Detecção Multiusuário Utilizando Algoritmos Heurísticos Evolucionários e de Busca Local

Multiuser Detection Using Evolutionary and Local Search Heuristic Algorithms

Fernando Ciriaco Dias Neto[1]; Taufik Abrão[2]; Paul Jean E. Jeszensky[3]

Resumo

Neste trabalho, são apresentados os principais algoritmos heurísticos baseados na teoria da evolução genética e de busca local. Esses algoritmos são aplicados ao problema da detecção multiusuário (MuD) para sistemas DS/CDMA em canais com desvanecimento Rayleigh Plano. A eficiência destes algoritmos é comparada através do compromisso desempenho versus complexidade computacional. A complexidade computacional é determinada em termos do número de operações necessárias para se alcançar o desempenho ML (ou muito próximo). Analisa-se, também, a perda de desempenho quando há ocorrência de erros na estimativa do canal.
Palavras-chaves: Detecção multiusuário. Algoritmos heurísticos. Evolucionários. Busca local. Complexidade computacional.

Abstract

The characteristics of the main heuristic algorithms based on genetic evolution theory and local search, applied to a DS/CDMA multi-user detection problem in Flat Rayleigh fading channel, are shown. The algorithms comparisons through the performance versus computational complexity tradeoff are carried out. The computational complexity is established in terms of the number of operations to reach the Maximum Likelihood (ML) performance. The estimation error effects on the performance are also considered.
Key words*:* Multiuser detection. Heuristic algorithms. Evolutionary. Local search. Computational complexity.

[1] Aluno do 5º ano do curso de Engenharia Elétrica pela Universidade Estadual de Londrina. Bolsista de Iniciação Científica P.I.B.I.C/CNPq na área de Telecomunicações. Comunicações sem fio, sistemas quase-síncronos e Detectores Multiusuário para sistemas DS-CDMA. E-mail: fernandociriaco@onda.com.br

[2] Doutor em Engenharia Elétrica em 2001, pela Escola Politécnica da Universidade de São Paulo. Professor adjunto do Departamento de Engenharia Elétrica da Universidade Estadual de Londrina, PR. E-mail: taufik@uel.

[3] Doutor em Engenharia Elétrica-Sistemas Eletrônicos e Livre Docente na área de Telecomunicações, pela Escola Politécnica da USP em 1992, respectivamente. Professor da EPUSP desde 1978, na área de Telecomunicações, E-mail: pjj@lcs.poli.usp.br.

Introdução

O sinal recebido por um detector convencional (CD), constituído de um banco de filtros casados (MFB), em um sistema DS-CDMA (*Direct Sequence/Code-Division Multiple Access*) síncrono em canal Rayleigh Plano não pode ser recuperado de forma ótima, pois este é afetado pela interferência de múltiplo acesso (MAI) e pelo efeito near-far (*NFR*), resultando em um sistema cuja capacidade está bem abaixo da capacidade do canal (MOSHAVI, 1996; VERDÚ, 1998).

Nas últimas duas décadas, uma grande variedade de detectores multiusuário (MuD) foram propostos na literatura no intuito de melhorar o desempenho obtido com o detector convencional. O melhor desempenho possível é obtido com o detector ótimo, porém às custas de uma complexidade computacional que cresce exponencialmente com o número de usuários, o que o torna inviável para implementação (VERDÚ, 1998).

Fazem-se necessárias, portanto, investigações na área de detecção multiusuário sub-ótimos que atendam aos critérios de alto desempenho e baixa complexidade.

Este trabalho faz uma síntese comparativa dos principais algoritmos heurísticos, baseados na teoria da evolução genética e de busca local, aplicáveis ao problema da detecção multiusuário.

Adicionalmente, esta análise visa a comparar a complexidade computacional destes algoritmos através do número de operações computacionais que cada receptor necessita para a demodulação.

Este trabalho divide-se em 7 seções. Inicialmente, a seção 2 trata da descrição do modelo matemático de um sistema DS/CDMA em canal Rayleigh Plano síncrono. Na seção 3, é apresentada a descrição do problema da detecção MuD a ser minimizado. Na seção 4, são apresentadas as características e os pseudocódigos dos principais algoritmos heurísticos evolucionários e de busca local aplicados ao problema da detecção MuD. Na seção 5, são obtidos resultados numéricos para diferentes condições do sistema via simulação Monte-Carlo. Na seção 6, são determinadas as expressões analíticas para a complexidade computacional dos algoritmos, encontrando o número de operações que cada detector necessitou para a demodulação através dos resultados numéricos obtidos na seção 5. Por fim, na seção 7, são apresentadas as principais conclusões deste estudo.

Modelo do Sistema

Em um sistema DS-CDMA com modulação BPSK (*Binary Phased Shift Keying*) em canal com desvanecimento, o sinal que chega ao receptor pode ser descrito, em banda base, por:

$$r(t)=\sum_{k=1}^{K} A_k b_k s_k (t-\tau_k)*h(t)+\eta(t) \qquad (1)$$

onde K é o número de usuários ativos no sistema, $t\in[0,T_b]$ e T_b é o período de bit, $A_k=\sqrt{E_k}$ é a amplitude e E_k é a energia do sinal transmitido do k-ésimo usuário, $b_k\in\{-1,+1\}$ é o bit de informação transmitido, s_k é a seqüência de assinatura atribuída ao k-ésimo usuário, $h(t)$ é a resposta impulsiva do canal e $\eta(t)$ representa o ruído AWGN.

Considerando um sistema síncrono, $\tau_k=0$, e um canal não seletivo em freqüência, caracterizando um canal Rayleigh Plano, ou seja:

$$h(t)=c_k\delta(t)=\beta_k e^{j\phi_k}\delta(t) \qquad (2)$$

onde c_k indica o coeficiente do canal para o k-ésimo usuário, β_k denota o módulo de c_k com uma distribuição Rayleigh e ϕ_k a fase de c_k com uma distribuição uniforme entre [0, 2π), pode-se reescrever a equação (1) como sendo:

$$r(t)=\sum_{k=1}^{K} A_k b_k c_k s_k(t)+\eta(t) \qquad (3)$$

O sinal à saída de um banco de filtros casados (detector convencional) pode ser escrito como:

$$y_k = \int_0^{T_b} r(t)s_k(t)dt = A_k b_k c_k + \sum_{j \neq k} A_j b_j c_j \rho_{k,j} + \eta_k \quad (4)$$

onde $\rho_{k,j}$ denota o *k*,*j*-ésimo elemento da matriz de correlação R, e η_k é o ruído AWGN filtrado para o *k*-ésimo usuário.

Descrição do Problema

Em Verdú (1998, p.162) foi mostrado que uma solução ótima para recuperar os bits de informação de (3) consiste em estimar a informação transmitida utilizando a saída de um detector de máxima verossimilhança (ML):

$$\hat{b} = \arg\left\{ \max_{b \in \{+1,-1\}^K} 2y^T c^H Ab - b^T cARAc^H b \right\} \quad (5)$$

onde o operador $(\cdot)^H$ indica transposta conjugada. O detector multiusuário ótimo consiste na busca do melhor vetor de bits de dados em um conjunto com todas as possibilidades, ou seja, é um problema com combinação NP-completa, no qual os algoritmos tradicionais são ineficientes.

Métodos heurísticos aplicados a esse tipo de problema têm-se tornado atraentes, pois permitem obter soluções ótimas (ou próximas) para problemas de combinação em um curto tempo e espaço de busca.

Sob a restrição de um espaço de busca, todos os algoritmos heurísticos buscam melhores soluções seguindo uma função objetivo, capaz de quantificar a tendência de melhoria em relação à solução ótima. Essa função é chamada de função custo (*fitness value*), e no contexto da detecção MuD é dada pela função de verossimilhança:

$$f(s) = 2y^T c^H As - s^T cARAc^H s \quad (6)$$

Cada algoritmo heurístico aplicado ao problema da detecção MuD busca maximizar (6) através de um vetor de bits candidatos cujo desempenho médio correspondente aproxima-se daquele obtido com um detector ML.

