MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE

ALGORITMOS EVOLUTIVOS HÍBRIDOS COM DETECÇÃO DE REGIÕES PROMISSORAS EM ESPAÇOS DE BUSCA CONTÍNUOS E DISCRETOS

Alexandre César Muniz de Oliveira

Tese de Doutorado em Computação Aplicada, orientada pelo Dr. Luiz Antonio Nogueira Lorena.

INPE
São José dos Campos
2004

55.521.14(811.3)

OLIVEIRA, A. C. M.

ALGORITMOS EVOLUTIVOS HÍBRIDOS COM DETECÇÃO DE REGIÕES PROMISSORAS EM ESPAÇOS DE BUSCA CONTÍNUOS E DISCRETOS / A. C. M. Oliveira. – São José dos Campos: INPE, 2004.

79p. – (INPE-5522-TDI/519).

1. Algoritmos evolutivos híbridos. 2. Heurísticas. 3. Agrupamentos. 4. Algoritmos evolutivos paralelos.

Aprovada pela Banca Examinadora em cumprimento a requisito exigido para a obtenção do Título de Doutor em Computação Aplicada.

Dr. Stephan Stephany _____
Presidente

Dr. Luiz Antonio Nogueira Lorena _____
Orientador

Dr. José Carlos Becceneri _____
Membro da banca

Dr. André Carlos Ponce de Leon Ferreira de Carvalho _____
Membro da banca

Dr. Luiz Satoru Ochi _____
Membro da banca

Candidato: Alexandre César Muniz de Oliveira

São José dos Campos, 29 de julho de 2004.

"O Signore, fa' di me uno strumento della tua Pace."

S. Francesco

Dedico este trabalho aos meus pais.
Que todo o meu esforço possa retribuir a dedicação
e a confiança que eles sempre depositaram em mim.

AGRADECIMENTOS

Agradeço a Deus, acima de tudo.

Agradeço a meu orientador, Dr. Luiz Antonio Nogueira Lorena, pelo apoio, incentivo e orientação segura e amiga.

Agradeço a todos, pessoas e entidades, que direta ou indiretamente, colaboraram para a elaboração deste trabalho e, de modo especial:

Ao Instituto Nacional de Pesquisas Espaciais (INPE) e ao Laboratório Associado de Computação e Matemática Aplicada (LAC) pela oportunidade e apoio.

Aos professores do LAC/INPE, que muito contribuíram para a minha formação, especialmente, Dra. Sandra Sandri, Dr. Stephan Stephany, Dr. Airam Preto e Dr. José Demísio Simões da Silva.

Aos membros da banca examinadora pela disposição em analisar este trabalho.

Aos colegas professores do Departamento de Informática da Universidade Federal do Maranhão, em especial, a amiga Maria Auxiliadora Freire.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo auxílio financeiro.

À Fundação de Amparo à Pesquisa do Estado de São Paulo pelo apoio financeiro no projeto de pesquisa "Física dos Materiais num Ambiente de Memória Distribuída" (proc. 01/03100-9).

Ao meu pai, Pedro Alexandre, pelo incentivo e por acreditar em mim sempre.

À minha mãe, Marlene, pelo amor, pela dedicação, e por cada lágrima de saudade.

A meu irmão e minha cunhada, Fernando César e Nathália, sempre presentes enquanto estive ausente.

A minha namorada Isabela, quem eu tanto admiro, pelo amor, carinho, atenção, assistência, paciência e incentivo em tudo o que faço.

A todos os amigos estudantes, de hoje, de ontem e de sempre, em especial, Leonardo, Fabrício, Aditya, e ao casal Elcio e Ana Paula.

RESUMO

Este trabalho apresenta três estratégias para intensificação de busca em algoritmos evolutivos híbridos. Essas estratégias formam o núcleo de três abordagens: o Treinamento Populacional em Heurísticas (TPH), o *Evolutionary Clustering Search* (ECS) e o Algoritmo Paralelo Hierárquico e Adaptativo com Competição Justa (*APHAC*). O TPH emprega heurísticas específicas do problema na definição de *fitness*, guiando a população para regiões promissoras representadas por indivíduos que não podem ser melhorados por tais heurísticas. No ECS, um agrupamento iterativo trabalha simultaneamente ao processo evolutivo, gerando um conjunto de *clusters* que são referência para regiões supostamente promissoras. Um processo alternativo de intensificação de busca, chamado de assimilação, torna a busca mais agressiva em regiões enquadradas por *clusters*. O *APHAC* implementa uma competição justa entre indivíduos com diferentes perfis de *fitness*, mantidos separados em diferentes subpopulações, evoluindo em paralelo. Um ambiente evolutivo heterogêneo serve de suporte ao emprego de diferentes estratégias de busca em cada subpopulação, incluindo algoritmos de busca local aplicados a indivíduos da subpopulação elite. As três abordagens são aplicadas a vários problemas-teste, definidos em espaços de busca contínuos e discretos. Os resultados obtidos são comparáveis e até superiores a vários outros enfoques encontrados na literatura.

HYBRID EVOLUTIONARY ALGORITHMS WITH DETECTION OF PROMISING AREAS IN CONTINUOUS AND DISCRETE SEARCH SPACES

ABSTRACT

This work presents three strategies for exploitation in hybrid evolutionary algorithms. These strategies are the base of three approaches: Population Training Heuristic (called TPH), Evolutionary Clustering Search (*ECS*) and the Parallel Adaptive Hierarchical Fair Competition Genetic Algorithm (called *APHAC*). The TPH employs problem-specific heuristics for fitness evaluation, guiding the population to settle down in promising search areas where such heuristic would not yield further improvement. In the ECS, an iterative clustering scheme works concurrently with an evolutionary algorithm generating a set of clusters that are used as references to assumed promising search areas. An alternative exploitation mechanism, called assimilation, makes the search strategy more aggressive in the areas framed by cluster. Finally, the APHAC implements a fair competition scheme by segregating individuals with different fitness ranges in different evolving demes. A heterogeneous evolutionary environment allows to employ a different search strategy in each deme. For instance, a local search operator is applied to the individuals of the elite deme. These three approaches are applied to standard test problems, associated to continuous or discrete search spaces. The results obtained with these approaches are similar, or even better, than those found in literature.

SUMÁRIO

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE TABELAS

Pág.

LISTA DE SÍMBOLOS

$\#$	–	Gene indeterminado de um esquema
α	–	Limiar de rejeição adaptativo
β, $\overrightarrow{\beta}$	–	Parâmetro de desempenho de assimilação, vetor de parâmetros
$\delta_{cons}, \delta_{ncons}$	–	*Ranking* construtivo e não-construtivo
ε	–	Incremento do limiar de rejeição adaptativo
$\wp$	–	Métrica de distância qualquer
$\varphi^H, \varphi^H(s_k)$	–	Vizinhança heurística qualquer, gerada a partir de s_k
Θ	–	Conjunto de operadores evolutivos
π	–	Permutação qualquer de padrões
$\rho(m)$	–	Probabilidade de mutação
$\rho(i)$	–	Percentual de indeterminação (aparições de #) em um esquema
$\rho(e)$	–	Percentual da população considerada base (elite)
AT^i	–	Limiar de admissão do nível i
blx_α	–	Parâmetros de desempenho do cruzamento BLX
$\mathcal{B}$	–	Estratégia de busca associada a um *cluster*
c, c_i, c_i'	–	Centro de *cluster*, centro i antes e depois da assimilação
$\|C_t\|$	–	Número de *clusters* em uma t geração qualquer
c_V'	–	Melhor centro avaliado em uma trajetória
D	–	Número de níveis, subpopulações, processadores
d	–	Constante de equilíbrio do *ranking*
dx	–	Passo inicial de busca do *HJD*
$\mathcal{E}$	–	Expressão lógica, instância SAT
$\|\mathcal{E}\|$	–	Número de cláusulas de uma expressão lógica
$\mathcal{E}_i$	–	Uma cláusula i qualquer de uma expressão lógica
f	–	Função objetivo, função de avaliação, ou aptidão de indivíduos
$f_{inf}, f_{inf}^i, f_{sup}, f_{sup}^i$	–	Menor e maior valor de função objetivo, para um certo nível i
$\hat{f}_{ij}$	–	Valor de função mapeada de aptidão do indivíduo j no nível i
$\mathcal{G}(\pi)$	–	Grafo gerado por uma permutação qualquer
g	–	Função heurística
G_{max}, G_{min}	–	Máximo e mínimo valor que pode assumir a função g
H	–	Heurística qualquer, heurística de treinamento
I, J	–	Número de padrões e de itens de uma instância *MOSP/GMLP*
L_SP	–	Probabilidade de busca local
l	–	Tamanho de uma vizinhança qualquer
m	–	Número de vizinhanças verificadas em uma busca local
mpa	–	Máximo de pilhas abertas
$\mathcal{MC}$	–	Número máximo de *clusters*
$\mathcal{NS}$	–	Número de indivíduos selecionados ou atualizados
ns_k	–	Número de seleções ocorridas com o indivíduo s_k
n	–	Número de variáveis de um problema
P	–	População de indivíduos
$\|P\|$	–	Tamanho da população

P_α	–	População no tempo de evolução α
P_i	–	População do nível i
$\mathcal{P}$	–	Matriz problema, instância *MOSP/GMLP*
$\mathcal{PD}$	–	Pressão de densidade
$\mathcal{Q}^\pi$	–	Matriz de pilhas abertas de uma permutação π
r, r_t	–	Raio dos *clusters* na geração t
s_k, x	–	Indivíduo, solução candidata
s_{base}, s_{guia}	–	Indivíduos base e guia, usados na seleção base-guia
s_V	–	Melhor solução dentro de uma vizinhança
t, T	–	Número de gerações atual e máximo de gerações previstas
tpa	–	Tempo de pilhas abertas
x^*	–	Solução ótima
x_i^{sup}, x_i^{inf}	–	limites superior e inferior de uma variável i qualquer

LISTA DE SIGLAS E ABREVIATURAS

*CS	–	*Clustering Search*, associado a uma metaheurística qualquer
AA	–	Analisador de Agrupamentos do *ECS*
AE	–	Algoritmo Evolutivo do *ECS*
AE, AEH, AEP	–	Algoritmo Evolutivo, Híbrido, Paralelo
$\mathcal{AGC}$	–	Algoritmo Genético Construtivo
$\mathcal{AGC}^H$	–	$\mathcal{AGC}$ com treinamento populacional em heurísticas
AGCR	–	Algoritmo Genético Codificado em Real
AGPD	–	Algoritmo Genético com População Dinâmica
AI	–	Agrupador Iterativo do *ECS*
AMP	–	Algoritmo Memético Paralelo
AO	–	Algoritmo de Otimização local do *ECS*
APHAC	–	Algoritmo Paralelo, Hierárquico, e Adaptativo com Competição justa
ASAP	–	*Adaptive evolutionary algorithm for the SAtisfiability Problem*
$\mathcal{ATP}$	–	Algoritmo de Treinamento Populacional
BLX	–	*Blend Crossover*
BOX	–	*Block Order Crossover*
C ANSI	–	Linguagem C padrão American National Standards Institute
CFO	–	Chamadas à Função Objetivo
CHA	–	*Continuous Hybrid Algorithm*
COL	–	Método *Collective*
ECS	–	*Evolutionary Clustering Search*
Genocop	–	Projeto coordenado por Zbigniew Michalewicz
GMLP	–	*Gate Matrix Layout Problem*
GRASP	–	*Greedy Randomized Adaptive Search Procedures*
HFC	–	*Hierarchical Fair Competition*
HJD	–	Busca direta de *Hooke e Jeeves*
MCO	–	*Micro Canonical Optimization*
MOSP	–	*Minimization of Open Stacks Problem*
MPI	–	*Message-Passing Interface*
MSE	–	Média das melhores soluções encontradas
OCL	–	*OptQuest Callable Library*
PMX	–	*Partially-Matching Crossover*
PS	–	Percentual ou número de execuções bem-sucedidas
PC	–	*Personal Computer*
PBO	–	Problema Bi-Objetivo
SA	–	*Simulated Annealing*
SAT	–	Problema de *satisfabilidade*
SNM	–	Simplex de Nelder e Mead
SPAA	–	Seleção com Pressão Auto-Adaptativa
TE	–	Média dos tempos de execução, em segundos
TPH	–	Treinamento Populacional em Heurísticas

CAPÍTULO 1

INTRODUÇÃO

Os mecanismos naturais que promovem a evolução dos seres vivos podem ser considerados processos inteligentes. A maioria das pesquisas sobre evolução estuda o comportamento inteligente de seres vivos em coletividade buscando se auto-organizar para atingir um dado objetivo. Nestes termos, são incluídos na modelagem mecanismos de auto-organização, adaptação, evolução, competição e cooperação, dentre outros.

O termo *computação evolutiva* é bem mais apropriado para definir a classe à qual pertencem os chamados *algoritmos evolutivos* (AE's). Tais algoritmos são inspirados no processo evolutivo biológico e agregam métodos computacionais que direcionam uma busca estocástica em um espaço de soluções.

A seleção natural de Charles Darwin, as leis de hereditariedade de Gregor Mendel e a teoria da mutação do holandês Hugo Marie de Vries explicam boa parte de toda evolução dos seres vivos ocorrida ao longo de milhões de anos-vida na Terra (Uzunian e Birner, 2004). A computação evolutiva é um arcabouço inspirado nesses três processos que pode ser empregado na construção de algoritmos para simular os efeitos do decorrer de gerações sobre uma população de indivíduos que representem soluções candidatas para um dado problema.

Quando se fala em computação evolutiva se tem em mente as quatro abordagens mais famosas: algoritmos genéticos (Holland, 1975), programação genética (Koza, 1992), estratégias evolutivas (Rechenberg, 1965) e programação evolutiva (Fogel et al., 1965).

A computação evolutiva tem sido comumente associada a áreas tais como inteligência computacional (Pedrycz, 1997), aprendizado de máquina (Mitchell, 1997) e *softcomputing* (Zadeh, 1994). Em inteligência artificial, os AE's são geralmente classificados como uma alternativa de busca *heurística* em espaço de estados (Russell e Norvig, 2003).

Espaço de estados e espaço de soluções estão estreitamente ligados. Um conjunto de soluções pode ser considerado um conjunto de estados que representem atribuições de valores às variáveis da função objetivo do problema. O espaço de soluções, dependendo dos valores que cada variável possa assumir, pode ser finito ou infinito, enumerável ou não, discreto ou contínuo, sujeito ou não a restrições.

O processo de otimização tenta encontrar a melhor solução viável, considerando o objetivo do problema e o conjunto de restrições, caso haja. Todavia, nem sempre se

sabe se a melhor solução obtida é a solução ótima, nem tampouco quão próxima se está dela. Algoritmos aplicáveis a problemas dessa natureza são em geral chamados de algoritmos de busca. A otimização, portanto, é uma busca, em um espaço de soluções, por uma com menor custo (minimização) ou maior ganho (maximização). Uma busca é considerada heurística quando se utiliza algum conhecimento sobre o problema (ou sobre o espaço de soluções) para *tomar atalhos* e encontrar a melhor solução mais rapidamente.

A palavra heurística (do grego *heuriskein*) significa o ato de descobrir. É comumente empregada para expressar o conhecimento empírico usado pelos humanos para tomada de decisão. O termo heurística está associado ao conhecimento circunstancial, não verificável, nem matematicamente comprovável (Blum e Roli, 2003).

Em otimização, heurísticas são procedimentos aproximativos, i.e., algoritmos de busca capazes de encontrar soluções aproximadas (sub-ótimas) de boa qualidade em tempo computacional razoável. AE's são considerados heurísticas de busca (ou otimizadores heurísticos), pois utilizam-se do conhecimento empírico e também teórico para, combinando soluções consideradas de boa qualidade, gerarem soluções ainda melhores.

1.1 A metaheurística evolutiva

Uma metaheurística de busca é uma estratégia de alto-nível capaz de guiar outras heurísticas para produzir soluções de qualidade em tempo computacional admissível. As heurísticas guiadas por tais meta-estratégias podem ser procedimentos completos de busca ou podem apenas encapsular descrições de movimentos para transformar uma solução candidata em outra.

As metaheurísticas são flexíveis, genéricas e, portanto, adaptáveis a um grande número de aplicações. Em geral, são providas de uma ou mais estratégias para torná-las mais robustas e, assim, com maiores chances de encontrar a melhor solução.

Com perfil de metaheurística, se destacam o *recozimento simulado* (*simulated annealing* - SA) (Kirkpatrick et al., 1983), procedimento de busca gulosa adaptativa aleatória (*Greedy Randomized Adaptive Search Procedure* - GRASP) (Feo e Resende, 1989), busca tabu (Glover, 1996), sistema de colônia formigas (Dorigo e Gambardella, 1997), bem como toda a família de algoritmos evolutivos.

Os AE's, de um modo geral, têm atraído o interesse de pesquisadores no mundo inteiro devido a seus méritos, tais como:

a) são facilmente implementáveis;

b) são modulares e facilmente adaptáveis a qualquer tipo de problema, mesmo os mais complexos;

c) são métodos de otimização global, mais robustos a ótimos locais;

d) necessitam de pouca informação sobre o problema, basicamente, custo ou ganho de cada possível solução;

e) podem encontrar soluções aproximadas, de boa qualidade e, até mesmo, ótimas;

f) podem otimizar um grande número de parâmetros discretos, contínuos ou combinações deles;

g) realizam buscas simultâneas em várias regiões do espaço de busca (paralelismo inerente);

h) podem ser adaptados para encontrar várias soluções ótimas;

i) podem ser eficientemente combinados com outras heurísticas de busca.

As principais deficiências dos AE's têm sido igualmente alvo de pesquisas. Sempre que se propõem alternativas para contornar dificuldades inerente à construção de AE's eficientes, de certa forma, se contribui para a sua popularização e se incorporam novas facilidades que possibilitam aplicá-los a uma gama ainda maior de problemas. Dentre as principais dificuldades relacionados à construção de AE's, tem-se:

a) podem ser bem mais lentos que outros métodos por trabalharem com uma população de soluções;

b) dificuldade para se definir um mecanismo apropriado de codificação e avaliação de soluções em indivíduos (Mitchell, 1998);

c) dificuldade para se ajustar adequadamente os valores de seus parâmetros de desempenho (Eiben et al., 1999);

d) fracos mecanismos de intensificação de busca, ocasionando baixas taxas de convergência (Areibi, 2001; Birru et al., 1999; Yen e Lee, 1997).

Os AE's são realmente métodos lentos se comparados a algumas heurísticas de busca que trabalham com melhoramentos sucessivos em uma única solução (métodos não-populacionais), como o recozimento simulado e a busca tabu. Parte dessa má fama dos AE's tem procedência em aplicações desenvolvidas para problemas que poderiam

ser resolvidos satisfatoriamente através de outras heurísticas computacionalmente mais leves e até mesmo por meio de métodos exatos. Velocidade de execução costuma ser o preço que se paga por uma maior robustez.

A codificação é uma das etapas mais importantes na construção de AE's e tem sido, por isso, uma relevante área de pesquisa dentro da computação evolutiva. Detalhes sobre codificação e as demais etapas de construção de um AE estão no Capítulo 2.

Formas alternativas de ajuste de parâmetros também tem sido alvo de pesquisa, especialmente, por entender-se que manter fixos parâmetros como tamanho da população, taxa de mutação e cruzamento (também detalhados no próximo capítulo) nem sempre é uma boa estratégia. Controle determinístico, adaptativo e auto-adaptativo de parâmetros são formas alternativas comumente empregadas (Eiben et al., 1999).

O controle adaptativo, por exemplo, pode considerar certos indicadores populacionais (tamanho da população, qualidade média dos indivíduos, etc) e, baseado em *inferência fuzzy* (Zadeh, 1994), estabelecer dinamicamente os valores dos parâmetros evolutivos a cada geração (Oliveira e Nascimento, 1998).

1.2 Regiões promissoras e mecanismos de intensificação de busca

O Algoritmo genético (AG) de John H. Holland e colaboradores (Holland, 1975; De Jong, 1975; Goldberg, 1989), também conhecido como algoritmo genético canônico, é substancialmente diferente dos AE's atualmente reconhecidos como algoritmos genéticos. Alternativas têm sido propostas para contornar a falta de resultados satisfatórios para problemas do mundo real, complexos, altamente não-lineares, com forte interação entre as variáveis.

Como alternativa de melhoria de desempenho, os AG's têm sido equipados com outros algoritmos de busca mais especializados que consideram a natureza do problema e/ou informações sobre o espaço de busca. Pode-se dizer que tais algoritmos especializados se constituem em um conjunto de heurísticas específicas para um dado problema. A grande maioria dos trabalhos recentes apenas preserva a essência da "computação inteligente baseada na evolução natural", mas utiliza uma grande variedade de heurísticas específicas para o problema em questão.

Neste trabalho, um algoritmo evolutivo híbrido (AEH) é definido como um algoritmo evolutivo que empregue alguma heurística de busca local, específica para o problema em enfoque, paralelamente ao processo evolutivo, como forma de acelerar a busca. Por heurística de busca local, entende-se um algoritmo capaz de fazer melhoramentos

sucessivos em uma solução até que seja encontrada a melhor solução dentro de uma certa localidade, definida pelo algoritmo.

A natureza continua sendo a inspiração para construção de modelos ainda flexíveis e facilmente adaptáveis, apesar do conhecimento específico incorporado. Prevalece assim a idéia de construção de modelos genéricos onde certas especificidades são encapsuladas em componentes de comportamento bem definido.

O conceito de *regiões promissoras* vem sendo associado ao emprego de estratégias capazes de identificar o potencial positivo de subespaços de busca(Chelouah e Siarry, 2003; Scott et al., 2002). Soluções, que representem tais regiões promissoras, devem ser, portanto, identificadas, segundo alguma estratégia, e posteriormente examinadas (intensificação de busca) por componentes heurísticos específicos.

A maioria dos AEH's traz como essência a idéia de que o processo evolutivo é apenas parte do processo global de busca. AG's, por exemplo, podem ficar responsáveis pela exploração do espaço de busca como um todo, gerando soluções representativas de regiões promissoras. Componentes heurísticos, paralelamente, ficam responsáveis por realizar uma *intensificação de busca* em tais regiões promissoras.

A estratégia para se detectar regiões promissoras se torna importante para o desempenho de um AEH quando a heurística de busca local tem um custo computacional relativamente alto. Aplicar tais heurísticas indiscriminadamente a todos os indivíduos de uma população durante centenas de gerações pode tornar o processo de busca impraticável.

O cenário dos AEH's vem reforçar ainda mais o paralelismo intrínseco dos algoritmos evolutivos, pois exploração e intensificação são processos que podem ocorrer independentemente, com identificação dinâmica de regiões promissoras e posterior atribuição a diferentes subpopulações. Além disso, à medida em que aumenta a complexidade dos componentes heurísticos, o processamento paralelo pode se tornar um importante aliado. O foco recente de tais métodos são problemas reais, que surgem na automação industrial, economia, engenharia, sistemas dinâmicos, geoprocessamento e que tendem a consumir uma grande quantidade de recursos computacionais (Hawick et al., 2003).

Existem várias estratégias para identificar regiões promissoras, possibilitando limitar a aplicação de busca local a certo grupo de indivíduos da população. Um critério bastante utilizado é o elitismo. De um modo geral, apenas um certo percentual da população, considerado de *indivíduos-elite*, sofre melhorias heurísticas (Yen e Lee, 1997; Navarro et al., 1999).

A idéia por trás do elitismo é que pequenas alterações em um indivíduo muito bom, podem deixá-lo ainda melhor. Além disso, o custo computacional necessário para melhorar um *indivíduo não tão bom* seria proibitivo. Um típico problema com o elitismo é que os *indivíduos-elite* podem estar mal distribuídos no espaço de busca, concentrados em poucas regiões e, dessa forma, a intensificação de busca convergiria inevitavelmente para as mesmas melhores soluções.

Outro critério é o probabilístico: a responsabilidade de se selecionar quais indivíduos são os mais promissores é entregue ao acaso. Assim, apenas uma parte aleatória da população é melhorada através de busca local (Birru et al., 1999). O critério probabilístico permite diversificar um pouco mais a busca local, o que pode ser desejável pois intensifica a busca em regiões que aparentemente não seriam tão promissoras, mas que poderiam ser particularmente interessantes para o processo de busca global. Em contrapartida, o acaso também faz com que indivíduos realmente promissores não venham a ser melhorados.

Existem outros critérios de detecção que, como esses, servem de inspiração para este trabalho e, por isso, são detalhados no Capítulo 2.

1.3 Motivação e relevância do trabalho

A computação evolutiva é uma área de pesquisa seguramente promissora. Existe uma grande variedade de contextos nos quais AE's podem ser aplicados. Alguns deles: planejamento de atividades, projetos de sistemas de distribuição de energia elétrica, apontamento de satélites, projetos de computadores e de *chips*, roteamento de veículos, alocação de tarefas ou trabalhadores ou máquinas, empacotamento de caixas em *containers*, seqüenciamento e corte de padrões, seqüenciamento genético.

Os algoritmos evolutivos híbridos têm dado respostas satisfatórias aos práticos e sugerido perguntas aos teóricos. A pesquisa pura e a pesquisa aplicada vivem de perguntas e respostas. Podem ser encontrados vários exemplos de empreendimentos ao redor do mundo que nasceram de pesquisa relacionada à computação evolutiva.

A OptTek, por exemplo, é uma companhia de desenvolvimento e de serviços de *software*, fundada por Fred Glover, pai e entusiasta da busca tabu, James P. Kelly e Manuel Laguna, em 1992. Os *softwares* da OptTek integram o estado-da-arte em termos de metaheurísticas, incluindo busca tabu, redes neurais e busca scatter (`http://www.opttek.com/company/index.html`).

Um outro exemplo de pesquisa que se voltou a software e serviços na área de otimização é a polonesa Nutech, fundada por Matthew Michalewicz, seu pai, Zbigniew Michalewicz

que é coordenador do projeto Genocop, e um terceiro sócio Dan Cullen. A missão da NuTech é desenvolver produtos, voltados para a área econômica, que possam avaliar variáveis dinâmicas em tempo real, identificar novas tendências e aprender com as experiências passadas de forma que o desempenho futuro possa ser otimizado. Seus produtos fornecem informação gerencial que podem evitar gargalos de produção, atrasos de entrega, decisões equivocadas de crédito, e perda de oportunidade de mercado. Algoritmos genéticos, redes neurais, programação evolutiva, sistemas de colônia de formigas, computação DNA e quântica formam um pouco do estado-da-arte que compõem os produtos da NuTech (`http://www.nutechsolutions.com`).

A empresa norueguesa Interagon AS provê soluções para a indústria de tecnologia farmacêutica, em áreas como farmacogenéticos, medicina personalizada, e alvos específicos (*drug target discovery*). Tecnologia patenteada de busca é usada para automatizar a análise de banco de dados biológicos, filtrando informação relevante. Parte desse processo é examinado por um software baseado em algoritmos evolutivos (programação evolutiva) empregando mutação aleatória, seleção Darwiniana de expressões, etc. A combinação dessas tecnologias aumentam o desempenho em ordens de magnitude acima de outras tecnologias disponíveis, baseados em procedimentos estatísticos (`http://www.interagon.com/technology.htm`).

Todos esses exemplos e outros tantos existentes ao redor do mundo são motivadores para se centrar esforços em algo que possa não só sugerir perguntas e investigação, como também construir as respostas e resultados, inserindo a pesquisa acadêmica no ciclo virtuoso de investimentos que provêem da atual demanda social por tecnologia.

1.4 Objetivos e contribuições

Este trabalho se propõe a contribuir especificamente no que diz respeito a estratégias para intensificação de busca em algoritmos evolutivos híbridos aplicados a problemas de otimização com espaços de busca tanto discretos quanto contínuos. Para tanto, são apresentadas três abordagens de algoritmo evolutivos híbridos providos de mecanismos de detecção de regiões promissoras do espaço de busca.

A primeira das abordagens é o Treinamento Populacional em Heurísticas (TPH) que trabalha com o conceito de adaptação à heurística de treinamento para ordenar indivíduos em uma população de tamanho variável. Os indivíduos mais adaptados à heurística de treinamento são considerados representantes de regiões promissoras e, em virtude disso, participam mais efetivamente do processo evolutivo.

O *Evolutionary Clustering Search* (*ECS*) ou Busca Evolutiva através de Agrupamentos agrega uma segunda proposta de detecção de regiões promissoras baseada na freqüência com que são amostrados indivíduos em certos subespaços de busca, identificados dinamicamente. O ECS trabalha com um conjunto de soluções de referência que representam as regiões supostamente promissoras.

Na terceira abordagem, o modelo hierárquico de competição justa (HFC), proposto em (Hu et al., 2002), é empregado para separar indivíduos em subpopulações com diferentes perfis de avaliação ao longo do processo evolutivo. Através dessa estratificação, cria-se um ambiente evolutivo heterogêneo, com estratégias de busca adequadas a cada perfil de indivíduos. Essa abordagem, baseada no HFC, é chamada de Algoritmo Paralelo, Hierárquico, e Adaptativo de Competição Justa (*APHAC*).

As abordagens apontam para uma direção de pesquisa que visa contribuir para o desenvolvimento de algoritmos flexíveis, adaptativos, aplicáveis a problemas de natureza diferentes, robustos, competitivos, eficientes tanto em termos de qualidade de soluções quanto em tempo computacional para obtê-las. Espera-se contribuir ainda com novos modelos de estruturação de AEH's, com grande potencial de paralelismo, novos operadores evolutivos e mecanismos de ajuste de seus parâmetros.

1.5 Organização da tese

Este documento está organizado em 6 capítulos, sendo este o primeiro e os demais são como se segue. No Capítulo 2, é traçado um paralelo entre evolução e otimização, enfatizando conceitos necessários ao entendimento deste trabalho, bem como os aspectos teóricos relevantes dos problemas para os quais foram construídas aplicações. Os capítulos 3, 4 e 5 concentram-se em cada uma das três abordagens, respectivamente, o TPH, o ECS e o *APHAC*. Nesses capítulos são apresentados seus respectivos resultados computacionais e comparados a outros enfoques encontrados na literatura. No capítulo 6, as principais contribuições são resumidas e novas direções de pesquisa, resultado do presente trabalho, são apresentadas.

CAPÍTULO 2

OTIMIZAÇÃO ATRAVÉS DA EVOLUÇÃO

Neste capítulo, é traçado um paralelo entre evolução e otimização. Os principais conceitos e definições, julgados necessários ao entendimento deste trabalho e à leitura dos demais capítulos, são brevemente abordados. Por fim, este capítulo também concentra os aspectos teóricos relevantes dos problemas para os quais foram construídas aplicações baseadas nas três abordagens, apresentadas posteriormente.

2.1 Seleção, cruzamento e mutação

A seleção natural é o mecanismo pelo qual o meio ambiente influencia o processo de evolução. O meio ambiente pode ser entendido como o conjunto de fatores sociais, biológicos, geográficos, climáticos, dentre outros, que determinam regras de sobrevivência e competição entre indivíduos dentro de uma mesma região geográfica. Na computação evolutiva, tudo o que interage com a população é simplificadamente considerado meio ambiente.

Os organismos com maior *aptidão* para viver em determinado meio ambiente são *selecionados*, por seus próprios méritos, a permanecerem vivos e se multiplicarem. Entretanto, o ambiente natural está em constante mudança. Nesse ponto os conceitos de aptidão e adaptação se separam. A adaptação de um indivíduo mede a sua capacidade de se adequar às constantes mudanças que tendem a ocorrer no meio ambiente. A aptidão, por sua vez, mede a adaptação de um indivíduo a determinado meio.

As leis de hereditariedade e dominância explicam questões relativas à genética da formação dos organismos. Um indivíduo, posto em termos de cromossomos, pode ser visto como uma dualidade de seu *genótipo* e *fenótipo*. O primeiro é o código genético propriamente dito, ou seja, todo o projeto genético codificado em uma cadeia de genes. Sabe-se que cada gene é um trecho de uma macro-molécula de DNA (*Deoxyribonucleic acid*), que contém o código para elaboração de determinada proteína. O fenótipo, por sua vez, é a expressão das características físicas observáveis do indivíduo. A dominância de *genes* diz que traços exteriores são resultados de combinações genéticas interiores, onde certas combinações dominam outras no processo de confecção do fenótipo.

A reprodução é uma propriedade óbvia das espécies, na qual o potencial reprodutivo é proporcional à aptidão de cada indivíduo. O genótipo determina o fenótipo que por sua vez determina a aptidão e a adaptação. Esses dois, por último, determinam a capacidade de reprodução.

Com a seleção natural e as leis *mendelianas* de cruzamento, o processo evolutivo tende a convergir para o melhor indivíduo permitido pelo conjunto de genes presentes na espécie. A mutação genética explica o aparecimento de novas características totalmente desconhecidas na espécie. Ela pode ocorrer tanto espontaneamente, devido erros de cópia de DNA durante o processo de reprodução, como pode também pode ser induzida pelos chamados agentes mutagênicos, como agentes químicos, radiação, etc.

A mutação introduz características novas que podem ser benéficas ou maléficas à adaptação do indivíduo ao meio em que ele vive. No caso benéfico, o indivíduo tende a se sobressair, aumentando a sua capacidade reprodutiva e, conseqüentemente, transmitindo esse material genético novo a um número maior de descendentes. No caso maléfico, esse indivíduo pode ter uma vida breve, limitando o efeito do novo material a um número pequeno de indivíduos. A longo prazo, pequenas mutações podem causar mudanças substanciais na população, incluindo, em tese, formação de novas espécies.

Pela lei do uso e desuso de Jean-Baptiste Lamarck (*Teoria Lamarquiana*), o meio ambiente provocaria a criação de novas características que seriam herdadas pelos descendentes dos seres modificados (Lamarck, 1914). Entretanto, o que parece ser a indução do meio ambiente, na verdade, é uma espécie de filtro ao que foi induzido por outros fatores.

A teoria mais aceita é que a seleção natural funciona como um grande filtro atuando sobre os indivíduos e espécies, determinando quais combinações genéticas são mais ou menos aptas a um determinado meio ambiente. Ao longo de gerações, a população como um todo, passa por um *aprimoramento genético* que torna seus indivíduos cada vez mais aptos e especializados em viver em determinado meio ambiente.

Do ponto de vista matemático, a evolução pode ser entendida como otimização. AE's são procedimentos aproximativos que usam a evolução natural como argumento para combinar um conjunto de soluções, progressivamente e, assim, ao final, obter soluções satisfatórias em um tempo computacional razoável.

As soluções são atribuições de valores às variáveis da função objetivo do problema. As variáveis podem assumir valores discretos, contínuos, binários e, em conseqüência de sua natureza, determinam a forma como as soluções devem ser combinadas para gerar novas soluções.

Apesar do sucesso alcançado pela computação evolutiva, alguns autores ressaltam que muito do que a natureza tem apresentado, em termos de mecanismos evolutivos, foi

severamente simplificado ou truncado pelo processo evolutivo simulado (Falqueto et al., 2000). Entretanto, à medida em que surgem novos desafios, novos problemas envolvendo formulações mais complexas, espaços de busca não-restritos, com ruídos e múltiplos objetivos, outros processos naturais como dominância genética, competição e cooperação inter-espécies podem ser incorporados ao arcabouço evolutivo atualmente utilizado.

2.2 Codificação de soluções

O algoritmo genético (Holland, 1975; De Jong, 1975; Goldberg, 1989) se baseia em três pilares: codificação de soluções, formulação de uma função de aptidão (*fitness*), e implementação de operadores evolutivos que permitam simular o decorrer de gerações (épocas evolutivas), nas quais a população de indivíduos assume diferentes estados.

A codificação binária[1], na qual os genes são representados por bits 1's e 0's, dá à técnica uma generalidade e um certo formalismo matemático que de outra forma ela não teria. A Hipótese dos Blocos Construtivos (*Building Block Hypothesis*)(Holland, 1975), aliada a trabalhos que comprovam o Princípio dos Alfabetos Mínimos (Rees e Koehler, 1999) fornecem alguma explicação sobre o funcionamento dos AG's, apesar da controvérsia que algumas dessas hipóteses sempre têm causado (Altenberg, 1995).

2.2.1 Blocos construtivos

A Hipótese dos Blocos Construtivos (HBC) trata da formação e conservação dos chamados *blocos construtivos* e tem servido de base teórica para os AG's(Holland, 1975). *Blocos construtivos* são subconjuntos de bits que supostamente conferem altos valores de aptidão aos cromossomos em que aparecem. Diz-se que indivíduos bem avaliados são formados pela justaposição de blocos supostamente bem avaliados.

Pela HBC, os AG's são otimizadores eficientes pois realizam uma busca paralela em todo o espaço de busca, combinando pequenos e *qualificados* blocos de genes em blocos maiores e igualmente qualificados. Assim, evitar a quebra (*disruption*) desses blocos é a base para o bom desempenho dos AG's.

Esquemas são um tipo especial de indivíduo constituído de alguns genes rotulados como # (ou *). Genes assim rotulados podem assumir quaisquer outros valores permitidos pelo alfabeto em uso. Diz-se que um esquema não representa somente uma solução, mas pode ser *instanciado* em um conjunto delas pela substituição dos #'s por outros símbolos do alfabeto. Em outras palavras, um esquema significa um subconjunto do espaço de busca.

[1] Alfabeto de cardinalidade dois.

Seja $\mathcal{H}$ um esquema qualquer, o número de genes definidos (com valor diferente de #) de $\mathcal{H}$ é chamado de *ordem* $\mathcal{O}(\mathcal{H})$. A maior distância entre dois genes definidos em um esquema é chamada de *comprimento de definição* $\mathcal{D}(\mathcal{H})$ ou, simplesmente, comprimento de esquema. Por exemplo, seja $\mathcal{H} = 10\#0\#\#11$, tem-se $\mathcal{O}(\mathcal{H}) = 5$ e $\mathcal{D}(\mathcal{H}) = 8$. Neste caso, $\mathcal{D}(\mathcal{H})$ é o próprio comprimento total do esquema $|\mathcal{H}|$.

O *Teorema dos Esquemas*, também conhecido como *Teorema Fundamental dos Algoritmos Genéticos*, estabelece que um esquema $\mathcal{H}$ pequeno, de baixa ordem, e com avaliação acima da média da população terá o número de seus representantes, $m(\mathcal{H}, t)$, exponencialmente incrementado de uma geração para outra (Goldberg, 1989). Ou seja:

$$m(\mathcal{H}, t+1) \geq \frac{\hat{\mu}(\mathcal{H}, t)}{\overline{f}(t)} m(\mathcal{H}, t)(1 - p_c \frac{\mathcal{D}(\mathcal{H})}{|\mathcal{H}| - 1})(1 - p_m)^{\mathcal{O}(\mathcal{H})} \tag{2.1}$$

onde $\hat{\mu}(\mathcal{H}, t)$ é a média da avaliação de todas as instâncias de $\mathcal{H}$, $\overline{f}(t)$ é a média de aptidão de todos os indivíduos da população, $1 - p_c$ dá a probabilidade de não ocorrer cruzamento, equivalente à probabilidade de não haver *quebra* do esquema $\mathcal{H}$, e $1 - p_m$ é a probabilidade de não haver quebra por mutação.

A equação 2.1 se refere a probabilidades de ocorrência de três eventos independentes: seleção, cruzamento e mutação. A seleção é função de quão acima da média $\overline{f}(t)$ for $\hat{\mu}(\mathcal{H}, t)$. Aqui surge um primeiro questionamento relativo a como se avaliar um esquema. No caso do Teorema dos Esquemas, um $\mathcal{H}$ é avaliado pela média de seus representantes na população. A probabilidade de um esquema ser quebrado em função de cruzamentos e mutações e, portanto, desaparecer das próximas gerações, é tanto maior quanto maiores forem sua ordem $\mathcal{O}(\mathcal{H})$ ou seu comprimento $\mathcal{D}(\mathcal{H})$.

2.2.2 Epistasia

Não é sempre verdade que a justaposição dos blocos construtivos resultem em boas cadeias genéticas. Blocos bem avaliados podem gerar blocos mal-avaliados, especialmente quando avaliados por funções objetivo ditas *AG-enganosas* (*GA-deceptive functions*). Nessas funções ocorre o que é chamado de *epistasia*.

O termo *epistasia* é definido pelo geneticistas como a supressão do efeito de um gene por outro gene *não-alelo*. Por *não-alelo*, entende-se que são genes responsáveis por traços fenótipos diferentes, uma vez que *alelo* é o termo da biologia pare definir os valores que o gene pode ter. Dessa forma, a epistasia difere da dominância genética, na qual um alelo mascara ou suprime o efeito de outro do mesmo gene (Leblanc e Lutton, 1998)

Voltando a evolução simulada, epistasia é um efeito indesejado causado por uma forte interação entre variáveis (em termos de fenótipo) ou entre genes (em termos de genótipo) devido a forma como a função de aptidão é construída ou devido a outros fatores como o mecanismo de codificação. Vários trabalhos tem devotado especial atenção à questões relativas a epistasia em problemas de otimização (Rodriguez-Tello e Torres-Jimenez, 2003; Seo et al., 2003).

2.2.3 Codificação binária ou real

A codificação binária pode ser usada para representar tanto valores numéricos quanto grafos ou outras representações de soluções para problemas. Árvores semânticas podem ser utilizadas para codificar algoritmos em programação genética (Koza, 1992). A codificação em real tem sido uma alternativa atraente para problemas de otimização numérica especialmente por trabalhar com o espaço de genótipos igual ao espaço de fenótipos (Davis, 1991; Michalewicz, 1996).

Para os práticos, a codificação binária limita a eficiência de um AG em alguns aspectos. Quando se trabalha com variáveis contínuas, por exemplo, existe a necessidade de um conhecimento prévio dos limites que cada uma delas pode assumir para que se possa definir o número de bits de cada variável. Além disso, a precisão da solução obtida também é afetada diretamente por essa quantidade de bits usados na codificação. Por fim, mecanismos de auto-adaptação e de *intensificação de busca* são mais facilmente incorporados quando se trabalha com uma codificação mais próxima do domínio real do problema.

Intensificação de busca é o termo usado neste trabalho para designar mecanismos de melhorias específicas que podem ou não ser aplicados a soluções representadas por indivíduos. Em geral, tais mecanismos estão associados à heurísticas e outros conhecimentos prévios sobre o problema em questão e são detalhados posteriormente.

O operador de mutação foi proposto inicialmente para introduzir na população, constantemente, material genético inovador (Goldberg, 1989). Todavia, a mutação também pode ser empregada para introduzir um ruído aleatório, reduzido deterministicamente ao longo das gerações, se constituindo assim, como um operador que intensifica a busca em torno de regiões de busca (Michalewicz, 1996).

Estratégias Evolutivas (EE) (Rechenberg, 1965) e Programação Evolutiva (PE) (Fogel et al., 1965) são enfoques evolutivos que trabalham com codificação real para problemas com variáveis contínuas e possuem mecanismos de auto-adaptação embutidos no processo

de mutação *gaussiana*. Esse mecanismo especial permite ao algoritmo evoluir seus próprios parâmetros evolutivos estratégicos (desvio padrão e covariância). A PE difere dos demais enfoques evolutivos por não trabalhar com cruzamento de indivíduos. O argumento usado é que a PE se inspira na macro evolução das espécies, onde indivíduos geram descendentes similares a seus progenitores de acordo com uma distribuição normal.

Outras formas alternativas de codificação utilizando alfabetos de cardinalidade maiores que dois têm sido empregadas nas mais diferentes aplicações. Os AE's estão sendo construídos cada vez mais especializados nos problemas aos quais se destinam e cada vez menos atrelados a conceitos de classes, de tipos de procedimentos, etc. O rico arcabouço evolutivo atualmente existente e a falta de trabalhos conclusivos que indiquem que um ou outro enfoque é *absolutamente* superior aos outros irá tornar estéril, no futuro, a discussão sobre rótulos.

2.3 Heurísticas

Heurísticas ou algoritmos heurísticos são desenvolvidos objetivando a resolução de problemas de elevado nível de complexidade, onde métodos exatos de resolução tendem a executar em tempo computacional impraticável. Há, portanto, um compromisso entre o esforço computacional da heurística e a qualidade da solução que ela consegue obter.

Em geral, algoritmos heurísticos buscam por soluções de boa qualidade, mas com pouco ou nenhum suporte para relatar o quão próximas da solução ótima estão as soluções encontradas. Durante as últimas décadas, muito se tem estudado heurísticas simples direcionadas à solução de problemas combinatórios *NP-árduos*. Basicamente, existem três grandes grupos delas: as construtivas, as de melhoria ou de busca local e as metaheurísticas.

2.3.1 Tipos de heurísticas

Heurísticas construtivas são aquelas que, a partir de uma solução vazia, adicionam-se novos elementos (variáveis, vértices, arestas) iterativamente até que seja obtida uma solução viável. As heurísticas gulosas se enquadram nesse comportamento, tentando a cada iteração, maximizar a melhoria local.

Uma heurística de melhoria (ou de busca local) é aquela em que, de acordo com regras preestabelecidas, modifica uma dada solução inicial, gerando novas soluções iterativamente até que um determinado critério de parada seja atingido. Tipicamente, o critério de parada é atingido quando a solução corrente não puder ser melhorada. Neste caso, diz-se que a solução corrente é um *ótimo local*, segundo a heurística. Um ótimo local

é o melhor ponto dentro de uma certa *localidade* definida pela relação de vizinhança que é induzida por uma dada heurística de busca (Blum e Roli, 2003).

Tanto as heurísticas construtivas, quanto as de melhoria, em geral, são procedimentos específicos para determinada classe de problemas e determinísticos. Para uma mesma solução inicial e os mesmos parâmetros de ajuste do método, sempre se chega a uma mesma solução final.

As chamadas metaheurísticas se constituem em um terceiro grupo de heurísticas. Em níveis de hierarquia, uma metaheurística é uma heurística que emprega outras heurísticas, com certo grau de flexibilidade para possibilitar sua fácil adaptação a uma grande variedade de problemas. Diferentemente dos outros dois tipos, estes métodos são providos de uma ou mais estratégias para torná-los mais robustos e assim terem maiores chances de *escapar* de ótimos locais.

Os AE's são baseados em população, o que permite explorar mais de uma região do espaço de busca ao mesmo tempo. O *recozimento simulado* opera com somente uma solução, mas aceita soluções piores do que a corrente, dependendo de certa probabilidade (Kirkpatrick et al., 1983). A busca tabu mantém mecanismos de memória que servem para evitar movimentos repetidos (Glover, 1996).

Mecanismos de *reinicialização*, quando soluções melhores não podem mais ser encontradas, também são comuns em algumas metaheurísticas, como o *GRASP* (Feo e Resende, 1989). Quando mais efetivos forem tais mecanismos, mais a heurística passa a ter uma característica de otimizador global, ou seja, algoritmo com capacidade de encontrar o ótimo global, independente da solução inicial ou conjunto delas.

Pode-se dizer que um algoritmo heurístico qualquer, em uma dada execução, *está preso* a um ótimo local, quando ele assumir um conjunto de pontos do espaço de busca a partir do qual ele não consegue mais alcançar nenhum outro ponto melhor, usando os valores correntes de seus parâmetros de desempenho.

Metaheurísticas são perfeitamente compatíveis com heurísticas construtivas e de melhoria no sentido de agregar em um único método robustez e precisão. Um exemplo de tal *simbiose* reside no mundo dos AE's: heurísticas de melhoria embutidas nos chamados *algoritmos evolutivos híbridos*. Em geral, heurísticas são embutidas nos operadores de cruzamento e mutação, respeitando as características conceituais de cada um.

2.3.2 Paisagem de aptidão

Um problema de busca é caracterizado por um *espaço de busca codificado*, que pode ser entendido como um conjunto de pontos S sobre o qual a busca é conduzida, e uma função de avaliação f responsável pela atribuição de um valor numérico para cada elemento $s \in \mathsf{S}$. Cada ponto significa uma solução, i.e., uma combinação diferente de valores que cada variável do problema pode assumir.

O espaço de busca codificado[2] não necessariamente é o mesmo espaço do domínio real do problema. Durante o processo de modelagem de um AE, o conjunto de soluções que os indivíduos podem representar passa por um processo de codificação que o torna computacionalmente implementável. A busca se dá no espaço de genótipos (espaço de busca codificado) e não no de fenótipos (espaço do domínio real), a menos que sejam idênticos. Para a avaliação de aptidão é necessário antes a conversão do genótipo do indivíduo para seu fenótipo.

Uma relatada vantagem de se trabalhar diretamente no espaço de fenótipos reside na possibilidade de se empregar mais facilmente heurísticas específicas do problema, colaborando com a busca evolutiva. Em geral, heurísticas fazem várias avaliações de soluções o que acarreta na necessidade de decodificação de genótipo para fenótipo sempre que for aplicada uma melhoria através de heurísticas (Michalewicz, 1996).

O desempenho dos AE's é sensível a forma como é feita codificação. A população de indivíduos, em uma dada geração, é uma amostra relativamente pequena dos pontos de S e são necessárias sucessivas *amostragens* para que soluções melhores sejam obtidas. Uma *amostragem* significa geração de indivíduos e a avaliação do ponto que ele assumiu. Mesmo o espaço de busca contínuo é discretizado através de amostras.

A função de avaliação f mede a qualidade da solução e está associada à função objetivo do problema. No caso dos AE's, f é também conhecida como aptidão (*fitness*) do indivíduo e serve para quantificar o genótipo do indivíduo em termos da função objetivo:

$$f : \mathsf{S} \rightarrow \Re^{\tau} \tag{2.2}$$

onde $\tau > 1$ significa f relacionando múltiplos objetivos (Coello Coello, 2000).

A função de avaliação pode ser a própria função objetivo do problema ou pode ser

[2]O termo *espaço de busca* é usado, neste trabalho, como uma simplificação para *espaço de busca codificado.*, mas têm o mesmo significado.

utilizado algum tipo de normalização para reduzir a diferença de avaliação entre indivíduos e assim evitar que um indivíduo super-avaliado domine excessivamente o processo de seleção (Goldberg, 1989).

A métrica de distância $\wp$ é um terceiro elemento de interesse para este trabalho. Segundo (Merz e Freisleben, 1999):

$$\wp : \mathsf{S} \times \mathsf{S} \rightarrow \Re^{+} \tag{2.3}$$

onde valem as seguintes propriedades:

a) $\wp(s_1, s_2) = 0 \Leftrightarrow s_1 = s_2$ e

b) $\wp(s_1, s_3) \leq \wp(s_1, s_2) + \wp(s_2, s_3)$.

Uma métrica de distância está relacionada ao esforço necessário para transformar uma solução em outra, ou seja, transitar de um ponto para outro no espaço de busca. Existem várias formas de defini-la. Por exemplo, seja uma heurística H que forneça regras de transformação de uma solução em outra. Pode-se definir $\wp(s_1, s_2)$ como sendo o número de vezes que é necessário aplicar H a s_1 para transformá-la em s_2. A distância de *hamming* e a distância *euclidiana* são típicas métricas usadas em espaços de busca discretos e contínuos, respectivamente.

A tripla $\pounds(S, f, \wp)$ é, portanto, um conjunto de pontos interligados, com diferentes valores de avaliação, formando o que é conhecido como paisagem de aptidão (*fitness landscape*), ou superfície de custo. Em analogia a geografia, em duas dimensões, pode-se observar *picos* (máximos locais ou globais), vales (mínimos locais ou globais), planícies e platôs (regiões onde o *fitness* se mantém constante).

Paisagem de aptidão é uma metáfora utilizada para descrever uma estrutura de vizinhança, ou seja, a forma como os pontos se conectam e os custos associados a cada conexão. A tripla $\pounds(S, f, \wp)$ pode ser entendida como um grafo ponderado (eventualmente dirigido) $\mathcal{G}(V, A)$ cujo o conjunto de vértices $V = S$ e o conjunto de arestas A é dado por:

$$A =< s, s_v > \in \{S \times S\}, \wp(s, s_v) = \wp_{min}(s, s_v) \tag{2.4}$$

onde $\wp_{min}(s, s_v) = \min\{\wp(s, s_x) + \wp(s_x, s_v)\}$, para qualquer $s_x \in S$.

No processo de geração de novas soluções a partir de uma solução inicial s e usando-se

uma dada heurística H, define-se vizinhança $\varphi^H(s)$ como uma relação de *proximidade* entre soluções no grafo $\mathcal{G}(V, A)$. A vizinhança de uma solução é um conjunto de soluções que podem ser geradas (ou alcançadas), a partir dela, executando-se movimentos permitidos pela heurística em uso.

Em um espaço de busca contínuo, por exemplo, a vizinhança pode ser definida como um conjunto $\varphi^r(s)$ de pontos dentro da hiperesfera de centro s e um raio r qualquer que pode estar relacionado à métrica euclidiana de distância.

Existem algumas características fundamentais das paisagens de aptidão que influenciam sobremaneira o desempenho de heurísticas. Vários métodos têm sido propostos para qualificar paisagens de aptidão (Reeves, C.R., 1999; Merz e Freisleben, 1999). As principais características são:

a) a diferença de aptidão entre soluções vizinhas (aspereza);

b) quantidade de ótimos locais;

c) distribuição de ótimos locais;

d) topologia da base de atração do ótimo global;

e) presença de epistasia.

No caso dos algoritmos genéticos, devido a sua característica estocástica, as regras de transição de pontos estão associadas a probabilidades de ocorrência. A cada geração, um AG faz movimentos em diferentes paisagens de aptidão. Um trabalho interessante sobre esse tema é encontrado em (Jones, 1995). Nele, é apresentado um modelo composto por uma paisagem de aptidão para cada operador evolutivo (mutação, cruzamento e seleção). Os exemplos que se seguem usam uma população de indivíduos codificados sobre um espaço de busca $S = \{0, 1\}^3$ (cadeias binárias de comprimento três).

Para uma mutação com probabilidade p de inverter um gene (conseqüentemente, a probabilidade de não inverter é $q = 1 - p$), a paisagem de mutação pode ser representada por um grafo onde cada vértice é um indivíduo e as arestas possuem probabilidades que variam entre p^3 (inversão de 3 genes) e q^3 (indivíduo não ser modificado). Em outras palavras, somente alguns dos indivíduos são movidos desde que um dos seus genes seja alterado.

Para um cruzamento $1-$ponto (Holland, 1975), pode-se descrever a paisagem de cruzamento como um grafo cujos vértices são pares de indivíduos e as arestas também estão associadas a probabilidades de ocorrência. Considerando 3 genes, há a possibilidade

de serem sorteados dois pontos diferentes de cruzamento a partir dos quais são feitas as trocas de genes entre pais, gerando dois filhos. Por isso, a probabilidade de transição de pontos é de $\frac{1}{2}$. A Figura 2.1 mostra os grafos que representam a paisagem de aptidão de cruzamento e mutação (Jones, 1995). A paisagem de cruzamento mostra apenas os pares cuja distância de *hamming* $\wp_{ham} = 3$ (paisagem parcial).

