

Let $\mathcal{X}^*$ be the set of optimal solutions for the optimization problem. It can represent:

1. all the global optima if there are more than one;
2. all the global and local optima (if the algorithm is designed to find them);
3. or all global Pareto-optimal solutions in a multi-objective context.

We say that the algorithm is globally convergent if:

$$\lim_{n\to\infty} \mathscr{P}\{\mathsf{X}_n^A \subseteq \mathcal{X}^*\} = 1 \tag{4.14}$$

in other words, if the probability that the archive population $A(n)$ has converged to a subset of the optimal solution set is equal to one when time goes to infinity. Therefore, we say that the online population $P(n)$ *locates* the optimal solution, while the offline population $A(n)$ *converges* to the optimal solution.

Observe that this criterion is very general. It is sufficient for the single objective context, even if there are several global optima. However, when the archive size is greater than 1, (either in a mono-objective or a multi-objective scenario), this criterion does not say anything about the size or the quality of the solutions in $A(n)$. For instance, if the archive size is 2, and the algorithm finds two Pareto-optimal solutions, it has converged based on this criterion, even though these two solutions are close to each other or they do not represent the $\mathcal{X}^*$ very well. As a consequence, there are many different states s_i^A that satisfy this criterion and therefore are considered as solutions to the problem. Of course, given some additional criteria, some of these states can be considered "better" than the others. The archive size is an user-defined parameter, and the update function should account for the diversity in $A(n)$.

For multi-objective problems, we assume that $|\mathcal{Y}^*| > \xi$, i.e., the cardinality of the Pareto set is greater than the maximum size of $A(n)$, which is almost always true in optimization problems with real variables, given that mathematically $|\mathcal{Y}^*| = \infty$, except in degenerate cases. After the discretization of the search space, we have $|\mathcal{Y}^*| < \infty$, but we still assume that its cardinality is greater than ξ. As a consequence, $A(n)$ has only some elements of $\mathcal{Y}^*$.

In order to prove that a given algorithm is globally convergent under this criterion, we will make use of the following Lemma from Markov chain theory (Iosifescu 1980):

Lemma 4.1. *A homogeneous Markov chain with finite state space and irreducible transition matrix visits every state infinitely often with probability one regardless of the initial distribution.*

Based on this Lemma, all we need to do is to prove that the transition matrix associated with the algorithm $\mathfrak{M}$ is irreducible. If so, we guarantee that all states, including those related to $\mathcal{X}^*$, will be visited in finite time. Therefore, given that the best solutions are always stored in $A(n)$, the algorithm will converge under the criterion defined above.

Due to the distinction between essential and inessential states, we can write the transition matrix in the following canonical form:

$$\mathbf{T} = \begin{pmatrix} \mathbf{P} & \mathbf{0} \\ \mathbf{R} & \mathbf{Q} \end{pmatrix} \tag{4.15}$$

where $\mathbf{P}$ is the transition matrix associated with the essential states, $\mathbf{R}$ is the transition matrix from inessential states to essential ones, and $\mathbf{Q}$ is the transition matrix between inessential states. This canonical form can be obtained by simple reordering of the states. Since the inessential states are not accessible from any essential state, this canonical form exists.

We want to analyze the behavior of this matrix in the limit $n \to \infty$. We have:

$$\mathbf{T}^n = \begin{pmatrix} \mathbf{P}^n & \mathbf{0} \\ \mathbf{R}^n & \mathbf{Q}^n \end{pmatrix} \tag{4.16}$$

with:

$$\mathbf{P}^n = \mathbf{P}\mathbf{P}^{n-1} \tag{4.17}$$

$$\mathbf{Q}^n = \mathbf{Q}\mathbf{Q}^{n-1} \tag{4.18}$$

$$\mathbf{R}^n = \mathbf{R}^{n-1}\mathbf{P}^{n-1} + \mathbf{Q}^{n-1}\mathbf{R}^{n-1} \tag{4.19}$$

Theorem 4.2. *Let* $\mathbf{Q}$ *be the transition matrix associated with the inessential states. It is possible to show that (Seneta 1981):*

$$\lim_{n\to\infty} \mathbf{Q}^n \to 0 \tag{4.20}$$

Proof. Since the matrix is stochastic, from any state $\mathsf{s}_i \in \mathcal{I}$, we have:

$$\sum_{\mathsf{s}_j \in \mathcal{I}} \tau_{ij}^{(n)} + \sum_{\mathsf{s}_j \in \mathcal{E}} \tau_{ij}^{(n)} = 1 \Rightarrow 1 - \sum_{\mathsf{s}_j \in \mathcal{I}} \tau_{ij}^{(n)} = \sum_{\mathsf{s}_j \in \mathcal{E}} \tau_{ij}^{(n)} > 0$$

because $\mathsf{s}_i \rightarrow \mathsf{s}_j$, if $\mathsf{s}_j \in \mathcal{E}$. Thus the probability of going to another inessential state is smaller than one:

$$\sum_{\mathsf{s}_j \in \mathcal{I}} \tau_{ij}^{(n)} < 1$$

Moreover,

$$\sum_{\mathsf{s}_j \in \mathcal{I}} \tau_{ij}^{(n+1)} = \sum_{\mathsf{s}_j \in \mathcal{I}} \sum_{\mathsf{s}_r \in \mathcal{I}} \tau_{ir}^{(n)} \tau_{rj} \leq \sum_{\mathsf{s}_r \in \mathcal{I}} \tau_{ir}^{(n)}$$

thus $\sum_{\mathsf{s}_j \in \mathcal{I}} \tau_{ij}^{(n)}$ is non-increasing with n. We can write:

$$\sum_{\mathsf{s}_j \in \mathcal{I}} \tau_{ij}^{(n)} < \theta < 1, \quad \forall \mathsf{s}_i \in \mathcal{I}$$

Also:

$$\begin{aligned} \sum_{\mathsf{s}_j \in \mathcal{I}} \tau_{ij}^{(mn+n)} &= \sum_{\mathsf{s}_r \in \mathcal{I}} \tau_{ir}^{(mn)} \sum_{\mathsf{s}_j \in \mathcal{I}} \tau_{rj}^{(n)} \\ &\leq \theta \sum_{\mathsf{s}_r \in \mathcal{I}} \tau_{ir}^{(mn)} \end{aligned}$$

Hence,

$$\sum_{\mathsf{s}_j \in \mathcal{I}} \tau_{ij}^{n(m+1)} \leq \theta \sum_{\mathsf{s}_r \in \mathcal{I}} \tau_{ir}^{(mn)} \leq \theta^{m+1}$$

and $\theta^{m+1} \rightarrow 0$ as $m \rightarrow \infty$.

Since this subsequence of $\sum_{\mathsf{s}_j \in \mathcal{I}} \tau_{ij}^{(n)}$ approaches zero geometrically fast with m, the entire sequence also approaches zero, because, as demonstrated before, it is positive and non-increasing with n.

The sequence $\sum_{\mathsf{s}_j \in \mathcal{I}} \tau_{ij}^{(n)}$ is associated with the ith row of the matrix $\mathbf{Q}$ and it is valid

for all rows. Thus, in matricial form, we have:

$$\mathbf{Q}^n \mathbf{u} \to 0$$

as $n \to \infty$, where $\mathbf{u}$ is a column vector with 1's in all positions. It means that:

$$\mathbf{Q}^n \to 0$$

as $n \to \infty$.

□

As an immediate result of this theorem, we have:

$$\mathscr{P}\{\mathsf{X}_n \in \mathcal{I}\} \to 0, \text{ when } n \to \infty. \tag{4.21}$$

This can be easily shown by considering an arbitrary initial probability distribution for the states:

$$\boldsymbol{\pi}(0) = \begin{pmatrix} \boldsymbol{\pi}_{\mathcal{E}}(0) & \boldsymbol{\pi}_{\mathcal{I}}(0) \end{pmatrix} \tag{4.22}$$

and calculating the probability distribution at time n:

$$\boldsymbol{\pi}(n) = \begin{pmatrix} \boldsymbol{\pi}_{\mathcal{E}}(0) & \boldsymbol{\pi}_{\mathcal{I}}(0) \end{pmatrix} \begin{pmatrix} \mathbf{P}^n & \mathbf{0} \\ \mathbf{R}^n & \mathbf{Q}^n \end{pmatrix} = \begin{pmatrix} \boldsymbol{\pi}_{\mathcal{E}}(0)\mathbf{P}^n + \boldsymbol{\pi}_{\mathcal{I}}(0)\mathbf{R}^n & \boldsymbol{\pi}_{\mathcal{I}}(0)\mathbf{Q}^n \end{pmatrix} \tag{4.23}$$

Thus:

$$\lim_{n\to\infty} \boldsymbol{\pi}_{\mathcal{I}}(n) = \lim_{n\to\infty} \boldsymbol{\pi}_{\mathcal{I}}(0)\mathbf{Q}^n = 0 \tag{4.24}$$

regardless of the initial probability distribution.

Since all essential states represent archive populations that are optimal, whose meaning depends on the context, the algorithm will finally converge to one of the essential states:

$$\mathscr{P}\{\mathsf{X}_n^A \subseteq \mathcal{X}^*\} = \mathscr{P}\{\mathsf{X}_n \in \mathcal{E}\} = 1 - \mathscr{P}\{\mathsf{X}_n \in \mathcal{I}\} \tag{4.25}$$

which becomes equal to one, when $n \to \infty$.

Nevertheless, there is one final step to complete the global convergence analysis. The proof before assumed that all essential states are accessible from any inessential state. So, to improve the solutions in $A(n)$, the following transition must be possible:

$$\left(\mathsf{s}_i^A; \mathsf{s}_i^P\right) \rightarrow \left(\mathsf{s}_i^A; \mathsf{s}_j^P\right) \rightarrow \left(\mathsf{s}_j^A; \mathsf{s}_j^P\right) \tag{4.26}$$

where s_j^P represents a population with better solutions than those in s_i^A.

Observe that the archive population does not undergo the selection and variation steps, and the last transition is performed by the update operator. Therefore, s_j^P must be accessible from s_i^P, in order to validate the complete transition. Thus, although the complete transition matrix is not irreducible, the transition matrix associated with the online population $P(n)$ must be irreducible, in order to guarantee that the online population will visit all states in the search space with finite time.

Let $\mathbf{G}$ be the transition matrix for $P(n)$. It is a function of the operators of the algorithm during one iteration, i.e., it is obtained by the product of transition matrices associated with the selection and variation steps (crossover and mutation in the case of genetic algorithms, for instance). Since we will analyze the product of stochastic matrices, it is important to provide some helpful definitions and properties.

Definition 4.5. A matrix $\mathbf{A}$ is said to be positive[2] if $a_{ij} > 0$, $\forall i, j$.

Definition 4.6. A matrix $\mathbf{A}$ is said to be non-negative if $a_{ij} \geq 0$, $\forall i, j$.

Definition 4.7. A square matrix $\mathbf{A}$ is said to be diagonal positive if all elements of its diagonal are positive.

Definition 4.8. A stochastic square matrix $\mathbf{A}$ representing the transition probabilities of a Markov chain process with state space $\mathcal{S}$ is said to be irreducible if $\forall \mathsf{s}_i, \mathsf{s}_j \in \mathcal{S}$, $\exists n \in \mathbb{N}$ such that $a_{ij}^{(n)} > 0$.

A positive matrix $\mathbf{A}$ is hence irreducible since it satisfies $a_{ij}^{(n)} > 0$, with $n = 1$. The converse is not necessarily true.

Definition 4.9. A matrix is said to be column-allowable (row-allowable) if each column (row) contains at least one positive entry. A matrix that is both column-allowable and row-allowable is an allowable matrix.

[2]Do not confuse with the concept of positive (or negative) definiteness

From the definition above, we see that a diagonal positive matrix is both column-allowable and row-allowable. Also, the product of two diagonal positive matrices is also a diagonal positive matrix.

Let us consider a standard genetic algorithm, using a selection operator, a crossover operator, and a mutation operator in the evolutionary iteration. In this case, the transition matrix $\mathbf{G}$ is:

$$\mathbf{G} = \mathbf{SCM} \tag{4.27}$$

where $\mathbf{S}$, $\mathbf{C}$, and $\mathbf{M}$ are respectively the transition matrices of selection, crossover, and mutation steps.

The matrices associated with selection and crossover are neither positive nor irreducible matrices. Therefore, the irreducible property should be established by the mutation operator - the global search component of the algorithm.

The roulette wheel and the tournament selections are both diagonal positive, and thus column-allowable, since there is a positive probability that $P(n)$ stay unchanged after application of these operators. For example, the probability of selecting an individual $\mathbf{p}^{(i)} \in P(n)$ is:

$$\mathscr{P}\{\text{selecting } \mathbf{p}^{(i)} \in P(n)\} = \frac{\phi(\mathbf{p}^{(i)})}{\sum_{i=1}^{\mu} \phi(\mathbf{p}^{(i)})} > 0 \tag{4.28}$$

and the probability that the population is the same after selection is:

$$\mathscr{P}\{S(n) \text{ be the same as } P(n)\} = \prod_{i=1}^{\mu} \mathscr{P}\{\text{selecting } \mathbf{p}^{(i)} \in P(n)\} > 0 \tag{4.29}$$

In the standard tournament selection, T individuals are randomly selected from $P(n)$, and that one with the highest fitness value is selected to $S(n)$. In order to see that this selection operator is also diagonal positive, we just need to prove that the probability that $P(n)$ stay unchanged after selection is positive. The probability of selecting $\mathbf{p}^{(i)}$ is:

$$\mathscr{P}\{\text{selecting } \mathbf{p}^{(i)} \in P(n) \text{ to tournament}\} = \frac{1}{\mu} \tag{4.30}$$

$$\mathscr{P}\{\text{selecting } \mathbf{p}^{(i)} \in P(n)\} \geq \left(\frac{1}{\mu}\right)^{T} > 0 \tag{4.31}$$

and again:

$$\mathscr{P}\{S(n) \text{ be the same as } P(n)\} = \prod_{i=1}^{\mu} \mathscr{P}\{\text{selecting } \mathbf{p}^{(i)} \in P(n)\} > 0 \tag{4.32}$$

Notice that, in Chapter 2, we made $\phi(\,\cdot\,) > 0$ and tournament selection with repetition just to guarantee a diagonal positive matrix for selection.

In general, crossover operators also generate diagonal positive transition matrices, since there is a positive probability that the individuals generated after crossover be the same. The product of two diagonal positive matrices is also a diagonal positive matrix, hence **SC** is diagonal positive and column-allowable.

Theorem 4.3. *If* $\mathbf{M}$ *is positive,* $\mathbf{G}$ *is positive and thus irreducible. Therefore, the archive population will converge to the solution set and the algorithm is globally convergent.*

Proof. Let $\mathbf{D} = \mathbf{SC}$ be a stochastic diagonal positive matrix. We have:

$$g_{ij} = \sum_{k} d_{ik} m_{kj} > 0$$

$\forall i, j$. Thus, $\mathbf{G}$ is positive and, following the Definition 4.8, it is irreducible.

□

Thus, when analyzing the global convergence property of an algorithm, we have to analyze the kind of transition matrices that its operators produce.

For instance, using binary coding and independent bitwise mutation, the individual $\mathbf{p}^{(i)} \in P(n)$ and its mutated version $\mathbf{m}^{(i)} \in M(n)$ are represented by two binary strings of size L. The probability of occurrence of this event is:

$$\mathscr{P}\{\mathbf{p}^{(i)} \rightarrow \mathbf{m}^{(i)}\} = \rho_m^{h(\mathbf{p}^{(i)}, \mathbf{m}^{(i)})} (1 - \rho_m)^{1 - h(\mathbf{p}^{(i)}, \mathbf{m}^{(i)})} > 0 \tag{4.33}$$

where $\rho_m > 0$ is the mutation probability, $h(\cdot, \cdot)$ is the Hamming distance of the two binary strings. Thus, the associated transition matrix is positive.

By using real coding, it is also possible to analyze specific mutation operators such as those presented in Chapter 2 to verify if they produce positive transition matrices. In theory, since the Gaussian probability density function has infinite support, the Gaussian mutation would generate a positive transition matrix. However, due to the finite precision of computers the probability of a disturbance value occurring far from the mean value is zero. For small standard deviations (with respect to the search space), the Gaussian mutation can be approximated by a function with compact support. There are mutation operators that employ probability density functions with compact support, likewise the uniform distribution or distributions related to a chaotic system. Their transition matrices are not positive, but they are irreducible. Since there is a positive probability that the selection and crossover operators do not change the individual, then there is a positive probability that any point in the search space be attained in $n > 1$ steps, by consecutive mutations. Hence $\exists n > 0$ such that $\tau_{ij}^{(n)} > 0$ and the transition matrix is irreducible by Definition 4.8. In this case, it is possible to show that if $\mathbf{M}$ is irreducible and the product $\mathbf{SC}$ is diagonal positive, then $\mathbf{G}$ is also an irreducible matrix (Agapie 1998).

In the next section, we analyze how the local search operator affects the transition matrix $\mathbf{G}$, and its convergence properties.

4.3 Analyzing the local search operator

In this section, we need to analyze the case when a local search is explicitly used in the evolutionary cycle. In our canonical hybrid algorithm, the local search is performed before the selection and variation steps. Thus, assuming that the product $\mathbf{SCM}$ is irreducible, we need to analyze what happens with the product:

$$\mathbf{H} = \mathbf{LG} = \mathbf{L}(\mathbf{SCM}) \tag{4.34}$$

where $\mathbf{L}$ is the transition matrix associated with the local search phase, $\mathbf{G}$ is the transition matrix associated with the global search algorithm, and $\mathbf{H}$ is the transition matrix associated with the hybrid algorihm, which results from the global-local search interaction.

4.3.1 Baldwinian approach

First, we consider the Baldwinian approach, which does not change the representation of the solutions in the search space. In this case, the transition matrix $\mathbf{L}$ is an identity matrix, because the populations $L(n)$ and $P(n)$ are the same. However, the transition probabilities in $\mathbf{S}$ will change, since the fitness values are modified by the local search. Nonetheless, the matrix $\mathbf{S}$ will retain its previous properties, specifically, it will still be diagonal positive, as discussed before, so global convergence is not affected by Baldwinian local search.

4.3.2 Lamarckian approach

In the Lamarckian approach, the population is modified by the local search. If the local search operator is stochastic (random local disturbance to the original point), then $\mathbf{L}$ is diagonal positive, because there is a non zero probability that the local search does not modify the individuals.

The algorithm is globally convergent under an argument similar to the one used in Theorem 4.3, leading to the following result:

Theorem 4.4. *If* $\mathbf{SCM}$ *is positive and the local search is stochastic, then* $\mathbf{H}$ *is positive and thus irreducible. Therefore, the archive population will converge to the solution set and the hybrid algorithm is globally convergent.*

Proof. Let the product $\mathbf{SC}$ be a stochastic diagonal positive matrix. Since $\mathbf{L}$ is diagonal positive, the product $\mathbf{D} = \mathbf{LSC}$ is also a diagonal positive matrix. We have:

$$h_{ij} = \sum_k d_{ik} m_{kj} > 0$$

$\forall i, j$. Thus, $\mathbf{H}$ is positive and, following the Definition 4.8, it is irreducible.

□

If the local search operator is deterministic, we can say that $\mathbf{L}$ is at least row-allowable. Based on these characteristics, we can state the following:

Theorem 4.5. *If* **SCM** *is positive and the local search is deterministic,* **L** *is at least row-allowable, then* **H** *is also positive and thus irreducible. Therefore, the archive population will converge to the solution set and the hybrid algorithm is globally convergent.*

Proof. Let $\mathbf{D} = \mathbf{SCM}$. Since $\mathbf{L}$ is row-allowable, there is at least one k such that l_{ik} is positive, then:

$$h_{ij} = \sum_k l_{ik} d_{kj} > 0$$

$\forall i, j$. Thus, $\mathbf{H}$ is positive and, following the Definition 4.8, it is irreducible.

□

However, when $\mathbf{G} = \mathbf{SCM}$ is irreducible, due to a mutation operator that produces a non-positive but irreducible transition matrix, and $\mathbf{L}$ is row-allowable, we can not state that $\mathbf{H}$ is irreducible. Therefore, we can not prove global convergence in general. This situation can be understood with an illustrative example, see Figure 4.1. In this Figure, we consider that the population is concentrated in the region of attraction of a local optimum, which is not the global optimum. Using a mutation with compact support, the population is able to escape from the local minimum in $n > 1$ steps, as already demonstrated. But if the local search is applied to all individuals at every generation, the population will never escape from the local minimum in this situation.

Nevertheless, it is possible to guarantee convergence if the number of individuals selected for local search σ is smaller than the population size μ, as done in the hybridization scheme discussed in Chapter 3. The result is stated in the following theorem:

Theorem 4.6. *If* $\sigma < \mu$, **SCM** *is irreducible and* **L** *is at least row-allowable, then* **H** *is also irreducible (but not positive in general). Therefore, the archive population will converge to the solution set and the hybrid algorithm is globally convergent.*

Proof. In this case, there is a positive probability that the individual(s) not selected for local search is(are) able to escape from the local minimum in $n > 1$ steps due to the irreducible mutation. Therefore, $\exists n$ such that $h_{ij}^{(n)} > 0$ and hence the product $\mathbf{H} = \mathbf{L}(\mathbf{SCM})$ will be irreducible when $\sigma < \mu$.