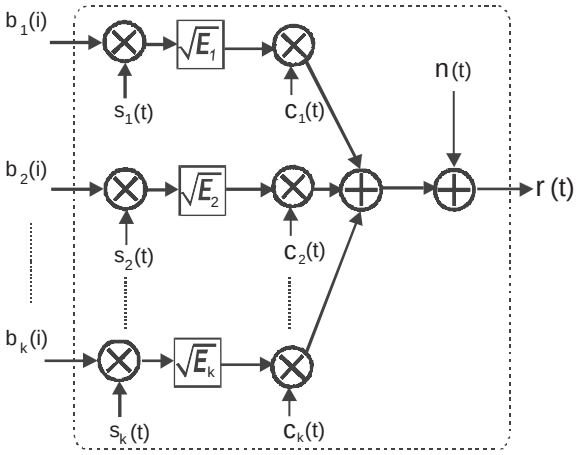
O esquema síncrono de transmissão-recepção adotado neste trabalho é mostrado na Fig. 1. O receptor é constituído de um banco de filtros casados, seguido de um algoritmo heurístico com o intuito de maximizar o desempenho.

Algoritmos Heurísticos

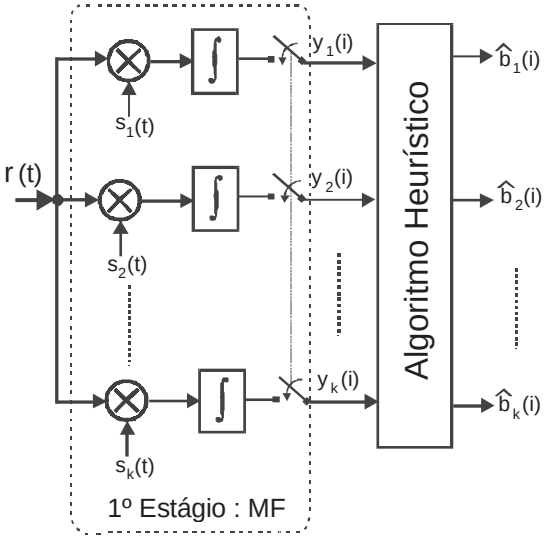
Em comparação com as técnicas exatas, os algoritmos heurísticos não garantem encontrar uma solução ótima após atingirem um critério de parada; mas estes têm demonstrado alta eficiência em problemas de larga combinação para casos práticos, além de poderem ser modificados facilmente, adaptando-se ao problema analisado (GOLDBARG; LUNA, 2000; TAN, 2001).

Algoritmos Evolucionários

Os algoritmos evolucionários constituem métodos de busca não determinística baseados em mecanismos de seleção e evolução natural seguindo a teoria da evolução das espécies de Darwin, sendo a programação evolucionária (EP) e os algoritmos genéticos (GA) os principais. Em Castro e Von Zuben (2002), foi introduzido um outro algoritmo baseado na evolução genética, denominado CLONALG, o qual visa representar a resposta adaptativa da imunização de um organismo na presença de um antígeno.



(a) Transmissor + Canal



(b) Receptor

Figura 1. Modelo em banda base para (a) transmissão + canal e (b) recepção utilizados na modelagem do sistema

As estratégias utilizadas para a diversificação do universo de busca são baseadas nos processos que ocorrem na reprodução das células, sendo chamadas de mutação e *crossover*. O critério de mutação consiste na troca de um gene escolhido aleatoriamente por um outro que possui uma característica diferente; por exemplo:

Pai	Mutação	Filho
-1 1 -1 1 -1 1	⟶	*-1 1 1 1 -1 1*

O critério de *crossover* utiliza dois indivíduos, com o objetivo de formar novos indivíduos a partir da troca de genes entre os indivíduos pais. O ponto de *crossover* também é escolhido aleatoriamente. Exemplo:

Pais	Crossover	Filhos
-1 1 : -1 1 -1 1	⟶	*-1 1 1 1 1 -1*
1 1 : 1 1 1 -1	⟶	*1 1 –1 1 -1 1*

A estratégia para a intensificação proposta neste trabalho consiste em clonar (replicar) os indivíduos que possuem melhores genes, no intuito de produzir uma população altamente evoluída na geração posterior.

Outra estratégia utilizada consiste na tentativa de quantificar o número necessário de mutações que um indivíduo deve sofrer para que este resulte em um melhor conjunto de genes (ABRÃO; CIRIACO; JESZENSKY, 2004). Esta estratégia visa adaptar a matriz de desvio padrão do algoritmo EP através dos parâmetros Eb/No, efeito near-far (*NFR*) e função custo.

Nos pseudocódigos dos algoritmos evolucionários descritos aqui, o número de usuários é denotado por *K*, *g* indica a geração atual, *G* o número de gerações total e *p* o tamanho da população.

Programação Evolucionária (EP)

O algoritmo heurístico de programação evolucionária (EP) é o algoritmo evolucionário mais simples, pois utiliza apenas o critério de mutação, com probabilidade p_m de ocorrência, como estratégia de diversificação e nenhuma estratégia de intensificação (FOGEL, 1994; LIM et al., 2003). Com isso, este algoritmo possui a menor complexidade computacional por geração dentre os evolucionários, mas a convergência é mais lenta, sendo necessárias gerações extras para atingir o desempenho desejado. Descreve-se a seguir o algoritmo EP:

a) Utiliza-se a saída do detector convencional como o vetor de genes do indivíduo inicial.

$$s_1 = \hat{b}_{CD} = sign(y) \tag{7}$$

b) Os outros vetores de genes dos indivíduos que formam a população da 1ª geração são determinados por um processo aleatório.

$$s_i = U\{-1,+1\}^K, i = 2,\ldots, p \tag{8}$$

onde $U\{-1,+1\}^K$ é um vetor binário uniformemente distribuído. A matriz de genes iniciais possui K linhas e p colunas (pais) e pode ser expressa por:

$$S = \left[s_1,\ldots, s_p \right] \tag{9}$$

c) Para cada s_i, i = 1,...,p, calcula-se o valor da função custo dada por (3):

d) Para g = 1, 2, ...,G:

i. Para cada s_i, i = 1,...,p, é introduzida a mutação, no intuito de se criar filhos que possuam melhores genes.

$$S_{i,j+p} = sign\left\{S_{i,j} + N\left(0,\sigma_{i,j}^2\right)\right\}, \quad \begin{matrix} i = 1,\ldots, K \\ j = 1,\ldots, p \end{matrix} \tag{10}$$

onde $S_{i,j}$ representa o i-ésimo gene e j-ésimo indivíduo, $N\left(0,\sigma_{i,j}^2\right)$ representa um processo aleatório Gaussiano com média 0 e variância $\sigma_{i,j}^2$. Esta variância é proporcional à porcentagem de troca dos genes, mutação.

ii. Para cada s_{i+p}, i = 1,...,2p, é calculado o valor da função custo como em (6):

iii. A população é ordenada de forma decrescente em relação ao valor obtido pela função custo. Os melhores p indivíduos, ou seja, que possuem maiores valores para a função custo, são selecionados para compor a base da nova geração.

iv. Retorna-se para a etapa d) até que o número de gerações g atinja um valor G pré-estabelecido.

e) Toma-se o indivíduo S_1 como sendo o vetor de saída do algoritmo.

Programação Evolucionária com Clonagem (EP-C)

Este algoritmo também é baseado na teoria da programação evolucionária, porém aqui são aplicadas as duas estratégias de intensificação descritas anteriormente, a clonagem e a utilização da matriz de desvio padrão adaptativa.

Na obtenção da matriz de desvio padrão otimizada, analisou-se a percentagem de troca dos genes do indivíduo candidato na *g*-ésima geração do algoritmo EP em função do desvio padrão total, σ_{EP}, a qual indicou um comportamento acentuadamente crescente na faixa de $\sigma_{EP} \approx [0,5;2]$, figura 2. Otimizações feitas a seguir levaram em consideração este intervalo. Considerou-se esta faixa para σ_{EP}, uma vez que, valores muito abaixo ou muito acima desta faixa representam ou uma chance remota de troca de genes ou uma chance elevada de troca, respectivamente. No último caso, é computacionalmente menos complexo gerar um novo indivíduo candidato. No primeiro caso, não há evolução (geração sem evolução).