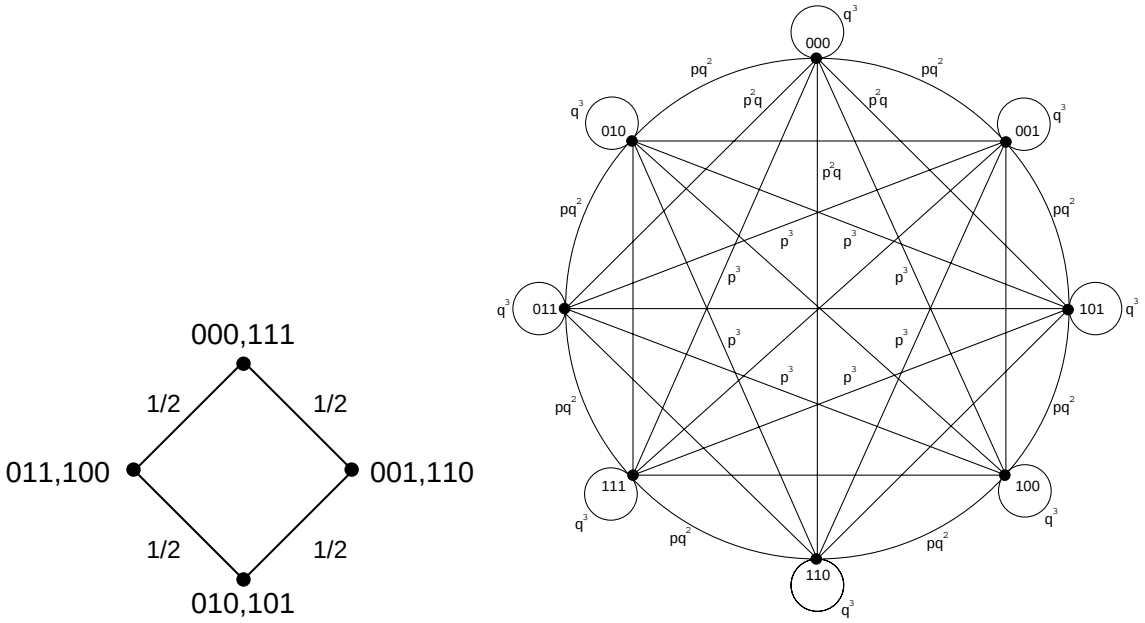


FIGURA 2.1 – Paisagem de cruzamento (parcial) e mutação. *Adaptado de* (Jones, 1995).

O cruzamento tem a característica de imprimir *saltos* maiores na paisagem de aptidão, dependendo de quão distantes estejam os pais. No exemplo apresentado, pais com $\wp_{ham} \leq 1$ gerariam pares idênticos de indivíduos. A mutação, por outro lado, tem desempenho mais discreto, especialmente quando adequadamente configurada, i.e., com baixas probabilidades de ocorrência. Algumas implementações da mutação de $1-$bit permitem que no máximo um gene (bit) possa ser alterado a cada ocorrência, o que alteraria um pouco o grafo de mutação da Figura 2.1.

O operador de seleção atua na população de indivíduos, produzindo uma nova população. Dessa forma, a população inteira, a cada geração, corresponde a um vértice no grafo representativo da paisagem de seleção. A transição se dá dependendo dos indivíduos selecionados para a próxima geração. Por esse enfoque, o grafo é dirigido e ponderado, o que ressalta uma certa assimetria até então não identificada nas outras paisagens. A Figura 2.2 mostra as três paisagens de aptidão do modelo proposto por (Jones, 1995).

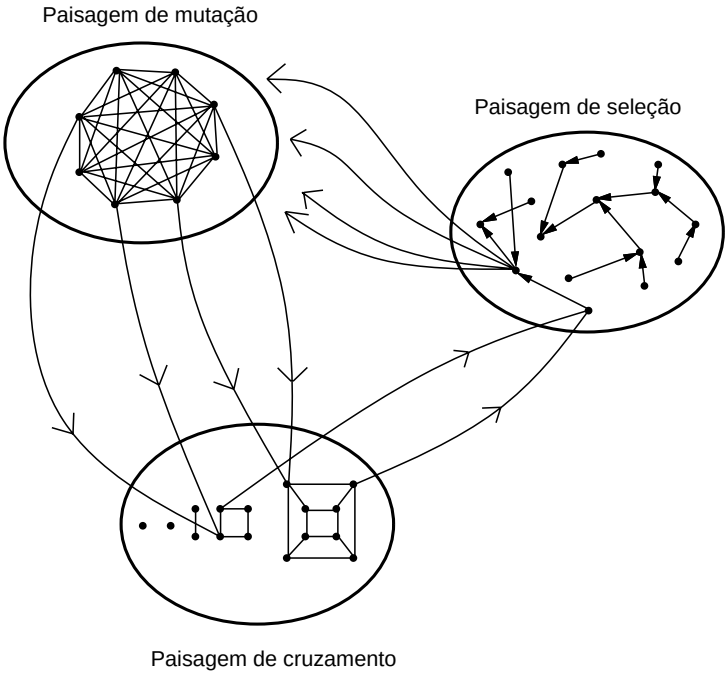


FIGURA 2.2 – Paisagem de aptidão de um AG. *Adaptado de* (Jones, 1995).

Considerando essas três paisagens de aptidão, pode-se conjecturar que um AG, em uma dada geração, não consegue convergir para o ótimo global, quando não for possível atingir um dos *vértices ótimos* (vértices contendo soluções ótimas) nas paisagens de mutação e cruzamento aplicando-se tais operadores nos indivíduos presentes em algum dos vértices da paisagem de seleção alcançáveis a partir do vértice atual. Assim, a eficácia de um AG depende de estratégias que possam aumentar a probabilidade de que sejam alcançados *vértices ótimos* ao longo das gerações. Em geral, tais estratégias devem combinar mecanismos de convergência (busca pelas melhores soluções) e diversificação (manter material genético informativo de todo o espaço de busca).

2.4 Algoritmos evolutivos híbridos

O principal argumento dos favoráveis ao hibridismo reside na suposta deficiência dos AG's em termos de dispor mecanismos efetivos de intensificação de busca (Areibi, 2001; Birru et al., 1999; Yen e Lee, 1997). Sem encontrar novas regiões de busca, nem fazer o refinamento apropriado das regiões encontradas, um AG tende a se estabilizar em um ponto, onde indivíduos melhores não são mais encontrados, e há baixa probabilidade de se escapar dele.

Considerando que, a cada geração a população tende a convergir progressivamente para uma crescente uniformidade, com perda significativa de informação sobre o espaço de

busca completo. Considerando ainda que a população, em uma dada geração, pode possuir uma solução suficientemente próxima (ou vizinha) à solução ótima, entende-se que a uma busca dirigida por alguma heurística específica poderia encontrar a solução ótima, sem depender tanto das reduzidíssimas probabilidades de transição de pontos das paisagens de aptidão dos AG's. Existe um consenso que a combinação de forma adequada de algoritmos evolutivos e heurísticas específicas tende a ser significativamente superior a versões canônicas dos AE's (Moscato, 1999).

Heurísticas específicas têm sido comumente associadas a diferentes etapas do processo evolutivo, desde a geração inicial de indivíduos melhorados (Glover, 1998), passando por formas mais *inteligentes* de cruzamentos, que mesclam nos descendentes os melhores blocos dos pais (Freisleben e Merz, 1996), até chegar em mutações que nada mais são que algoritmos de busca local (Merz, 2003).

Apesar de sua natureza populacional, onde uma grande parte do espaço de busca é avaliado, os AE's podem ter desempenho melhorado pela geração de uma população inicial menos aleatória. Essa prática surge da necessidade de correção de indivíduos inviáveis, especificamente nos casos em que codificação permite genótipos que não correspondem a fenótipos válidos. Por esse motivo, mecanismos de correção e de melhoramento podem ser introduzidos desde a geração inicial da população. Todavia, cuidados devem ser tomados com relação à diversidade populacional, evitando-se que a população inicial seja mal distribuída sobre o espaço de busca (Feltl e Raidl, 2004).

Outro exemplo de heurísticas específicas conduzindo o processo evolutivo pode ser encontrado em (Navarro et al., 1999). Nesse trabalho, é proposto um enfoque alternativo para solução do problema de inversão magneto-telúrico, que consiste em se obter a condutividade do subsolo a partir da condutividade da superfície. São propostos três novos tipos de operadores evolutivos específicos: um cruzamento uniforme espacial, um operador unário de busca local do tipo descida da encosta (*hill-climbing*), e um operador unário de homogeneização. Esse último explora uma peculiaridade do problema que é o fato de regiões vizinhas provavelmente possuírem condutividades semelhantes.

É comum se definir busca local como um processo de aprendizado em indivíduos. No aprendizado *Lamarquiano*, em alusão ao autor da lei do uso e desuso (Lamarck, 1914), se admite a possibilidade de serem incorporadas aos genótipos as melhorias adquiridas pelos indivíduos por *aprendizado* (Moscato, 1999). Neste modelo de aprendizado, há uma perda da memória do indivíduo, através de sua descaracterização genotípica.

Uma alternativa ao aprendizado *Lamarquiano*, baseada na proposta de J. M. Baldwin

(Baldwin, 1896) (aprendizado *Baldwiniano*), constitui-se na suposição de que o aprendizado adquirido através da interação de indivíduos com o meio ambiente poderia ser herdado indiretamente através da maior probabilidade de procriarem que indivíduos com boa capacidade de aprendizado têm. Em outras palavras, indivíduos que têm tendência nata para se adaptarem ao meio (ou aprender com ele) têm maiores chances de legarem como herança o seu genótipo, incluindo sua capacidade de adquirir novas habilidades.

Algoritmos meméticos (AM) são enfoques populacionais que apresentam, segundo seus entusiastas, desempenho significativamente superior aos algoritmos genéticos tradicionais (Moscato, 1999). Os AM's são caracterizados como uma coleção de agentes que realizam explorações autônomas (aprendizado por busca local) do espaço de busca, cooperando entre si através da recombinação e competindo por recursos durante o processo de seleção e atualização. Existem vários trabalhos que reivindicam a designação de algoritmos meméticos basicamente por acrescentar otimizadores locais na melhoria de indivíduos (Areibi, 2001; Mendes e Linhares, 2004).

Os *memes*, presentes nos AM's, se referem a elementos culturais, religiosos, políticos e sociais. Por exemplo, algumas religiões contêm memes associados a sexualidade, autoridade, humildade e outros aspectos da condição humana. Na natureza, comportamentos sociais ou instintivos, como a capacidade de reação à aproximação de animais peçonhentos, também estão associados à informação memética.

Os memes influenciam a sobrevivência de indivíduos, assim como os genes. Quando comparada com a evolução memética, a evolução biológica é extremamente lenta. Memes podem ser transferidos horizontalmente (entre indivíduos na mesma geração) ou verticalmente (de pai para filho, em gerações diferentes), diferentemente dos genes que apenas podem ser herdados.

A transgenética computacional, por outro lado, trabalha com memes de forma mais ampla, empregando operadores epigenéticos, onde há troca de material memético horizontalmente, além da reprodução sexual. Os memes passam a ter o propósito de organizar ou reestruturar os genes (Goldbarg e Gouvea, 2001).

2.5 Detecção de regiões promissoras

Como já foi mencionado, o conceito de *regiões promissoras* vem sendo associado ao emprego de estratégias capazes de identificar o potencial positivo de subespaços de busca. Esses subespaços podem ser melhor explorados por heurísticas específicas para o problema em questão.

Heurísticas de melhoria são procedimentos razoavelmente eficazes quando se pretende acrescentar maior precisão aos AE's, entretanto apresentam um significativo custo computacional. O espaço de busca deve ser uniformemente coberto, dando maior ênfase a regiões com maior possibilidade de sucesso na busca pelo ótimo global. São, portanto, necessárias estratégias coerentes para que essas heurísticas sejam aplicadas de forma racional, em regiões realmente promissoras do espaço de busca. Em outras palavras, o processo de intensificação de busca deve ser discriminatório, em uma parcela da população definida através de algum critério.

Uma forma de discriminar é aplicar melhoramentos apenas em indivíduos com boa qualidade, admitindo-se uma espécie de limiar de qualidade a partir do qual o indivíduo é factível de ser melhorado (Yen e Lee, 1997). Esse critério é estritamente elitista.

Um outro critério igualmente popular é o probabilístico. Uma parte aleatória da população é melhorada através de busca local. Uma mistura desses dois critério também pode ser utilizada. Por exemplo, dentro do grupo de indivíduos considerados elite, pode-se escolher ao acaso aqueles que serão melhorados.

Regiões promissoras podem ser representadas por *esquemas* com avaliação acima da média (Goldberg, 1989). Sabe-se que descendentes produzidos por recombinação tendem a *habitar* as mesmas regiões de seus pais ou, posto de outra forma, manter algumas características genéticas dos pais. A seleção contínua de indivíduos bem avaliados causa um aumento no número de *bons esquemas* nas próxima gerações (Holland, 1975). É dessa forma que os AG's intensificam a busca em torno das regiões promissoras, através de um esperado aumento da amostragem nessas regiões.

Hipoteticamente, blocos genéticos pequenos, de boa qualidade tendem a se disseminar na população, através dos indivíduos que os contém. A população, portanto, tende a convergir para um perfil de indivíduos no qual predominam os blocos construtivos. Diz-se que uma população convergiu para um determinado esquema quando grande parte dela são *instâncias* desse esquema.

Regiões promissoras podem ser detectadas por mérito de avaliação ou de freqüência (Rosca e Ballard, 1994). Por mérito de avaliação, algumas soluções pertencentes a uma certa região podem ser consideradas para dizer o quão promissora é aquela região (Haynes, 1997). Estratégias híbridas baseadas em elitismo empregam exatamente essa filosofia: os melhores indivíduos são considerados como representantes de regiões promissoras e é somente sobre esse grupo elite que é aplicada a intensificação de busca.

Por mérito de freqüência, regiões mais amostradas podem ser consideradas promissoras. Essa idéia nasce de uma interpretação inversa da Teoria dos Esquemas, ou seja, o aumento no número de blocos construtivos se dá através de uma maior freqüência de participação no processo evolutivo dos indivíduos que os contenham em seu genótipo.

2.6 Problemas de otimização em foco

Nesta seção, são apresentados três problemas para os quais são construídas aplicações baseadas nas abordagens propostas neste trabalho. São eles:

a) problemas de seqüenciamento de padrões, onde se inserem os problemas de minimização de pilhas abertas de leiaute de matriz-porta;

b) problemas de *satisfabilidade* (problemas SAT);

c) minimização numérica sem restrição.

2.6.1 Problema de seqüenciamento de padrões

Problemas de seqüenciamento de padrões podem ser expressos por uma matriz de números inteiros $\mathcal{P}_{i,j}$ cujo o objetivo é encontrar uma permutação de linhas ou padrões (pedidos de clientes, portas em circuitos *VLSI*, padrões de corte, etc) que minimize alguma função objetivo (Fink e Voss, 1999). As funções objetivo consideradas aqui diferem de outros problemas combinatoriais, tais como o *problema do caixeiro viajante* devido a forma como é avaliada uma dada permutação. Em problemas de seqüenciamento de padrões, a avaliação não pode ser computada somando-se valores que dependem apenas dos padrões adjacentes.

Um Problema de Minimização de Pilhas Abertas (*Minimization of Open Stacks Problem - MOSP*) ocorre em certas situações da produção industrial, onde padrões de ítens com diferentes especificações (formatos, tamanhos, etc) precisam ser produzidos e temporariamente mantidos em pátios durante a produção. Por exemplo, considere uma indústria de componentes de madeira prensada (padrões) que são formados por pedaços (ítens) de madeira de diferentes tamanhos. À medida em que os padrões são processados, ítens são cortados e colocados em pilhas de acordo com seus tamanhos: uma pilha para cada tipo de ítem. Cada pilha fica aberta enquanto houver elementos a serem cortados de um mesmo ítem.

Deve-se determinar a seqüência de padrões de corte que minimize o máximo de pilhas abertas durante o processo de corte. Tipicamente, esse é um problema que ocorre devido à limitações de espaço físico, uma vez que o acúmulo de pilhas pode causar a necessidade de remoção temporária de uma ou outra pilha, retardando o processo de produção.

Um *MOSP* é dado por uma matriz binária de $I \times J$, representando padrões (linhas) e ítens (colunas), onde $\mathcal{P}_{i,j} = 1$ se o padrão i contiver o ítem j, e $\mathcal{P}_{i,j} = 0$ caso contrário. Os padrões são seqüencialmente processados, i.e., todos os ítens de cada padrão são cortados por vez. A seqüência de padrões determina o número máximo de pilhas abertas ao mesmo tempo, aqui chamado de *mpa*.

Outra matriz binária, aqui chamada de matriz de pilhas abertas $\mathcal{Q}^\pi$, é usada para calcular o *mpa* de uma dada permutação π. Ela é obtida a partir de $\mathcal{P}$ seguindo-se as seguintes regras:

- $Q^\pi_{ij} = 1$ se $\exists x$ e $y | \pi(x) \leq i \leq \pi(y)$ e $P_{x,j} = P_{y,j} = 1$;
- $Q^\pi_{ij} = 0$, caso contrário;

onde $\pi(a)$ é a posição do padrão a na permutação π. A matriz $\mathcal{Q}^\pi$ mostra a propriedade dos *1's-consecutivos* (Golumbic, 1980) aplicada a $\mathcal{P}$: em cada coluna, 0's entre 1's são substituídos por 1's. A soma de 1's, por linha, resulta no número máximo de pilhas abertas a cada padrão processado.

A Figura 2.3 mostra um exemplo de $\mathcal{P}$, sua correspondente matriz $\mathcal{Q}^\pi$, e o número de pilhas abertas a cada padrão processado. No exemplo, quando o padrão 1 é cortado, 3 pilhas são abertas. Nenhuma pilha é aberta para os padrões 2 e 3, mas o padrão 4 requer 5 pilhas abertas. No máximo, 7 pilhas ($mpa = \max\{3, 3, 3, 5, 6, 7, 7, 5, 3\} = 7$) são necessárias para processar a permutação $\pi_0 = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$.

Recentemente, vários aspectos do *MOSP* e de outros problemas relacionados, tais como o Problema de Leiaute de Matriz-Porta (*Gate Matrix Layout Problem - GMLP*), têm sido apresentados, incluindo provas de que ambos são problemas *NP-árduos* (Möhring, 1990; Kashiwabara e Fujisawa, 1979; Linhares, 2002; Becceneri, 1999; Yanasse, 1996).

O *GMLP* está relacionado com arranjos lógicos unidimensionais e programáveis (Möhring, 1990; Kashiwabara e Fujisawa, 1979). Um problema que surge em projetos de circuitos VLSI (*Very Large Scale Integration*) é o de arranjar um conjunto de portas (fios horizontais de um circuito *VLSI*), os quais são interconectadas através de redes (fios verticais), de modo que o número de *trilhas* é minimizado. O conceito de trilhas se refere ao espaço físico do circuito onde as redes são acomodadas. A minimização de trilhas é possível acomodando redes que não se sobreponham em uma mesma trilha. Se nenhuma rede se sobrepor é necessária uma única trilha para acomodá-las.

Na Figura 2.3 pode ser vista uma instância *GMLP*. O número de pilhas abertas é equivalente ao número de redes sobrepostas (ou trilhas). A Figura 2.4 mostra a melhor

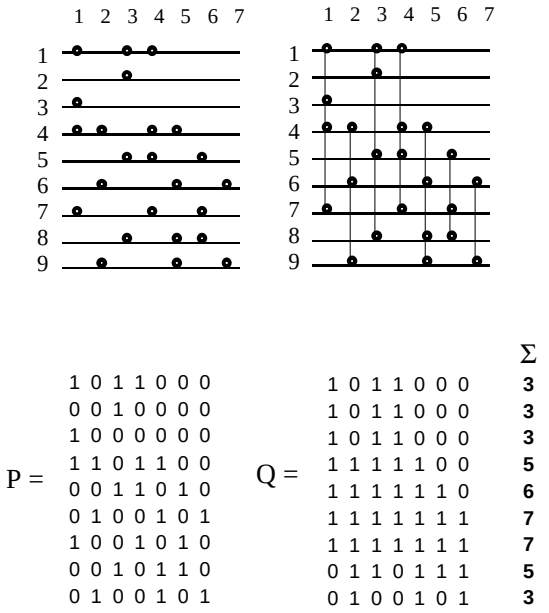


FIGURA 2.3 – Instâncias *MOSP* e *GLMP*: matrizes original e correspondente.

solução para o exemplo da Figura 2.3. No máximo, 5 pilhas são necessárias para processar a permutação $\pi_* = \{6, 9, 5, 8, 7, 5, 1, 2, 3\}$.

2.6.2 Problema de *satisfabilidade* (SAT)

O SAT é um problema combinatório clássico, central em um grande número de áreas na computação e engenharia: planejamento em inteligência artificial (Kautz e Selman, 1992), visão computacional, projeto de circuitos. Outros problemas de combinatória como coloração de grafos, escalonamento e satisfação de restrição são alguns exemplos de afinidades com o SAT. Uma boa leitura sobre teoria e aplicações do SAT pode ser encontrada em (Gu et al., 1997).

Uma instância SAT é uma expressão lógica $\mathcal{E}$ composta de $|\mathcal{E}|$ cláusulas distintas conectadas por operadores lógicos $\wedge's$ ($E-$lógico). Cada cláusula $\mathcal{E}_i$ pode conter um número qualquer de literais conectados por operadores lógicos $\vee's$ ($OU-$lógico). Um literal é o valor de qualquer uma das n variáveis lógicas, eventualmente negado. Essa formulação, chamada de Forma Normal Conjuntiva (*Conjuntive Normal Form - CNF*) é exemplificada a seguir:

$$\mathcal{E} = (x_1 \vee x_5 \vee \overline{x_6}) \wedge (\overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_5}) \wedge (x_4 \vee x_3 \vee \overline{x_2} \vee \overline{x_1})$$

Existem outras formulações para SAT, como por exemplo a Forma Normal Disjuntiva

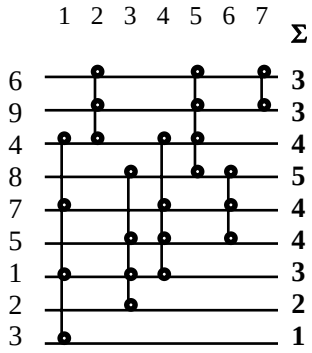


FIGURA 2.4 – Melhor solução para o exemplo da Figura 2.3.

(*Disjunctive Normal Form - DNF* e a formulação contínua (restrita e irrestrita)(Gu et al., 1997).

Uma atribuição é o conjunto de n valores lógicos que permitem que uma expressão seja avaliada. No exemplo anterior, atribuindo-se $x_i = \{V, F, F, F, F, V\}$, torna-se a expressão avaliada como verdadeira (*.V.*).

O objetivo de um problema SAT é determinar se existe uma atribuição de valores para as variáveis que satisfaça a expressão, i.e., torne a expressão verdadeira. Se existir pelo menos uma solução, diz-se que o problema SAT é *satisfazível*, caso contrário, *não-satisfazível*. Em uma formulação *CNF*, uma expressão satisfazível significa que todas as suas cláusulas devem ser satisfeitas. Quando o problema é não-satisfazível, é considerada como melhor solução, a atribuição que satisfaça um maior número de cláusulas.

Instâncias SAT podem apresentar um número k fixo de literais por cláusula. Tais problemas são chamados de k-SAT. Enquanto o 2-SAT pode ser resolvido polinomialmente, a família de problemas k-SAT, $k \geq 3$, pertence a classe de problemas *NP-Completos* (Garey e Johnson, 1979; Cook, 1971).

2.6.3 Minimização de funções numéricas sem restrição

Otimização numérica sem restrição é o caso mais genérico de otimização não-linear na qual não existem restrições na forma de equações $A(x) = b$ ou inequações $A(x) \leq b$ (Bazaraa et al., 1990). Podem existir, todavia, restrições de domínio de variável.

Todo problema para o qual se consiga modelar uma função objetivo não-linear, de variáveis definidas em $\mathbb{R}$, sujeita ou não a restrições de domínio, pode ser considerado otimização numérica. Em problemas práticos, questões de restrição de domínio surgem em função do conhecimento prévio dos valores que as variáveis podem assumir. De certa

forma, esse conhecimento viabiliza a busca por uma solução ótima em um espaço de busca que, apesar de infinito, é limitado.

Várias funções-teste são encontradas na literatura para esse tipo de problema também conhecido como otimização de parâmetros de função contínua. A apresentação geral do problema é:

$$\begin{aligned} \min \ / \ \max \quad & f(x), \quad x = (x_1, x_2, x_3, \ldots, x_n)^T \in \mathbb{R}^n \\ & \text{sujeito a } x_i^{inf} < x_i < x_i^{sup} \end{aligned} \tag{2.6}$$

Na maioria das funções-teste encontradas na literatura, os limites superior x_i^{sup} e inferior x_i^{inf}, que cada variável pode assumir, são definidos a *priori* e são parte do problema, uma vez que limitam o espaço de busca sobre as regiões mais desafiadoras.

Na Tabela 2.1 são apresentadas as funções-teste usadas neste trabalho, juntamente com os limites de domínio, comum a todas as variáveis, e a solução ótima conhecida. Observa-se que algumas funções se apresentam com um número fixo de variáveis n, como é o caso de *Michalewicz* com $n = 5$ e $n = 10$.

TABELA 2.1 – Funções-teste.

Função	var	ótimo	$x_i^{inf}; x_i^{sup}$	Função	var	ótimo	$x_i^{inf}; x_i^{sup}$
Ackley	n	0	-15 ; 30	Goldstein	2	3	-2 ; 2
Griewank	n	0	-600; 600	Easom	2	-1	-100;100
Schwefel	n	0	-500;500	Shekel 10	4	-10,536	-10 ; 10
Rosenbrock	n	0	-5,12;5,12	Langerman	5	-1,4	0;10
Rastrigin	n	0	-5,12;5,12	Langerman	10	-1,4	0;10
Sphere	n	0	-5,12;5,12	Michalewicz	5	-4,687	0; π
Zakharov	n	0	-5 ; 10	Michalewicz	10	-9,66	0; π
				Hartman	6	-3,322	0 ; 1

As funções-teste que podem apresentar um número n qualquer de variáveis aparecem com essa informação na coluna *var*. Essas funções-teste são particularmente interessantes pois, mesmo variando-se n, são mantidas as mesmas soluções ótimas conhecidas.

De Jong construiu um ambiente de teste com cinco funções para minimização e duas medidas para julgar o desempenho de algoritmos genéticos (De Jong, 1975). A primeira

medida, chamada de desempenho *on-line*, é uma média de todas as avaliações de função objetivo até uma dada geração. De certa forma, essa medida penaliza o algoritmo que avalie muitos indivíduos de baixa qualidade e premia aquele que rapidamente encontre valores razoáveis de aptidão. A segunda, definida como *off-line*, é a média de aptidão de todos os melhores indivíduos obtidos até a presente geração.

Sphere e *Rosenbrock* são as duas mais famosas funções construídas por (De Jong, 1975). A primeira (veja equação 2.7) tem topologia suave, unimodal, convexa, simétrica e é relativamente fácil quando usada com poucas variáveis (abaixo de 50). *Rosenbrock* é considerada difícil devido a sua topologia se assemelhar com um vale estreito de pouca inclinação e que se prolonga em forma de parábola. A Figura 2.5h mostra a topologia da função de *Rosenbrock*. Algoritmos que não são capazes de descobrir boas direções de busca se perdem nesse famoso vale. Além disso, *Rosenbrock* é conhecida por ter um alto grau de *epistasia*, ou seja, forte interação entre suas variáveis. Isso é visível na equação 2.8.

O mais extenso conjunto de funções-teste foi construído por Schwefel e consiste de 62 funções que cobrem uma enorme variedade de diferentes topologias (Schwefel, 1981). A mais famosa delas é conhecida por *Schwefel* e faz parte do conjunto de funções usadas neste trabalho.

As funções de *Schwefel*, *Rastrigin*, e *Griewank* (equações 2.9, 2.10 e 2.11) são típicos exemplos de funções não-lineares e multimodais. *Rastrigin* é a mais difícil das três devido ao número de mínimos locais: algo em torno de milhões (Digalakis e Margaritis, 2002). A função de *Schwefel* é um pouco mais fácil que *Rastrigin*, mas tem um segundo melhor valor de função objetivo bem distante do valor ótimo.

As funções *Shekel*, *Michalewicz* (Figura 2.5i) e *Langerman* (Figura 2.5j), fizeram parte do primeiro torneio internacional em otimização evolutiva (*First International Contest on Evolutionary Optimisation - 1ICEO*), corrido na Universidade de Nagoya, Japão (Bersini et al., 1996). A seguir, são mostradas as Equações de algumas funções-teste.

$$
\begin{aligned}
f_{Sph}(x_i) &= \sum_{i-1}^{n} x_i^2 \qquad (2.7)\\
&\quad -5,12 < x_i < 5,12\\
&\quad \min(f_{Sph}) = f(0,\ldots,0) = 0
\end{aligned}
$$

$$
\begin{aligned}
f_{Ros}(x) &= \sum_{i=1}^{n-1}\big(100(x_{i+1}-x_i^2)^2+(1-x_i)^2\big) \\
&\quad (-5,12 \leq x_i < 5,12) \\
&\quad min(f_{Ros}(x)) = f(1,1,\ldots,1) = 0
\end{aligned} \tag{2.8}
$$

$$
\begin{aligned}
f_{Sch}(x) &= 418,982887n + \sum_{i=1}^{n} -x_i \sin\Big(\sqrt{|x_i|}\Big) \\
&\quad (-500 \leq x_i < 500) \\
&\quad min(f_{Sch}(x)) = f(420,9687,\ldots,420,9687) = 0
\end{aligned} \tag{2.9}
$$

$$
\begin{aligned}
f_{Ras}(x) &= 10n + \sum_{i=1}^{n}\big(x_i^2 - 10\cos(2\pi x_i)\big) \\
&\quad (-5,12 \leq x_i < 5,12) \\
&\quad min(f_{Ras}(x)) = f(0,0,\ldots,0) = 0
\end{aligned} \tag{2.10}
$$

$$
\begin{aligned}
f_{Gri}(x) &= 1 + \sum_{i=1}^{n}\frac{x_i^2}{4000} - \prod_{i=1}^{n}\Big(\cos\big(\frac{x_i}{\sqrt{i}}\big)\Big) \\
&\quad (-600 \leq x_i < 600) \\
&\quad min(f_{Gri}(x)) = f(0,0,\ldots,0) = 0
\end{aligned} \tag{2.11}
$$

$$
\begin{aligned}
f_{Ack}(x_i) = -a\cdot\exp\left(-b\sqrt{1/n\cdot\sum_{i=1}^{n}x_i^2}\right) - \exp\left(1/n\cdot\sum_{i=1}^{n}\cos\left(c\cdot x_i\right)\right) + a + e \\
a = 20; b = 0,2; c = 2\pi \qquad -32,768 < x_i < 32,768 \\
\min(f_{Ack}) = f(0,\ldots,0) = 0
\end{aligned} \tag{2.12}
$$

Informações sobre as funções-testes que não estão referenciadas neste capítulo podem ser

encontradas nos artigos onde elas foram utilizadas para testes[3]. A Figura 2.5 mostra os gráficos tridimensionais de onze funções.

[3]Vários códigos fonte de funções-teste estão disponíveis em `http://www.ewh.ieee.org/soc/es/May2001/14/Begin.htm`.

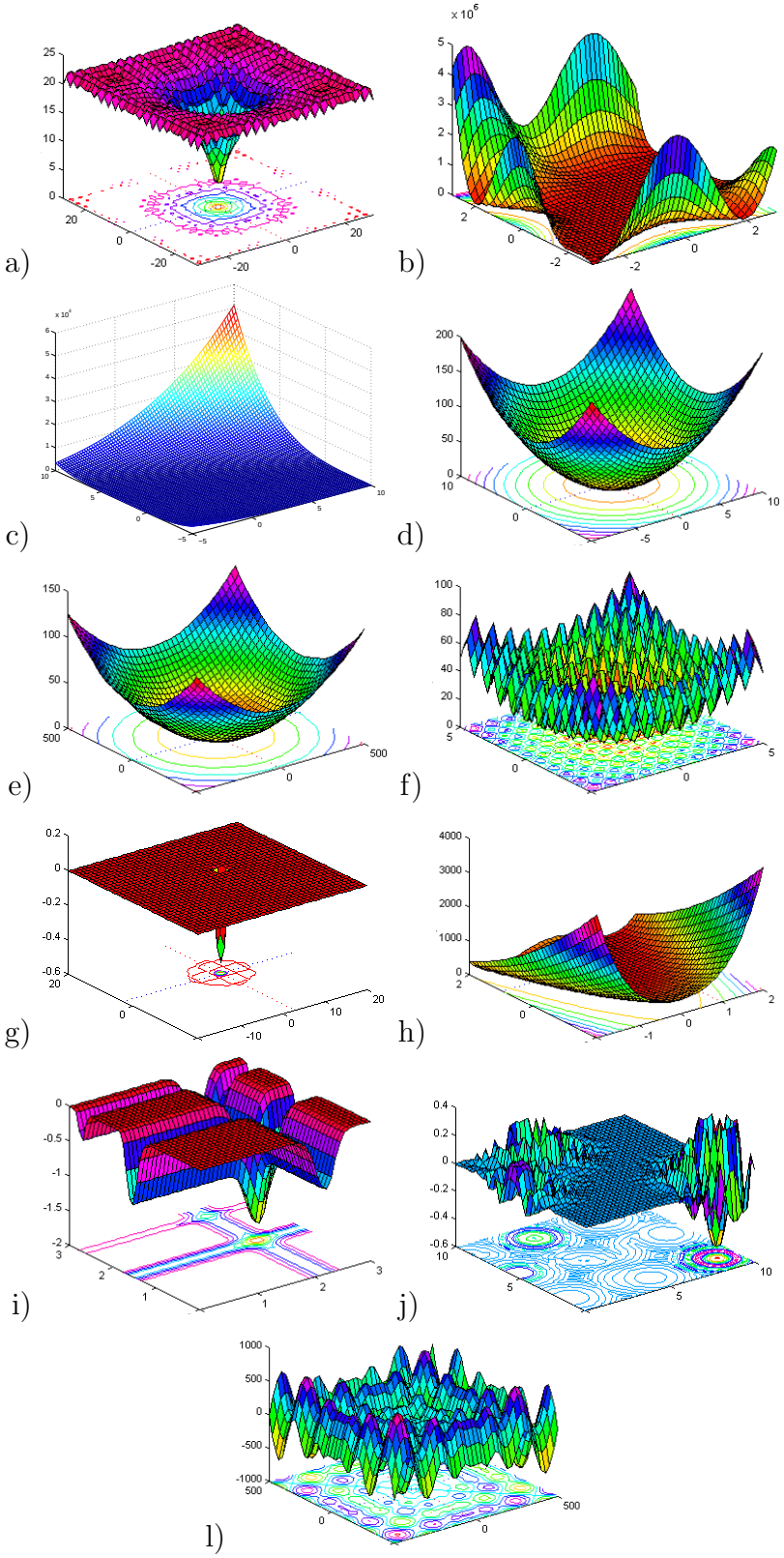


FIGURA 2.5 – Gráficos de: a) *Ackley*, b) *Goldstein*, c) *Zakharov*, d) *Sphere*, e) *Griewank*, f) *Rastrigin*, g) *Easom*, h) *Rosenbrock*, i) *Michalewicz*, j) *Langerman*, l)*Schwefel*.

CAPÍTULO 3

DETECÇÃO DE REGIÕES PROMISSORAS POR TREINAMENTO POPULACIONAL EM HEURÍSTICAS

Esquemas (*under-specified chromosomes*) são soluções incompletas para um dado problema e representam hiperplanos do espaço de busca (Goldberg, 1989). Quanto menor a ordem, mais soluções completas (estruturas) podem ser instanciadas a partir de um dado esquema, ou seja, mais pontos do espaço de busca podem ser representados por ele. Regiões promissoras, por sua vez, podem ser representadas por esquemas cujo hiperplano associado contenha soluções de qualidade. Pequenas modificações, introduzidas através de heurísticas de busca nessas soluções, podem gerar outras ainda melhores.

O Treinamento Populacional em Heurísticas (TPH), apresentado neste capítulo, estabelece um mecanismo de detecção de regiões promissoras baseado na avaliação de indivíduos através de heurísticas específicas do problema. Basicamente, indivíduos bem avaliados não podem ser melhorados por tais heurísticas e, por isso, são considerados referência no espaço de busca para as operações evolutivas.

3.1 Regiões promissoras e a teoria dos esquemas

Instanciar um esquema produz uma família de estruturas que representam amostras do espaço de busca com certa afinidade com o esquema que as produziu. A avaliação de esquemas pode ter um papel ativo no processo evolutivo, determinando quais subespaços de busca devem ser melhor explorados.

Os algoritmos genéticos *messy* (*messy-GA's*) trabalham diretamente com esquemas, utilizando codificação binária. A idéia geral é melhorar o desempenho do algoritmo genético através de um processo progressivo de junção de pequenas cadeias bem adaptadas, construindo progressivamente boas estruturas. Para avaliar os esquemas, inicialmente, Goldberg e seus colegas propuseram um procedimento baseado na geração aleatória de valores para as posições que ainda possuem indeterminação (#) e cada estrutura é avaliada pela função objetivo. O *fitness* do esquema é estimado através da qualidade média dos esquemas (Goldberg et al., 1989).

Posteriormente, Goldberg e seus colegas desenvolveram um método denominado *templates* competitivos para avaliar esquemas. A idéia é verificar se um esquema pode render uma instância (ótimo local), a partir da qual não se possa obter melhor avaliação através da alteração de um único bit. Inicialmente, um ótimo local é obtido através da técnica conhecida como *subida de encosta* (*hill-climbing*) e usada como *templates* competitivos

para complementar esquemas durante a avaliação. Eventualmente, estruturas melhores, resultantes desse processo, passam a representar novos *templates* competitivos (Goldberg et al., 1993; Knjazew e Goldberg, 2000).

Esquemas também podem ser avaliados diretamente, da mesma forma que estruturas, bastando para isso modificar a função objetivo para interpretar a ausência de informação (#). Em otimização combinatória, por exemplo, é possível a avaliação direta de esquemas. O maior desafio, entretanto, é manter na mesma população esquemas e estruturas competindo com base nessa avaliação em comum. Lorena e colaboradores propuseram uma avaliação bi-objetivo capaz de balancear a competição entre esquemas e estruturas (Lorena e Furtado, 2001).

Alternativamente, pode-se separar estruturas e esquemas em subpopulações diferentes (Haynes, 1997). A subpopulação de estruturas evolui normalmente e determina a avaliação da subpopulação de esquemas. Cada esquema, pode-se assim dizer, recebe um voto ponderado de uma estrutura *vizinha*. Nesse caso, instâncias são utilizadas como referência para avaliação de esquemas.

Em (Topchy et al., 1996), foi utilizado um algoritmo híbrido, combinando um algoritmo genético (AG) e um procedimento de gradiente descendente *regra-delta* para treinamento de redes neurais. Cada indivíduo codifica uma camada (não uma rede inteira), dessa forma reduzindo o espaço de busca e definindo subespaços onde métodos de gradiente sempre encontram a melhor combinação possível. Assim, o AG trabalha com uma população de esquemas que são avaliados por instanciação, aplicando-se uma busca local a partir do indivíduo.

Heurísticas construtivas podem ser úteis para instanciação de esquemas, permitindo uma estimativa de avaliação da região de busca que eles representam e a realização de operações de busca local a partir da solução completa (Oliveira e Lorena, 2002b). Por outro lado, heurísticas de melhoria podem ser inapropriadas para gerar vizinhança a partir de soluções incompletas. Entretanto, pode-se usar o mesmo mecanismo de busca local para identificar um bom esquema. Considerando que a solução melhorada reside na vizinhança da solução original, através de alguma relação de similaridade entre ambas, pode-se identificar um esquema e sua respectiva avaliação.

Na Figura 3.1, é mostrado um exemplo de avaliação indireta de esquemas através de procedimento heurístico. A partir de uma solução original, é gerado um conjunto de *soluções vizinhas*, buscando-se por uma solução melhor que a original. Quanto maior for a similaridade entre a solução original e a melhorada, maior será a ordem do esquema

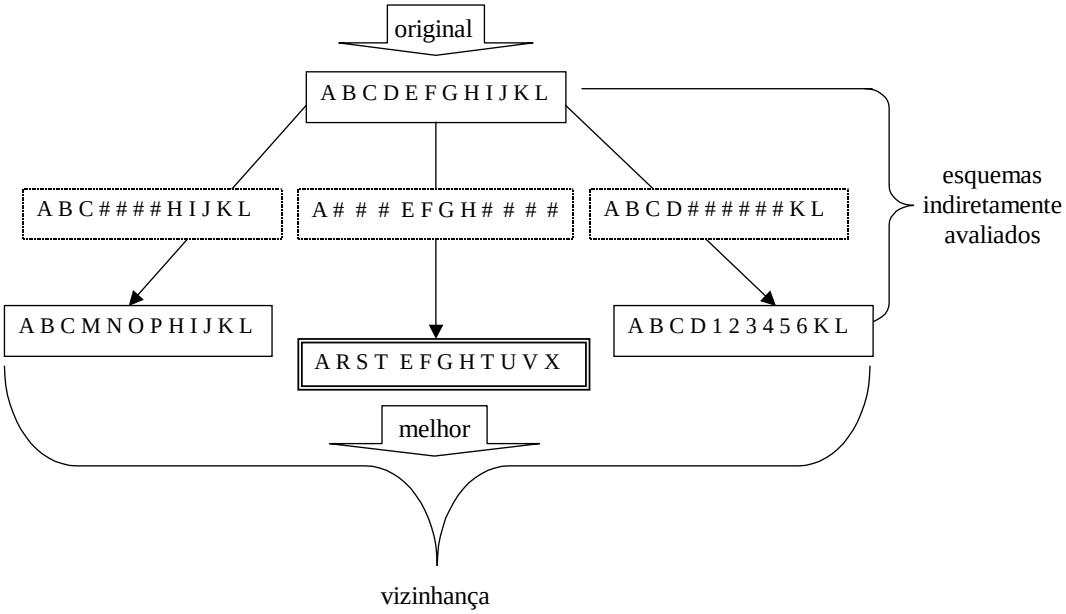


FIGURA 3.1 – Esquemas avaliados através de busca local.

avaliado. Observa-se que esse não é um procedimento explícito de avaliação de esquemas. O esquema é avaliado implicitamente através de busca local.

O Algoritmo Genético Construtivo ($\mathcal{AGC}$) foi proposto inicialmente para Problemas de Localização de Facilidades (Lorena e Furtado, 2001), posteriormente aplicado também a Problemas de Escala Temporal de Atividades (*Timetabling*) (Ribeiro Filho e Lorena, 2001), Coloração de Grafos (Ribeiro Filho e Lorena, 2000a), Células de Manufatura (Ribeiro Filho e Lorena, 2000b), Rotulação de Mapas (Yamamoto e Lorena, 2003) e Escalonamento *flow-shop* (Nagano et al., 2004).

Basicamente, o $\mathcal{AGC}$ difere dos messy-GA´s (Goldberg et al., 1993, 1989) por avaliar diretamente indivíduos do tipo esquema e também difere dos enfoques tradicionais por algumas outras características, tais como uma população dinâmica de esquemas e estruturas e pela possibilidade de serem empregadas heurísticas específicas para a avaliação de indivíduos.

O $\mathcal{AGC}$ é iniciado com uma população de esquemas, isto é, admite-se um certo percentual de indeterminação na composição de cada indivíduo inicialmente. Os esquemas são avaliados através de funções que determinam o quão promissor eles são. Os melhores esquemas são incentivados a recombinarem-se com outros de tal forma que, ao longo das gerações, novos esquemas e estruturas são produzidos, agregando mais informações sobre o problema e apresentando melhor avaliação. Um mecanismo de *poda* elimina os esquemas

e estruturas que não obtiverem boa avaliação. A população, dessa forma, possui tamanho dinâmico. Espera-se, ao final do processo, que estruturas de alta qualidade sejam obtidas, por meio de agregações sucessivas de informações sobre o problema.

Em sua forma original, o $\mathcal{AGC}$ promove a avaliação de cada indivíduo (estrutura ou esquema) através de duas funções f e g (avaliação $f - g$) que são construídas, considerando aspectos específicos do problema em questão, de forma que um indivíduo com $|f - g| \approx 0$ corresponda a uma solução ótima. O papel que essas funções de avaliação assumem depende da modelagem específica feita para o problema em questão (Lorena e Furtado, 2001).

Uma vez que esquemas e estruturas são avaliados da mesma forma, deve existir uma diferença numérica, em suas avaliações de função objetivo, proporcional à quantidade de informação presente neles. Os esquemas, por apresentarem menos informação, tendem a valores, para a função objetivo, menores que as estruturas. É desejável que estruturas sejam *construídas* a partir de esquemas ao longo das gerações e, portanto, haja uma maximização de função objetivo (fase construtiva), ao mesmo tempo em que indivíduos bem-adaptados vão sendo gerados e haja uma minimização de intervalo $f - g$ (fase ótima). O problema original é então transformado em um problema bi-objetivo (PBO) que contempla justamente essas duas fases.

O processo evolutivo, em geral, é suficiente para indicar quais esquemas devem ser avaliados e explorados. Mecanismos alternativos de detecção de esquemas promissores, baseados em conhecimento específico sobre o problema, podem ser embutidos, promovendo uma aceleração no processo de convergência para regiões promissoras do espaço de busca. A idéia aqui emergente é a de analisar heuristicamente o aprendizado acumulado de cada indivíduo e direcionar a exploração de regiões do espaço de busca ocupadas por aqueles mais adaptados. Diz-se que o processo evolutivo realiza um treinamento sobre a população, segundo uma heurística específica.

O Treinamento Populacional em Heurísticas (TPH) tem raízes sedimentadas na dupla avaliação de indivíduos e consiste em se avaliar a vizinhança da solução que o indivíduo representa, utilizando-se heurísticas específicas (chamadas de heurísticas de treinamento). O indivíduo (esquema ou estrutura) é bem avaliado em função do que ele é e não do que ele poderá vir a ser. As melhorias obtidas no processo de avaliação não são incorporadas ao genótipo (aprendizado *Baldwiniano*). São apresentados, a seguir, os fundamentos do TPH, bem como as aplicações desenvolvidas usando-se essa proposta e os experimentos que evidenciam sua consistência.

3.2 Fundamentos do Treinamento Populacional em Heurísticas

Algoritmos evolutivos trabalham, basicamente, por amostragem discreta do espaço de busca. Cada indivíduo da população possui uma avaliação de aptidão, geralmente expressa por meio de uma *função objetivo*, que está sempre vinculada ao problema. Em se tratando de um algoritmo genético canônico, a função objetivo é o único elo de ligação entre a população de soluções codificadas em indivíduos e o problema em questão.

Existem várias formas de se incorporar mais informações sobre o problema a um modelo evolutivo. Uma forma largamente usada é desenvolver novos operadores com heurísticas específicas embutidas. No contexto de treinamento populacional, heurísticas se encaixam como elos que se ligam diretamente ao problema (natureza, peculiaridades, estratégias para solução), permitindo uma sistemática redução do espaço de busca e, conseqüentemente, guiando a busca para regiões mais promissoras.

O termo treinamento é comumente usado para designar um processo no qual um modelo computacional, uma vez estimulado, consegue assimilar ou se adaptar a um determinado ambiente inicialmente desconhecido para ele. Os parâmetros livres de modelos em treinamento são ajustados segundo alguma medida de desempenho ou comportamento esperado.

O treinamento de uma rede neural, por exemplo, consiste no ajuste de seus parâmetros livres (número de neurônios, conexões sinápticas) visando a assimilação de um comportamento desejado. A capacidade de *aprender* um determinado ambiente (ou problema) é uma das características marcantes das RNAs (Haykin, 1994). De modo similar, algoritmos evolutivos realizam treinamento sobre uma população de indivíduos com relação a um objetivo esperado. Os indivíduos melhor adaptados à função objetivo são privilegiados, em detrimento de outros menos adaptados. Após o treinamento, o estado assumido pela população é capaz de apresentar soluções para o problema proposto.

O treinamento populacional engloba mecanismos com fins de induzir uma população de indivíduos a assumir uma característica desejada, ou seja, torná-la bem-adaptada a essa característica. Os mecanismos de indução são os tradicionais: privilegiar os indivíduos mais adaptados e penalizar os menos. A característica desejada (ou conjunto delas) são informações adicionais sobre o problema, inseridas no processo evolutivo por meio de heurísticas específicas. A função objetivo deixa, portanto, de ser a única ligação da população ao problema e passa a fazer par com informações colhidas heuristicamente, usadas como referência de treinamento.

A adaptação é um processo pelo qual os indivíduos vão adquirindo caracteres adequados para viverem em determinado ambiente ao longo de gerações. A dinâmica populacional tende a gerar uma farta recombinação genética. Aqueles indivíduos cujos fenótipos forem mais adequados à vida em um determinado ambiente irão participar mais ativamente do processo evolutivo. Assim, o que determina a adaptação do indivíduo não é vontade própria, mas um conjunto de características adquiridas ao longo da evolução e a importância dessas características para sobrevivência do indivíduo em um determinado meio ambiente.

A mesma idéia se aplica ao contexto de treinamento de população. A interação ocorre entre indivíduo e conhecimento heurístico de quão bom esse indivíduo é. Algoritmos heurísticos buscam na *vizinhança* por novas soluções viáveis heuristicamente *melhores* que as originais. Se houver uma solução heuristicamente *melhor*, diz-se que o indivíduo original não está bem-adaptado à *heurística de treinamento* empregada. Caso contrário, o indivíduo original é o *melhor* dentro da *vizinhança* estabelecida pela heurística.

Um procedimento abstrato para quantificar a adaptação de um indivíduo s_k com relação a uma heurística H, pode ser colocado como:

a) gerar um conjunto de soluções vizinhas a s_k;

b) avaliar heuristicamente cada uma delas;

c) identificar a melhor, s_V;

d) calcular a distância $\wp(s_k, s_V)$;

e) calcular a adaptação, considerando a distância $\wp(s_k, s_V)$.

As abstrações presentes nesse procedimento são: a geração e avaliação de soluções vizinhas a s_k e a *métrica de distância* entre s_k e s_V. Os passos a e b podem ser encapsulados em um único procedimento que identifique a melhor solução dentro de uma vizinhança estabelecida pela heurística de treinamento. O cálculo de distância $\wp$ pode utilizar alguma métrica a ser definida a *priori*.

A distância de *hamming* pode ser usada para comparar genótipos de indivíduos codificados em binário, assim como a métrica *euclidiana* pode medir distâncias em termos de fenótipos. Distâncias também podem ser medidas em termos de *fitness*, considerando a diferença entre as avaliações da função objetivo f. Dessa forma, a heurística de treinamento é usada para gerar uma vizinhança que é avaliada diretamente através de f e um indivíduo que não puder ser heuristicamente melhorado representa, hipoteticamente,

um mínimo (ou máximo) local segundo a heurística de treinamento. Uma outra métrica que pode ser utilizada diz respeito a movimentos heurísticos, ou seja, o número de movimentos necessários para transformar uma solução em outra. Alguns trabalhos têm mostrado que essa pode não ser uma tarefa trivial (Linhares, 2002; Reeves, C.R., 1999).

3.3 Formalização da proposta

Treinamento Populacional em Heurísticas (TPH) pode ser definido como:

$$TPH = \{P, \Theta, f, H, \wp, \delta\} \tag{3.1}$$

onde P é a população de indivíduos s_k, amostrados do espaço de busca codificado S, logo $s_k \in \mathsf{S}$. Θ é o conjunto de operadores evolutivos específicos para gerar novas soluções em S.

$$\Theta : \mathsf{S} \rightarrow \mathsf{S} \tag{3.2}$$

A função objetivo f realiza o mapeamento de S para $\mathbb{R}$, i.e.:

$$f : \mathsf{S} \rightarrow \mathbb{R} \tag{3.3}$$

A heurística de treinamento H pode ser definida pelo par:

$$H = \{\varphi^H, g\} \tag{3.4}$$

O conhecimento sobre o problema é usado para definir a função heurística g:

$$g : \mathsf{S} \rightarrow \mathbb{R} \tag{3.5}$$

A relação de vizinhança φ^H é uma operação do tipo:

$$\varphi^H : \mathsf{S} \rightarrow \mathsf{S}^l \tag{3.6}$$

onde $l+1$ é o *tamanho da vizinhança* ou o número de soluções vizinhas a s_k, incluindo ele mesmo ($l \geq 2$). Assim, φ^H pode ser entendida como um conjunto de soluções obtidas a partir de uma solução qualquer s_k através de alguma heurística específica:

$$\varphi^H(s_k) = \{s_k, s_{v1}, s_{v2}, \ldots, s_{vl}\} \tag{3.7}$$

A avaliação de $\varphi^H(s_k)$ pode usar a própria função objetivo f, considerando que ela fornece uma avaliação precisa de quão bom é cada solução vizinha a s_k. Assim para problemas de minimização pode-se redefinir g como:

$$g(s_k) = f(s_V), s_V \in \{s_1, s_2, \ldots, s_l, s_k\} \subseteq \varphi^H, f(s_V) \leq f(s_k) \tag{3.8}$$

O melhor vizinho de s_k é denotado por s_V. O conceito de distância $\wp$ é relativo ao esforço necessário para se alcançar s_V a partir de s_k através de movimentos da heurística H. Menor distância significa maior adaptação de s_k, independentemente da natureza do problema (minimização ou maximização). Nas aplicações desenvolvidas neste trabalho, $\wp(s_k, s_V)$ é dada por:

$$\wp(s_k, s_V) = |f(s_k) - g(s_k)| \tag{3.9}$$

Finalmente, a população é ordenada por uma função δ que relaciona a adaptação total do indivíduo. Duas propostas de TPH são sugeridas neste trabalho. O enfoque *construtivo* e o *não-construtivo*. O primeiro trabalha inicialmente com uma população de esquemas e o segundo não permite esquemas. A principal razão para a existência dessas duas propostas é agregar maior flexibilidade ao TPH, permitindo o emprego de uma gama maior de tipos de codificação, heurísticas e operadores evolutivos.

Esquemas nem sempre trazem alguma vantagem para o processo de codificação, especialmente em aplicações que se propõem trabalhar com o espaço de genótipos idêntico ao de fenótipos (alfabetos de cardinalidade superior). Em aplicações assim, muitas vezes os parâmetros da função objetivo estão fortemente relacionados e a eliminação de um deles inviabiliza a avaliação de indivíduos de forma coerente (Oliveira e Lorena, 2002c).

3.3.1 Proposta construtiva

No enfoque *construtivo* do TPH, uma população de esquemas é gerada e avaliada diretamente através das funções $f - g$, essencialmente da mesma forma que o $\mathcal{AGC}$ original (Lorena e Furtado, 2001). A função g incorpora a heurística de treinamento que acrescenta informação específica do problema para avaliar o quão promissora é a família de soluções associadas ao esquema. Para evitar ambigüidades com relação a nomes, esta proposta é denotada por $\mathcal{AGC}^H$ e significa um algoritmo genético construtivo com treinamento populacional em heurísticas.