□

Finally, we need to analyze the situation when the local search operator is not applied at every generation, but at every constant number of generations, let us say, at every

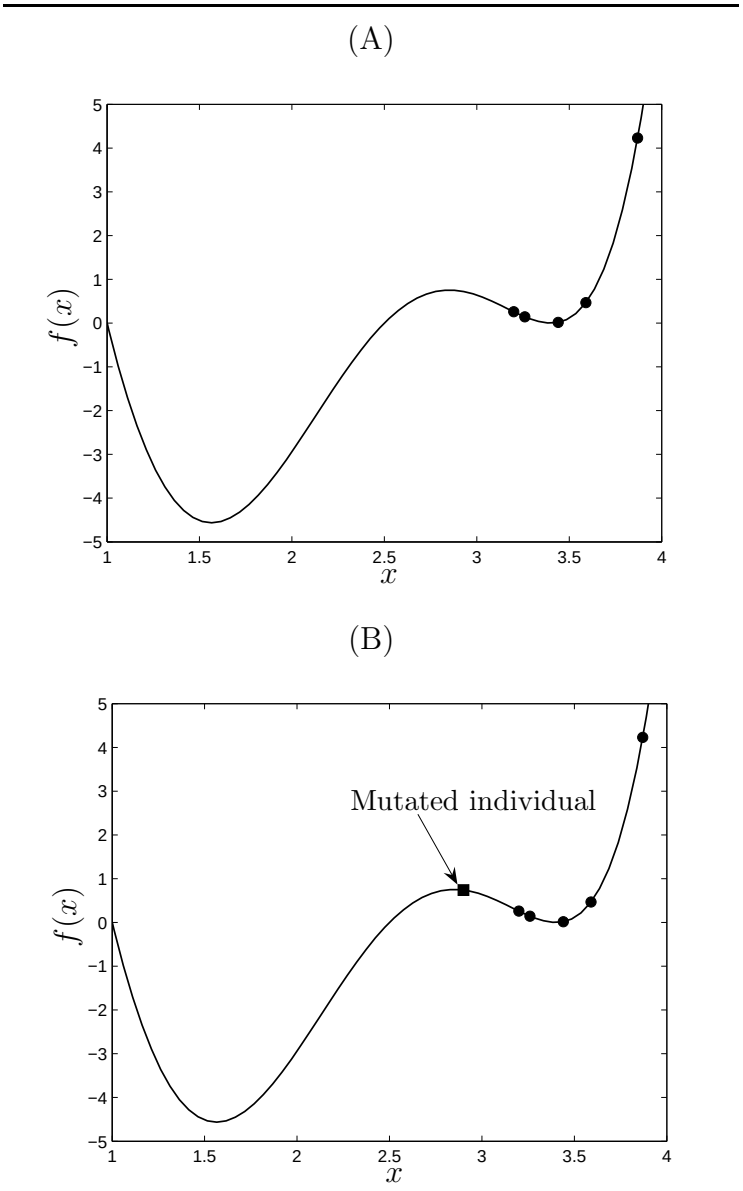

Figure 4.1: (A) The population of an evolutionary algorithm at a given iteration is stuck in the region of attraction of a local minimum. (B) A mutation operator with compact support produces new solutions inside the same region of attraction, as shown in the figure. In the next evolutionary cycle, the local search will concentrate the population, including the mutated individuals, around the local minimum. The population has no chance of escaping the local minimum in this situation.

n_L generations. The transition matrix becomes:

$$\mathbf{H} = \mathbf{L}\underbrace{\mathbf{SCM} \times \mathbf{SCM} \times \cdots \times \mathbf{SCM}}_{n_L \text{ times}} \tag{4.35}$$

then:

$$\mathbf{H}^n = (\mathbf{L}\mathbf{G}^{n_L})^n \tag{4.36}$$

For this situation, we have the following result:

Theorem 4.7. *If $\sigma < \mu$, $\mathbf{G} = \mathbf{SCM}$ is irreducible, $\mathbf{L}$ is at least row-allowable, and the local search is applied at every n_L generations, then $\mathbf{H}$ is also irreducible (but not positive in general). Therefore, the archive population will converge to the solution set and the hybrid algorithm is still globally convergent.*

Proof. The hybrid algorithm using local search at every n_L generations can be seen as a hybrid algorithm with an extended generation, that is, one in which the local search is applied first and then the selection, crossover and mutation operators are applied n_L times. Since $\mathbf{G}$ is irreducible, the sequence of n_L iterations of $\mathbf{G}$ is also irreducible, i.e., $\mathbf{G}^{n_L}$ is irreducible. Consequently, $\mathbf{H} = \mathbf{L}\mathbf{G}^{n_L}$ remains irreducible from Theorem 4.6. □

Therefore, given the considerations above, the hybrid algorithm using an explicit local search phase is also globally convergent as long as the non-hybrid algorithm is globally convergent. The local search phase will not affect this property except in very special cases.

This global convergence analysis by means of Markov chain theory allows us to state only if the algorithm is globally convergent or not under the very general criterion in (4.14). The analysis does not say anything about the convergence rate of the algorithm. For example, the random search algorithm, using uniform sampling in the search space, is globally convergent under the Markov chain analysis, because its transition matrix is positive. As long as we use an archive population to store the best solutions, the random search is globally convergent. Simple enumeration (given that the search space is finite) is also globally convergent. Nonetheless, assessing the global convergence property of an algorithm is an obvious requirement to calculate its convergence rate. Moreover, strategies like the random search and simple enumeration are very inefficient in practice. As discussed, the transition matrices $\mathbf{L}$, $\mathbf{S}$, and $\mathbf{C}$ do not affect the global convergence of the algorithm, which stands on the transition matrix $\mathbf{M}$. This is important, but they may affect its convergence time. As a consequence of the *No Free Lunch* theorems (Wolpert and Macready 1997), discussed in Chapter 2, an evolutionary algorithm can

outperfom the random search for a given class of problems, when the local search and crossover operators exploit some knowledge of that class of problems, or implicitly rely on some common structure of these problems. On the other hand, simple random search does not exploit any structure at all. Evolutionary algorithms are far from being random search methods, because the matrices $\mathbf{L}$, $\mathbf{S}$, and $\mathbf{C}$ can improve the performance of the algorithm in comparison to simple random search for a specific class of problems.

In the next section, we proceed with the Markov chain analysis in order to develop additional conclusions about convergence times.

4.4 Convergence time

Let us define the random variable η_i as the time[3] required for the Markov chain to converge as defined in (4.14) starting from the state $\mathsf{s}_i \in \mathcal{S}$, so:

$$\eta_i = \min_n \{n \geq 0 : \mathsf{X}_n^A \subseteq \mathcal{X}^* \mid \mathsf{X}_0 = \mathsf{s}_i \in \mathcal{S}\} \tag{4.37}$$

Observe that when $\mathsf{s}_i \in \mathcal{E}$, $\eta_i = 0$ by definition. When $\mathsf{s}_i \in \mathcal{I}$, $\eta_i > 0$.

Each realization of the process X_n will give a diferent value for the convergence time. Thus, we need to consider the expectation of this variable, represented by $\mathscr{E}[\eta_i]$. According to the Markov chain theory, we have the following theorem for the mean convergence times:

Theorem 4.8. *Let η_i be the convergence time for a Markov chain $\{\mathsf{X}_n \in \mathcal{S} : n \in \mathbb{N}\}$ when starting from $\mathsf{s}_i \in \mathcal{I}$. All the mean convergence times are given by the vector:*

$$\mathscr{E}[\boldsymbol{\eta}] = (\mathbf{I} - \mathbf{Q})^{-1}\mathbf{u} \tag{4.38}$$

where $\mathbf{Q}$ is the transition matrix between the inessential (transient) states, and $\mathbf{u}$ is a vector of 1's in all positions.

Proof. Let ν_{ij} be the number of times the state $\mathsf{s}_j \in \mathcal{I}$ is visited when starting from

[3]Time here is discrete and actually means the number of iterations

$\mathsf{s}_i \in \mathcal{I}$ until convergence. Thus:

$$\mathscr{E}[\nu_{ij}] = \delta_{ij} + \sum_k Q_{ik}\mathscr{E}[\nu_{kj}]$$

where δ_{ij} is equal to one when $i = j$ and zero otherwise. In matricial form, we have:

$$\mathbf{E} = \mathbf{I} + \mathbf{QE}$$

and:

$$\mathbf{E} = (\mathbf{I} - \mathbf{Q})^{-1}$$

Finally, the mean convergence time for each state is obtained by summing up each row of $\mathbf{E}$:

$$\mathscr{E}[\boldsymbol{\eta}] = (\mathbf{I} - \mathbf{Q})^{-1}\mathbf{u}$$

which completes the proof.

□

Nonetheless, we are interested in the mean convergence time regardless of the initial state, so we can define the following mean convergence time for the algorithm:

$$\langle N \rangle = \sum_{\mathsf{s}_i \in \mathcal{S}} \pi_i(0)\mathscr{E}[\eta_i] = \boldsymbol{\pi}(0)\mathscr{E}[\boldsymbol{\eta}] \tag{4.39}$$

where $\pi_i(0) = \mathscr{P}\{\mathsf{X}_0 = \mathsf{s}_i\}$

It is very difficult to analyze these equations in practice, since the state space is huge, even for small-sized problems and small populations. Nevertheless, it is possible to get some insights on the effect of the local search in hybrid algorithms. As mentioned before, the transition matrix $\mathbf{L}$ (associated with the local search) does not affect the global convergence properties of the algorithm. But this matrix changes the overall transition matrix $\mathbf{H}$ by changing the transition probabilities between states that represent populations with local differences among the individuals. Thus the probability that the individual moves to better regions within a local vicinity is increased. All states reach the "better" neighbour states in less time, including the "optimal" states. It is possible to admit that the mean times $\mathscr{E}[\eta_i]$ will decrease with the employment of local search,

and hence the time defined by $\langle N \rangle$ will also decrease.

Let us try to reach this conclusion with a different argument. For instance, let us make the following assumptions about our optimization problem:

1. The search space $\mathcal{X}$ is Lebesgue measurable;
2. The objective functions $f : \mathcal{X} \mapsto \mathcal{Y}$ are measurable functions.

A Lebesgue measure $m(\mathcal{X})$ of a set $\mathcal{X}$ satisfies:

$$m\left(\{\mathbf{x} : f(\mathbf{x}) \leq \min_{\mathbf{y} \in \mathcal{X}} f(\mathbf{y}) + a\}\right) > 0 \tag{4.40}$$

for any positive number $a > 0$.

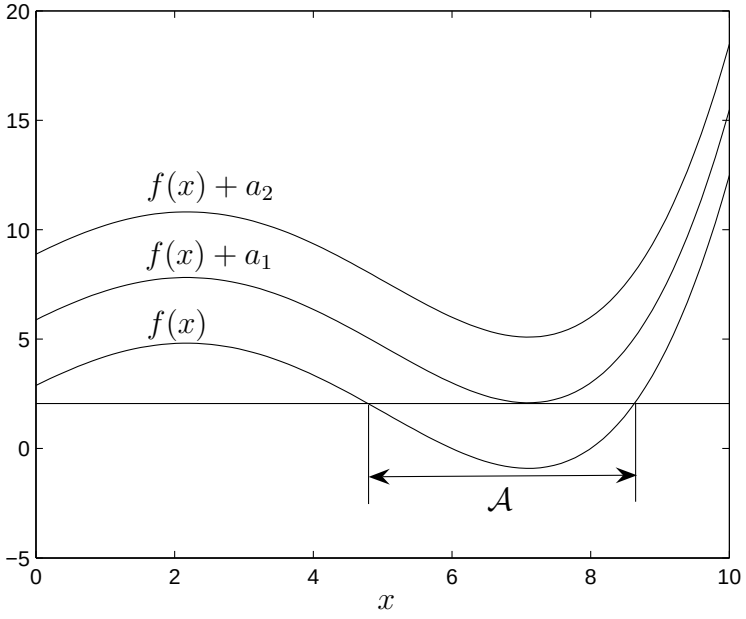

Figure 4.2: Illustration of the Lebesgue measure definition. The set of points $\mathcal{A}$ contains all points x for which $f(x)$ is lesser or equal to the minimum value of $f(x) + a_1$. The set $\mathcal{A}$ has a nonzero "volume" (in this case a length), thus $m(\mathcal{A}) > 0$. For a_2, the set $\mathcal{A}$ is equal to the search space $\mathcal{X}$.

Figure 4.2 illustrates this concept. Observe that for any $a > 0$ in Figure 4.2, we find a set of points with a measurable "volume", that is, a set of points satisfying $m(\mathcal{A}) > 0$. This definition precludes the existence of isolated (and discontinuous) points in the

search space. Isolated points would lead to sets with zero volume for some values of a, i.e., the set:

$$\left\{\mathbf{x} : f(\mathbf{x}) \leq \min_{\mathbf{y}\in\mathcal{X}} f(\mathbf{y}) + a\right\} \tag{4.41}$$

would have only the isolated point.

This definition does not preclude nondifferentiable functions, as long as the nondifferentiability does not lead to isolated points. We remark that the analysis by Markov chains implicitly assumed these characteristics, since the probability of sampling an isolated point is zero.

Given these formal considerations, we now divide the search space $\mathcal{X}$ into N_B basins $\mathcal{B}_i$, $i = 1, \ldots, N_B$. Inside some of these basins lie N_G regions considered the optimal regions, such as:

$$\mathcal{R}_G = \bigcup_{j=1}^{N_G} \mathcal{R}_{G_j} \tag{4.42}$$

Assuming that the functions are Lebesgue measurable functions, we have $m(\mathcal{B}_i) > 0$ and $m(\mathcal{R}_{G_j}) > 0$. Consequently, the probability of achieving any of these basins and any of the optimal regions is greater than zero for an evolutionary algorithm that produces an irreducible transition matrix for the online population.

The probability distribution of the states after application of the selection, crossover, and mutation operators is:

$$\tilde{\boldsymbol{\pi}} = \boldsymbol{\pi}(n)\mathbf{G} \tag{4.43}$$

The probability of achieving $\mathcal{R}_G$ with one iteration of the evolutionary algorithm is:

$$\mathscr{P}_G = \sum_k \tilde{\pi}_k(n), \quad \mathsf{s}_k \text{ has at least one individual in } \mathcal{R}_G \tag{4.44}$$

Individuals in the basins that contain the regions $\mathcal{R}_{G_i}$ can attain the optimal region after one or more iterations of the local searcher. The other basins represent regions of the search space in which the local search does not lead to $\mathcal{R}_G$ or regions in which the

local searcher fails (deceptive regions). The probability of achieving these basins is:

$$\mathscr{P}_{B_i} = \sum_k \tilde{\pi}_k(n), \quad \mathsf{s}_k \text{ has at least one individual in } \mathcal{B}_i \tag{4.45}$$

For the hybrid algorithm, the probability of one successful iteration is:

$$\mathscr{P}_G + \sum_{i:\exists j | \mathcal{R}_{G_j} \subset \mathcal{B}_i} \mathscr{P}_{B_i} \tag{4.46}$$

The probability of nonconvergence in n iterations is:

$$\{1 - \mathscr{P}_G\}^n \tag{4.47}$$

for the standard global search algorithm and:

$$\left\{1 - \left(\mathscr{P}_G + \sum_{i:\exists j | \mathcal{R}_{G_j} \subset \mathcal{B}_i} \mathscr{P}_{B_i}\right)\right\}^n \tag{4.48}$$

for the hybrid algorithm.

Both probabilities naturally go to zero as n goes to infinity. The important point is that the hybrid algorithm has a bigger probability of converging in less iterations, since achieving any of the basins $\mathcal{B}_i$ gives more chance of achieving the optimal regions through local search. Of course, that comes at the price of a more expensive generation. This aspect is discussed in the next section.

4.5 Computational cost

It is important to say that the mean convergence time as defined by $\langle N \rangle$ represents in fact the mean number of steps of the Markov chain, that is, it represents the number of iterations of the evolutionary algorithm, not the physical runtime in a computer (measured in seconds, for example). Consequently, even though the hybrid algorithm converges in less iterations, it is not useful in practice if it takes more time than the conventional one does. For practical purposes, we require that the total optimization time (on average) by using the hybrid algorithm be smaller than the total time (on average) needed when using the non-hybrid algorithm. We need to develop this relation

in detail.

Before proceeding, we turn to computational complexity issues first. The quantity $\langle N \rangle$ is hereafter termed the averaged number of generations. We introduce $\langle T \rangle$ as the averaged total optimization time, which is of more practical interest. A theoretical value for $\langle N \rangle$ was devised previously in terms of the Markov chain theory. With this value, we can estimate the time complexity of an evolutionary algorithm to converge in a given problem as:

$$O(\mu, \lambda, \xi, d)\langle N \rangle = [O_r(\mu, \lambda, \xi, d) + O_e(\mu, \lambda, \xi, d)] \langle N \rangle \tag{4.49}$$

where $O_r(\,\cdot\,)$ is the number of operations in the reproduction step and $O_e(\,\cdot\,)$ is the number of operations in the evaluation step. They are related to parameters of the algorithm and the number of variables d.

The number of operations involved in the algorithms analyzed in Chapter 2 are not difficult to analyze. We briefly present some estimates next.

The roulette wheel selection has time complexity of $O(\mu^2)$, since we need to search the vector of cumulative probabilities (of size μ) μ times to return $S(n)$. The complexity of tournament selection is of $O(T\mu)$, leading to $O(\mu^2)$ if we notice that the tournament size T is a fraction of μ. The complexity of stochastic selection operators is independent of d because they operate only with the fitness values.

The complexity of deterministic selections of evolution strategies (also viewed as substitution operators in Algorithm 2.19) is dominated by the sorting algorithm. Thus, we have $O(\mu \log \mu)$ for ES(μ, λ) and $O((\mu + \lambda) \log(\mu + \lambda))$ for ES$(\mu + \lambda)$.

All recombination operators presented in Chapter 2 for evolution strategies have a complexity of $O(\lambda d)$, as well as the mutation operator. The crossover and mutation operators of genetic algorithms have also complexity of $O(\lambda d)$, or $O(\mu d)$, since in genetic algorithms we have $\lambda = \mu$.

In multi-objective algorithms, the fitness assignment and archive reduction steps are the most expensive steps. In MOGA, the rank classification and fitness sharing schemes are both of $O(m\mu^2)$, where m is the number of objectives. The archive reduction is in the worst case of complexity $O(m\mu(\mu + \xi)^2)$, which occurs when $P(n)$ and $A(n)$ are nondominated and the size of the archive before reduction becomes $\mu + \xi$. However, that does not occur very often. In NSGA2, the complexity of the fast-nondominated

sorting is $O(m\mu^2)$. The complexity of the crowding distance assignment used in the ES$(\mu+\mu)$ substitution is $O(m\mu \log \mu)$. Finally, in SPEA2 the calculation of the strength values and raw fitness requires $O(m(\mu+\xi)^2)$ operations. The worst case complexity of the reduction algorithm is $O(m\mu(\mu+\xi)^2)$, likewise in MOGA, but on average the complexity will be lower than $O(m(\mu+\xi)^2 \log(\mu+\xi))$, as individuals differ with respect to the second or third nearest neighbour (Zitzler, Laumanns and Thiele 2001).

Given these individual complexities, one can estimate the overall complexity of the various algorithms presented in Chapter 2.

We now turn to the complexity of the local search methodology proposed in Chapter 3. Each step in the local search is analyzed separately.

The cost of building the local data set depends on the size of the global data set, which grows every generation. The number of solutions generated by Algorithm 2.19 at generation n is $\mu+\lambda n$. The storage cost of the global data set at generation n is of order of $O((\mu+\lambda n)(m+p+q+d))$, and assuming an average number of generations as $\langle N \rangle$ we get an average storage cost of $O((\mu+\lambda\langle N\rangle)(m+p+q+d))$. The local storage cost is of the order of $O((m+p+q+d)N_L)$ in the worst case, where N_L is the maximum number of points in the local data set $\mathcal{L}$. Therefore, the operation of building the local data has a worst case complexity of $O((\mu+\lambda N_{\max})N_L d)$, where $N_{\max}$ is the maximum number of generations[4].

The cost of the generation of each approximation by the multiquadric interpolation technique is dominated by the assembling of the $\mathbf{Q}$ matrix, which is of $O(dN_L^2)$ because we need to calculate the distance between each pair of centers. Since the matrix is full, the storage cost is of the order of $O(N_L^2)$, in the worst case. Of course, many local approximations are generated with less than N_L points. The solution of the system of equations has also complexity of $O(N_L^2)$. Finally, considering all objective and constraint functions, we get $O((m+p+q)(d+1)N_L^2)$.

In the mono-objective case, the auxiliary problem is solved directly via SQP method. The cost of this step can be roughly estimated as follows: the computation of the gradients of all functions in the auxiliary problem requires $(1+p+q)d$ evaluations of the multiquadric approximations. Nevertheless, as explained in Chapter 3, the gradients can be obtained inexpensively together with the evaluations of the approximations. Thus, we can assume that with $(1+p+q)$ evaluations of the multiquadric approximations we also

[4]Theoretically, $N_{\max}$ would be the maximum value in the vector $\mathscr{E}[\boldsymbol{\eta}]$. In practice, $N_{\max}$ is the maximum number of generations that the user allows the algorithm to run.

obtain all the gradients. The cost of evaluating the multiquadric model is of $O(dN_L)$ in the worst case, so each iteration of the SQP method has complexity of $O((1+p+q)dN_L)$. The BFGS approximation of the Hessian takes at least $d+1$ iterations to converge in a quadratic problem. Therefore, we can estimate a complexity of $O((1+p+q)d(d+1)N_L)$ for solving the local search problem in the mono-objective case.

In the multi-objective case, we need to solve m mono-objective problems to find the vectors that define the cone $\mathcal{K}$ in the objective space. After that, we solve the problem obtained by the goal attainment formulation. Therefore, we can estimate a complexity of $O([m(1+p+q)+(m+p+q)]\,d(d+1)N_L)$ for solving the local search problem in the multi-objective case.

An overal expression for the whole process, including trust region updates, is given by:

$$\begin{aligned} O_{LS} = (1+n_{TR})\left\{O((\mu+\lambda N_{\max})N_L d) + O((m+p+q)(d+1)N_L^2) + \right. \\ \left. + O([(m+p+q)+\text{step}(m-1)m(1+p+q)]\,d(d+1)N_L)\right\} + n_{TR}O_e(d) \end{aligned}$$

where $n_{TR} \geq 0$ is the number of trust region updates, and $O_e(\,\cdot\,)$ is the complexity of the evaluation of the real functions. Additionally:

$$\text{step}(x) = \begin{cases} 0, & \text{if } x \leq 0 \\ 1, & \text{if } x > 0 \end{cases} \tag{4.50}$$

Using the step function, the expression for O_{LS} becomes equally valid for the mono and multi-objective cases.

By inspecting the expression for O_{LS}, we note that the complexity of the proposed approximation-based local search is:

- Linear with the number of individuals;
- Linear with the number of objective and constraint functions;
- Linear with the maximum number of generations;
- Linear with the number of trust region updates;
- Quadratic with the number of variables;

- Quadratic with the maximum number of points in the local data set;

Since N_L is greater than the number of variables d and the number of objective and constraint functions, we can see that the complexity of the local search is dominated by the term $O((m+p+q)(d+1)N_L^2)$. Nonetheless, since $N_{\max}$ and λ are relatively big numbers, the term $O((\mu+\lambda N_{\max})N_L d)$ is also important.

As we can see, the computational complexity of the operations in typical evolutionary algorithms is polynomial with respect to μ, λ, ξ, and d. The computational complexity of the evaluation step depends on the calculations made by the black-box function. The important conclusion here is that the memetic algorithm preserves the polynomial complexity of the original method. The approximation-based local search adds the operations associated with local data assembling and approximation building, which have polynomial complexity, and increase storage requirements, but this increase in storage is linear. Therefore, the essential factor determining the exponential time or not of the optimization process resides in $\langle N \rangle$. This in turn depends strongly on the problem being solved, making some problems more difficult than others.