Assim, adotando-se $\sigma_{EP} = [0,5;2]$, verifica-se que a faixa de troca de pico estará confinada ao intervalo $\approx$ 5% a 60%, respectivamente. Efetivamente, está-se restringindo a probabilidade média de troca do gene na *g*-ésima geração à faixa de 2,5% a 30 %, garantindo um compromisso entre variabilidade (entropia) e evolução.

Em função dos parâmetros *SNR*, *NFR* e função custo, obteve-se uma expressão para o desvio padrão do processo EP referente à probabilidade de mudança de genes do indivíduo candidato na *g*-ésima geração (ABRÃO; CIRIACO; JESZENSKY, 2004):

$$\sigma_{EP(i,j)} = \sqrt{\frac{4}{Eb/No}} - \sqrt{\frac{NFR(i)}{32}} + 1 - \frac{\varphi(j)}{\max\left[\varphi(j)\right]} \tag{12}$$

onde os índices *i* e *j* referem-se às linhas e colunas da matriz do desvio padrão, que refletem os genes (*i*-ésima linha) e os indivíduos candidatos (*j*-ésima coluna).

O primeiro termo em (12) é devido ao efeito da relação sinal-ruído, sendo neste caso um escalar, idêntico para todos os elementos da matriz desvio padrão (σ_{EP}). O segundo termo é devido ao efeito Near-Far. Quanto maior o efeito Near-Far, menor será o desvio padrão. O efeito Near-Far é incluído em cada gene da *i*-ésima linha, isto é, resultará em valor idêntico para todas as gerações do EP. O último termo é devido ao efeito da função custo. Este efeito terá um impacto sobre cada indivíduo (*j*-ésima coluna), implicando em um aumento do desvio padrão toda vez que o valor da função custo do *j*-ésimo indivíduo for menor que o maior valor da função custo obtido.

A matriz de desvio padrão é calculada a cada geração (apenas o último termo), pois os valores da função custo são diferentes a cada iteração.

Para que $\sigma_{EP} \in [0,5;2]$, algumas condições foram impostas:

Se $\sigma_{EP(i,j)} < 0,5$, adota-se $\sigma_{EP(i,j)} = 0,5$.

Se $\sigma_{EP(i,j)} > 2$, adota-se $\sigma_{EP(i,j)} = 2$.

com $i = 1,...,K$ e $j = 1,...,2p$.

Figura 2: Porcentagem de troca dos genes em relação ao desvio padrão

A estratégia de clonagem substitui a etapa *d)iii.* do algoritmo EP pela seguinte etapa:

d)iii. A população é ordenada de forma decrescente em relação ao valor obtido pela função custo. Deve-se escolher apenas uma percentagem de indivíduos para que sejam usados na próxima geração.

$$S = \Omega\{S_{i,j}\}, \text{ com } i = 1,\ldots,K \text{ e } j = 1,\ldots,\frac{p}{I_C} \qquad (13)$$

onde I_C é o índice de clonagem e $\Omega\{\cdot\}$ é o operador de clonagem que cria I_C cópias de cada indivíduo. Deve-se usar um índice de clonagem cuja divisão p/ I_C seja um número inteiro. O índice de clonagem é dado por:

$$I_C = p \cdot i_{\%} \qquad (14)$$

onde $i_{\%}$ é o índice de seleção dos indivíduos que possuem maior função custo e dado em porcentagem.

Estas modificações visam melhorar a taxa de convergência, diminuindo o número de gerações necessárias para se alcançar o desempenho ML.

Algoritmo Genético (GA)

O algoritmo GA é similar ao algoritmo EP, pois estes se baseiam na teoria de evolução genética; no entanto, o algoritmo GA utiliza os critérios de mutação e *crossover* para a estratégia de diversificação e nenhum critério para a intensificação (ERGÜN; HACIOGLU, 2000; YEN; HANZO, 2001). O processo de *crossover* visa criar novos indivíduos que contém características mescladas em relação à geração que os precedeu, num processo de alta probabilidade, p_c,. No algoritmo GA, a porcentagem de mutação, p_m, é bem menor que a do EP.

O algoritmo GA é idêntico ao algoritmo EP, porém incluindo o efeito do critério de *crossover*, antes da etapa *d)i*, sendo descrita por:

d) 0. É introduzido o efeito da recombinação genética, o que cria dois filhos combinando-se sub-

partes de dois pais (com probabilidade de crossover = p_c). A matriz de pais é dada por:

$$S = \left[s_1, \ldots, s_p \right] \quad (15)$$

Então a matriz de filhos deve ser criada trocando-se bits dos vetores pais a partir de pontos de crossover. Estes pontos são gerados aleatoriamente.

Algoritmo de Clonagem Seletiva (CLONALG)

Este algoritmo foi proposto por Castro e Von Zuben (2002) e é baseado na resposta imunológica de um sistema na presença de um antígeno. Quando um antígeno é introduzido o organismo produz diversos tipos de anticorpos no intuito de combater o antígeno introduzido. Apenas os anticorpos que tiverem maior afinidade com o antígeno são reproduzidos. Quando este antígeno novamente é introduzido no organismo, a resposta a ele se torna mais rápida, pois este reconhece o antígeno e utiliza os melhores tipos de anticorpos. Quanto mais vezes o organismo for colocado em contato com o antígeno, mais rápido se torna à resposta a ele.

Esta análise é equivalente à estratégia do algoritmo EP, onde apenas os melhores indivíduos, que possuem melhores genes, são utilizados na próxima geração.

Este algoritmo utiliza a estratégia de mutação como critério de diversificação e o critério de clonagem como critério de intensificação:

a) Utiliza-se a saída do detector convencional como o vetor de genes do indivíduo inicial, como em (7).

b) Os outros vetores de genes dos anticorpos que formam a população da 1ª geração são determinados por um processo aleatório como em (8).

c) Para g = 1, 2, ...,G:

i. Para cada s_i, i = 1,...,p, calcula-se o valor da função custo dada por (6):

ii. Apenas os n melhores anticorpos são selecionados.

iii. Utiliza-se o critério de clonagem, criando $N_C(i)$ cópias de cada anticorpo, onde $N_C = \sum_{i=1}^{n} round\left(\frac{\upsilon p}{i}\right)$, sendo υ um fator multiplicativo que estabelece um compromisso entre complexidade e convergência *(CASTRO; VON ZUBEN, 2002).*

Com isso, os anticorpos são clonados com valores diferentes, dependendo da classificação em relação ao valor da função custo.

iv. Para cada s_i, i = 1,...,N_c, é utilizado o critério da mutação, visando diversificar a busca.O critério de mutação é introduzido segundo uma função baseada na afinidade do anticorpo com o antígeno dada por:

$$\zeta = 100 \cdot \exp\left(-\rho f_n\right) \quad (16)$$

onde ζ é a porcentagem de mutação, ρ controla o decréscimo da função e $f_n = \frac{f_{best}}{f_i}$ é a afinidade do anticorpo ao antígeno, sendo f_{best} o valor da função custo para o melhor anticorpo e f_i o valor da função para o anticorpo a ser mutado.

v. Para cada s_i, i = 1,..., N_c, calcula-se o valor da função custo dada por (6).

vi. A população é ordenada de forma decrescente em relação ao valor obtido pela função custo. Os melhores n anticorpos, ou seja, que possuem maiores valores para a função custo, são selecionados para compor a base da nova geração.

vii. São gerados p-n anticorpos determinados por um processo aleatório.

$$s_i = U\left\{-1,+1\right\}^K, i = 1, \ldots, p-n \quad (17)$$

onde $U\{-1,+1\}^K$ é um vetor binário uniformemente distribuído. A população dos anticorpos é, então, formada pelos n anticorpos selecionados na etapa c) vi. e pelos $p-n$ anticorpos gerados em (17). Portanto, a matriz de anticorpos possui K linhas e p colunas (anticorpos) e pode ser expressa como em (9).

viii. Retorna-se para a etapa c) até que o número de gerações g atinja o valor G pré-estabelecido.

d) Toma-se o anticorpo S_1 como sendo o vetor de saída do algoritmo.