Assim como no $\mathcal{AGC}$ original, o objetivo do problema é transformado em um problema de minimização de intervalos que contemplam o desempenho com relação a uma fase ótima e uma fase construtiva. Posto na formulação do TPH, o problema bi-objetivo (PBO) é definido como:

$$\begin{cases} \min \quad |f(s_k) - g(s_k)| \\ \\ \max \quad f(s_k) \\ \end{cases} \tag{3.10}$$

As funções f e g operam na mesma faixa de valores, isto é, no intervalo de possíveis valores de função objetivo. Define-se um certo G_{max} como uma estimativa de um limite superior para todos os possíveis valores que as funções f e g podem assumir. No início do processo evolutivo, G_{max} pode ser calculado analiticamente, considerando a instância do problema em questão, ou pode ser estimado por amostragem dentro da população inicial. O intervalo $G_{max} - g(s_k)$ fornece a distância, em termos de avaliação, entre o indivíduo s_k e o limite superior G_{max}. Essa distância é usada no *ranking* construtivo (δ_{cons}) para estimar o quão completa é a solução representada pelo indivíduo.

A Figura 3.2 mostra uma representação de um espaço de busca formado por 16 possíveis soluções com 4 bits. Cada solução possui 4 outras soluções vizinhas que podem ser obtidas por movimentos heurísticos do tipo troca de um bit. Trata-se de uma situação possível na qual a avaliação f de uma solução pode corresponder à avaliação g de outra. Esse exemplo se trata de uma minimização, por isso, $g(s_k) \leq f(s_k)$. Identificando cada solução por um índice que cresce da esquerda para direita e de cima para baixo, o conjunto $\{s_4, s_5, s_7, s_{12}\}$ são vizinhas a s_8 que por sua vez apresenta $|f(s_8) - g(s_8)| > 0$.

Da mesma forma que o $\mathcal{AGC}$ original, o processo evolutivo no $\mathcal{AGC}^H$ é conduzido

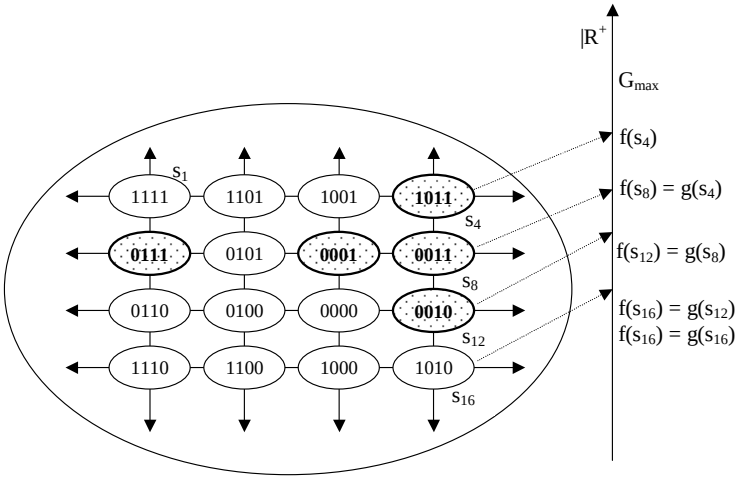


FIGURA 3.2 – Exemplo de valores de avaliação $f - g$.

considerando um limiar de rejeição que contempla ambos os objetivos do PBO. Seja um parâmetro α e uma constante $d \in [0, 1]$, a expressão:

$$|f(s_k) - g(s_k)| > d \cdot G_{max} - \alpha \cdot d \cdot [G_{max} - g(s_k)] \tag{3.11}$$

apresenta uma condição para rejeição de um esquema ou estrutura s_k. O lado direito da equação 3.11 é o limiar, composto do valor esperado para a minimização do intervalo $d \cdot G_{max}$. A constante d tem o papel de equilibrar os componentes da equação. No caso do $\mathcal{AGC}^H$, $d \in [0,10\ 0,20]$.

A equação 3.11 pode ser examinada variando-se o valor de α. Para $\alpha = 0$, esquemas e estruturas, são avaliados unicamente pela diferença $f-g$. À medida que α é incrementado, os esquemas são mais penalizados que as estruturas pela diferença $G_{max} - g$.

O parâmetro α está relacionado com o tempo de evolução. Considerando que os bons esquemas precisam ser preservados para serem recombinados, α inicia a partir de *zero* e é lentamente incrementado, de geração em geração. A população no tempo de evolução α, denotada por P_α, possui tamanho dinâmico de acordo com o valor de α e pode, inclusive, ser esvaziada durante o processo. O parâmetro α é isolado na equação 3.12 e o lado direito corresponde a um valor de δ_{cons} que é atribuído a cada indivíduo da população:

$$\delta_{cons}(s_k) = \frac{d \cdot G_{max} - |f(s_k) - g(s_k)|}{d \cdot [G_{max} - g(s_k)]} \tag{3.12}$$

Quando são criados, esquemas e estruturas recebem os seus correspondentes valores de *ranking* que será comparado com o parâmetro α, a cada geração, para efeito de eliminação dos indivíduos mal-adaptados ($\delta_{cons}(s_k) \leq \alpha$). Assim, os indivíduos com maior δ são melhores com relação ao PBO, sobrevivem por um número maior de gerações e, conseqüentemente, se reproduzem mais.

Os indivíduos são mantidos na população, em ordem descendente de *ranking*. Indivíduos são selecionados, recombinados e, eventualmente, sofrem mutação. O procedimento de seleção privilegia os indivíduos do topo do *ranking* (mais adaptados). A recombinação, por sua vez, constrói novos indivíduos a partir da informação contida nos indivíduos pais. A mutação, por sua vez, é algum tipo de busca local agressiva que ocorre com alta probabilidade sobre as estruturas geradas no processo de recombinação.

3.3.2 Proposta não-construtiva

Em otimização combinatória, a fase construtiva do $\mathcal{AGC}^H$ funciona como esperado: a medida em que estruturas vão sendo *construídas*, vão ocupando o topo do *ranking*, o que garante a elas uma maior participação no processo evolutivo. Isso ocorre sobretudo por que o acréscimo de informação nos esquemas minimiza o intervalo $[G_{max} - g(s_k)]$.

Entretanto, não se pode garantir, para todos os tipos de problemas, que estruturas sempre tenham valor de função objetivo maior que esquemas. Por exemplo, em problemas de otimização numérica, soluções incompletas não podem ser avaliadas como se fossem um subconjunto do espaço de busca. Muitas vezes os parâmetros da função objetivo estão fortemente relacionados e a eliminação de um deles poderia até causar descontinuidades no espaço de busca, como uma divisão por *zero*. Em face disso, um enfoque não-construtivo, chamado de *Algoritmo de Treinamento Populacional* ($\mathcal{ATP}$), foi idealizado.

Apesar de não trabalhar explicitamente com esquemas, pode-se dizer que o enfoque não-construtivo avalia blocos genéticos comuns às soluções dentro de uma mesma vizinhança. O conceito de vizinhança, adotado neste trabalho, diz respeito à família de soluções geradas segundo uma heurística de treinamento. Indivíduos *vizinhos* possuem genes em comum ou *similares*, sugerindo certa *proximidade* entre genótipos.

Quando um indivíduo s_k é o melhor dentro da vizinhança $\varphi^H(s_k)$, diz-se que ele está plenamente adaptado à heurística de treinamento H e assim todo o bloco genético que o compõe está igualmente bem avaliado. Quando existe um ou mais indivíduos melhores na vizinhança, o indivíduo é penalizado no processo de avaliação proporcionalmente a distância entre ele o melhor vizinho s_V. Os genes *similares* entre s_V e s_k correspondem

aos blocos genéticos bem avaliados.

A Figura 3.3 mostra dois exemplos de vizinhança, com diferentes codificações, onde as transições entre soluções simbolizam esquemas avaliados. Comparado a um grafo, s_{vi} e s_k são vértices e a aresta ligando-os corresponde a um esquema de ordem igual aos genes *similares*. Observa-se que, no exemplo com codificação numérica, admite-se um certo limiar ($\pm 0,5$, no caso) para considerar genes *similares*. Esses exemplos são ilustrativos de como, em tese, se avalia um esquema indiretamente através de uma busca heurística em uma vizinhança.

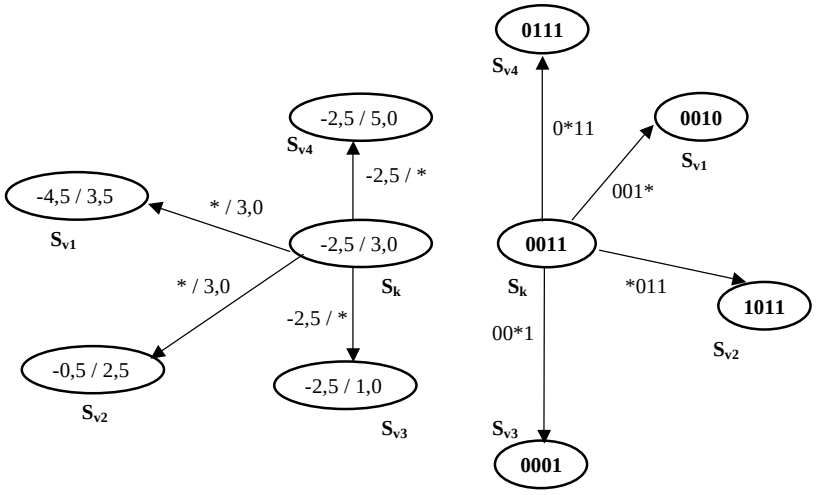


FIGURA 3.3 – Exemplos de avaliação indireta de esquemas.

No *ranking* não-construtivo, além do componente referente a adaptação do indivíduo com relação à heurística de treinamento, $\min\{f - g\}$, um segundo componente considera a avaliação da função objetivo. No caso de problemas de minimização, quanto maior for $G_{max} - g(s_k)$ melhor será a avaliação do indivíduo. A constante d mais uma vez é usada para equilibrar ambos os componentes e, no caso do $\mathcal{ATP}$, é geralmente ajustada para um valor próximo a $1/G_{max}$. O *ranking* não-construtivo, δ_{ncons}, é dado por:

$$\delta_{ncons}(s_k) = d \cdot [G_{max} - g(s_k)] - |f(s_k) - g(s_k)| \tag{3.13}$$

Da mesma forma que no $\mathcal{AGC}^H$, a heurística H é usada para gerar um conjunto de soluções a partir de s_k que serão avaliadas para se escolher a melhor. A melhor avaliação encontrada é usada como valor de $g(s_k)$. Mais uma vez está sendo utilizada como

avaliação heurística a própria função objetivo aplicada sobre uma vizinhança estabelecida por H. A distância, mais uma vez, é calculada em termos de função objetivo.

A equação 3.13 é específica para problemas de minimização. Para maximização, é introduzido um limitante inferior G_{min}:

$$\delta(s_k) = d \cdot [g(s_k) - G_{min}] - \mid f(s_k) - g(s_k) \mid \tag{3.14}$$

Da mesma forma que acontece com o $\mathcal{AGC}^H$, antes da inicialização da população inicial, é gerado um pequeno conjunto aleatório de indivíduos e a maior (ou menor) avaliação de função objetivo dentre eles é atribuída ao limitante G_{max} (ou G_{min}). Durante o processo evolutivo, todo indivíduo com $g \geq G_{max}$ (ou $g \leq G_{min}$) é descartado.

Cada indivíduo s_k gerado, primeiramente recebe uma avaliação de função objetivo $f(s_k)$. A seguir, a heurística de treinamento H é aplicada para avaliar a vizinhança $\varphi^H(s_k)$ e atribuir um valor para $g(s_k)$. A diferença $\mid f - g \mid$ é tirada do valor de $d \cdot [G_{max} - g]$, penalizando o indivíduo. Dessa forma, os indivíduos com melhores *rankings* são aqueles com baixos valores de função objetivo e que estejam bem-adaptados à heurística de treinamento.

O *Algoritmo de Treinamento Populacional* ($\mathcal{ATP}$) foi aplicado para problemas de minimização numérica sem restrições (Oliveira e Lorena, 2002c), para problemas de seqüenciamento de padrões e problemas de escalonamento de tripulação de ônibus (Mauri e Lorena, 2004).

3.4 Aplicação para problemas de seqüenciamento de padrões

O TPH foi aplicado para dois problemas similares de seqüenciamento de padrões: Minimização de Pilhas Abertas (*Minimization of Open Stack Problem - MOSP*) e Leiaute de Matriz-Porta (*Gate Matrix Layout Problem - GMLP*). Os aspectos teóricos desses dois problemas são basicamente os mesmos. A diferença é tão somente relativa a seus enunciados. Uma descrição do *MOSP* mais detalhada é apresentada na seção 2.6, juntamente com as peculiaridades do *GMLP*.

3.4.1 Aspectos de modelagem

O relacionamento entre *GMLP*'s e *MOSP*'s permitiu que o $\mathcal{AGC}^H$ e o $\mathcal{ATP}$ fossem modelados para ambos de forma similar em vários aspectos como codificação, avaliação, recombinação e mutação. Na Tabela 3.1, é mostrado como soluções foram codificadas

para *GMLP*'s e *MOSP*'s. Os símbolos '?' aparecem nas colunas associadas a posições do indivíduo com indeterminações (#). As posições com "#" podem ser preenchidas com qualquer símbolo usado na representação, preservando a viabilidade da solução.

TABELA 3.1 – Codificação de indivíduos para *GMLP*.

	1	0	0	0	1	1	0	1	0
	0	0	0	1	0	1	1	0	0
	0	1	1	0	0	0	0	1	1
	1	0	1	0	0	1	0	1	0
	0	0	0	1	0	1	1	0	1
	1	0	1	0	0	0	0	0	1
	0	0	0	1	0	0	1	0	0
s_j =	7	2	5	9	3	4	6	1	8

	1	?	0	?	?	?	0	1	?
	0	?	0	?	?	?	1	0	?
	0	?	1	?	?	?	0	1	?
	1	?	1	?	?	?	0	1	?
	0	?	0	?	?	?	1	0	?
	1	?	1	?	?	?	0	0	?
	0	?	0	?	?	?	1	0	?
s_k =	7	#	5	#	#	#	6	1	#

O objetivo do *MOSP* (ou *GMLP*) é minimizar o número de máximo de pilhas abertas (ou trilhas). Com intuito de se gerar uma paisagem de aptidão mais suave, com estados intermediários de avaliação entre soluções com mesmo *mpa*, um segundo objetivo é introduzido: minimizar o tempo no qual as pilhas permanecem abertas (*tpa*). O segundo objetivo induz que pilhas sejam fechadas o mais rapidamente possível, ao mesmo tempo, liberando os pedidos de clientes com um mínimo de retardo. Do ponto de vista do *GMLP*, o *tpa* reflete o total de metal necessário para cobrir todo o circuito *VLSI* e também deve ser minimizado. O *tpa* pode ser calculado através da soma de todos os "1" 's de $\mathcal{Q}^{\pi}$.

Para *MOSP* (ou *GMLP*), a função f reflete o custo total de uma dada permutação π de padrões (ou portas). Ambos, *mpa* (objetivo primário) e *tpa* (objetivo secundário) são considerados nesta formulação de *fitness*:

$$f(s_k) = I \cdot J \cdot \max_{i \in I} \left\{ \sum_{j \in J} \mathcal{Q}_{ij}^{\pi} \right\} + \sum_{i \in I} \sum_{j \in J} \mathcal{Q}_{ij}^{\pi} \tag{3.15}$$

onde o produto $I \cdot J$ é um peso que reforça o objetivo primário. Essa formatação de função objetivo tem sido usada por outras abordagens não-evolutivas para *GMLP* (Linhares et al., 1999).

Para o *MOSP* (*GMLP*), a função de aptidão f retrata o custo da solução codificada no indivíduo em termos de máximo de pilhas abertas (ou número de trilhas) e de tempo de pilha aberta (total de metal) usados para processar a permutação.

No caso do $\mathcal{AGC}^{H}$, se s_k é um esquema, as colunas com rótulos indefinidos (#) são desconsideradas, como se tais colunas não existissem. A matriz $\mathcal{Q}^{\pi}$ usada para calcular

$f(s_k)$ conteria somente colunas representadas por rótulos definidos.

3.4.2 Heurística 2-Opt

O algoritmo 2-Opt foi proposto primeiramente por (Croes, 1958) e é baseado em trocas entre pares de arestas de grafos que representem soluções para problemas de permutação. O movimento de troca remove duas arestas, quebrando o circuito em dois caminhos, e os reconecta da outra maneira possível.

Em problemas do tipo *caixeiro viajante* (PCV) existe um custo (distância) associado a cada par de arestas. Com essa informação, movimentos 2-Opt somente são realizados se as novas arestas incluídas possuírem custo menor que as removidas. É esperado que a substituição de arestas de maior custo por outras de menor custo reduza o custo total do circuito ou seja a distância total entre todas as cidades a serem percorridas pelo caixeiro viajante.

Como já foi dito anteriormente, as funções objetivo em problemas de seqüenciamento de padrões não podem ser computadas somando-se valores que dependem apenas dos padrões adjacentes. Toda a seqüência de padrões tem que ser avaliada para que se saiba se a troca de arestas resultou em um ganho de avaliação. Portanto, as heurísticas do tipo k-Opt (2-Opt, 3-Opt e Lin-Kernighan) não podem ser aplicadas para problemas de seqüenciamento da mesma forma como são utilizadas para o PCV.

A idéia básica, baseada na troca de arestas, permanece. Entretanto, para cada movimento testado, uma avaliação completa de função objetivo deve ser computada. O custo computacional significativo que outras possibilidades k-Opt imporiam ao TPH foi decisivo para a opção que foi feita de se empregar a 2-Opt como heurística de treinamento.

A troca de 2 padrões (*2-Troca*) também se oferece como uma opção de heurística de treinamento com complexidade similar à 2-Opt ($O(n^2)$). A 2-Troca tem sido usada como operador de busca local para problemas de seqüenciamento de padrões em trabalhos recentes (Linhares, 2002). A Figura 3.4 mostra as heurísticas 2-Opt e 2-Troca aplicadas a uma mesma solução inicial.

Alguns estudos sobre *paisagens de fitness* associadas a PCV sugerem que os mínimos locais tendem a se concentrar em regiões conhecidas como *maciço central* ou (*grande vale*). O maciço central contem somente uma pequena porção do espaço de busca completo. Estratégias têm sido propostas para detectar esse maciço e explorá-lo de forma conveniente. Quando comparadas, a heurística 2-Troca tem apresentado desempenho inferior a 2-Opt (Preux e Talbi, 1999).

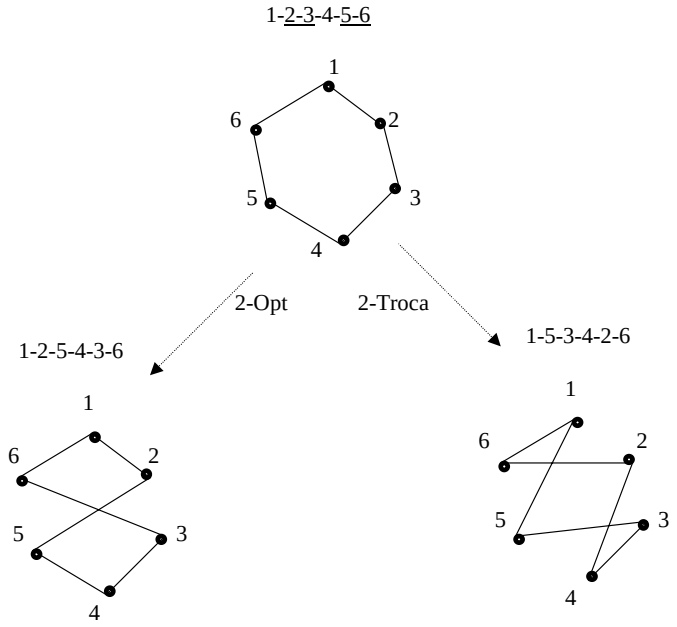


FIGURA 3.4 – Exemplo de movimentos 2-Opt e 2-Troca.

Em (Fonlupt et al., 1999), conclui-se que existe uma forte correlação entre o *fitness* e distância entre soluções vizinhas geradas através da heurística 2-Opt. Essa correlação cai quando é utilizada a heurística 2-Troca. Em outras palavras, pequenas variações no genótipo (soluções dentro de uma mesma vizinhança) geram pequenas variações de *fitness*, provando que, uma vez atingindo-se o maciço, a heurística 2-Opt é mais confiável para gerar soluções que permaneçam nele. O uso de 2-Troca como único mecanismo de geração de soluções vizinhas causa o que é chamado de *diluição do maciço* (Preux e Talbi, 1999).

Seja π uma permutação de padrões qualquer representada por um grafo não direcionado e acíclico $\mathcal{G}(\pi)$. Cada posição de π significa um vértice no grafo $\mathcal{G}(\pi)$ e subentende-se a existência de uma aresta apenas entre vértices consecutivos. O primeiro e o último vértices não são consecutivos, diferentemente do que ocorre com permutações para o PCV.

A variação do algoritmo 2-Opt proposta neste trabalho avalia cada nova permutação segundo a função objetivo. As posições p e q indicam as arestas $< p, p+1 >$ e $< q, q+1 >$, respectivamente, a serem removidas a cada movimento. Os possíveis pares (p, q) são gerados sistematicamente até que todas as possíveis combinações de arestas sejam removidas. Pressupõe-se simetria, ou seja, a aresta $< 1, 2 >$ é igual a aresta $< 2, 1 >$, por exemplo.

$s_V := s_k$
Seja $\mathcal{G}(s_k)$ o grafo de s_k
$i := rand(1, I-1)$
for $p := i$ até $i + l - 1$ **do**
  for $q := p + 1$ até $i + l$ **do**
    Remova as arestas $< p, p+1 >$ e $< q, q+1 >$ de $\mathcal{G}(s_k)$
    Reconecte os vértices de $\mathcal{G}(s_k)$ gerando $\mathcal{G}'(s_v^{pq})$
    if $f(s_v^{pq})$ é melhor que $f(s_V)$ **then**
      $s_V := s_v^{pq}$
      $g(s_k) := f(s_v^{pq})$
    end if
  end for
end for

Observa-se que esse procedimento apenas estima o melhor vizinho, uma vez que apenas uma parte l de φ^{2-Opt} (vizinhança 2-Opt) é avaliada. O tamanho de vizinhança foi definido na equação 3.6 e é um parâmetro de ajuste, tanto do $\mathcal{AGC}^H$ quanto do $\mathcal{ATP}$. O número de permutações a serem avaliadas é dado por:

$$nAval^{2opt} = \frac{l \cdot (l-1)}{2} \tag{3.16}$$

Primeiramente, uma posição inicial é escolhida ao acaso e um processo iterativo é iniciado a partir daí, tomando duas a duas posições como referência para movimentos do tipo 2-Opt. Por exemplo, para um indivíduo com 10 padrões ($I = 10$) e $l = 4$, a partir da posição 2, o seguinte conjunto de posições de referência é gerado: {(2,3),(2,4),(2,5),(3,4),(3,5),(4,5)}, i.e., um total de $\frac{4 \cdot 3}{2} = 6$ diferentes permutações.

Cada par de posições indica as duas arestas a serem removidas. Depois de removidas as arestas, os vértices são reconectados em uma nova disposição, gerando uma nova solução, que é vizinha a anterior por um movimento do tipo 2-Opt. Essa nova solução é então avaliada para efeito do cálculo da função g.

Posições de referência consecutivas causam remoção de arestas consecutivas que não podem ser reconectadas em uma nova disposição. Movimentos do tipo 2-Opt nesses casos gerariam soluções idênticas. Para que pontos de referência consecutivos possam ser úteis para o processo, foi criado um movimento 2-Opt alternativo. A Figura 3.5 mostra duas formas de se aplicar movimentos do tipo 2-Opt para duas situações distintas: pontos de referência consecutivos e não-consecutivos.

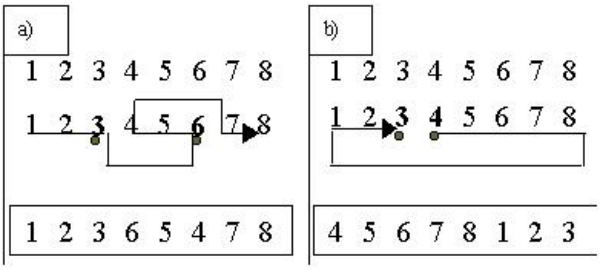


FIGURA 3.5 – Exemplos de movimentos 2-Opt com pontos de referência: a)não-consecutivos, e b)consecutivos.

O algoritmo 2-Opt proposto é de complexidade $O(n^2)$. Cada avaliação de função objetivo para o *MOSP/GMLP* é também $O(n^2)$. Portanto, o procedimento total de avaliar φ^{2-Opt} nesta aplicação é $O(n^4)$. Mantendo-se $l < I$ e fixo, pode-se reduzir significativamente o custo computacional de g.

3.4.3 Heurística construtiva de *Faggioli-Bentivoglio*

Um outro procedimento usado como heurística de treinamento é baseado no algoritmo proposto por (Faggioli e Bentivoglio, 1998). Trata-se de uma heurística construtiva na qual, a partir de um padrão inicial, novos padrões vão sendo anexados, minimizando diferenças entre o último anexado e os subseqüentes até que uma permutação completa seja obtida.

A variação do algoritmo proposto neste trabalho avalia apenas a permutação final, proveniente do processo de construção. O parâmetro l é usado para indicar o tamanho do grupo inicial de padrões. Seja π' uma permutação parcial qualquer contendo $I - l$ padrões copiados de um indivíduo qualquer s_k, ou seja:

$$\pi' = s_{k,1\ldots I-l} \tag{3.17}$$

O conjunto de padrões a serem incluídos posteriormente em π' é denotado por:

$$\varpi = s_k - \{\pi'\} \tag{3.18}$$

A vizinhança φ^{fagg} é definida por todas as estruturas que se iniciam com π'. A função de avaliação $g(s_k)$ avalia o melhor vizinho de s_k construído iterativamente, a partir de π', minimizando-se a soma das diferenças entre padrões adjacentes, segundo um critério

de três estágios definidos em (Faggioli e Bentivoglio, 1998).

No primeiro estágio, dentre os padrões de ϖ, é escolhido aquele que menos abrir novas pilhas (ou iniciar novas redes, no caso do *GMLP*). Uma pilha é aberta quando o padrão seqüenciado contém um tipo de ítem que ainda não tenha sido empilhado, isto é, o $i-$ésimo ítem apresente uma transição $0-1$ do padrão anterior para o próximo.

No segundo estágio, dentre os padrões selecionados no primeiro estágio, é escolhido aquele que fechar um maior número de pilhas (ou finalizar um maior número de redes). Uma pilha é fechada (ou removida) quando o padrão seqüenciado finaliza um tipo de ítem que esteja aberto, isto é, ocorra uma transição $1-0$ no $i-$ésimo ítem de um padrão para o próximo.

No último estágio, caso ainda haja empate com relação aos dois estágios anteriores, é escolhido o padrão que mantiver um maior número de pilhas abertas, ou seja, continuar produzindo um maior número de ítens. A produção de um ítem continua quando um padrão seqüenciado contém um ítem para o qual que já fora criada uma pilha, isto é, o $i-$ésimo ítem apresente uma transição $1-1$ de um padrão para o próximo.

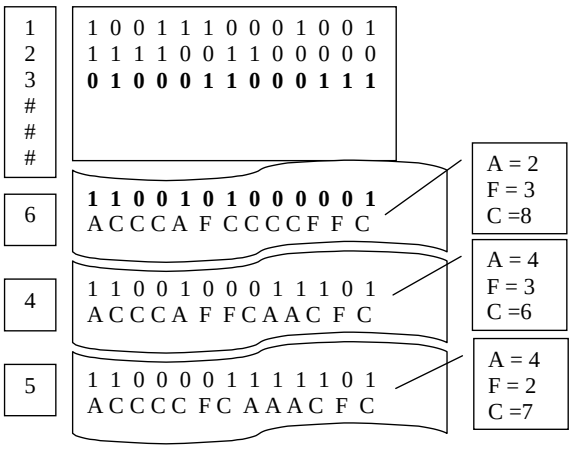


FIGURA 3.6 – Exemplo dos três critérios da heurística de *Faggioli-Bentivoglio*.

A Figura 3.6 mostra uma solução incompleta contendo três padrões (1,2 e 3) e outros três padrões candidatos com seus respectivos valores para os critérios da heurística de *Faggioli- Bentivoglio*. Observa-se que apenas o último padrão inserido na permutação (no caso, o padrão 3) é confrontado com os subseqüentes. Para cada um dos subseqüentes, ao lado, é mostrado o número de novas pilhas abertas (A), número de pilhas fechadas (F) e processos que continuam (C). O padrão 6 é o melhor com relação ao primeiro critério

(menor A) e deverá ser anexado à permutação. O padrão 4 é o segundo melhor. Ele leva vantagem com relação ao padrão 5 por fechar um maior número de pilhas (F= 3).

Se ao final dos três estágios, mais de um padrão for candidato a ser anexado, um deles é escolhido ao acaso. A estrutura resultante desse processo é então avaliada segundo a função objetivo. Dessa forma, para avaliar φ^{fagg}, g^{fagg} é definida como:

$$g^{fagg}(s_k) = \min\left\{f(s_V), f(s_k)\right\}, s_V = \min\left\{Fagg(s_i)\right\}, s_i \in \varphi^{fagg} \tag{3.19}$$

onde $Fagg(s_i)$ reflete os três estágios de similaridade proposto em (Faggioli e Bentivoglio, 1998).

3.4.4 Implementação do $\mathcal{AGC}^H$

Como foi previamente mencionado, o $\mathcal{AGC}^H$ é iniciado com uma população de esquemas, isto é, admite-se um percentual de indeterminação, $\rho(i)$, na composição da permutação que compõe cada indivíduo da primeira geração. Após ser atribuído um número fixo (dependente de $\rho(i)$) de rótulos "#" para posições aleatórias, alguns rótulos de padrões são colocados nas posições restantes da permutação, sempre conservando a viabilidade da solução (sem repetição de rótulos).

Ao longo das gerações, a população tende a aumentar pela adição de novos indivíduos bem-adaptados. O modelo de atualização é não-geracional, ou seja, os descendentes passam a competir com os demais indivíduos tão logo sejam gerados. Esquemas tendem a ser removidos juntamente com as estruturas mal-adaptadas.

A seleção de estruturas e/ou esquemas para recombinação é conduzida privilegiando os indivíduos com maior *ranking* δ. Dois indivíduos são selecionados para cada recombinação. O primeiro é chamado de indivíduo base s_{base} e o segundo de indivíduo guia s_{guia}. O indivíduo base é selecionado aleatoriamente das primeiras posições da população P_α, ou seja, da elite da população segundo o ranking δ. O indivíduo guia é selecionado aleatoriamente dentro da população inteira. Esse algoritmo de seleção é chamado de seleção *base-guia* (Lorena e Furtado, 2001).

O percentual da população que é considerada elite, $\rho(e)$, é um parâmetro do $\mathcal{AGC}^H$ e deve ser ajustado tornando o processo de seleção menos ou mais elitista. A Figura 3.7 mostra a organização da população com relação ao processo de seleção.

A recombinação é determinante para o bom desempenho de qualquer algoritmo genético.

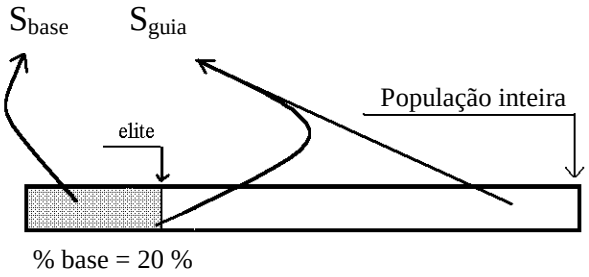


FIGURA 3.7 – Seleção *base-guia*.

O cruzamento *base-guia* tem como objetivo agrupar blocos supostamente de qualidade do indivíduo base nos descendentes, mas aceitando desvios induzidos por blocos do indivíduo guia (Lorena e Furtado, 2001).

A Tabela 3.2 mostra duas formas diferentes de cruzamento *base-guia* para problemas de seqüenciamento que têm a mesma filosofia: compor um novo indivíduo viável a partir da soluções (completas ou parciais) de dois outros previamente selecionados. As formas privilegiam a informação contida no indivíduo base, desde que esta não seja uma indeterminação "#". Caso o gene do indivíduo base seja "#" ou já esteja presente no indivíduo novo, o gene do indivíduo guia é considerada na composição. Caso este também seja "#" ou já esteja presente no indivíduo novo, uma indeterminação é posta nessa posição.

TABELA 3.2 – Exemplos de cruzamentos *base-guia*.

Base	1	5	#	8	#	#	3	4		1	5	#	8	#	#	3	4
Guia	#	2	#	4	3	8	5	7		#	2	#	4	3	8	5	7
Novo	1	5	#	8	3	#	#	4		1	5	#	8	#	#	3	4

A diferença entre essas duas formas está na ordem em que os indivíduos são inspecionados. Na primeira, os indivíduos são inspecionados da esquerda para a direita, executando-se o algoritmo que é mostrado a seguir (Oliveira e Lorena, 2002b):

```
for i := 1 até I do
  if s_base(i) = # e s_guia(i) = # then
    s_novo(i) := #
  end if
  if s_base(i) ≠ # e s_guia(i) = # then
    if s_base(i) ∉ s_novo then
      s_novo(i) := s_base(i)
    else
      s_novo(i) := #
    end if
  end if
  if s_base(i) = # e s_guia(i) ≠ # then
    if s_guia(i) ∉ s_novo then
      s_novo(i) := s_guia(i)
    else
      s_novo(i) := #
    end if
  end if
  if s_base(i) ≠ # e s_guia(i) ≠ # then
    if s_base(i) ∉ s_novo then
      s_novo(i) := s_base(i)
    else
      if s_guia(i) ∉ s_novo then
        s_novo(i) := s_guia(i)
      else
        s_novo(i) := #
      end if
    end if
  end if
end for
```

Na segunda forma, toda a informação do indivíduo base é copiada para o novo indivíduo e, só depois disso, os genes do indivíduo guia são postos no novo indivíduo, caso não haja perda de viabilidade (Lorena e Furtado, 2001). O $\mathcal{AGC}^H$ pode ser resumido no seguinte pseudo-código:

```
Dados G_max e d;
α := 0;
ε := 0,005
Inicialize(P_α);
for all s_k ∈ P_α do
  compute f(s_k), g(s_k), δ(s_k);
end for
while não condição de parada do
  while número de cruzamentos do
    Seleção(s_base, s_guia);
    Complemente(s_base, s_comp);
    Mutação(s_comp);
    Compute f(s_comp), g(s_comp), δ(s_comp);
    Atualize(s_comp, P_α);
    Cruzamento(s_base, s_guia, s_novo);
    Compute f(s_novo), g(s_novo), δ(s_novo);
    Atualize(s_novo, P_α);
  end while
  α := α + ε;
  for all s_k ∈ P_α e δ(s_k) < α  do
    Elimine(s_k, P_α) ;
  end for
end while
```

O incremento ε é um passo linear que incrementa o limiar de rejeição descrito na seção 3.3.1. A condição de parada é determinada pelo experimento em questão. O processo de complemento de indivíduo, mostrado no pseudo-código do $\mathcal{AGC}^H$, é necessário para que se possa aplicar o operador de mutação. A mutação é uma busca local do tipo 2-Opt bem mais *agressiva* que o algoritmo apresentado como heurística de treinamento 2-Opt. Ela é aplicada somente sobre estruturas, daí a necessidade de, sempre que indivíduos base forem esquemas, completá-los. A mutação é explicada posteriormente, pois faz parte da implementação tanto do $\mathcal{AGC}^H$ e $\mathcal{ATP}$.

O operador de complemento substitui os rótulos "#" por rótulos de padrões que ainda não estejam presentes no indivíduo base. Para o preenchimento de esquemas, é usado um algoritmo baseado em minimização de diferenças similar à heurística construtiva de *Faggioli-Bentivoglio*. A diferença se resume basicamente na forma como estruturas são construídas. Um padrão candidato é aquele que ainda não pertence ao esquema e que deverá ser posto em uma posição i qualquer, contendo "#". A adjacência da posição i é

comparada com cada um dos rótulos candidatos, sendo escolhido aquele que apresentar a menor *diferença inter-padrão*. Seja a um padrão candidato e $s_{k,i-1}$ o padrão contido na $i-$ésima posição de s_k, a *diferença inter-padrão*, denotada por $dif(a,i)$, é dada por:

$$dif(a,i) = \sum_{j \in J} xor(\mathcal{P}(a,j), \mathcal{P}(s_{k,i-1},j)) + \sum_{j \in J} xor(\mathcal{P}(a,j), \mathcal{P}(s_{k,i+1},j)) \tag{3.20}$$

onde *xor* é a operação lógica *ou-exclusivo* que é aplicada item-a-item da matriz binária $\mathcal{P}$ (instância).

O rótulo candidato que possuir menor dif substitui o rótulo "#", passando a interferir no processo de complemento. Observa-se que, eventualmente, $\mathcal{P}(s_{k,i-1},j)$ ou $\mathcal{P}(s_{k,i+1},j)$ podem não existir para $i = 1$ e para $i = I$, respectivamente. Nesses casos dif é computado somente com um dos dois termos. A Figura 3.8 mostra um exemplo de operação de complemento. Como pode ser observado, há 4 padrões candidatos à posição 5, com indeterminação. O padrão candidato 3 apresenta a menor *diferença inter-padrão* ($dif = 2$) com relação às colunas vizinhas e deverá ser inserido na permutação.

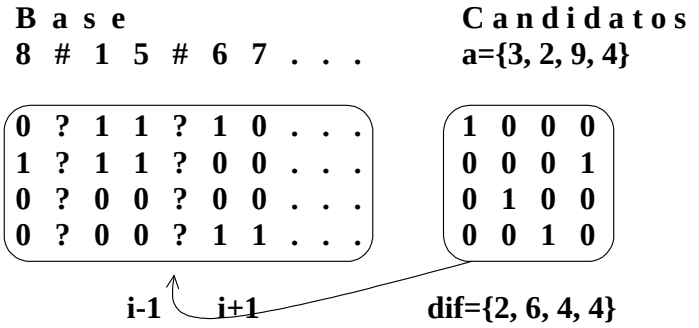


FIGURA 3.8 – Heurística de complemento de indivíduos base para mutação.

3.4.5 Implementação do $\mathcal{ATP}$

O $\mathcal{ATP}$ trabalha com uma população dinâmica de estruturas. O limiar de rejeição, α, é inicializado com o valor do δ do indivíduo menos adaptado (menor δ). Durante o processo evolutivo, α é atualizado através de incrementos adaptativos que consideram o tamanho atual da população, $|P|$, bem como a atual faixa de valores de *ranking* (do melhor, $\delta_{|P|}$, ao pior, δ_1), além do número de gerações que faltam para o término da evolução. O

incremento adaptativo ε é dado por:

$$\varepsilon = \xi \cdot |P| \cdot \frac{(\delta_{|P|} - \delta_1)}{T - t} \tag{3.21}$$

onde T é o total de gerações previstas para o processo evolutivo e ξ é uma constante usada para dar mais ou menos velocidade ao processo de esvaziamento da população. Se os incrementos forem maiores, mais rapidamente indivíduos mal-adaptados serão eliminados aumentando a possibilidade de esvaziamento da população.

Assim como ocorre com o $\mathcal{AGC}$ original e o $\mathcal{AGC}^H$, no $\mathcal{ATP}$ a população tende a crescer, inicialmente, aceitando todos os novos indivíduos. Com o passar das gerações, o aumento do valor de α determina uma quantidade cada vez maior de indivíduos a serem eliminados. O processo de seleção também utiliza o algoritmo *base-guia* sugerido por (Lorena e Furtado, 2001) e explicado na seção 3.4.4.

O $\mathcal{ATP}$ permitiu a implementação de algoritmos de cruzamento encontrados na literatura, como o *Partially-Matching Crossover (PMX)* (Goldberg, 1989), uma vez que ele não trabalha com esquemas. O cruzamento que apresentou melhores resultados foi uma variante do *Order Crossover (OX)* chamado de *Block Order Crossover (BOX)* (Syswerda, 1989).

Os pais A e B são combinados em um único filho através de cópia de blocos aleatórios de ambos os pais. Blocos copiados de um pai não são copiados do outro, mantendo a viabilidade do indivíduo resultante. Esse tipo de cruzamento atende melhor aos requisitos do TPH pois tende a quebrar menos os blocos construtivos que possivelmente estão mais presentes no indivíduo base. A Figura 3.9 mostra um exemplo do *BOX*.

Com relação aos algoritmos implementados, o $\mathcal{ATP}$ é equivalente ao $\mathcal{AGC}^H$, com exceção do trecho relativo a complemento de esquemas. Além disso, o incremento de α deixa de ser linear e passa a ser adaptativo. A mutação 2-Opt, por sua vez, é aplicada em ambos os algoritmos, sempre no indivíduo base.

3.4.6 Mutação 2-Opt

A mutação 2-Opt é usada tanto no $\mathcal{AGC}^H$ e $\mathcal{ATP}$ para intensificar a busca em regiões promissoras representadas pelos indivíduos bem-adaptados à heurística de treinamento. Os indivíduos mais promissores são aqueles considerados base.

O processo de busca local é similar ao que foi descrito na heurística 2-Opt, na

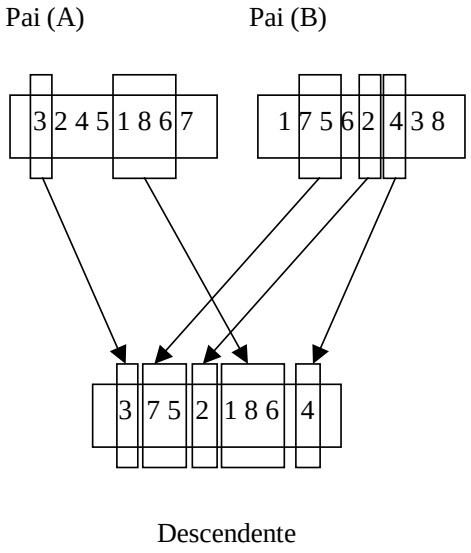


FIGURA 3.9 – Exemplo de *BOX*.

seção 3.4.2. Entretanto, como heurística de treinamento, é verificada apenas uma vizinhança de tamanho l. Como busca local, a heurística 2-Opt é usada para gerar sucessivas vizinhanças, cada uma de tamanho l. O número de vizinhanças depende do sucesso da busca a cada vizinhança. Um limite é imposto para evitar um custo computacional excessivo. Pode-se dizer que a mutação 2-Opt depende de três parâmetros: a probabilidade $\rho(m)$ de ocorrer, o tamanho l de cada vizinhança e o número máximo m de vizinhanças a serem verificadas a cada busca local.

A mutação explora uma árvore de busca, na qual a melhor estrutura encontrada, a cada nível, é usada como ponto de partida para o próximo nível. O processo pára quando não se conseguir obter uma solução melhor ou quando for atingido o limite de m vizinhanças. A Figura 3.10 mostra a árvore de busca gerada pelo algoritmo de mutação.

3.4.7 Resultados computacionais

O $\mathcal{AGC}^H$ foi codificado em *C ANSI* e aplicado a mais de 300 instâncias *MOSP* e *GMLP* encontradas na literatura[1]. Todos os resultados obtidos ao longo do período em que durou este trabalho estão condensados nesta seção. São apresentados resultados que mostram a competitividade dos algoritmos propostos com relação a outras metaheurísticas recentes. Alguns experimentos relacionados ao comportamento dos enfoques construtivo e não-construtivo são também apresentados visando dar suporte às conjecturas que estão colocadas ao final deste capítulo.

[1]Todas as instâncias estão disponíveis em `http://www.lac.inpe.br/~lorena/instancias.html`

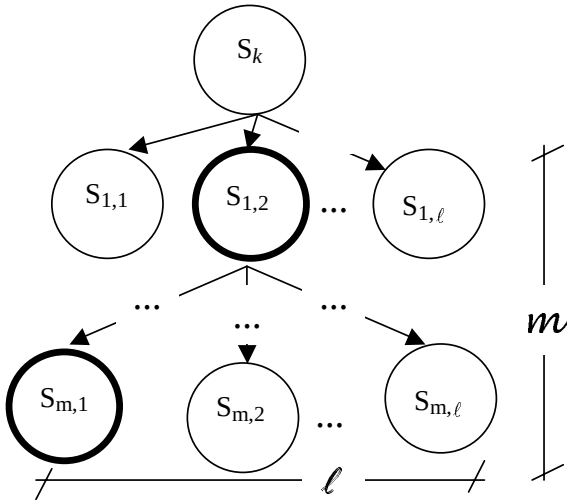


FIGURA 3.10 – Árvore de busca gerada pela mutação 2-Opt.

3.4.7.1 $\mathcal{AGC}^H$ aplicado a instâncias *GMLP*

Esta seção sintetiza um primeiro conjunto de experimentos, no qual o $\mathcal{AGC}^H$ foi aplicado a instâncias *GMLP*, empregando a heurística 2-Opt para treinamento em uma máquina *Intel Pentium II 266MHz e memória de 64MB* (Oliveira e Lorena, 2002b). As instâncias são mostradas na Tabela 3.3 e são as maiores encontradas na literatura. Algumas delas são consideradas desafios, pois apenas recentemente foram encontradas as melhores soluções dessas instâncias (Linhares et al., 1999).

O ajuste de parâmetros do $\mathcal{AGC}^H$ foi baseado na experiência obtida em aplicações anteriores do $\mathcal{AGC}$ (Lorena e Furtado, 2001; Ribeiro Filho e Lorena, 2001). Com relação aos parâmetros relativos à mutação 2-Opt (l e m), procurou-se balancear ganho/custo, evitando-se execuções muito lentas que nem sempre logravam êxito. Os parâmetros do $\mathcal{AGC}^H$ são mostrados a seguir:

- constante de equilíbrio $d = 0,15$ (usualmente com valores entre $0,10$ e $0,20$);
- incremento de limiar de rejeição $\varepsilon = 0,005$;
- tamanho inicial da população $|P_0| = 100$;
- percentual de indeterminação da população inicial $\rho(i) = 50\%$;
- percentual de indivíduos elite $\rho(e) = 20\%$;
- percentual de mutação 2-Opt sobre indivíduos base $\rho(m) = 100\%$;
- tamanho de vizinhança 2-Opt $l = 20$;

TABELA 3.3 – Instâncias *GMLP* usadas no primeiro conjunto de testes do $\mathcal{AGC}^H$.

instância	portas	redes	solução
wli	10	11	4
wsn	25	17	8
v4000	17	10	5
v4050	16	13	5
v4090	27	23	10
v4470	47	37	9
x0	48	40	11
w1	21	18	4
w2	33	48	14
w3	70	84	18
w4	141	202	27

- número de vizinhanças 2-Opt avaliadas a cada mutação $m = 20$;

A Tabela 3.4 mostra os resultados obtidos pelo $\mathcal{AGC}^H$, em 10 execuções, comparados aos obtidos pelo enfoque conhecido como *Microcanonical Optimization (MCO)*, ambos executando em plataformas de *hardware* similares (Linhares et al., 1999). O *MCO* é similar ao enfoque *recozimento simulado*, mas ao invés de emular o comportamento de um sistema sobre controle de temperatura, ele considera vários sistemas termicamente isolados, i.e., sistemas que não interagem com seu ambiente e tem energia constante (Linhares et al., 1999). O *MCO* foi escolhido por ter sido o melhor enfoque até então encontrado na literatura.

A condição de término para uma execução do $\mathcal{AGC}^H$ foi encontrar a melhor solução conhecida ou 50 gerações sem haver nenhuma melhora na melhor solução encontrada. O $\mathcal{AGC}^H$ encontrou todas as melhores soluções conhecidas em até 10 execuções. A freqüência com que essas melhores soluções foram encontradas também se mostrou satisfatória. Para *GMLP*'s considerados pequenos (com até 40 portas), como as instâncias *wsn, wli, v4000, v4050* a freqüência de acerto do $\mathcal{AGC}^H$ foi de 100% (exceto para a instância v4090 com 90%). Para instâncias consideradas grandes (acima de 70 portas), como *w3* (70 portas) e *w4* (141 portas), a freqüência ficou em 50% (5 em 10 execuções) e 30% (3 em 10 execuções), respectivamente.

O *MCO* também obteve todas as melhores soluções conhecidas. Entretanto, para algumas instâncias foram necessárias 1000 execuções. O tempo total de execução do *MCO* em 1000 execuções foi estimado com base no tempo médio de uma execução. Em (Linhares et al., 1999), alguns desses tempos são realmente pequenos (inferiores a 1 segundo). Para esses casos, o tempo médio de uma execução foi estimado como $0,01$ segundo.

TABELA 3.4 – Comparação entre $\mathcal{AGC}^H$ e *MCO* para instâncias *GMLP*.

	MCO			$\mathcal{AGC}^H$		
Instância	Trilhas		Tempo (s)	Trilhas	Tempo (s)	Execuções
	10 execuções	1000 execuções	1000 execuções	10 execuções	10 execuções	bem-sucedidas
wli	5	4	10,0	4	5,00	10
wsn	8	-	10,0	8	15,00	10
v4050	5	-	10,0	5	5,00	10
v4000	6	5	10,0	5	5,00	10
v4470	10	9	700,0	9	665,00	10
v4090	10	-	100,0	10	20,33	9
x0	11	11	700,0	11	755,60	8
w1	4	-	10,0	4	10,00	10
w2	14	-	400,0	14	185,00	10
w3	21	18	3900,0	18	3062,50	5
w4	32	27	617000	27	52246,67	3

O tempo de uma execução do $\mathcal{AGC}^H$ foi superior ao do *MCO*. Entretanto, o $\mathcal{AGC}^H$ se mostrou bem mais robusto, necessitando de bem menos execuções para encontrar as melhores soluções conhecidas. Por esse prisma, o tempo total de 10 experimentos do $\mathcal{AGC}^H$ é inferior ao tempo total dos 1000 experimentos que foram realizados pelo *MCO* (Oliveira e Lorena, 2002b).

Com relação a freqüência de acerto em 1000 execuções, em (Linhares et al., 1999), os autores apenas comentam que foi encontrada a melhor solução conhecida em $36,3\%$ das execuções para a instância *wli* (10x11). Não há nenhuma informação sobre as demais instâncias. Um percentual similar, em torno de 30% (3 em 10 execuções), foi encontrado pelo $\mathcal{AGC}^H$ para a instância *w4*, bem maior (141x202) e teoricamente bem mais difícil.

3.4.7.2 $\mathcal{AGC}^H$ e $\mathcal{ATP}$ aplicados a instâncias *MOSP*

O segundo conjunto de experimentos envolve o $\mathcal{AGC}^H$ e o $\mathcal{ATP}$. Ambos foram aplicados a instâncias *MOSP*, empregando a heurística 2-Opt para treinamento. Os resultados do $\mathcal{AGC}^H$ foram extraídos de (Oliveira e Lorena, 2002a).

Um conjunto de 300 instâncias *MOSP* encontradas em (Faggioli e Bentivoglio, 1998) foi utilizado nestes experimentos. Essas instâncias têm diferentes números de padrões ($I \in \{10, 15, 20, 25, 30, 40\}$), cada um com diferentes números de ítens ($J \in \{10, 20, 30, 40, 50\}$). Cada grupo (par I, J) compreende 10 instâncias com diferentes soluções, algumas delas somente encontradas em trabalhos recentes (Oliveira e Lorena, 2002a; Linhares, 2002).

O $\mathcal{AGC}^H$ foi ajustado da mesma forma que na seção anterior. A seguir, são mostrados os parâmetros do $\mathcal{ATP}$. Observa-se que a constante d pode ser calculada dinamicamente e não chega a ser um parâmetro propriamente dito. Várias execuções foram necessárias para se chegar a esses valores de parâmetros para o $\mathcal{ATP}$.

- constante de equilíbrio $d = 1/G_{max}$(ou seja, um valor determinado dinamicamente);
- incremento de limiar de rejeição é adaptativo;
- tamanho inicial da população $|P_0| = 50$;
- percentual de indeterminação da população inicial $\rho(i) = 0\%$ (não trabalha com esquemas);
- percentual de indivíduos elite $\rho(e) = 20\%$;
- percentual de mutação 2-Opt sobre indivíduos base $\rho(m) = 20\%$;
- tamanho de vizinhança 2-Opt $l = 20$;
- número de vizinhanças 2-Opt avaliadas a cada mutação $m = 20$;

Ambos $\mathcal{AGC}^H$ e $\mathcal{ATP}$ tiveram seus desempenhos comparados com outros enfoques encontrados na literatura (Faggioli e Bentivoglio, 1998): uma busca tabu (TS) baseada em um processo de seleção otimizada de movimentos e uma busca local generalizada (GLS) que emprega a heurística construtiva de *Faggioli-Bentivoglio* em uma busca tabu simplificada que somente aceita movimentos com melhoria (Faggioli e Bentivoglio, 1998). Além desses, um outro método foi incluído na comparação: o método Collective (COL) que explora medidas de distâncias entre permutações e emprega a heurística de 2-Troca para guiar uma algoritmo do tipo *recozimento simulado* (Linhares, 2002).

A Tabela 3.5 mostra as médias de *mpa* obtidas por cada método em cada grupo de instâncias. Não se sabe exatamente qual é a solução de cada instância separadamente. Os resultados têm sido publicados sempre como uma média das soluções de cada grupo. As melhores médias que aparecem na Tabela 3.5 não necessariamente indicam que todas as melhores soluções foram encontradas. Mas pode-se presumir, pelos resultados obtidos nestes experimentos, que as médias das melhores soluções de cada grupo estão presentes nessa tabela.

O $\mathcal{AGC}^H$ e o $\mathcal{ATP}$ obtiveram as menores médias, mas este último foi melhor em dois grupos de instâncias (valores em negrito) para os quais ainda não havia sido encontradas soluções melhores. Especificamente, as duas instâncias que determinaram a diferença nos grupos $20x40$ e $25x40$ são *p2040n6* e *p2540n3* (sexta e terceira instâncias, respectivamente). Por isso, essas instâncias passaram a ser consideradas as mais difíceis e foram utilizadas em outros experimentos neste trabalho. Apesar do aparente insucesso do $\mathcal{AGC}^H$ nessas instâncias, novos experimentos com mais execuções (em torno de 20) mostraram que seu desempenho é similar ao $\mathcal{ATP}$.

TABELA 3.5 – Melhores médias encontradas para cada grupo de instâncias *MOSP*.

I	J	COL	TS	GLS	$\mathcal{AGC}^H$	$\mathcal{ATP}$
10	10	5,5	5,5	5,5	5,5	5,5
	20	6,2	6,2	6,2	6,2	6,2
	30	6,1	6,1	6,2	6,1	6,1
	40	7,7	7,7	7,7	7,7	7,7
	50	8,2	8,2	8,2	8,2	8,2
15	10	6,6	6,6	6,6	6,6	6,6
	20	7,2	7,2	7,5	7,2	7,2
	30	7,3	7,4	7,6	7,3	7,3
	40	7,2	7,3	7,4	7,2	7,2
	50	7,4	7,6	7,6	7,4	7,4
20	10	7,5	7,7	7,5	7,5	7,5
	20	8,5	8,7	8,6	8,5	8,5
	30	9,0	9,2	8,9	8,8	8,8
	40	8,6	8,6	8,7	8,6	**8,5**
	50	7,9	8,0	8,2	7,9	7,9
25	10	8,0	8,0	8,0	8,0	8,0
	20	9,8	9,8	9,9	9,8	9,8
	30	10,6	10,7	10,6	10,5	10,5
	40	10,4	10,7	10,6	10,4	**10,3**
	50	10,0	10,1	10,2	10,0	10,0
30	10	7,8	7,8	7,8	7,8	7,8
	20	11,2	11,2	11,2	11,1	11,1
	30	12,2	12,6	12,2	12,2	12,2
	40	12,1	12,6	12,4	12,1	12,1
	50	11,2	12,0	11,8	11,2	11,2
40	10	8,4	8,4	8,4	8,4	8,4
	20	13,0	13,1	13,1	13,0	13,0
	30	14,5	14,7	14,6	14,5	14,5
	40	15,0	15,3	15,3	14,9	14,9
	50	14,6	15,3	14,9	14,6	14,6

3.4.7.3 Outras heurísticas de treinamento

O objetivo neste terceiro conjunto de experimentos é verificar como diferentes heurísticas de treinamento interferem no desempenho do TPH. Uma outra versão do $\mathcal{ATP}$ foi construída e nela é empregada a heurística construtiva de *Faggioli-Bentivoglio* (*fagg*) para treinamento. Para diferenciar da versão anterior, usando 2-Opt, ambas são denotadas por $\mathcal{ATP}^{2opt}$ e $\mathcal{ATP}^{fagg}$. Também foi construída uma versão $\mathcal{AGC}^{fagg}$.