In this thesis, the approximation-based local search is proposed to deal with expensive black-box functions. The complexity analysis is important but limited because the complexity of the evaluation step is unknown. Some assumptions were discussed in Chapter 3 regarding the employment of approximations in the local search phase. These considerations will be quantified next in order to derive additional conclusions, but first we need to define the following variables:

- $\langle T_G \rangle$ is the averaged total time consumed by the standard algorithm to converge to the solution within a given accuracy;
- $\langle T_H \rangle$ is the averaged total time consumed by the hybrid algorithm to converge to the solution within a given accuracy;
- $\langle N_G \rangle$ is the averaged number of generations required by the standard algorithm to converge, as defined by criterion (4.14);
- $\langle N_H \rangle$ is the averaged number of generations required by the hybrid algorithm to converge, as defined by criterion (4.14);
- t_e is the computational time consumed to evaluate one solution;
- t_s is the computational time consumed by the selection step;

- t_v is the computational time consumed by the variation step;
- t_{LS} is the computational time consumed by the local search of one individual;
- t_{app} is the computational time consumed to generate the local approximations;
- t_{opt} is the computational time consumed to optimize the local approximations;

Based on these definitions and considering the same population size for both algorithms, we have:

$$\langle T_G \rangle = (\mu t_e + t_s + t_v)\langle N_G \rangle \tag{4.51}$$

$$\langle T_H \rangle = (\mu t_e + t_s + t_v + \sigma t_{LS})\langle N_H \rangle \tag{4.52}$$

By imposing the condition $\langle T_H \rangle < \langle T_G \rangle$, we get:

$$\langle N_H \rangle < \langle N_G \rangle \frac{(\mu t_e + t_s + t_v)}{(\mu t_e + t_s + t_v + \sigma t_{LS})} \tag{4.53}$$

Assuming $t_e >> t_s + t_v$, which is usually the case in CAD problems, we have:

$$\langle N_H \rangle < \langle N_G \rangle \frac{\mu t_e}{(\mu t_e + \sigma t_{LS})} \tag{4.54}$$

Based on this relation, we observe that even if the hybrid algorithm converges in less generations (on average) compared to the basic algorithm, $\langle T_H \rangle$ can be greater. The more significant the amount σt_{LS} is when compared to μt_e, the smaller the value $\langle N_H \rangle$ should be in comparison to $\langle N_G \rangle$ to satisfy the initial condition $\langle T_H \rangle < \langle T_G \rangle$.

When the local search is directly applied to the problem, we have:

$$t_{LS} = t_{LS}(t_e) \tag{4.55}$$

that is, the time consumed for local search is a function of the time consumed to evaluate one solution by the real functions. In order to reduce the cost of the exploitation component of memetic algorithms, we propose the utilization of local approximations of the functions involved in the problem.

As mentioned before, evolutionary algorithms can be viewed as adaptive sampling methods. This algorithm performs a number of samples of the search space during the search process. This information could be valuable to generate local approximations for the local search phase. Thus, when using an indirect local search, by means of approximated functions, we have:

$$t_{LS} = (m + p + q)t_{app} + t_{opt} \tag{4.56}$$

which is the case if the proposed methodology is applied every generation ($n_L = 1$) with no trust region updates ($n_{TR} = 0$). Moreover, if $\sigma((m + p + q)t_{app} + t_{opt}) << \mu t_e$, that is, the time to evaluate a solution is dominant in the problem, $\langle N_H \rangle$ could be slightly smaller than $\langle N_G \rangle$, and the initial condition would still be satisfied and thus the additional complexity introduced by the methodology would be justifiable. This relation makes evident the context of problems in which the proposed methodology can be considered beneficial.

Let us now consider the situation with $n_{TR} > 0$ and $n_L = 1$. In this case, we have:

$$t_{LS} = (1 + n_{TR})\left[(m + p + q)t_{app} + t_{opt}\right] + n_{TR}t_e \tag{4.57}$$

which reduces to (4.56) if $n_{TR} = 0$. Note that this expression is the time equivalent of the expression for O_{LS} derived before.

If we assume $t_e >> t_{app} + t_{opt}$, we get:

$$t_{LS} \cong n_{TR}t_e \tag{4.58}$$

Using this result in relation (4.54), we obtain:

$$\langle N_H \rangle < \langle N_G \rangle \frac{\mu t_e}{(\mu t_e + \sigma n_{TR} t_e)} = \langle N_G \rangle \frac{\mu}{(\mu + \sigma n_{TR})} \tag{4.59}$$

This relation shows us that safer choices for σ and n_{TR} are those that lead to relatively small values of the product σn_{TR} in comparison with μ. On the other hand, if the problem at hand has low modality, we can converge with lesser iterations if we spend more effort in local search. This relation shows that the time can also be smaller with a greater product σn_{TR}, as long as the hybrid algorithm converges (and it potentially can) in many fewer generations than the algorithm without local search.

Finally, we consider the case when the local search is not applied at every generation. In this case, we have:

$$\langle T_H \rangle = (\mu t_e + t_s + t_v + \sigma t_{LS}/n_L)\langle N_H \rangle \tag{4.60}$$

Using the same development as before, we obtain:

$$\langle N_H \rangle < \langle N_G \rangle \frac{\mu t_e}{(\mu t_e + \sigma n_{TR} t_e/n_L)} = \langle N_G \rangle \frac{\mu}{(\mu + \sigma n_{TR}/n_L)} \tag{4.61}$$

which allows us to use greater values for σ and n_{TR}, as long as we use some interval between the local searches.

Assuming that $\langle N_H \rangle < \langle N_G \rangle$, we can define a $0 < \delta < 1$ such that[5]:

$$\langle N_H \rangle = \delta \langle N_G \rangle \tag{4.62}$$

With this definition, we can derive an additional relationship from (4.54):

$$t_e > \frac{t_{LS}}{n_L} \left[\frac{\sigma}{\mu}\right] \frac{\delta}{(1-\delta)} \rightarrow t_{e,\min} = \frac{t_{LS}}{n_L} \left[\frac{\sigma}{\mu}\right] \frac{\delta}{(1-\delta)} \tag{4.63}$$

This equation is useful to find the minimum value for t_e, in which the proposed methodology for hybrid algorithms is useful. For instance, in a computational experiment using well-known benchmark test functions, it is possible to estimate $\langle N_G \rangle$ and $\langle N_H \rangle$, and also t_{LS}. With these values, it is possible to find the minimum evaluation time required for that test problem to satisfy the initial condition $\langle T_H \rangle < \langle T_G \rangle$.

4.6 Discussion

This Chapter was divided in two main parts. In the first part, a discussion of the effect of the proposed local search methodology on global convergence properties of standard evolutionary algorithms is devised by using the Markov chain theory. In the second part, we turned to computational complexity issues of the local search and its computational overhead in CAD problems.

The Markov chain analysis is very useful to analyze global convergence of population-

[5]A hybrid algorithm that does not satisfy the condition $\langle N_H \rangle < \langle N_G \rangle$ is not useful in practice.

based optimization algorithms (Rudolph 1994, Agapie 1998), but it has been particularly applied to analyze standard genetic algorithms. Unfortunately, the analysis has some limitations, for instance, it is not valid for evolution strategies and differential evolution algorithms. In the case of standard genetic algorithms, the crossover and mutation rates are constant throughout the evolutionary process. That leads to time-independent transition matrix, thus the relation

$$\boldsymbol{\pi}(n) = \boldsymbol{\pi}(0)\mathbf{T}^n$$

is valid because $\mathbf{T}$ is constant.

In evolution strategies, the transition matrix associated with the recombination is time-independent but the one associated with the mutation operator is not constant, because the mutation rate is adaptive. Genetic algorithms can also have adaptive crossover and mutation rates. In general, adaptive evolutionary algorithms cannot be analyzed by Markov chain theory because their transition matrices are time-dependent. In these situations, we need to reccur to other analysis methods, as illustrated in (Greenwood and Zhu 2001) for adaptive evolution strategies. Adaptive memetic algorithms are analyzed with Markov chains in (Ong, Lim, Zhu and Wong 2006) but assuming that the transition matrix produced by the genetic operators is positive.

Regarding differential evolution, its global convergence proof is still an open issue in the literature, albeit there has been empirical evidence that the algorithm is globally convergent (Price, Storn and Lampinen 2005). It is not possible to state its global convergence via Markov chains because the transition matrix associated with the differential mutation is neither positive nor irreducible. Therefore, the algorithm is not globally convergent under Markov chain analysis.

The important point of the analysis presented here is that as long as we can say that the evolutionary algorithm is globally convergent under Markov chain analysis, the local search operator preserves this characteristic.

The second important point made in this Chapter is that the local search operator preserves the polynomial complexity of standard evolutionary algorithms. Therefore, the additional complexity in the memetic algorithm is not dramatic and would be justifiable in some applications. The final relations developed in this text show under which conditions the hybrid algorithm is advantageous. These relations show that the proposed

methodology can be very interesting for expensive optimization problems, in which the evaluation time t_e of a single solution is very big in comparison to the time required for performing the normal operations of the algorithm, including the local search. This scenario, commonly found in CAD problems, is the one in which the additional complexity of the memetic algorithm is justifiable.

Another limitation of the Markov chain analysis is that it is not practical for simulation purposes, because the simulation of the Markov chain in this scenario is very expensive due to huge state spaces. Therefore, the most commonly used way to evaluate convergence rates in practice is through empirical and statistical methods, by applying the algorithm to known problems and monitoring the evolution of the archive population $A(t)$. We need not only to monitor the evolution of the algorithm with the number of iterations, but we also need to monitor the computational times, either in seconds or number of floating point operations, required for each algorithm to perform one single iteration, particularly when comparing a hybrid algorithm with local search against its non-hybrid version. This methodology is adopted in the experimental results of this thesis.

In the next Chapter we test the evolutionary framework developed in Chapter 2 together with the local search methodology in Chapter 3 in practical CAD problems in the context of electromagnetic design.

Chapter 5

Results

"To assert that the earth revolves around the sun is as erroneous as to claim that Jesus was not born of a virgin."

— Cardinal Bellarmine, during the trial of Galileo, 1615

"Have you ever wondered how we know the things that we know? How do we know, for instance, that the stars, which look like tiny pinpricks in the sky, are really huge balls of fire like the sun and are very far away? And how do we know that Earth is a smaller ball whirling round one of those stars, the sun? The answer to these questions is evidence. [...] [N]ow I want to move on from evidence, which is a good reason for believing something, and warn you against three bad reasons for believing anything. They are called 'tradition,' 'authority,' and 'revelation.' "

— Richard Dawkins, 1941–

5.1 Preview

This Chapter presents the results obtained with the theory described previously. The problems investigated range from analytical test problems to practical benchmark problems in electromagnetics. As briefly mentioned before, the subtle and nonlinear interactions among components of evolutionary algorithms complicate their formal analysis. Because of this, most of our conclusions and understanding are derived from experi-

ments and with the philosophical support of experimentalism. The sections describing each experiment begin with the description of the problems, followed by the methodology adopted to verify the hypothesis under consideration for each particular experiment.

The first group of experiments is performed over three analytical optimization problems, each one with specific characteristics. The second problem investigated is a relatively well-known problem in shape optimization of electromagnetic devices - the shape optimization of the pole face of a magnetizer device. This problem is used to illustrate the overhead of the approximation-based local search in the evolutionary process. The third problem studied is the TEAM benchmark problem 22, which was proposed in the mid-1990's, but which is hitherto a versatile benchmark problem for optimization in electromagnetics. This problem can be stated as either a mono-objective or a multi-objective problem, with inequality and equality constraints, serving therefore as a good illustration of optimization problem instances.

5.2 Analytical problems

5.2.1 Problem statement

In this first experiment we apply a typical genetic algorithm configuration from the framework developed in Chapter 2 to analytical unconstrained optimization problems. The goal here is to observe the effect of the local search on the convergence speed of the evolutionary algorithm in terms of the number of generations. For this purpose, we consider the minimization of the following functions:

$$f_1(\mathbf{x}) = \sum_{i=1}^{d-1} 100 \left(x_i^2 - x_{i+1}\right)^2 + (1 - x_i)^2 \tag{5.1}$$

whose search space is $\mathcal{X}_1 = \{\mathbf{x} : -2 \leq x_k \leq 2,\ k = 1, \ldots, d\}$.

$$f_2(\mathbf{x}) = 2.6164 + \frac{1}{d} \sum_{i=1}^{d} 0.01 \left[(x_i + 0.5)^4 - 30x_i^2 - 20x_i\right] \tag{5.2}$$

whose search space is $\mathcal{X}_2 = \{\mathbf{x} : -6 \leq x_k \leq 6,\ k = 1, \ldots, d\}$.

$$f_3(\mathbf{x}) = 10d + \sum_{i=1}^{d} \left[x_i^2 - 10\cos(2\pi x_i)\right] \tag{5.3}$$

whose search space is $\mathcal{X}_3 = \{\mathbf{x} : -5 \leq x_k \leq 5,\ k = 1, \ldots, d\}$.

As it can be seen from (5.1)-(5.3), all these functions are scalable, that is, they can be defined for any number of variables. That characteristic makes them very useful for testing optimization algorithms. Additionally, the three functions have $f(\mathbf{x}^*) = 0$ at the global optimum.

The function f_1 is the Rosenbrock function, which is a unimodal and non-convex function. The surface defined by the Rosenbrock function, see Figure 5.1(A), is dominated by a large gradual slope. This slope is raised along one edge to a fine point. Despite the apparent simplicity of this function, it is notoriously hard for evolutionary algorithms in general.

The function f_2 is a multimodal function with a moderate degree of multimodality, with 2^d local optima in the given search space. Figure 5.1(B) illustrates its surface for $d = 2$.

The function f_3 is the Rastrigin function, with a high degree of multimodality, see Figure 5.1(C). The Rastrigin function is another difficult problem as it defines a search space with many local optima, due to the sinusoidal function added to the quadratic function.

5.2.2 Methodology

The goal of this section is to compare a typical genetic algorithm with its hybridized version using approximation-based local search. Hereafter, the former algorithm is referred to as GA, and the latter is referred to as MQMA, i.e., a memetic algorithm employing the multiquadric approximation for the local search.

Each run of each algorithm generates a sequence of solutions in the archive population of the form $\mathcal{S}_k = \{f(\mathbf{x} \in A(1)), \ldots, f(\mathbf{x} \in A(k))\}$ in k steps, that is, the best solution in each step. The comparison of these two stochastic optimizers is based on the definition of concrete performance measures. There is no canonical rule to define a measure for this purpose, and here we adopt some typical measures found in the literature:

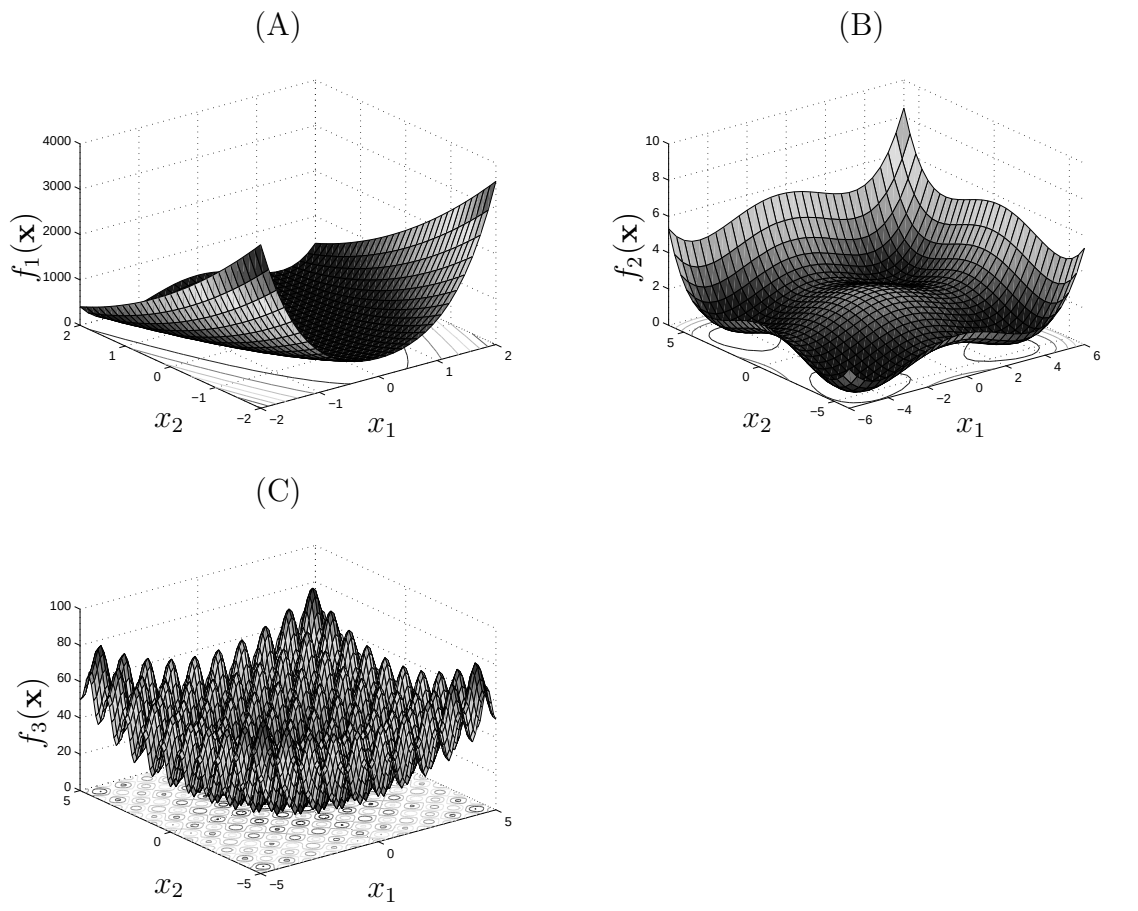

Figure 5.1: Surfaces of the two-dimensional versions of (A) $f_1(\mathbf{x})$, (B) $f_2(\mathbf{x})$, and (C) $f_3(\mathbf{x})$.

- To measure the algorithm speed, the average number of evaluations needed to reach a specific solution or a specific level can be used. The maximum number of evaluations can be used for those runs finding no solutions.

- The previous measure does not provide a "picture" of the progress of the algorithm. For this purpose, one can use the mean convergence curve, which is simply the mean of the sequences $\mathcal{S}_k$ produced in each run.

- Another progress measure is the mean convergence velocity curve. The convergence velocity $c(n)$ is defined as

$$c(n) = \log\left(\sqrt{\frac{f(\mathbf{x} \in A(n))}{f(\mathbf{x} \in A(1))}}\right) \tag{5.4}$$

The mean convergence velocity curve $\bar{c}(n)$ is given by the mean of the convergence velocities over all runs. Notice that $c(1)$ is always equal to zero.

- If the optimal solution is known, the percentage of runs terminated successfully (success rate) can be used to measure the performance of an algorithm.

Having specified the measures for basing our comparison, we now provide the setup of each algorithm. Both the GA and the MQMA shared the following setup:

- $\mu = \lambda = 50$ and $\xi = 1$;
- ES(μ, μ) for the substitution;
- Roulette wheel as selection operator;
- Real-biased convex crossover with $\rho_c = 0.8$, $\rho_{pol} = 0.8$, and extrapolation of 0.1;
- Gaussian mutation with $\rho_m = 0.1$;
- The only stop criterion is the maximum number of generations, set as 200;

The MQMA used MQ-based local search with a rectangular vicinity with fixed vicinity size of $\epsilon = 0.1$, that is, 10% of the search space size. The maximum number of points used to build the local approximation is $N_L = 200$. The local search was applied to the best individual in the offspring at every generation, hence $\sigma = 1$ and $n_L = 1$.

All three test functions were minimized in a 4-dimensional search space.

5.2.3 Results

Due to the stochastic behavior of both algorithms, we need to perform a given number of runs to draw meaningful conclusions. Therefore, each algorithm was executed 40 times on each problem. Figures 5.2 to 5.4 show the mean convergence velocity of the GA and the MQMA on each problem. As we can see, the memetic algorithm converged faster than the typical GA on these problems. These results confirm the hypothesis that the local search speeds up the convergence of the genetic algorithm.

Table 5.1 shows the success rate of both algorithms on all problems, as well as an estimate of the average number of generations to converge $\langle N \rangle$. The maximum number

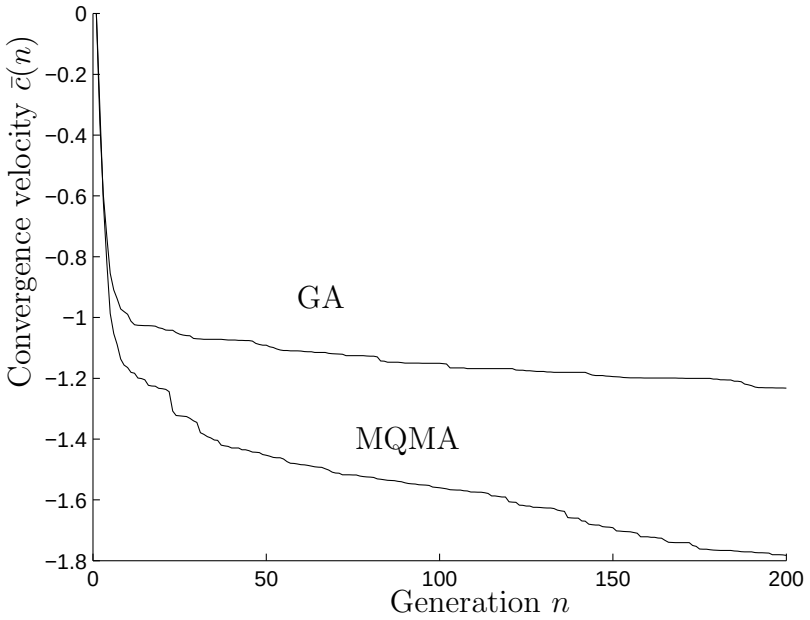

Figure 5.2: Mean convergence velocity $\bar{c}(n)$ for the optimization of $f_1(\mathbf{x})$.

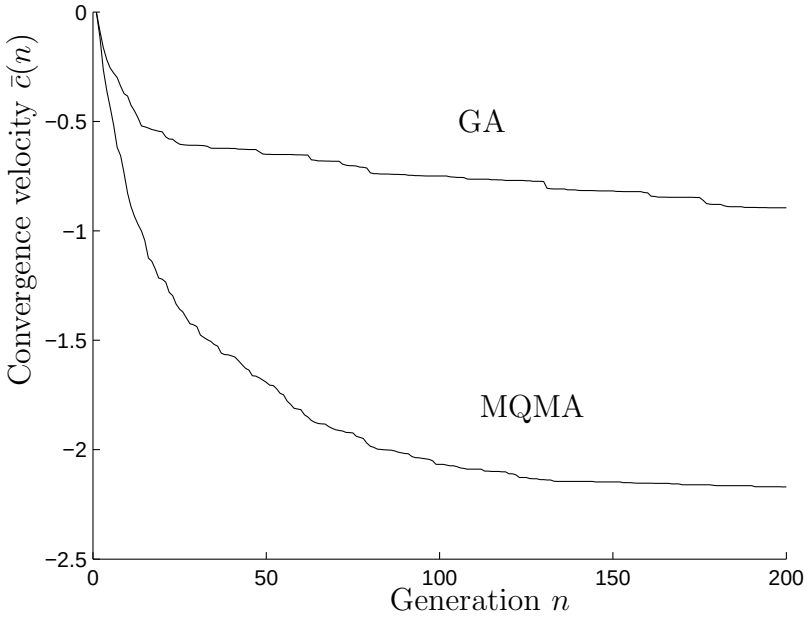

Figure 5.3: Mean convergence velocity $\bar{c}(n)$ for the optimization of $f_2(\mathbf{x})$.