Algoritmos de Busca Local

O mesmo problema de detecção multiusuário pode ser atacado utilizando algoritmos heurísticos de busca local (REEVES, 1993). Estes algoritmos de busca determinística procuram por uma solução ao redor de uma vizinhança (REEVES, 1993; TAN, 2001). Esta vizinhança é baseada na distância de Hamming.

Existem duas estratégias de deslocamento no interior do espaço de busca: a estratégia de deslocamento pelo maior ganho e a estratégia de deslocamento pelo primeiro ganho. A primeira utiliza a melhor solução de uma vizinhança como entrada da próxima iteração. A outra estratégia utiliza a primeira solução que apresentar um maior valor que a solução atual. Neste trabalho, utilizou-se a estratégia de deslocamento pelo maior ganho.

Nos pseudocódigos, o número de usuários é denotado por *K*, *m* indica a iteração atual, *Mt* o número de iterações total e o operador $\overline{(\cdot)}$ indica o maior valor ou melhor vetor de todos os $(\cdot)$, no sentido de atender um critério específico.

Busca Local 1-opt (1-opt LS)

O algoritmo 1-opt LS (*1-optimum Local Search*) é um algoritmo de busca local que procura por uma solução ao redor de uma vizinhança composta por todas as possíveis soluções cuja distância de Hamming é igual a 1.

Para que este algoritmo não retorne para um máximo local, propôs aqui um procedimento adicional à estratégia de deslocamento padrão.

No contexto da detecção MuD, o procedimento adicional utilizado junto ao deslocamento consiste em não retornar a última solução visitada e tida como solução corrente, tentando fazer com que o algoritmo não retorne ao último máximo local. Apesar de esta estratégia melhorar a taxa de escape de máximos locais, este procedimento adicional não garante que o algoritmo atinja o máximo global.

a) Utiliza-se a saída do detector convencional como a solução inicial como em (7).

b) Calcula-se a energia inicial do sistema através da função custo (6). Inicializa-se:

$$s_{best} \leftarrow s_1 \qquad (18)$$
$$f_{best} \leftarrow f(s_1)$$

c) Para m = 1, 2, ..., Mt:

i. Encontram-se as K possíveis soluções na vizinhança $N(s_m)$ cuja distância de Hamming é igual a 1.

$$N(s_m) = \left\{ s_i \in \{-1,1\}^K \text{ tal que } \|s_i - s_m\| = 1 \right\} \qquad (19)$$

onde i = 1,…,K.

ii. Calcula-se a energia do sistema para cada uma das soluções possíveis através da função custo (7).

iii. Encontra-se o ganho g_i para cada uma das energias calculadas:

$$g_i = f(s_i) - f(s_{best})$$
$$s_{m-1} \leftarrow s_m \quad , \; m = 2,...,M \qquad (20)$$

iv. Determina-se qual o maior ganho de energia e qual é a solução correspondente.

Se $\overline{g}_i > 0$:

$$s_m \leftarrow \overline{s}_i$$
$$s_{best} \leftarrow \overline{s}_i \qquad (21)$$

Se $\overline{g}_i \le 0$:

$$s_m \leftarrow \overline{s}_i \quad , \text{ onde: } \overline{g}_i \neq 0 \;\; e \;\; \overline{s}_i \neq s_{m-1} \qquad (22)$$

v. Retorna-se para a etapa c) até que o número de iterações m chegue ao valor Mt pré-estabelecido.

d) Toma-se o vetor s_{best} como o vetor de saída do algoritmo.

Busca Local k-opt (k-opt LS)

O algoritmo *k*-opt LS também é um algoritmo de busca local baseado na mesma estratégia que o algoritmo 1-opt. LS. Porém, este algoritmo procura soluções em uma vizinhança maior, cuja distância de Hamming é igual ou menor que *k*. Portanto, para o algoritmo *k*-opt. LS é necessário apenas substituir a etapa *c) i.* do algoritmo 1-opt. LS por:

c) Para m = 1, 2, ..., Mt:

i.Encontra-se todas as $E=\sum_{i=1}^{k}\frac{K!}{(K-i)!i!}$ soluções possíveis na vizinhança $N(s_m)$ cuja distância de Hamming é menor ou igual a k.

$$N(s_m)=\left\{s_i\in\{-1,1\}^K \text{ s.t. } \|s_i-s_m\|\le k\right\} \tag{23}$$

onde: i = 1,…,E; K é o número de usuários e E é o número de soluções candidatas contidas nesta vizinhança.

Recozimento Simulado (SA)

O conceito do algoritmo SA (*Simulated Annealing*) está associado ao princípio da termodinâmica e metalurgia, onde um sólido aquecido a uma temperatura muito alta e então gradualmente resfriado, tenderá a se solidificar de modo a formar uma estrutura de menor energia possível (KIRKPATRICK; VECCHI, 1983; CERNY, 1985; TAN, 2001).

Para escapar de soluções locais, o SA utiliza uma função de probabilidade de aceitação, proporcional à temperatura, podendo então aceitar uma solução particular que possui um menor valor de energia. Isso faz com que o algoritmo saia de uma região de máximo local e procure pelo máximo global em outras regiões. Ao longo das iterações, a temperatura sofre decréscimos baseados na função de distribuição de Boltzman.

O SA utilizado é baseado na busca local do 1-opt LS, procurando uma melhor solução na vizinhança cuja distância de Hamming é igual a 1. Sendo chamado de SA-LS (KATAYAMA; NARIHISA, 2001).

a) Utiliza-se a saída do detector convencional como a solução inicial, como em (7).

b) Calcula-se a energia inicial do sistema através da função custo (6). Inicializa-se:

$$\begin{aligned} s_{best} &\leftarrow s_1 \\ f_{best} &\leftarrow f(s_1) \\ f(s_m) &\leftarrow f(s_1) \end{aligned} \tag{24}$$

c) Para m = 1,2,...Mt:

i. Encontra-se as K possíveis soluções na vizinhança $N(s_m)$ cuja distância de Hamming é igual a 1, como em (7).

ii. Calcula-se a energia do sistema para cada uma das soluções possíveis através da função custo dada em (3).

iii. Se $\overline{f(s_i)} > f_{best}$:

$$\begin{aligned} s_{best} &\leftarrow \overline{s_i} \\ f_{best} &\leftarrow \overline{f(s_i)} \end{aligned} \tag{25}$$

iv. Calcula-se a variação de energia.

$$\Delta E = f(s_m) - \overline{f(s_i)} \tag{26}$$

Se $\Delta E < 0$:

$$\begin{aligned} s_{m+1} &\leftarrow \overline{s_i} \\ f(s_{m+1}) &\leftarrow \overline{f(s_i)} \end{aligned} \tag{27}$$

Se $\Delta E \ge 0$:

Gera-se um número aleatório $p \in [0,1]$.

Se $p < p(m)$:

$$\begin{aligned} s_{m+1} &\leftarrow \overline{s_i} \\ f(s_{m+1}) &\leftarrow \overline{f(s_i)} \end{aligned} \tag{28}$$

onde $p(m)$ é o critério de aceitação

v. Retorna-se à etapa c) até que o número de iterações m chegue ao valor Mt pré-estabelecido.

d) Toma-se o vetor s_{best} como o vetor de saída do algoritmo.

O critério da probabilidade de aceitação, $p(m)$, é inspirado na termodinâmica, onde a distribuição de Boltzman é usualmente utilizada:

$$p(m) = \exp\left[-\frac{\Delta E}{T(m)}\right] \tag{29}$$

onde ΔE é definido como em (26) e $T(m)$ é a temperatura na iteração *m*. Geralmente, a temperatura inicial possui um valor alto, $T(0) > 0$, sendo gradativamente reduzida através do processo de resfriamento. Existem vários métodos para o processo de resfriamento.

