Recombinação Baseada em Redes Neurais com Função de Base Radial q-Gaussiana

Renato Tinós
Departamento de Computação e Matemática, FFCLRP, Universidade de São Paulo (USP)
Ribeirão Preto, SP, Brasil
rtinos@ffclrp.usp.br

***Resumo*—Recombinação assume um papel muito importante em metaherísticas populacionais. Em geral, recombinação é aplicada sem que maneiras eficientes de combinar informações de diferentes soluções sejam avaliadas. Recentemente, propusemos um método baseado em redes neurais com função de base radial (RBF) Gaussiana para gerar máscaras de recombinação em algoritmos genéticos (AGs) aplicados à otimização pseudo-Booleana. As máscaras são geradas com base nas informações contidas nos pais e em recombinações bem sucedidas passadas. Propomos aqui o uso de redes neurais com função de base radial q-Gaussiana para gerar as máscaras de recombinação em AGs aplicados a problemas de otimização pseudo-Booleana. A RBF q-Gaussiana permite reproduzir diferentes RBFs, como a Gaussiana, a multi-quadrática inversa, e a função de Cauchy, mudando apenas um parâmetro real (q). Resultados de experimentos com o problema NK landscapes indicaram que o crossover com rede RBF q-Gaussiana gerou mais recombinações com sucesso do que crossover com rede RBF Gaussiana, de dois pontos ou uniforme. Entretanto, tal fato não implicou em um melhor desempenho para o AG.**

***Keywords*—I.2.m.c Evolutionary computing and genetic algorithms; I.2.6.c Connectionism and neural nets; G.1.6 Optimization**

I. Introdução

A recombinação de soluções é muito importante em metaheurísticas populacionais [1]. O uso de uma população de soluções permite explorar informações contidas em várias soluções para criar novas soluções. Recombinação também é importante em outras estratégias de otimização, por exemplo, para combinar soluções geradas em diferentes execuções de um algoritmo ou produzidas por diferentes algoritmos [2].

Portanto, investigar operadores eficientes de recombinação é muito relevante. Geralmente, recombinação "cega" é usada, ou seja, recombinação é aplicada sem que a maneiras eficientes de combinar informações de diferentes pais sejam avaliadas. Exemplos típicos são crossover uniforme e de k-pontos em algoritmos genéticos. Além disso, os operadores de recombinação são em geral aleatórios. Recentemente, tem havido interesse em desenvolver operadores de recombinação eficientes ou até mesmo ótimos [3]. A recombinação ótima gera a melhor solução possível resultante da recombinação de dois ou mais pais. No entanto, o problema de achar a recombinação ótima é, em geral, NP-hard [4].

Já a recombinação eficiente explora informações diversas a fim de combinar eficientemente duas ou mais soluções [5]. Essas infomações podem ser: i) características da instância do problema de otimização; ii) soluções passadas; iii) informações sobre a dinâmica da otimização; iv) população atual. Alguns operadores de recombinação eficiente exploram a interação entre as variáveis de decisão a fim de encontrar a melhor maneira de realizar a recombinação. Por exemplo, o network crossover [6] gera máscaras de recombinação baseadas no grafo de interações das variáveis (*variable interaction graph* - VIG). Dada uma instância de um problema de otimização[1] com função de fitness $f(\mathbf{x})$, sendo $\mathbf{x}$ o vetor de variáveis de decisão, o VIG é um grafo $G = (V, E)$ no qual V é o conjunto de variáveis de decisão e o conjunto de arestas E contém todos os pares de variáveis que interagem em $f(\mathbf{x})$. Como em operadores de recombinação tradicionais, as máscaras de recombinação geradas por este operador não dependem das soluções pai.

Por outro lado, no *partition crossover* (PX), as soluções pais são também utilizadas para gerar as máscaras de recombinação [7]. Operadores do tipo PX são operadores de recombinação determinísticos que usam as variáveis de decisão comuns aos pais para decompor o VIG e, como conseqüência, a função de fitness $f(\mathbf{x})$. O primeiro passo do PX é remover do VIG os vértices (variáveis de decisão) comuns aos dois pais. Os componentes conectados do gráfo resultante são componentes recombinantes, i.e., componentes que podem ser eficientemente recombinados a fim de gerar descendentes. As variáveis de decisão em um mesmo componente recombinante devem ser herdades de apenas um dos pais. PX seleciona a melhor solução parcial de um ou outro pai para cada componente recombinante. Fazendo assim, o melhor de 2^k descendentes é encontrado, dado que o número de componentes recombinantes é k. Por recombinar os componentes conectados compostos por variáveis de decisão que interagem entre si na função de fitness, a avaliação do descendente é mais correlacionada com a avaliação dos pais do que em operadores de crossover tradicional. Como resultado, se os pais são ótimos locais em relação a um dado operador de busca local, então os componentes recombinantes são localmente ótimos em relação a este operador. Foi observado em diversos experimentos que os descendentes também são, com muita frequência, verdadeiros ótimos locais [7], ou seja, ao recombinar dois