200 is used for those runs in which the algorithm has not converged. This Table shows

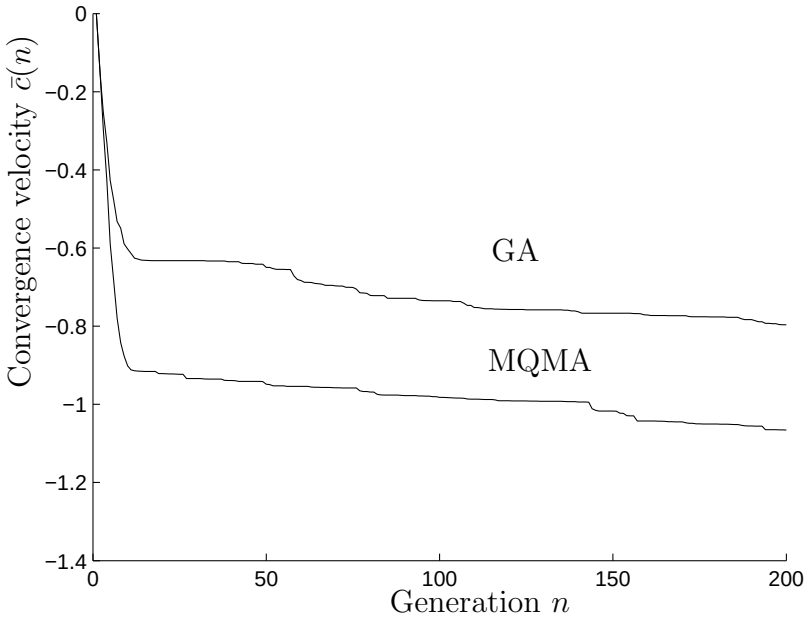

Figure 5.4: Mean convergence velocity $\bar{c}(n)$ for the optimization of $f_3(\mathbf{x})$.

that the local search not only has improved the convergence speed of the algorithm but also it has increased the success rate.

The smallest difference in $\langle N \rangle$ is observed in the Rastrigin function, while the biggest difference appears in the minimization of f_2. The Rastrigin function is highly multimodal, therefore one would expect that the local search would not present a great impact on the performance. The observed effect on the performance may be understood by noting that the Rastrigin function is in fact a quadratic function plus noise. The local approximation is not capable of modeling this noise, due to undersampling, specially in the first generations, but can capture the global trend of the quadratic component in the objective function. The mean curves in Figure 5.4 clearly shows a fast decrease in the first 15 generations. After that, the slopes of the mean convergence velocity curves for both algorithms are similar. After some generations, the noise component was also "learned" during the evolutionary process, then reducing the impact of the local search because the basins of attraction become small. That is an interesting side effect of the approximation-based local search: it can filter a highly multimodal function, capturing the global trend in the objective function.

The mean velocities for f_1 and f_2 present different slopes for a wider range of generations, showing that the local search is having an important impact in the search process.

Problem	Algorithm	Success rate	$\langle N \rangle$
$f_1(\mathbf{x})$	GA	22.5%	171.9
	MQMA	67.5%	105.0
$f_2(\mathbf{x})$	GA	67.5%	106.8
	MQMA	95.0%	21.5
$f_3(\mathbf{x})$	GA	7.5%	186.9
	MQMA	20.0%	166.5

Table 5.1: Results for the analytical functions.

This is because f_1 is unimodal and f_2, although multimodal, has a low to medium degree of multimodality. The basins of attractions in f_2 are fairly convex and present no obstacle for a gradient-based local search. This is why the performance in f_2 is even better than in f_1. The non-convexity of f_1 poses some difficulty for the approximation, due to the gradual slope of the "banana" region, and also for the local search.

5.3 Magnetizer design and computational cost

5.3.1 Definition of the magnetizer problem

The magnetizer problem concerns with the design of its pole shape. The version used here is based on the problem described in (O. A. Mohammed 1997), which is also explored in detail in (Mohammed 1999).

The geometry of the device is shown in Figure 5.5. A high current is applied to the coil, which causes a magnetic field to be set through the material that is to be magnetized. The region to be magnetized is assumed to be made of nonmagnetic material. The pole face and the outer shell have relative permeability of 2000.

The objective is to minimize the difference between the calculated magnetic flux density along the chord AB and a predefined form:

$$D = \sum_{n=1}^{59} |B_{i,calc} - B_{i,ref}| \tag{5.5}$$

where $B_{i,calc}$ and $B_{i,ref}$ are respectively the magnitude of the calculated magnetic flux density and the reference value at the ith point along the chord AB. The reference value is given by:

$$B_{i,ref} = 0.32 \sin(\pi\theta_i/180), \quad \theta_i = 31^\circ, \ldots, 89^\circ \tag{5.6}$$

The magnetized material is placed across a 60° angle. Points A and B are located at a 1° distance from either end of the magnetized region, therefore making a 58° angle. At point B, the flux density is maximum, and it is expected to follow a cosine function along AB.

In the original problem, the optimization variables are the radial distances from each control point to the reference point ($\rho = 25, \theta = \pi/2$). The angular coordinate of each point is kept constant with 18° between two consecutive points. The shape

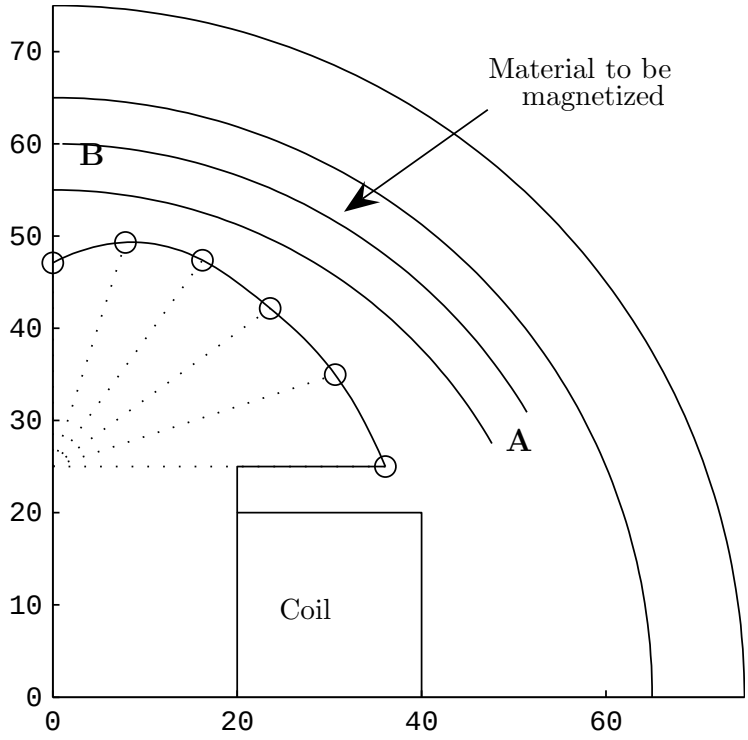

Figure 5.5: Geometry of the magnetizer problem in mm. The radius of the stator is $75mm$.

of the pole face is determined from a spline interpolation of the control points. Alas, some configurations tested by the evolutionary algorithm produce very oscillating and complex curves, see Figure 5.6. In (O. A. Mohammed 1997), the author circumvented this problem by adding a penalty term to the objective function. The penalty term is added whenever the slopes of the line segments connecting consecutive control points are positive. The penalty increases with the number of line segments with a positive slope. This strategy imposes smooth shapes with negative slopes, since the curve modeling the pole face is expected to decrease monotonically. The problem with adopting this strategy is that some configurations cannot even be analyzed by the finite-element solver, because the spline curve may touch the magnetic material of the outer shell and thus make the solver crash. One could check this condition before calling the solver, but it would complicate the problem, and the objective function, since a dummy value would have to be used in substitution for the value provided by the solver.

The search for smooth shapes can be achieved by simply using a different parameterization in order to avoid non-manufacturable shapes. Here we adopt a new parameterization for this problem, in which the radius of each control point is defined as

$$r_i = r_{i-1} + x_i, \quad i = 1, \ldots, 6 \tag{5.7}$$

with $r_0 = 20$. The optimization variables are the increments $0 \leq x_1 \leq 6$ and $0 \leq x_i \leq 5$, $i = 2, \ldots, 6$ instead of the radius themselves. The control points are ordered from 1 to 6

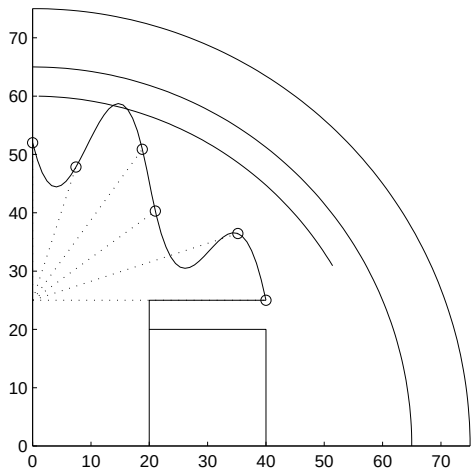

Figure 5.6: Example of a bad pole shape tested by the evolutionary algorithm.

in clockwise rotation. With this parameterization, we do not need to adopt the penalty scheme for positive slopes.

The FEMM 4.0 software[1] is used to solve the magnetostatic field problem involved in the evaluation of the objective function.

5.3.2 Results

The methodology for the magnetizer problem is similar to the one used for the analytical problems. Both the GA and the MQMA shared the following setup:

- $\mu = \lambda = 50$ and $\xi = 1$;
- ES(μ, μ) for the substitution;
- Roulette wheel as selection operator;
- Simulated binary crossover with $\rho_c = 0.8$, $\eta = 2$;
- Gaussian mutation with $\rho_m = 0.1$, $s = 0.2$;
- The only stop criterion is the maximum number of generations, set as 100;

The MQMA used MQ-based local search with a rectangular vicinity with fixed vicinity size of $\epsilon = 0.1$, that is, 10% of the search space size. The maximum number of points used to build the local approximation is $N_L = 400$. The local search was applied to the best individual in the offspring at every 5 generations, hence $\sigma = 1$ and $n_L = 5$.

Each algorithm was executed 30 times on the magnetizer problem and mean convergence curves were generated. Figure 5.7 shows the mean convergence curves for each algorithm. The hybrid algorithm converged in less generations, and accordingly less time, showing that the local approximation and local search methodology was advantageous for this problem. The next section discusses the overhead of the local search in this problem. This result together with the overhead analysis is presented in (Guimarães, Lowther and Ramírez 2007).

[1]Finite Element Method Magnetics - FEMM - is authored by David Meeker. It is available at http://femm.foster-miller.net/ and licensed under the terms of the Aladdin Free Public License.

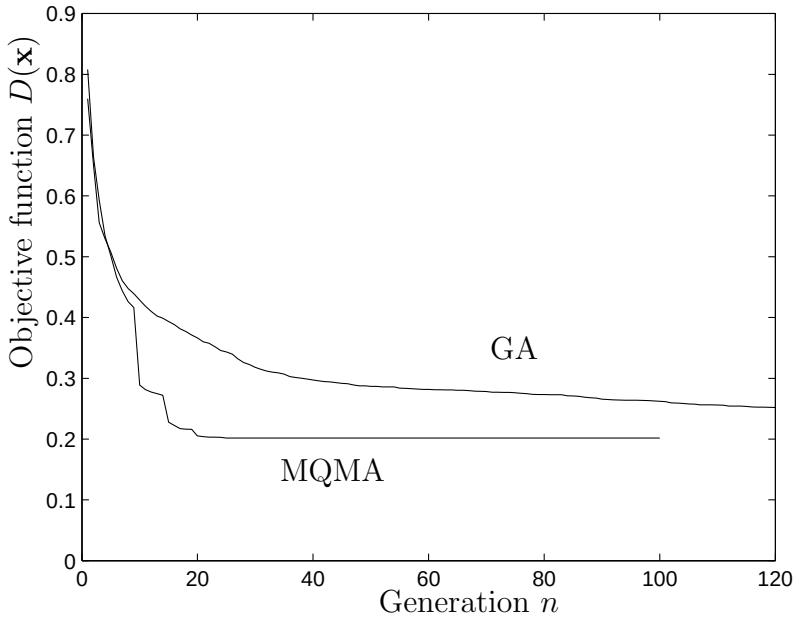

Figure 5.7: Mean convergence curve for the magnetizer problem.

5.3.3 Overhead of the local search

Chapter 4 presented a discussion on the computational complexity of the proposed local search. The computational times in (4.53) can be given in physical time or in number of floating point operations - flops, which has the advantage of being machine independent. In this section we use physical time to discuss computational cost for two reasons: (i) in practical problems, such as the magnetizer, the objective function is a black-box function, thus the access to the number of flops used is difficult or even impossible and (ii) the evolutionary framework is implemented in the Matlab environment. Since the Matlab version 6, the counting of floating point operations is no longer practical.

In the previous experiments, we also monitored the times in seconds taken to perform each of the following steps: evaluation of one individual (t_e), local search of one individual (t_{LS}), selection (t_s), and variation (t_v). Table 5.2 shows the mean times for each step in the magnetizer problem and also in the analytical functions f_1, f_2 and f_3.

In the analytical problems, the local search was the most time-consuming step. The overhead of the MQ-based local search is very high, because the evaluation of the objective functions in these problems is very fast. The evaluation of the multiquadric approx-

Step	t_e	t_s	t_v	t_{LS}
	Analytical function f_1			
Mean values $[s]$	2.7×10^{-6}	1.0×10^{-3}	0.45×10^{-3}	2.24
Normalized	1	377	167	830,890
	Analytical function f_2			
Mean values $[s]$	2.2×10^{-6}	0.8×10^{-3}	0.37×10^{-3}	2.20
Normalized	1	380	166	999,160
	Analytical function f_3			
Mean values $[s]$	1.9×10^{-6}	0.7×10^{-3}	0.26×10^{-3}	1.89
Normalized	1	393	139	994,410
	Magnetizer problem			
Mean values $[s]$	1.23	0.65×10^{-3}	0.72×10^{-3}	2.35
Normalized	1	0.53×10^{-3}	0.59×10^{-3}	1.91

Table 5.2: Mean times

imation is even more expensive than the evaluation of the objective function. Therefore, in these problems, although the memetic algorithms converged in less iterations, they took much more time to do so.

In the magnetizer problem, the situation inverts. Although t_{LS} is still greater than t_e, the time to evaluate the whole population dominates the process, and the overhead of the local search becomes small. Since $\sigma = 1$, $n_L = 5$ and $\mu = 50$, we have:

$$\langle N_H \rangle < \langle N_G \rangle \frac{50t_e + t_s + t_v}{(50t_e + t_s + t_v + 1t_{LS}/5)} \tag{5.8}$$

and:

$$\frac{t_{LS}}{50t_e} = 0.038 \tag{5.9}$$

The time taken to perform the local search of one individual is negligible, being less than 4% of the time taken to evaluate the whole population. With $n_L > 1$, the overhead of the local search is further reduced.

Table 5.2 presents the mean time to perform the local search of one individual, but this time increases with the number of generations, as discussed in Chapter 4. As the

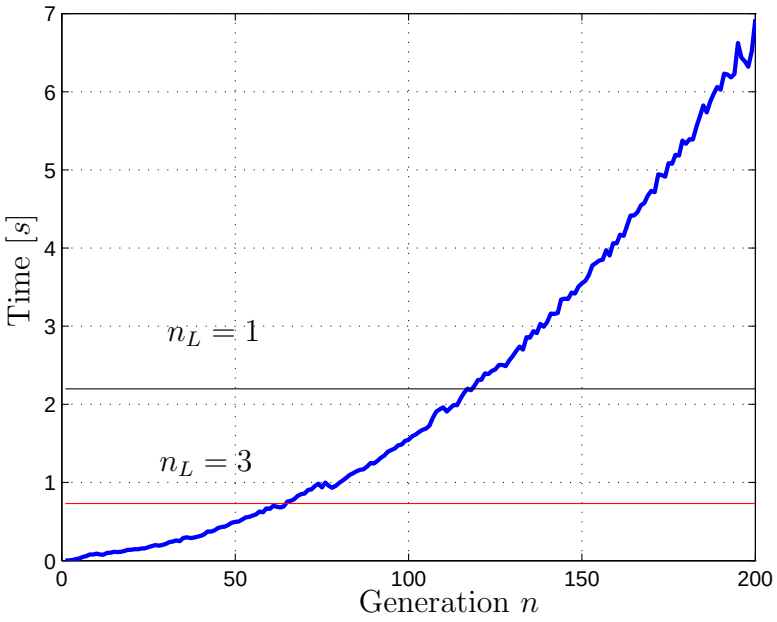

Figure 5.8: Local search time as a function of n, and averaged over 40 independent runs over the function f_2. The mean time for $n_L = 1$ and $n_L = 3$ is also shown.

generation goes, the size of the global data set increases linearly with n, and the time to find the local data set increases. Moreover, as the number of samples increases towards the end of the evolutionary process, the probability of having N_L samples available to build the approximation also becomes larger, thus increasing the time to generate the local approximations and the time to optimize them. Figure 5.8 shows the time taken to perform the local search as a function of the generation counter n for the problem f_2. This curve was obtained by taking the average over all 40 runs. The slope of this curve changes along the process. In the first generations, many (if not all) local approximations are built with less than N_L points. In the last generations, most local approximations use the maximum N_L points, because more samples are available in the global data set and the probability of performing local searches in similar regions of the search space increases. In the last generations, the slope of the curve in Figure 5.8 is almost constant, with the time increasing linearly with n as expected. Figure 5.8 also shows the mean times t_{LS}/n_L obtained when $n_L = 1$ and $n_L = 3$.

For each analytical problem, we can use the values in Table 5.1 as estimates of $\langle N_H \rangle$ and $\langle N_G \rangle$. These values and the mean times in Table 5.2 can be used in (4.63) to obtain the minimal times $t_{e,\min}$ that would make the MQMAs converge in less time than the

GAs, see Table 5.3. This Table shows that if the time to evaluate f_1, f_2 and f_3 was some tenths of a second, the memetic algorithm would have converged in less time than the genetic algorithm.

	f_1	f_2	f_3
t_{LS} [s]	2.24	2.20	1.89
δ	0.611	0.201	0.891
$t_{e,\min}$ [s]	0.071	0.011	0.310

Table 5.3: Minimal evaluation times

Back to the magnetizer problem, we can use the mean times in Table 5.2 in (5.8), getting the condition

$$\langle N_H \rangle < 0.9924 \langle N_G \rangle \tag{5.10}$$

which makes $\langle T_H \rangle < \langle T_G \rangle$. We note that in this example, the evaluation time is dominant and the overhead of the local search is not significant. Therefore, if the memetic algorithm converges in less generations than the genetic algorithm, it also converges in less time.

5.4 Superconducting magnetic energy storage system

The TEAM Workshop problem 22 was proposed by (Brandstätter and Ring 1996) and accepted as a benchmark problem for optimization in electromagnetics. From then on, many papers were published presenting diverse solutions for the problem, see for instance (Alotto, Caiti, Molinari and Repetto 1996, Ebner, Magele, Brandstatter and Richter 1998, Ioan, Ciuprina and Szigeti 1999, Takahashi, Saldanha, Dias-Filho and Ramírez 2003, Guimarães, Barros and Ramírez 2003), amongst others. These works were mainly concerned with the validation of some optimization methodology or technique in electromagnetic design.

The problem 22 consists in optimizing the dimensions of a super-conducting magnetic energy storage device, known by the acronym SMES. Such a device has many relevant applications (Luongo 1996) and an optimal configuration for it is an important issue to

resolve. There are different formulations for the problem in the literature, depending on the technique employed to solve the optimization.

This problem presents one inequality constraint and one equality constraint, and can be formulated as either a mono or multi-objective problem. Thus this benchmark problem is a good example of an optimization problem instance within the context of CAD problems in electromagnetics.

5.4.1 Characterization of the problem

The main idea of the SMES is the construction of a device that allows the energy storage and its availability when necessary. The problem 22 consists in the optimization of the configuration of a SMES with two solenoids. This problem has three distinct objectives:

- The stray field, evaluated 10 meters away from the device, must be as small as possible;
- The stored energy must be equal to 180MJ (50kWh);
- The physical condition that guarantees the superconductivity, named the quench condition, must not be violated.

The analysis of the SMES involves its stored energy E, the maximum field in the superconductor B_{max} and the mean stray field at 21 evaluation points B_{stray} .

In far field evaluations we may achieve precise results at low computational cost using directly the Biot-Savart law (Ioan, Ciuprina and Szigeti 1999). Nevertheless, in energy evaluations we have an additional space integration over the whole domain of the problem, and so a high computation cost. Using the finite element method to solve a magneto-static formulation of SMES, we may reach precise energy values at relatively low cost. On the other side, the finite element method fails in far field evaluations, due to the infinite domain truncation.

Figure 5.9 shows the geometry of the problem. B_{stray} is evaluated in 21 points along the lines $(y = 10m; 0 \leq z \leq 10m)$ and $(z = 10m; 0 \leq y \leq 10m)$. These values are

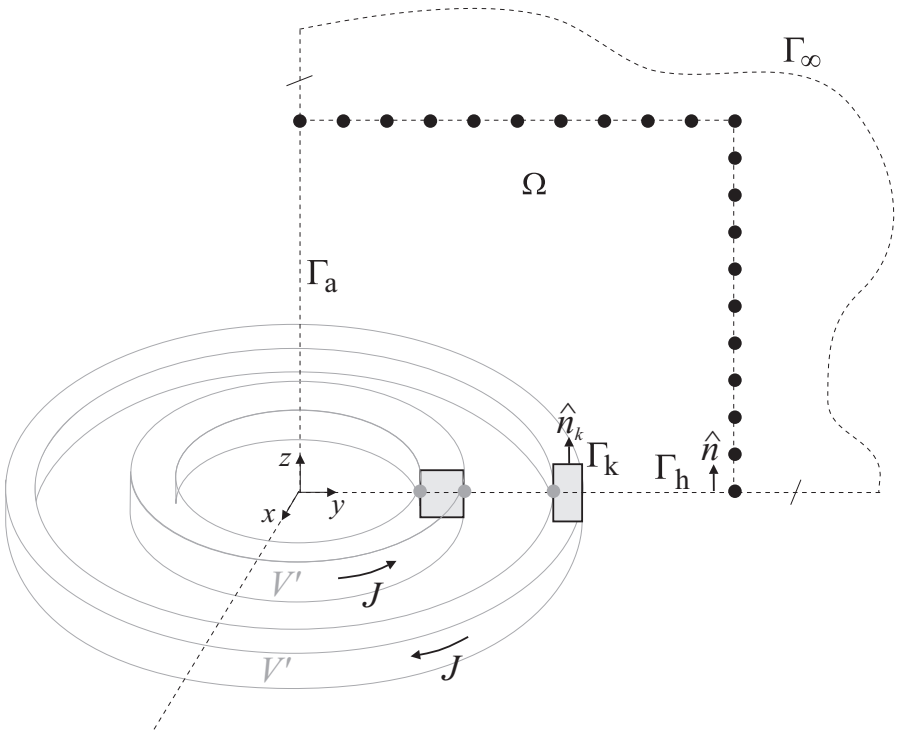

Figure 5.9: Finite Element Method model of the SMES device.

unified by the expression:

$$B_{stray} = \sqrt{\frac{1}{21}\sum_{i=1}^{21} |B_{stray_i}|^2} \tag{5.11}$$

where B_{stray_i} is the value for the magnetic flux density evaluated in one of the 21 points. This radiated field by the SMES may be formulated as the summation of the contributions of each of the two loops, given by the numeric solution of integrals derived from the Biot-Savart law.