Um dos mais eficientes para vários problemas de otimização consiste em:

a) Inicializar a temperatura $T(0)$.

b) Manter a temperatura constante por L_{SA} iterações consecutivas.

c) A cada série de L_{SA} iterações, ocorre um decréscimo da temperatura através de um fator fixo $\gamma \in [0,1]$. A cada kL_{SA} iterações, a temperatura decai:

$$T(kL_{SA}) = \gamma^k T(0) \tag{30}$$

Portanto, o algoritmo SA deve ser iniciado com os três parâmetros: temperatura inicial, $T(0)$, taxa de resfriamento, γ, e tamanho da série (platô), L_{SA}.

Busca Tabu de Termo Curto (STTS)

O algoritmo STTS (*Short Term Tabu Search*) é baseado no modo determinístico de funcionamento de uma memória. A memória é implementada por meio da gravação de características de deslocamento de soluções previamente visitadas (GLOVER, 1977, 1986, GLOVER; LAGUNA, 1997; TAN, 2001). Esta é descrita pela lista Tabu, a qual é formada pelo passado recente de busca, sendo chamada de efeito de memória de termo curto (*Short Term*).

Estas características de deslocamento são proibidas pela lista Tabu por um certo número de iterações. Isso ajuda a evitar retornos para um máximo local, promovendo uma diversificação na busca de soluções.

a) Utiliza-se a saída do detector convencional como a solução inicial, como em (7). A lista TABU, T_{LIST}, está vazia

b) Calcula-se a energia inicial do sistema através da função custo dada em (6). Inicializa-se:

$$\begin{aligned} s_{best} &\leftarrow s_1 \\ f_{best} &\leftarrow f(s_1) \end{aligned} \tag{31}$$

– Inicializa-se:

$$f_m \leftarrow -\infty \tag{32}$$

c) Para m = 1,2,...Mt;

i. Para j =1,...,K:

• Cria-se um vetor característica (C_j), visando à realização da permutação. Este vetor contém K-1 valores 1, sendo o j-ésimo valor igual a -1.

Realiza-se a permutação do vetor s_m, multiplicando cada elemento pelo seu respectivo valor no vetor característica, ou seja:

$$s_i \leftarrow s_m * C_i \quad , \; i = j \tag{33}$$

Retorna-se ao início desta etapa até que j=K .

ii. Calcula-se a energia atual para todos os s_i vetores candidatos através de (6).

iii. Se $\overline{C_i} \not\subset T_{LIST}$ ou $\overline{f(s_i)} > f_{best}$ (critério de aspiração):

$$s_{m+1} \leftarrow \overline{s_i}$$
$$f_{m+1} \leftarrow \overline{f(s_i)} \quad (34)$$

Caso contrário:

$$\overline{f(s_i)} \leftarrow \overline{f(s_i)} \text{ , para } \overline{C_i} \not\subset T_{LIST} \quad (35)$$
$$\text{repetiçao} \leftarrow \text{repetiçao} + 1$$

iv. Tome a melhor solução desta iteração como a solução corrente da próxima iteração, como em (34).

Se $\overline{f(s_i)} > f_{best}$:

$$s_{best} \leftarrow \overline{s_i}$$
$$f_{best} \leftarrow \overline{f(s_i)} \quad (36)$$

Como $\overline{f(s_i)} > f_{best}$, sendo portanto, a melhor solução visitada até o momento, deve-se utilizar o princípio da intensificação. Este princípio visa identificar regiões atrativas, examinando todas os possíveis deslocamentos da solução $\overline{s_i}$; ou seja, na próxima iteração é desconsiderada a etapa c) iv., pulando diretamente de c) iii. para a etapa c) v. Com isso, esta nova solução é aceita independentemente da lista Tabu e do critério de aspiração.

v. Transfira a característica de deslocamento da melhor solução para a lista Tabu, ou seja:

$$T_{LIST}(m) \leftarrow \overline{C_i} \quad (37)$$

Se a lista Tabu estiver completado P períodos, esta é esvaziada, ou seja:

$$T_{LIST}(P) \leftarrow [\ ] \quad (38)$$

vi. Retorna-se para a etapa c) até que o número de iterações m chegue ao valor Mt pré-estabelecido.

d) Toma-se o vetor S_{best} como o vetor de saída do algoritmo.

O critério de aspiração utilizado aceita uma nova solução, independente da lista Tabu, quando esta possuir um valor de energia maior que todas as soluções já visitadas anteriormente.

Busca Tabu Reativa (RTS)

A lista Tabu do algoritmo STTS é implementada utilizando os efeitos de termo curto (*short term)*. Mas isto não garante escapar de retornos aos máximos locais. Adicionalmente, a escolha de um período de proibição (P) fixo, adequado para cada problema, torna-se uma tarefa difícil, pois um período pequeno é insuficiente para evitar retornos a máximos locais e um período grande demais reduz a quantidade de deslocamentos possíveis, acarretando em uma busca ineficiente.

O algoritmo RTS (*Reactive Tabu Search*) combina o efeito do termo curto com outro efeito de memória para evitar retornos aos máximos locais e garantir uma busca eficiente. Este efeito é conhecido como memória de termo longo (*Long Term*), o qual alterna entre as fases de intensificação e diversificação da busca (BATTITI; TECCHIOLLI, 1994; TAN, 2001).

A memória de termo longo do STTS é constituída pelo efeito de memória de termo curto do algoritmo STTS adaptando-se o período de proibição durante a busca, fazendo o valor deste período assumir diferentes valores a cada iteração ($P(m)$).

O período de proibição é inicializado com um pequeno valor, $P(0)$, sendo adaptado na ocorrência de repetições.

Quando uma repetição é encontrada, encoraja-se a diversificação pelo acréscimo do período $P(m)$ em passos de 2. Para que essa diversificação não assuma valores muito altos após algumas iterações, reduz-se o período $P(m)$ em passos de 2 quando:

$$\| f_m - f_{best} \| < \psi f_{best} \text{ , onde } \quad 0 < \psi < 1$$

Em Tan (2001, p.54) foi obtido experimentalmente um valor robusto para a constante que controla a redução do período $P(m)$, sendo $\psi = 0,3$.

O algoritmo RTS é idêntico ao STTS, exceto que a etapa *c) v*, que deve ser substituída por:

c) Transfira a característica da melhor solução para a lista Tabu, ou seja:

$$T_{LIST}(m) \leftarrow \overline{C_i} \tag{39}$$

Se a lista Tabu estiver completado $P(m)$ períodos, esta é esvaziada, ou seja:

$$T_{LIST}(P(m)) \leftarrow [\ ] \tag{40}$$

Onde o valor de $P(m)$ deve ser atualizado a cada iteração.

Resultados Numéricos

Nesta seção são compilados resultados de desempenho para o sistema DS/CDMA, utilizando os algoritmos heurísticos evolucionários e de busca local no processo de detecção MuD.

Os seguintes parâmetros foram considerados: seqüências aleatórias de comprimento $N = 32$, número de usuários K = 12 e 24 usuários, resultando em um carregamento $L = K/N =$ 0,375 e 0,75 respectivamente; região de relação sinal-ruído média ($E_b/N_0 = 15dB$) e cenários com controle perfeito de potência e com disparidades na faixa de NFR = 15 a 30dB.

Para os parâmetros do canal, adotou-se o modelo de Jakes modificado (DENT; BOTTOMLEY; CROFT, 1993), com freqüência da portadora $f_c = 2$ GHz, número de osciladores $N_d = 36$ e velocidades de deslocamento dos móveis uniformemente distribuídas entre 0 e 120 km/h.

Em todas as simulações Monte Carlo adotou-se um número mínimo de erros/ponto = 100. Para efeito de comparação foram incluídos os desempenhos dos detectores convencional (CD), de máxima verossimilhança (ML) e o limite quando há apenas um único usuário ativo no sistema (*SuB = Single user Bound*) com modulação BPSK e canal Rayleigh Plano(PROAKIS, 1989).

Os valores adotados para os parâmetros dos algoritmos heurísticos foram obtidos em duas etapas: a) simulações preliminares foram conduzidas adotando-se valores típicos encontrados na literatura; b) simulações adicionais foram feitas visando otimização dos parâmetros, de modo não exaustivo, porém com resultados de desempenho superiores aos obtidos na etapa a).