A Tabela 3.6 mostra a comparação entre as quatro versões de algoritmos usando TPH em 20 execuções para as instâncias *MOSP p2040n6* e *p2540n3*. Foram consideradas

as médias das melhores soluções encontradas (MSE), o número de chamadas à função objetivo (CFO), e a média do tempo de execução em segundos (TE). CFO e TE foram computados considerando somente as execuções bem-sucedidas, i.e, aquelas que encontraram a melhor solução conhecida para essas instâncias. O percentual de execuções bem-sucedidas (PS) também é apresentado na tabela.

TABELA 3.6 – Comparação entre $\mathcal{ATP}$ e $\mathcal{AGC}^{H}$ empregando as heurísticas 2-Opt e construtiva de *Faggioli-Bentivoglio*.

	$\mathcal{ATP}^{2opt}$				$\mathcal{ATP}^{fagg}$			
Instâncias	MSE	PS (%)	CFO (10^6)	TE (s)	MSE	PS (%)	CFO (10^6)	TE (s)
p2040n6	$8,7 \pm 0,5$	30	0,578	20,0	$8,7 \pm 0,5$	30	0,405	14,3
p2540n3	$10,7 \pm 0,5$	30	0,414	21,7	$10,8 \pm 0,4$	20	0,708	30,0
	$\mathcal{AGC}^{2opt}$				$\mathcal{AGC}^{fagg}$			
p2040n6	$8,9 \pm 0,3$	10	0,322	11,0	$8,9 \pm 0,3$	10	0,303	10,0
p2540n3	$10,7 \pm 0,5$	30	0,747	32,3	$10,9 \pm 0,3$	10	1,019	36,0

Segundo a Tabela 3.6, $\mathcal{ATP}^{2opt}$ obteve os melhores MSE e PS nas instâncias *p2040n6* e *p2540n3*. Ainda com relação a MSE e PS, $\mathcal{ATP}^{fagg}$ e $\mathcal{AGC}^{2opt}$ obtiveram resultados semelhantes. Não fica nítido uma versão vencedora, exceto pelo critério CFO: $\mathcal{AGC}^{2opt}$ fez menos chamadas à função objetivo. Tal fato pôde ser constatado para outras instâncias.

Apesar do $\mathcal{AGC}^{2opt}$ ter alcançado resultados bons, mesmo os melhores em alguns casos, o $\mathcal{AGC}^{fagg}$ falhou em encontrar bons resultados e teve o pior desempenho dentre todos. Uma justificativa para isso está relacionada à forma como a implementação da heurística construtiva de *Faggioli-Bentivoglio* trata esquemas. Assim como a heurística 2-Opt, ela desconsidera padrões contendo rótulos "#" e isso pode não ser uma forma eficiente de tratar a falta de informação.

A Tabela 3.7 mostra uma comparação entre os enfoques baseados em TPH e um algoritmo memético paralelo (AMP) (Mendes e Linhares, 2004) para a instância *GMLP w4*, a maior instância encontrada na literatura. O AMP apresenta um novo operador de busca local baseado em 2-Troca provido de um mecanismo de redução de movimentos que descarta trocas inúteis evitando chamadas desnecessárias à função objetivo. A Tabela 3.6 mostra a comparação entre $\mathcal{ATP}^{fagg}$, $\mathcal{ATP}^{2opt}$, $\mathcal{ATP}^{2opt}$ e o AMP em 10 execuções. A legenda PS, nessa tabela, se refere ao número de vezes em que a melhor solução foi encontrada.

Observando a Tabela 3.6, $\mathcal{AGC}^{2opt}$ obteve os melhores MSE e PS para *w4*. $\mathcal{ATP}^{fagg}$ e

TABELA 3.7 – Comparação entre $\mathcal{ATP}^{2opt}$, $\mathcal{ATP}^{fagg}$, $\mathcal{AGC}^{2opt}$ e AMP para a instância *w4*.

	MSE	PS	CFO		MSE	PS	CFO
$\mathcal{ATP}^{2opt}$	28,6	2	8.488.438	$\mathcal{AGC}^{2opt}$	28,0	3	6.537.706
$\mathcal{ATP}^{fagg}$	28,3	2	9.330.802	AMP	29,4	2	9.428.591

$\mathcal{ATP}^{2opt}$ têm desempenhos razoavelmente similares com relação a MSE, mas o último efetuou menos chamadas à função objetivo. Todos os enfoques baseados em treinamento populacional apresentaram desempenho melhor que AMP.

Os algoritmos empregando a heurística construtiva de *Faggioli-Bentivoglio* deveriam ter realizado menos chamadas à função objetivo que aqueles usando a heurística 2-Opt. Porém, isso não pôde ser comprovado nos experimentos realizados. As duas prováveis causas são: a) a mutação 2-Opt, presente em todas as versões, domina o número de chamadas à função objetivo e a heurística de treinamento não é relevante para a computação total de CFO; ou b) a heurística 2-Opt é realmente superior no sentido de estabelecer as mais promissoras regiões de busca e o processo guiado por ela tem desempenho superior que compensa o seu aparente maior custo computacional.

3.4.7.4 Convergência e conservação de blocos construtivos

Todas as versões desenvolvidas de TPH são agora comparadas com relação a convergência. O experimento consiste em 10 execuções com cada uma das versões, aferindo-se, em intervalos regulares de chamadas à função objetivo, o *fitness* do melhor indivíduo até então obtido.

A trilha de convergência de cada versão pode ser observada na Figura 3.11. O gráfico mostra a média da melhor solução encontrada a cada amostragem que ocorre a cada 10^5 avaliações de função objetivo para a instância *w4* nas 10 execuções do experimento. Essa instância foi escolhida por se tratar da maior instância conhecida, na qual provavelmente, a mutação 2-Opt tem menor participação no desempenho total do método.

O gráfico de convergência foi dividido em duas figuras, dando maior clareza às diferenças de desempenhos dos métodos. Observa-se que um quinto algoritmo teve sua convergência acrescentada ao gráfico. Trata-se de um algoritmo genético com população dinâmica sem busca local (AGPD). Esse algoritmo tem o papel de mostrar uma referência de desempenho de um algoritmo sem nenhuma das contribuições apresentadas neste trabalho. O AGPD emprega a mesma estrutura algorítmica dos demais, mas utiliza a própria função objetivo (equação 3.15) para ordenar indivíduos na população dinâmica.

Ademais, ele emprega o cruzamento BOX, uma mutação que realiza movimentos 2-Troca aleatórias e a seleção *base-guia*, nos mesmos moldes dos demais.

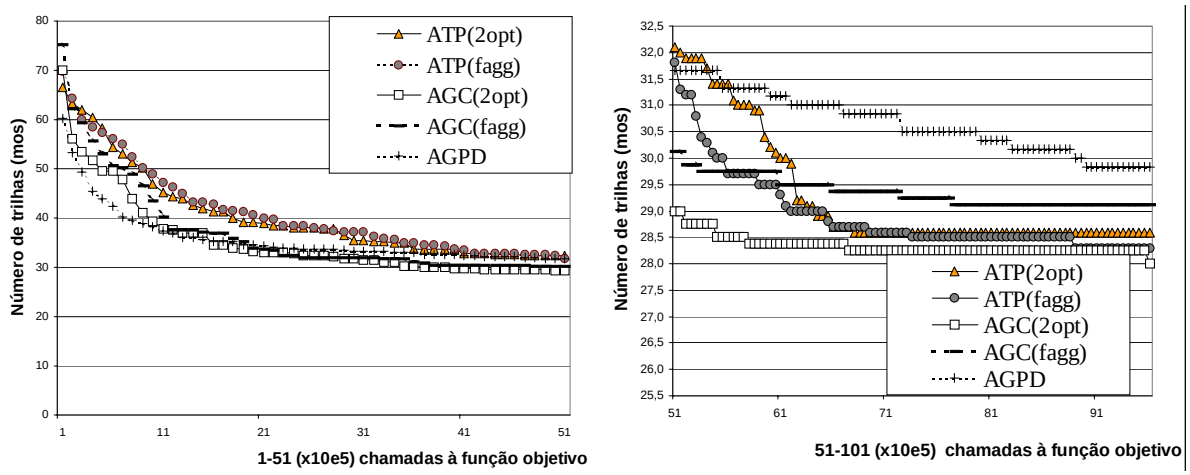


FIGURA 3.11 – Convergência para a instância *w4*.

Como pode ser observado na Figura 3.11, o AGPD se aproxima rapidamente da melhor solução conhecida (27 trilhas), mas não é capaz de alcançá-la em nenhuma das 10 execuções. A boa aproximação (melhor inclusive que os outros enfoques) se deve ao reduzido número de chamadas à função objetivo: a cada geração um único indivíduo é avaliado segundo a função objetivo.

Analisando-se as populações finais, ao término de cada execução, pode-se atribuir o mau desempenho do AGPD à perda de diversidade populacional. Apesar da população crescer dinamicamente, preservando por mais gerações um número maior de indivíduos mal-avaliados e diversificados, os indivíduos bem avaliados são representantes de soluções *muito próximas* dentro do espaço de genótipos. Com o correr das gerações, a elite da população tende a dominar o processo de seleção e fazer a população convergir prematuramente. O $\mathcal{ATP}^{fagg}$ lentamente se aproxima de 27 trilhas, alcançando uma boa média ao final, bem melhor que o $\mathcal{ATP}^{2opt}$. O $\mathcal{AGC}^{H}$, por sua vez, tem a melhor convergência, com a versão $\mathcal{AGC}^{2opt}$, e também a pior, com $\mathcal{AGC}^{fagg}$, dentre todos os enfoques.

As instâncias *p2040n6* e *p2540n3* são particularmente interessantes, pois apesar de terem poucos padrões, são relativamente difíceis. Em todos os experimentos realizados neste trabalho foram encontradas três soluções para cada uma dessas instâncias que, até o momento, são consideradas ótimas ($mpa = 8$ e $mpa = 10$, respectivamente). São elas:

p2040n6 ($mpa = 8$)																			
11	2	9	8	16	3	5	7	1	4	10	18	14	6	20	19	12	13	15	17
11	2	9	16	8	3	5	7	1	4	10	18	14	20	6	19	12	13	15	17
2	11	16	9	8	3	5	7	1	4	10	18	14	6	20	19	12	13	15	17
p2540n3 ($mpa = 10$)																			
2	14	7	23	3	15	16	24	13	4	11	8	19	18	12	21	9	1	10	$\cdots$ 25
25	22	20	5	17	6	10	9	21	12	18	19	11	4	8	13	24	16	15	$\cdots$ 14
20	5	22	25	17	6	10	9	21	12	18	19	11	4	8	13	24	2	14	$\cdots$ 3

Podem existir mais soluções para essas instâncias com mesmos valores ótimos de *mpa*. Mas essas soluções já podem ser utilizadas para se fazer um levantamento de como ficaram distribuídos na população cada um dos blocos construtivos que aparecem nelas, ao final de uma execução.

Formando-se todas as combinações possíveis de blocos B^i de diferentes tamanhos i de uma solução com I padrões e considerando-se que $ab = ba$, tem-se $I - 1$ blocos B^2, $I - 2$ blocos B^3, e assim sucessivamente até que se tenha um único bloco B^I. A partir da população final obtida nas execuções que lograram êxito, pode-se fazer uma estatística de aparição desses blocos.

As populações finais foram divididas em dois grupos: o base, formado por 20% mais bem-adaptados, e o guia, formado pelo restante da população. No processo de seleção *base-guia*, esse mesmo percentual é utilizado na escolha do indivíduo s_{base}.

A população final de uma execução bem-sucedida é uma população bem-adaptada, de maneira que o grupo guia, mesmo estando fora dos 20% mais adaptados, ainda é formado por indivíduos de qualidade. O tamanho das populações, ao término das execuções, oscilou entre 8 e 33 indivíduos. Sobre esse montante de indivíduos foi calculado o percentual de vezes em que cada um dos blocos foi encontrado em cada um dos grupos base e guia.

Para a instância *p2040n6*, 76% dos blocos de tamanho 2 foram encontrados no grupo base e, para a instância *p2540n3*, esse número caiu para 65%. Os blocos de tamanho 3 e 4 não apareceram entre os indivíduos do grupo guia, apenas no grupo base. Para se ter uma idéia dos números absolutos, para a instância *p2040n6*, o número de aparições totais dos blocos foi o seguinte: $B^2 = 112$, $B^3 = 24$, $B^4 = 14$.

Pode-se concluir que os blocos construtivos, que compõem alguma das melhores soluções, apareceram mais no grupo de indivíduos mais adaptados (base), evidenciando que o critério de adaptação prestigia os indivíduos formados por blocos de qualidade. Do contrário, seria de se supor que tais blocos deveriam aparecer uniformemente ao longo da população final, uma vez que todos eles tiveram uma participação acentuada em função

de sua posição no *ranking*. Os valores induzem a que se acredite que há uma forte relação entre a qualidade dos blocos e o critério adotado para se ordenar a população, no caso, o valor de δ.

3.5 Aplicação para problemas de *satisfabilidade* (SAT)

O enfoque construtivo foi aplicado a problemas SAT motivado principalmente por questões de codificação, tendo em vista a possibilidade de se atribuir um significado lógico para indeterminação de genes, permitindo a avaliação de esquemas diretamente pela função objetivo. Nesta seção, são apresentados os aspectos relativos a modelagem, implementação e os resultados obtidos por um $\mathcal{AGC}^H$ para problemas SAT.

3.5.1 Aspectos de modelagem

A codificação de expressões lógicas em indivíduos esquemas é bastante natural. O desenvolvimento de uma versão do $\mathcal{AGC}^H$ para problemas SAT foi motivado pelo fato de uma solução incompleta poder ser avaliada como *verdadeiro* ou *falso*. Por exemplo, uma cláusula $(.V. \vee .F. \vee \#)$, apesar da presença de um símbolo "#", pode ser avaliada como verdadeira. Isso garante a existência de soluções viáveis mesmo que incompletas.

Uma instância SAT é dada por uma expressão lógica $\mathcal{E}$. Esta aplicação se atém somente a instâncias na Forma Normal Conjuntiva (CNF). Um indivíduo s_k representa uma atribuição de valores às variáveis que aparecem em $\mathcal{E}$. Em se tratando de um esquema, apenas parte das variáveis recebem valores. Dessa forma, cada gene codifica uma variável da expressão lógica que pode assumir os valores $.V.$, $.F.$ ou nenhum dos dois (#).

O $\mathcal{AGC}^H$ tratou o SAT como um problema de maximização, onde a função de aptidão f é o número de cláusulas satisfeitas uma dada atribuição representada por s_k. Uma solução ótima para uma instância satisfazível é encontrada quando o número de cláusulas satisfeitas é igual ao número de cláusulas total da expressão. Seja a expressão:

$$\mathcal{E}_1 = (x_1 \vee x_5 \vee \overline{x_6}) \wedge (\overline{x_2} \vee \overline{x_3}) \wedge (\overline{x_1} \vee \overline{x_5}) \wedge (x_4 \vee x_3 \vee \overline{x_2} \vee \overline{x_1})$$

a estrutura $s_1 = \{V, F, F, F, V, F\}$ é uma atribuição válida para $\mathcal{E}_1$, resultando em $f(s_1) = 3$. A terceira cláusula não é satisfeita com s_1. Pode-se generalizar a função de aptidão f

como sendo:

$$f(s_k) = \sum_{i=1}^{|\mathcal{E}|} \left\{ s_{k,a_{i,1}} \vee_{j=2}^{|\mathcal{E}_i|} s_{k,a_{i,j}} \right\} \tag{3.23}$$

onde $|\mathcal{E}|$ é o número de cláusulas na expressão, $|\mathcal{E}_i|$ é o número de variáveis na $i-$ésima cláusula $\mathcal{E}_i$, e a_{ij} é o índice do $j-$ésimo literal na $i-$ésima cláusula que corresponde a $z-$ésima variável de s_k.

Em se tratando de um esquema, o símbolo "#" é avaliado dentro de uma cláusula conforme a Tabela 3.8. Se uma dada cláusula for avaliada como "#", ou seja, valor lógico indeterminado, ela não entra no somatório de cláusulas satisfeitas.

TABELA 3.8 – Tabela-verdade para operações lógicas com indeterminação.

X	Y	$X \vee Y$
0	#	#
1	#	1
#	#	#

A função g, por sua vez, avalia a vizinhança do indivíduo através de heurísticas específicas para problemas SAT. Um outro ponto que tem motivado esta aplicação é o número heurísticas desenvolvidas para o SAT.

3.5.2 Heurísticas relacionadas

Existem várias heurísticas ou algoritmos aproximativos para problemas SAT. Um esboço de uma heurística pode ser dado por:

```
while houver melhoria do
    Escolha uma variável segundo uma estratégia;
    troque o valor dessa variável;
end while
```

A diferença entre uma ou outra abordagem se resume basicamente na estratégia de escolha da variável a ter seu valor trocado (comutado). Uma estratégia primitiva seria escolher uma variável aleatoriamente, desde que a escolha satisfaça mais cláusulas. Estratégias gulosas realizam o seguinte cálculo: para cada cláusula não-satisfeita escolha a

variável que, uma vez *comutada*[2], aumentar o número de cláusulas satisfeitas, *quebrando* o mínimo de outras. O termo *quebrar* é comumente utilizado em se tratando de cláusulas que se tornem não-satisfeita depois de algum movimento heurístico (Hoos e Stutzle, 2000).

Uma vez que não haja mais melhorias no número de cláusulas satisfeitas, métodos puramente baseados em *subida da encosta* param. Para conferir um pouco mais de robustez à heurística de busca, pode-se permitir movimentos que não necessariamente melhorem a solução até então encontrada.

Em termos de desempenho, as heurísticas para SAT comumente são avaliadas com relação ao número médio de comutações necessárias para alcançar soluções ótimas. Diferentemente de outros problemas, nos quais o número de avaliações de função objetivo é mais focalizado. Como uma atribuição SAT pode ser avaliada sempre linearmente, toda a competência do método reside no número de comutações (*flips*) que ele costuma fazer para achar uma solução.

A descrição de várias heurísticas pode ser encontrado em (Hoos e Stutzle, 2000; Gu et al., 1997; Schuurmans e Southey, 2001). Segue um resumo das estratégias de algumas delas:

a) GSAT: comuta a variável que resultar num menor número de cláusulas não-satisfeitas. Em caso de quebra, é escolhida uma variável aleatoriamente;

b) HSAT: similar a GSAT, exceto nos casos de quebra, os quais são decididos em favor da variável menos recentemente comutada;

c) Novelty: seleciona uma cláusula aleatoriamente, comuta a variável que resultaria em um menor número total de outras cláusulas não-satisfeitas a menos que a variável seja a mais recentemente comutada na cláusula escolhida.

Para efeito de avaliação g, foi utilizada a heurística proposta em (Gent, 1998). Sejam $Sinal(\mathcal{E}, s_{k,z})^+$ e $Sinal(\mathcal{E}, s_{k,z})^-$ as notações que dizem se a $z-$ésima variável de s_k aparece mais vezes *afirmada* e *negada* em $\mathcal{E}$, respectivamente. O procedimento baseado na heurística de *Gent* é dado por:

[2]Comutar (ou chavear) é trocar o valor lógico de uma variável.

```
Procedimento Gent(E, s_k)
s_V := s_k
ini := rand(1, n − l)
for z := ini até ini + l do
  if s_{k,z} = .F. e Sinal(E, s_{k,z})^+ then
    s_{k,z} := .V.;
    Guarda f(s_k) em g(s_k), se for melhor;
  end if
  if s_{k,z} = .V. e Sinal(E, s_{k,z})^− then
    s_{k,j} := .F.;
    Guarda f(s_k) em g(s_k), se for melhor;
  end if
end for
```

A heurística considera que variáveis com valor $.F.$ e que aparecem mais vezes positivamente na expressão podem ser comutadas, o que provavelmente melhorará a avaliação da expressão, e vice-versa.

O procedimento heurístico considera apenas uma parte l das variáveis que compõem o indivíduo, ou seja, a vizinhança foi limitada da mesma forma que em outras aplicações do $\mathcal{AGC}^H$. Essa heurística não garante que a solução ótima seja encontrada (Gent, 1998). Em virtude disso, outra heurística baseada no WalkSAT (Selman et al., 1994) foi utilizada como busca local e é descrita posteriormente.

3.5.3 Aspectos de implementação

A população inicial é composta exclusivamente de esquemas, i.e., parte das variáveis recebe um rótulo "#". O método de seleção, como nas implementações anteriores do $\mathcal{AGC}$, é o *base-guia*.

Uma recombinação clausular, proposta neste trabalho, recombina blocos de qualidade encontrados nos indivíduos s_{base} e s_{guia}, previamente selecionados. Inicialmente, s_{base} é copiado para s_{novo}. As variáveis das cláusulas não-satisfeitas em s_{base} são substituídas pelas variáveis de cláusulas satisfeitas de s_{guia}. Ao final, as variáveis que aparecem em cláusulas não-satisfeitas de s_{base} e s_{guia} que ainda não foram copiadas são escolhidas aleatoriamente, finalizando a composição de s_{novo}. Uma vez existindo variáveis comuns a várias cláusulas, uma cláusula satisfeita copiada para s_{novo} pode quebrar outras previamente copiadas, a recombinação clausular não necessariamente gera indivíduos melhores.

Um operador de busca local baseado na metaheurística WalkSAT (Selman et al., 1994) foi implementado neste trabalho. O WalkSAT incorpora movimentos aleatórios quando não consegue incrementar o número de cláusulas satisfeitas. A estratégia de escolha da variável a ser comutada é como se segue. Se a variável puder ser comutada sem quebrar nenhuma outra cláusula, então é feito. Caso contrário, ou é escolhida a variável que causar um menor número de quebras, ou é escolhida uma variável qualquer, com probabilidade uniforme. Esses movimentos são repetidos até que nenhuma cláusula seja satisfeita. No algoritmo original, existe um mecanismo de reinício que não foi incorporado na implementação deste operador, chamado de *MutaçãoWS*. Além disso, apenas um número máximo de comutações é permitido.

O algoritmo empregado pelo $\mathcal{AGC}^H$ é similar ao implementado para problemas de seqüenciamento de padrões. Entretanto, como já foi mencionado, a implementação para o SAT dispensa a necessidade de complementar esquemas, uma vez que rótulos "#" podem ser descartados sem prejuízo para o processo de avaliação.

3.5.4 Resultados computacionais

O $\mathcal{AGC}^H$ foi codificado em *C ANSI* e testado em aproximadamente 60 instâncias SAT *satisfazíveis*[3]. Os nomes das instâncias em língua inglesa estão preservados neste texto. Os experimentos foram conduzidos em uma máquina *Intel Pentium II 266MHz e memória 64MB* e os resultados do $\mathcal{AGC}^H$ foram comparados com outros dois enfoques: um enfoque evolutivo a chamado de *ASAP* (*Adaptive evolutionary algorithm for the SAtisfiability Problem*) (Rossi et al., 2000) e o procedimento WalkSAT (Selman et al., 1994)[4].

A motivação para inclusão do WalkSAT nestes experimentos, embora ele não seja um enfoque evolutivo, vem de sua popularidade. O WalkSAT é comumente usado como referência para comparações. Com relação ao *ASAP*, recentemente, cinco enfoques evolutivos, juntamente com o WalkSAT, foram comparados em experimentos envolvendo 662 instâncias SAT, algumas delas com até 100 variáveis (Gottlieb et al., 2002). O mais recente dos métodos incluídos nessa comparação foi o *ASAP*.

O *ASAP* é um algoritmo baseado em população que utiliza heurísticas específicas do SAT. Uma tabela é dinamicamente preenchida com indivíduos de melhores *fitness*. Quando é encontrado um *fitness* melhor essa tabela é esvaziada. Quando a tabela fica cheia, os indivíduos são comparados em nível de gene. Aqueles que não têm o mesmo valor em todos os indivíduos são rotulados como *congelados*. Os genes *congelados*

[3]Disponíveis em `http://www.intellektik.informatik.tu-darmstadt.de/SATLIB/`
[4]Disponível em `http://www.cs.washington.edu/homes/kautz/walksat/`

não participam da mutação adaptativa, nem de uma heurística comutadora, ambas responsáveis pela evolução da população. Segundo seus autores, o *ASAP* escapa de mínimos locais adaptando a probabilidade de mutação de cada gene *não-congelado* (Rossi et al., 2000)

Os experimentos objetivaram, inicialmente, testar a robustez do $\mathcal{AGC}^H$. A taxa de acerto foi medida em várias simulações no experimento 1. Os três experimentos subseqüentes objetivaram compará-lo com o *ASAP* e o WalkSAT. Nas tabelas mostradas a seguir, n é o número total de variáveis e $|\mathcal{E}|$ se refere ao número de cláusulas da instância.

A Tabela 3.9 mostra o desempenho do $\mathcal{AGC}^H$ no experimento 1 (instâncias *Uniform Random-3-SAT*). O número de tentativas para cada instância variou conforme o tempo médio necessário para cada execução. Um mínimo de 12 tentativas foram executadas, chegando, em alguns casos, até a 1000 (instâncias menores). A taxa de acerto (%) e a média do número de cláusulas satisfeitas encontradas são mostrados na tabela. O tempo médio foi computado em segundos (s).

TABELA 3.9 – Desempenho do $\mathcal{AGC}^H$ no experimento 1.

Instância	n	$\|\mathcal{E}\|$	tentativas	satisfeitas	tempo (s)	acerto (%)
uf20	20	91	1000	91,00	0,01	100,00
uf50	50	218	112	218,00	0,06	100,00
uf75	75	325	100	325,00	0,54	100,00
uf125	125	538	93	537,97	16,20	97,85
uf150	150	645	12	644,91	36,33	91,67
uf175	175	753	12	753,00	48,42	100,00
uf200	200	860	55	859,76	174,83	76,36
uf225	225	960	12	959,66	217,25	66,67
uf250	250	1065	12	1064,58	279,50	58,33

A Tabela 3.10 mostra o desempenho do $\mathcal{AGC}^H$ comparado ao *ASAP* e WalkSAT no experimento 2 (*aim*:*Artificially generated Random-3-SAT*). Existe uma peculiaridade com relação as instâncias *aim*. Algumas heurísticas de busca, como WalkSAT e variantes da família GSAT, mostram desempenho fraco com relação a elas.

Os experimentos foram conduzidos usando-se o $\mathcal{AGC}^H$ e o WalkSAT. Sendo assim, em todos foram medidos o número de comutações, tempo de execução e taxas de acerto para cada instância. Nos resultados do *ASAP*, não há informação sobre a taxa de acerto para esta classe de instâncias (Rossi et al., 2000). Seus autores mostram somente se a solução foi encontrada ou não. Entretanto, há um relato sobre a falha em encontrar uma solução

TABELA 3.10 – Comparação entre $\mathcal{AGC}^H$, *ASAP* e WalkSAT no experimento 2.

			tempo médio (s)			acerto (%)	
Instância	n	$\|\mathcal{E}\|$	$\mathcal{AGC}^H$	WalkSAT	*ASAP*	$\mathcal{AGC}^H$	WalkSAT
aim-50-1-6-yes1	50	80	49,50	40,00	-	**50,00**	25,00
aim-50-2-0-yes1	50	100	17,00	38,75	18,76	**100,00**	50,00
aim-50-3-4-yes1	50	170	8,25	0,00	0,05	100,00	100,00
aim-50-6-0-yes1	50	300	45,50	0,00	0,01	100,00	100,00
aim-100-1-6-yes1	100	160	44,00	50,50	-	0,00	0,00
aim-100-2-0-yes1	100	200	36,50	50,25	-	**50,00**	0,00
aim-100-3-4-yes1	100	340	13,75	0,25	0,33	100,00	100,00
aim-100-6-0-yes1	100	600	86,50	0,00	0,02	100,00	100,00
aim-200-1-6-yes1	200	320	91,25	52,00	-	0,00	0,00
aim-200-2-0-yes1	200	400	62,75	57,00	-	0,00	0,00
aim-200-3-4-yes1	200	680	106,50	0,75	20,70	75,00	**100,00**
aim-200-6-0-yes1	200	1200	274,75	0,00	0,11	100,00	100,00

para a instância *aim-100-2-0-yes1*. Além disso, o *ASAP* não foi testado em todas as instâncias *aim* (Rossi et al., 2000) (símbolos "-" na tabela).

O $\mathcal{AGC}^H$ e o WalkSAT foram executados 10 vezes para cada instância no experimento 2. O *ASAP* também foi codificado em C, mas executado em uma máquina *Intel Pentium II 350MHz e memória 64MB* (Rossi et al., 2000). Apesar de plataformas diferentes, os tempos medidos podem dar uma noção sobre a velocidade do algoritmo. O *ASAP* obteve tempos médios de execução similares ao WalkSAT e melhores que o $\mathcal{AGC}^H$ nas instâncias em que os três foram executados.

Nas instâncias *aim-50-1-6-yes1*, *aim-50-2-0-yes1* e *aim-100-2-0-yes1*, para as quais o WalkSAT não consegue bons resultados, o $\mathcal{AGC}^H$ obteve tempos similares ao WalkSAT e melhores taxas de acerto. No cômpito geral, em termos de taxa de acerto, o $\mathcal{AGC}^H$ e o WalkSAT apresentaram resultados semelhantes, com uma pequena vantagem para o $\mathcal{AGC}^H$ que obteve algo em torno de $64,58\%$ de acerto para as instâncias *aim* contra $56,25\%$ do WalkSAT. Analisando caso a caso, o $\mathcal{AGC}^H$ obteve melhor taxa de acerto em três instâncias contra uma do WalkSAT.

Um outro aspecto relevante de desempenho é o número de comutações (*flips*) necessárias para se encontrar uma solução. As comutações foram medidas nas 10 tentativas dos experimentos 3 e 4, além da média do tempo de execução. Apenas os enfoques evolutivos foram considerados neste experimento. As Tabelas 3.11 e 3.12 mostram a comparação entre o $\mathcal{AGC}^H$ e o *ASAP* com relação ao número de *flips*. De acordo com (Rossi et al., 2000), somente foram computados os *flips* que foram aceitos, isto é, *flips* que melhoram a solução. O $\mathcal{AGC}^H$, por outro lado, computou todos os *flips* e por essa razão ele parece

pior que o *ASAP*. Mas em algumas instâncias, pode-se observar que a diferença entre as médias dos tempos de execução não coadunam com a diferença no número de *flips*.

Desconsiderando as diferenças de plataformas de execução, a média total dos tempos de execução da Tabela 3.11 mostra uma pequena vantagem para o $\mathcal{AGC}^H$: $8s$ contra $11,025s$ do *ASAP*. Neste experimento, o $\mathcal{AGC}^H$ obteve melhor média de tempo de execução em 3 das 13 instâncias comparadas (em negrito).

TABELA 3.11 – Comparação entre $\mathcal{AGC}^H$ e *ASAP* no experimento 3.

			$\mathcal{AGC}^H$		ASAP	
Instância	n	$\lvert\mathcal{E}\rvert$	média *flips*	tempo(s)	tempo(s)	*flips* aceitos
jnh201	100	800	9050	**0,01**	0,42	2458
jnh204	100	800	8575840	8,00	6,49	36046
jnh205	100	800	3808117	5,00	3,59	20037
jnh209	100	800	7857142	8,00	6,22	35115
jnh210	100	800	20937	1,00	0,39	2288
jnh212	100	800	36414924	**18,00**	78,00	429970
jnh213	100	800	7881500	8,00	0,85	4890
jnh217	100	800	342009	2,00	0,56	3219
jnh218	100	800	577410	1,00	0,66	3820
jnh1	100	850	5050725	**7,00**	16,32	83459
jnh12	100	850	1779953	4,00	2,57	12669
jnh17	100	850	1105,532	3,00	1,67	8477
jnh301	100	900	98153693	39,00	25,58	115059

Na Tabela 3.12, entretanto, considerando-se as instâncias testadas pelo *ASAP*, os resultados mudam: a média total de tempo do $\mathcal{AGC}^H$ ficou em torno de $18,20$ contra $15,03s$ do *ASAP*. Apesar da comparação desfavorável, $\mathcal{AGC}^H$ executou em menor tempo em 8 das 14 instâncias comparadas neste experimento (em negrito).

3.6 Aplicação para minimização de funções numéricas sem restrição

O *Algoritmo de Treinamento Populacional* ($\mathcal{ATP}$) foi aplicado para problemas de minimização numérica sem restrição motivado principalmente pela possibilidade de utilizar o enfoque não-construtivo do TPH em uma classe de problemas na qual a codificação em real facilitaria a utilização de heurísticas de busca local, mas impossibilitaria a modelagem com esquemas (Oliveira e Lorena, 2002c).

TABELA 3.12 – Comparação entre $\mathcal{AGC}^H$ e *ASAP* no experimento 4.

			$\mathcal{AGC}^H$		ASAP	
Instância	n	$\|\mathcal{E}\|$	média *flips*	tempo (s)	tempo (s)	*flips* aceitos
ii8a1	66	186	6951	**0,001**	0,009	767
ii08a2	180	800	11098	**0,001**	0,010	494
ii08a3	264	1552	8101	0,100	0,043	1407
ii08a4	396	2798	41656	**0,100**	0,187	4807
ii08b1	336	2068	3728	**0,001**	0,014	759
ii08b2	576	4088	548253	**0,400**	0,460	20590
ii08b3	816	6108	3974260	1,700	0,996	44949
ii08c1	510	3065	56488	**0,001**	0,019	881
ii08c2	950	6689	80566	**0,100**	0,186	7806
ii08d1	530	3207	61378	**0,001**	0,153	6192
ii08d2	930	6547	96822	0,200	0,198	7707
ii08e1	520	3136	17704	0,100	0,050	1886
ii08e2	870	6121	602892	0,500	0,214	7842
ii32b1	228	1374	580100	1,600	-	-
ii32b2	261	2558	56630934	23,800	-	-
ii32b3	348	5734	472971189	128,700	-	-
ii32b4	381	6918	1199971920	251,600	207,930	312122
ii32c1	225	1280	4926	0,200	-	-
ii32c2	249	2182	33650	0,800	-	-
ii32c3	279	3272	7284390	13,000	-	-
ii32d1	332	2703	726440	1,800	-	-
ii32d2	404	5153	271023840	86,700	-	-
ii32e1	222	1186	8163	0,200	-	-
ii32e2	267	2746	18660	0,500	-	-
ii32e3	330	5020	5522740	16,000	-	-
ii32e4	387	7106	94661230	69,300	-	-

3.6.1 Modelagem não-construtiva e implementação

A principal razão para a existência do enfoque não-construtivo reside na possibilidade de utilização dos conceitos do TPH em problemas que não são facilmente modelados para trabalharem com esquemas. Além disso, como foi visto em problemas de seqüenciamento, nem sempre se consegue bons resultados trabalhando com vizinhanças geradas por heurísticas a partir de soluções incompletas.

Para a minimização das funções-teste, foi utilizada uma codificação de indivíduos usando valores reais. A função de aptidão f retrata o valor da função objetivo, enquanto a função g avalia a vizinhança, segundo uma heurística de treinamento específica para esse tipo de problema/codificação.

O $\mathcal{ATP}$ utilizou algoritmo similar à versão desenvolvida para problemas de seqüencia-

mento de padrões. Foi empregada a seleção *base-guia*, o cruzamento BLX *blend crossover* (Eshelman e Schaffer, 1993), e a mutação não-uniforme (Michalewicz, 1996). Os dois últimos são operadores evolutivos populares e de desempenho considerado satisfatório (Herrera et al., 1998).

3.6.2 Heurísticas de treinamento

Inicialmente, foram testadas heurísticas de busca local, como o Simplex de Nelder e Mead (SNM) (Nelder e Mead, 1965) e o método gradiente por diferenças finitas. Estes métodos possuem um custo computacional significativo, pois sua complexidade é função do número de dimensões do problema. O SNM, por exemplo, possui complexidade $O(n^{2.11})$, além da conhecida necessidade de armazenamento de n vetores para cada um dos vértices do simplex (Nelder e Mead, 1965). O método gradiente ainda possui uma outra desvantagem, como a possibilidade se tornar inviável para ser aplicado em funções com espaços de busca descontínuos.

Procurando uma heurística de treinamento computacionalmente mais leve, foi implementada uma heurística de melhoria chamada de busca *em-Linha*. Ela inspeciona uma vizinhança de poucos pontos uniformemente espalhados entre o ponto em questão e um ponto aleatório escolhido dentre o conjunto dos indivíduos *base*. Dessa forma, o grupo de elite da população, em termos de *ranking*, é usado como referência para estabelecer uma vizinhança para os demais indivíduos da população.

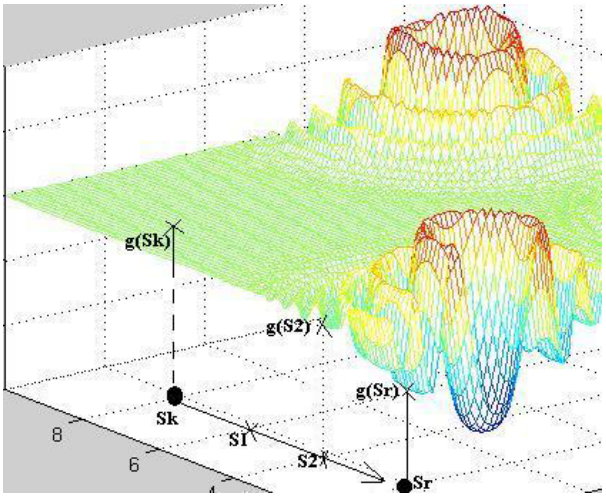


FIGURA 3.12 – Exemplo bidimensional do procedimento de busca *em-Linha*.

A Figura 3.12 mostra um exemplo da busca *em-Linha* para o caso bidimensional. Os

pontos s_k e s_r são respectivamente o indivíduo a ser avaliado e o indivíduo referência, escolhido aleatoriamente dentro do conjunto de indivíduos *base*. A linha entre s_k e s_r é dividida em intervalos iguais e pontos dentro desse intervalo são avaliados diretamente pela função objetivo.

Enquanto forem encontradas soluções melhores, pontos na mesma linha, mas fora desse intervalo, também são avaliados. Dessa forma, são consideradas soluções de uma da mesma vizinhança *em-Linha*, um número fixo delas entre s_k e s_r e mais um número variável após s_r (desde que sejam soluções de qualidade). Pelo exemplo da Figura 3.12, o ponto melhor avaliado pela heurística é o ponto s_2. Essa avaliação é usada como valor de g.

Apesar de não ter sido inicialmente relacionada ao *path-relinking* (*PR*) (Glover, 1998), a busca *em-Linha* trabalha com os mesmos princípios. A trajetória entre duas soluções é explorada e uma delas faz parte de um conjunto de soluções elite. Neste contexto, são considerados a elite da população 20% (base) dos indivíduos melhor adaptados. O *PR* é discutido com mais detalhes no Capítulo 4.

Uma vez computadas as funções f e g, o δ_{ncons} é calculado segundo a equação 3.13, aqui repetida:

$$\delta_{ncons}(s_k) = d \cdot [G_{max} - g(s_k)] - |f(s_k) - g(s_k)|$$

Na Figura 3.13, os efeitos da avaliação dos intervalos $|f(s_k) - g(s_k)|$ (I1) e $d \cdot [G_{max} - g(s_k)]$ (I2) são mostrados separadamente. Indivíduos com valores altos para I2 estão posicionados nas regiões mais promissoras (Figura 3.13a), enquanto aqueles com valores mínimos para I1 ($f - g \simeq 0$) estão posicionados em *platôs* ou regiões que são prováveis mínimos locais (Figura 3.13b). A população inteira preenche uma significante parte do espaço de busca. As regiões de busca não ocupadas por indivíduo algum são aquelas em que o valor de g se aproxima de G_{max} e indivíduos inicialmente gerados nessas regiões são progressivamente eliminados pelo limiar de rejeição com incrementos adaptativos, α.

A população dinâmica permite uma boa ocupação dos subespaços de busca considerados promissores. O TPH acrescenta informação relativa a vizinhança gerada pela heurística *em-Linha* no processo de avaliação de indivíduos.

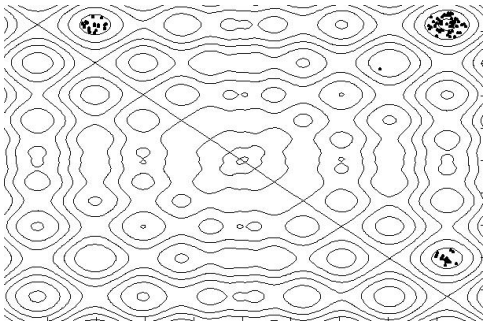
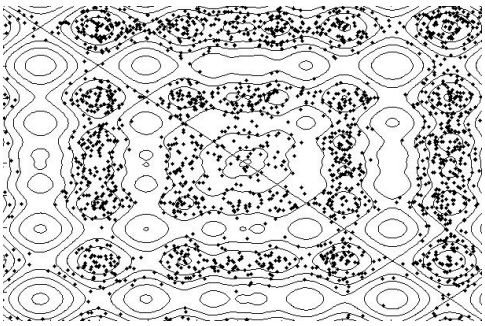
a)  b) 

FIGURA 3.13 – Indivíduos no mapa de contorno da função *Schwefel* bidimensional: a)indivíduos que maximizam I2 e b)indivíduos que minimizam I1.

3.6.3 Resultados computacionais

Os experimentos computacionais foram realizados em duas etapas. Uma etapa para observar o comportamento do $\mathcal{ATP}$ no espaço bidimensional de algumas funções multimodais. A outra etapa para aferir o desempenho do algoritmo, comparando-o com um outro algoritmo genético.

Para comparação, foi implementado um Algoritmo Genético Codificado em Real (AGCR), sem busca local e apresentando ainda as seguintes diferenças em relação ao $\mathcal{ATP}$: população fixa com *fitness* baseado simplesmente em função objetivo e atualização de população pelo método *steady-state* (ou não-geracional).

Na atualização *steady-state*, os filhos recém-gerados entram diretamente na população, passando a competir com seus próprios pais. Como a população é fixa, o novo indivíduo deve ocupar o local antes ocupado por um indivíduo pior. É necessário, então, que haja um procedimento para se encontrar um novo indivíduo pior a cada geração para ser o alvo da próxima substituição.

Foi observado nos experimentos com funções bidimensionais, que a população no $\mathcal{ATP}$ cresce sempre enquanto estiverem sendo gerados novos indivíduos com bons *rankings*. Evidentemente, o limiar de rejeição adaptativo é usado para controlar a população, eliminando os indivíduos mal-adaptados.

À medida em que vão sendo gerados indivíduos em regiões promissoras, este recebem melhores *rankings*. Existe uma tendência para que a população pare de crescer e, a medida em que os piores indivíduos vão sendo eliminados, a população se esvazie ou permaneça apenas em torno dos mínimos locais.

Esse comportamento pode ser observado na seqüência de amostras visualizadas na Figura

3.14. Os gráficos mostram o mapa de contorno da função *Langerman* bidimensional e as regiões do espaço de busca ocupadas por indivíduos, após 100, 500, 1000 e 5000 gerações, respectivamente.

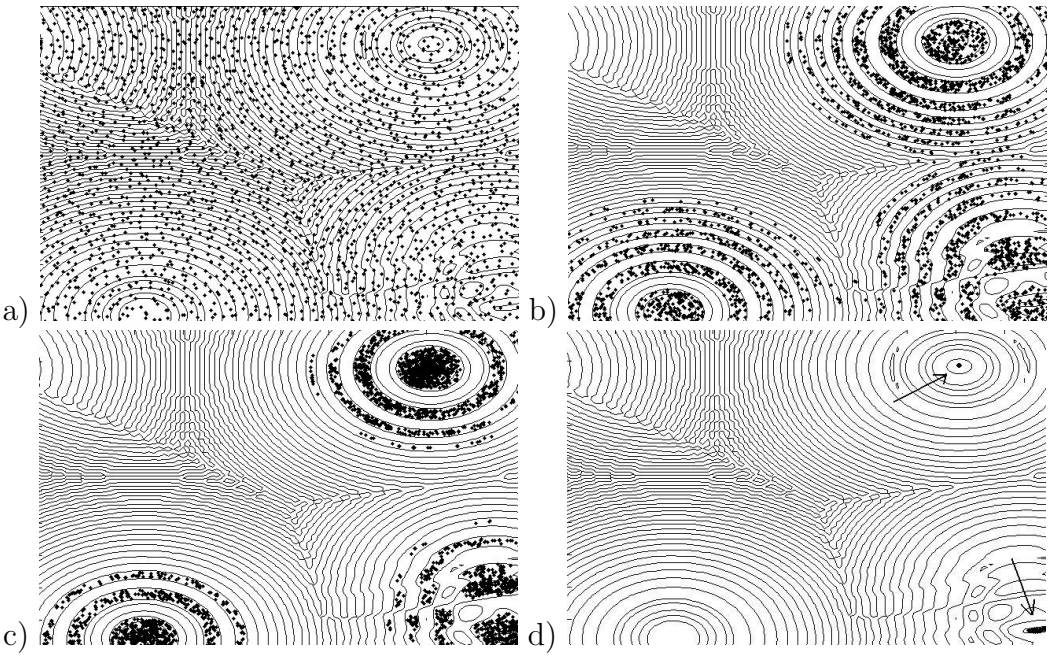


FIGURA 3.14 – Indivíduos no mapa de contorno da função de *Langerman* bidimensional após a)100, b)500, c)1000 e d)5000 gerações.

Observa-se que no último quadro, apenas dois grupos de indivíduos se mantêm na população. Um grupo em torno do mínimo global (canto inferior direito) e o outro em um ponto de mínimo local (canto superior direito). A outra região promissora foi descartada, pela eliminação dos indivíduos em torno dela.

Os desempenhos do AGCR e $\mathcal{ATP}$ foram medidos através do erro entre a média das melhores soluções encontradas e as soluções ótimas esperadas. As funções utilizadas possuem mínimos conhecidos. A Tabela 3.13 mostra um sumário dos resultados (Oliveira e Lorena, 2002c).

Como pode ser observado na Tabela 3.13, em 5 dimensões, o $\mathcal{ATP}$ apresentou um melhor desempenho global, mesmo sem considerar o mal desempenho do AGCR na função de *Rosenbrock*. Em 10 dimensões, mais uma vez o AGCR teve um mal desempenho na função de *Rosenbrock* (melhor solução encontrada foi 6,10). Excluindo o resultado da função de *Rosenbrock* dos resultados de ambos, existe um equilíbrio de desempenho: $\mathcal{ATP}$ obteve 23,95% de erro, contra 26,75% obtido pelo AGCR (Oliveira e Lorena,

TABELA 3.13 – Comparação entre $\mathcal{ATP}$ e AGCR para seis funções-teste, com $n = 5$ e $n = 10$.

	gap		CFO	
Função(n)	$\mathcal{ATP}$	AGCR	$\mathcal{ATP}$	AGCR
Rosembrock(5)	**0,011**	0,35	486024,5	534000,0
Rastrigin(5)	0	0	178334,3	98992,7
Griewank(5)	0,008	**0,005**	304628,8	540362,4
Langerman(5)	**0,437**	1,042	103839,0	751024,0
Michalewicz(5)	0,005	**0**	83140,8	92533,8
Schwefel(5)	0,243	**0,006**	359142,7	404562,2
Rosembrock(10)	**0,101**	6,639	703015,5	751000,0
Rastrigin(10)	0,006	**0**	142098,4	243424,2
Griewank(10)	0,008	0,008	353668,3	574628,8
Langerman(10)	**1,396**	1,397	422314,4	751000,0
Michalewicz(10)	**0,105**	1,967	691504,1	751000,0
Schwefel(10)	254,387	**236,877**	368686,5	544419,2

2002c). Analisando caso a caso, cada um dos algoritmos obteve melhor solução em cinco funções-teste (soluções em negrito).

O número de chamadas à função objetivo, tanto do AGCR quanto do $\mathcal{ATP}$, é considerado alto. Como pode ser examinado no Capítulo 4, resultados de outros enfoques foram bem melhores que os obtidos pelo $\mathcal{ATP}$.

3.7 Considerações finais

O Treinamento Populacional em Heurísticas (TPH) tem raízes sedimentadas na dupla avaliação de indivíduos proposta por (Lorena e Furtado, 2001) e consiste basicamente na avaliação de indivíduos através de heurísticas específicas. O indivíduo (esquema ou estrutura) é bem avaliado quando não pode ser melhorado pela heurística de treinamento. Esse mecanismo alternativo permite que outras características do indivíduo sejam consideradas na composição de sua aptidão. Em outras palavras, há uma certa flexibilidade na avaliação do indivíduo que permite aferir heuristicamente o aprendizado por ele adquirido, ao longo das gerações, acerca do problema em questão.

Nesse aspecto, a principal contribuição deste trabalho é disponibilizar um arcabouço que pode ser usado para nortear futuras aplicações envolvendo dupla avaliação de indivíduos utilizando heurísticas. As questões relativas ao desempenho do método são tratadas a seguir.

3.7.1 Compromisso entre tamanho de vizinhança e custo computacional

Uma aplicação de TPH requer a existência de um algoritmo heurístico capaz de gerar um conjunto de finito de soluções a partir de qualquer $s_k \in \mathsf{S}$. É desejável que a heurística empregada tenha as seguintes propriedades:

a) configurável para um número máximo de passos, o que define a *precisão máxima* do procedimento;

b) capacidade de encontrar a solução ótima;

c) baixo custo computacional.

O número de soluções vizinhas que podem ser geradas por φ^H está sujeito à característica intrínseca de H e à *precisão máxima* l, estabelecida como parâmetro. Idealmente, l deve ser ajustado de modo a não sobrecarregar a avaliação da população.

Deve-se observar também questões sobre a paisagem de aptidão gerada pela heurística de treinamento. O espaço de busca codificado equivale a um grafo, onde cada solução candidata forma um vértice. A heurística de treinamento deve ser capaz de encontrar o vértice x^* onde a função objetivo assume o valor procurado (mínimo ou máximo). Em outras palavras, deve existir uma vizinhança φ^H de tamanho $l \geq 2/x^* \in \varphi^H$. O valor 2 é uma restrição que obriga que haja ao menos uma solução não ótima na vizinhança da solução ótima x^*.

Naturalmente, essa condição é desejável, mas não imprescindível, pois existem outros mecanismos geradores de movimentos inter-vértices que fazem parte do conjunto de operadores evolutivos Θ. Todo o espaço de busca deve estar conectado de modo que haja sobreposição de vizinhanças. A intercomunicação entre vizinhanças pode ser escrita como:

$$\exists s_k / s^* \in \varphi^H(s_k), \exists s_{k-1} / s_k \in \varphi^H(s_{k-1}), \quad \cdots, \exists s_0 / s_1 \in \varphi^H(s_0), \quad \{s^*, s_k, s_{k-1}, \cdots, s_1, s_0\} \in \mathsf{S}$$

Essas colocações sugerem que a pesquisa em TPH seja enriquecida com estudos sobre as paisagens de aptidão que são induzidas pelos operadores evolutivos, heurísticas de treinamento, procedimentos de busca local. Estudos que possam indicar quais associações de operadores são mais indicadas para uma ou outra codificação.

3.7.2 Múltiplas heurísticas de treinamento

Adicionar conhecimento heurístico a um modelo evolutivo abre algumas perspectivas pouco ou ainda não exploradas na literatura, repleta de heurísticas desenvolvidas para casos gerais e específicos de problemas de otimização. Muitas dessas heurísticas exploram características diferentes do mesmo problema. Desenvolver um algoritmo de treinamento populacional passa obrigatoriamente pela definição da heurística de treinamento a ser empregada. Melhor ainda, de forma mais abrangente, quantas heurísticas a serem utilizadas. Nesse caso , a população de indivíduos P seria definida pelo par:

$$P = \{(s_k, h_j)\}, s_k \in \mathsf{S}, h_j \in H \tag{3.25}$$

onde H é o conjunto de heurísticas de treinamento.

No treinamento de uma população por uma única heurística, toda a população é avaliada da mesma forma, ou seja, todos os indivíduos têm probabilidade de participar do processo evolutivo proporcional a essa avaliação comum. Assim, a única variável envolvida na avaliação são os seus próprios genótipos.

Por outro lado, diferentes heurísticas estabelecem diferentes vizinhanças e, conseqüentemente, diferentes visões do mesmo espaço de busca. Múltiplas heurísticas também influenciam a avaliação do indivíduo. Heurísticas pouco eficientes tendem a privilegiar indivíduos bem-treinados em detrimento de outros, cuja eficiência de suas heurísticas acabaria por penalizá-los.

O enfoque multi-heurístico do treinamento populacional pode empregar várias heurísticas cooperando ou competindo entre si através dos indivíduos associados a elas. Basicamente, tem-se os seguintes modelos:

a) uma única população treinada com diferentes heurísticas, competitivamente;

b) uma única população treinada com diferentes heurísticas, cooperativamente;

c) múltiplas populações treinadas com diferentes heurísticas, paralelamente;

No primeiro enfoque, competição entre heurísticas significa competição entre indivíduos, na mesma população, treinados com diferentes heurísticas. A possibilidade de serem empregadas mais de uma heurística de treinamento competitivamente, equivale, a grosso modo, a serem empregadas mais de uma função objetivo para um problema mono-objetivo.

A cooperação, no segundo enfoque, tem o significado de se considerar, para efeito de avaliação de cada indivíduo, a informação provida por cada uma das heurísticas, atribuindo-se diferentes graus de relevância a cada informação. Assim, pesos podem ser usados para equilibrar a participação de todos indivíduos no processo evolutivo. Isso permitiria manter as diferentes visões do espaço de busca dentro da população ao longo das gerações, de forma um pouco mais independente das heurísticas de cada indivíduo.

Pode-se ainda separar logicamente indivíduos em diferentes subpopulações, treinadas com diferentes heurísticas, preservando-os de uma competição injusta. Neste caso, é esperado que cada subpopulação convirja para diferentes perfis, induzidas por diferentes visões do mesmo espaço de busca. Uma população tende a se tornar especializada em determinada característica perseguida pelo treinamento.