We may evaluate the energy through the finite element method, using a magnetostatic formulation with magnetic vector potential $\vec{A}$, considering the axis-symmetry to simplify it into a 2D problem, as shown in Figure 5.9.

The full-version of the problem has eight optimization variables: the current density in both coils and the coil shapes, defined by the radius, height and width of their 2D cuts. Nonetheless, a simplified version with only three variables can also be defined, in which the dimensions of the outer coil are optimized. Table 5.4 presents the ranges for the three variable version as well as the values for the fixed parameters. The variables

Variable	r_1	h_1	d_1	r_2	h_2	d_2	J_1	J_2
Unit	m	m	m	m	m	m	MA/m^2	MA/m^2
min	–	–	–	2.6	0.408	0.1	–	–
max	–	–	–	3.4	2.2	0.4	–	–
fixed	2.0	1.6	0.27	–	–	–	22.5	22.5

Table 5.4: Parameters for the problem 22

r_1, h_1, d_1 are respectively the radius, the height and the width of the inner coil. The variables r_2, h_2, d_2 represent the same for the outer coil. J_1 and J_2 are respectively the current density in the inner and outer coil.

The quench condition is obtained from the operating curve of the superconductor material. The linearization of this curve provides:

$$|J| = (-6.4|B| + 54.0)\, A/mm^2 \tag{5.12}$$

which, for the three variable version reduces to:

$$|B| = 4.92T \tag{5.13}$$

given that the current densities are fixed and equal to $22.5A/mm^2$.

Thus, the biggest value for the magnetic flux density cannot exceed this limit, or the superconductivity state is violated. The biggest value for the flux density occurs in one of the following three points:

1. inner wall of the inner coil: $q_1 = (r_1 - d_1/2, 0)$

2. outer wall of the inner coil: $q_2 = (r_1 + d_1/2, 0)$

3. inner wall of the outer coil: $q_3 = (r_2 - d_2/2, 0)$

Thus, the magnetic flux density is evaluated only in these three points and the biggest value $B_{\max}$ is taken.

5.4.2 Mono-objective version

The mono-objective version of the problem 22 is stated as follows:

$$\text{subject to: } \begin{cases} \min f(\mathbf{x}) = B_{stray} \\ \mathbf{x} \in \mathcal{X} \\ g(\mathbf{x}) = B_{\max} - 4.92T \leq 0 \\ h(\mathbf{x}) = |E - 180MJ|/180MJ = 0 \end{cases} \tag{5.14}$$

We will apply standard evolutionary algorithms and its memetic versions of the framework in this problem. For this purpose, we selected 12 EAs whose configurations are shown in Table 5.5. As we can see in Table 5.5, there are 6 GAs, 2 DEs, 2 ES and 2 EPs. In Table 5.5, the substitution type is coded as follows:

- 1: ES(μ, μ) substitution;
- 2: ES$(\mu + \lambda)$ substitution;
- 3: ES$(1 + 1)$ for each parent-offspring pair;

Each one of these algorithms is hybridized with the MQ-based local search in two different ways:

- MA1 uses a rectangular vicinity with fixed vicinity size of $\epsilon = 0.1$. The maximum number of points used to build the local approximation is $N_L = 400$. The local search has $\sigma = 1$, $n_L = 4$, and $n_{TR} = 0$.
- MA2 uses a rectangular vicinity with fixed vicinity size of $\epsilon = 0.1$. The maximum number of points used to build the local approximation is $N_L = 400$. The local search has $\sigma = 10$, $n_L = 4$, and $n_{TR} = 1$.

MA2 has a more intensive local search step than MA1, with the local search applied to ten individuals in the population. Additionaly, MA2 has one trust region iteration, adding one extra evaluation for each individual selected for local search. Contrastingly, MA1 adds no extra cost in terms of calls to the finite-element solver.

Each evolutionary algorithm and each memetic algorithm is executed 10 times over (5.14). In each case, we derive mean convergence velocity curves, as before. The results

EA #	μ	λ	Selection operator	Variation operators		Substitution type
1	40	40	Roulette wheel	Real-biased crossover with $\rho_c = 0.8$, $\xi = 0.1$, $\rho_{bias} = 0.8$	Gaussian mutation with $\rho_m = 0.1$, $s = 0.1$	1
2	40	40	Roulette wheel	Gen. real-biased crossover with $\rho_c = 0.8$, $\xi = 0.1$, $\rho_{bias} = 0.8$	Gaussian mutation with $\rho_m = 0.1$, $s = 0.1$	1
3	40	40	Roulette wheel	Convex crossover with $\rho_c = 0.8$, $\xi = 0.1$	Gaussian mutation with $\rho_m = 0.1$, $s = 0.1$	1
4	40	40	Tournament $T = 2$	Real-biased crossover with $\rho_c = 0.8$, $\xi = 0.1$, $\rho_{bias} = 0.8$	Chaotic mutation (logistic) with $\rho_m = 0.1$, $s = 0.1$	1
5	40	40	Tournament $T = 2$	Gen. real-biased crossover with $\rho_c = 0.8$, $\xi = 0.1$, $\rho_{bias} = 0.8$	Chaotic mutation (logistic) with $\rho_m = 0.1$, $s = 0.1$	1
6	40	40	Tournament $T = 2$	Convex crossover with $\rho_c = 0.8$, $\xi = 0.1$	Chaotic mutation (logistic) with $\rho_m = 0.1$, $s = 0.1$	1
7	40	40	None	Differential mutation with $\mathbf{x}_{r1}$ =rand and $w = 1$		3
8	40	40	None	Differential mutation with $\mathbf{x}_{r1}$ =best and $w = 1$		3
9	10	40	Uniform roulette	Convex recombination with $\rho_c = 0.6$, $\xi = 0.1$	Gaussian mutation with $\rho_m = 1.0$, $s = 0.2$	2
10	10	40	Uniform roulette	Panmictic recombination with $\rho_c = 0.6$, $\xi = 0.1$	Gaussian mutation with $\rho_m = 1.0$, $s = 0.2$	2
11	40	40	None	Gaussian mutation with $\rho_m = 1.0$, $s = 0.2$		3
12	40	40	None	Chaotic mutation (logistic) with $\rho_m = 1.0$, $s = 0.2$		3

Table 5.5: Configuration for the 12 EAs tested on problem 22.

are shown in Figures 5.10 to 5.21. All the 12 EAs tested have shown an increased convergence speed when associated to local search.

Table 5.6 shows the mean number of evaluations for each algorithm after 100 generations. The percentage of local searches executed for each memetic algorithm is also reported in this Table. Remember that the local search is executed when the minimum number of points is available in the local data set. Note that the DE, ES and EP algorithms present the lowest percentages. This is possibly due to the different sampling mechanisms associated with each algorithm. The interaction among selection and variation operators produce the sampling behavior of the algorithm, which interfere in the data set used by the local search.

	evaluations			% local searches		
	EA	MA1	MA2	EA	MA1	MA2
1 (GA)	4040	4040	4289.8	–	99.60	99.92
2 (GA)	4040	4040	4288.5	–	99.20	99.40
3 (GA)	4040	4040	4289.7	–	99.20	99.88
4 (GA)	4040	4040	4289.3	–	98.80	99.72
5 (GA)	4040	4040	4288.0	–	99.20	99.20
6 (GA)	4040	4040	4288.1	–	98.80	99.24
7 (DE)	4040	4040	4272.1	–	92.00	92.84
8 (DE)	4040	4040	4280.1	–	96.00	96.04
9 (ES)	4010	4010	4239.7	–	93.60	91.88
10 (ES)	4010	4010	4239.2	–	93.60	91.68
11 (EP)	4040	4040	4244.4	–	84.80	81.76
12 (EP)	4040	4040	4248.5	–	86.40	83.40

Table 5.6: Percentage of executed local searches for each algorithm.

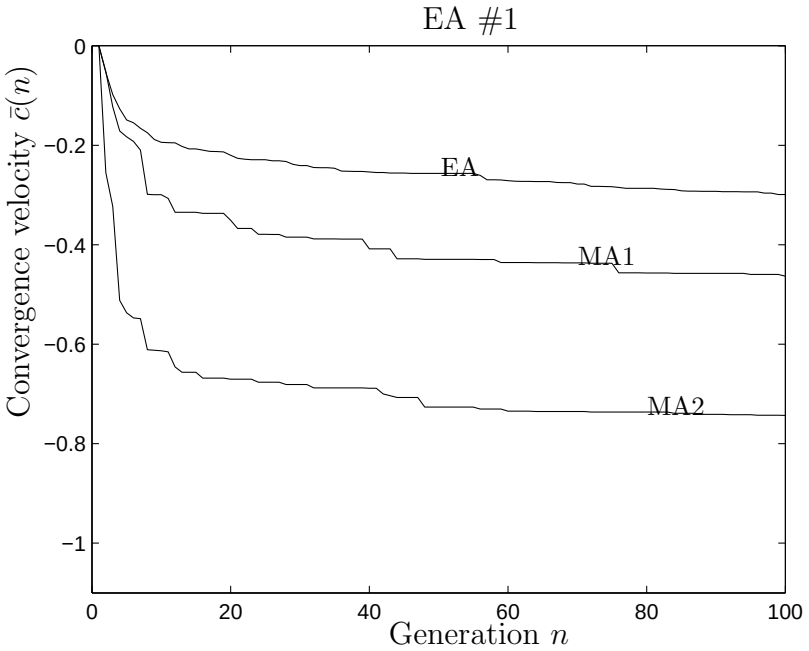

Figure 5.10: Convergence velocity for EA configuration #1.

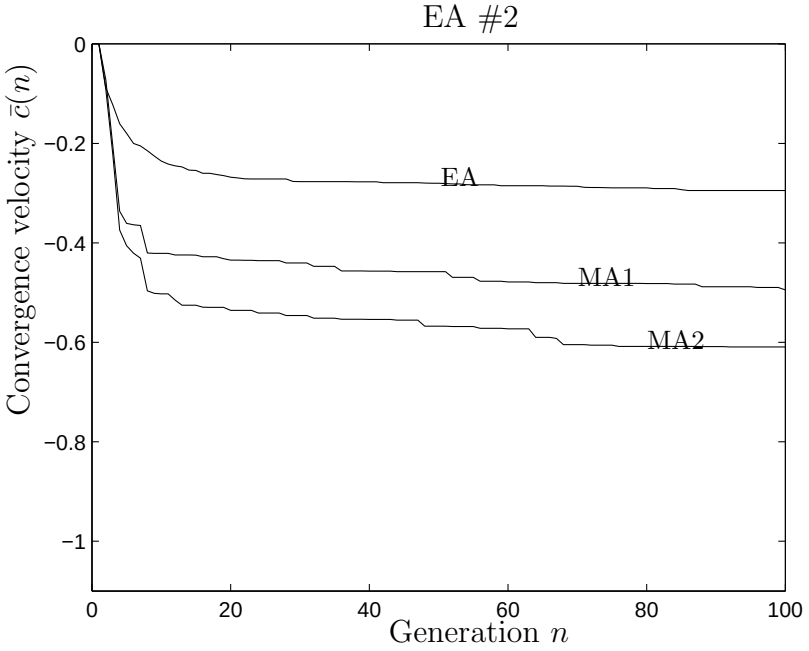

Figure 5.11: Convergence velocity for EA configuration #2.

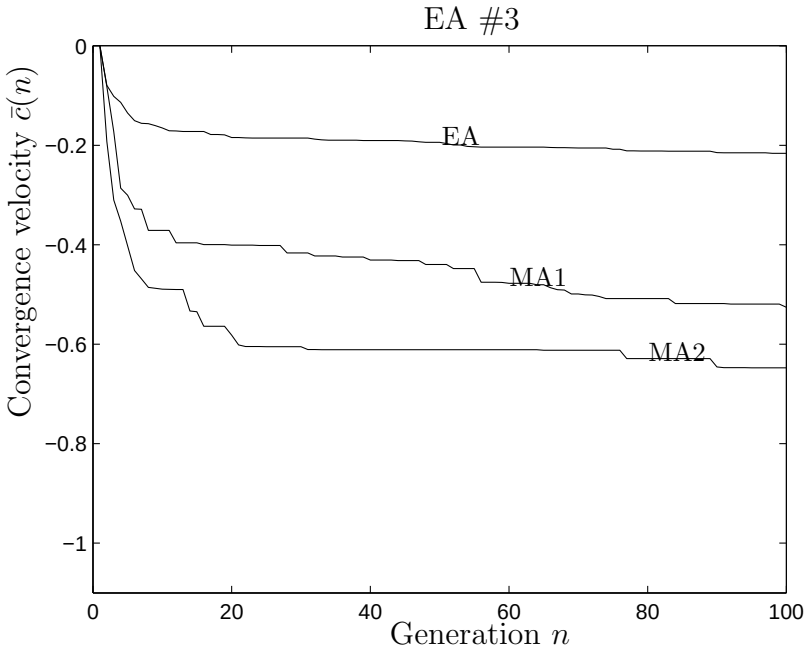

Figure 5.12: Convergence velocity for EA configuration #3.

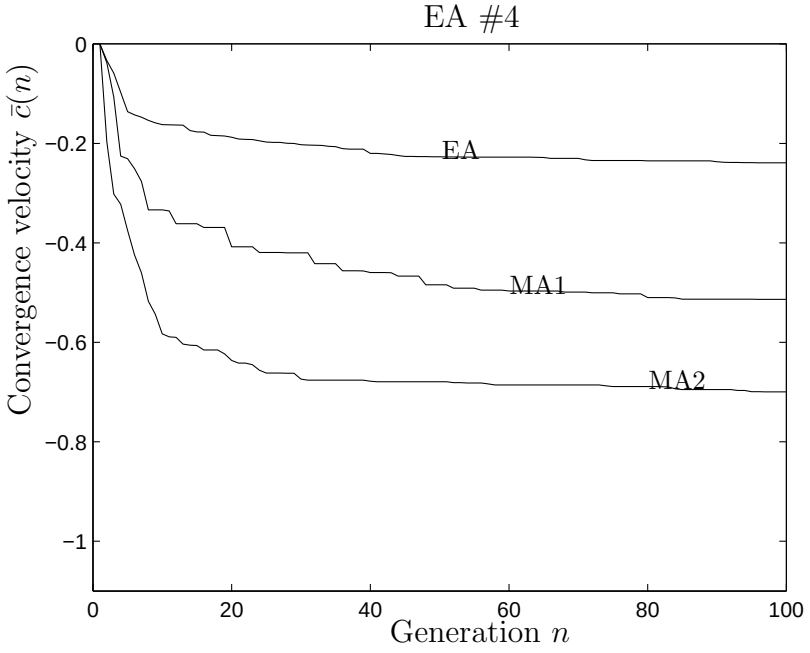

Figure 5.13: Convergence velocity for EA configuration #4.

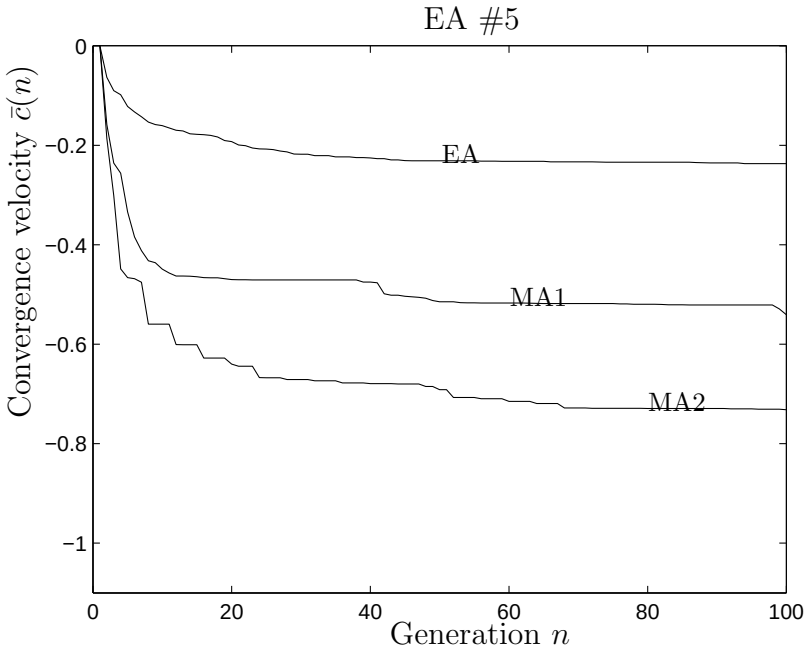

Figure 5.14: Convergence velocity for EA configuration #5.

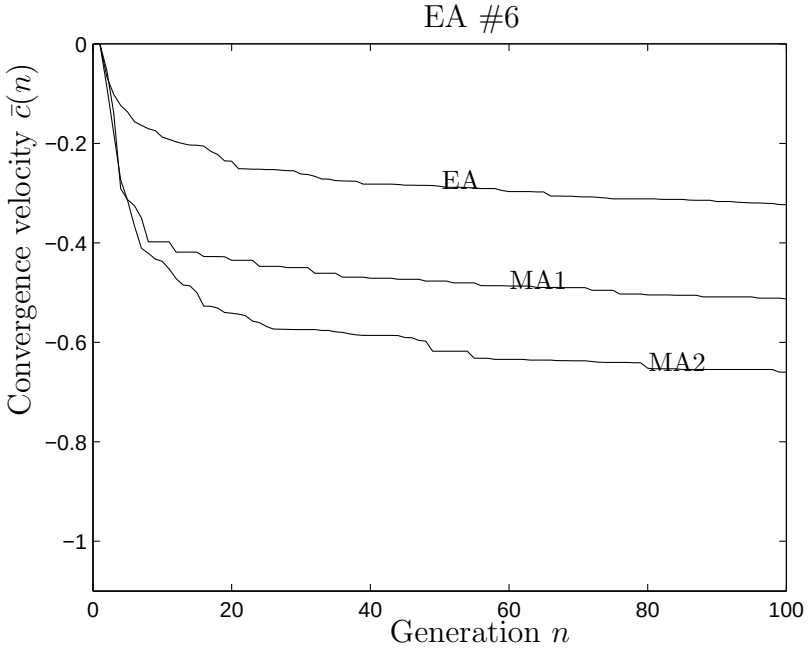

Figure 5.15: Convergence velocity for EA configuration #6.

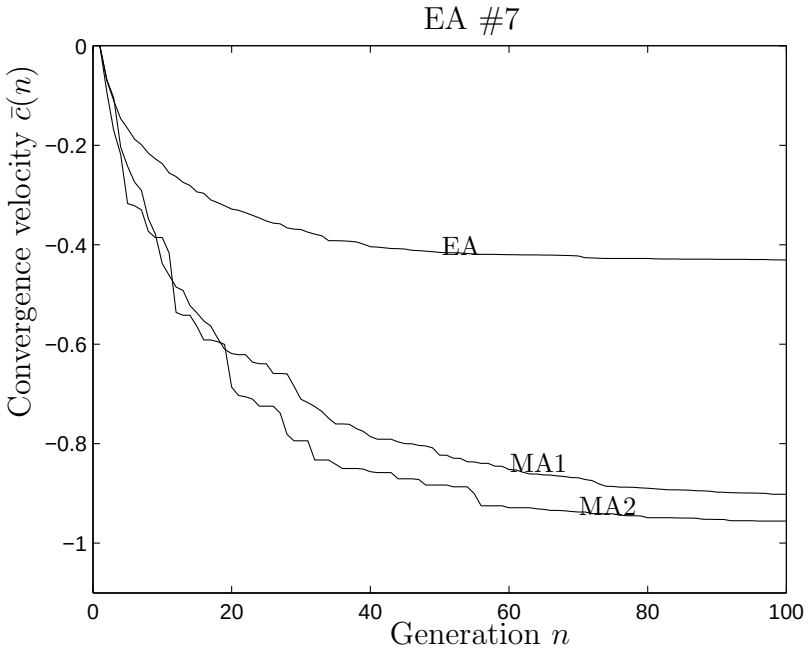

Figure 5.16: Convergence velocity for EA configuration #7.

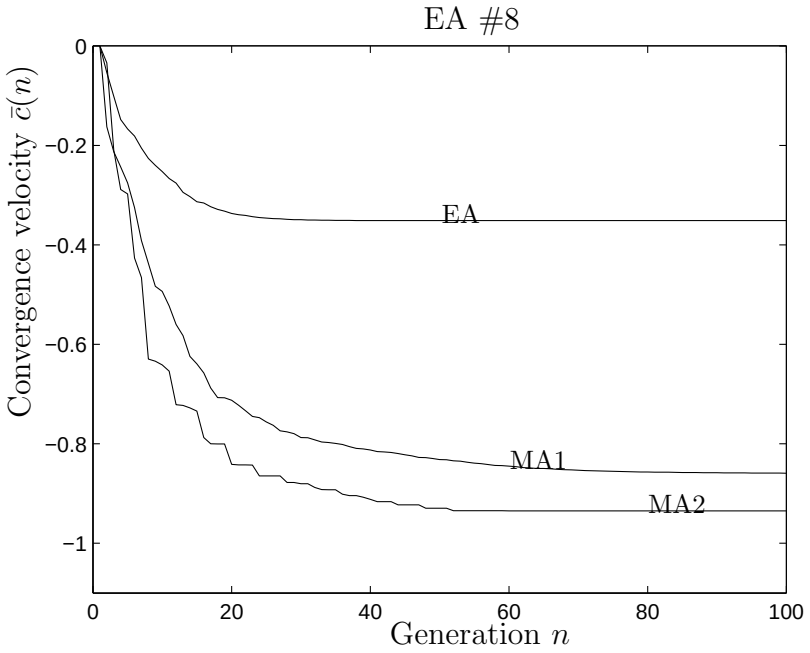

Figure 5.17: Convergence velocity for EA configuration #8.