Para os algoritmos evolucionários na condição de baixo carregamento do sistema, $L = 0,375$, adotou-se p = 50; p_m = 15% para o algoritmo EP; $i_{\%}$ = 10% para o EP-C; p_c = 85% e p_m = 5% para o GA; n = 10, $\upsilon = 0,25$ e ρ = 5 para o algoritmo CLONALG. Em alto carregamento, $L = 0,75$, apenas os valores de p, n e υ foram modificados, adotando-se p = 160, n = 20 e $\upsilon = 0,2$.

Para os algoritmos de busca local adotou-se k = 3 para o algoritmo k-opt LS; $T(0) = 0,3K$, $\gamma = 0,9$ e $L_{SA} = 2$ para o algoritmo SA-LS; para o algoritmo STTS adotou-se $P = 3$ e para o algoritmo RTS adotou-se $P(0) = 1$.

Desempenho dos Algoritmos Evolucionários

A figura 3 mostra que os algoritmos evolucionários convergem para o desempenho ML após um certo número de gerações. Pode-se perceber o efeito dos critérios de intensificação e de diversificação para cada algoritmo, sendo que o algoritmo EP-C alcançou o desempenho ML mais rapidamente.

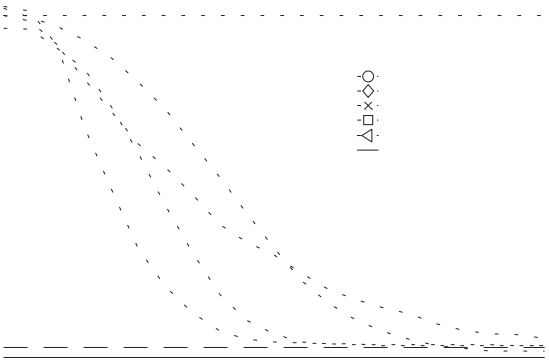


Figura 3. Desempenho dos algoritmos heurísticos evolucionários para baixo carregamento e com controle perfeito de potência

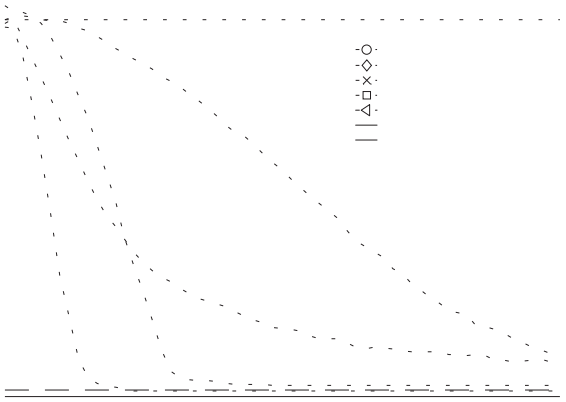


Figura 4. Desempenho dos algoritmos heurísticos evolucionários para alto carregamento e com controle perfeito de potência

Como melhoria adicional, há um ganho de convergência com a utilização da clonagem e da matriz adaptativa de desvio padrão para o algoritmo EP-C em relação ao algoritmo tradicional EP. Embora a estratégia de clonagem do algoritmo CLONALG apresente boas características, a taxa de convergência mostrou-se ser lenta, causada pela ineficiente estratégia de mutação baseada na equação (16). O algoritmo GA mostrou ter uma boa convergência, pois utiliza a estratégia de *crossover* e a estratégia de mutação como princípios de diversificação.

A figura 4 mostra o desempenho alcançado pelos algoritmos em um cenário com controle perfeito de potência e com alto carregamento, $L = 0,75$.

Nota-se que os algoritmos EP e CLONALG não atingiram o desempenho ML para G = 60 gerações, mostrando que há perda de convergência à medida que o carregamento cresce. Esta perda de convergência pode ser explicada pela ineficiente estratégia de mutação do algoritmo CLONALG e pela falta de uma estratégia de intensificação do algoritmo EP. Novamente o algoritmo EP-C alcançou o desempenho ML com o menor número de gerações, mostrando que suas estratégias de intensificação, clonagem e matriz de desvio padrão adaptativa, são eficientes para a detecção MuD síncrona. Embora o algoritmo GA tenha boas estratégias de diversificação, este não possui estratégia de intensificação, acarretando perda de convergência com o aumento do carregamento.

Finalmente, a figura 5 mostra o desempenho alcançado pelos algoritmos em um cenário com usuários divididos em níveis distintos de potência e alto carregamento: $L = 0,75$, 8 usuários com *NFR* = 0dB, 8 usuários com *NFR* = 15dB e 8 usuários com *NFR* = 30dB, sendo o desempenho medido em relação aos usuários com menor nível de potência. Percebe-se que o algoritmo EP não converge para G=60, evidenciando sua ineficiente estratégia de diversificação e ausência de uma estratégia de intensificação.

O algoritmo CLONALG convergiu com g≈52, o que mostra que apesar de sua ineficiente estratégia de mutação, esta conseguiu identificar os anticorpos por afinidade devido a existência do efeito *NFR*, fazendo com que a porcentagem de mutação dos anticorpos com menor afinidade fosse maior.

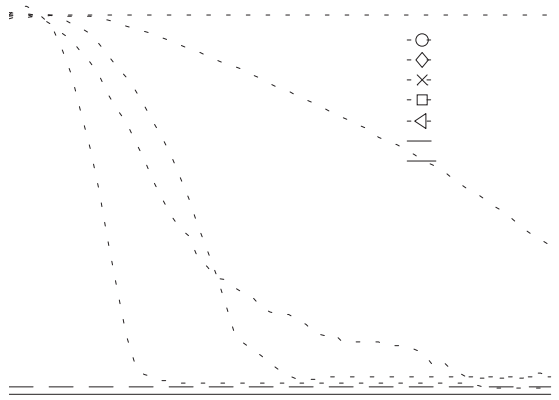


Figura 5. Desempenho dos algoritmos heurísticos evolucionários para alto carregamento e fortes disparidades de potência

Novamente o algoritmo EP-C mostrou possuir a melhor convergência, pois atingiu o desempenho ML com o menor número de gerações. Estes resultados mostram a robustez do algoritmo EP-C ao aumento do carregamento e à presença do efeito *NFR*, indicando que as estratégias conjuntas de intensificação, clonagem e matriz de desvio padrão adaptativa, são altamente eficientes.

Percebe-se que a convergência do algoritmo GA é afetada tanto pelo aumento do carregamento quanto pela disparidade de potência entre os usuários (*NFR*).

Desempenho dos Algoritmos de Busca Local

Pela figura 6, percebe-se que os algoritmos 1-opt LS, SA-LS, STTS e RTS convergem praticamente para o mesmo desempenho utilizando o mesmo número de iterações, mostrando que o universo de busca baseada na equação (6) não apresenta regiões de máximo local críticas, pois até as estratégias de escape menos elaboradas conseguem atingir o mesmo desempenho que outras estratégias mais sofisticadas na mesma iteração.

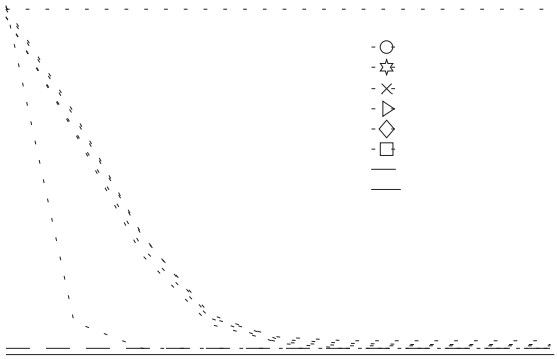


Figura 6. Desempenho dos algoritmos heurísticos de busca local para baixo carregamento e com controle perfeito de potência

O algoritmo 3-opt LS convergiu com um menor número de iterações, pois este procura uma solução em um espaço de busca maior por iteração.

A figura 7 mostra o desempenho alcançado pelos algoritmos em um cenário com controle perfeito de potência e com alto carregamento, $L = 0,75$.