Pode-se ainda promover trocas de indivíduos entre as subpopulações. A migração teria o objetivo de transportar os indivíduos mais aptos de cada subpopulação para outras subpopulações, difundindo material genético diferenciado. Questões relativas a adaptação e aptidão são relevantes nesse contexto. Indivíduos especializados em determinada heurística tem alta avaliação, mas podem ficar restritos a uma única subpopulação, pois, devido a terem fraca adaptação, seriam mal avaliados em outras subpopulações e rapidamente eliminados. No contexto global, o melhor indivíduo seria aquele com maior poder de adaptação. As perspectivas de algoritmos evolutivos paralelos, sob a ótica sugerida por estes novos conceitos, são pelo menos razoáveis.

3.7.3 Novos cenários

O TPH obteve resultados comparáveis a outros enfoques encontrados na literatura e até melhores, especialmente para problemas de seqüenciamento de padrões (Oliveira e Lorena, 2002b,a). Os melhores indivíduos dentro de uma vizinhança são privilegiados pelo processo evolutivo, participando sobremaneira de cruzamentos e de procedimentos heurísticos de melhoria.

Avaliação de vizinhanças é um processo com um certo custo computacional, o que induz ao projetista a fazer algumas simplificações, tais como, examinar parte da vizinhança, apenas estimando e não efetivamente definindo se o indivíduo é ou não promissor.

Uma questão que esteve presente no início deste trabalho diz respeito aos indivíduos mal-adaptados. Eles são avaliados como tal, por ter sido encontrada em sua vizinhança pelo menos uma solução heuristicamente melhor. O indivíduo recebe um *ranking* referente a sua adaptação e é mantido na população por um período condizente com sua avaliação.

Todavia, a melhor solução da vizinhança de um indivíduo mal-adaptado é descartada (salvo quando se trata da melhor solução encontrada até o momento) e não contribui diretamente para o processo evolutivo.

A justificativa para o descarte está no fato de sua presença na população não ser fundamentalmente necessária, uma vez que o indivíduo que a originou é uma referência para sua localização. Além do mais, vários indivíduos diferentes podem ter a mesma solução melhor em suas vizinhanças (indivíduos vizinhos ou vizinhanças sobrepostas).

Um mecanismo alternativo a mera exclusão dessas soluções chegou a ser testado. O indivíduo foi definido por duas soluções: a original que efetivamente participa do processo evolutivo e a melhor encontrada na vizinhança. Durante a poda de indivíduos mal-adaptados, a melhor solução encontrada substitui a original, conservando o indivíduo na população e reavaliando-o para obtenção de um novo *ranking*. Essa reavaliação implica em mudança de ordem dos indivíduos podados dentro da população dinâmica e em pesados algoritmos de manipulação de estruturas de dados. Ao final, o desempenho alcançado não justificou as perdas com relação aos tempos computacionais.

Com relação ao TPH para otimização numérica, não foi possível utilizar heurísticas de treinamento mais efetivas, tais como o Simplex de Nelder e Mead, devido ao custo computacional. Foi utilizada, como alternativa, uma heurística de busca *em-Linha*, semelhante ao mecanismo de *path-relinking*. Todavia, o conjunto de referência é formado por indivíduos bem-adaptados à heurística de treinamento, sem necessariamente representarem soluções diversificada dentro do espaço de busca. Essa é a provável maior dificuldade do TPH quando aplicado a espaços contínuos, os quais requerem mecanismos mais racionais para discriminar os indivíduos a serem melhorados através de heurísticas de busca local.

Além disso, no TPH, não foram incluídos mecanismos explícitos de diversificação populacional. O conceito de melhor indivíduo em uma vizinhança pode vir a ser empregado de tal forma a sistematizar a exploração dessas vizinhanças, evitando-se sobreposições delas e conseqüentemente excessiva intensificação em torno das regiões promissoras.

Com base nessas observações, um novo cenário é, de antemão, visualizado, onde regiões promissoras são melhor delimitadas, evitando-se sobreposição. O *path-relinking* pode ser melhor aproveitado, como um efetivo mecanismo de intensificação. Por esses motivos, este trabalho avança no sentido de se contribuir com uma nova metaheurística evolutiva que possa ser competitiva em uma gama maior de problemas.

CAPÍTULO 4

DETECÇÃO DE REGIÕES PROMISSORAS POR AGRUPAMENTO DE GENÓTIPOS

Há um previsto aumento no número de instâncias de esquemas que representem regiões onde se concentram indivíduos bem avaliados. A localização dos grupos de indivíduos que se formam ao longo do processo de evolução pode fornecer informação sobre quais regiões têm um maior potencial de serem encontradas soluções de qualidade.

O *Evolutionary Clustering Search* (ECS) ou busca evolutiva através de agrupamentos, como o próprio nome sugere, utiliza um processo de agrupamento de indivíduos para guiar uma busca evolutiva. O processo de agrupamento, resultado da própria dinâmica evolutiva, é usado para se gerar um conjunto de soluções de referência para regiões supostamente promissoras.

4.1 Regiões promissoras e agrupamentos

No cenário de algoritmos evolutivos híbridos (AEH), os ecossistemas naturais têm inspirado a construção de algoritmos flexíveis, coerentes e eficazes, nos quais o processo evolutivo é somente parte do processo de solução (Li et al., 2002). O espaço de busca pode ser entendido como regiões geográficas onde grupos de indivíduos vivem e se reproduzem, interagindo com o meio-ambiente, promovendo uma forma de exploração desse meio. A concentração de indivíduos em determinadas regiões pode indicar recursos naturais abundantes.

A idéia de detectar regiões promissoras por mérito de freqüência nasce de uma interpretação inversa da Teoria dos Esquemas (Holland, 1975). Os indivíduos que mais participam do processo evolutivo são aqueles que contém, em seu genótipo, blocos construtivos bem avaliados. Considerando que blocos construtivos representam famílias de soluções candidatas (regiões do espaço de busca) com genes em comum, pode-se estabelecer uma forma coerente de detecção de regiões promissoras baseada na quantidade de soluções geradas em cada região de busca.

Pode-se prever os benefícios de se intensificar a busca nas regiões mais amostradas observando o comportamento típico de AG's. O processo evolutivo tende a privilegiar os indivíduos melhor avaliados que por sua vez representam regiões de busca. As funções-teste usadas para otimização numérica são úteis para se esboçar o comportamento de um AG ao longo das gerações com relação a supostas regiões promissoras.

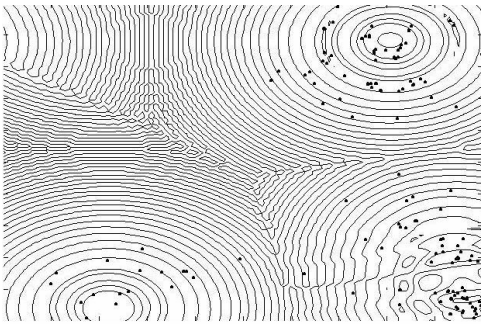
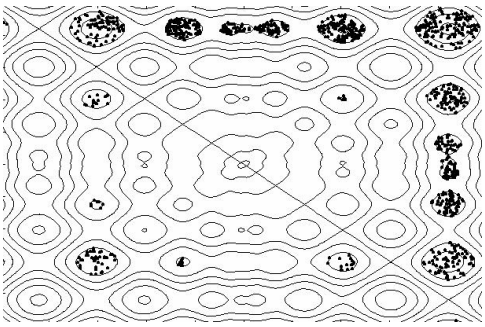
a)  b) 

FIGURA 4.1 – Típica convergência de AG's sobre regiões promissoras: a) função de *Langerman* e b) função de *Schwefel*.

Na Figura 4.1, é mostrado o mapa de contorno em duas dimensões das funções de *Langerman* e *Schwefel*. Os pontos são soluções candidatas (indivíduos) sobre a superfície de função em uma simulação típica. Pode-se observar que as regiões promissoras funcionam como atratores de indivíduos. Isso é facilmente explicado pelo fato de os melhores indivíduos serem mais selecionados e, portanto, gerarem mais descendentes com suas características genéticas. Posto em outros termos, os descendentes tendem a *habitar* as mesmas regiões de seus pais, aumentando a amostragem nessas regiões.

Como foi mencionado anteriormente, heurísticas de melhoria podem ser combinadas com AG's de diferentes formas para resolver problemas de otimização, em geral, acrescentando maior eficácia e precisão ao processo de busca. Entretanto, trabalhos envolvendo algoritmos híbridos, em geral, relatam um aumento no número de chamadas à função objetivo que, para problemas reais, pode ter um custo computacional significativo (Birru et al., 1999).

O principal desafio de tais métodos híbridos é a definição de estratégias eficientes para cobrir todo o espaço de busca, possibilitando a aplicação de busca local somente em indivíduos realmente promissores. O elitismo pode ser considerado um critério para aplicação de heurísticas de melhoria em indivíduos da população. Outro critério é o probabilístico, no qual uma parte aleatória da população é melhorada através de busca local.

A busca *scatter* (*Scatter Search - SS*) possui um mecanismo explícito de manutenção de diversidade através de dois subconjuntos de soluções candidatas: um com as melhores soluções e outro com as soluções mais *distantes* das melhores (diversificadas). Toda solução gerada é melhorada através de um otimizador local. O *SS* possui uma sistemática de exploração do espaço de busca, utilizando soluções de qualidade e diversificadas, combinada com a intensificação por busca local. Tanto os melhores indivíduos quanto

aqueles mais afastados das regiões promissoras são considerados nessa sistemática (Glover, 1998).

Mais recentemente, uma nova estratégia foi proposta através do Algoritmo Híbrido Contínuo (*Continuous Hybrid Algorithm - CHA*) (Chelouah e Siarry, 2003). Nele, o processo evolutivo ocorre normalmente, sem intervenção de otimizadores locais, até ser detectada uma região promissora, segundo um critério baseado na perda de diversidade populacional: quando a maior distância entre o melhor e os demais indivíduos da população for menor que um certo *raio*. A seguir, o domínio de busca é reduzido, um *simplex* inicial é construído dentro dessa região, em torno do melhor indivíduo, e uma busca local baseada no *Simplex de Nelder e Mead* (Nelder e Mead, 1965) é iniciada.

No que tange processo evolutivo como um todo, o *CHA* tem uma limitação. A intensificação é iniciada uma vez, depois da perda de diversidade, e o processo evolutivo não pode ser continuado, a seguir, a menos que uma nova população seja gerada.

Dentro dessa mesma linha, outro enfoque tem sido proposto, também para otimização numérica, chamado UEGO (Jelasity et al., 2001). O UEGO executa o método *descida da encosta* (*hill-climbing*), paralelamente, em regiões de busca restritas, chamadas de agrupamentos (*clusters*). O volume dos *clusters* são reduzidos a medida em que a busca prossegue, através de um efeito de esfriamento similar a um *recozimento simulado*. O UEGO não pareceu encontrar bons resultados para funções com muitas dimensões (Jelasity et al., 2001).

Alguns enfoques, como o UEGO, tem evocado o conceito de espécies, quando manipulando otimização multimodal (Jelasity et al., 2001) ou otimização multi-objetivo (Li et al., 2002). A idéia básica é dividir a população em espécies, de acordo com certas similaridades. Cada espécie evolui em torno de um indivíduo dominante, numa área delimitada, independentemente das demais.

Outra idéia inspirada na natureza é a de nichos. Um ecossistema natural possui diferentes subsistemas (nichos) que contém muitas espécies (subpopulações). Tais nichos podem evoluir em paralelo, convergindo para diferentes pontos ótimos no espaço de busca. Os critérios que definem se um indivíduo pertence a um ou de outro nicho é parte da estratégia específica de cada algoritmo. Ao invés de evoluir uma única população de indivíduos similares, ecossistemas evoluem subpopulações para ocupar diferentes nichos. A idéia de nichos foi introduzida em AG's inicialmente para manter a diversidade populacional (Deb e Goldberg, 1989; Goldberg e Richardson, 1987). Posteriormente, foi estendida para torná-los capazes de encontrar múltiplas soluções.

O método *Clustering Based Niching* (*CBN*) aplica conceitos de espécies para preservar diversidade, capacitando-o a encontrar múltiplas soluções. O *CBN* modela espécies usando múltiplas populações e identifica cada espécie através de agrupamento de genótipos ou fenótipos. Há também uma preocupação em manter o processo de agrupamento independente do algoritmo evolutivo e menos impactante quanto possível (Streichert et al., 2003)

A proposta neste capítulo focaliza a detecção de regiões promissoras através de agrupamento de genótipos[1]. Este enfoque é chamado de *Evolutionary Clustering Search* (ECS) ou busca evolutiva através de agrupamentos. Neste texto, a sigla relativa ao nome em inglês é usada para referenciar a proposta.

4.2 Linhas gerais do *Evolutionary Clustering Search*

No *Evolutionary Clustering Search*, um processo de agrupamento iterativo trabalha simultaneamente com um algoritmo evolutivo, contabilizando as operações (seleção ou atualização de indivíduos) ocorridas em regiões do espaço de busca e identificando aquelas que merecem especial interesse. Em outras palavras, o processo de agrupamento identifica grupos de indivíduos com similaridades, significando um padrão predominante de genótipos e conseqüentemente uma certa região do espaço de busca.

O termo *cluster* é utilizado neste texto como a entidade abstrata que referência um agrupamento de indivíduos. Os *clusters* funcionam como janelas flutuantes que enquadram regiões de busca. Grupos de soluções candidatas relativamente próximas podem corresponder a relevantes regiões de atração que devem ser exploradas tão logo sejam detectadas, através de heurísticas de busca local específicas.

Além do evidente benefício de se determinar quais regiões devam ser exploradas, o próprio processo de agrupamento, através da interação entre indivíduos internos ao *cluster*, pode interferir na forma como região é explorada, determinando um tipo de intensificação de busca nessas regiões. É esperado uma melhoria no processo de convergência associado a uma diminuição no esforço computacional em virtude do emprego mais racional de otimizadores locais.

[1]Poderia ser agrupamentos de fenótipos visto que, nas aplicações desenvolvidas neste trabalho, o espaço de fenótipos é igual ao de genótipos.

4.3 Formalização da proposta

O *Evolutionary Clustering Search* (ECS) procura localizar regiões promissoras através do enquadramento destas por *clusters*. Um *cluster* é definido pela tripla:

$$\mathcal{C} = \{c, r, \mathcal{B}\} \tag{4.1}$$

onde c é o *centro*, r é o *raio* e $\mathcal{B}$ é uma *estratégia de busca* associada ao *cluster*.

O *centro* é um indivíduo (ou solução), representante do *cluster*, que identifica a sua localização dentro do espaço de busca. Considerando a região de busca como uma hiperesfera, o *raio* estabelece a *distância* máxima, a partir do centro, até a qual um indivíduo pode ser associado ao *cluster*.

A métrica de *distância* está relacionada à estratégia $\mathcal{B}$ e ao espaço de busca. A distância euclidiana, por exemplo, é uma métrica definida no espaço de fenótipos e que pode também refletir o esforço necessário para se alcançar uma dada solução a partir de outra através de movimentos de uma dada heurística de melhoria.

A estratégia $\mathcal{B}$ é uma sistemática de intensificação de busca, na qual indivíduos de um *cluster* interagem entre si, ao longo do processo de agrupamento, gerando novos indivíduos na mesma região. A interação entre indivíduos ocorre por meio de movimentos heurísticos em resposta à *ativação* de *clusters* por operações de seleção ou atualização. Podem ser empregadas, ainda como parte de $\mathcal{B}$, heurísticas de melhoria, considerando c e r como parâmetros, caso a região associada a um dado *cluster* seja considerada promissora.

Inicialmente, o centro c é obtido aleatoriamente, mas ele tende a *deslizar* progressivamente em direção a regiões de busca realmente promissoras que estejam próximas. O volume total do *cluster* é definido através do raio r e pode ser calculado considerando a natureza do problema. Um importante requisito é que r deve definir um subespaço de busca apropriado para a estratégia de busca $\mathcal{B}$.

Em problemas de otimização numérica, r deve ser suficiente para cobrir todo o espaço de busca com um certo número de *clusters*. Em otimização combinatória, r pode ser definido em termos de número de movimentos necessários para transformar uma solução candidata em outra dentro de uma vizinhança definida por uma heurística qualquer. O raio está, portanto, associado a estratégia de busca referente ao *cluster*.

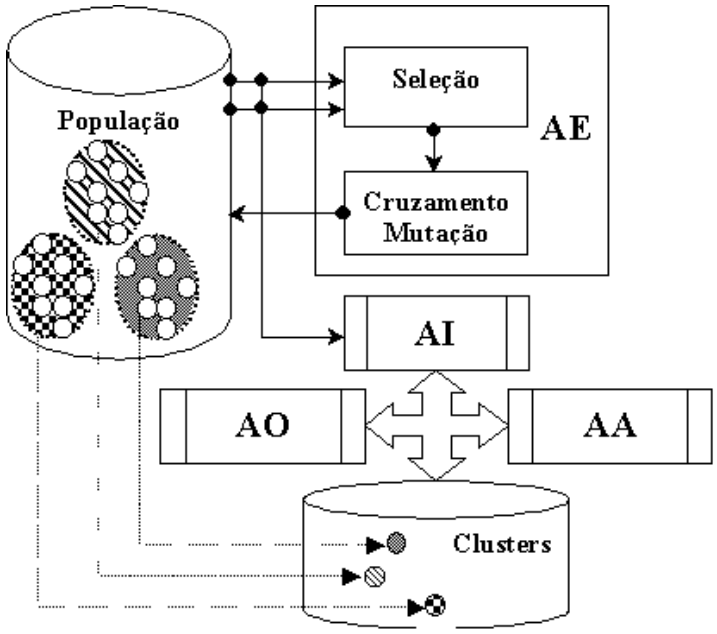


FIGURA 4.2 – Diagrama conceitual do *ECS*.

O ECS pode manter *clusters* com volumes diferentes, assim como pode empregar diferentes estratégias de busca em cada *cluster*, dependendo do subespaço de busca enquadrado por ele[2]. Considerando-se um problema monobjetivo, pode-se simplificar esta proposta para se trabalhar com uma única estratégia de busca, comum a todos os *clusters*. Os componentes do ECS estão conceitualmente descritos a seguir. Detalhes sobre as aplicações estão explicados posteriormente.

4.3.1 Componentes

O ECS consiste de quatro componentes com atribuições diferentes e conceitualmente independentes. São eles:

a) um algoritmo evolutivo (AE);

b) um agrupador iterativo (AI);

c) um analisador de agrupamentos (AA); e

d) um algoritmo de otimização local (AO).

Na Figura 4.2, é mostrado como os quatro componentes, a população de indivíduos e os *clusters* interagem entre si. Cada *cluster* é representado por uma solução candidata que possui um certo grau de similaridade com um grupo de indivíduos da população.

[2] *Clusters* com raios e estratégias diferentes não foram testados neste trabalho.

O componente AE funciona como um gerador contínuo de soluções candidatas. A população evolui independente dos processos ocorrendo nos demais componentes. Indivíduos são selecionados, recombinados, e atualizados para as próximas gerações, ao mesmo tempo em que são mantidos *clusters* representativos desses indivíduos.

O componente AI coleta informação similar (soluções candidatas) em grupos, mantendo uma solução representativa associada a essa informação, chamada de centro de *cluster*. O termo *informação* é usado aqui em lugar do termo *indivíduo*. Ambos são soluções candidatas em diferentes contextos: indivíduos evoluem e a informação é agrupada. Qualquer solução candidata que não for parte da população é chamada de informação.

Para evitar esforço computacional extra, o componente AI foi desenhado como um processo iterativo que forma grupos através da leitura da informação presente nos indivíduos selecionados ou atualizados pelo componente AE. Não se trata de um processo massivo, em lote, mas um processo simultâneo à evolução ocorrendo em AE. Um grau de similaridade, baseado em alguma métrica de distância, deve ser definido, *a priori*, para permitir o processo de agrupamento.

O componente AA verifica cada *cluster*, em intervalos regulares de gerações, indicando quais são os promissores. A *densidade* é o número de seleções ou atualizações ocorridas recentemente no *cluster*. O AA é também responsável pela eliminação de *clusters* com *baixa densidade*.

Finalmente, o componente AO é um otimizador local que provê a intensificação de busca nas supostas regiões promissoras, enquadradas por *clusters*. Este processo ocorre após o AA ter descoberto um *cluster* alvo. Entretanto, ele pode ser suprimido do *ECS* em virtude da intensificação de busca ser um processo contínuo inerente ao AI, executada sempre que uma informação nova é agrupada.

4.3.2 Processo de agrupamento

O processo de agrupamento descrito aqui é baseado na idéia de que um sistema pode aprender sobre um ambiente externo com a participação das crenças do próprio sistema (Yager, 1990). Dessa forma, a informação que foi previamente aprendida participa do processo de aprendizado futuro. O aprendizado participativo serve de inspiração para um processo similar onde se deseja agrupar novos dados com a participação de agrupamentos já consolidados (Silva, 2003).

O componente AI é o núcleo do ECS, trabalhando como um classificador de informação que mantém no sistema apenas aquela que for relevante para o processo de intensificação

de busca. Toda a informação gerada por AE (indivíduos) é lida por AI que tenta agrupá-la como uma informação conhecida. Se a informação for considerada suficientemente nova, ela é mantida como um centro em um novo *cluster*. Caso contrário, a informação é considerada redundante, causando uma perturbação no centro de *cluster* mais similar. Tal perturbação é chamada de *assimilação* e consiste basicamente em atualizar o centro com a nova informação recebida.

Existem várias métricas de distâncias que podem ser utilizadas dependendo de fatores, tais como codificação, heurísticas de vizinhança, etc. No Capítulo 3 são citadas as métricas baseadas em fenótipos, *fitness* e de movimentos heurísticos. Em geral, as que apresentam menores custos computacionais são as distâncias que não utilizam chamadas à função objetivo (distâncias de genótipos ou fenótipos), nem tampouco analisam paisagens de aptidão para computar quantidade de movimentos para transformar uma solução em outra (Reeves, C.R., 1999).

4.3.3 Processo de assimilação

A assimilação é aplicada ao centro mais *próximo*, c_i, considerando o indivíduo lido, s_k, e gerando uma atualização de centro, c_i'. A forma geral para a assimilação é:

$$c_i' = c_i \oplus \beta(s_k \ominus c_i) \tag{4.2}$$

onde $\oplus$ e $\ominus$ são operações abstratas sobre as soluções candidatas c_i e s_k que significam, respectivamente, soma e subtração de soluções. O intervalo $(s_k \ominus c_i)$ é o vetor de diferenças entre cada uma das n variáveis que compõem as soluções s_k e c_i, calculado de acordo com alguma métrica de distância.

Um certo percentual β desse vetor é usado como passo para atualizar c_i, gerando c_i'. De acordo com β, a assimilação pode assumir diferentes formas:

a) um parâmetro constante $\beta \in [0, 1]$, indicando que c_i move-se em direção a s_k a um passo constante que pode ser mais ou menos conservador (assimilação simples);

b) um vetor de parâmetros constantes e distintos $\beta_j \in [0, 1]$, onde cada elemento é usado para gerar uma nova solução candidata entre c_i e s_k e cada solução é avaliada, segundo a função objetivo. O novo centro c_i' é posicionado nessa melhor solução encontrada (assimilação por caminho);

c) um vetor aleatório $n-$dimensional que é usado como parâmetro de cruzamen-

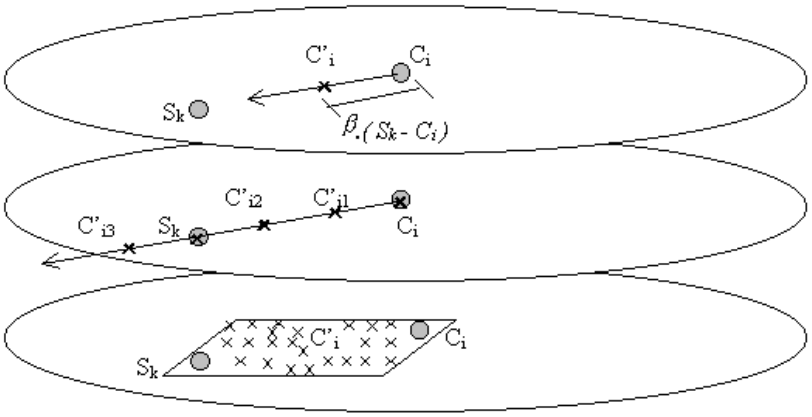


FIGURA 4.3 – Formas de assimilação: simples, caminho e recombinação.

to entre c_i e s_k. Dessa forma, c'_i pode assumir uma posição aleatória dentro do hiperplano contendo s_k e c_i (assimilação por recombinação).

A assimilação simples e assimilação por recombinação geram um único ponto interno a ser avaliado a seguir. Não há nenhum compromisso desse ponto ser o melhor ponto dentro do *cluster*. Na assimilação por caminho, ao contrário, podem ser gerados e avaliados diversos pontos internos (interpolação) e até alguns externos (extrapolação). O número de pontos a serem amostrados determina o quão custoso esse tipo de assimilação pode ser. Apesar do custo, há uma aparente vantagem em se manter o centro na melhor solução candidata até então encontrada no *cluster*.

Movimentos de extrapolação desse intervalo podem conduzir o centro para fora dos limites do *cluster*, sendo portanto imperativo a utilização de mecanismos que evitem que *clusters* se sobreponham ou, uma vez sobrepostos, seja eliminado um deles.

A eliminação de *clusters* sobrepostos tende a ocorrer naturalmente através de uma espécie de competição entre eles. Indivíduos que pertençam a mais de um *cluster* são agrupados no mais recente deles, não necessariamente naquele com centro mais próximo. Dessa forma, um *cluster* compartilhando indivíduos com outros mais recentes tende a perder densidade e ser eliminado posteriormente.

4.3.4 Assimilação por caminho e *path-relinking*

Movimentos exploratórios no caminho que interliga duas soluções candidatas são comumente referenciados como *path-relinking* (*PR*) (Glover, 1998). *PR* é uma estratégia de intensificação usada para explorar trajetórias conectando soluções. A idéia foi concebida inicialmente para as buscas tabu e *scatter*, mas tem sido satisfatoriamente empregada

em GRASP (*Greedy Randomized Adaptive Search Procedure*) (Aiex et al., 2003; Festa, 2003; Resende e Ribeiro, 2003).

As trajetórias são construídas usando-se duas ou mais soluções. Seja uma solução inicial s_0 e uma solução guia s_g, a cada passo um novo atributo de s_g é incorporado por s_0, gerando uma nova solução que é avaliada e eventualmente melhorada através de busca local. É comum que a solução guia tenha alta qualidade e faça parte de um conjunto de soluções de referência (Glover, 1998). Aplicações baseadas em busca *scatter*, por exemplo, costumam empregar tanto o *PR* quanto heurísticas de melhoria.

No caso do ECS usando assimilação por caminho, é mantido apenas um conjunto de soluções de referência (centros de *cluster*) que pode vir a ser de alta qualidade e diversificado ao mesmo tempo. A alta qualidade é resultado das melhorias obtidas através de *PR*. A diversificação advem de cada centro ser representante de uma região do espaço de busca enquadrada por um *cluster*.

Visto por esse prisma, o ECS é similar à busca *scatter*. A diferença reside, entre outros, no número de conjuntos de referência empregados por cada um, na sistemática de inclusão de soluções nesses conjuntos de referência e na forma como é aplicada a busca local: no caso do *ECS*, apenas quando um dado *cluster* é considerado promissor.

Uma questão fundamental em se tratando de *path-relinking* diz respeito às trajetórias entre soluções. Heurísticas de vizinhança diferentes geram paisagens de aptidão diferentes e conseqüentemente trajetórias diferentes (Reeves, C.R., 1999). A forma como está *estruturado* o espaço de busca pode ser determinante para o bom desempenho de um método de busca (Preux e Talbi, 1999). Algoritmos utilizando codificação binária, por exemplo, podem fixar os elementos comuns entre s_0 e s_g e, progressivamente, irem incorporando um ao outro (Alfandari et al., 2003). Existe ainda um passo que define quão progressiva é essa incorporação de atributos.

Em geral, técnicas baseadas em *path-relinking* procuram gerar soluções intermediárias eqüidistantes uma das outras, segundo a mesma heurística de vizinhança que induz a paisagem de aptidão. Eventualmente, tais soluções podem extrapolar o subespaço de busca que compreende s_0 e s_g (Reeves e Yamada, 1998). Entretanto, gerar soluções intermediárias utilizando uma heurística de vizinhança específica pode não ser uma tarefa trivial.

Em (Linhares, 2002), é feita uma comparação entre métricas de distância associadas à heurística 2-Opt e à heurística 2-Troca para problemas de seqüenciamento de padrões.

Gerar a menor seqüência 2-Opt de soluções intermediárias entre s_0 e s_g é um problema *NP-árduo*. Por outro lado, pode-se gerar uma seqüência 2-Troca através de um algoritmo linear (Linhares, 2002). Na prática, pode ser mais conveniente ter-se heurísticas diferentes para cada um dos movimentos exploratórios utilizados em um determinado algoritmo de busca (Reeves, C.R., 1999). Inclusive, pode haver benefícios em termos de desempenho em se trabalhar com múltiplas paisagens de aptidão.

4.3.5 Análise de *clusters*

Sempre que um *cluster* atinge uma certa densidade, significando que um certo padrão de informação se tornou predominantemente gerado pelo processo de evolução, tal *cluster* é melhor investigado para acelerar o processo de convergência nele.

Heurísticas podem ser utilizadas para detectar se um *cluster* está sendo ativado significativamente. Por exemplo, considere que, a cada geração t, um número $\mathcal{NS}$ de indivíduos são selecionados ou atualizados. Se a população e o número total de *clusters*, $|C_t|$, estiverem uniformemente distribuídos pelo espaço de busca, cada um deles deveria ser ativado, a cada geração, $\mathcal{NI}$ vezes, onde:

$$\mathcal{NI} = \frac{\mathcal{NS}}{|C_t|} \tag{4.3}$$

A cada geração, os *clusters* com baixa densidade são eliminados, como parte de um mecanismo que permite descobrir outros centros de informação, mantendo-se enquadrados somente os mais ativos. A eliminação de *clusters* não afeta a população diretamente, pois apenas os centros de informação eliminados são considerados irrelevantes para o processo. Os indivíduos continuam existindo até que sejam substituídos, dependendo da política de atualização de indivíduos adotada pelo algoritmo evolutivo.

Em analogia aos ecossistemas naturais, os indivíduos em cada região interagem com o meio-ambiente, promovendo uma forma de exploração desse meio. Uma vez esgotados os recursos naturais de uma região, os indivíduos podem migrar para outras regiões ainda não exploradas.

4.4 Aplicação para minimização de funções numéricas sem restrição

Nesta etapa, foram escolhidas 13 funções-teste, com dimensões variadas, dependendo do objetivo do experimento. São elas: *Michalewicz*, *Langerman*, *Shekel*, *Rosenbrock*, *Sphere*, *Schwefel*, *Griewank*, *Rastrigin*, *Ackley*, *Goldstein*, *Zakharov*, *Easom* e *Hartman*. A seguir, os detalhes de implementação e os resultados obtidos são apresentados.

4.4.1 Implementação

Para minimização de funções numéricas sem restrição, o ECS foi implementado da seguinte forma. O componente AE é um algoritmo genético do tipo não-geracional (*steady-state*) (Davis, 1991), utilizando codificação real. São empregados alguns operadores genéticos bem conhecidos, tais como cruzamento BLX (*Blend crossover*) (Eshelman e Schaffer, 1993), e a mutação não-uniforme (Michalewicz, 1996).

Resumidamente, a cada geração um número constante $\mathcal{NS}$ de indivíduos são selecionados e recombinados. Seus descendentes eventualmente sofrem mutação e são atualizados na mesma população original (atualização do tipo *steady-state*). Não há uma unidade de tempo explícita chamada de geração. Uma geração é implicitamente definida como um certo número de seleções. Filhos podem diretamente competir com seus pais nas operações de seleção que ocorrerem após eles terem sido gerados.

4.4.1.1 Seleção com pressão auto-adaptativa

O *ECS* empregou um operador *seleção com pressão auto-adaptativa* (SPAA), proposta neste trabalho, e inspirada na função de avaliação com ajuste de pressão seletiva utilizada em (De Oliveira e Gatto, 1995). A designação *pressão seletiva* está associada a algum tipo de controle da pressão de seleção, forçando alterações na freqüência com que são selecionados indivíduos em uma população (Bäck, 1994).

A pressão de seleção é dita auto-adaptativa devido ao próprio indivíduo determinar, dinamicamente, a cada seleção, a sua pressão de seleção que varia de acordo com dois parâmetros: o número ns_k de vezes em que o indivíduo tem sido selecionado e sua qualidade relativa ao melhor indivíduo até então encontrado.

O algoritmo da SPAA é similar ao da roleta (Goldberg, 1989), no qual um indivíduo é selecionado quando um certo limiar de contribuições de *fitness* é alcançado. A contribuição de cada indivíduo, ou seja, a *avaliação seletiva* Z de um indivíduo s_k, a cada seleção, é dada por:

$$Z(s_k) = \frac{1}{|f^* - f(s_k) + 1|^{ns_k}} \tag{4.4}$$

onde $|f^* - f(s_k)|$ fornece a diferença entre a avaliação do indivíduo s_k e a avaliação do melhor indivíduo até então encontrado. Para evitar problemas numéricos, ns_k é considerado até um certo valor de teto, em geral 5.

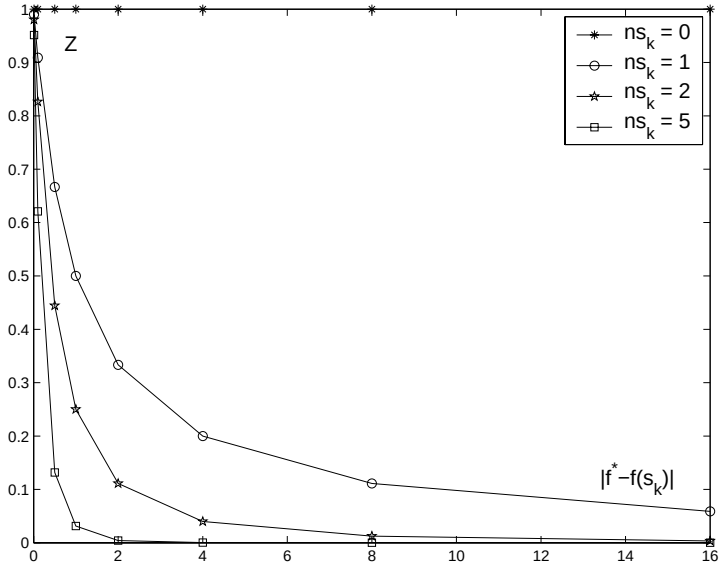


FIGURA 4.4 – Seleção com pressão auto-adaptativa.

Observa-se que $Z(s_k) \in [0, 1]$, onde 0 significa um indivíduo mal avaliado. Um indivíduo é bem avaliado em duas situações: a) quando $|f^* - f(s_k)| = 0$, pois independentemente de ns_k $Z(s_k) = 1$, e b) quando $ns_k = 0$, pois, nesse caso, s_k ainda não foi selecionado.

A SPAA também é adaptativa, pois o processo evolutivo interfere no processo de seleção (Eiben et al., 1999). À medida em que a população evolui e novos indivíduos melhores são encontrados, a avaliação seletiva de cada indivíduo também muda em função disso.

A Figura 4.4 mostra a avaliação seletiva para diferentes $|f^* - f(s_k)|$ e ns_k. Para $ns_k = 0$ todos os indivíduos têm $Z = 1$, ou seja, a SPAA seleciona de forma aleatória, sem considerar a qualidade do indivíduo. À medida em que são selecionados ($ns_k > 0$), Z tende a decair exponencialmente.

A participação de indivíduos de qualidade é importante no processo de assimilação do ECS. A SPAA possibilita que tanto indivíduos de qualidade quanto aqueles com menor participação tenham grande chances de serem selecionados.

4.4.1.2 Componente AI

O componente AI executa um agrupamento iterativo de cada indivíduo selecionado. Um número máximo de *clusters* $\mathcal{MC}$ deve ser definido *a priori*, evitando-se que o processo de agrupamento se torne demasiadamente lento, em especial, em aplicações para problemas com muitas variáveis. O $i-$ésimo *cluster* tem seu próprio centro c_i e um raio r_t que é

comum aos demais *clusters*. A cada geração, r_t é calculado por:

$$r_t = \frac{x^{sup} - x^{inf}}{2\sqrt[n]{|C_t|}} \tag{4.5}$$

onde $|C_t|$ é o atual número de *clusters* (inicialmente, $|C_t| = \mathcal{MC}$), x^{sup} e x^{inf} são, respectivamente, os conhecidos limites superior e inferior do domínio da variável x, considerando que todas têm o mesmo domínio.

Se um indivíduo selecionado s_k está *distante* de todos os centros (uma distância acima de r_t), então um novo *cluster* deve ser criado. Apesar de $\mathcal{MC}$ prevenir a criação de um número muito grande de *clusters*, essa limitação não se constitui em um problema, visto que espera-se que os *clusters* possam se deslocar pelo espaço de busca sempre que uma região desse espaço começar a atrair mais indivíduos.

4.4.1.3 Processo de assimilação

A assimilação é um processo que pode ser implementado de diferentes formas. Algumas delas foram testadas neste trabalho. O indivíduo selecionado s_k e o centro c_i mais próximo participam da assimilação através de alguma operação que use a nova informação contida em s_k para causar uma perturbação na localização do centro. Em outras palavras, o centro vai adquirir parte dos atributos contidos na nova informação. O parâmetro β está associado ao grau de desordem no processo de assimilação. A assimilação simples para variáveis reais é dada por:

$$c_i' = c_i + \beta(s_k - c_i) \tag{4.6}$$

onde β é um escalar constante entre entre $]0, 1[$. Os melhores resultados para otimização numérica foram obtidos mantendo-se os centros mais conservadores em relação a nova informação ($\beta = 0.05$) (Oliveira e Lorena, 2004).

A assimilação por recombinação para variáveis reais pode ser escrita como:

$$c_i' = c_i + \overrightarrow{\beta} \cdot (s_k - c_i) \tag{4.7}$$

onde $\overrightarrow{\beta}$, neste caso, é um vetor aleatório $n-$dimensional, i.e., na mesma dimensão de s_k e c_i que significa um conjunto de percentuais. O produto "$\cdot$" aplica cada um desses

percentuais à diferença $(s_k - c_i)$, gerando um novo vetor a ser adicionado ao atual centro.

Com intuito de aproveitar o potencial desse tipo de assimilação que nada mais é que uma operação genética entre indivíduos do mesmo *cluster*, essa formulação foi estendida para simbolizar o cruzamento BLX, usado nesta aplicação.

O parâmetro blx_α define o percentual de *ampliação* do hiperplano Ψ contendo s_k e c_i. Em outras palavras, o novo centro c'_i pode ser deslocado para ponto aleatório dentro de Ψ, ampliado um certo percentual $blx_\alpha\%$. Dessa forma, para um $blx_\alpha \in [0, 1]$, indicando o percentual de ampliação de Ψ, pode-se redefinir $\overrightarrow{\beta}$ em função de blx_α e de um vetor aleatório $n-$dimensional $\overrightarrow{\beta'} \in [0, 1]$:

$$\overrightarrow{\beta} = -blx_\alpha + \overrightarrow{\beta'}(1 + 2blx_\alpha) \tag{4.8}$$

Dessa forma, $\overrightarrow{\beta}$ é um vetor aleatório com valores em $[-blx_\alpha, 1 + blx_\alpha]$ e blx_α passa a ser o parâmetro de desempenho desse tipo de assimilação.

A assimilação por caminho está associada ao *path-relinking* (PR). Diferentemente dos dois métodos anteriores, o PR realiza movimentos exploratórios na trajetória que interconecta s_k e c_i. Dessa forma, o próprio processo de assimilação já é uma forma de intensificação de busca dentro do *cluster*. O novo centro c'_i é dado por:

$$\begin{array}{rcrcl} c'_i & = & c'_V, & f(c'_V) & = & \min\left\{f(c'_1), f(c'_2), \cdots, f(c'_\eta)\right\} \\ & & & c'_j & = & c_i + \beta_j(s_k - c_i) \\ & & & \beta_j & \in & \{\beta_1, \beta_2, \cdots, \beta_\eta\} \end{array} \tag{4.9}$$

onde β, neste caso, é um conjunto com η escalares distintos uniformemente distribuídos no intervalo $]0, 1[$. Cada β_j é usado para gerar um centro c'_j que é avaliado e o melhor deles, c'_V, ao final do processo passa a ser o novo centro c'_i.

Basicamente, a escolha do método de assimilação deve considerar também aspectos de custo computacional. A assimilação por caminho juntamente com um método de busca local agressivo podem tornar o ECS razoavelmente mais lento que com os demais métodos de assimilação.

O componente AI, empregando a métrica de distância euclidiana calculada para cada *cluster* em um problema $n-$dimensional, tem complexidade $O(\mathcal{MC} \cdot n)$. Quando o

processo de assimilação emprega PR, o custo de avaliação da função objetivo passa a fazer parte da complexidade do componente AI.

4.4.1.4 Componente AA

O componente AA é evocado quando um *cluster* é atualizado com algum indivíduo selecionado que pertença a ele. Quando isso ocorre o AA verifica se o *cluster* já pode ser considerado *promissor*. Além dessa intervenção, o AA também executa um *esfriamento* de todos os *clusters* que foram ativados a cada geração e elimina aqueles que não foram ativados.

O *esfriamento* restaura o estado inicial de um *cluster*, i.e., coloca-o como se ainda não houvesse ocorrido nenhuma ativação. De certa forma, isso limita os efeitos das ativações a uma única geração. Nesta aplicação, o esfriamento não zerou a quantidade de ativações do *cluster*, apenas colocou-o em um estado chamado de *inativo*, sem eliminá-lo.

Os *clusters esfriados* eventualmente podem ser *reaquecidos* através de seleções de indivíduos em suas regiões de busca. Um *cluster* que permaneça inativo após uma geração é eliminado pelo componente AA posteriormente. Um *cluster* se torna *promissor* quando atinge uma certa densidade λ_t dada por:

$$\lambda_t \geq \mathcal{PD} \cdot \frac{\mathcal{NS}}{|C_t|} \tag{4.10}$$

onde $\mathcal{PD}$ é a pressão de densidade que permite controlar a sensibilidade do componente AA. Esse parâmetro indica quantas vezes a densidade deve estar acima do normal $\frac{\mathcal{NS}}{|C_t|}$ para que um *cluster* seja considerado promissor. Nesta aplicação, desempenhos satisfatórios foram obtidos empiricamente ajustando-se $\mathcal{NS} = 200$ e $\mathcal{PD} = 2,5$. Observa-se que alterações em $|C_t|$ causam alterações em λ_t e no raio r_t.

4.4.1.5 Componente AO

O componente AO foi implementado baseado na busca direta de *Hooke e Jeeves* (*HJD*) (Hooke e Jeeves, 1961). A *HJD* apresenta algumas características interessantes: boa convergência, baixos requisitos de memória e faz uso de cálculos matemáticos básicos. O método essencialmente trabalha através de dois tipos de movimentos. A cada iteração existe um movimento exploratório de passo discreto por direção de coordenada. No algoritmo mostrado a seguir, o movimento exploratório bem-sucedido é tentado nas duas direções, indicado por $\pm dx$.

```
Seja (xc, fc) a solução a ser melhorada e sua respectiva avaliação;
Seja n o número de variáveis em xc;
Seja (dx, γ) o passo de busca e o fator de redução de passo, respectivamente;
while não ocorrer condição de parada do
  fp := fc;
  xr := xc;
  Movimentos Exploratórios: 1 movimento xr := xr ± dx por variável;
  Guarda o melhor xr obtido e sua respectiva avaliação fp;
  reduzPasso := .V.;
  while fp < fc do
    reduzPasso := .F.;
    fc := fp;
    xp := xc;
    xc := xr;
    for i := 1 até n do
      Extrapola a linha favorável: xr[i] := xr[i] + (xr[i] − xp[i]);
    end for
    Avalia xr: fp := função objetivo (xr);
    Movimentos Exploratórios: 1 movimento xr := xr ± dx por variável;
  end while
  if reduzPasso = .V. then
    Reduz passo: dx := γdx;
  end if
end while
```

Supondo que a linha interligando o primeiro e último pontos desse movimento exploratório represente uma direção favorável, uma extrapolação é feita nessa mesma direção antes das variáveis serem exploradas novamente. Sua eficiência depende da escolha de um parâmetro chamado de passo inicial dx. Nesta implementação, dx foi ajustado para 5% do raio inicial.

O *Simplex de Nelder e Mead*(SNM) (Nelder e Mead, 1965) foi investigado como possibilidade para o componente AO. A decisão de se utilizar o *HJD* foi centrada na literatura sobre métodos de busca local diretos. Para poucas variáveis, o SNM é um método robusto e confiável, apesar de relativo um custo computacional. Além disso, como já foi mencionado anteriormente, são necessários n vetores de parâmetros onde são armazenados os pontos que formam o *simplex*. O número de chamadas à função objetivo cresce quadraticamente, i.e., complexidade $O(n^{2.11})$ (Nelder e Mead, 1965).

Por outro lado, o *HJD* é computacionalmente de menor custo. Foi encontrado, empiricamente que o número de chamadas à função objetivo cresce linearmente, i.e., complexidade $O(n)$ (Hooke e Jeeves, 1961). Uma boa coletânea de experimentos comparando esses dois e outros métodos pode ser encontrada em (Schwefel, 1995).

4.4.2 Resultados computacionais

O ECS foi codificado em C ANSI e executado em uma arquitetura *IA-32, com processador AMD Athlon 1.67 GHz e memória de 512 MB*. O tamanho da população variou com valores em $\{10, 30, 100\}$, dependendo do número de variáveis independentes da função objetivo. O número máximo de *clusters* $\mathcal{MC}$ foi ajustado para 20 para todas as funções-teste.

O parâmetro blx_α, referente ao cruzamento BLX, foi fixado em 25%. A mutação não-uniforme, por ter um papel importante na diversificação populacional do *ECS*, foi mantida constante em 10%. Pode-se dizer que é uma taxa relativamente alta de mutação que, em geral, fica em torno de 1% em AG's típicos (Goldberg, 1989).

Os experimentos foram condensados em três seções, de acordo com seus respectivos objetivos. Na primeira seção, são apresentados resultados que mostram como se comporta o método. Na segunda seção, o foco é sobre os diferentes métodos de assimilação. E por fim, na terceira seção de experimentos, o *ECS* foi comparado com outros métodos encontrados na literatura para avaliar a sua competitividade (Oliveira e Lorena, 2004).

4.4.2.1 Comportamento geral do ECS

Este primeiro conjunto de experimentos tem o objetivo de mostrar o comportamento do *ECS*, especialmente no que diz respeito a interação entre o processo evolutivo e o agrupamento iterativo. Basicamente, são observáveis, nos gráficos que se seguem, como a população converge para as regiões mais promissoras do espaço de busca e em que pontos dessas regiões tendem a se localizar os centros de *clusters*, identificados ao longo das gerações.

Duas funções de características diferentes foram escolhidas para exemplificar o comportamento do ECS. A função de *Schwefel* é altamente multimodal com variáveis definidas entre $[-500, 500]$. A função de *Rosenbrock* é unimodal, mas apresenta um vale estreito e de pouca inclinação onde, em geral, se concentra a grande maioria da população.

Na Figura 4.5, são mostrados quatro mapas de contorno em duas dimensões da função de *Schwefel*. As linhas de cores verde e azul representam regiões com menor valor

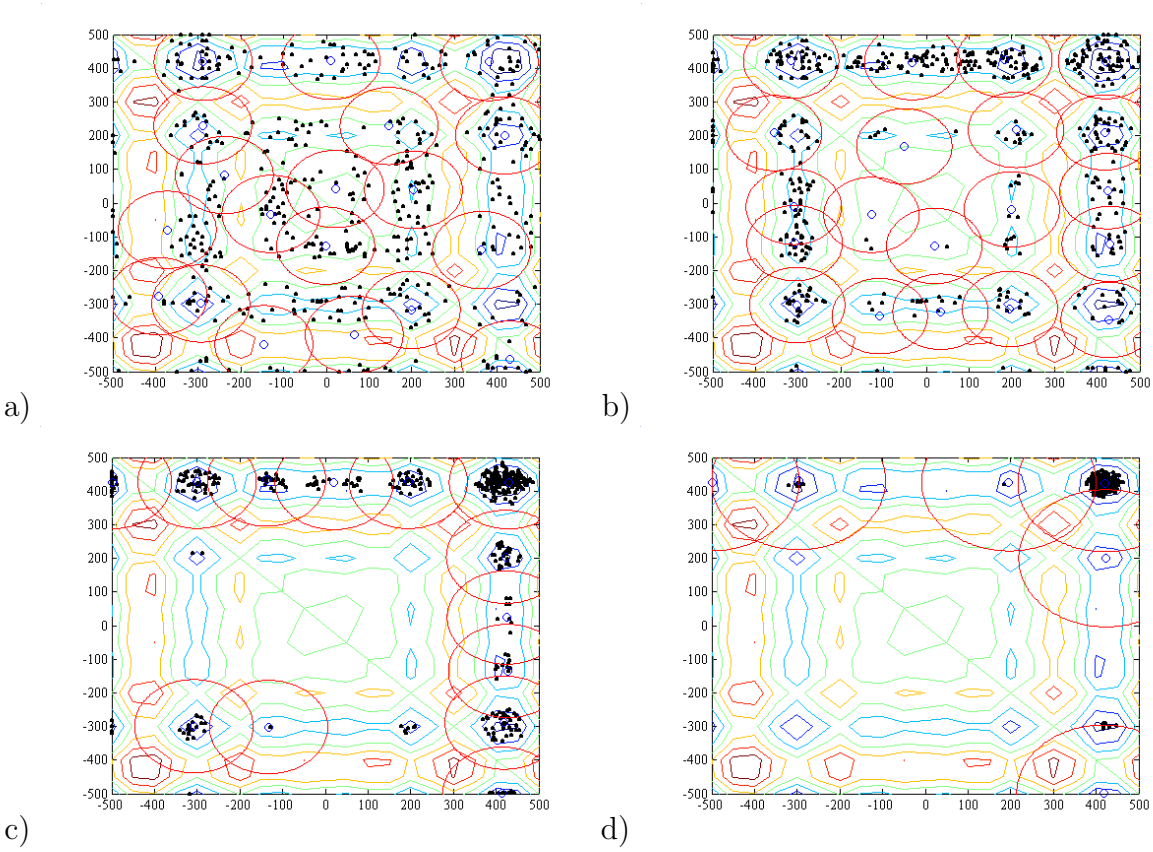


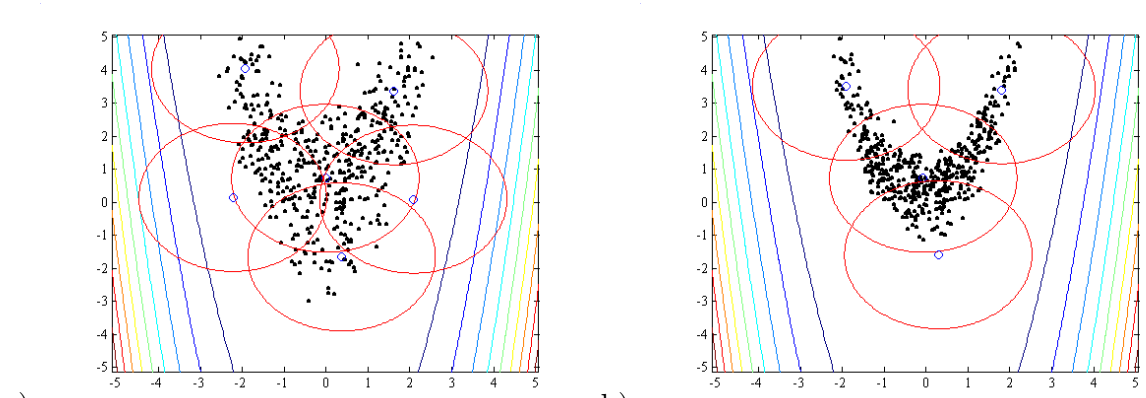
FIGURA 4.5 – Mapas de contorno da função de *Schwefel* após a)2, b)7, c)12 e d)17 gerações do *ECS*.

de função objetivo. O mínimo global de *Schwefel* bidimensional é aproximadamente $x^* = \{420, 420\}$. Cada mapa reflete a localização dos indivíduos, dos centros com seus respectivos diâmetros ao final das gerações 2, 7, 12 e 17. Cada geração, nesse experimento, equivale a seleção de $\mathcal{NS} = 200$ indivíduos.

Um experimento similar é mostrado na Figura 4.6. Desta vez são dois mapas de contorno em duas dimensões da função de *Rosenbrock*. O mínimo global de *Rosenbrock* bidimensional é exatamente $x^* = \{1, 1\}$. Cada mapa reflete a localização dos indivíduos, dos centros com seus respectivos diâmetros ao final das gerações 2 e 7.

Pode-se observar que os *clusters* permanecem uniformemente espalhados, enquadrando as regiões de busca onde se aglomeram indivíduos. Eventualmente, algumas regiões ficam "descobertas" por falta de *clusters* disponíveis. Entretanto, as regiões com maior densidade de indivíduos atraem os centros dos *clusters* mais próximos.

O últimos mapas de contorno, nas figuras anteriores, mostram a convergência da população para poucas regiões com conseqüente redução no número de *clusters*. Quando



a) b)

FIGURA 4.6 – Mapas de contorno da função de *Rosenbrock* em a)2 e b)7 gerações do *ECS*.

isso ocorre, os raios de todos os *clusters* são aumentados, refletindo a redução de $|C_t|$ na equação 4.5. A idéia por trás desse comportamento é aumentar a área de pertinência, fazendo com que, nas últimas gerações, os poucos *clusters* restantes tenham suas regiões mais inspecionadas pelo componente AO.

4.4.2.2 Desempenho de diferentes métodos de assimilação

Este segundo conjunto de experimentos foi realizado utilizando um número mais representativo de funções-teste visando aferir o desempenho dos três métodos de assimilação propostos neste trabalho: simples, por recombinação e por caminho. Dentro de cada um desses métodos, atribuiu-se diferentes valores aos parâmetros de desempenho de cada um deles. Ao todo foram 8 diferentes versões de assimilação testadas em 20 execuções com cada uma.

Em cada execução, foi aferido, em intervalos regulares de CFO, o *fitness* do melhor indivíduo até então obtido. A média do melhor *fitness* tomada a cada um desses intervalos constitui-se a trilha de convergência do algoritmo. Os gráficos de convergência foram postos em escala logarítmica para facilitar a visualização do desempenho final de cada versão do *ECS*. O eixo das ordenadas representa a média da melhor solução tomada a cada 10^3 CFO (eixo das abscissas).

O *ECS* com assimilação simples foi testado com os percentuais $\beta = \{5\%, 25\%\}$, usados no deslocamento do centro em direção ao ponto assimilado. Valores superiores a 25% deixam os centros instáveis, na medida em que qualquer ponto mais afastado, durante a assimilação, pode atraí-los consideravelmente.

O *ECS* com assimilação por recombinação foi testado com parâmetro $blx_\alpha = \{5\%, 25\%\}$.

Esses valores de blx_α influenciam a espansão do hiperplano onde é gerado o novo centro durante o processo de assimilação. Dessa forma, $blx_\alpha = 5\%$ significa uma expansão relativamente pequena, impondo também um caráter mais conservador à assimilação.