[1]Neste artigo, estamos interessados em problemas de otimização pseudo-Booleanos, i.e., nos quais as soluções candidatas são representadas por vetores binários e a função de fitness retorna um número real.

ótimos, PX gera descendentes que são também ótimos com grande frequência.

Para permitir a aplicação eficiente do PX, o VIG deve ser eficientemente determinado. Em diversos problemas de otimização [7], [8], [9], o VIG pode ser calculado analisando-se a função de fitness. No entanto, o VIG pode não ser disponível, ou facilmente computado, para alguns problemas de otimização. Além disso, a função de fitness deve ser linearmente separável de modo a permitir que soluções parciais dentro dos componentes recombinantes sejam avaliadas independentemente. Outro problema é que, eventualmente, a remoção dos vértices comuns aos pais não decompõe o VIG. Este pode ser o caso quando o VIG é muito denso, ou seja, o grau de epistasia é alto, ou quando há poucas variáveis de decisão comuns nos pais.

Recentemente, propusemos um método baseado em *redes neurais artificiais* (RNAs) para gerar máscaras de recombinação para *algoritmos genéticos* (AGs) aplicados à otimização pseudo-Booleana [10]. As máscaras são geradas com base nas informações contidas nos pais e em recombinações bem sucedidas passadas. As entradas da RNA são dadas pelas soluções pai e as saídas determinam a máscara de recombinação. O novo operador de crossover deve ser usado em conjunto com outros operadores de recombinação. Recombinações bem sucedidas geradas por diferentes operadores de recombinação compõem o conjunto de treinamento da RNA. A RNA é treinada on-line, ou seja, durante a otimização realizada pelo AG. Para manter o custo computacional para o treinameto da RNA baixo, uma rede com função de base radial (*radial basis function network* - RBFN) é utilizada. RBFNs podem ser rapidamente treinadas e unidades radiais (neurônios na camada escondida) podem ser adicionadas e removidas cada vez que a RNA precisa ser treinada novamente.

RBFNs utilizam funções de base radial como funções de ativação para os neurônios da camada oculta. Em [10], funções Gaussianas foram utilizadas como funções de ativação radiais. Outras funções podem também ser utilizadas, sendo que a escolha de uma ou outra pode ter implicações para o treinamento e desempenho da RBFN. Propusemos no passado o uso da função q-Gaussiana como função de ativação em RBFNs [11], [12]. Uma propriedade interessante da função q-Gaussiana é que ela permite reproduzir diferentes funções de base radial, como a Gaussiana, a multi-quadrática inversa, e a função de Cauchy, mudando apenas um parâmetro real (q). Aqui, propomos o uso de RBFNs com função de base radial q-Gaussiana para gerar as máscaras de recombinação em AGs aplicados a problemas de otimização pseudo-Booleana. A RBFN utilizada para gerar as máscaras de recombinação emprega unidades radiais com diferentes parâmetros q, ou seja, diferentes formas da RBF são utilizadas em uma mesma rede.

Na seção seguinte, RBFNs são discutidas. Na Seção III, a função de base radial q-Gaussiana é formalmente apresentada. O operador de recombinação proposto aqui é introduzido na Seção III. Experimentos com o problema NK landscapes foram realizados com o objetivo de comparar os operadores de recombinação baseados em RBFNs com funções de base radial Gaussina e q-Gaussiana. Nos experimetos, o crossover proposto é ainda comparado com crossover de dois pontos, uniforme e com o PX. Os resultados dos experimentos são apresentados na Seção V. Finalmente, as conclusões e trabalhos futuros são apresentados na Seção VI.