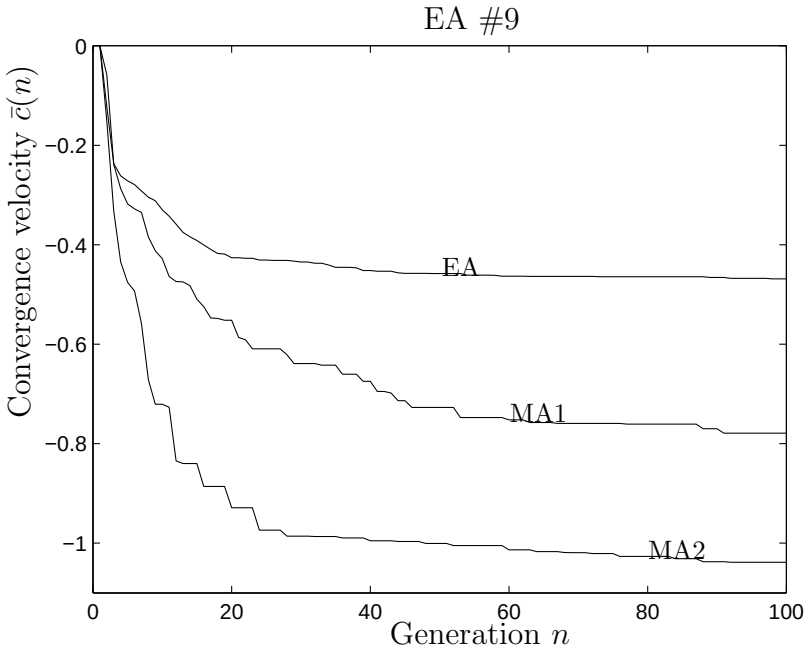

Figure 5.18: Convergence velocity for EA configuration #9.

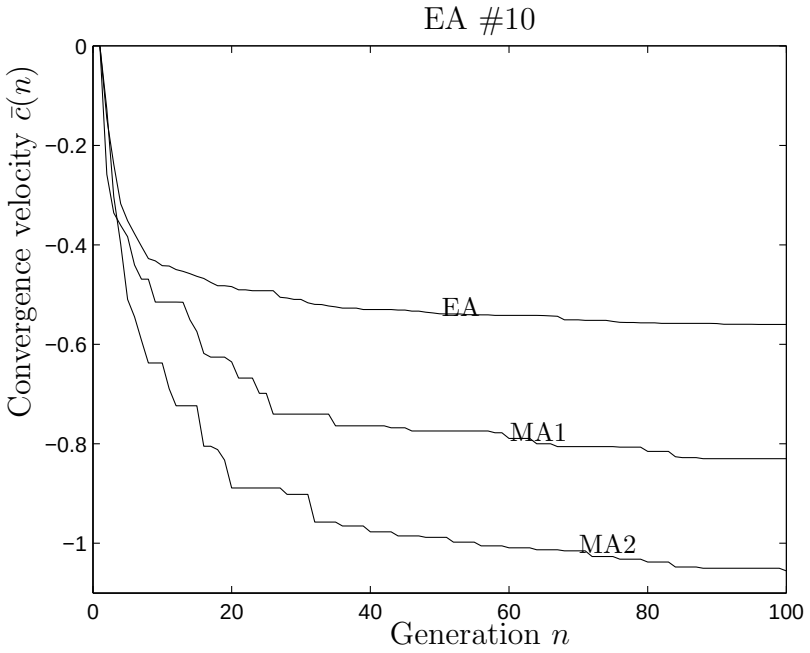

Figure 5.19: Convergence velocity for EA configuration #10.

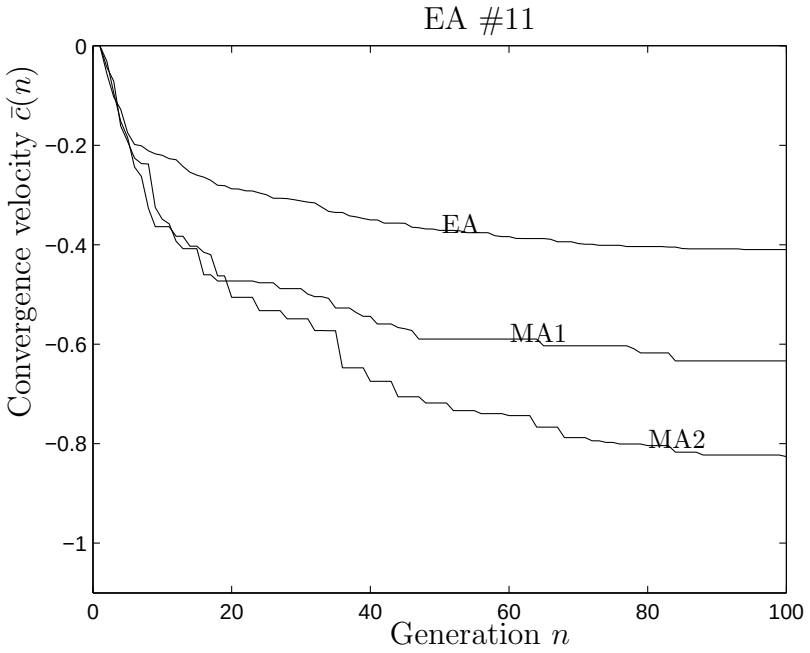

Figure 5.20: Convergence velocity for EA configuration #11.

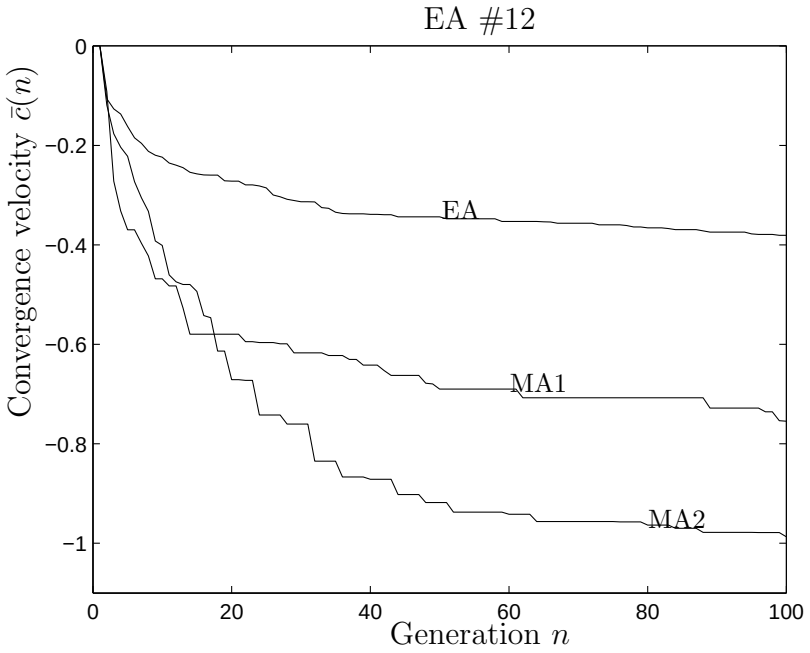

Figure 5.21: Convergence velocity for EA configuration #12.

Finally, we conclude this section with a brief discussion on the overhead of the local search. For MA1, we have the condition:

$$\langle N_H \rangle < \langle N_G \rangle \frac{\mu t_e}{(\mu t_e + t_{LS}/n_L)} \tag{5.15}$$

where t_e in problem 22 is 2.6 seconds. Thus, t_{LS}/n_L is negligible in comparison with μt_e, making the ratio above very close to one. The condition $\langle T_H \rangle < \langle T_G \rangle$ becomes $\langle N_H \rangle < \langle N_G \rangle$, that is, if the hybrid algorithm converges in less generations, it also converges in less time.

For MA2, the local search consumes some function evaluations, thus:

$$\langle N_H \rangle < \langle N_G \rangle \frac{\mu}{(\mu + \sigma n_{TR}/n_L)} = 0.941 \langle N_G \rangle \tag{5.16}$$

Observe that the overhead of MA2 is also small. When the local search is more intense, with a high product σn_{TR}, the overhead can be decreased by n_L, the frequency of the local search.

5.4.3 Multi-objective version

The multi-objective version of the problem 22 is stated as follows:

$$\begin{aligned} \min \mathbf{f}(\mathbf{x}) &= \begin{bmatrix} f_1(\mathbf{x}) = B_{stray} \\ f_2(\mathbf{x}) = |E - 180MJ|/180MJ \end{bmatrix} \\ \text{subject to: } &\begin{cases} \mathbf{x} \in \mathcal{X} \\ g(\mathbf{x}) = B_{\max} - 4.92T \leq 0 \end{cases} \end{aligned} \tag{5.17}$$

Here we compare MOGA, NSGA2 and SPEA2 with their hybrid versions. Each algorithm was executed 12 times on the multi-objective version of the problem 22. However, as discussed in Chapter 2 regarding the NFL theorems, the performance of multi-objective optimizers is itself a multi-objective problem. This issue has been the subject of some discussion in the literature, see for instance (Okabe, Jin and Sendhoff 2003, Zitzler, Thiele, Laumanns, Fonseca and Fonseca 2003, Knowles and Corne 2004) and there is no agreement upon the metric to evaluate performance of algorithms for multi-objective problems. In addition to the intrinsic difficulty of comparing multi-objective optimizers,

the true Pareto front of the problem 22 is unknown, thus reducing the set of metrics useful for this comparison.

In this experiment, we employ three different strategies to compare the quality and the speed of two algorithms, which are described next. Let $\mathcal{A}$ and $\mathcal{B}$ be the non-dominated sets produced by each algorithm, $\mathcal{C}$ = non-dominated points of $(\mathcal{A} \cup \mathcal{B})$, $\bar{\mathcal{A}}$ the subset of $\mathcal{C}$ that has been originated from $\mathcal{A}$, and $\bar{\mathcal{B}}$ the subset of $\mathcal{C}$ that has been originated from $\mathcal{B}$.

(1) **Attainment Function**

The attainment function (Fonseca, Fonseca and Hall 2001) provides a description of the distribution of the non-dominated set $\mathcal{A}$ using the notion of goal-attainment. It is defined by the function $\alpha_{\mathcal{A}}(\,\cdot\,) : \mathbb{R}^d \rightarrow [0, 1]$ with

$$\alpha_{\mathcal{A}}(z) = P(A_1 \leq z \vee A_2 \leq \vee \cdots \vee A_m \leq z) \tag{5.18}$$

where $P(\,\cdot\,)$ is the probability density function. The symbol $\vee$ denotes the logical *or*. The expression $\alpha_{\mathcal{A}}(z)$ corresponds to the probability of at least one element of $\mathcal{A}$ being smaller than or equal to a certain goal-vector, that is, the probability of an optimizer finding at least one solution which attains the goal-vector in a single run.

The attainment function can be estimated from a sample of n independent runs of an optimizer via the empirical attainment function (EAF) defined as

$$\alpha_n(z) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}(A_1 \prec z) \tag{5.19}$$

where A_i is the i-th run of the optimizer and $\mathbb{I}(\,\cdot\,)$ is the indicator function, which evaluates to one if its arguments is true and zero if its argument is false. In other words, the EAF gives for each objective vector in the objective space the relative frequency that it was attained, i.e, weakly dominated by an approximation set, with respect to n runs.

The outcomes $\mathcal{A}$ and $\mathcal{B}$ of two optimizers can be compared by performing a corresponding test on the resulting two EAFs. In order to test if two EAFs are different, the Kolmogorov-Smirnov (KS) test can be applied. This test measures the maximum difference between the EAFs and assesses the statistical significance of this

difference. For the comparison of two different algorithms, the following null and alternative hypothesis are formulated:

- *Null hypothesis:* there is no performance difference between the two algorithms;
- *Alternative hypothesis:* there is a performance difference between the two algorithms.

The test only determines if there is a *significant* difference between the two EAFs, based on the maximum difference. It does not determine whether one algorithm's entire EAF is above the other one or not.

An algorithm that computes the EAF and KS test, for bi-objective problems only, is available at *http://www.tik.ee.ethz.ch/sop/pisa*. For problems with more objectives, the computation of the EAF and KS test is an open subject, and there is no algorithm publicly available for this task.

(2) **Non-Dominated Combined Set Ratio (NDCSR)**

This is a simple measure that does not need the availability of the Pareto set. The NDCRS for the set $\mathcal{A}$ is equal to the number of elements in $\bar{\mathcal{A}}$ divided by the number of elements in $\mathcal{C}$.

(3) **S-Metric**

A definition of the S-metric is given in (Zitzler and Thiele 1999). It calculates the hyper-volume of the multidimensional region enclosed by $\mathcal{A}$ and a reference point, hence computing the size of the region that $\mathcal{A}$ dominates, see Figure 5.22. The basic idea is that the larger the volume dominated by the solution set $\mathcal{A}$ in the objective space is, the better this set is. This metric only requires the knowledge of some upper boundary of the region within which all the feasible points will lie. It requires a computational effort that grows fast with the dimension of the objective space, rendering it unusable for problems with many objectives.

(4) **Convergence velocity**

The convergence velocity of a multi-objective algorithm can be calculated using the values of the S-Metric above for the archive population at every generation.

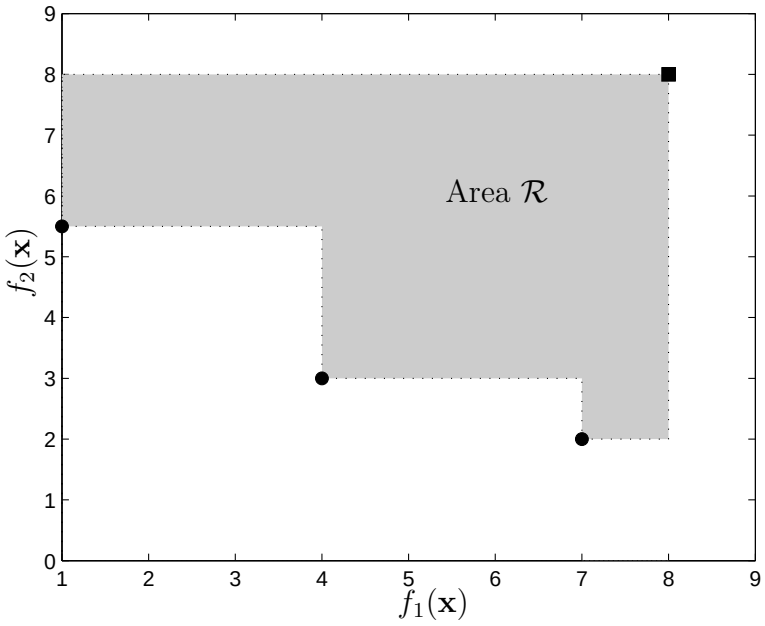

Figure 5.22: Illustration of the S-metric concept. The point ■ represents the reference point. The S-metric is maximized along the evolutionary process.

In this way, the convergence speed for the maximization of the S-metric can be measured. The expression for the convergence velocity is the same one presented for the mono-objective problems, but using the S-metric values.

The attainment function and KS test are used only to check if there is a significant difference between the outcomes of two algorithms.

The NDCSR metric is used to verify the expected effect of the local search in the evolutionary algorithm, namely to produce improved solutions. It is expected that the hybrid versions produce better solutions within the outcome set, thus leading to a higher NDSCR.

The S-metric is used only as a basis for obtaining convergence velocity curves. It is expected that the hybrid algorithm will converge faster and therefore maximize the S-metric more rapidly.

The multi-objective local search in the hybrid versions of MOGA, NSGA2 and SPEA2 uses $\sigma = 10$, $n_L = 4$, and $n_{TR} = 0$.

The KS test was applied over the nondominated sets generated by MOGA and MQ-

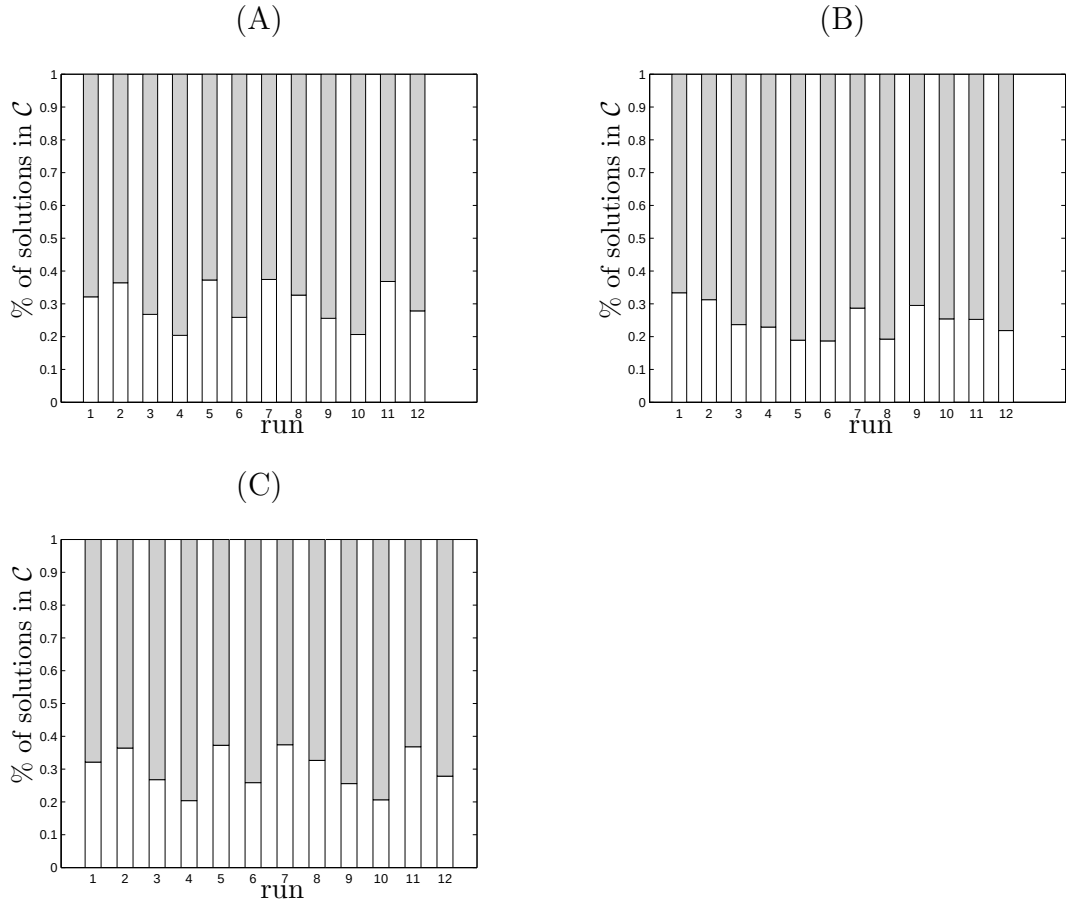

Figure 5.23: NDSCR metric for (A) MOGA and MQ-MOGA, (B) NSGA2 and MQ-NSGA2 and (C) SPEA2 and MQ-SPEA2. The gray bars show the proportion of solutions in $\mathcal{C}$ that come from the memetic algorithm.

MOGA, by NSGA2 and MQ-NSGA2, and by SPEA2 and MQ-SPEA2. All tests led to the rejection of the null hypothesis, which means that the local search produced a statistically significant difference between the outcomes.

Figure 5.23 shows the result for the NDSCR metric. As expected, the nondominated sets generated by the memetic algorithms contributed more points to $\mathcal{C}$, as an effect of the local search.

Finally, Figures 5.24-5.26 depict the S-metric convergence velocity curves for each algorithm. It is possible to notice that the hybrid versions of MOGA, NSGA2 and SPEA2 converged faster than its basic versions in the task of maximizing the S-metric.

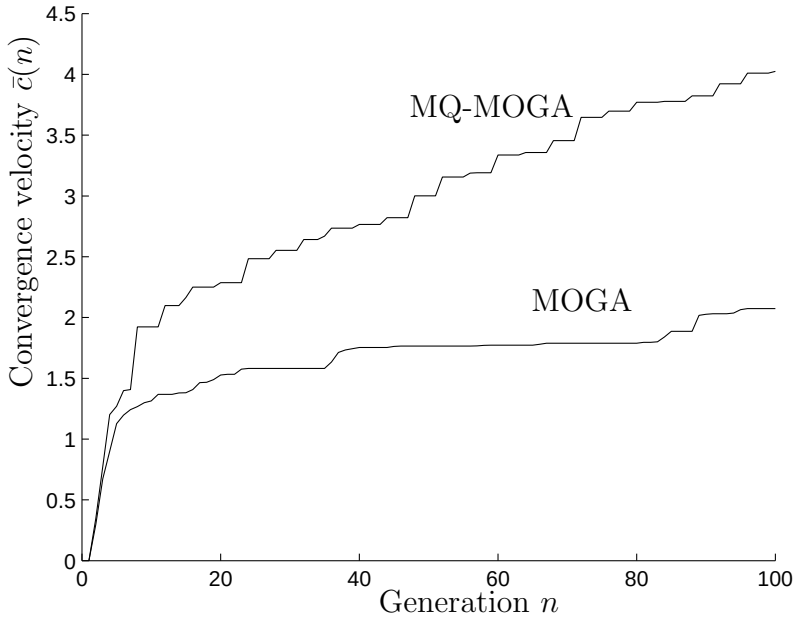

Figure 5.24: Convergence velocity for MOGA.

5.5 Discussion

The problems discussed in this Chapter show the applicability and powerfulness of the theory presented throughout the thesis. In particular, the results illustrate the benefit of the idea of coupling local approximations and local search methods to get a low-overhead approximation-based local search operator for memetic algorithms.

In general, the results obtained by the memetic algorithms confirm the expected decrease in the number of generations required to converge to good solutions, as discussed in Chapter 4. The metrics employed and the statistical tests performed show that the local search has an important effect in the search process of the evolutionary algorithm. Moreover, the practical problems in electromagnetics illustrate the low overhead of the local search due to the use of local approximations.

Finally, it is worth noting that even though the practical problems studied here are relatively fast to evaluate, the overhead of the local search is still negligible. It is reasonable to conjecture that in more complex problems that require more time for evaluating a single solution will benefit even more from the proposed methodology.