O *ECS* com assimilação por caminho foi testado com dois conjuntos de η escalares distintos uniformemente distribuídos no intervalo $]0, 1[$. O primeiro conjunto é formado pelos escalares $\{0,1; 0,2; 0,3; \ldots 0,9\}$ ($\eta = 9$) e o segundo por $\{0,25; 0,5; 0,75\}$ ($\eta = 3$). A razão de distribuição desses escalares é, respectivamente, 10% e 25% do intervalo. Esses dois valores definem o tamanho do passo utilizado para fazer amostragens na trajetória entre o centro e o ponto assimilado.

A assimilação por caminho equivale a η assimilações simples com o adendo que cada uma delas é avaliada e o centro vai se deslocar para a melhor solução encontrada na trajetória. Quanto mais soluções amostradas, mais CFO. O centro passa a representar a melhor solução encontrada no *cluster*.

Os valores 10% e 25% são utilizados nos gráficos que se seguem para representar os 2 conjuntos de β. O valor de 5% chegou a ser testado, mas gerou excessivas CFO. O percentual de 50%, que faz apenas uma amostra na trajetória, equivale a uma assimilação simples com $\beta = 50\%$ e por isso não foi testado.

Um outro aspecto que foi investigado diz respeito a possibilidade de utilização da assimilação por caminho como único método de intensificação, eliminando-se a necessidade de implementação de uma busca local no componente AO. Neste contexto, a atividade de um dado *cluster* induziria sucessivas assimilações com conseqüentes avaliações de soluções vizinhas. Processo similar tem sido usado em aplicações que usam *path-relinking* como método para intensificação de busca em torno de soluções de referência (Glover, 1998). Para este teste, foram desenvolvidas outras duas versões do *ECS* com assimilação por caminho, com os mesmos conjuntos de β, mas sem a busca local baseada no algoritmo do *Hooke e Jeeves*.

A Figura 4.7 mostra a convergência do *ECS* para a função de *Griewank* (*Gri*) com $n = 100$ variáveis. Essa função-teste é multimodal e, como as demais, não-linear. *Gri* apresenta pequenos e numerosos mínimos locais em torno do mínimo global, além de uma intermediária dependência entre as variáveis (Digalakis e Margaritis, 2002).

Como pode ser observado na Figura 4.7, até 21×10^3 CFO, os métodos que não empregam busca local apresentam uma convergência melhor que as demais. Entretanto, a partir daí, prevalece o chamado *efeito escada*, causado por uma brusca melhoria na avaliação

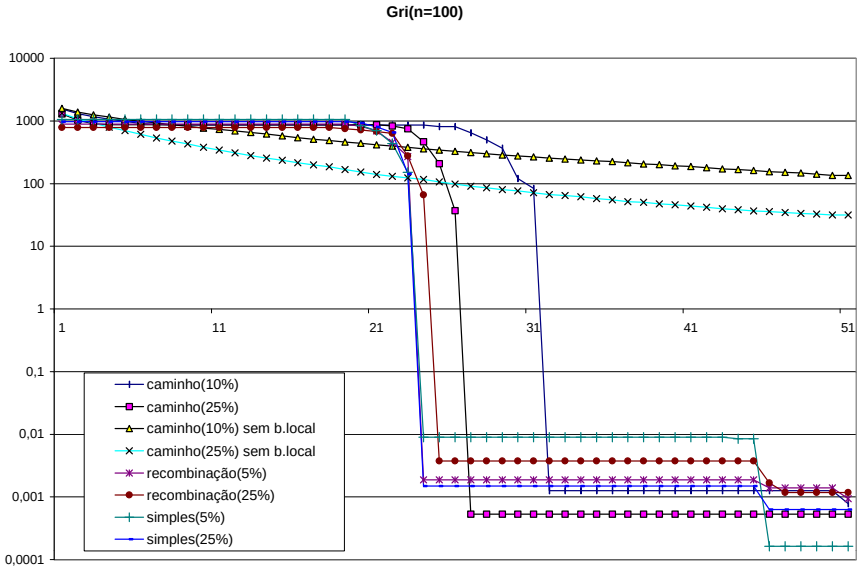


FIGURA 4.7 – Convergência para a função de *Griewank*.

do melhor indivíduo. O componente AO, empregando busca local, é o responsável por esse efeito.

No gráfico em escala logarítmica, a *escada* é percebida quando o *fitness* do melhor indivíduo se aproxima de 0. Entretanto, o processo de sucessivas melhorias bruscas ocorre várias vezes ao longo da evolução, dependendo da detecção de regiões promissoras. Como o gráfico de convergência espelha a média do *fitness* do melhor indivíduo nas 20 execuções, a queda brusca nessa média corresponde a uma melhoria em todos os melhores indivíduos dessas 20 execuções mais ou menos nos mesmos instantes de evolução (entre 21 e 32×10^3 CFO). Pode-se dizer que o *ECS* com busca local apresenta um certo padrão de comportamento.

O *ECS* com assimilação por caminho com passo de 25%, como pode ser observado na Figura 4.7, foi a primeira das versões a conseguir média abaixo de $0,001$. Todavia, com a assimilação simples a 5%, ao final de 51×10^3 CFO, o *ECS* obteve a melhor média para *Gri*. Por outro lado, as versões sem busca local (10% e 25%) não convergiram para médias satisfatórias.

A Figura 4.8 mostra a convergência do *ECS* para a função de *Rastrigin* (*Ras*) com $n = 20$ variáveis. Essa função-teste também é multimodal, com vários mínimos locais, mas com nenhuma dependência entre as variáveis (Digalakis e Margaritis, 2002). Como pode ser observado na figura, o efeito escada é mais discreto, mas ainda é suficiente para apresentar um padrão de convergência melhor nas versões com busca local, especialmente na versão

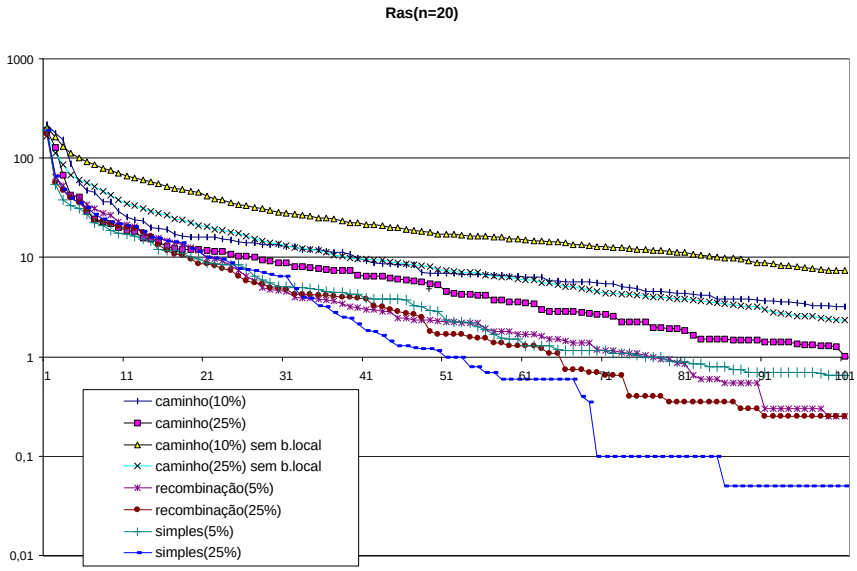


FIGURA 4.8 – Convergência para a função de *Rastrigin*.

com assimilação simples com 25%, notadamente a melhor para *Ras*. Outra observação é o fraco desempenho de todas as versões utilizando assimilação por caminho (com e sem busca local).

A Figura 4.9 mostra a convergência do *ECS* para a função de *Rosenbrock* (*Ros*) com $n = 50$ variáveis. Essa função-teste é unimodal e com uma forte dependência entre as variáveis (Digalakis e Margaritis, 2002). Como pode ser observado na figura, até 41×10^3 CFO, a assimilação por caminho sem busca local apresentou convergência melhor que os demais. Entretanto, a busca local foi determinante para o melhor desempenho final do *ECS* com assimilação por recombinação com 5% de expansão, o único a obter uma média de $0,001$.

A Figura 4.10 mostra a convergência do *ECS* para a função de *Schwefel* (*Sch*) com $n = 30$ variáveis. Essa função-teste é considerada um pouco mais fácil que *Ras*. Entretanto, com $n = 30$, *Sch* foi uma que apresentou uma significativa dificuldade para o *ECS*. Apenas em 14 das 20 execuções, o *ECS* obteve uma solução em torno de $0,001$.

Considerando todas as 20 execuções, a assimilação por caminho com passo de 25% (com ou sem busca local) apresentou desempenho destacado, como pode ser observado na Figura 4.10. Entretanto, analisando apenas as 14 melhores execuções de cada método, todos têm desempenhos próximos, com ligeira vantagem para a assimilação simples 5% e por caminho com passo de 10% (a primeira a apresentar a média abaixo de $0,001$).

A Figura 4.11 mostra a convergência do *ECS* nessas 14 melhores execuções. Nessa

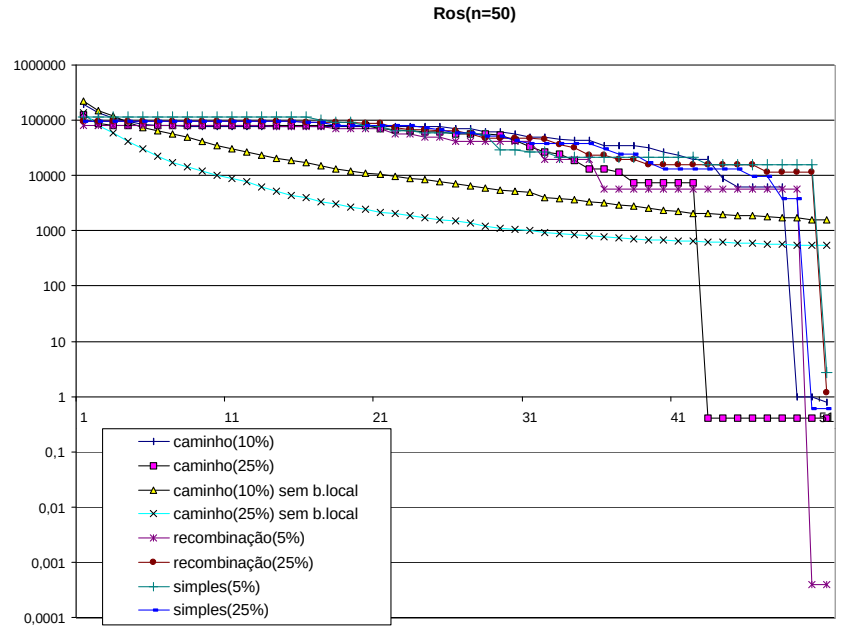


FIGURA 4.9 – Convergência para a função de *Rosenbrock*.

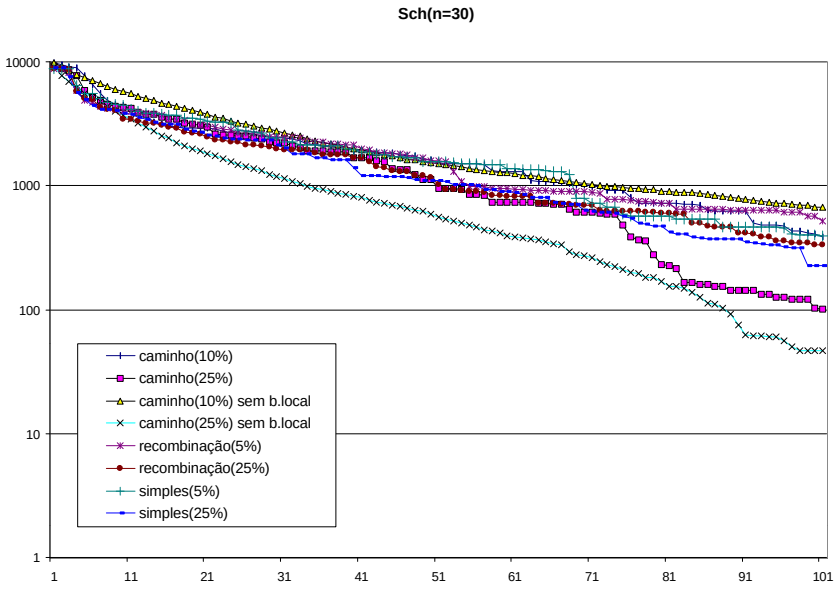


FIGURA 4.10 – Convergência para a função de *Schwefel*.

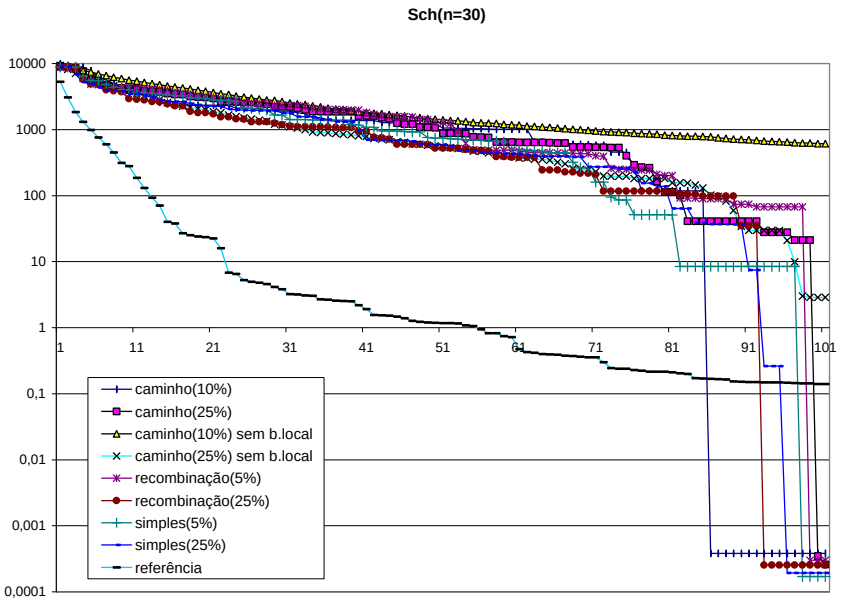


FIGURA 4.11 – Convergência para a função de *Schwefel* (14 melhores execuções).

mesma figura, é mostrado ainda o desempenho de um algoritmo genético padrão, sem busca local, codificado em real (AGCR), como uma referência de convergência. Pode ser observado que ele apresenta uma trilha de convergência suave, consistente e com uma boa aproximação do ótimo em 0.

Por fazer menos CFO, o AGCR superou o *ECS* com assimilação por caminho para *Schwefel*. Funções altamente multimodais, apresentando uma disposição uniforme de mínimos locais bem avaliados, como é o caso de *Sch*, fornecem condições a um AG padrão de manter uma população diversificada por muito mais gerações. Dessa forma, eles garantem um desempenho razoável que não obteriam com funções do tipo *Rosenbrock*, a qual apresenta uma única região bem avaliada (um vale) onde os melhores indivíduos se concentram, com pouca diversidade.

Pode-se dizer que muitas dessas funções-teste podem ser resolvidas satisfatoriamente por AG's padrão. Apenas em casos onde ocorra epistasia ou ocorra um número suficientemente grande de variáveis, é justificável desenvolver-se algoritmos evolutivos mais elaborados.

Ao final deste conjunto de testes, pode-se concluir que a implementação do *path-relinking* utilizada na assimilação por caminho, por si só, não foi suficiente para imprimir uma busca local satisfatória no *ECS*. Outra observação a ser ressaltada é que não houve um método de assimilação vencedor em todas as funções-teste usadas. Em vista disso, foi

escolhida a assimilação simples (5%) para um novo conjunto de testes com o objetivo de mostrar a competitividade do *ECS* nesse tipo de problema.

4.4.2.3 Comparação com outros métodos

Nestes experimentos, o ECS é comparado com outros enfoques encontrados na literatura. Inicialmente, o Genocop III (Michalewicz, 1996) e o *OptQuest Callable Library* (*OCL*) (Laguna e Martí, 2002) são usados na comparação.

Genocop III é a terceira versão de um algoritmo genético projetado para procurar por soluções ótimas em problemas de otimização com variáveis contínuas, com restrições lineares e não -lineares. O *OCL* é um software comercial para otimização de sistemas complexos baseado na busca *scatter* (Glover, 1998). Ambos os enfoques foram executados usando valores de parâmetros recomendados e seus resultados foram publicados em (Laguna e Martí, 2002).

Os resultados apresentados na Tabela 4.1 foram obtidos em 20 tentativas, permitindo ao ECS executar até 10.000 chamadas à função objetivo, da mesma forma em que o Genocop III e o *OCL* foram testados em (Laguna e Martí, 2002). A média das melhores soluções encontradas (MSE) foi considerada nesta comparação. A média dos tempos de execução (TE), em segundos, é somente ilustrativa, devido às plataformas de execução não serem as mesmas. Os valores em **negrito** indicam as melhores médias das soluções encontradas em cada função-teste. Como pode ser observado, o ECS obteve as melhores médias de solução em duas funções-teste (*Sphere* e *Rosenbrock*), enquanto o *OCL* e Genocop III obtiveram melhores médias em *Rastrigin* e *Schwefel*, respectivamente.

TABELA 4.1 – Comparação entre ECS, *OCL* e Genocop III.

		ECS		OCL		Genocop III	
Função	n	MSE	TE	MSE	TE	MSE	TE
Ackley	50	0,000	0,181	0,000	16,800	0,000	13,400
Ackley	100	0,000	0,374	0,000	103,600	0,000	46,600
Sphere	100	**0,000**	0,128	2,419	60,300	1114,451	43,700
Griewank	20	0,000	0,123	0,000	3,800	1,076	2,600
Rastrigin	10	1,087	0,036	**0,000**	4,500	1,026	0,900
Rastrigin	20	10,129	0,063	**0,000**	6,300	10,508	2,500
Rosenbrock	6	**0,002**	0,065	5,950	6,300	273,309	0,800
Rosenbrock	8	**0,000**	0,077	0,484	3,200	5,601	0,800
Rosenbrock	20	**0,003**	0,022	5,600	6,900	7,685	2,800
Schwefel	10	118,160	0,042	844,069	1,800	**1,387**	0,800
Schwefel	20	1360,397	0,047	1506,067	2,400	**134,491**	2,100

O ECS é agora comparado com um outro enfoque que trabalha com a mesma idéia de detecção de regiões promissoras: o Algoritmo Contínuo Híbrido (*Continuous Hybrid Algorithm-CHA*), descrito na seção 4.1. Os resultados apresentados pelo *CHA* foram tirados de (Chelouah e Siarry, 2003), onde seus autores trabalharam com uma série de funções $n-$dimensionais. As mais desafiadoras delas são usadas nesta comparação.

Os resultados apresentados na Tabela 4.2 foram obtidos permitindo ao ECS executar até 100.000 avaliações de função objetivo em cada uma das 20 tentativas em que se constituiu este experimento. Não há informação sobre um limite semelhante usado pelo *CHA*. A média das diferenças (*gap*) entre a solução encontrada e a solução mínima esperada, a média das chamadas à função objetivo (CFO) e o percentual de sucesso (PS) foram considerados nesta comparação.

Nos experimentos do ECS, o PS reflete o número de tentativas que alcançaram um *gap* mínimo de $0,001$. O PS dos experimentos realizados pelo *CHA* não foram calculados de uma forma clássica. De acordo com seus autores, tal taxa considera progresso em termos de melhor solução, desde a primeira geração (Chelouah e Siarry, 2003).

Como pode ser observado, o ECS parece ter desempenho melhor que o CHA nas funções mostradas na Tabela 4.2, exceto pela função *Zakharov*, para a qual o ECS não encontrou a melhor solução conhecida. É sabido que, em duas dimensões, *Zakharov* é unimodal com o mínimo global situado na borda de uma larga planície (veja Figura 2.5c). Não foi encontrada uma razão que justificasse esse fraco desempenho do ECS para *Zakharov*.

Para a função *Shekel*, embora o ECS tenha encontrado melhores *gaps*, o percentual de sucesso PS não foi tão bom quanto o obtido pelo *CHA*. Os valores em **negrito** indicam em qual aspecto o ECS foi pior que o *CHA*.

Considerando a média global $\overline{MSE}$ de todas as melhores soluções encontradas na Tabela 4.1, o *ECS* obteve $\overline{MSE} = 135,43$ contra $\overline{MSE} = 214,96$ do *OCL* e $\overline{MSE} = 140,86$ do Genocop III. Com relação ao experimento da Tabela 4.2, a média global do *ECS* não foi melhor que a média global do *CHA* em virtude de seu fraco desempenho para a função de *Zakharov* ($n = 50$). Entretanto, desconsiderando essa função, o *ECS* alcançou as médias globais $\overline{gap} = 0,0007$, $\overline{CFO} = 14.950,11$ e $\overline{PS} = 95\%$. O *CHA*, por sua vez, alcançou $\overline{gap} = 0,0071$, $\overline{CFO} = 22.730,88$ e $\overline{PS} = 91\%$. Considerando a qualidade das soluções em todas as funções-testes nas duas tabelas, perfazendo 21 casos no total, o *ECS* foi superior em 12 casos e inferior em 6.

Outros resultados obtidos pelo ECS são mostrados na Tabela 4.3. O *gap* mínimo de $0,001$

TABELA 4.2 – Comparação entre ECS e *CHA*.

				ECS				*CHA*
Função	n	TE	*gap*	CFO	PS	gap	CFO	PS
Eason	2	0,002	0,00060	593,5	100	0,001	952,0	100
Goldstein	2	0,001	0,00060	**345,4**	100	0,001	259,0	100
Hartman	6	0,003	0,00000	633,9	100	0,008	930,0	100
Rosenbrock	5	0,007	0,00040	2561,7	100	0,018	3290,0	100
Rosenbrock	10	0,023	0,00005	8979,5	100	0,008	14563,0	83
Rosenbrock	50	0,049	0,00015	32780,6	100	0,005	55356,0	79
Rosenbrock	100	0,286	0,00444	85821,0	80	0,008	124302,0	72
Shekel	4	0,003	0,00007	506,8	75	0,015	635,0	**85**
Zakharov	10	0,004	0,00050	2328,6	100	**1e-6**	4291,0	100
Zakharov	50	0,153	33,75020	100040,6	0	**1e-5**	75520,0	100

foi alcançado para todas as funções-teste em algumas ou todas as 20 tentativas. O pior desempenho ficou para a função *Michalewicz* e *Langerman*, com PS em torno de 65%).

TABELA 4.3 – Resultados do ECS para outras funções-teste.

Function	var	TE	gap	CFO	PS
Griewank	50	0,053	0,00010	5024,550	100
Griewank	100	0,432	0,00000	24344,450	100
Langerman	5	0,023	0,00000	5047,684	95
Langerman	10	0,075	0,00000	17686,692	65
Michalewicz	5	0,054	0,00035	12869,550	100
Michalewicz	10	0,222	0,00038	37671,923	65
Rastrigin	10	0,100	0,00060	26379,950	100
Rastrigin	20	0,339	0,00078	71952,667	90
Schwefel	20	0,211	0,00035	39987,950	100
Schwefel	30	0,591	0,00029	90853,429	70

É significativo o ganho, em termos de desempenho, do *ECS* com relação ao TPH, apresentado no Capítulo 3. Para algumas funções-teste, o enfoque $\mathcal{ATP}$ não obteve soluções satisfatórias. A busca local é mais racionalmente aplicada no *ECS* e isso se reflete no número de CFO e, conseqüentemente, no tempo de execução. Nas seções seguintes, o *ECS* é aplicado aos problemas de seqüenciamento de padrões e novas comparações com os enfoques baseados em TPH são realizadas.

4.5 Aplicação para problemas de seqüenciamento de padrões

O potencial do *ECS* foi testado para o Problema de Leiaute de Matriz-Porta (*Gate Matrix Layout Problem - GMLP*). A instância *GMLP*, conhecida como *w4*, foi escolhida por ser a maior encontrada na literatura. Nesta seção, são descritos os detalhes da implementação do *ECS* e os resultados obtidos.

4.5.1 Implementação

As principais diferenças entre esta e a implementação do ECS para otimização numérica (ECSno), anteriormente apresentada, são:

a) a codificação e avaliação do indivíduo: similar à empregada no $\mathcal{ATP}$;

b) busca local: heurística 2-Opt.

c) métrica de distância: heurística 2-Troca;

d) tipos de assimilação: por recombinação e por caminho.

Uma mutação do tipo 2-Troca (onde um movimento de troca de padrões é efetuado ao acaso), a seleção com pressão auto-adaptativa e o cruzamento *BOX* fecham o conjunto de operadores evolutivos do componente AE.

O componente AA realiza as mesmas operações da aplicação anterior. O componente AO emprega a busca local 2-Opt com limite de m vizinhanças, similar à mutação empregada no TPH (veja seção 3.4.7).

O *ECS* para seqüenciamento de padrões (ECSps) utilizou a assimilação por recombinação e por caminho. Foram construídas duas versões que diferem unicamente pelos métodos de assimilação empregados.

A assimilação por recombinação é implementada através do cruzamento *BOX*, o mesmo empregado na seção 3.4.5. Ressalta-se mais uma vez que, no caso da assimilação por recombinação, não há o compromisso do centro de *cluster* ser uma solução de qualidade. Portanto, a busca local é requerida para que o ECS seja efetivo com esse tipo de assimilação.

4.5.1.1 Múltiplas paisagens de aptidão

O agrupamento iterativo utiliza uma métrica de distância que deve refletir o esforço para se transformar uma solução em outra, considerando a paisagem de aptidão induzida por uma dada heurística de busca.

Pode-se dizer que os AG's trabalham com várias paisagens de aptidão, uma para cada operador genético (Jones, 1995). No caso específico dos algoritmos evolutivos híbridos, somam-se a essas paisagens, aquelas induzidas pelas heurísticas de busca local.

O ECSps utiliza a métrica 2-Troca para calcular a distância entre duas soluções no processo de agrupamento. Além disso, a 2-Troca também é usada para realizar amostragens do subespaço de busca entre um dado indivíduo selecionado e o centro do *cluster*, durante a assimilação por caminho.

De certa forma, o aspecto multi-paisagens de aptidão, presente no *ECS*, pode vir a ser benéfico para o processo de busca, pois possibilita que se obtenha uma solução ótima através de vários movimentos heurísticos diferentes.

A opção pela heurística 2-Troca como métrica de distância, entretanto, é muito mais em função da impossibilidade de se gerar soluções intermediárias entre duas outras usando movimentos 2-Opt em tempo computacional satisfatório (Linhares, 2002).

4.5.1.2 Assimilação por movimentos 2-Troca

A assimilação por caminho depende da métrica de distância empregada. Quanto mais distantes, mais soluções intermediárias existem entre duas soluções. O processo de amostragem, dependendo do número de variáveis do problema, pode ser significativamente custoso, uma vez que cada solução amostrada deve ser avaliada pela função objetivo.

Na Tabela 4.4, é mostrada uma trajetória 2-Troca completa entre duas soluções, c_i e s_k. Há a necessidade de $I-1=8$ comparações, onde I é o número de padrões na permutação (número de variáveis).

TABELA 4.4 – Exemplo de trajetória completa entre um centro c_i e um indivíduo s_k.

c_i=	1	2	3	4	5	6	7	8	9	comparação	troca	avaliação
1)	4	2	3	1	5	6	7	8	9	1	1	1
2)	4	8	3	1	5	6	7	2	9	1	1	1
3)	4	8	5	1	3	6	7	2	9	1	1	1
4)	4	8	5	9	3	6	7	2	1	1	1	1
5)	4	8	5	9	1	6	7	2	3	1	1	1
6)	4	8	5	9	1	7	6	2	3	1	1	1
7)	4	8	5	9	1	7	6	2	3	1		
8)	4	8	5	9	1	7	6	2	3	1		
s_k=	4	8	5	9	1	7	6	2	3	8	6	6

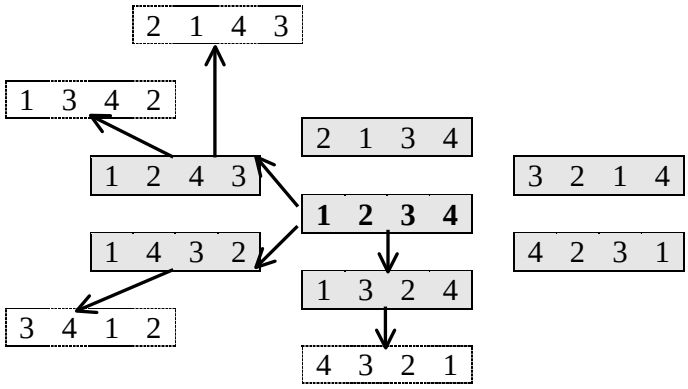


FIGURA 4.12 – Exemplos de trajetórias 2-Troca em permutação com quatro padrões.

Cada comparação corresponde a uma iteração no algoritmo de assimilação. Em cada iteração, ainda pode ocorrer uma troca entre padrões e uma avaliação de função objetivo. O centro vai ser deslocado para a melhor solução avaliada nessa trajetória.

Nas duas últimas iterações, não mais havia diferença entre a amostra corrente e a solução assimilada, portanto não foi necessário haver troca entre padrões, e nem tampouco avaliações de função objetivo. Efetivamente, ocorreram 6 trocas de padrões e, conseqüentemente, houve a necessidade de 6 avaliações de função objetivo.

A distância $\wp^{2-Troca}(c_i, s_k)$ não necessariamente é 6. Pela equação 2.4, $\wp$ é a menor distância heurística entre duas soluções. Poderia haver uma outra seqüência menor que 6 conduzindo c_i a s_k.

Dada a necessidade do *ECS* requerer o cálculo de $\wp$ para associar cada indivíduo selecionado ao seu respectivo centro de pertinência durante o processo de agrupamento, $\wp$ é estimado considerando o número de padrões que aparecem em posições diferentes de cada permutação (variáveis com valores diferentes). O número de padrões em posições diferentes é ainda subtraído de 1, pois, mesmo que todos os I padrões estivessem em posições diferentes nas duas permutações, seriam geradas no máximo $I - 1$ soluções intermediárias.

A Figura 4.12 mostra algumas trajetórias 2-Troca que podem ocorrer a partir do centro $\{1, 2, 3, 4\}$. As soluções em caixas brancas são aquelas a serem assimiladas. As soluções que aparecem em caixas cinzas pertencem a vizinhança $\varphi^{2-Troca}$ do centro.

Observa-se que as trocas de padrões não necessariamente ocorrem sempre da esquerda para a direita, como no exemplo da Tabela 4.4. A troca de 2 padrões diferentes a cada iteração é o que determina os movimentos na paisagem 2-Troca. O algoritmo

implementado na assimilação por caminho escolhe pares de pontos aleatórios para efetuar as trocas de padrões e limita o número de trocas que ocorrem a cada assimilação. Diferentemente da assimilação por caminho do ECSno, esta é não-determinística.

A assimilação por caminho pode ser utilizada como único mecanismo de intensificação, sem a necessidade de se aplicar algum tipo de busca local sobre as regiões promissoras detectadas. Como já foi mencionado anteriormente, os centros formam um conjunto de soluções de alta qualidade e representativas. Nesse caso, o processo de assimilação por caminho é similar a uma intensificação de busca em torno dessas soluções de referência.

4.5.1.3 Volume do *cluster*

O volume é usado para definir a área de pertinência dos *cluster*. O raio do *cluster* está relacionado com a métrica de distância empregada, no caso a heurística 2-Troca. Pode-se estabelecer uma forma conceitualmente similar à equação 4.5 para cálculo de r_t:

$$r_t = \left\lceil \frac{I-1}{2\sqrt[I]{|C_t|}} \right\rceil \tag{4.11}$$

onde $|C_t|$ é o atual número de *clusters*, I é o número de padrões do problema, utilizado na estimativa da maior distância 2-Troca encontrada no espaço de busca.

Examinando-se a equação 4.11, para $|C_t| = 20$ e $20 < I < 140$ (valores usualmente encontrados nas instâncias utilizadas neste trabalho), calcula-se que $r_t \simeq 0,5I$. Em outras palavras, um indivíduo pertence a um determinado *cluster* quando há uma coincidência de padrões em torno de meia permutação.

Entretanto, essa forma de calcular r_t não se mostrou eficiente pois requer uma quantidade razoável de coincidências entre duas permutações para que haja ativação de algum dos *clusters*. Sendo assim, há uma tendência de detecção de poucas regiões promissoras ao longo das gerações. Os melhores resultados foram obtidos com:

$$r_t = \lceil 0,9I \rceil \tag{4.12}$$

ou seja, um raio razoavelmente grande, pois requer apenas 10% de coincidência para que um indivíduo seja considerado similar a um dado *cluster*.

4.5.2 Resultados computacionais

O ECS para seqüenciamento de padrões foi codificado em C ANSI executado em uma arquitetura *IA-32, com processador AMD Athlon 1.67 GHz e memória de 512 MB*. Foram realizados experimentos com objetivo, primeiramente, de evidenciar a flexibilidade do método, uma vez que, para esta aplicação, foram necessários apenas ajustes nos parâmetros de desempenho do método. Além das modificações específicas para o tipo de problema apresentadas nas seções anteriores, não houve modificações nos algoritmos que compõem o núcleo do *ECS*.

Um segundo objetivo é mostrar que o método pode também ser competitivo para problemas de seqüenciamento com as escolhas que foram feitas em termos de métodos de assimilação, busca local, ajustes, codificação, etc. Para tanto, o *ECS* foi comparado com os algoritmos propostos no TPH.

Os parâmetros de desempenho foram ajustados como se segue. O tamanho da população foi 1200 indivíduos. O número de indivíduos selecionados a cada geração foi o mesmo usado para otimização numérica, ou seja, $\mathcal{NS} = 200$. A mutação 2-Troca foi mantida com taxa constante de 10%.

O número máximo de *clusters* $\mathcal{MC}$ foi ajustado para 20. A pressão de densidade que está relacionada com a sensibilidade para detecção de regiões promissoras foi reduzida para $\mathcal{PD} = 1$, bem inferior aos $2,5$ da aplicação anterior.

O número de amostras tomadas a cada assimilação por caminho foi 4: aproximadamente o mesmo usado no *ECSno*. Cada solução é gerada dentro da trajetória através de um movimento 2-Troca a partir da solução anterior. Os movimentos 2-Troca trocam duas posições aleatórias na permutação. Depois de 4 avaliações de função objetivo, a melhor solução é escolhida como centro do *cluster*.

Foi observado, ao longo dos experimentos, que o *ECS* para seqüenciamento de padrões difere do anterior em termos de número de *clusters* $|C_t|$ a cada geração. Na seqüência de mapas de contorno mostrada na Figura 4.5, referente a aplicação para otimização numérica do *ECS*, fica evidente que, com o decorrer do processo evolutivo, $|C_t|$ sempre decresce.

A Figura 4.13 mostra $|C_t|$ em função de t em uma execução típica do *ECS* para seqüenciamento de padrões. Apesar de iniciado com $|C_0| = \mathcal{MC} = 20$, após o processo de esfriamento de *clusters* com baixa densidade, o *ECS* não consegue manter 20 *clusters* nas primeiras gerações. Pode-se observar, por exemplo, que até a centésima geração, apenas

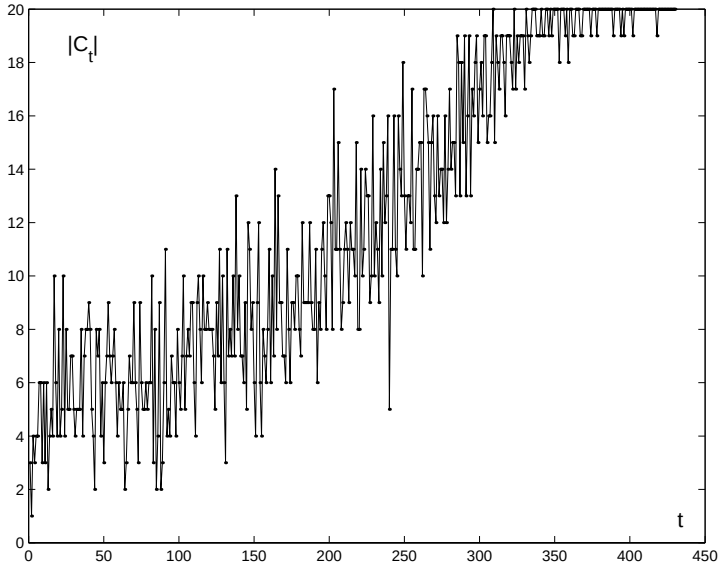


FIGURA 4.13 – Número de *clusters* $|C_t|$ ao longo das gerações t.

uma única vez o ECS conseguiu manter 10 *clusters*.

Esse comportamento deve-se ao tipo de métrica de distância utilizada. Pelo que foi definido em termos de raio, um indivíduo pode ser associado a um dado centro de *cluster* quando ocorrer um mínimo de 10% de coincidência entre eles.

Um número reduzido de *clusters* indica haver pouca coincidência devido a uma provável diversidade populacional. Os *clusters* são criados, mas a falta de atividade nas regiões enquadradas por eles faz com que sejam eliminados a cada geração. Todavia, no decorrer do processo evolutivo, a convergência em torno de regiões promissoras tende a aumentar o número de *clusters* em virtude do aumento de atividade nessas regiões. Observa-se ainda, na Figura 4.13, que somente após a tricentésima geração começa a ocorrer a saturação, i.e., $|C_t|$ atinge o limite de 20 *clusters*.

Naturalmente, perdurando a execução, com a continuação do processo de convergência, $|C_t|$ tende a decair novamente até que apenas um único perfil de indivíduo seja encontrado na população e assim $|C_t| \rightarrow 1$. A população relativamente grande, em torno dos 1200 indivíduos, e as altas taxas de mutação usadas colaboram para manter a diversidade populacional durante um número de gerações suficiente para que todas as regiões promissoras sejam devidamente exploradas.

A assimilação por recombinação foi testada, mas os resultados não foram bons, quando comparados aos obtidos com a assimilação por caminho. Apesar de ter sido encontrada uma única vez a solução ótima para a instância *w4* com apenas 166.835 CFO, a média

TABELA 4.5 – Comparação entre *ECS* e os enfoques de TPH para a instância *w4*.

	MSE	PS	CFO		MSE	PS	CFO
$\mathcal{ATP}^{2opt}$	$28,6 \pm 0,97$	2	8.488.438	$\mathcal{AGC}^{2opt}$	$28,0 \pm 0,82$	3	6.537.706
$\mathcal{ATP}^{fagg}$	$28,3 \pm 1,16$	2	9.330.802	*ECS*	$27,8 \pm 1,03$	5	1.104.348

das melhores soluções, obtidas em 10 execuções, ficou em torno de $30,4$ usando esse tipo de assimilação.

Na Tabela 4.5, é mostrada uma comparação entre os melhores enfoques baseados no TPH e o *ECS*, este último usando assimilação por caminho, em 10 execuções para a instância *GMLP w4*. Foi acrescida também a informação concernente à variabilidade dos resultados. A legenda PS, nesta tabela, se refere ao número de vezes em que a melhor solução foi encontrada.

O *ECS* apresentou um desempenho superior aos enfoques de *TPH*, especialmente, no que tange o número de CFO. O $\mathcal{AGC}^{2opt}$, que também emprega uma busca local 2-Opt, tende a realizar 6 vezes mais CFO que o *ECS*.

Uma importante estatística, que pôde ser colhida nestes experimentos, diz respeito a eficiência da busca local durante todo o processo evolutivo. O procedimento 2-Opt foi chamado $84,67$ vezes, em média, nas 10 execuções do *ECS*. Desse total, algo em torno de 28% encontraram soluções melhores do que a melhor solução até então encontrada. Em 5 execuções realizadas com o $\mathcal{AGC}^{2opt}$, esse percentual não passou dos 20% nas $102,2$ vezes (em média) em que o procedimento 2-Opt foi chamado.

4.6 Considerações finais

O *Evolutionary Clustering Search* (ECS) tenta localizar regiões promissoras através do agrupamentos de indivíduos em subespaços de busca com atividade acima de um suposto normal. Quando um *cluster* atinge uma certa densidade de indivíduos selecionados, seu centro é usado como ponto de partida para uma busca local.

Além disso, o próprio processo de agrupamento realiza operações de intensificação de busca, considerando os indivíduos agrupados e os centros de *clusters*. Esse processo é chamado de assimilação e permite o uso de algoritmos baseados em *path-relinking* (*PR*) nos quais trajetórias que levam a soluções de alta qualidade podem ser exploradas na busca por soluções melhores.

Foram construídas duas aplicações do ECS: uma para minimização de funções numéricas sem restrição e outra para problemas de seqüenciamento de padrões. Em ambas, o *ECS*

se mostrou competitivo, especialmente quando associado a mecanismos de busca local sobre os *clusters* promissores.

Uma vez que os resultados obtidos foram satisfatórios e comparáveis ao TPH, pode-se dizer que o *ECS* é suficientemente flexível para ser aplicado a problemas com espaços de busca contínuos e discretos.

4.6.1 Controle de redundância

Algumas questões relativa a desempenho são passíveis de consideração. O processo de agrupamento não tem um mecanismo direto de diversificação populacional. O *ECS* depende fortemente de altas taxas de mutação para manter a diversidade populacional. Em outras palavras, o processo evolutivo não sofre uma explícita intervenção do processo de agrupamento em termos de manutenção de indivíduos representativos de todo o espaço de busca.

A melhoria dos centros, que participam do processo de assimilação, indiretamente induz a geração de novos indivíduos bem avaliados e ao mesmo tempo representativos que têm grande chance de sobreviver e portanto de manter uma certa diversidade populacional. Mas esse mecanismo pode não ser suficiente para a manutenção de diversidade populacional.

O controle de redundância populacional, eliminando-se indivíduos em agrupamentos super-populosos, poderia ser realizado através do processo de agrupamento se este último fosse parte da etapa de atualização de novos indivíduos. Dessa forma, seria mais facilmente implementável uma espécie de filtro, o qual permitiria a atualização de parte dos indivíduos associados aos *clusters*.

Um algoritmo evolutivo geracional, com agrupamento na atualização de indivíduos chegou a ser implementado mas não obteve resultados satisfatórios. O processo de convergência aparentemente ficou prejudicado por esse filtro de indivíduos redundantes.

A questão relacionada a diversidade é pertinente tendo em vista que o *ECS* não obteve resultados satisfatórios para algumas funções-teste, como *Zakharov*. Além disso, aprimoramentos podem ser incorporados ao *ECS*, tornando-o mais apropriado a aplicações reais, encontradas em certas áreas da indústria que trabalham com milhares de variáveis. Em alguns contextos, não é tão importante a velocidade com que um método de otimização encontra uma solução satisfatória, mas sim o fato dele realmente encontrar soluções satisfatórias em um tempo aceitável.

4.6.2 Novos cenários

Existem fatores que influenciam a freqüência na qual soluções candidatas são amostradas e que podem causar *falsas* regiões promissoras. Tais fatores estão relacionados a codificação e operadores evolutivos utilizados. Além disso, algumas regiões promissoras podem não ser suficientemente amostradas para causar alguma suposição de que elas venham a ser promissoras. Por esse motivo, o *ECS* está sujeito a falhas ocasionadas por uma má distribuição de indivíduos no espaço de busca.

O processo evolutivo, que de certa forma dá suporte ao processo de agrupamento, pode ser a *causa da vitória* ou o *motivo da derrota*. O grande atrativo de um algoritmo evolutivo é a população de soluções que funciona como uma memória de soluções. Todavia, a população pode convergir prematuramente e arrastar com ela todos os *clusters*, perdendo-se completamente os pontos de referência para algumas regiões do espaço de busca.

Um argumento emergente é substituir o AE por uma outra metaheurística, como *GRASP*, por exemplo, capaz de *alimentar* o processo de assimilação com um grande número de soluções diferentes. Talvez tais soluções sequer precisem ser de qualidade. Os próprios *clusters* tendem a se constituir, ao longo do tempo, em soluções de referência. A geração contínua de soluções como suporte a um processo de busca é alvo de estudo no próximo capítulo. Por enquanto, neste trabalho, o *Evolutionary Clustering Search* apenas explorou uma parte *evolutiva* da busca através de agrupamentos. Outras abordagens estão previstas com o emergente **CS*.

Voltando ao *ECS*, uma proposta imediata de aplicação seria a construção de uma versão paralela na qual cada região promissora poderia ser, por exemplo, examinada por um outro algoritmo genético (ou uma outra heurística) trabalhando com uma nova população gerada dentro das fronteiras da região detectada.

Alocação dinâmica de regiões promissoras a diferentes processadores poderia ser organizada hierarquicamente, trabalhando em diferentes níveis de detalhamento do espaço de busca, sob a supervisão de um processador responsável pela coleta das melhores soluções encontradas em cada nível.

A principal dificuldade, no caso do *ECS* paralelo posto dessa forma, está na questão dinâmica do processo de detecção de regiões promissoras por genótipo. Espaços de busca com milhares de variáveis iriam requerer um sofisticado método de alocação dinâmica de regiões de busca a um número, em geral, fixo de máquinas paralelas. Todavia, regiões de busca assim alocadas é uma idéia apropriada a aplicações em sistemas de computação

em grade (Foster e Kesselman, 1998).

Com base nessas observações, o problema de detecção de regiões promissoras foi deslocado para o contexto de problemas que manipulam um grande número de variáveis, onde o processamento paralelo pode ser um forte aliado por possibilitar a construção de algoritmos mais adequados às longas simulações, necessárias em problemas mais complexos.

CAPÍTULO 5

DETECÇÃO DE REGIÕES PROMISSORAS POR AMBIENTES EVOLUTIVOS HETEROGÊNEOS

Como ficou evidenciado nos capítulos anteriores, algoritmos evolutivos híbridos implementados seqüencialmente são efetivos na solução de problemas em diferentes domínios. Entretanto, dependendo do número de variáveis envolvidas no processo de otimização, alguns problemas podem ocorrer.

O espaço de busca pode se tornar significativamente grande de tal forma que seja necessário grandes populações para uma uniforme amostragem. Populações grandes, por sua vez, podem tornar o tempo de espera (ou tempo de execução) consideravelmente longo para se obter soluções razoáveis. Além disso, tais populações, em geral, tendem a convergir para um padrão de genótipo subótimo do qual dificilmente conseguem escapar, mesmo com altas taxas de mutação (Rees e Koehler, 1999).

Um outro problema relacionado a quantidade de variáveis está mais relacionado aos algoritmos evolutivos híbridos (AEH's). Heurísticas de melhoria, tipicamente incorporadas aos AEH's, tendem a se tornar significativamente lentas, deixando o processo evolutivo significativamente lento.

Neste capítulo, o modelo hierárquico de competição justa (*Hierarchical Fair Competition - HFC*, proposto em (Hu et al., 2002), é empregado para separar indivíduos em subpopulações com diferentes perfis de aptidão (*fitness*). Através dessa estratificação, uma competição justa entre indivíduos com perfis diferentes dá sustentação a um mecanismo de detecção de regiões promissoras baseado em elitismo. Uma aplicação para otimização numérica sem restrição é implementada, acrescentando melhorias ao algoritmo evolutivo originalmente sugerido em (Hu et al., 2002).

5.1 Sugestões anteriores de paralelização

Nos capítulos anteriores foram consideradas algumas formas de paralelismo para adequar o TPH e o *ECS* às longas simulações, em geral, necessárias a problemas mais complexos. No caso do TPH, a separação de indivíduos em diferentes subpopulações, treinados com diferentes heurísticas, de certa forma, promoveria a evolução em diferentes paisagens de aptidão, paralelamente. No caso do *ECS*, regiões promissoras do espaço de busca seriam alocadas dinamicamente, sob demanda, a diferentes subpopulações que evoluiriam igualmente em paralelo.

Algoritmos evolutivos paralelos (AEP's) realmente oferecem novos dispositivos para superar as limitações das versões seqüenciais. Os AEP's não são apenas extensões dos AE's seqüenciais, mas se constituem verdadeiramente em um novo paradigma evolutivo que é capaz de desempenhar uma busca envolvendo novos operadores evolutivos, novos mecanismos de evolução, etc (Nowostawski e Poli, 1999).

Os AEP's podem explorar diferentes subespaços de busca, especializando cada subpopulação em determinada característica, como seria no caso do enfoque multi-heurístico paralelo do TPH. Alternativamente, promovendo a divisão do espaço de genótipos entre subpopulações, como é sugerido no caso do *ECS*.

Neste capítulo, um modelo de algoritmo evolutivo paralelo é empregado para separar indivíduos em subpopulações com diferentes perfis de aptidão, ou seja, uma decomposição de domínio de *fitness*. Através dessa estratificação, dois benefícios são alcançados: detecção de regiões promissoras e competição justa entre indivíduos.

O primeiro benefício é objeto de estudo deste trabalho. Regiões promissoras representadas por uma elite de indivíduos é tomada como base para intensificação de busca. A elite é assim considerada por estar dentro de uma faixa de aptidão considerada de alta qualidade. As demais faixas de aptidão são associadas a outras subpopulações que dão suporte à elite, provendo-a de material genético evoluído sob condições justas de competição.

A competição justa é um conceito associado ao modelo hierárquico de competição justa (*Hierarchical Fair Competition - HFC* (Hu et al., 2002). Nele, a população é estratificada hierarquicamente e cada indivíduo é atribuído a uma subpopulação, dependendo da faixa de *fitness* a que ele corresponder. A competição justa surge exatamente pelo fato de que apenas indivíduos com mesmo perfil de *fitness* competem entre si em suas respectivas subpopulações.

Alguns conceitos relativos a computação paralela são abordados na próxima seção para permitir uma melhor compreensão do trabalho apresentado neste capítulo.

5.2 Arquiteturas paralelas de memória distribuída

Máquinas paralelas tem se difundido largamente devido a sua relação custo-benefício favorável e à impossibilidade de se obter um desempenho computacional expressivo com um único processador (Kumar et al., 2003). Em termos de arquitetura, as máquinas paralelas dividem-se naquelas de memória compartilhada (ou multiprocessadores), em que todos os processadores acessam um único espaço de endereçamento de memória, e naquelas de

memória distribuída (ou multicomputadores/*clusters*), compostas por máquinas ou nós independentes com espaços de endereçamento disjuntos e interconectadas por uma rede de comunicação.

Correntemente, os supercomputadores mais rápidos são *clusters* compostos por nós multiprocessados e interconectados por redes de alta velocidade. Por outro lado, há anos os chamados *clusters Beowulf* vem se popularizando. *Clusters* são compostos por microcomputadores do tipo PC que utilizam o sistema operacional *Linux* e que são interconectados por uma rede padrão *Fast Ethernet* ou *Gigabit Ethernet*. Oferecem um desempenho computacional razoável a um custo baixo e, além disso, há de se considerar que os microcomputadores atuais dispõem de arquitetura avançada, em termos de processador, memória, barramento interno, etc. Some-se a isso *software* gratuito como compiladores, bibliotecas de comunicação e o próprio sistema operacional.

Os *clusters* são compostos tipicamente por máquinas fabricadas em larga escala, à diferença dos multiprocessadores, que apresentam um custo por processador bem mais elevado e tem limitações de escalabilidade (comumente pode-se ter até 4 ou 8 processadores por máquina).

Nos *clusters*, a necessidade de sincronização e a dependência de dados entre processadores exigem a comunicação entre os nós, uma vez que suas memórias são independentes. Isto é feito por meio de uma biblioteca de comunicação por troca de mensagens. A biblioteca de comunicação que vem sendo mais utilizada atualmente é a *Message Passing Interface (MPI)*, que provê um conjunto de rotinas prático, portável, eficiente, flexível e largamente difundido (Pacheco, 1996; Gropp et al., 1999).

5.3 Algoritmos evolutivos paralelos

A designação algoritmo evolutivo paralelo é bem genérica e compreende qualquer algoritmo evolutivo com um ou mais processos ocorrendo explicitamente em paralelo. Um AEP pode ser decomposto em diferentes tarefas, gerando assim, diversos processos a serem distribuídos entre múltiplos processadores (decomposição funcional). São exemplos de tarefas que podem ser distribuídas: avaliação, seleção e cruzamento de indivíduos. Também pode ser feita a decomposição em nível de problema (decomposição de domínio), como por exemplo, a divisão entre processadores do espaço de busca ou do espaço de *fitness* (Carmona, 1989).

Muito embora as primeiras experiências tenham visualizado tão somente o ganho de desempenho por paralelizar operações genéticas ocorrendo dentro de uma única evolução,

os AEP's logo se tornaram modelos muito mais elaborados, combinando múltiplas populações com alguma independência.

A cooperação entre subpopulações paralelas está fundamentada em novos operadores evolutivos, especificamente criados para promover o intercâmbio de indivíduos. A migração é um exemplo de operador específico para cooperação entre subpopulações que simula os processos migratórios observados na natureza. Da mesma forma, outros aspectos que igualmente influenciam a evolução podem ser melhor simulados através de paralelismo. Por exemplo, a competição e cooperação inter e intra-espécies, inclusive considerando questões normalmente simplificadas como a separação de grupos de indivíduos em regiões geográficas distintas e a influência das distâncias no processo de cruzamento (Falqueto et al., 2000).

A paralelização de algoritmos evolutivos requer algumas decisões, tais como:

a) por que paralelizar: ganhos em tempo de execução ou qualidade das soluções;

b) o que paralelizar: avaliação do indivíduo, operações genéticas, busca local, ou o processo evolutivo;

c) como paralelizar: modelos de paralelismo, formas de implementação.

Há casos em que um determinado algoritmo encontra soluções de qualidade com tempo de resposta alto. Em casos assim, operações ocorrendo em paralelo, como a avaliação de *fitness*, podem significar ganhos de tempo que por si só justifiquem a construção de uma versão paralela para tal algoritmo. Por outro lado, certos problemas podem ser melhor abordados utilizando argumentos tais como *dividir para conquistar*. O espaço de busca ou o espaço de objetivos (Coello Coello, 2000) podem ser focalizados por diferentes subpopulações, permitindo um ganho de qualidade nas soluções obtidas.

Neste trabalho, uma maior atenção é direcionada aos populares modelos de algoritmos evolutivos paralelos multi-populacionais[1]. Tais modelos podem combinar diferentes topologias de comunicação, políticas de migração, consolidando-se definitivamente como uma nova classe de algoritmos evolutivos, bem mais elaborados do que simplesmente algoritmos evolutivos implementado paralelamente.

5.3.1 Modelos paralelos multi-populacionais

O primeiros modelos paralelos procuraram explorar o conhecido paralelismo intrínseco dos AE's, distribuindo indivíduos entre processadores, mas sempre atrelados a um único

[1] O termo *modelos paralelos* é utilizado neste trabalho em simplificação *ao modelos de algoritmos evolutivos paralelos*.

processo evolutivo em uma única população. Do ponto de vista de computação paralela, pode-se visualizar tais AEP's como seguindo um modelo mestre-escravo, no qual diversos processadores escravos executam operações evolutivas em indivíduos ou conjuntos diferentes de indivíduos sendo coordenados pelo processador mestre (Nowostawski e Poli, 1999).

Os modelos multi-populacionais podem ter granularidade grossa (*coarse-grained*) ou granularidade fina (*fine-grained*). Em computação paralela, define-se granularidade como a razão entre a computação e comunicação. A granularidade fina caracteriza-se por um custo de comunicação elevado comparativamente ao custo de processamento relativo a cálculos, ao contrário da granularidade grossa (Nowostawski e Poli, 1999).