II. Redes RBF

Uma função de base radial (*radial basis function* - RBF) depende da distância entre o exemplo e o centro $\mathbf{c}$. RBFNs são geralmente compostas de uma camada oculta com ativação do tipo RBF e uma camada de saída com ativação linear [13]. Tal arquitetura, que é empregada aqui, permite treinar a RBFN em duas fases: i) seleção dos parâmetros das RBFs; ii) cálculo dos pesos na camada de saída. Quando um exemplo $\mathbf{u}$ é apresentado na entrada, as saídas da RBFN são dadas por:

$$y_k(\mathbf{u}) = \sum_{j=1}^{N_h} w_{kj} h_j(\mathbf{u}) \tag{1}$$

sendo que $k = 1, \ldots, N_o$ é o índice do neurônio de saída, N_o é o número de neurônios de saída, $h_j(\mathbf{u})$ é a ativação da j-ésima unidade radial (neurônio na camada escondida), N_h é o número de unidades radiais, e w_{kj} é o peso entre a j-ésima unidade radial e o k-ésimo neurônio de saída. A ativação da j-ésima unidade radial é definida por:

$$h_j(\mathbf{u}) = \phi_j\big(d_j(\mathbf{u})\big) = \phi_j\left(\frac{\|\mathbf{u} - \mathbf{c}_j\|^2}{r_j^2}\right) \tag{2}$$

sendo que $\phi_j(.)$, $\mathbf{c}_j$, e r_j são, respectivamente, a RBF, o centro, e a largura da j-ésima unidade radial. Quando a RBF Gaussiana é utilizada, $\phi_j\big(d_j(\mathbf{u})\big) = e^{-d_j(\mathbf{u})}$. Aqui, propomos o uso da RBF q-Gaussiana, que é descrita na Seção III.

O primeiro passo para treinar a RBFN é a seleção do número de unidades radiais e os seus respectivos parâmetros. Aqui, usamos um método iterativo que é descrito na Seção IV. A largura da RBF é mantida fixa durante o treinamento. Quando uma unidade radial é definida, a sua ativação pode ser determinada para cada exemplo do conjunto de treinamento. Então, o vetor de peso para o k-ésimo neurônio de saída [14] pode ser calculado como:

$$\hat{\mathbf{w}}_k = \left(\mathbf{H}^{\mathrm{T}}\mathbf{H}\right)^{-1}\mathbf{H}^{\mathrm{T}}\hat{\mathbf{y}}_k \tag{3}$$

sendo que o vetor $\hat{\mathbf{y}}_k$ contém as saídas desejadas para o k-ésimo neurônio de saída para cada exemplo de treinamento e $\mathbf{H}$ é a matriz de projeto composta pelas ativações das unidades radiais (Eq. 2) para cada exemplo do conjunto de treinamento.

III. Função de Base Radial q-Gaussiana

Em [11], [12], propõe-se utilizar a função q-Gaussiana como função de ativação das unidades radiais em RBFNs. A função q-Gaussiana para a j-ésima unidade radial é definida como:

$$\phi_j(d) = e_{q_j}^{-d} \tag{4}$$

sendo $q_j \in \mathbb{R}$ um parâmetro real associado à unidade radial e e_q^x a função q-exponencial de x [15], que é dada por:

$$e_q^x \equiv \begin{cases} \Big(1+(1-q)x\Big)^{\frac{1}{1-q}} & \text{se } \Big(1+(1-q)x\Big) \geq 0 \\ 0 & \text{caso contrário} \end{cases} \quad (5)$$

que torna-se a função exponencial padrão no limite $q \to 1$ [16]. Para alguns valores de q, a função de base radial q-Gaussiana (eqs. 4 e 5) reproduz funções de base radial conhecidas [12]. Esta propriedade é ilustrada na Figura 1.

O principal benefício do uso da função q-Gaussiana como RBF nas RBFNs é que a forma da função de ativação pode ser modificada continuamente alterando o valor de q. Como q é um parâmetro real, uma pequena alteração em seu valor representa uma pequena modificação na forma da função de base radial. Usando RBFs com valores diferentes de q, diferentes formas de funções de base radial são permitidas em uma RBFN. Neste trabalho, utilizamos uma *RBFN com função de base radial q-Gaussiana* (qRBFN) com diferentes valores de q nas unidade radiais para gerar máscaras de recombinação em AGs.