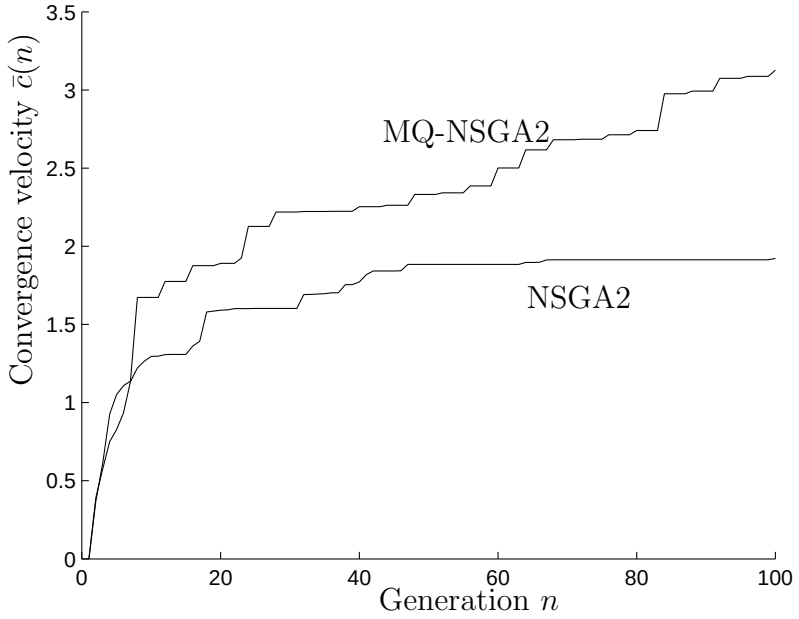

Figure 5.25: Convergence velocity for NSGA2.

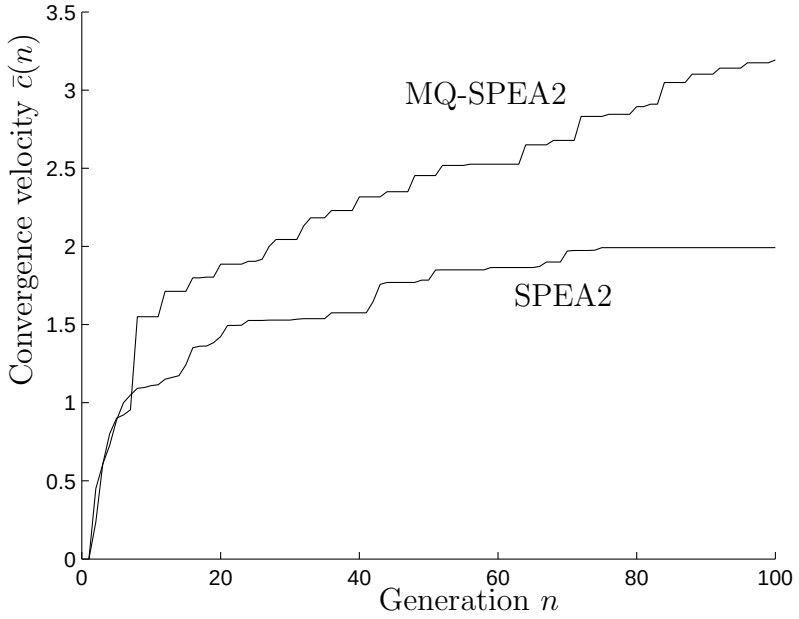

Figure 5.26: Convergence velocity for SPEA2.

Chapter 6

Conclusion

"Bon début promet bonne fin."
— French proverb

"The most educated person in the world now has to admit that he or she knows less and less, but at least knows less and less about more and more."
— Cristopher Hitchens, 1949–

6.1 Conclusions

In Chapter 1 a general introduction of evolutionary and memetic algorithms in CAD problems was given. The automated design process was presented as the combination of an analysis problem and a synthesis problem. The former is concerned with the simulation and analysis of the computational model being optimized. The solution of the analysis problem, characteristically involving the solution of a system of partial differential equations, provides the evaluation of points tested by the optimization algorithm. The synthesis problem is defined as an optimization problem instance, which is tackled with an adequate optimization algorithm:

$$\begin{gathered} \min \mathbf{f}\left(\mathbf{x}, \vec{R}(\mathbf{x})\right) \in \mathbb{R}^m \\ \text{subject to: } \mathbf{x} \in \Omega \end{gathered} \tag{6.1}$$

in which Ω is the feasible set, mathematically defined by:

$$\Omega = \left\{ \mathbf{x} \in \mathcal{X} : g_i\left(\mathbf{x}, \vec{R}(\mathbf{x})\right) \leq 0,\ i = 1, \ldots, p; \quad h_j\left(\mathbf{x}, \vec{R}(\mathbf{x})\right) = 0,\ j = 1, \ldots, q \right\} \tag{6.2}$$

The response of the computational model within the analysis problem for a given $\mathbf{x}$ is represented by $\vec{R}$. The objective and constraint functions in the optimization problem depend on the value of this response, which is performed by a computationally intensive method. In this scenario, the optimization algorithm must not only find the solution of (6.1) but also minimize the number of calls to the software involved in the calculation of $\vec{R}$. This is the typical scenario in practical CAD problems. Chapter 1 also reviewed the state of the art concerning the optimization with evolutionary and memetic algorithms and with the use of approximation techniques. The objectives of the thesis were also defined.

Chapter 2 began with the formal statement of the optimization problem, for both mono and multi-objective cases. An overview of evolutionary algorithms was also provided. This Chapter started from a general model of an evolutionary algorithm then moved to particular implementations of specific algorithms found in the literature. This presentation converged to a unified model of evolutionary algorithms, which is the core idea of the evolutionary framework developed in this thesis. A more general model is devised and presented in Algorithm 2.19, so that all evolutionary and memetic algorithms can be modeled. The framework implements different operators for the substitution, selection and variation steps, and different functions for fitness calculation. The combinations of these different operators and functions produce a specific evolutionary algorithm, including canonical algorithms in the literature.

Chapter 3 presented an overview of hybridization schemes, from the classical hybridization scheme to the Baldwinian and Lamarckian approaches. The multiquadric-based local search operator is also presented for mono and multi-objective problems.

The local search problem is defined as an approximated version of the global problem:

$$\begin{aligned} &\min \tilde{\mathbf{f}}(\mathbf{x}) \in \mathbb{R}^m \\ &\text{subject to: } \mathbf{x} \in \widetilde{\Omega} \cap \mathcal{V}(\mathbf{x}^{(c)}, \epsilon) \end{aligned} \tag{6.3}$$

in which an improved version of $\mathbf{x}^{(c)}$ is obtained within its local vicinity and by using local approximations. A trust region update method was also developed in Chapter 3, which allows the adaptation of the vicinity size and a more precise local search at the cost of additional n_{TR} calls to the expensive black-box functions.

Chapter 4 was dedicated to the formal analysis of memetic algorithms. This analysis was divided into two main parts. The first part of the Chapter investigated the effect of the local search operator on the global convergence properties of standard evolutionary algorithms via Markov chain theory. Evolutionary algorithms are stochastic processes which depend only on the previous state, and the state is represented by the populations $P(n)$ and $A(n)$. With this observation, we can treat evolutionary algorithms as Markov chains whose transition probabilities are defined by their operators. Therefore, the analysis is performed by studying the effect of the transition matrix generated by the local search operator on the properties of the overall transition matrix of the algorithm. The second part of the Chapter studied the computational cost of memetic algorithms. The computational complexity of the local search operators presented in Chapter 3 was derived. Finally, expressions for the overhead of the local search were developed based on the condition that $\langle T_H \rangle < \langle T_G \rangle$. By imposing this condition, we found the necessary relations such that the hybrid algorithm takes less time than the algorithm without local search.

The analysis in Chapter 4 led to two important results about the proposed local search methodology:

- The memetic algorithm preserves the global convergence properties of standard evolutionary algorithms. When the memetic algorithm employs Lamarckian local search and an irreducible mutation operator, it will be globally convergent if $\sigma < \mu$, i.e., if the number of individuals selected for local search is smaller than the number of individuals in the population.

- The memetic algorithm preserves the polynomial complexity of standard evolutionary algorithms. The complexity of the local search is dominated by the term $O((m+p+q)(d+1)N_L^2)$, therefore it is quadratic with the maximum number of

points N_L used to build the local approximation.

- The overhead of the local search can be reduced by using $n_L > 1$, which allows one to employ higher values for σ and n_{TR}. That is the balance between frequency and intensity of the local search.

Chapter 5 presented the results of the problems investigated. In total six problems were discussed. Three analytical test functions without constraints; one unconstrained magnetostatic problem, the design of the pole shape of a magnetizer device; and two versions of a constrained magnetostatic problem, the well known benchmark problem 22. The three analytical test functions were used to illustrate the increase in convergence speed due to the MQ-based local search. The magnetizer problem shows the negligible overhead of the approximation-based local search when dealing with computationally intensive functions. The analytical functions were fast to evaluate, hence the memetic algorithm took more time than the genetic algorithm, but this situation clearly inverts when the time to evaluate (t_e) increases. The mono-objective version of the problem 22 was used here as a representative instance of CAD problems in electromagnetic design. Twelve different evolutionary algorithms from the framework presented in Chapter 2 were hybridized with the MQ-based local search. The experiment shows that all tested evolutionary algorithms benefited from the use of the local search operator. This is empirical evidence that, for the class of problems delimited in this thesis, evolutionary algorithms in general can be greatly improved by their association with approximation-based local search operators. The bi-objective version of problem 22 was used to illustrate the multi-objective MQ-based local search operator. The results also show that all three multi-objective memetic algorithms performed better than their basic genetic versions. The local search has not only improved the convergence speed measured by the S-metric, but has also improved the quality of the outcome sets, regarding the metrics NDCSR and S-metric.

The practical problems investigated in Chapter 5 were still relatively fast. These problems, ranging from small to medium scale, consume only some seconds to evaluate a single trial solution. More complex real-world CAD problems, ranging from medium to large scale applications, could take minutes to hours to complete one evaluation, especially when dealing with nonlinear 3D problems. In this scenario, any new idea for saving computational effort is welcome, and memetic algorithms employing approximation-based local search operators are very promising. They can potentially reduce the number of generations to converge to good solutions, and they do that with

a very small overhead. For real-world problems that increase t_e to as long as minutes and hours, this additional overhead is totally negligible and any additional complexity in the evolutionary algorithm is fully justified.

The experiment with the analytical problems led to an interest result. For the Rastrigin problem, the local search had an important effect in the first generations of the process. As discussed in Chapter 5, this is possibly due to the filtering effect of the approximation. In the first generations, the samples available were not sufficient to capture the fine details of the Rastrigin function, and only a coarse behavior was modeled. In the case of the Rastrigin function, this coarse behavior is a quadratic one, making the local search benefit from this filtering characteristic. As more generations passed, more samples were obtained and the multiquadric method was able to model the fine details of the Rastrigin surface. The local search could not advance much, getting stuck in local minima. In this situation, an approximation technique would be more interesting than an exact interpolation method. The approximated model could be relaxed to filter oscillations in the function surface, thus exploiting only its global trend. On the other hand, an interpolation technique could be more precise, which is better in the end of the optimization process, when the population probably converged to a very good region in the search space. The difficulty here is how to take advantage of the approximation in one case and the interpolation in the other.

There is another class of problems in which this filtering characteristic would be very promising. It is the class of robust optimization problems defined as optimization problems where the objective and constraint functions are subject to noise. When dealing with noisy functions, memetic algorithms would not probably perform any better than evolutionary algorithms, because the local search method would get stuck in false minima created by noise. An approximation-based local search operator would not suffer from this hindrance, since the approximation could filter this noise, and capture the average behavior of the function. Therefore, the approximation-based local search methodology devised in this thesis could provide great improvements for evolutionary algorithms in the context of noisy optimization.

Another interesting possibility for the methodology in this thesis is its potential for parallelism. It may sound obvious, because evolutionary algorithms have always claimed to be appropriate for and worthy of parallel applications. Nevertheless, with the rise of multi-core processors, parallel programming has left the distant and expensive buildings in some universities and research centers, with all their demotivating protocols, to appear in personal computers. This fact will surely boost the use of parallel programming.

The overhead of the local search can be further reduced with the exploitation of this parallel capability now largely available. The application of the local search operator to more individuals in the population is more relevant in situations when a more diverse exploitation component is needed, as in multi-objective evolutionary algorithms, and in evolutionary algorithms that search for various optima in mono-objective problems. In these cases, it is useful to dedicate the local search effort to more individuals in the population, instead of applying it only to the best one.

Finally, it is worth commenting on the combination of local and global approximations. The use of local approximations does not represent the end of the use of global approximation methodologies. In fact, different approaches reviewed in the state of the art presented in Chapter 1 can be combined to further sophisticate the optimization process, namely ideas as fitness inheritance, local approximations for evaluating fitness of some individuals in the population, local approximations for the local search phase, and dynamic global approximations can be synergistically put together into a framework for optimization in CAD problems, with the clear goals of increasing the reliability of the optimization process and reducing its total time, in other words, satisfying industrial and practical needs.

6.2 Contributions of this thesis

In conclusion, the original contributions of this thesis are claimed again here:

- The organization and implementation of a framework of evolutionary techniques for CAD problems. The mono and multi-objective evolutionary algorithms are integrated into a unified algorithm. Therefore, all algorithms in the framework can be viewed as specific instances of this algorithm. Depending on the operators and parameters provided, this algorithm behaves as an evolution strategy, a genetic algorithm, a multi-objective genetic algorithm, etc. Having this unified algorithm is very useful for implementation purposes. It can model evolutionary algorithms such as genetic algorithms, evolution strategies, differential evolution, and also memetic algorithms.

- The proposition of the local search procedure that integrates radial basis function approximations and the sequential quadratic programming method for performing the local improvement of a given solution in the population of the evolutionary

algorithm. The procedure is also extended to multi-objective problems. It is also derived an optional trust region update methodology for the local search.

- The derivation of new theorem results for the effect of the local search on the global convergence properties of standard algorithms. The analysis of the computational complexity of the proposed local search is also developed. The analysis leads to an overall expression for the computational complexity that accounts for any number of objective and constraint functions and for any number of trust region updates. An important result obtained from this analysis is that the memetic algorithms implemented in this work preserve the polynomial complexity of conventional evolutionary algorithms.

- The application of the implemented framework in the optimization of electromagnetic devices. The magnetizer problem illustrates the unimportance of the overhead of the proposed local search in CAD problems. The mono and multi-objective formulations of the TEAM benchmark problem 22 are also used to test the framework and to verify the increase in the convergence speed of hybrid evolutionary algorithms.

6.3 Suggestions for future work

The following points summarize the suggestions for future work:

1. Comparison of Baldwinian and Lamarckian approaches in the class of problems under interest in this study. Although the Lamarckian approach is more common in optimization with real variables, it would be interesting to verify if there is any statistically significant difference between the two approaches. The Baldwinian approach will affect only stochastic selection mechanisms. It is not quite clear if the Lamarckian approach has a more important effect on diversity than the Baldwinian one. That is because the selection pressure within the selection operator can strongly reduce diversity, whereas some variation operators can spread solutions that are very concentrated. These nonlinear interactions among the operators are quite difficult to treat formally, and only through experiments can one draw meaningful conclusions.

2. Comparison of approximation techniques and interpolation techniques. The latter guarantee that the data points fit within machine precision, while the former do

not guarantee that the approximation obtained is exact on the data points. This could sound like a disadvantage, but in fact it can be advantageous. For example, in highly multimodal problems this relaxation in the approximation has the potential of filtering high frequence oscillations in the function landscape, hence capturing mainly its global trend. This effect was observed in the first generations of the MQMA used for the Rastrigin function. In situations when the objective and constraint functions have some noise, the use of an approximation method instead of an interpolation one is also recommended. On the other hand, the generation of approximation techniques likewise neural networks and neuro-fuzzy networks is itself an optimization problem, called in this context a learning problem. Therefore, the time taken to build the local approximations increases.

3. Still related to the previous item, it would be worth evaluating the difference in performance when simpler approximations, e.g. quadratic approximations, are used in place of approximations of higher order. In fact, one could develop a mixing strategy: simpler approximations are used in the beginning of the evolutionary process, when only the global trend is desired, and at the end of the process, we switch to more powerful approximations. In this way, a memetic algorithm based on different local approximations could be devised.

4. Use of other mathematical tools for the convergence analysis of memetic algorithms, in order to further investigate the effect of the local search operators in the evolutionary algorithm. For instance, the modelling of performance parameters of the memetic algorithm as state variables in dynamical systems.

5. Study of strategies for adapting the parameters of the local search, specifically n_L and n_{TR}. For example, it is not useful to spend most of the local search effort in the first generations, because this phase is rather exploratory. Thus the parameter n_{TR} can be increased in later generations, when a better local approximation is required and the exploitation component becomes more important.

6. Study of the tuning of the parameters of the local search via the design of specific statistical experiments. As already mentioned, the theoretical analysis of hybrid evolutionary algorithms is important but limited. The statistical analysis of well-designed experiments can help the understanding and configuration of these parameters.

Colophon

This thesis was created in LaTeX 2_ε using the "`hepthesis`" class produced by A. Buckley and available at:

http://www.ctan.org/tex-archive/macros/latex/contrib/hepthesis/hepthesis.pdf

The algorithms in this thesis were written using the "`algorithm2e`", a package for algorithms in LaTeX 2_ε produced by C. Fiorio and available at:

http://www.tug.org/tex-archive/help/Catalogue/entries/algorithm2e.html

Bibliography

Agapie, A.: 1998, Genetic algorithms: minimal conditions for convergence, *Proceedings of the Third European Conference on Artificial Evolution*, Vol. 1363 of *Lecture Notes in Computer Science*, pp. 183–193.

Almaghrawi, A. and Lowther, D. A.: 2007, Heterogeneity and loosely coupled systems in electromagnetic device analysis, *Proceedings of the 16th International Conference on the Computation of Electromagnetic Fields, June 24-28, 2007*.

Alotto, P., Caiti, A., Molinari, G. and Repetto, M.: 1996, A multiquadrics-based algorithm for the acceleration of simulated annealing optimization procedures, *IEEE Transactions on Magnetics* **32**(3), 1198–1201.

Alotto, P. and Nervi, M.: 2001, An efficient hybrid algorithm for the optimization of problems with several local minima, *International Journal of Numerical Methods in Engineering* **50**(4), 847–868.

Baumgartner, U., Magele, C. and Renhart, W.: 2004, Pareto optimality and particle swarm optimization, *IEEE Transactions on Magnetics* **40**(2), 1172–1175.

Bazaraa, M. S., Sherali, H. D. and Shetty, C. M.: 1979, *Nonlinear Programming: Theory and Algorithms*, John Wiley & Sons.

Bäck, T.: 1996, *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, New York.

Berry, A. and Vamplew, P.: 2003, A simplified artificial life model for multiobjective optimisation: a preliminary report, *Proceedings of the IEEE Congress on Evolutionary Computation*, Vol. 2, IEEE Press, pp. 1331–1339.

Bhattacharya, M. and Guojun, L.: 2003, DAFHEA: a dynamic approximate fitness-based hybrid EA for optimisation problems, *Proceedings of the IEEE Congress on Evolutionary Computation*, Vol. 3, IEEE Press, pp. 1879–1886.

Blackmore, S.: 2000, *The Meme Machine*, Oxford University Press.

Brandstätter, B. R. and Ring, W.: 1996, Optimization of a smes device under nonlinear constraints, *Proceedings of the Seventh International IGTE Symposium*, pp. 277–281.

Brooker, A. J., Dennis, J., Frank, P. D., Serafini, D. B., Torczon, V. and Trosset, M.: 1998, A rigorous framework for optimization of expensive functions by surrogates, *Structural Optimization* **17**, 1–13.

Bueche, D.: 2003, Accelerating evolutionary algorithms using fitness function models, *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 166–169. Workshop on Learning, Adaptation and Approximation in Evolutionary Computation.

Burke, E. K. and Smith, A. J.: 2000, Hybrid evolutionary techniques for the maintenance scheduling problem, *IEEE Transactions on Power Systems* **15**(1), 122–128.

Campelo, F., Guimarães, F. G. and Igarashi, H.: 2007, Overview of artificial immune systems for multi-objective optimization, *Proceedings of the Fourth International Conference on Evolutionary Multi-Criterion Optimization*.

Campelo, F., Guimarães, F. G., Igarashi, H. and Ramírez, J. A.: 2005, A clonal selection algorithm for optimization in electromagnetics, *IEEE Transactions on Magnetics* **41**(5), 1736–1739.

Campelo, F., Guimarães, F. G., Igarashi, H., Ramírez, J. A. and Noguchi, S.: 2006, A modified immune network algorithm for multimodal electromagnetic problems, *IEEE Transactions on Magnetics* **42**(4), 1111–1114.

Canova, A., Freschi, F. and Repetto, M.: 2005, Hybrid method coupling ais and zeroth order deterministic search, *COMPEL: The International Journal for Computation and Mathematics in Electrical and Electronic Engineering* **24**(3), 784–795.

Canova, A., Gruosso, G. and Repetto, M.: 2003, Magnetic design optimization and objective function approximation, *IEEE Transactions on Magnetics* **39**(5), 2154–2162.

Chun, J.-S., Kim, M.-K., Jung, H.-K. and Hong, S.-K.: 1997, Shape optimization of electromagnetic devices using immune algorithm, *IEEE Transactions on Magnetics* **33**(2), 1876–1879.

Coello, C. A. C.: 2002, Theoretical and numerical constraint handling techniques used with evolutionary algorithms: a survey of the state of the art, *Computer Methods in Applied Mechanics and Engineering* **191**(11–12), 1245–1287.

Coello, C. A. C.: 2006, Evolutionary multi-objective optimization: a historical view of the field, *IEEE Computational Intelligence Magazine* **1**(1), 28–36.

Coello, C. A. C., Pulido, G. T. and Lechuga, M. S.: 2004, Handling multiple objectives with particle swarm optimization, *IEEE Transactions on Evolutionary Computation* **8**(3), 256–279.

Coello, C. A. C., Veldhuizen, D. A. V. and Lamont, G. B.: 2002, *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers, New York.

Corne, D. W. and Knowles, J. D.: 2003, Some multiobjective optimizers are better than others, *Proceedings of the IEEE Congress on Evolutionary Computation*, Vol. 4, IEEE Press, pp. 2506–2512.

Corne, D. W., Knowles, J. D. and Oates, M. J.: 2000, The pareto envelope-based selection algorithm for multiobjective optimisation, *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pp. 839–848.

Costa, L. and Oliveira, P.: 2002, An evolution strategy for multiobjective optimization, *Proceedings of the IEEE Congress on Evolutionary Computation*, Vol. 1, IEEE Press, pp. 97–102.

Coulomb, J., Kobetski, A., Costa, M. C., Maréchal, Y. and Jönsson, U.: 2003, Comparison of radial basis function approximation techniques, *COMPEL: The International Journal for Computation and Mathematics in Electrical and Electronic Engineering* **22**(3), 616–629.

Czyzak, P. and Jaszkiewicz, A.: 1998, Pareto simulated annealing - a metaheuristic technique for multiple-objective combinatorial optimization, *Journal of Multi-Criteria Decision Analysis* **7**(1), 34–47.

Davey, K. R.: 2003, Examination of various techniques for the acceleration of multivariable optimization problems, *IEEE Transactions on Magnetics* **39**(3), 1293–1296.