Considerando os AEP's, a computação refere-se ao processo evolutivo propriamente dito que ocorre em cada subpopulação, enquanto que a comunicação deve-se à troca de informação entre processadores, tipicamente, relativa à migração de indivíduos entre subpopulações.

Os algoritmos evolutivos com granularidade grossa, em geral, apresentam poucas subpopulações com muitos indivíduos (comparativamente a um GA equivalente), sendo que grande parcela do tempo de execução é gasto com o processo evolutivo e somente ocasionalmente há troca de indivíduos entre as subpopulações. Os algoritmos evolutivos com granularidade fina, tipicamente, apresentam muitas subpopulações com menos indivíduos e políticas de migração que implicam em muita comunicação (Nowostawski e Poli, 1999).

A granularidade influencia o tempo de execução do algoritmo. Algoritmos com granularidade fina são melhor implementados em máquinas paralelas de memória compartilhada (multiprocessadores), nas quais o custo de comunicação é representado pela contenção ao acesso da memória comum por parte dos processadores, devido a dependências de dados (indivíduos migrantes) e a necessidades de sincronismo (Kumar et al., 2003).

Os algoritmos com granularidade grossa, por outro lado, dado o baixo volume de comunicação, podem ser implementados também em máquinas paralelas de memória distribuída, arquiteturas com relação custo/benefício mais favorável do que as máquinas paralelas de memória compartilhadas (Kumar et al., 2003).

5.3.2 Políticas de migração

A política de migração pode ser completamente estabelecida através de alguns parâmetros essenciais ao bom desempenho de um AEP. Os principais parâmetros são

(Nowostawski e Poli, 1999):

a) a topologia que define as conexões entre subpopulações;

b) a freqüência com que são realizadas as migrações de indivíduos;

c) o número de indivíduos que são trocados a cada migração;

d) o critério de escolha dos imigrantes.

A topologia define restrições em termos de comunicação entre subpopulações e, portanto, possíveis rotas migratórias. Em algoritmos com granularidade fina, os indivíduos interagem com outros que estejam na mesma *vizinhança* definida dinâmica ou estaticamente através da topologia (Nowostawski e Poli, 1999).

As topologias mais usadas em AEP's são ilha, caminho-de-pedras (*stepping-stones*), grade, hipercubo, etc (Alba e Troya, 1999). A topologia ilha caracteriza-se por uma ampla difusão (*broadcast*) de indivíduos entre subpopulações. Uma subpopulação pode enviar ou receber indivíduos de quaisquer outras.

Na topologia caminho-de-pedras, a troca de indivíduos pode ocorrer apenas entre subpopulações adjacentes, em um anel. Somente algumas rotas de migração são permitidas, possibilitando uma disseminação de material genético mais gradual entre as subpopulações. A Figura 5.1 ilustra as topologias mais comuns em AEP's de granularidade grossa:

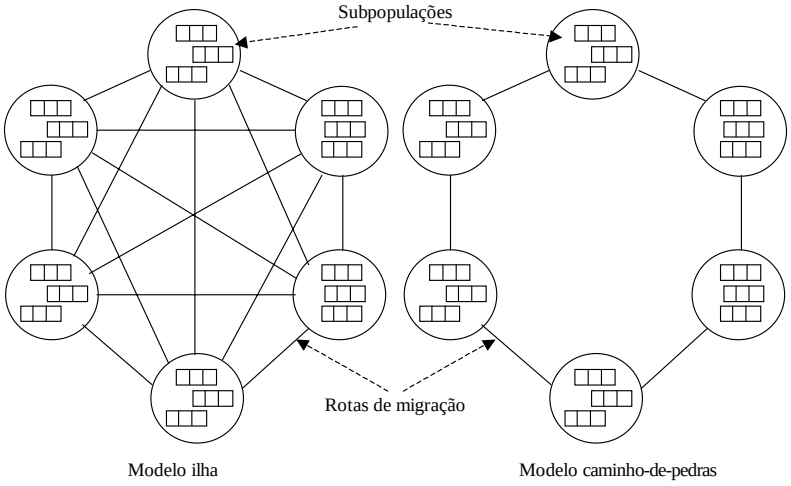


FIGURA 5.1 – Topologias ilha e caminho-de-pedras.

Uma topologia comumente usada em algoritmos com granularidade fina, executada em máquinas paralelas de memória compartilhada, é ilustrada na Figura 5.2. As subpopulações, que correspondem aos nós de uma malha, são interconectadas formando uma grade bidimensional. As subpopulações são uniformemente distribuídas entre os processadores e a migração ocorre entre subpopulações adjacentes, sejam de um mesmo processador ou de processadores diferentes.

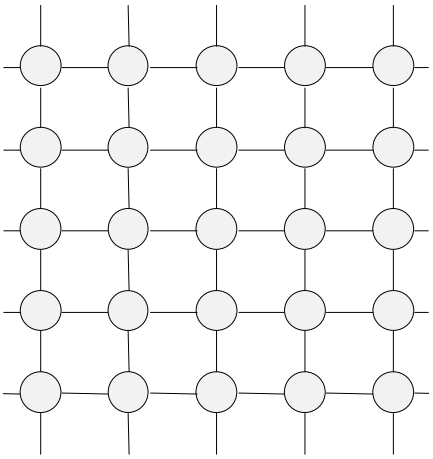


FIGURA 5.2 – Topologia em grade bidimensional.

Uma suposta vantagem dos AEP's reside na possibilidade de, através de uma topologia um tanto quanto restritiva, evitar a conhecida convergência prematura. Os modelos populacionais baseados em ilha, por serem pouco restritivos, podem se tornar equivalentes a um AG seqüencial. Isto, de certa forma, reduz o ganho com o paralelismo, na medida em que o sistema como um todo equivale a uma única grande população. A convergência prematura, em casos em que não há compromisso com relação ao tempo de execução, como em geral ocorre em problemas abordados por enfoques paralelos, pode ser remediada através de operadores do tipo *epidemia* (Medeiros e Guimarães, 2004). Nesse tipo de operação, todas as subpopulações são reiniciadas e apenas os melhores indivíduos são preservados.

Quanto mais restritivos forem a topologia e freqüência de migração, melhor será aproveitado o potencial de um AEP, permitindo que cada subpopulação realmente se especialize em um determinado subespaço de busca. Mesmo que as subpopulações convirjam para perfis de indivíduos subótimos, espera-se que cada uma delas convirja para diferentes perfis e que a migração promova uma certa diversificação populacional ao longo das gerações.

O critério de escolha de quais indivíduos devam ser migrados também é parâmetro de política de migração. Um critério elitista é simplesmente migrar os melhores de cada subpopulação. Entretanto, pode-se estabelecer outros que vislumbrem a troca de indivíduos aleatoriamente.

5.4 O modelo hierárquico de competição justa

O modelo *HFC* foi recentemente proposto por (Hu et al., 2002), inspirado em processos de estratificação que costumam ocorrer em algumas sociedades com o objetivo de preservar indivíduos menos hábeis (ou ainda em formação) de uma competição com outros mais hábeis. A estratificação constrói uma hierarquia de classes onde cada indivíduo se enquadra em uma delas, ali competindo com seus pares de habilidades similares. Um exemplo de um sistema hierárquico com competição justa é mostrado na Figura 5.3.

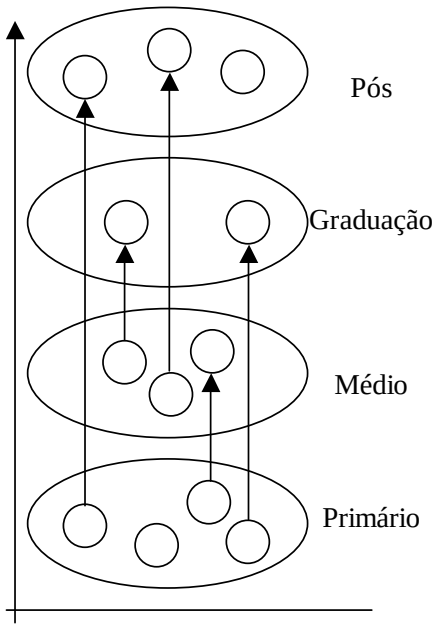


FIGURA 5.3 – Sistema acadêmico de ensino.

Pode-se dizer que a população é segregada em *castas* (ou classes sociais) de acordo com a aptidão de cada indivíduo. Mecanismos de promoção de indivíduos para as castas superiores, de acordo com o limiar de admissão de cada casta, compõem a política de migração do modelo.

Duas versões seqüenciais de um algoritmo baseado em *programação genética*, utilizando o modelo *HFC*, foram testadas com sucesso para problemas de síntese de circuitos analógicos (Hu et al., 2002). Uma delas, a mais promissora, definiu os limiares de

admissão, adaptativamente, de acordo com o domínio de *fitness* do problema em questão, sem a necessidade prévia de conhecimento desse domínio.

5.4.1 Diversidade populacional

Convergência é uma característica desejável para um AE, mas é um processo que deve ser controlado para se evitar mínimos locais. De um modo geral, os AG's promovem a convergência através da progressiva substituição dos indivíduos com baixa aptidão por outros mais qualificados. O próprio processo de seleção, por definição, deve privilegiar também os melhores indivíduos.

AG's, inspirados na teoria da evolução das espécies, são intencionalmente *injustos*. Indivíduos competem entre si e os melhores prevalecem, restringindo a diversidade populacional, tão importante para se preservar uma amostragem uniformemente distribuída pelo espaço de busca.

Preservar indivíduos menos adaptados, em termos biológicos, também preserva material genético que possa ser importante para se gerar um *super-indivíduo*, com as melhores características genéticas encontradas na população inteira. O modelo *HFC* foi proposto com o objetivo de prover mecanismos para se reduzir o risco de uma convergência prematura (Hu et al., 2002).

O mecanismo de estratificação de indivíduos permite preservar indivíduos diversificados e que poderiam não sobreviver se competissem diretamente com outros de alta qualidade. Em nível global, o *HFC* promove alguma *justiça* entre competidores com aptidões *significativamente* diferentes. Mas para aqueles indivíduos *sutilmente* diferentes, dentro da mesma casta, a competição ocorre como em qualquer outro algoritmo evolutivo (AE).

5.4.2 Topologia e política de migração

No modelo *HFC*, múltiplas subpopulações são organizadas em hierarquia de modo que cada uma delas somente contenha indivíduos dentro de sua faixa de aptidão específica. Dessa forma, o domínio de aptidão é mapeado para um número finito de faixas de aptidão definidas por limiares de admissão. Cada faixa de aptidão está associada a um *nível* de hierarquia. Cada nível se constitue de uma subpopulação, um conjunto de operadores evolutivos e seus respectivos parâmetros de desempenho. Existe ainda a possibilidade, ainda não explorada, de utilização de mais de uma subpopulação, com diferentes topologias em cada nível (Hu et al., 2002).

Indivíduos são movidos das castas mais baixas para as mais altas se, e somente se,

eles excederem o limiar de admissão de alguma casta superior. Nenhum indivíduo é movido para castas inferiores. Por isso, o operador de migração é unidirecional. A Figura 5.4 ilustra a topologia do modelo *HFC*. As setas indicam os movimentos migratórios possíveis. A *subpopulação de acesso* (ou nível de acesso) pode enviar indivíduos para todas outras subpopulações, enquanto a *subpopulação elite* (ou nível elite) só pode receber indivíduos (Hu et al., 2002).

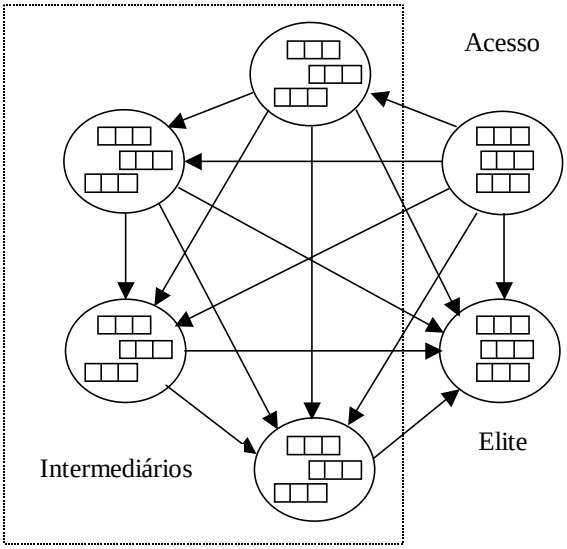


FIGURA 5.4 – Topologia do modelo *HFC*.

Com respeito a freqüência de migração, o modelo *HFC* estabelece que indivíduos devem ser movidos em intervalos regulares, usando um *buffer* de admissão para coletar os *imigrantes* que chegam de outras subpopulações. Qualquer indivíduo com aptidão fora da faixa de sua subpopulação deve ser migrado para a subpopulação apropriada. Dessa forma, o número de indivíduos a serem migrados não pode ser definido a *priori*, pois é dependente da dinâmica do processo evolutivo.

5.4.3 Geração contínua de indivíduos

Pode-se dizer que a competição justa é suficiente como um mecanismo explícito de diversidade populacional. Todavia, há ainda um segundo mecanismo sugerido em (Hu et al., 2002): a geração contínua de indivíduos. Como é sabido, os AE's inicializam a população uma única vez, não mais promovendo reinicializações, salvo quando ocorrem operações do tipo *epidemia*.

O modelo *HFC* contém um mecanismo elegante de geração de novos indivíduos. Uma vez que a casta acesso apenas envia indivíduos para as castas superiores, criam-se *lacunas*

de indivíduos que devem ser preenchidas com novos indivíduos. Esses novos indivíduos são gerados aleatoriamente e, eventualmente, podem migrar para as suas subpopulações correspondentes, dependendo de sua aptidão.

Sem a necessidade de epidemias, que têm um certo custo computacional, o modelo *HFC* permite a contínua renovação de material genético ao longo das gerações, afetando não somente a subpopulação de acesso, mas também todas as superiores.

5.4.4 Adaptabilidade

As faixas de aptidão são determinadas de acordo com o problema e o número de níveis. Um *estágio de calibragem*, que evolui uma ou mais subpopulações durante algumas gerações preliminares, é empregado para se determinar os valores iniciais de faixas de aptidão.

Em (Hu et al., 2002), durante esse estágio, a aptidão média f_μ, o desvio-padrão σ_f e a aptidão do melhor indivíduo f_* são calculados e usados para definir os limiares de admissão de cada subpopulação. Somente após, os indivíduos da população inicial são classificados, de acordo com suas aptidões, e movidos para seus níveis correspondentes.

Em intervalos regulares de gerações, um *estágio de atualização* é executado para que seja recalculado os indicadores f_μ, σ_f e f_*, após o que, todos os limiares dos níveis superiores ao nível de acesso são redefinidos. Ambos os estágios de *calibragem* e de *atualização* são empregados como parte do mecanismo de adaptação proposto em (Hu et al., 2002) e visam manusear a falta de conhecimento prévio sobre o problema.

5.5 Detecção de regiões promissoras através de competição justa

Um ambiente evolutivo heterogêneo pode ser incluído no modelo *HFC*, na medida em que se pode manter subpopulações com diferentes tamanhos, evoluindo através de operadores ajustados com diferentes valores de parâmetros de desempenho (Hu et al., 2002).

A proposta contida neste capítulo direciona o potencial do modelo *HFC* para lidar com a questão de como melhor paralelizar um algoritmo evolutivo híbrido. Por conseguinte, retorna-se à temática de detecção de regiões promissoras. A chave da questão reside no ambiente evolutivo heterogêneo. Cada nível da hierarquia pode assumir um papel diferente na evolução como um todo. As castas não têm compromisso de manterem um equilíbrio entre diversidade e convergência. Isso pode ser alcançado *no todo*, sem que ocorra *nas partes*.

O ambiente evolutivo heterogêneo permite que cada subpopulação tenha uma estratégia de busca diferente, dependendo de parâmetros evolutivos tais como pressão de seleção, probabilidade de mutação e cruzamento, por exemplo. Várias questões surgem com relação a possíveis estratégias globais de ajuste de parâmetros, discutidas posteriormente. Por enquanto, pode-se dizer que o modelo *HFC* dá suporte à implementação de um critério de detecção de regiões promissoras, baseado em elitismo, mas com alguns melhoramentos. A estratificação de indivíduos cria um ambiente propício para que heurísticas de busca local sejam aplicadas aleatoriamente a indivíduos considerados promissores, ou seja, indivíduos da *subpopulação elite*.

A falta de representatividade dos *indivíduos-elite* está entre os principais problemas associados ao elitismo como critério para detecção de regiões promissoras. Busca local aplicada a indivíduos da elite tendem a encontrar os mesmos melhores indivíduos, conduzindo a população para as mesmas melhores soluções.

O modelo *HFC* prevê a possibilidade de geração contínua de indivíduos. O movimento contínuo de indivíduos, que *sobem* níveis, *alimenta* a subpopulação elite com material genético novo e diversificado, permitindo que o elitismo possa ser usado sem a necessidade de estratégias sobressalentes para se evitar uma *super-amostragem* de regiões, com conseqüente perda de diversidade populacional. O elitismo, com o suporte de vários mecanismos de diversificação populacional, se constitue em uma estratégia apropriada para detecção de regiões promissoras.

5.6 Algoritmo Paralelo Hierárquico e Adaptativo com Competição Justa

Em (Hu et al., 2002), um algoritmo baseado em *programação genética* foi implementado para problemas de síntese de circuitos analógicos. Apesar de empregar o modelo de competição justa e de fazer várias proposições tais como, limiares adaptativos de admissão, *buffers* de transferência de indivíduos, entre outros, o algoritmo não foi executado em ambiente provido de múltiplos processadores, ou seja, a implementação não foi efetivamente paralelizada. Algumas das melhorias sugeridas em (Hu et al., 2002):

a) determinação adaptativa do número de níveis;

b) mecanismo mais sofisticado para cálculo de limiares de admissão;

c) implementação em ambiente com múltiplos processadores (paralela);

d) aplicação do *HFC* a problemas de grande porte.

Nesta seção, um algoritmo evolutivo paralelo é proposto para problemas de otimização numérica sem restrição. Este algoritmo é chamado de Algoritmo Paralelo Hierárquico

e Adaptativo com Competição Justa (*APHAC*). O *APHAC* incorpora várias melhorias em relação ao algoritmo evolutivo originalmente proposto em (Hu et al., 2002), com intuito de construir uma versão ainda mais fiel ao modelo *HFC*, mais robusta e adequada a problemas de grande porte. Para verificar se este objetivo foi alcançado, os testes computacionais utilizaram funções com centenas de variáveis (Oliveira et al., 2004).

O *APHAC* apresenta também a migração assíncrona entre subpopulações, que não foi implementada anteriormente e, ainda utiliza o ambiente evolutivo heterogêneo como suporte para o emprego de algoritmos de busca local em indivíduos promissores encontrados na subpopulação elite. Cada subpopulação é mapeada para um processador diferente e eventuais trocas de indivíduos (migração) são efetuadas usando chamadas à biblioteca de comunicação *MPI*.

5.6.1 Estrutura geral do algoritmo proposto

O *APHAC* pode ser resumido em cinco estágios: calibragem, adaptação, evolução, envio e recebimento. Em tempo de execução, o código objeto do *APHAC* é replicado e cada cópia é executada independentemente das demais, com sua própria subpopulação, em um processador. O número D de processadores usados em cada execução é pré-definido. Cada nível recebe uma numeração entre 0 e $D-1$, dependendo de sua posição na hierarquia. O nível de acesso é o primeiro nível ou nível 0.

A migração de indivíduos é feita via *hardware* de rede, através de chamadas às funções providas pela *MPI*. Portanto, cada um dos estágios é executado em um único processador, em uma dada replicação do *APHAC*. A discussão que se segue, considera uma replicação do *APHAC* isoladamente.

Inicialmente, são gerados, de forma aleatória, um número M pré-definido de indivíduos em cada subpopulação. O estágio de calibragem é executado uma única vez e se constitue em um número pequeno e pré-definido de gerações (seleção, cruzamento, mutação) sobre esses indivíduos. Ao final, os limites de domínio de *fitness* (f_{inf} e f_{sup}), que foram obtidos durante o processo, são guardados, bem como os limiares de admissão AT_i de cada subpopulação e outros *indicadores de domínio*, como média de *fitness*, que são detalhados posteriormente.

Todos os *indicadores de domínio* são calculados localmente, i.e., em cada uma das replicações. Eles são chamados de *indicadores locais de domínio*. Os indicadores locais de domínio são convertidos em *indicadores globais de domínio* através de um processo de consolidação, executado no nível elite. A consolidação de f_{inf} e f_{sup} é dada por:

$$f_{inf} = \min\left\{f_{inf}^{i}\right\}, i = 0, \ldots, D-1$$
$$f_{sup} = \max\left\{f_{sup}^{i}\right\}, i = 0, \ldots, D-1 \quad (5.1)$$

onde f_{inf}^{i} e f_{sup}^{i} são os limites de domínio calculados localmente em cada um dos D processadores.

O cálculo dos limiares de admissão são explicados na seção 5.6.2. Após a consolidação, todos os indicadores globais e os limiares de admissão são propagados para os demais processadores e são, a partir de então, usados nos cálculos para os quais se destinam.

O estágio de adaptação é executado regularmente, durante o processo evolutivo, para fazer ajustes nos limiares de admissão que foram calculados anteriormente. Ao contrário, os limites de domínio não são mais alterados no decorrer da evolução. O algoritmo a seguir mostra os cinco estágios do *APHAC*.

```
for all níveis i do
  Inicializa (P_i);
  CALIBRAGEM: Evolui (P_i, PoucasIterações);
  CALIBRAGEM: Consolidação(f_sup, f_inf, AT_i);
  repeat
    ADAPTAÇÃO: AT_i;
    EVOLUÇÃO: Evolui (P_i, MaximoIterações);
    ENVIO: Envia indivíduos superiores para respectivo nível;
    RECEBIMENTO: Recebe indivíduos para (P_{i>0});
    RECEBIMENTO: Complementa Aleatoriamente (P_0);
  until (critério de parada)
end for
```

Após o estágio de calibragem, a subpopulação P_i recebe um valor de *fitness* que equivale ao seu limiar de admissão AT_i. Não há garantias, a *priori*, de que todos os indivíduos em P_i estejam condizentes com AT_i. À medida em que a subpopulação evolui, os indivíduos que forem selecionados e estiverem fora da faixa de *fitness* são progressivamente movidos para seus devidos níveis.

O movimento de indivíduos está condicionado à existência de indivíduos a serem enviados ou recebidos. No caso da subpopulação de acesso, ocorre a geração contínua de indivíduos, complementando P_0 nos espaços criados após o envio de indivíduos superiores. A

transferência de indivíduos é detalhada posteriormente.

5.6.2 Mapeamento de aptidão

Não é uma tarefa trivial dividir o espaço de aptidão (*fitness*) em faixas uniformes. O espaço de busca de funções multimodais, pode apresentar, metaforizando a geografia, uma topologia peculiar formada de vales, planaltos, planícies e picos. O modelo *HFC* prevê um mapeamento de faixas de *fitness* para cada subpopulação. Entretanto, não há garantias de uma distribuição uniforme de subespaços de busca para cada uma das faixas de *fitness*. Algumas subpopulações podem ficar responsáveis por vastas planícies, por exemplo.

5.6.2.1 Problemas comuns

Não há conhecimento prévio sobre a topologia do problema em questão. O estágio de calibragem apenas possibilita determinar a faixa total de valores de *fitness*. Como pode ser observado no exemplo da Figura 5.5, subespaços extensos podem comportar muitos indivíduos diferentes mapeados para a mesma faixa de *fitness*. Por outro lado, outras faixas de *fitness* podem apresentar uma espécie de *carência* de soluções candidatas.

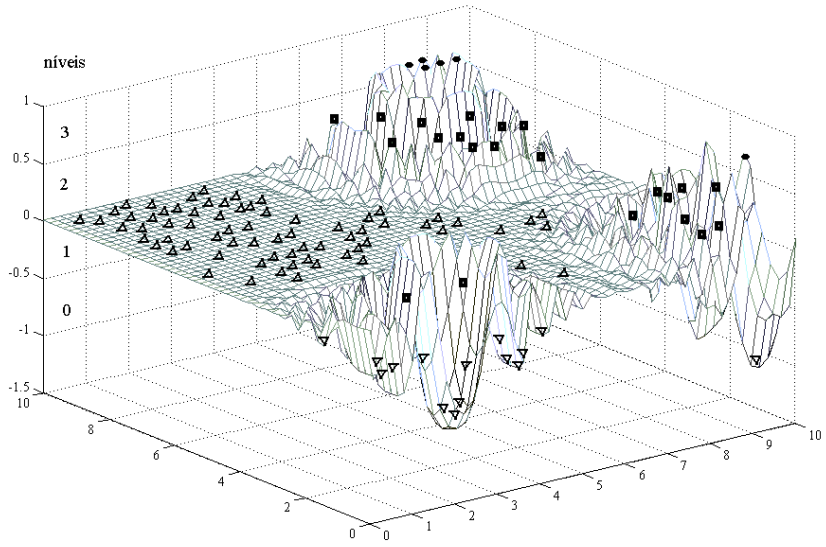


FIGURA 5.5 – Exemplo de divisão de espaço de *fitness* de uma função multimodal.

Subespaços de busca extensos ou ínfimos são igualmente indesejáveis para a dinâmica de evolução estratificada. Pode-se conjecturar que subpopulações focalizando tais subespaços podem convergir prematuramente por motivos diferentes, mas igualmente contribuindo para um desempenho fraco do algoritmo como um todo.

No caso de subespaços extensos, a subpopulação encontraria os mesmos problemas já mencionados no preâmbulo deste capítulo, ou seja, incapacidade de dispersão uniforme de indivíduos por todo o subespaço de busca. Mesmo com os benefícios da geração contínua de indivíduo, com o passar das gerações, haveria uma grande massa deles concentrados em poucas regiões, atraindo descendentes para essas mesmas poucas regiões.

No caso de subespaço ínfimo, por outro lado, a subpopulação tende a receber poucos indivíduos das camadas inferiores. Assim, a fraca renovação de indivíduos em um dado nível, da mesma forma, pouco contribui para o desempenho do algoritmo como um todo.

Em tese, a melhor forma de contornar tal problema, seria implementar uma etapa de calibragem mais sofisticada, na qual seriam estimados, além da faixa de *fitness* do problema, indicadores de densidade (Silverman, 1986) que viabilizariam a subdivisão de faixas de *fitness* em faixas menores ou a agregação de faixas adjacentes pouco densas.

Um outro problema comum é o *colapso de faixas de fitness*. Na etapa de calibragem, certas topologias de função podem tornar a faixa de *fitness* $[f_{inf}, f_{sup}]$ muito estreita, causando problemas numéricos nos cálculos dos limiares de admissão.

A solução encontrada para se evitar o *colapso de faixas de fitness*, no *APHAC*, é apresentada a seguir. A estimativa de densidade populacional não está implementada nesta etapa de trabalho.

5.6.2.2 Mapeamento proposto

Para contornar o *colapso de faixas de fitness*, no *APHAC*, a função de aptidão (*fitness*) não é a mesma função objetivo, f. Seja R um escalar constante, um mapeamento é feito para o intervalo linear entre $[0_-, R_+]$, provendo uma normalização de valores no domínio de *fitness*. A notação $[0_-, R_+]$ se refere à possibilidade de extrapolação desse intervalo caso sejam encontrados, no decorrer da evolução, indivíduos com **fitness** fora do intervalo $[f_{inf}, f_{sup}]$.

O *APHAC* é iniciado com o estágio de calibragem e cada nível i guarda o mais alto e mais baixo valores para a função objetivo (f^i_{sup} e f^i_{inf}, respectivamente). Posteriormente, os valores f_{sup} e f_{inf} são consolidados pelo nível elite e propagados para os demais níveis para que sirvam de parâmetros para o mapeamento. O valor de *fitness* $\hat{f}_{ij}$ do $j-$ésimo indivíduo em um nível i é dado por:

$$\hat{f}_{ij} = R\left[(f_{ij} - f_{inf})/(f_{sup} - f_{inf})\right] \tag{5.2}$$

Para problemas de minimização, o limiar de admissão AT_i para o nível i é dado por:

$$\begin{aligned} AT_{[i=0]} &= +\infty \\ AT_{[i>0]} &= \overline{\hat{f}} + \frac{(i-1)}{(D-2)} \left[\overline{\hat{f}} - \overline{\hat{f}}^{*} \right] \end{aligned} \tag{5.3}$$

onde D é o número de níveis (subpopulações), $\overline{\hat{f}}$ é a menor média de valores de *fitness* e $\overline{\hat{f}}^{*}$ é a menor *média dos valores de fitness acima da média*. Seja M o tamanho de cada subpopulação (igual para todas) e $\overline{\hat{f}_i}$ a média de *fitness* em um dado nível i, tem-se:

$$\begin{aligned} \overline{\hat{f}} &= \min_i^D \left[\tfrac{1}{M} \textstyle\sum_j^M \hat{f}_{ij} \right] \\ \overline{\hat{f}}^{*} &= \min_i^D \left[\tfrac{1}{M} \textstyle\sum_j^M \max(0, \overline{\hat{f}_i} - \hat{f}_{ij}) \right] \end{aligned} \tag{5.4}$$

Durante a evolução propriamente dita, em cada subpopulação, podem ser gerados indivíduos com função objetivo fora da faixa de valores $[f_{sup}, f_{inf}]$. Eles são tratados como casos especiais de mapeamento. A Figura 5.6 exemplifica o mecanismo de mapeamento $f \rightarrow \hat{f}$ para os três possíveis casos, definidos por:

a) $\hat{f}_{ij} = R\,[(f_{inf} - f_{ij})/(f_{sup} - f_{ij})], \quad f_{ij} \in [-\infty, f_{inf}]$

b) $\hat{f}_{ij} = R\,[(f_{ij} - f_{inf})/(f_{sup} - f_{inf})], \quad f_{ij} \in [f_{inf}, f_{sup}]$

c) $\hat{f}_{ij} = R\,[(1 + f_{ij} - f_{sup})/(f_{ij} - f_{inf})], \quad f_{ij} \in [f_{sup}, \infty]$

5.6.3 Migração assíncrona

Após o estágio de calibragem, com o correr da evolução, cada subpopulação pode apresentar indivíduos classificados como *inferiores*, *pertinentes* ou *superiores*, de acordo com seus respectivos valores de *fitness*. Para uma dada subpopulação, os *inferiores* são aqueles que pertencem a alguma faixa inferior de *fitness*, os *pertinentes* pertencem à faixa de *fitness* da subpopulação onde foram gerados e os *superiores*, por sua vez, pertencem a alguma faixa superior de *fitness*. A subpopulação de acesso não tem indivíduos *inferiores*, nem tampouco a subpopulação elite tem indivíduos *superiores*. Nas subpopulações intermediárias, podem ocorrer os três tipos de indivíduos.

Os indivíduos inferiores são mantidos na subpopulação, no nível hierárquico onde foram gerados, juntamente com os indivíduos pertinentes. Há uma tendência desses indivíduos

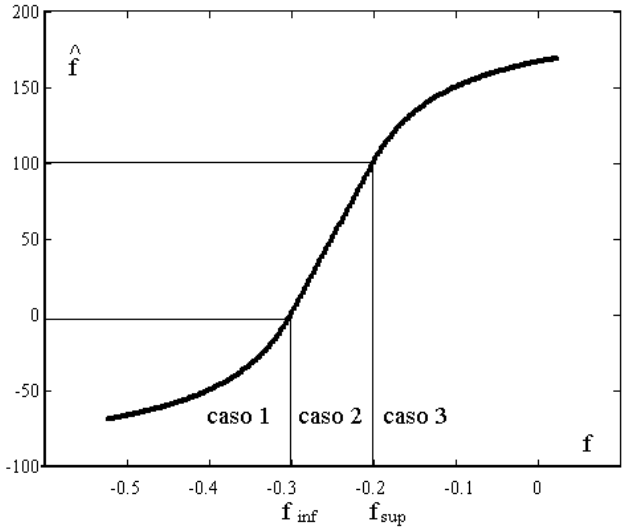


FIGURA 5.6 – Mapeamento de função objetivo para *fitness* usando $R = 100$.

serem suplantados por indivíduos melhores e, por isso, desaparecer. O modelo *HFC* não prevê a migração para níveis inferiores (Hu et al., 2002). Uma possível desvantagem na implementação dessa característica estaria na sobrecarga do sistema como um todo em função do número de migrações que poderiam ocorrer.

Cada nível na hierarquia tem interesse apenas pelos indivíduos superiores que eventualmente forem gerados. Nas implementações originais baseadas no *HFC*, os indivíduos superiores são mantidos nas subpopulações de origem até o chamado *tempo de migração*, que ocorre em intervalos regulares de geração. Até antes do *tempo de migração*, os indivíduos superiores participam do processo evolutivo nas suas respectivas subpopulações de origem (Hu et al., 2002).

Pode-se dizer que a migração, nesse caso, ocorre de forma síncrona, com a participação de todos os níveis, verificando quais indivíduos estão fora das faixas de *fitness* de suas subpopulações e copiando esses indivíduos para os *buffers* de admissão das subpopulações apropriadas.

No *APHAC*, a migração ocorre de uma forma completamente assíncrona. Indivíduos superiores são postos em chamados *buffers de saída*, excluídos das suas subpopulações de origem, tão logo sejam identificados como tal. Dessa forma, indivíduos superiores não afetam o processo evolutivo, por não competirem em nenhum momento com os demais indivíduos de suas subpopulações de origem. Somente depois de serem enviados para seus devidos níveis, é que eles vão contribuir para o processo evolutivo.

Os *buffers* têm um tamanho máximo, previamente determinado. Quando o *buffer* de saída de uma subpopulação estiver cheio de indivíduos superiores, colhidos durante uma

geração, o processo evolutivo dessa subpopulação é parado e todos os indivíduos desse *buffer* são *exportados* para os seus devidos destinos.

Seguinte à parada no processo evolutivo local, é verificado se há indivíduos a serem *importados*, advindos de níveis inferiores. Caso haja, esses indivíduos são recebidos em um chamado *buffer de entrada* que já faz parte da subpopulação e, portanto, já permite aos indivíduos importados participarem do processo evolutivo na próxima geração.

A *MPI* provê funções de comunicação que mantém os indivíduos exportados em *buffers de sistema*[2] até o momento em que eles são efetivamente recebidos pelos processos aos quais se destinam. Os *buffers de sistema* também são limitados e o *APHAC* prevê a consulta de tempos em tempos para saber se não há *indivíduos chegando*, prontos para serem armazenados nos *buffers* de entrada. A consulta ocorre depois de cada geração, mesmo que não ocorra a parada por conta do *buffer* de saída cheio.

Uma geração, no *APHAC*, corresponde a uma unidade de evolução que é interrompida por três possíveis eventos: *buffer* de saída cheio, um número pré-estabelecido de iterações, ou obtida a melhor solução (solução ótima). Sempre que um desses eventos ocorrer, o processo evolutivo do nível é interrompido, o indivíduos no *buffer* de saída são exportados e o *buffer* de entrada é carregado com os indivíduos que estiverem aguardando para serem recebidos.

O *buffer* de entrada, como já foi mencionado, já faz parte da subpopulação e, portanto, os indivíduos postos lá podem participar do processo evolutivo na geração seguinte. Entretanto, esses indivíduos precisam ser deslocados para a subpopulação propriamente dita, pois o *buffer* de entrada é uma área de comunicação necessária para os futuros recebimentos de indivíduos. A transferência de indivíduos compreende desde o instante que eles são identificados como superiores em suas subpopulações de origem até serem efetivados como indivíduos aptos a participarem do processo evolutivo em suas devidas subpopulações. A Figura 5.7 mostra o processo assíncrono de transferência de indivíduos.

Os indivíduos importados substituem os indivíduos menos aptos subpopulações de destino, ou seja, o mecanismo *steady-state* de atualização de indivíduos (Davis, 1991). O *steady-state* implementado neste trabalho, não considera a subpopulação inteira na procura pelo pior indivíduo. Apenas uma pequena parte dela é verificada aleatoriamente.

[2]A forma como é realizado internamente o processo de comunicação depende da implementação/versão da biblioteca em uso.

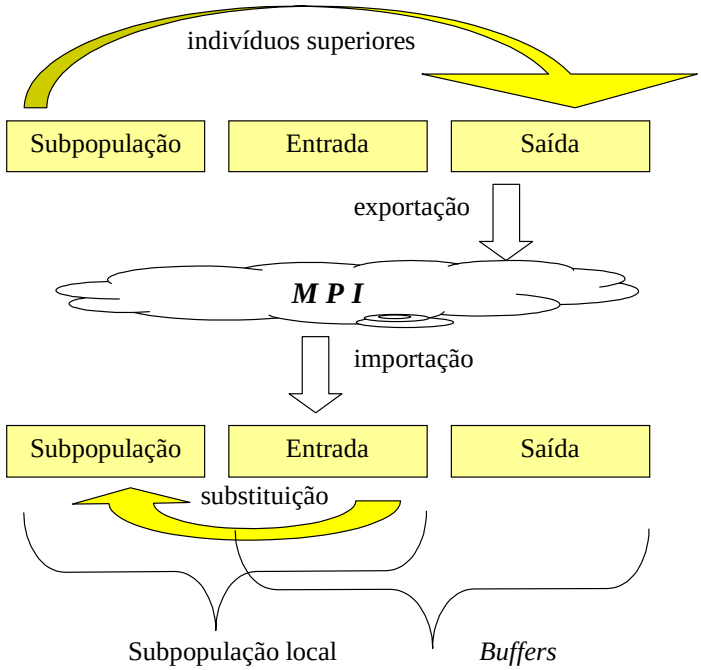


FIGURA 5.7 – Transferência de um indivíduo de uma subpopulação para outra.

5.6.4 Ambiente evolutivo heterogêneo

O ambiente evolutivo heterogêneo foi implementado através de diferentes valores de parâmetros evolutivos para cada nível. Os parâmetros evolutivos determinam o desempenho dos operadores evolutivos, tornando-os mais ou menos diversificadores. Por exemplo, altas taxas de mutação, em geral, causam alterações mais duradouras no perfil de genótipos da população. A Figura 5.8 traz um esboço do ambiente evolutivo heterogêneo.

O ajuste dos parâmetros evolutivos segue uma estratégia de diversificação populacional que pode ser tanto *crescente* quanto *decrescente*. Na *estratégia crescente* de diversificação, os parâmetros evolutivos são ajustados para induzir uma maior diversidade nas subpopulações dos níveis mais altos da hierarquia do *APAC*.

A idéia por trás da estratégia crescente está na busca por um equilíbrio entre convergência e diversidade. Os níveis superiores trabalham com indivíduos de alta qualidade, representantes de poucas regiões do espaço de busca e há uma tendência para convergência nessas poucas regiões. Os níveis inferiores sofrem mais influência da geração contínua de indivíduos que é um processo razoavelmente diversificador. A estratégia crescente procura a diversificação na subpopulação elite e convergência na subpopulação de acesso. Os níveis intermediários, por sua vez, assumem perfis intermediários, linearmente proporcionais.

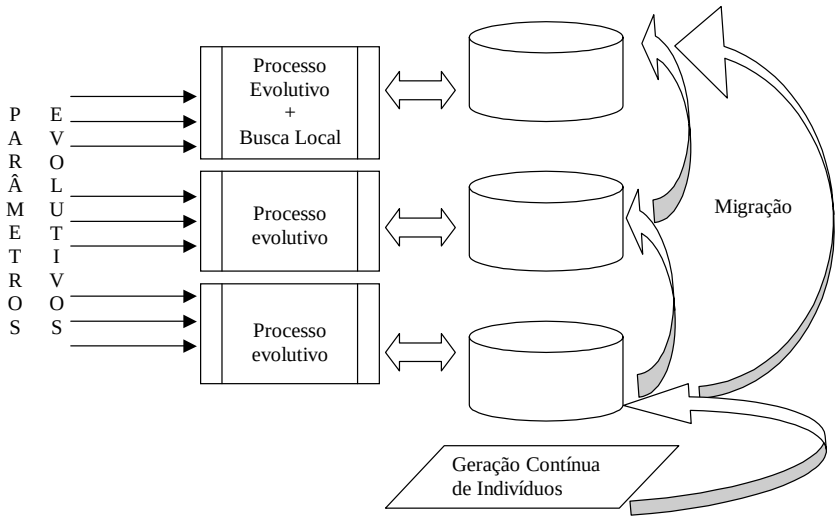


FIGURA 5.8 – Ambiente evolutivo heterogêneo.

Por outro lado, a *estratégia decrescente* traz uma idéia diferente. Considerando que os melhores indivíduos são também os mais promissores em se tratando de pontos de referência para uma intensificação de busca, os operadores evolutivos devem ter um comportamento menos diversificador, procurando uma convergência em torno desses indivíduos promissores. A diversificação mais acentuada nos níveis inferiores, por sua vez, tende a potencializar ainda mais o efeito da geração contínua de indivíduos.

A estratégia decrescente parece ser um pouco mais apropriada à linha de raciocínio seguida neste trabalho. Entretanto, o *APHAC* com a estratégia crescente obteve os melhores resultados e por isso foi adotado nos experimentos que são apresentados posteriormente. A taxa $\rho(m)$ de mutação não-uniforme (Michalewicz, 1996), o parâmetro blx_α do cruzamento BLX (Eshelman e Schaffer, 1993) foram ajustados, para cada nível i, da seguinte forma:

$$\rho(m)^i = \frac{i}{2(D-1)}, i = 0, \ldots, D-1 \tag{5.5}$$

$$blx_\alpha^i = 0,10 + \frac{0,25\ i}{(D-1)}, i = 0, \ldots, D-1 \tag{5.6}$$

Há ainda um terceiro parâmetro que é a taxa de busca local. Conforme mencionado anteriormente, a busca local ocorre apenas em indivíduos aleatoriamente selecionados na subpopulação elite. Ou seja, equivale a uma taxa nula de busca local para todos os níveis com exceção do elite. A probabilidade L_SP com que são selecionados indivíduos da subpopulação elite é alvo de estudo nos experimentos realizados neste trabalho.

Para esta aplicação, a busca local também empregou o mesmo algoritmo, descrito na seção 4.4.1.5, baseado na busca direta de *Hooke e Jeeves* (*HJD*). O desempenho do HDJ depende da escolha do passo inicial dx. Nesta implementação, dx foi ajustado para assumir valores aleatórios entre 1% e 10% da diferença $(x^{sup} - x^{inf})$, onde x^{sup} e x^{inf} são os conhecidos limites superior e inferior do domínio da variável x.

O *ECS* empregou a *seleção com pressão auto-adaptativa* (SPAA), descrita na seção 4.4.1.1. Como foi mencionado, a pressão de seleção do SPAA é controlada pelo número de vezes em que o indivíduo fora selecionado até então e sua distância para o atual melhor indivíduo em termos de *fitness*. A equação 4.4 é aqui repetida:

$$Z(s_k) = \frac{1}{|f^* - f(s_k) + 1|^{ns_k}}$$

Uma possível adaptação para a equação 4.4, particularmente interessante para *APHAC*, seria utilizar, ao invés do número de seleções ns_k do indivíduo, um certo K, correspondendo ao parâmetro de ajuste de pressão de seleção para cada nível da hierarquia. Quanto maior K, maior seria a pressão de seleção.

A estratégia crescente ou decrescente de diversidade ajustaria o valor de K, progressivamente para cada nível, alterando o desempenho do operador de seleção. Isso reforçaria ainda mais o ambiente evolutivo heterogêneo, tornando a seleção de indivíduos, assim como o cruzamento e a mutação, também adaptada ao perfil dos indivíduos em cada nível. Entretanto, essa versão puramente adaptativa, não obteve resultados melhores que a versão auto-adaptativa. Assim, optou-se SPAA nos experimentos apresentados neste trabalho. Do ponto de vista do processo de seleção, os ambientes evolutivos que compõem o *APHAC* são homogêneos.

5.6.5 Algoritmo completo

Todas as características previamente descritas aparecem no pseudo-código completo do *APHAC* apresentado a seguir:

```
for all níveis i do
  Inicialize (P_i);
  Evolua (P_i, PoucasIterações);
  Atribua parâmetros (ρ(m)^i, L_S P^i, blx^i_α);
  repeat
    if tempo de adaptar then
      Compute a equação 5.4: f̄, f̄*;
      Compute a equação 5.3: AT_i;
      Envie AT_i para cada P_i;
    end if
    Evolua(P_i,MaxIterações);
    while (buffer de saída não vazio) do
      Envie indivíduo para RESPECTIVA P_i;
    end while
    if (i = acesso) then
      Complemente (P_i);
    else
      while (indivíduos chegando) do
        Receba Indivíduo;
        Mova Indivíduo para buffer de entrada;
      end while
    end if
  until (critério de parada)
end for
```

O critério de parada é dado por um número específico de chamadas da função objetivo ou por alguma métrica relativa a qualidade da melhor solução encontrada.

5.7 Minimização de funções numéricas sem restrição

Para estes experimentos, foram empregadas quatro funções-teste que podem ser generalizadas para um número qualquer de variáveis, sem que o mínimo global seja alterado. São elas: *Rosenbrock* (Ros), *Schwefel* (Sch), *Griewank* (Gri) e *Rastrigin* (Ras). Pretende-se mostrar que o *APHAC* fornece um mecanismo apropriado à longas execuções, em geral, necessárias para solução de problemas com centenas de variáveis. Para tanto, foram feitos testes com essas quatro funções, variando-se n de 200 a 700.

5.7.1 Ambiente de teste

Os experimentos ocorreram em uma plataforma paralela de memória distribuída (*cluster de PC's*) com 17 nós (processadores), arquitetura *IA-32, com processador AMD Athlon 1.67 GHz, memória de 512 MB* e rede de interconexão padrão *Fast Ethernet*. Alguns parâmetros do *APHAC* foram ajustados da seguinte forma:

- tamanho das subpopulações e de cada um dos *buffers* foi fixado em 100 (100+ 100 + 100, no total);
- o estágio de calibragem evolui a população inicial durante 10 gerações;
- o estágio de adaptação ocorre a cada 2000 gerações;

Os testes objetivam, primeiramente, provar a eficácia do algoritmo proposto, i.e., sua capacidade de encontrar soluções satisfatórias para minimização de funções numéricas sem restrição com centenas de variáveis. Dentro desse aspecto, avaliar a influência do número de subpopulações (processadores) em termos de tempo de execução e qualidade das soluções obtidas.

A influência da busca local também é relevante e, por isso, diferentes probabilidades L_SP de busca local foram testadas. O impacto do aumento no número de buscas locais é especialmente importante em se tratando de problemas envolvendo muitas variáveis. Por fim, um outro aspecto em foco é a transferência de indivíduos entre subpopulações.

Como tem sido a retórica deste trabalho, a competitividade também é importante no que tange eficácia da abordagem. Entretanto, não foram encontrados na literatura trabalhos recentes aplicando algoritmos evolutivos a funções-teste com centenas de variáveis.

Um trabalho recente introduz mecanismos para escolha de taxas de mutação e de cruzamento ótimos em algoritmos evolutivos paralelos (Miki et al., 2000). Esse mesmo trabalho mostra que as melhores taxas dependem do tamanho da população e diferem entre as implementações seqüencial e paralela do mesmo algoritmo. Todavia, esse trabalho não traz experimentos com funções-teste com mais de 10 variáveis (Miki et al., 2000).

Um outro algoritmo evolutivo paralelo híbrido, empregando Simplex de Nelder e Mead (Nelder e Mead, 1965), chegou a ser aplicado a funções-teste com até 30 variáveis (Guanqi e Shouyi, 2003). Mas ainda não serve de base de comparação com o *APHAC*. Em virtude disso, a competitividade do *APHAC* foi posta a prova unicamente com relação a uma versão similar, executada seqüencialmente.

5.7.2 Resultados computacionais

Cada experimento consistiu de 10 execuções e cada execução foi limitada a 20×10^6 chamadas à função objetivo (CFO), número que pode ser obtido por qualquer uma das replicações do *APHAC*. A geração contínua de indivíduos é responsável por uma grande parte das CFO. A Figura 5.9 mostra o número de CFO acumuladas durante uma execução, usando 5 subpopulações, para a função Ras($n = 200$). Observa-se que os níveis de acesso (P_0) e elite (P_4), nessa ordem, são os que mais fazem CFO. Entretanto, não é sempre assim que o *APHAC* se comporta. Em outra execução, aumentando-se o número de subpopulações para 15, o nível elite foi o responsável por $94,14\%$ do total de CFO.

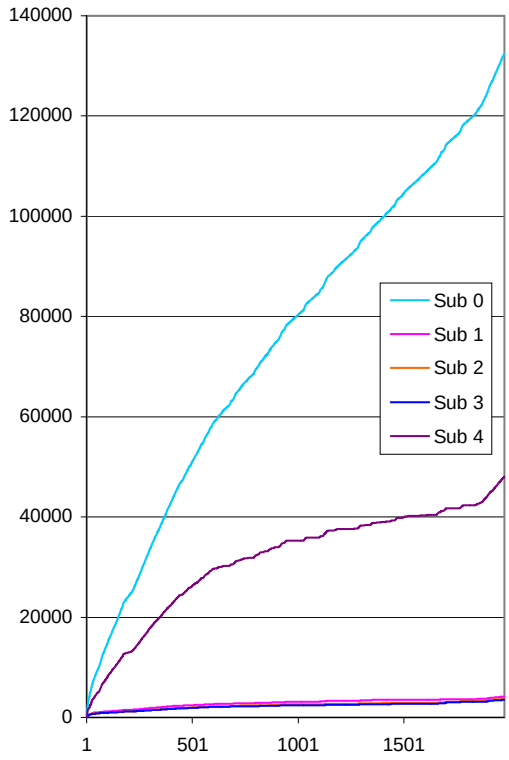


FIGURA 5.9 – Chamadas à função objetivo acumuladas para cada um dos níveis.

A Tabela 5.1 apresenta os resultados obtidos pelo *APHAC* para as funções-teste Ros($n = 500$), Sch($n = 200$), Gri($n = 200/700$) e Ras($n = 200$). São mostrados nesta tabela, o número de execuções bem-sucedidas, NS, ou seja, o número de vezes em que o *APHAC* obteve uma solução com valor igual ou inferior a $0,001$, a média das melhores soluções obtidas nas 10 execuções, denotada por MSE, e a média dos tempos de execução, TE, medidos em segundos apenas nas execuções bem-sucedidas.

Quando se fala em tempo de execução em processos executados paralelamente, em geral, se refere ao tempo total entre o primeiro processo a ser iniciado e o último processo a ser finalizado. Assim, o tempo de execução considerado para efeito de desempenho, neste experimento, é o maior medido dentre os tempos de todos os processos em cada execução. Assim, pode-se dizer que TE é a média de todos os maiores tempos de execução.

A probabilidade de ocorrer busca local no nível elite foi fixado em valores pequenos ($L_SP = \{0\%; 0,1\%; 0,5\%\}$). Para $L_SP = 0\%$ (sem busca local), não foram encontradas soluções satisfatórias para Ros($n \geq 200$) nem tampouco para Gri($n > 200$). Assim, quando ajustado com $L_SP = 0\%$, somente são mostrados os resultados para Gri, Ras e Sch (todos com $n = 200$). Para $L_SP = 0,1\%$ e $L_SP = 0,5\%$, a solução esperada $(0,001)$ foi obtida para Gri($n = 700$) and Ros($n = 500$). Os experimentos foram executados com $D = \{6, 9, 12, 15\}$.

TABELA 5.1 – Desempenho do *APHAC* em 10 execuções.

Func	n	MSE	NS	TE(s)	MSE	NS	TE(s)	MSE	NS	TE(s)	MSE	NS	TE(s)
		L_SP 0%		D = 6	L_SP 0%		D = 9	L_SP 0%		D = 12	L_SP 0%		D = 15
Gri	200	0,0010	8	212,1	0,0034	6	246,5	0,0032	9	196,3	0,0010	10	149,8
Ras	200	0,0009	8	338,8	0,0016	8	375,8	0,0015	8	335,3	0,0021	4	348,7
Sch	200	0,0010	10	439,4	0,0009	10	323,0	0,0010	10	409,0	0,0010	9	537,6
		L_SP 0,1%		D = 6	L_SP 0,1%		D = 9	L_SP 0,1%		D = 12	L_SP 0,1%		D = 15
Gri	700	0,0003	10	753,1	0,0005	10	725,8	0,0002	8	751,1	0,0001	10	660,4
Ras	200	0,0026	8	379,8	0,0034	6	327,2	0,0023	6	319,1	0,0013	5	305,5
Ros	500	0,0007	10	506,7	0,0005	8	312,9	0,0007	10	392,1	0,0004	10	344,9
Sch	200	0,0009	10	318,5	0,0010	8	334,2	0,0009	8	288,2	0,0009	10	445,7
		L_SP 0,5%		D = 6	L_SP 0,5%		D = 9	L_SP 0,5%		D = 12	L_SP 0,5%		D = 15
Gri	700	0,0003	10	781,8	0,0002	10	825,4	0,0002	10	842,1	0,0001	10	583,3
Ras	200	0,0022	8	322,4	0,0023	6	317,3	0,0029	8	330,6	0,0026	8	336,8
Ros	500	0,0006	10	415,2	0,0009	10	491,6	0,0008	10	585,0	0,0003	8	322,3
Sch	200	0,0010	10	266,7	0,0011	8	318,5	0,0010	8	267,9	0,0018	8	348,8

Com relação ao número de processadores, não é possível eleger o melhor D para todos os problemas. Observa-se que o crescimento linear do número deles não afeta linearmente o desempenho do método. Em geral, o que se espera de algoritmos paralelos é que, a medida em são incluídos mais processadores, o desempenho responda positivamente ao aumento no poder de processamento.

Sabe-se que, a partir de um certo número de processadores, não há mais ganho de tempo de execução em virtude do aumento do tráfego de rede ou devido a custos computacionais relativos ao processo de comunicação (Kumar et al., 2003). Isoladamente, algumas funções-teste apresentam um padrão de desempenho em função do aumento de D. Mas, isso não é observado no cômpito geral. Na Figura 5.10, os tempos de execução obtidos podem ser melhor visualizados.

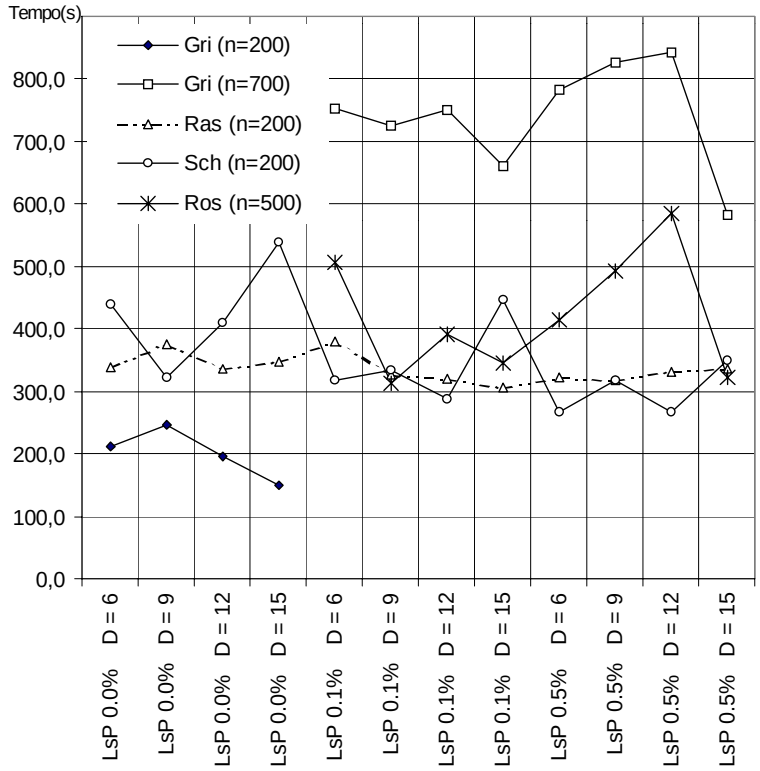


FIGURA 5.10 – Média dos maiores tempos de execução para cada função-teste.