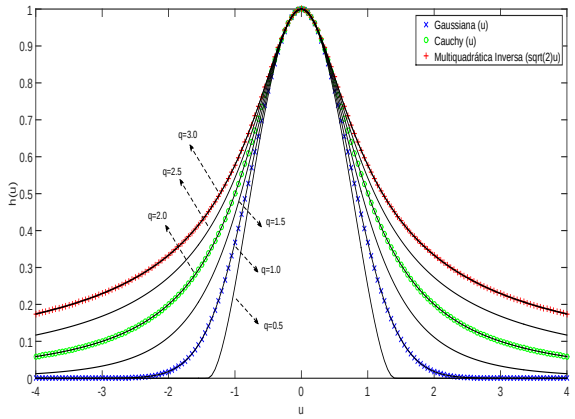

Figura 1. Ativações das unidades radiais para uma única entrada u, $c = 0$ e $r = 1$, para diferentes funções de base radial: Gaussiana, Cauchy, multiquadrática inversa, e q-Gaussiana com diferentes valores de q (linhas sólidas). Repare que a função q-Gaussiana reproduz: i) Gaussiana, quando $q = 1$; ii) Cauchy, quando $q = 2$; iii) multiquadrática inversa, quando $q = 3$.

IV. Crossover qRBFN

Propomos usar a qRBFN para calcular a máscara de recombinação utilizada para gerar um descendente $\mathbf{o} \in \mathbb{B}^N$, dados dois pais, $\mathbf{p^1} \in \mathbb{B}^N$ e $\mathbf{p^2} \in \mathbb{B}^N$, em que N é a dimensão do problema de otimização. Aqui, o parâmetro N_o (Seção II) é igual a N, ou seja, o número de saídas da RBFN é igual ao número de variáveis de decisão do problema de otimização. A máscara de recombinação é um vetor $\mathbf{m} \in \mathbb{B}^N$. Se $m_k = 0$, então $o_k = p_k^1$; caso contrário, $o_k = p_k^2$. O k-ésimo elemento de $\mathbf{m}$ é calculado em função da k-ésma saída da RBFN:

$$m_k = \psi\big(y_k(\mathbf{u})\big) \quad (6)$$

sendo $\mathbf{u} = \big[\mathbf{p^1} - \mathbf{p^2}; \mathbf{p^2}\big]$, i.e., o vetor de entradas com dimensão $2N$ é composto de dois vetores. O primeiro é formado pela diferença entre os dois pais, enquanto que o segundo é formado pelo pai 2. A diferença entre os pais permite detectar se os valores de uma variável de decisão para ambos os pais são iguais ou não. Se os valores são iguais, não importa se o descendente herda o valor de um ou outro pai. No entanto, também é importante verificar o valor da variável de decisão, o que explica o uso do pai 2 nas entradas da qRBFN. Em diversos operadores de crossover eficientes, como o PX, a máscara de crossover é definida explorando as informações contidas nos pais e a interação entre as variáveis de decisão. Aqui, esperamos que a qRBFN seja capaz de estimar a interação entre as variáveis de decisão. O Alg. 1 mostra o pseudo-Código para o *crossover qRBFN* (qRBFNX). A diferença principal em relação ao *crossover RBFN* (RBFNX) proposto em [10] é o uso da qRBFN, ao invés da RBFN com função de ativação Gaussiana, no passo 2 do Alg. 1. Na qRBFN usada aqui, unidades radiais são criadas com valores do parâmetro q da função q-Gaussiana gerados aleatoriamente.

Algorithm 1 $\mathbf{o}$=qRBFNX($\mathbf{p^1}$, $\mathbf{p^2}$)

1: $\mathbf{u} = [\mathbf{p^1} - \mathbf{p^2}; \mathbf{p^2}]$
2: $\mathbf{y}$=qRBFN($\mathbf{u}$)
3: **for** $k = 1$ **to** N **do**
4: $\quad m_k = \psi(y_k)$
5: $\quad$ **if** $(m_k = 0)$ **then**
6: $\quad\quad o_k = p_k^1$
7: $\quad$ **else**
8: $\quad\quad o_k = p_k^2$
9: $\quad$ **end if**
10: **end for**

Dois tipos de qRBFNX são propostos baseados em diferentes formas de computar a função $\psi(.)$ no Alg. 1. No primeiro, chamado de *qRBFNX determinístico* (dqRBFNX), a função $\psi(.)$ é dada por:

$$m_k = \psi\big(y_k(\mathbf{u})\big) = \begin{cases} 0, \text{ se } y_k(\mathbf{u}) < 0.5 \\ 1, \text{ caso contrário} \end{cases} \quad (7)$$