Dawkins, R.: 1976, *The Selfish Gene*, Oxford University Press.

de Castro, L. N. and Timmis, J.: 2002, An artificial immune network for multimodal function optimization, *Proceedings of the IEEE Congress on Evolutionary Computation*, Vol. 1, pp. 669–674.

de Castro, L. N. and Von Zuben, F. J.: 2002, Learning and optimization using the clonal selection principle, *IEEE Transactions on Evolutionary Computation* **6**(3), 239–251.

Deb, K.: 2001, *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons, Chichester, UK.

Deb, K., Agarwal, S., Pratab, A. and Meyarivan, T.: 2000, A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II, *Technical Report 200001*, KanGAL, Indian Institute of Technology, Kanpur, India.

Deb, K. and Beyer, H.-G.: 2001, Self-adaptive genetic algorithms with simulated binary crossover, *Evolutionary Computation Journal* **9**(2), 197–221.

Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T.: 2002, A fast and elitist multiobjective genetic algorithm: Nsga-ii, *IEEE Transactions on Evolutionary Computation* **6**(2), 182–197.

Ebner, T., Magele, C., Brandstatter, B. R. and Richter, K. R.: 1998, Utilizing feedforward neural networks for acceleration of global optimization procedures, *IEEE Transactions on Magnetics* **34**(5), 2928–2931.

Engelbrecht, A. P.: 2005, *Fundamentals of Computational Swarm Intelligence*, Wiley.

Esquivel, S. C. and Coello, C. A. C.: 2003, On the use of particle swarm optimization with multimodal functions, *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1130–1136.

Farina, M. and Sykulski, J. K.: 2001, Comparative study of evolution strategies combined with approximation techniques for practical electromagnetic optimization problems, *IEEE Transactions on Magnetics* **37**(5), 3216–3220.

Fogel, D. B.: 1995, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, IEEE Press, New Jersey.

Fonseca, C. M. and Fleming, P. J.: 1993, Multiobjective genetic algorithms, *Proceedings of the IEE Colloquium on Genetic Algorithms for Control Systems Engineering*, pp. 6/1–5.

Fonseca, C. M. and Fleming, P. J.: 1998a, Multiobjective optimization and multiple constraint handling with evolutionary algorithms. i. a unified formulation, *IEEE Transactions on Systems, Man and Cybernetics, Part A* **28**(1), 26–37.

Fonseca, C. M. and Fleming, P. J.: 1998b, Multiobjective optimization and multiple constraint handling with evolutionary algorithms. ii. application example, *IEEE Transactions on Systems, Man and Cybernetics, Part A* **28**(1), 38–47.

Fonseca, V. G., Fonseca, C. M. and Hall, A. O.: 2001, Inferential performance assessment of stochastic optimisers and the attainment function, *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization, Lecture Notes in Computer Science*, Vol. 1993, Springer-Verlag, pp. 213–225.

Gallardo, J. A. and Lowther, D. A.: 1999, The optimization of electromagnetic devices using niching genetic algorithms, *COMPEL: The International Journal for Computation and Mathematics in Electrical and Electronic Engineering* **18**, 285–297.

Gen, M. and Cheng, R.: 1997, *Genetic Algorithms and Engineering Design*, John Wiley & Sons, New York.

Goldberg, D. E.: 1989, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Company, Reading, Massachusetts.

Greenwood, G. W. and Zhu, Q. J.: 2001, Convergence in evolutionary programs with self-adaptation, *Evolutionary Computation Journal* **9**(2), 147–157.

Grefenstette, J. J.: 1991, Lamarckian learning in multi-agent environments, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 303–310.

Guimarães, F. G.: 2004, *Investigação de novas abordagens para problemas de otimização com restrições de igualdade (portuguese)*, Master's thesis, School of Engineering, Federal University of Minas Gerais.

Guimarães, F. G., Barros, P. H. L. and Ramírez, J. A.: 2003, An object-oriented library based on computational intelligence techniques for optimization in electromagnetics, *IEEE Transactions on Magnetics* **39**(4), 2121–2124.

Guimarães, F. G., Campelo, F., Igarashi, H., Lowther, D. A. and Ramírez, J. A.: 2007, Optimization of cost functions using evolutionary algorithms with local learning and local search, *IEEE Transactions on Magnetics* .

Guimarães, F. G., Campelo, F., Saldanha, R. R., Igarashi, H., Takahashi, R. H. C. and Ramírez, J. A.: 2006, A multiobjective proposal for the TEAM benchmark problem 22, *IEEE Transactions on Magnetics* **42**(4), 1471–1474.

Guimarães, F. G., Lowther, D. A. and Ramírez, J. A.: 2007, Analysis of the computational cost of approximation-based hybrid evolutionary algorithms in electromagnetic design, *Proceedings of the 16th International Conference on the Computation of Electromagnetic Fields, June 24-28, 2007.*

Guimarães, F. G. and Ramírez, J. A.: 2006, Improving the design of clustered neural fuzzy models for optimization, *Journal of Inverse Problems in Science and Engineering* **14**(6), 609–621.

Guimarães, F. G., Wanner, E. F., Campelo, F., Takahashi, R. H. C., Igarashi, H., Lowther, D. A. and Ramírez, J. A.: 2006, Local learning and search in memetic algorithms, *Proceedings of the IEEE Congress on Evolutionary Computation*, IEEE Press, pp. 2936–2943. IEEE World Congress on Computational Intelligence.

Hardy, R. L.: 1971, Multiquadric equations of topography and other irregular surfaces, *Geophysical Research* **176**, 1905–1915.

Hardy, R. L.: 1990, Theory and applications of the multiquadric-biharmonic method: 20 years of discovery 1968-1988, *Computers Mathematics with Applications* **19**(8–9), 163–208.

Hart, W. E., Krasnogor, N. and Smith, J. E.: 2004, *Recent Advances in Memetic Algorithms*, Vol. 166 of *Studies in Fuzziness and Soft Computing*, Springer.

Hawking, S.: 2001, *The universe in a nutshell*, Bantam Books.

Hinton, G. E. and Nowlan, S. J.: 1987, How learning can guide evolution, *Complex Systems* **1**, 495–502.

Ho, S. L., Yang, S., Wong, H. C. and Ni, G.: 2003, A simulated annealing algorithm for multiobjective optimizations of electromagnetic devices, *IEEE Transactions on Magnetics* **39**(3), 1285–1288.

Horn, J., Nafpliotis, N. and Goldberg, D. E.: 1994, A niched pareto genetic algorithm for multiobjective optimization, *Proceedings of the First IEEE International Conference on Evolutionary Computation*, Vol. 1, IEEE Press, pp. 82–87.

Hu, X., Huang, Z. and Wang, Z.: 2003, Hybridization of the multi-objective evolutionary algorithms and the gradient-based algorithms, *Proceedings of the IEEE Congress on Evolutionary Computation*, Vol. 2, pp. 870–877.

Hui-Zhi, Y., Fa-Chao, L. and Cong-Man, W.: 2005, A density clustering based niching genetic algorithm for multimodal optimization, *Proceedings of the International Conference on Machine Learning and Cybernetics*, Vol. 3, pp. 1599–1604.

Ioan, D., Ciuprina, G. and Szigeti, A.: 1999, Embedded stochastic-deterministic optimization method with accuracy control, *IEEE Transactions on Magnetics* **35**(3), 1702–1705.

Iosifescu, M.: 1980, *Finite Markov Processes and Their Applications*, Wiley.

Ishibuchi, H. and Murata, T.: 1996, Multi-objective genetic local search algorithm, *Proceedings of the IEEE International Conference on Evolutionary Computation*, pp. 119–124.

Ishibuchi, H. and Murata, T.: 1998, A multi-objective genetic local search algorithm and its application to flowshop scheduling, *IEEE Transactions on Systems, Man, and Cybernetics - Part C* **28**(3), 392–403.

Ishibuchi, H. and Murata, T.: 1999, Local search procedures in a multi-objective genetic local search algorithm for scheduling problems, *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, Vol. 1, IEEE Press, pp. 665–670.

Ishibuchi, H., Yoshida, T. and Murata, T.: 2003, Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling, *IEEE Transactions on Evolutionary Computation* **7**(2), 204–223.

Ishikawa, T., Tsukui, Y. and Matsunami, M.: 1996, Optimization of electromagnetic devices using artificial neural network with quasi-newton algorithm, *IEEE Transactions on Magnetics* **32**(3), 1226–1229.

Jang, J. S. R., Sun, C. T. and Mizutani, E.: 1997, *Neuro-Fuzzy and Soft Computing*, Prentice Hall.

Jin, Y.: 2005, A comprehensive survey of fitness approximation in evolutionary computation, *Soft Computing: A Fusion of Foundations, Methodologies and Applications* **9**(1), 3–12.

Jin, Y., Olhofer, M. and Sendhoff, B.: 2002, A framework for evolutionary optimization with approximate fitness functions, *IEEE Transactions on Evolutionary Computation* **6**(5), 481–494.

Kelley, C. T.: 1999, *Iterative Methods for Optimization*, Vol. 18 of *Frontiers in Applied Mathematics*, Society for Industrial & Applied Mathematics.

Kennedy, J. and Eberhart, R. C.: 1995, Particle swarm optimization, *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948.

Kimura, S. and Konagaya, A.: 2003, High dimensional function optimization using a new genetic local search suitable for parallel computers, *Proceedings of the IEEE Congress on Evolutionary Computation*, Vol. 1, IEEE Press, pp. 335–342.

Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P.: 1983, Optimization by simulated annealing, *Science* **220**(4598), 671–680.

Knowles, J. D.: 2002, *Local-Search and Hybrid Evolutionary Algorithms for Pareto Optimization*, PhD thesis, Department of Computer Science, University of Reading.

Knowles, J. D.: 2006, ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems, *IEEE Transactions on Evolutionary Computation* **10**(1), 50–66.

Knowles, J. D. and Corne, D. W.: 1999, The pareto archived evolution strategy: a new baseline algorithm for pareto multiobjective optimisation, *Proceedings of the IEEE Congress on Evolutionary Computation*, Vol. 1, pp. 105–110.

Knowles, J. D. and Corne, D. W.: 2000a, Approximating the nondominated front using the pareto archived evolution strategy, *Evolutionary Computation Journal* **8**(2), 149–172.

Knowles, J. D. and Corne, D. W.: 2000b, M-PAES: a memetic algorithm for multiobjective optimization, *Proceedings of the IEEE Congress on Evolutionary Computation*, Vol. 1, IEEE Press, pp. 325–332.

Knowles, J. D. and Corne, D. W.: 2004, *Memetic algorithms for multiobjective optimization: issues, methods and prospects*, Vol. 166 of *Studies in Fuzziness and Soft Computing*, Springer, pp. 313–352.

Koza, J. R.: 1992, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press.

Krasnogor, N.: 2002, *Studies on the Theory and Design Space of Memetic Algorithms*, PhD thesis, University of the West of England.

Krasnogor, N. and Smith, J.: 2005, A tutorial for competent memetic algorithms: model, taxonomy, and design issues, *IEEE Transactions on Evolutionary Computation* **9**(5), 474–488.

Lebensztajn, L., Marretto, C. A. R., Costa, M. C. and Coulomb, J. L.: 2004, Kriging: a useful tool for electromagnetic device optimization, *IEEE Transactions on Magnetics* **40**(2), 1196–1199.

Lozano, M., Herrera, F., Krasnogor, N. and Molina, D.: 2004, Real-coded memetic algorithms with crossover hill-climbing, *Evolutionary Computation Journal* **12**(3), 273–302.

Luongo, C. A.: 1996, Superconducting storage systems: An overview, *IEEE Transactions on Magnetics* **32**(4), 2214–2223.

Madavan, N. K.: 2002, Multiobjective optimization using a pareto differential evolution approach, *Proceedings of the IEEE Congress on Evolutionary Computation*, Vol. 2, IEEE Press, pp. 1145–1150.

Malik, Z. and Rashid, K.: 2000, Comparison of optimization by response surface methodology with neurofuzzy methods, *IEEE Transactions on Magnetics* **36**(1), 241–257.

Mayley, G.: 1996, Landscapes, learning costs and genetic assimilation, *Evolutionary Computation Journal* **4**(3), 213–234.

Merz, P. and Freisleben, B.: 1997, Genetic local search for the TSP: new results, *Proceedings of the IEEE International Conference on Evolutionary Computation*, pp. 159–164.

Merz, P. and Freisleben, B.: 1999, A comparison of memetic algorithms, tabu search, and ant colonies for the quadratic assignment problem, *Proceedings of the IEEE Congress on Evolutionary Computation*, Vol. 3, IEEE Press, pp. 2063–2070.

Mitchell, M.: 1998, *An Introduction to Genetic Algorithms*, Bradford Books.

Mohammed, O. A.: 1999, *Optimal design of magnetostatic devices: the genetic algorithm approach and system optimization strategies*, Wiley Series in Microwave and Optical Engineering, John Wiley & Sons, New York, chapter 14, pp. 435–462.

Montes, E. M. and Coello, C. A. C.: 2005, A simple multimembered evolution strategy to solve constrained optimization problems, *IEEE Transactions on Evolutionary Computation* **9**(1), 1–17.

Moscato, P.: 1989, On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms, *Technical Report C3P Report 826*, Caltech Concurrent Computation Program.

Moscato, P.: 1999, *Memetic algorithms: a short introduction*, McGraw-Hill, New York, pp. 219–234.

Nam, D. and Park, C. H.: 2000, Multiobjective simulated annealing: a comparative study to evolutionary algorithms, *International Journal of Fuzzy Systems* **2**(2), 87–97.

Norris, J. R.: 1997, *Markov Chains*, Cambridge Series in Statistical and Probabilistic Mathematics, Cambridge University Press.

O. A. Mohammed, G. F. U.: 1997, A hybrid technique for the optimal design of electromagnetic devices using direct search and genetic algorithms, *IEEE Transactions on Magnetics* **33**(2), 1931–1934.

Okabe, T., Jin, Y. and Sendhoff, B.: 2003, A critical survey of performance indices for multi-objective optimisation, *Proceedings of the IEEE Congress on Evolutionary Computation*, IEEE Press, pp. 878–885.

Ong, Y. W. and Keane, A. J.: 2004, Meta-lamarckian learning in memetic algorithms, *IEEE Transactions on Evolutionary Computation* **8**(2), 99–110.

Ong, Y. W., Lim, M. H., Zhu, N. and Wong, K. W.: 2006, Classification of adaptive memetic algorithms: a comparative study, *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics* **36**(1), 141–152.

Ortiz-Boyer, D., Hervás-Martínez, C. and García-Pedrajas, N.: 2005, CIXL2: a crossover operator for evolutionary algorithms based on population features, *Journal of Artificial Intelligence Research* **24**, 1–48.

Park, J. and Sandberg, I. W.: 1991, Universal approximation using radial-basis-function networks, *Neural Computation* **3**(2), 246–257.

Parsopoulos, K. E. and Vrahatis, M. N.: 2002, Particle swarm optimization method for constrained optimization problems, *in* P. Sincak, J. Vascak, V. Kvasnicka and J. Pospicha (eds), *Intelligent Technologies - Theory and Applications: New Trends in Intelligent Technologies*, Vol. 76 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, pp. 214–220.

Price, K. V., Storn, R. M. and Lampinen, J. A.: 2005, *Differential Evolution: A Practical Approach to Global Optimization*, Natural Computing Series, Springer.

Ramirez-Rosado, I. J. and Dominguez-Navarro, J. A.: 2006, New multiobjective tabu search algorithm for fuzzy optimal planning of power distribution systems, *IEEE Transactions on Power Systems* **21**(1), 224–233.

Rashid, K., Ramírez, J. A. and Freeman, E. M.: 1999, Optimization of electromagnetic devices using computational intelligence techniques, *IEEE Transactions on Magnetics* **35**(5), 3727–3729.

Rashid, K., Ramírez, J. A. and Freeman, E. M.: 2000, Hybrid optimization in electromagnetics using sensitivity information from a neuro-fuzzy model, *IEEE Transactions on Magnetics* **36**(3), 1061–1065.

Ratle, A.: 2001, Kriging as a surrogate fitness landscape in evolutionary optimization, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* **15**, 37–49.

Regis, R. G. and Shoemaker, C. A.: 2004, Local function approximation in evolutionary algorithms for the optimization of costly functions, *IEEE Transactions on Evolutionary Computation* **8**(5), 490–505.

Renders, J.-M. and Flasse, S. P.: 1996, Hybrid methods using genetic algorithms for global optimization, *IEEE Transactions on Systems, Man and Cybernetics - Part B* **26**(2), 243–258.

Renner, G. and Eckart, A.: 2003, Genetic algorithms in computer aided design, *Computer Aided Design* **35**, 709–726.

Ross, B. J.: 1999, *A lamarckian evolution strategy for genetic algorithms*, Vol. 3, CRC Press, pp. 1–16.

Rudolph, G.: 1994, Convergence properties of canonical genetic algorithms, *IEEE Transactions on Neural Networks* **5**(1), 96–101.

Sareni, B., Krahenbuhl, L. and Nicolas, A.: 1998, Niching genetic algorithms for optimization in electromagnetics I. Fundamentals, *IEEE Transactions on Magnetics* **34**(5), 2984–2987.

Sastry, K., Goldberg, D. E. and Pelikan, M.: 2001, Don't evaluate, inherit, *Proceedings of Genetic and Evolutionary Computation Conference*, pp. 551–558.

Schaffer, J. D.: 1984, *Some Experiments in Machine Learning Using Vector Evaluated Genetic Algorithms*, PhD thesis, Vanderbilt University.

Seneta, E.: 1981, *Non-negative Matrices and Markov Chains*, Springer Series in Statistics, Springer-Verlag.

Shennan, S.: 2003, *Genes, Memes and Human History: Darwinian Archaeology and Cultural Evolution*, Thames & Hudson.

Srinivas, N. and Deb, K.: 1994, Multiobjective optimization using nondominated sorting in genetic algorithms, *Evolutionary Computation Journal* **2**(3), 221–248.

Starzýnski, J. and Wincenciak, S.: 1998, On the effective coupling of optimization algorithms to solve inverse problems of electromagnetism, *COMPEL: The International Journal for Computation and Mathematics in Electrical and Electronic Engineering* **17**(1), 160–165.

Storn, R. M. and Price, K. V.: 1997, Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces, *Journal of Global Optimization* **11**, 341–359.

Takahashi, R. H. C., Ramírez, J. A., Vasconcelos, J. A. and Saldanha, R. R.: 2001, Sensitivity analysis for optimization problems solved by stochastic methods, *IEEE Transactions on Magnetics* **37**(5), 3566–3569.

Takahashi, R. H. C., Saldanha, R. R., Dias-Filho, W. and Ramírez, J. A.: 2003, A new constrained ellipsoidal algorithm for nonlinear optimization with equality constraints, *IEEE Transactions on Magnetics* **39**(3), 1289–1292.

Takahashi, R. H. C., Vasconcelos, J. A., Ramírez, J. A. and Krahenbuhl, L.: 2003, A multiobjective methodology for evaluating genetic operators, *IEEE Transactions on Magnetics* **39**(3), 1321–1324.

Tsao, D. and Webb, J. P.: 2005, Construction of device performance models using adaptive interpolation and sensitivities, *IEEE Transactions on Magnetics* **41**(5), 1768–1771.

Turney, P.: 1996, How to shift bias: lessons from the baldwin effect, *Evolutionary Computation Journal* **4**(3), 271–295.

Vasconcelos, J. A., Saldanha, R. R., Krahenbuhl, L. and Nicolas, A.: 1997, Genetic algorithm coupled with a deterministic method for optimization in electromagnetics, *IEEE Transactions on Magnetics* **33**(2), 1860–1863.

Venkatraman, S. and Yen, G. G.: 2005, A generic framework for constrained optimization using genetic algorithms, *IEEE Transactions on Evolutionary Computation* **9**(4), 424–435.

Wanner, E. F., Guimarães, F. G., Takahashi, R. H. C., Lowther, D. A. and Ramírez, J. A.: 2007, Hybrid genetic algorithms using quadratic local search operators, *COMPEL: The International Journal for Computation and Mathematics in Electrical and Electronic Engineering* **26**(3), 773–787.

Wanner, E. F., Guimarães, F. G., Takahashi, R. H. C., Lowther, D. A. and Ramírez, J. A.: 2008, Multiobjective memetic algorithms with quadratic approximation-based local search for expensive optimization in electromagnetics, *accepted for IEEE Transactions on Magnetics* .

Wanner, E. F., Guimarães, F. G., Takahashi, R. H. C., Saldanha, R. R. and Fleming, P. J.: 2005, Constraint quadratic approximation operator for treating equality constraints with genetic algorithms, *Proceedings of the IEEE Congress on Evolutionary Computation*, Vol. 3, pp. 2255–2262.

Wolpert, D. H. and Macready, W. G.: 1997, No free lunch theorems for optimization, *IEEE Transactions on Evolutionary Computation* **1**(1), 67–82.

Yang, S. Y., Cardoso, J. R., Ho, S. L., Ni, P. H., Machado, J. M. and Lo, E. W. C.: 2004, An improved tabu-based vector optimal algorithm for design optimizations of electromagnetic devices, *IEEE Transactions on Magnetics* **40**(2), 1140–1143.

Yang, S. Y., Park, L.-J., Park, C. H. and Ra, J. W.: 1995, A hybrid algorithm using genetic algorithm and gradient-based algorithm for iterative microwave inverse scattering, *Proceedings of the IEEE International Conference on Evolutionary Computation*, Vol. 1, pp. 450–455.

Zhang, L. B., Zhou, C. G., Liu, X. H., Ma, Z. Q., Ma, M. and Liang, Y. C.: 2003, Solving multi objective optimization problems using particle swarm optimization, *Proceedings of the IEEE Congress on Evolutionary Computation*, Vol. 4, pp. 2400–2405.

Zhou, Z., Ong, Y. S. and Lim, M. H.: 2007, Memetic algorithm using multi-surrogates for computationally expensive optimization problems, *Soft Computing* (11), 957–971.

Zhou, Z., Ong, Y. S., Nair, P. B., Keane, A. J. and Lum, K. Y.: 2007, Combining global and local surrogate models to accelerate evolutionary optimization, *IEEE Transactions on Systems, Man, and Cybernetics – Part C: Applications and Reviews* **37**(1), 66–76.

Zielinski, K. and Laur, R.: 2006, Constrained single-objective optimization using particle swarm optimization, *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 443–450.

Zitzler, E., Laumanns, M. and Thiele, L.: 2001, SPEA2: improving the strength pareto evolutionary algorithm, *Technical Report 103*, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland.

Zitzler, E. and Thiele, L.: 1999, Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach, *IEEE Transactions on Evolutionary Computation* **3**(4), 257–271.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M. and Fonseca, V. G.: 2003, Performance assessment of multiobjective optimizers: an analysis and review, *IEEE Transactions on Evolutionary Computation* **7**(2), 117–132.