Observação semelhante pode ser aplicada à probabilidade de busca local. Nos casos Ras($n = 200$) e Sch($n = 200$), o *APHAC* com operador de busca local ($L_SP > 0$) não obteve melhores resultados que a versão sem busca loca. Isso indica que o mecanismo de detecção de regiões promissoras do *APHAC* não foi eficiente em todos os casos. Para Gri($n = 200/700$), pode-se observar que os melhores resultados foram obtidos com $D = 15$.

Apesar de não ser possível predizer o comportamento de múltiplas populações evoluindo em processadores diferentes, amostrando subespaços de buscas que não estão uniformemente divididos, espera-se que algoritmos paralelos sejam pelo menos mais rápidos que seus pares seqüenciais.

O *APHAC* foi comparado com uma versão seqüencial, o Algoritmo Genético Codificado em Real (AGCR), muito similar ao procedimento *Evolua(P_i,MaxIterações)*, descrito no pseudo-código apresentado na seção 5.6.5. O AGCR emprega os mesmos operadores evolutivos e de busca local com probabilidade de $0,5\%$. O AGCR foi executado seqüencialmente em uma máquina do mesmo *cluster de PC's*. O mesmo critério de parada foi usado em 10 execuções. As médias dos tempos de execução para cada função foram: Gri($n = 200$) $= 566,36s$, Ras($n = 200$)$= 480,57s$ e Sch($n = 200$)$= 529,47s$.

Não foram encontradas soluções satisfatórias para Ros($n \geq 200$) e Gri($n > 200$). Além disso, o número de execuções bem-sucedidas, NS, caiu significativamente para valores inferiores a 4 execuções bem-sucedidas. Como pode ser observado na Tabela 5.1, o *APHAC* apresentou $4 \leq NS \leq 10$.

5.7.3 Comportamento geral do *APHAC*

Examinando-se os experimentos realizados, observou-se que o principal desafio para o *APHAC* é manter todas as subpopulações em contínua atividade. Se uma subpopulação qualquer convergir para um único perfil, ela pode deixar de enviar indivíduos para as subpopulações superiores. Isso evidentemente não é interessante para o bom desempenho do algoritmo, mas também não chega a ser determinante para o fracasso da busca, visto que as subpopulações inferiores não são afetadas pela convergência das subpopulações acima delas e a migração continuaria ocorrendo por outras vias.

Por exemplo, a Figura 5.11 mostra dois gráficos de comportamento do *APHAC*, executado para as funções Ras($n = 200$) (multimodal) e Ros($n = 200$) (monomodal), usando 5 subpopulações. Neles, as legendas *Env Sub i* e *Rec Sub i* significam, respectivamente, número acumulado de indivíduos enviados e recebidos para o nível i, ao longo de várias gerações (eixo das abscissas). Apesar de aparecerem na legenda, o nível de acesso não recebe e o nível elite não envia indivíduos.

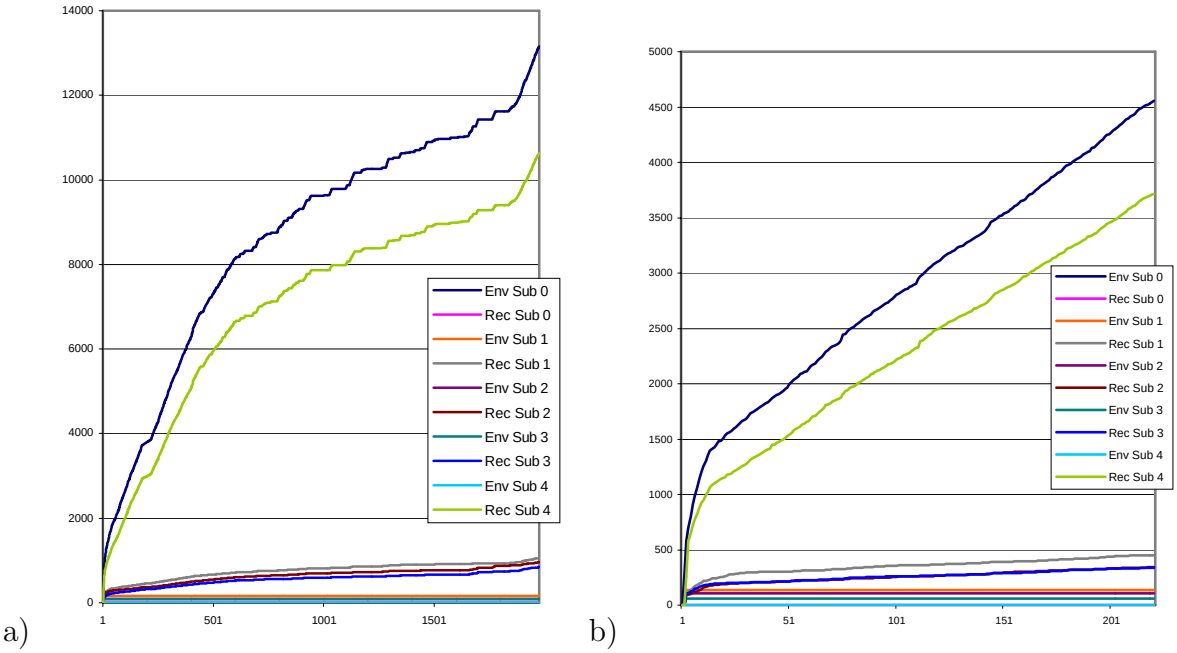


FIGURA 5.11 – Indivíduos enviados e recebidos para as funções-teste de *Rastrigin* e *Rosenbrock*, usando cinco níveis.

Como pode ser observado na Figura 5.11, para ambas as funções-teste, a maioria dos indivíduos que chegaram ao nível elite vieram do nível de acesso. Uma pequena parte dos indivíduos, enviados por esse nível, foram endereçados para os níveis intermediários 1, 2 e 3. Os níveis intermediários, nestas execuções, pouco contribuíram com indivíduos para o nível elite, pois, receberam bem mais do que enviaram indivíduos.

A geração contínua de indivíduos não garante bons indivíduos distribuídos para cada subpopulação. Os níveis superiores (nível elite, principalmente) podem não receber um número significativo de indivíduos de qualidade gerados aleatoriamente. Assim, pode-se concluir que o bom desempenho do nível de acesso, em termos de geração de indivíduos de qualidade, ocorre por mérito próprio.

A Tabela 5.2 mostra os percentuais de indivíduos enviados e recebidos, bem como a distribuição do número de CFO em outra execução para a função Ras($n = 200$), desta vez usando 15 subpopulações.

TABELA 5.2 – Percentuais de participação na migração e no número de CFO para Ras($n = 200$), usando quinze níveis.

	Enviados	Recebidos	CFO
Elite	0,00%	90,21%	94,14%
Intermediários	4,16%	9,79%	0,08%
Acesso	95,84%	0,00%	5,78%

Observa-se que todos os 13 níveis intermediários tiveram participação de $4,16\%$, $9,79\%$ e $0,08\%$ no envio e recebimento de indivíduos, bem como de CFO. Estes percentuais não se referem a indivíduos enviados ou recebidos para os níveis de acesso e elite, unicamente. Por exemplo, os $4,16\%$ é referente também a indivíduos enviados para subpopulações em níveis intermediários. Mesmo não se identificando exatamente quais rotas de migração exatamente foram as mais utilizadas, pode-se concluir que, tanto para 5 quanto para 15 níveis, houve uma super-utilização da rota de migração entre os níveis de acesso e elite. Este comportamento ocorreu com outras funções e provavelmente se deve ao fato desses níveis serem os extremos da hierarquia e, portanto, mais ativos.

Os novos indivíduos que são recebidos por uma subpopulação qualquer são postos no *buffer* de entrada e passam a competir normalmente com os demais indivíduos. Em termos globais, o modelo *HFC* promove uma competição justa. Entretanto, em termos locais, foi observado que indivíduos recém-chegados, em geral, têm qualidade inferior àqueles que foram evoluídos na própria subpopulação. Internamente, essa competição é

tão injusta quanto for a largura das faixas de *fitness*. A SPAA tem um papel importante, dando oportunidade que cada indivíduo recém-chegado tenha uma alta pressão de seleção, em um primeiro instante.

Faixas estreitas, de certa forma, tornam mais homogêneo o perfil dos indivíduos pertinentes, reduzindo assim a competição injusta interna. Esse raciocínio não pode ser demasiadamente estendido, pois a competição é necessária para que haja evolução e conseqüentemente otimização. O número de faixas a permitir uma competição equilibrada depende do problema. Mas, pelos resultados obtidos para as quatro funções-teste, pode-se supor que o algoritmo é suficientemente robusto para trabalhar bem com várias possibilidades.

5.8 Considerações finais

O algoritmo paralelo, hierárquico e adaptativo de competição justa (*APHAC*) separa indivíduos em subpopulações com diferentes perfis de *fitness*. Através dessa estratificação, uma competição justa entre indivíduos com perfis diferentes dá sustentação a um mecanismo de detecção de regiões promissoras baseado em elitismo. Uma aplicação para otimização numérica sem restrição foi implementada, acrescentando melhorias ao algoritmo evolutivo originalmente sugerido em (Hu et al., 2002).

A implementação proposta é efetivamente paralela. A biblioteca de comunicação *MPI* operacionaliza a migração assíncrona de indivíduos. O *APHAC* apresenta um estratificação completa, não só de subpopulações com diferentes perfis, mas também em termos de estratégias de busca em cada uma delas, incluindo a possibilidade de busca local aplicada em indivíduos da subpopulação elite, supostamente os mais promissores. Estas novas características provêem uma alta fidelidade ao modelo *HFC* original.

O *APHAC* foi testado em funções-teste numéricas com centenas de variáveis e os resultados foram melhores que um algoritmo padrão seqüencial. Entretanto, não foi possível estabelecer um número ideal de processadores e de probabilidade de busca local para todas as funções-teste testadas.

O modelo *HFC* divide uniformemente o espaço de *fitness*, atribuindo cada faixa a uma subpopulação. Não há, entretanto, a garantia de que o espaço de busca seja igualmente dividido entre os processadores. É possível, por exemplo, que certas faixas de *fitness* sejam mapeadas para subespaços de busca extensos e o processo de busca nessas regiões se torne ineficiente. Ainda assim, o tempo de execução para se encontrar soluções de qualidade foi substancialmente reduzido com o *APHAC*.

A melhor forma de balancear o fluxo de indivíduos entre as subpopulações, evitando espaços super ou sub-explorados, seria implementar-se um estágio de calibragem no qual seriam estimados também indicadores de densidade demográfica que permitiriam uma distribuição de subespaços mais equilibrada.

Existe um aspecto promissor na concepção do *APHAC*. A geração contínua de indivíduos, em associação com uma estratégia que equilibre diversificação e convergência, fornece um mecanismo apropriado para longas execuções. Soluções satisfatórias para a função de *Griewank*, testada com até 700 variáveis, por exemplo, foram encontradas em menos de 15 minutos.

Problemas de otimização de grande-porte são comumente encontrados em áreas como treinamento de redes neurais e problemas inversos. Alguns deles com aplicações industriais, otimizando milhares de variáveis, tais como problema de detecção de falhas estruturais (Chiwiacowsky et al., 2003). Problemas como esse requerem tempos de execução mais longos, em máquinas de grande poder de processamento, preferencialmente em arquiteturas paralelas.

CAPÍTULO 6

CONCLUSÃO

Um algoritmo evolutivo híbrido (AEH) pode ser definido como um algoritmo evolutivo que empregue alguma heurística de busca local, específica para o problema em enfoque, paralelamente ao processo evolutivo, como forma de melhorar a qualidade das soluções encontradas.

O termo *detecção de regiões promissoras* está associado a mecanismos capazes de identificar o potencial positivo de certos subespaços de busca, ao longo do processo evolutivo, gerando a possibilidade de interferência nas estratégias de busca associadas com cada um deles.

A estratégia para se detectar regiões promissoras se torna particularmente importante para o desempenho de um AEH quando a heurística de busca local tem um custo computacional relativamente alto. Aplicar tais heurísticas indiscriminadamente a todos os indivíduos de uma população durante centenas de gerações pode tornar o processo de busca impraticável.

Este trabalho apresentou três estratégias para intensificação de busca em algoritmos evolutivos híbridos. Essas estratégias formam o núcleo de três abordagens: o Treinamento Populacional em Heurísticas (TPH), o *Evolutionary Clustering Search* (ECS) e o Algoritmo Paralelo Hierárquico e Adaptativo com Competição Justa (*APHAC*). As principais contribuições de cada uma dessas abordagens, bem como as próximas etapas deste trabalho são apresentadas a seguir.

6.1 Resumo das contribuições

O TPH tem raízes sedimentadas na dupla avaliação de indivíduos proposta por (Lorena e Furtado, 2001) e consiste em se avaliar a vizinhança da solução que o indivíduo representa, através de heurísticas específicas do problema (heurísticas de treinamento). O indivíduo (esquema ou estrutura) é bem avaliado em função de seu aprendizado acumulado ao longo das gerações. Diz-se que o processo evolutivo realiza um treinamento sobre a população, segundo uma heurística específica.

Quanto maior for a adaptação do indivíduo à heurística de treinamento, maiores são suas chances de participação no processo evolutivo. Esse mecanismo permite que outras características do indivíduo sejam consideradas na composição de sua aptidão, além da função objetivo.

Foram implementadas duas versões do TPH. O $\mathcal{A}$lgoritmo $\mathcal{G}$enético $\mathcal{C}$onstrutivo com treinamento populacional em Heurísticas ($\mathcal{AGC}^H$) trabalha com uma população inicial de esquemas ordenadas através de um *ranking construtivo*. O $\mathcal{A}$lgoritmo de $\mathcal{T}$reinamento $\mathcal{P}$opulacional ($\mathcal{ATP}$) trabalha somente com estruturas ordenadas por outro tipo de *ranking*, chamado de *não-construtivo*.

O $\mathcal{AGC}^H$ foi aplicado a problemas de seqüenciamento (Oliveira e Lorena, 2002a) e problemas SAT. O $\mathcal{ATP}$ foi aplicado a problemas de minimização numérica sem restrição (Oliveira e Lorena, 2002c), a problemas de seqüenciamento de padrões e problemas de escalonamento de tripulação de ônibus (Mauri e Lorena, 2004).

O TPH obteve resultados comparáveis a outros enfoques encontrados na literatura e até melhores, especialmente para problemas de seqüenciamento (Oliveira e Lorena, 2002b,a). Para instâncias desses problemas, foram testadas heurísticas de treinamento diferentes, com destaque para a heurística 2-Opt. Os enfoques baseados em treinamento populacional apresentaram os melhores desempenhos, encontrando novas melhores soluções que passam a ser referências para futuras comparações de desempenho.

Com relação ao TPH para otimização numérica, não foi possível utilizar heurísticas de treinamento mais efetivas tais como, o Simplex de Nelder e Mead, devido ao custo computacional. Foi utilizada, como alternativa, um algoritmo de busca semelhante ao mecanismo de *path-relinking*. Todavia, o conjunto de referência, usado para gerar trajetórias de busca entre soluções, é formado por indivíduos bem-adaptados à heurística de treinamento, sem necessariamente representarem soluções diversificadas dentro do espaço de busca. Essa é a provável maior dificuldade do TPH quando aplicado a espaços contínuos, os quais requerem mecanismos mais efetivos para discriminar os indivíduos a serem melhorados através de heurísticas de busca local.

O *ECS* utiliza um processo de agrupamento de indivíduos para guiar uma busca evolutiva. O processo de agrupamento é usado para gerar um conjunto de centros de *clusters* que se constituem em referências para cada região promissora do espaço de busca.

Diferentemente do TPH, o ECS inclui um raio, calculado com base em um métrica de distância, em torno de cada solução do conjunto de referência (*clusters*), e também um mecanismo que elimina soluções desse conjunto que estejam muito próximas. Além disso, no ECS, as melhorias ocorridas através de um mecanismo chamado de assimilação são incorporadas às soluções do conjunto de referência, tornando-as representativas em termos de diversidade e de qualidade.

Os *clusters* funcionam como janelas flutuantes que enquadram regiões de busca. Grupos de soluções candidatas relativamente próximas podem corresponder a relevantes regiões de atração que devem ser exploradas tão logo sejam detectadas. Além do benefício de se determinar quais regiões devam ser exploradas, o próprio processo de agrupamento, através da assimilação, interfere na forma como região é explorada, determinando um tipo de intensificação de busca nessas regiões. Quando um *cluster* atinge uma certa densidade de indivíduos selecionados, seu centro é usado como ponto de partida para uma busca local.

Foram testados vários algoritmos de assimilação. Pôde-se concluir que a implementação do *path-relinking* utilizada na assimilação por caminho, por si só, não foi suficiente para imprimir uma busca local satisfatória no *ECS*. Outra observação a ser ressaltada é que não houve um método de assimilação vencedor em todas as funções-teste usadas.

O *ECS*, usando assimilação simples, obteve resultados competitivos em treze funções-teste encontradas na literatura, inclusive com desempenho superior aos enfoques de *TPH*, especialmente no que tange, o número de chamadas à função objetivo (CFO). O $\mathcal{AGC}^{2opt}$, por exemplo, tende a realizar 6 vezes mais CFO que o *ECS*. Dessa forma, para os problemas testados, o *ECS* se mostrou capaz de aplicar busca local de forma racional, com ganhos também em termos de tempo de execução.

Foram sugeridas versões paralelas como forma de adequar o TPH e o *ECS* às longas simulações necessárias a problemas mais complexos, sem perda de diversidade populacional. Trilhando nessa direção, foi proposto um *A*lgoritmo *P*aralelo, *H*ierárquico, e *A*daptativo de *C*ompetição justa (*APHAC*), baseado no modelo proposto por (Hu et al., 2002), e aplicado em minimização de funções numéricas, sem restrição, com centenas de variáveis.

A competição justa é alcançada através de uma estratificação de indivíduos em faixas distintas de valores de aptidão, nas quais cada subpopulação evolui independentemente das demais, permitindo a competição entre indivíduos com mesmo perfil de *fitness*. Através dessa estratificação, cria-se um ambiente evolutivo heterogêneo, com estratégias de busca adequadas a cada perfil de indivíduo. Cada subpopulação é mapeada para um processador diferente e eventuais trocas de indivíduos (migração) são efetuadas usando chamadas à biblioteca de comunicação *MPI*.

O *APHAC* implementa todas as características originais previstas em (Hu et al., 2002), tais como adaptação e geração contínua de indivíduos. Ademais, são implementados uma migração assíncrona entre subpopulações, na qual cada subpopulação envia, independen-

temente das demais, indivíduos com perfis superiores para as devidas subpopulações de destino, apropriadas a esses indivíduos.

O ambiente evolutivo heterogêneo dá suporte ao emprego de algoritmos de busca local em indivíduos promissores encontrados na subpopulação elite. Por esse mecanismo, há um fluxo unidirecional com origem na geração contínua de indivíduos, no nível de acesso, e término com uma busca local. Cada indivíduo participa do processo evolutivo em sua subpopulação, gerando descendentes que podem ser movidos para os níveis superiores e assim sucessivamente até serem considerados promissores.

O *APHAC* foi testado em funções-teste numéricas com centenas de variáveis e os resultados foram melhores que um algoritmo padrão seqüencial. Entretanto, não foi possível estabelecer um número ideal de processadores e de probabilidade de busca local para todas as funções testadas. O número de faixas a permitir uma competição equilibrada depende do problema. Mas, pelos resultados obtidos para as quatro funções-teste, pode-se supor que o algoritmo é suficientemente robusto para trabalhar bem com várias possibilidades.

6.2 Trabalhos futuros

As três abordagens apontam para uma direção de pesquisa que visa contribuir para o desenvolvimento de algoritmos adaptativos, aplicáveis a problemas de natureza diferentes, com grande potencial de paralelismo, robustos, competitivos, eficientes tanto em termos de qualidade de soluções quanto em tempo computacional para obtê-las.

O TPH foi aplicado a problemas de escalonamento de tripulação de ônibus, o qual consiste na atribuição da tarefa de condução dos veículos às tripulações, de tal forma que todas as viagens realizadas sejam executadas com o menor custo possível (Mauri e Lorena, 2004). Esse é um caminho natural para todas as abordagens aqui propostas: aproveitar o arcabouço disponibilizado por este trabalho para nortear futuras aplicações.

Além das possibilidades práticas, entende-se que certos aspectos teóricos do TPH ainda podem ser alvo de refinamentos. Por exemplo, a utilização de outras métricas de distâncias para medir a adaptação do indivíduo à heurística de treinamento. Evitar chamadas à função objetivo durante o processo de avaliação do indivíduo pode trazer benefícios ao tempo de execução nas futuras implementações baseadas nesta abordagem.

Algoritmos evolutivos paralelos (AEP's) realmente oferecem dispositivos sobressalentes para superar as limitações das versões seqüenciais. Os AEP's não são apenas extensões dos AE's seqüenciais, mas se constituem verdadeiramente em um novo paradigma evo-

lutivo que é capaz de desempenhar uma busca envolvendo novos operadores evolutivos, novos mecanismos de evolução.

Com relação ao potencial de paralelismo do TPH, a separação de indivíduos em diferentes subpopulações, treinados com diferentes heurísticas, de certa forma, promoveria a evolução de subpopulações em diferentes paisagens de aptidão. Esse aspecto de múltiplas heurísticas de treinamento parece ser uma promissora direção de pesquisa, especialmente, para problemas multi-objetivo.

A migração teria o objetivo de transportar os indivíduos mais aptos de cada subpopulação para outras subpopulações, difundindo material genético diferenciado. Questões relativas a adaptação e aptidão seriam relevantes neste contexto. Indivíduos especializados em determinada heurística teriam alta avaliação, mas poderiam ficar restritos a uma única subpopulação, pois devido a terem fraca adaptação, seriam mal avaliados em outras subpopulações e rapidamente eliminados. No contexto global, o melhor indivíduo seria aquele com maior poder de adaptação.

No caso do *ECS*, existem várias possibilidades que vão bem além de aplicá-lo a novos problemas. Um argumento emergente é substituir o componente AE por uma outra metaheurística capaz de *alimentar* o processo de assimilação com um grande número de soluções diferentes, sem o vício que a perda de diversificação pode causar. Trabalhos futuros devem investigar a geração contínua de indivíduos diretamente para o processo de assimilação, mantendo o conjunto de centros de *clusters* em regiões promissoras do espaço de busca de forma mais robusta. Trabalhos nessa direção podem contribuir para a criação de uma meta-abordagem, o emergente **CS*, que poderia ser estendido em outras direções, associando-se a outras metaheurísticas.

Além disso, o próprio enfoque evolutivo da busca através de agrupamentos permite avanços na direção dos algoritmos evolutivos paralelos (AEP's). Alocação dinâmica de regiões promissoras a diferentes processadores poderia ser organizada hierarquicamente, trabalhando em diferentes níveis de detalhamento do espaço de busca, sob a supervisão de um processador responsável pela coleta das melhores soluções encontradas em cada nível.

A principal dificuldade, no caso do *ECS* paralelo posto dessa forma, está na questão dinâmica do processo de detecção de regiões promissoras por genótipo. Espaços de busca com milhares de variáveis iriam requerer um sofisticado método de alocação de regiões de busca a um número, em geral, fixo de processadores. Todavia, regiões de busca, alocadas dinamicamente para um número indeterminado de processadores é uma idéia apropriada

a aplicações em sistemas de computação em grade (Foster e Kesselman, 1998).

Está em andamento na Universidade Federal do Maranhão (UFMA), um projeto entitulado "Agentes Móveis para Computação em Grade" (*Mobile Agents Technology for Grid Computing - MAG*), do Departamento de Informática[1]. O objetivo geral do projeto é o desenvolvimento de um arcabouço baseado em agentes móveis a ser utilizado em aplicações em uma grade de computadores. Em uma segunda fase desse projeto, relacionada com aplicações, subespaços de busca de problemas de grande porte podem ser divididos entre os computadores da grade, com alocação dinâmica desses subespaços para *agentes evolutivos*.

O problema de detecção de regiões promissoras é deslocado para o contexto de problemas que manipulam um grande número de variáveis, onde o processamento paralelo pode ser um forte aliado por possibilitar a construção de algoritmos mais adequados às longas simulações, necessárias a problemas mais complexos.

O *APHAC*, por sua vez, também abre várias possibilidades de trabalhos futuros. O modelo hierárquico de competição justa divide uniformemente o espaço de *fitness*, atribuindo cada faixa a uma subpopulação. Não há, entretanto, a garantia de que o espaço de busca propriamente dito seja igualmente dividido entre os processadores. A melhor forma de se balancear o fluxo de indivíduos entre as subpopulações, evitando-se espaços super ou sub-explorados, seria implementar-se um estágio de calibragem no qual seriam estimados também indicadores de densidade demográfica que permitiriam a subdivisão e agregação de faixas adjacentes.

Uma segunda possibilidade é a determinação adaptativa do número de níveis. Neste trabalho, um número fixo de processadores é definido *a priori*. Entretanto, pesquisas futuras, podem determinar um suposto número ideal de processadores, dependendo de cada problema, considerando a topologia de cada superfície de função objetivo.

O atual *APHAC* tem uma aplicação imediata. Problemas de otimização de grande-porte são comumente encontrados em aplicações industriais, otimizando milhares de variáveis, tais como problema de detecção de falhas estruturais em aeronaves (Chiwiacowsky et al., 2003). Problemas como esse requerem tempos de execução mais longos, em máquinas de grande poder de processamento, preferencialmente em arquiteturas paralelas.

[1]Informações disponíveis em `http://www.deinf.ufma.br/~mag/`

REFERÊNCIAS BIBLIOGRÁFICAS

Aiex, R. M.; Binato, S.; Resende, M. G. C. Parallel GRASP with path-relinking for job shop scheduling. **Parallel Computation**, v. 29, n. 4, p. 393–430, 2003. 122

Alba, E.; Troya, J. M. A survey of parallel distributed genetic algorithms. **Complexity**, v. 4, n. 4, p. 31–52, 1999. 156

Alfandari, L.; Plateau, A.; Tolla, P. Path-relinking algorithm for the generalized assignment problem. In: Resende, M.; de Sousa, J. ed. **Metaheuristics: Computer decision-making**. Kluwer Academic Publishers, 2003. Combinatorial Optimization Book Series, p. 1–18. 122

Altenberg, L. The schema theorem and price's theorem. In: Foundations of Genetic Algorithms. **Proceedings**. Estes Park, Colorado, USA: Morgan Kaufmann Publishers, 1995. v. 3, p. 23–49. 37

Areibi, S. Memetic algorithms for VLSI physical design: implementation issues. In: Workshop on Memetic Algorithms (WOMA), 2, California (USA). **Proceedings**. San Francisco: Morgan Kaufmann, 2001. p. 140–145. 29, 46, 48

Bäck, T. Selective pressure in evolutionary algorithms: a characterization of selection mechanisms. In: IEEE Conference on Evolutionary Computation(ICEC'94), 1, Orlando, Florida (USA). **Proceedings**. IEEE Press, 1994. p. 57–62. 124

Baldwin, J. M. A new factor in evolution. **The American Naturalist**, v. 30, p. 441–451, 1896. Reprint in: Adaptive individual in evolving populations: models and algorithms, R. K. Belew and M. Mitchell (eds.), 1996, pp. 59–80, Reading, MA: Addison Wesley. 48

Bazaraa, M. S.; Jarvis, J. J.; Sherali, H. D. **Linear programming and network flows (2nd ed.)**. John Wiley & Sons, Inc., 1990. 53

Becceneri, J. C. **O problema de sequenciamento de padrões para minimização do número máximo de pilhas abertas em ambientes de corte industriais**. São José dos Campos, SP, Brasil. Tese – Instituto Tecnológico de Aeronáutica (ITA), 1999. 51

Bersini, H.; Dorigo, M.; Langerman, S.; Seront, G.; Gambardella, L. Results of the first international contest on evolutionary optimisation (1st ICEO). In: IEEE International Conference on Evolutionary Computation (ICEC'96), 3, Nagoya (Japan). **Proceedings**. IEEE Press, 1996. p. 611–615. 55

Birru, H. K.; Chellapilla, K.; Rao, S. Local search operators in fast evolutionary programming. In: Congress on Evolutionary Computation(CEC'99), Washington (USA). **Proceedings**. Washington: IEEE, 1999. v. 2, p. 1506–1513. 29, 32, 46, 114

Blum, C.; Roli, A. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. **ACM Computing Surveys**, v. 35, n. 3, p. 268–308, 2003. 28, 41

Carmona, E. A. Parallelizing a large scientific code - methods, issues, and concerns. In: ACM/IEEE conference on Supercomputing, Reno(United States), 1989. **Proceedings**. ACM Press, 1989. p. 21–31. 153

Chelouah, R.; Siarry, P. Genetic and Nelder-Mead algorithms hybridized for a more accurate global optimization of continuous multiminima functions. **European Journal of Operational Research**, v. 148, n. 2, p. 335–348, 2003. 31, 115, 139

Chiwiacowsky, L.; Campos Velho, H.; Gasbarri, P. The damage identification problem: a hybrid approach. In: Thematic Congress on Dynamics and Control (DINCON 2003), 2, São José dos Campos (SP), Brazil. **Proceedings**. São José dos Campos (SP): DINCON, 2003. p. 1393–1402. 181, 188

Coello Coello, C. A. An updated survey of GA-based multiobjective optimization techniques. **ACM Computing Surveys**, v. 32, n. 2, p. 109–143, June 2000. 42, 154

Cook, S. The complexity of theorem-proving procedures. In: ACM Symposium on Theory of Computing(STOC1971), 3, Shaker Heights, Ohio, United States. **Proceedings**. ACM Press, 1971. p. 151–158. 53

Croes, G. A method for solving travelling salesman problems. **Operations Research.**, v. 6, p. 791–812, 1958. 73

Davis, L. **Handbook of genetic algorithms**. New York: Van Nostrand Reinhold Computer Library, 1991. 39, 124, 169

De Jong, K. A. **An analysis of the behavior of a class of genetic adaptive systems**. Ann Arbor, MI. Tese – University of Michigan, 1975. 30, 37, 54, 55

De Oliveira, P. P. B.; Gatto, R. C. Using coevolving genetic algorithms to find the roots of a function. In: International Fuzzy Systems Association World Congress (IFSA-95), 6, São Paulo, Brasil. **Proceedings**. São Paulo: IFSA, 1995. v. 1, p. 165–168. 124

Deb, K.; Goldberg, D. E. An investigation of niche and species formation in genetic function optimization. In: International Conference on Genetic Algorithms (ICGA'89), 3, Virginia, USA. **Proceedings**. George Mason University, Fairfax: Morgan Kaufmann Publishers Inc., 1989. p. 42–50. 115

Digalakis, J.; Margaritis, K. An experimental study of benchmarking functions for genetic algorithms. **International Journal of Computer Mathematics**, v. 79, n. 4, p. 403–416, 2002. 55, 133, 134, 135

Dorigo, M.; Gambardella, L. M. Ant colony system: a cooperative learning approach to the traveling salesman problem. **IEEE Transactions on Evolutionary Computation**, v. 1, n. 1, p. 53–66. <http://citeseer.ist.psu.edu/article/dorigo96ant.html>, April 1997. 28

Eiben, A. E.; Hinterding, R.; Michalewicz, Z. Parameter control in evolutionary algorithms. **IEEE Transaction on Evolutionary Computation**, v. 3, n. 2, p. 124–141. <http://citeseer.ist.psu.edu/eiben00parameter.html>, 1999. 29, 30, 125

Eshelman, L. J.; Schaffer, J. D. Real-coded genetic algorithms and interval schemata. In: Foundations of Genetic Algorithms. **Proceedings**. San Mateo, CA: Morgan Kaufmann Publishers, 1993. v. 2, p. 187–202. 103, 124, 171

Faggioli, E.; Bentivoglio, C. Heuristic and exact methods for the cutting sequencing problem. **European Journal of Operational Research**, v. 110, n. 3, p. 564–575, 1998. 76, 77, 78, 87, 88

Falqueto, J.; Barreto, J. M.; Borges, P. S. S. Amplification of perspectives in the use of evolutionary computation. In: International Symposium on Bio-Informatics and Biomedical Engineering (BIBE2000), Virginia (USA). **Proceedings**. Arlington: IEEE Press, 2000. p. 150–157. 37, 154

Feltl, H.; Raidl, G. R. An improved hybrid genetic algorithm for the generalized assignment problem. In: ACM Symposium on Applied Computing (SAC2004), 19, Nicosia, Cyprus. **Proceedings**. ACM Press, 2004. p. 990–995. 47

Feo, T.; Resende, M. A probabilistic heuristic for a computationally difficult set covering problem. **Operations Research Letters**, v. 8, n. 2, p. 67–71, 1989. 28, 41

Festa, P. Greedy randomized adaptive search procedures. **AIROnews**, v. 7, n. 4, p. 7–11, 2003. 122

Fink, A.; Voss, S. Applications of modern heuristic search methods to pattern sequencing problems. **Computers and Operations Research**, v. 26, n. 1, p. 17–34, 1999. 50

Fogel, L. J.; Owens, A. J.; Walsh, M. J. Artificial intelligence through a simulation of evolution. In: Biophysics and Cybernetic Systems: Proceedings of 2nd Cybernetic Sciences Symposium. **Proceedings**. Washington, D.C.: Spartan Books, 1965. p. 131–155. 27, 39

Fonlupt, C.; Robilliard, D.; Preux, P.; Talbi, E. G. Fitness landscapes and performance of metaheuristics. In: Voss, S.; Martello, S.; Osman, I.; Roucairol, C. ed. **Metaheuristics: Advances and Trends in Local Search Paradigms for Optimization**. Kluwer Academic Press, 1999. Cap. 18. 74

Foster, I.; Kesselman, C. **The grid: BluePrint for new computing infrastructures**. Morgan Kaufmann Publishers Inc., 1998. 150, 188

Freisleben, B.; Merz, P. A genetic local search algorithm for solving symmetric and asymmetric salesman problems. In: IEEE International Conference on Evolutionary Computation (ICEC'96), 3, Nagoya(Japan). **Proceedings**. IEEE Press, 1996. p. 616–621. 47

Garey, M.; Johnson, D. **Computers and intractability: A guide to the theory of NP-Completeness**. San Francisco: W.H. Freeman and Company, 1979. 53

Gent, I. **On the stupid algorithm for satisfiability**. Technical Report APES-03-1998. Strathclyde University, 1998. 96, 97

Glover, F. Tabu search and adaptive memory programing. Advances, applications and challenges. In: Barr, R. S.; Helgason, R. V.; Kennington, J. L. ed. **Interfaces in Computer Science and Operations Research**. Norwell, MA, USA: Kluwer Academic Publishers, 1996. v. 7, Cap. 1, p. 1–75. 28, 41

——. A template for scatter search and path relinking. **Selected Papers from the Third European Conference on Artificial Evolution**. Springer-Verlag, 1998. p. 3–54. 47, 104, 115, 121, 122, 133, 138

Goldbarg, M. C.; Gouvea, E. F. Extra-intracellular transgenetic algorithm. In: Genetic and Evolutionary Computation Conference (GECCO-2001), San Francisco, California, USA. **Proceedings**. Morgan Kaufmann, 2001. p. 115–121. 48

Goldberg, D.; Korb, B.; Deb, K. Messy genetic algorithms: motivation, analysis, and first results. **Complex Systems**, v. 3, n. 5, p. 493–530, 1989. 59, 61

Goldberg, D. E. **Genetic algorithms in search, optimization and machine learning**. Reading, Massachusetts: Addison-Wesley Publishing Company, 1989. 30, 37, 38, 39, 43, 49, 59, 83, 124, 130

Goldberg, D. E.; Deb, K.; Kargupta, H.; Harik, G. Rapid, accurate optimization of difficult problems using fast messy genetic algorithms. In: International Conference on Genetic Algorithms(ICGA'93), 5, San Mateo(CA). **Proceedings**. San Mateo: Morgan Kaufman, 1993. p. 56–64. 60, 61

Goldberg, D. E.; Richardson, J. Genetic algorithms with sharing for multimodal function optimization. In: International Conference on Genetic Algorithms and their application, 2, Massachusetts (USA). **Proceedings**. Lawrence Erlbaum Associates, Inc., 1987. p. 41–49. 115

Golumbic, M. **Algorithmic graph theory and perfect graphs**. New York: Academic Press, 1980. 51

Gottlieb, J.; Marchiori, E.; Rossi, C. Evolutionary algorithms for the satisfiability problem. **Evolutionary Computation**, v. 10, n. 1, p. 35–50, 2002. 98

Gropp, W.; Lusk, E.; A., S. **Using MPI: portable parallel programming with the message passing interface**. 2. ed. Cambridge, USA: The MIT Press, 1999. 153

Gu, J.; Purdom, P. W.; Franco, J.; Wah, B. W. Algorithms for the satisfiability (SAT) problem: a survey. In: Du, D.-Z.; Gu, J.; Pardalos, P. ed. **The satisfiability problem: theory and applications**. American Mathematical Society, 1997. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, p. 19–152. 52, 53, 96

Guanqi, G.; Shouyi, Y. Evolutionary parallel local search for function optimization. **IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)**, v. 33, n. 6, p. 864–876, 2003. 174

Hawick, K. A.; Coddington, P. D.; James, H. A. Distributed frameworks and parallel algorithms for processing large-scale geographic data. **Parallel Computation**, v. 29, n. 10, p. 1297–1333, 2003. 31

Haykin, S. **Neural networks**. Macmillan College Publishing Company, 1994. 63

Haynes, T. Voting for Schemata, 1997. <http: //www.umsl.edu/ haynes/vote_ga.ps>. 49, 60

Herrera, F.; Lozano, M.; Verdegay, J. L. Tackling Real-Coded Genetic Algorithms: Operators and Tools for Behavioural Analysis. **Artif. Intell. Rev.**, v. 12, n. 4, p. 265–319, 1998. 103

Holland, J. H. **Adaptation in natural and artificial systems**. Ann Arbor, MI. Tese – University of Michigan, 1975. 27, 30, 37, 44, 49, 113

Hooke, R.; Jeeves, T. A. "Direct search" solution of numerical and statistical problems. **Journal of the ACM**, v. 8, n. 2, p. 212–229, 1961. 128, 130

Hoos, H. H.; Stutzle, T. Local search algorithms for SAT: an empirical evaluation. **Journal of Automated Reasoning**, v. 24, n. 4, p. 421–481. <http://citeseer.ist.psu.edu/hoos99local.html>, 2000. 96

Hu, J.; Goodman, E. D.; Seo, K.; Pei, M. Adaptive hierarchical fair competition (AHFC) model for parallel evolutionary algorithms. In: Genetic and Evolutionary Computation Conference (GECCO 2002), San Francisco (USA). **Proceedings**. New York: Morgan Kaufmann Publishers, 2002. p. 772–779. 34, 151, 152, 158, 159, 160, 161, 162, 163, 168, 180, 185

Jelasity, M.; Ortigosa, P.; García, I. UEGO, an abstract clustering technique for multimodal global optimization. **Journal of Heuristics**, v. 7, n. 3, p. 215–233, 2001. 115

Jones, T. **Evolutionary algorithms, fitness landscapes and search**. Tese – University of New Mexico, 1995. 44, 45, 46, 142

Kashiwabara, T.; Fujisawa, T. NP-Completeness of the problem of finding a minimum clique number interval graph containing a given graph as subgraph. In: IEEE International Symposium of Circuits and Systems (ISCAS'79), Tokyo, Japan. **Proceedings**. Tokyo: IEEE, 1979. p. 657–660. 51

Kautz, H. A.; Selman, B. Planning as satisfiability. In: European Conference on Artificial Intelligence (ECAI'92), 10, Austria. **Proceedings**. Wiley, 1992. p. 359–363. <http://citeseer.ist.psu.edu/kautz92planning.html>. 52

Kirkpatrick, S.; Gelatt, C. D.; Vecchi, M. P. Optimization by simulated annealing. **Science**, v. 220, n. 4598, p. 671–680. <http://citeseer.ist.psu.edu/kirkpatrick83optimization.html>, May 1983. 28, 41

Knjazew, D.; Goldberg, D. E. Large-Scale permutation optimization with the ordering messy genetic algorithm. In: Parallel Problem Solving from Nature (PPSN), 6, Paris, France. **Proceedings**. Berlin: Springer, 2000. p. 631–640. 60

Koza, J. R. **Genetic programming: on the programming of computers by natural selection**. Cambridge, MA: MIT Press, 1992. 27, 39

Kumar, V.; Grama, A.; Gupta, A.; Karypis, G. **Introduction to parallel computing**. Addison Wesley, 2003. 152, 155, 176

Laguna, M.; Martí, R. The OptQuest callable library. In: Voss, S.; Woodruff, D. L. ed. **Optimization Software Class Libraries**. Kluwer Academic Publishers, 2002. p. 193–218. 138

Lamarck, J. B. Of the influence of the environment on the activities and habits of animals, and the influence of the activities and habits of these living bodies in modifying their organization and structure. In: Lamarck, J. B. ed. **Zoological Philosophy**. London: Macmillan, 1914. p. 106–127. Reprint in: Adaptive individuals in evolving populations: models and algorithms, ed. R. K. Belew and M. Mitchell. 36, 47

Leblanc, B.; Lutton, E. Bitwise regularity and GA-hardness. In: IEEE International Conference on Evolutionary Computation (ICEC'98), 5, Anchorage, Alaska. **Proceedings**. IEEE Press, 1998. 38

Li, J.-P.; Balazs, M. E.; Parks, G. T.; Clarkson, P. J. A species conserving genetic algorithm for multimodal function optimization. **Evolutionary Computation**, v. 10, n. 3, p. 207–234, 2002. 113, 115

Linhares, A. **Industrial pattern sequencing problems: some complexity results and new local search models**. São José dos Campos, SP, Brasil. Tese – Instituto Nacional de Pesquisas Espaciais (INPE), 2002. 51, 65, 73, 87, 88, 122, 123, 142

Linhares, A.; Yanasse, H.; Torreão, J. Linear gate assignment: a fast statistical mechanics approach. **IEEE Transactions on Computer-Aided Designed of Integrated Circuits and Systems**, v. 18, n. 12, p. 1750–1758, 1999. 72, 85, 86, 87

Lorena, L.; Furtado, J. Constructive genetic algorithm for clustering problems. **Evolutionary Computation**, v. 9, n. 3, p. 309–327, 2001. 60, 61, 62, 67, 78, 79, 80, 83, 85, 107, 183

Mauri, G. R.; Lorena, L. A. N. Driver scheduling generation using a population training algorithm. In: Brazilian Workshop on Evolutionary Computation (BEC'2004), 1, São Luís, MA (Brasil). **Proceedings**. IEEE, 2004. Accepted. 71, 184, 186

Medeiros, F. L. L.; Guimarães, L. N. F. Hybrid Genetic Algorithm Applied to the Steady State Determination of Nonlinear Dynamic Systems. In: Brazilian Workshop on Evolutionary Computation (BEC'2004), 1, São Luís MA (Brasil). **Proceedings**. IEEE, 2004. Accepted. 157

Mendes, A.; Linhares, A. A multiple population evolutionary approach to gate matrix layout. **International Journal of Systems Science**, v. 35, n. 1, p. 13–23, 2004. 48, 90

Merz, P. The compact memetic algorithm. In: Workshop on Memetic Algorithms (WOMA), 4, Chicago (USA). **Proceedings**. Morgan Kaufmann, 2003. p. 234–241. 47

Merz, P.; Freisleben, B. Fitness landscapes and memetic algorithm design. In: Corne, D.; Dorigo, M.; Glover, F. ed. **New Ideas in Optimization**. London: McGraw-Hill, 1999. p. 245–260. 43, 44

Möhring, R. Graph problems related to gate matrix layout and PLA folding. **Computational Graph Theory**, v. 7, p. 17–51, 1990. 51

Michalewicz, Z. **Genetic algorithms + data structures = evolution programs**. Berlin: Springer-Verlag, 1996. 387 p. 39, 42, 103, 124, 138, 171

Miki, M.; Hiroyasu, T.; Kaneko, M. A parallel genetic algorithm with distributed environment scheme. In: International Conference on Parallel and Distributed Processing Techiniques and Applications. **Proceedings**. Morgan Kaufmann, 2000. v. 2, p. 619–625. 174

Mitchell, M. **An Introduction to Genetic Algorithms**. MIT Press, 1998. 29

Mitchell, T. M. **Machine learning**. New York: McGraw-Hill, 1997. 27

Moscato, P. Memetic algorithms: a short introduction. In: Corne, D.; Dorigo, M.; Glover, F. ed. **New Ideas in Optimization**. London: McGraw-Hill, 1999. p. 219–234. 47, 48

Nagano, M. S.; Ruiz, R.; Lorena, L. A. N. A constructive genetic algorithm for permutation flowshop scheduling. **European Journal of Operational Research**, 2004. Accepted. 61

Navarro, P. L. K. G.; De Oliveira, P. P. B.; Ramos, F. M.; Campos-Velho, H. F. Magnetotelluric inversion using problem-specific genetic operators. In: Genetic and

Evolutionary Computation Conference(GECCO1999), Orlando, Florida, USA. **Proceedings**. Morgan Kaufmann, 1999. v. 2, p. 1580–1587. 31, 47

Nelder, J.; Mead, R. A simplex method for function minimization. **Computer Journal**, v. 7, n. 23, p. 308–313, 1965. 103, 115, 129, 174

Nowostawski, M.; Poli, R. Parallel genetic algorithm taxonomy. In: International conference on knowledge-based intelligent information engineering systems (KES'99), 3, Adelaide (AU). **Proceedings**. Adelaide: IEEE, 1999. p. 88–92. 152, 155, 156

Oliveira, A. C. M.; Lorena, L. A. N. 2-Opt population training for minimization of open stack problem. In: Brazilian Symposium on Artificial intelligence(SBIA2002), 16, Recife(Brasil), 2002. **Advances in Artificial Intelligence**. Berlin: Springer-Verlag, 2002a. v. 2507, p. 313–323. 87, 110, 184

——. A constructive genetic algorithm for gate matrix layout problems. **IEEE Transaction on Computer-Aided Designed of Integrated Circuits and Systems**, v. 21, n. 8, p. 969–974, 2002b. 60, 80, 85, 87, 110, 184

——. Real-coded evolutionary approaches to unconstrained numerical optimization. In: Congress of Logic Applied to Technology (LAPTEC2002), 3, São Paulo, Brazil. **Advances in Logic, Artificial Intelligence and Robotics**. Plêiade, 2002c. v. 2, p. 10–15. 66, 71, 101, 106, 184

——. Detecting promising areas by evolutionary clustering search. In: Brazilian Symposium on Artificial Intelligence (SBIA2004), 17, São Luís MA (Brasil), 2004. **Advances in Artificial Intelligence**. Berlin: Springer-Verlag, 2004. Accepted. 126, 130

Oliveira, A. C. M.; Lorena, L. A. N.; Stephany, S.; Preto, A. J. An adaptive hierarchical fair competition genetic algorithm for numerical optimization. In: Brazilian Workshop on Evolutionary Computation (BEC'2004), 1, São Luís MA (Brasil). **Proceedings**. IEEE, 2004. Accepted. 163

Oliveira, A. C. M.; Nascimento, E. Uma contribuição ao estudo de algoritmos evolutivos fuzzy. In: Ibero-American Conference on AI on Progress in Artificial Intelligence (IBERAMIA'98), 6, Lisboa, Portugal. **Proceedings**. Berlin: Springer, 1998. v. 1484 of **LNAI**, p. 175–186. 30

Pacheco, P. **Parallel Programming with MPI**. San Francisco, USA: Morgan Kaufmann Publishers, 1996. 153

Pedrycz, W. **Computational intelligence: an introduction**. CRC Press, Inc., 1997. 284 p. 27

Preux, P.; Talbi, E.-G. Towards hybrid evolutionary algorithms. **International Transactions in Operational Research**, v. 6, n. 6, p. 557–570, 1999. 73, 74, 122

Rechenberg, I. **Cybernetic solution path of an experimental problem**. Farnborough, Hants: Royal Aircraft establishment, Library translation No. 1122, 1965. 27, 39

Rees, J.; Koehler, G. J. An investigation of GA performance results for different cardinality alphabets. In: Davis, L. D.; De Jong, K.; Vose, M. D.; Whitley, L. D. ed. **Evolutionary Algorithms**. New York: Springer, 1999. p. 191–206. <http://citeseer.ist.psu.edu/rees97investigation.html>. 37, 151

Reeves, C. R.; Yamada, T. Genetic algorithms, path relinking, and the flowshop sequencing problem. **Evolutionary Computation**, v. 6, n. 1, p. 45–60, 1998. 122

Reeves, C.R. Landscapes, operators and heuristic search. **Annals of Operations Research**, v. 86, p. 473–490, 1999. 44, 65, 120, 122, 123

Resende, M.; Ribeiro, C. GRASP and path-relinking: Recent advances and applications. In: Metaheuristics International Conference (MIC2003), 5, Kyoto (Japan). **Proceedings**. Kluwer Academic Publishers, 2003. p. T6–1 – T6–6. 122

Ribeiro Filho, G.; Lorena, L. A. N. Constructive genetic algorithm and column generation: an application to graph coloring. In: Conference of the Association of Asian-Pacific Operations Research Societies (APORS'2000), 5, Cingapura. **Proceedings**. APORS, 2000a. <http://www.ise.nus.edu.sg/proceedings/apors2000/fullpapers/02-03-fp.htm>. 61

——. Constructive genetic algorithm for machine-part cell formation problem. In: Buildings Competencies for International Manufacturing - Perspectives for developing countries. **Proceedings**. UFRGS/FEENG, 2000b. p. 340–348. 61

——. A constructive evolutionary approach to school timetabling. In: Applications of Evolutionary Computing. EvoWorkshops2001: EvoCOP, EvoFlight, EvoIASP, EvoLearn, and EvoSTIM. **Proceedings**. Como, Italy: Springer-Verlag, 2001. v. 2037, p. 130–139. <http://citeseer.ist.psu.edu/377914.html>. 61, 85

Rodriguez-Tello, E.; Torres-Jimenez, J. ERA: an algorithm for reducing the epistasis of SAT problems. In: Genetic and Evolutionary Computation Conference

(GECCO-2003), Chicago, IL, USA. **Proceedings**. Springer, 2003. v. 2724 of **LNCS**, p. 1283–1294. 39

Rosca, J. P.; Ballard, D. H. **Genetic programming with adaptive representations**. Technical Report 489. University of Rochester, Computer Science Department, Rochester, NY, USA, Fev 1994. 49

Rossi, C.; Marchiori, E.; Kok, J. N. An adaptive evolutionary algorithm for the satisfiability problem. In: ACM Symposium on Applied Computing (SAC2000), 15, Villa Olmo, Como, Italy. **Proceedings**. ACM Press, 2000. p. 463–469. 98, 99, 100

Russell, S.; Norvig, P. **Artificial intelligence: a modern approach**. 2. ed. Englewood Cliffs: Prentice-Hall, 2003. 27

Schuurmans, D.; Southey, F. Local search characteristics of incomplete SAT procedures. **Artificial Intelligence**, v. 132, n. 2, p. 121–150. <http://citeseer.ist.psu.edu/schuurmans00local.html>, 2001. 96

Schwefel, H. P. **Numerical optimization of computer models**. Chichester: Wiley and Sons, 1981. 55

——. **Evolution and optimum seeking**. New York: Wiley, 1995. 130

Scott, S. D.; Lesh, N.; Klau, G. W. Investigating human-computer optimization. **Proceedings of the SIGCHI conference on Human factors in computing systems**. ACM Press, 2002. p. 155–162. 31

Selman, B.; Kautz, H. A.; Cohen, B. Noise strategies for improving local search. In: National Conference on Artificial Intelligence (AAAI'94), 20, Seattle, Washington. **Proceedings**. Seattle: AAAI Press, 1994. p. 337–343. <http://citeseer.ist.psu.edu/selman94noise.html>. 97, 98

Seo, D.-I.; Kim, Y.-H.; Moon, B.-R. New entropy-based measures of gene significance and epistasis. In: Genetic and Evolutionary Computation Conference(GECCO-2003), Chicago, IL, USA. **Proceedings**. Chicago: Springer-Verlag, 2003. v. 2724 of **LNCS**, p. 1345–1356. 39

Silva, L. R. S. **Aprendizagem participativa em agrupamento nebuloso de dados**. Campinas. Dissertação – Universidade Estadual de Campinas, Março 2003. 119

Silverman, B. **Density Estimation for Statistics and Data Analysis**. London: Chapman and Hall, 1986. 166

Streichert, F.; Stein, G.; Ulmer, H.; Zell, A. A clustering based niching method for evolutionary algorithms. In: Genetic and Evolutionary Computation Conference(GECCO-2003), Chicago, IL, USA. **Proceedings**. Chicago: Springer-Verlag, 2003. v. 2723 of **LNCS**, p. 644–645. 116

Syswerda, G. Uniform crossover in genetic algorithms. In: International Conference on Genetic Algorithms(ICGA'89), 3, Virginia, USA. **Proceedings**. George Mason University, Fairfax: Morgan Kaufmann, 1989. p. 2–9. 83

Topchy, A.; Lebedko, O.; Miagkikh, V. Fast learning in multilayered networks by means of hybrid evolutionary and gradient algorithms. In: International Conference on Evolutionary Computation and its Applications (EvCA'96), 1, Moscow, Russia. **Proceedings**. Russian Academy of Sciences, 1996. p. 390–398. <http://web.cps.msu.edu/ miagkikh/web/6.ps.gz>. 60

Uzunian, A.; Birner, E. **Biologia**. 2. ed. Harbra, 2004. 27

Yager, R. A model of participatory learning. **IEEE Transactions on Systems, Man and Cybernetics**, v. 20, n. 5, p. 1229–1234, 1990. 119

Yamamoto, M.; Lorena, L. A. N. A constructive genetic approach to point-feature cartographic label placement. In: Metaheuristics International Conference (MIC2003), 5, Kyoto(Japan). **Proceedings**. Kluwer Academic Publishers, 2003. p. 84–1 – 84–7. 61

Yanasse, H. Minimization of open orders-polynomial algorithms for some special cases. **Pesquisa Operacional**, v. 16, n. 1, p. 1–26, 1996. 51

Yen, J.; Lee, B. A simplex genetic algorithm hybrid. In: IEEE International Conference on Evolutionary Computation (ICEC'97), 4, Indianapolis (USA). **Proceedings**. Indianapolis: IEEE Press, 1997. p. 175–180. 29, 31, 46, 49

Zadeh, L. A. Soft Computing and Fuzzy Logic. **IEEE Software**, v. 11, n. 6, p. 48–56, 1994. 27, 30