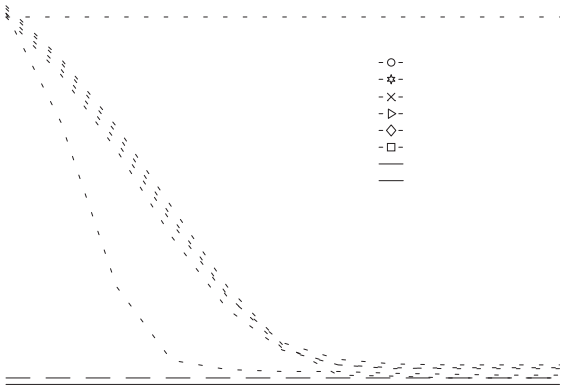


Figura7. Desempenho dos algoritmos heurísticos de busca local para alto carregamento e com controle perfeito de potência

Nota-se que os algoritmos continuaram convergindo com o mesmo número de iterações, o que corrobora a idéia da utilização de algoritmos com estratégias mais simples para escapar de soluções locais. Os algoritmos de busca local apresentaram robustez ao aumento do carregamento quando há controle perfeito de potência, pois todos os algoritmos convergiram, mostrando que não houve perda de convergência.

Finalmente, a figura 8 mostra o desempenho alcançado pelos algoritmos em um cenário com usuários divididos em níveis distintos de potência e alto carregamento: $L = 0,75$, 8 usuários com *NFR* = 0dB, 8 usuários com *NFR* = 15dB e 8 usuários com *NFR* = 30dB, sendo o desempenho medido em relação aos usuários com menor nível de potência. Note-se que os algoritmos necessitaram de mais iterações para convergirem nesta condição do que na condição da figura 7, evidenciando uma grande perda de convergência quando há disparidade de potência entre os usuários ($NFR \neq 0$).

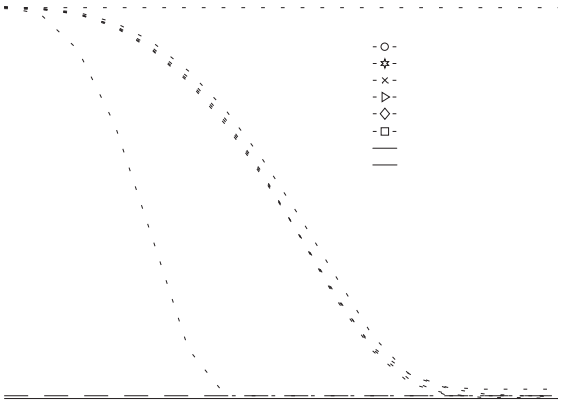


Figura 8. Desempenho dos algoritmos heurísticos de busca local

Isso indica que a convergência dos algoritmos heurísticos de busca local, aplicados ao problema da detecção MuD para sistemas DS/CDMA em canais com desvanecimento Rayleigh Plano, não é robusta à presença do efeito Near-Far, trazendo uma dificuldade adicional na determinação do número necessário de iterações para se alcançar o desempenho ML nas diferentes condições de *NFR*.

Desempenho com Erros nas Estimativas de Canal

Utilizou-se apenas o algoritmo 1-opt LS para a obtenção da degradação de desempenho considerando erros nas estimativas de canal, pois todos os algoritmos heurísticos analisados procuram maximizar a mesma função custo, equação (6) e um erro na estimativa de um parâmetro desta função acarreta em perda na quantificação da melhoria em relação à solução ótima para todos os algoritmos.

Esta degradação está sintetizada na figura 9. O erro foi introduzido na estimativa do módulo (M) e/ou da fase (F) dos coeficientes de canal, para cada usuário e atualizados a cada símbolo transmitido. Este erro foi modelado através de uma distribuição Gaussiana com média igual aos valores verdadeiros dos coeficientes de canal (módulo e fase) e desvio padrão dado por:

$$\sigma_M \in [0,10 \quad 0,20 \quad 0,25 \quad 0,50] \times \beta$$

$$\sigma_F \in [0,10 \quad 0,20 \quad 0,25 \quad 0,50] \times \phi$$

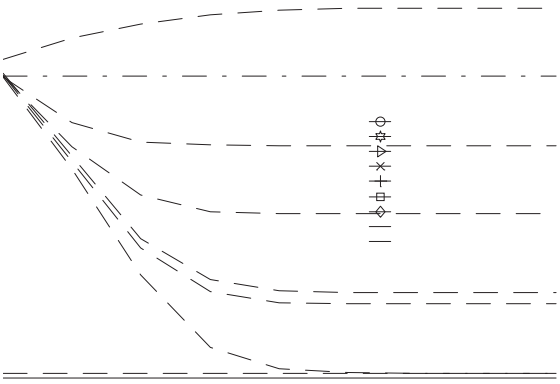


Figura 9. Degradação do desempenho quando há erro na estimativa do canal

Erros simultâneos da ordem de 50% no módulo e fase do canal acarretam em uma grande degradação

do desempenho, tornado o algoritmo 1-opt LS ineficiente. Observa-se que a tolerância a erros na estimativa da fase do canal é menor que a tolerância a erros na estimativa do módulo.

Valores aceitáveis para erros na estimativa do canal para o caso de canal Rayleigh Plano são da ordem de até 10% para o módulo e fase dos coeficientes de canal.

Complexidade Computacional

No intuito de expressar a complexidade dos algoritmos analisados, deve-se encontrar o número de operações envolvidas em cada cálculo da função custo.

Cada cálculo da função custo é obtida como em (6), onde as operações $f_1 = 2y^T c^H A$ e $f_2 = cARAc^H$ podem ser obtidas antes do laço de cálculo da função custo. Para cada cálculo computa-se as operações $f_1.b$ e $b^T.f_2.b$, que em termos de operações é equivalente à $K^2 + 2K$ multiplicações e 1 transposição de ordem *K*.

Para o detector ML, o número de operações cresce exponencialmente com o número de usuários, i.e., $O(2^K)$, onde o operador $O(\cdot)$ indica proporcionalidade ao argumento. São necessárias 2^K gerações de bits de ordem K e 2^K cálculos da função custo para a detecção simultânea de 1 bit dos *K* usuários.

Complexidade para os Algoritmos Evolucionários

Para o algoritmo EP o número de operações cresce dependendo da relação $O(pg)$, sendo necessárias $pg + p - 1$ gerações de bits de ordem K, pg seleções de ordem K, $pg + p$ cálculos da função custo e $2pg$ ordenações de ordem 1. Da mesma forma, o algoritmo EP-C possui uma complexidade que também cresce dependendo da relação $O(pg)$, sendo necessárias $pg + p - 1$ gerações de bits de ordem K, $pg + p$ cálculos da função custo e $2pg$ ordenações de ordem 1. Em contraste com o EP, o algoritmo EP-C realiza pg/I_C seleções de ordem K e gI_C clonagens de ordem K.

A complexidade computacional para o algoritmo GA também cresce na dependencia da relação $O(pg)$, podendo ser obtida adicionando a complexidade da etapa de recombinação genética, *crossover*, à complexidade do algoritmo EP. Essa etapa realiza $pg.p_c$ operações de ordem K. Para o algoritmo CLONALG, o número de operações cresce dependendo da relação $O(g(p + N_C))$, sendo necessárias $(p-n)g + p - 1$ gerações de bits de ordem K, $2ng$ seleções de ordem K, $(p + N_C)g$ cálculos da função custo, $N_C g$ ordenações de ordem 1, $N_C g$ clonagens de ordem K e $N_C g$ cálculos de γ.

Complexidade para os Algoritmos de Busca Local

Para o algoritmo 1-opt LS, o número de operações cresce dependendo da relação $O(mK^3)$, sendo necessárias mK trocas de bits de ordem K, $mK+1$ cálculos da função custo, m seleções de bits de ordem K e m comparações de ordem 1.

O algoritmo k-opt LS possui uma complexidade que cresce dependendo da relação $O(mEK^2)$, sendo necessárias mE trocas de bits de ordem K, $mE+1$ cálculos da função custo, m seleções de bits de ordem K e m comparações de ordem 1.

Para o algoritmo AS, o número de operações também cresce na proporção $O(mK^3)$, onde são realizadas mK trocas de bits de ordem K, $mK+1$ cômputos da função custo; $4m$ comparações de ordem K, $m(K+1)$ gerações de número aleatório, m cálculos de $P(m)$ e m cálculos de $T(m)$.