Já no *qRBFNX estocástico* (sqRBFNX), $\psi(.)$ é dada por:

$$m_k = \psi\big(y_k(\mathbf{u})\big) = \begin{cases} 0, \text{ se } y_k(\mathbf{u}) < r \\ 1, \text{ caso contrário} \end{cases} \quad (8)$$

sendo r um número real aleatoriamente gerado no intervalo $[0, 1]$. No sqRBFNX, a saída $y_k(\mathbf{u})$ controla a probabilidade de gerar um valor 0 ou 1 no k-ésimo componente da máscara de crossover $\mathbf{m}$.

Os algoritmos 2 e 3 mostram o pseudo-código do AG com qRBFNX. Cada indivíduo $P(i)$, pertencente à população $\mathbf{P}$, possui um cromossomo $P(i).\mathbf{x}$ e um fitness $P(i).f$. Em cada geração (Alg. 3), mutação ou crossover é aplicado, sendo que a taxa p_c é usada para aleatoriamente selecionar um ou outro operador. Quando crossover é selecionado (passo 5 do Alg. 3), um tipo de cruzamento é selecionado aleatoriamente. Três tipos de crossover são considerados. Quando a qRBFN ainda não foi treinada, *crossover uniforme* (UX) e *crossover*

de 2 pontos (2X) são selecionados aleatoriamente com igual probabilidade. Depois que a qRBFN foi treinada pela primeira vez, qRBFNX é aplicado em 90% das vezes em que crossover é selecionado no Alg. 3. Os outros dois tipos (UX e 2X) são aplicados cada um com 5% de probabilidade de ocorrência.

Algorithm 2 AG com qRBFNX

```
1: U = {}
2: Ŷ = {}
3: initialization(P);
4: while ¬ (stopping criterion) do
5:    Q=generation(P)
6:    if |U| = N_t then
7:       trainqRBFN(U, Ŷ)
8:       U = {}
9:       Ŷ = {}
10:   end if
11:   P = Q
12: end while
```

Algorithm 3 Q=generation(P)

```
1: for i = 1 to |P| do
2:    j_1=selection(P)
3:    if (p_c > r) then
4:       j_2=selection(P)
5:       o=crossover(P(j_1).x , P(j_2).x)
6:       f_o=evaluation(o)
7:       if (|U| < N_t and f_o > P(j_1).f and f_o > P(j_2).f)
         then
8:          for k = 1 to N do
9:             if (P(j_1).x_k = o_k) then
10:               ŷ_k = 0
11:            else
12:               ŷ_k = 1
13:            end if
14:         end for
15:         U=add([P(j_1).x − P(j_2).x; P(j_2).x])
16:         Ŷ=add(ŷ)
17:      end if
18:   else
19:      o=mutation(P(j_1).x)
20:      f_o=evaluation(o)
21:   end if
22:   Q(i).x = o
23:   Q(i).f = f_o
24: end for
```

A qRBFN é treinada toda vez que o conjunto de treinamento está completo, i.e., depois da ocorrência de N_t recombinações com sucesso desde o começo da execução ou do último treinamento. De modo a treinar antes a primeira qRBFN, o tamanho do primeiro conjunto de treinamento é $\frac{N_t}{2}$. Uma recombinação com sucesso ocorre quando o filho tem melhor fitness que os dois pais. O conjunto de treinamento é composto por 2 subconjuntos: i) $\mathbf{U}$, com as entradas; ii) $\hat{\mathbf{Y}}$, com as saídas desejadas, $\hat{\mathbf{y}}$, sendo $\hat{y}_k = 0$ se o k-ésimo elemento do filho $\mathbf{o}$ foi herdado do pai $P(j_1)$, e $\hat{y}_k = 1$ se ele foi herdado do pai $P(j_2)$. Os exemplos de treinamento (recombinações com sucesso) são adicionados ao conjunto de treinamento corrente nos passos 15 e 16 do Alg. 3. No início da otimização e toda vez que a qRBFN é treinada, o conjunto de treinamento é esvaziado (passos 1, 2, 8, e 9 do Alg. 2).