O algoritmo STTS possui uma complexidade que cresce dependendo da relação $O(mK^3)$, sendo necessárias mK troca de bits de ordem K, $mK+1$ cálculos da função custo, mK gerações de bits de ordem K, $4m$ comparações de ordem 1 e $0{,}5m(P^2+P)$ comparações de ordem K.

Para o algoritmo RTS o número de operações também cresce na proporção $O(mK^3)$, onde são realizadas mK troca de bits de ordem K, $mK+1$ cômputos da função custo; mK gerações de bits de

ordem K, $4m$ comparações de ordem 1 e $0,5m\left(\overline{P}_m^2+\overline{P}_m\right)$ comparações de ordem K, onde $\overline{P}_m$ é a média do período da proibição da lista Tabu.

Número de Operações

Assumindo que os tempos computacionais das operações sejam idênticos, pode-se expressar a complexidade computacional dos receptores em termos de operações substituindo o número de operações de cada *fitness value* e somando todas as outras operações multiplicadas por suas respectivas ordens, como indicado na tabela 1.

Usando os valores numéricos obtidos nas simulações das figuras 3 à 8 para as variáveis g (geração em que houve convergência), K, p, I_C, N_C, n, m (número da iteração referente à convergência), K, E, P e $\overline{P}_m$, é possível expressar a complexidade de cada algoritmo, em termos do número de operações, para se atingir o desempenho ML. A tabela 2 sintetiza estes resultados.

Tabela 1. Complexidade dos detectores em termos de operações

Detector	Número de operações
ML	$2^K K(K+4)$
EP	$pg\left(K^2+7K\right)+K\left(Kp+4p-1\right)$
EP-C	$pg\left(K^2+6K+K/I_C\right)+K\left(Kp+4p-1+gI_C\right)$
GA	$pg\left(K^2+K\left(7+p_C\right)\right)+K\left(Kp+4p-1\right)$
CLONALG	$g\left(p+N_C\right)\left(K^2+5K\right)+K\left(g\left(n-p+N_C/K\right)+p-1\right)$
1-opt LS	$\left(m\left(K^2+1\right)+K\right)\left(K+4\right)-3m-K$
k-opt LS	$\left(m\left(EK+1\right)+K\right)\left(K+4\right)-3m-K$
SA-LS	$\left(m\left(K^2+5\right)+K\right)\left(K+4\right)-17m-K$
STTS	$\left(mK^2+K\right)\left(K+5\right)+0,5K\left(m\left(P^2+P\right)-4\right)$
RTS	$\left(mK^2+K\right)\left(K+5\right)+0,5K\left(m\left(\overline{P}_m^2+\overline{P}_m\right)-4\right)$

Tabela 2. Complexidade (número de operações$\times 10^3$)

Receptor	Fig.3 e 6	Fig. 4 e 7	Fig. 5 e 8
ML	786	11274290	11274290
EP	203	>7250*	>7850**
EP-C	140	1730	2300
GA	151	3040	4020
CLONALG	437	>11330*	9820
1-opt LS	9,45	113,8	210,6
k-opt LS	85,83	6250	9370
SA-LS	9,65	114,5	211,9
STTS	10,12	118,6	219,7
RTS	10,04	118	218,7

* Complexidade subestimada: g = 60; ** idem: g = 65

Conclusões

Uma grande variedade de algoritmos heurísticos analisados aqui resultaram em desempenhos muito próximos ao obtido com o detector ótimo ML. Isso é válido para canais síncronos Rayleigh plano.

Entre os algoritmos heurísticos analisados, a escolha mais apropriada consiste em utilizar o algoritmo sub-ótimo 1-opt LS, pois este resultou em menor complexidade de implementação, atingindo o mesmo desempenho do ML para carregamentos elevados e relação sinal-ruído médias. Deve-se levar em consideração apenas que, com exceção do algoritmo EP-C, todos os algoritmos heurísticos mostraram perda de convergência na presença de efeito *NFR*, trazendo uma maior dificuldade na determinação do número de iterações/gerações necessárias para se alcançar o desempenho ML nesta condição de operação do sistema.

A análise da degradação de desempenho dos detectores, devido à introdução de erros nas estimativas dos coeficientes de canal, indicou ser possível tolerar simultaneamente erros da ordem de 10% no módulo e fase, sem que o desempenho seja degradado consideravelmente.

O sistema DS/CDMA com algoritmos heurísticos analisados mostrou-se mais robusto aos erros nas estimativas do módulo dos coeficientes de canal do que aos erros nas estimativas de fase.

Referências

ABRÃO, T.; CIRIACO, F.; JESZENSKY, P. J. Evolutionary programming with cloning and adaptive cost function applied to multi-user DS-CDMA systems. In: IEEE INTERNATIONAL SYMPOSIUM ON SPREAD SPECTRUM TECHNIQUES AND APPLICATIONS, ISSSTA 4, 2004, Sydney.

BATTITI, R.; TECCHIOLLI, G. The Reactive Tabu Search. *ORSA Journal on Computing*, Baltimore, n.2. p.126-140. 1994.

CASTRO, L. N.; VON ZUBEN, F. J. Learning and Optimization Using the Clonal Selection Principle. In: *IEEE Transactions on Evolutinary Computation*, New York, v.6, n.3, p.239-251, jun. 2002.

CERNY, V. Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. *Journal of Optimization Theory and Applications*, New York, v.45, p.41-51, 1985.

DENT, P.; BOTTOMLEY, G E.; CROFT, T. Jakes fading model revisited. *Electronics Letters*, New York, v.29, n.3, p.1162-1163, jun. 1993.

ERGÜN, C.; HACIOGLU, K. Multiuser Detection Using a Genetic Algorithm in CDMA Communications Systems. *IEEE Transactions on Communications*, New York, v.48, n.8, p.1374-1383, aug. 2000.

FOGEL, D. B. An Introduction to Simulated Evolutionary Optimization. *IEEE Transactions on Neural Networks*, New York, v.5, n.1, p.3-13, 1994.

GLOVER, F. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, Kidlington, n.5, p.533-549, 1986.

GLOVER, F. Heuristic for Integer Programming Using Surrogate Constraints. *Decision Sciences*, Atlanta, v.8, p.156-166, 1977.

GLOVER, F.; LAGUNA, M. *Tabu Search*. Boston: Kluwer Academic Publishers, 1997. 408p.

GOLDBARG, M. C.; LUNA, H. P. L. *Otimização Combinatória e Programação Linear***.** Rio de Janeiro: Campus, 2000. 649p.

KATAYAMA, K.; NARIHISA, H. Performance of simulated annealing-based heuristic for unconstrained binary quadratic programming problem. *European Journal of Operational Research*, Amsterdam, v.134, p.103-119, 2001.

KIRKPATRICK, S.; VECCHI, M. P. Optimization by simulated annealing. *Science*, Washington, v.220, p.671-680, 1983.

LIM, H. S.; RAO, M.V.C., TAN, A. W. C.; CHUAH, H. T. Multiuser Detection for DS-CDMA Systems Using Evolutionary Programming. *IEEE Communications Letters*, New York, v.7, n.3, p.101-103, mar. 2003.

MOSHAVI, S. Multi-User detection for DS-CDMA communications. *IEEE Communications Magazine*, New York, v.34, p.132-136, oct. 1996.

PROAKIS, J. *Digital Communications*. New York: McGraw-Hill, 1989. 928p.

REEVES, C. *Modern Heuristic techniques for combinatorial problems*. Oxford: Blackwell Scientific, 1993. 320p.

TAN, P. H. *Multiuser detection in CDMA-Combinatorial optimization Methods*. 2001. 93f. Thesis (Degree of Licentiate of Engineering) – Department of Computer Engineering, Chalmers University of Technology, Göteborg, 2001.

VERDÚ, Sergio. *Multiuser Detection*. New York: Cambridge University Press, 1998. 451p.

YEN, K.; HANZO, L. Genetic Algorithm Assisted Joint Multiuser Symbol Detection and Fading Channel Estimation for Synchronous CDMA Systems. *IEEE Journal on Selected Areas in Communications*, New York, v.19, p.985-997, 2001.