No passo 7 do Alg. 2, a qRBFN é treinada em duas etapas. Primeiro, os parâmetros das unidades radiais (centros e parâmetros q) são ajustados. No primeiro treinamento, $N_h = N_h^i$ unidades radiais são adicionadas. Nos treinamentos seguintes, uma unidade radial é adicionada, até um limite de $N_h = N_h^f$. Aqui, $N_h^i = \frac{N_h^f}{5}$, sendo o número máximo de unidades radiais definido como $N_h^f = 30$.

Cada vez que um neurônio é adicionado, seu vetor de centros é aleatoriamente definido dentro do espaço de entradas. Em [10], os centros eram definidos como exemplos do conjunto de treinamento $\mathbf{U}$, escolhidos aleatoriamente. Entretanto, tal procedimento pode fazer com que alguns exemplos não sejam representados pelos centros. Os centros das unidades radiais são ajustados de acordo com uma regra auto-organizável [14]. Primeiro, um exemplo do conjunto de treinamento, $\mathbf{x} \in \mathbf{U}$, é aleatoriamente escolhido. Então, o índice i_k da unidade radial com centro mais próximo a $\mathbf{x}$ é definido. O vetor de centros da unidade com índice i_k é então ajustado por:

$$\mathbf{c}_{i_k} = \mathbf{c}_{i_k} + 0.05(\mathbf{x} - \mathbf{c}_{i_k}) \tag{9}$$

sendo que o valor que pondera o segundo termo da equação (0.05) foi obtido em experimentos preliminares.

Cada vez que uma unidade radial é criada com índice j, q_j assume um valor aleatório uniformemente gerado no intervalo entre 0.5 e 3.0. Quando q é alto, exemplos distantes produzem maiores ativações que quando q é baixo (ver Figura 1).

Cada vez que a qRBFN é treinada, a regra de ajuste (Eq. 9) é aplicada $5N_h$ vezes, cada vez com um exemplo $\mathbf{x}$. Depois que as unidades radiais são ajustadas, as ativações para os exemplos de treinamento ($\mathbf{U}$) são calculadas e a Eq. 3 é aplicada para cálculo dos pesos na camada de saída da qRBFN. Em [10], mostra-se que o algoritmo para treinamento da RBFN utilizada no RBFNX tem complexidade de tempo $O(N^2)$, sendo N a dimensão do problema de otimização. Depois que a RBFN é treinada, cada aplicação do RBFNX tem complexidade de tempo $O(N)$, a mesma complexidade de UX, 2X e PX. As alterações apresentadas aqui para o qRBFNX não alteram a complexidade de tempo do operador.

V. Experimentos

Os operadores sqRBFNX e dqRBFNX foram comparados entre si e com outros operadores de recombinação. Para isso, tais operadores foram implementados em AGs aplicados ao problema NK landscapes [17]. A plataforma computacional utilizada foi um servidor com 2 processadores Intel Xeon E5-2620 v2 (15 MB Cache, 2.10 GHz) e 32 GB de RAM.

A. Descrição dos Experimentos

O problema NK landscapes é NP-completo, sendo que características diversas da superfície de fitness, tais como tamanho e número de ótimos locais, são dependentes de dois parâmetros, N e K. A função de fitness é dada por:

$$f(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} f_i(\mathbf{x}, \mathbf{s}_i) \tag{10}$$

sendo que $\mathbf{x} \in \mathbb{B}^N$ é o vetor codificando a solução candidata e $\mathbf{s}_i \in \mathbb{B}^N$ é uma máscara com K uns. Cada elemento 1 na máscara indica os elementos de $\mathbf{x}$ que infuenciam a subfunção f_i. Os valores de f_i para cada combinação das entradas são aleatoriamente gerados entre 0 e 1. Aqui são apresentados resultados de experimentos com os modelos adjacente ($adjacent$) e aletório ($random$) para interação entre as variáveis de decisão, considerando-se diferentes valores para N e K.

AGs com seleção por torneio, elitismo, mutação binária e diferentes tipos de crossover são utilizados. Os seguintes AGs são considerados de acordo com o tipo de crossover utilizado:

- **AG com 2X+UX**, i.e., AG no qual 2X e UX são empregados. Cada vez que crossover é aplicado, UX ou 2X é aleatoriamente selecionado com igual probabilidade de ocorrência;
- **AG com PX**. É importante salientar que PX [7] pode ser aplicado apenas quando o VIG é facilmente obtido por meio da análise da função de fitness. Este é o caso do problema NK landscapes;
- **AG com sRBFNX+2X+UX**, i.e., no qual 2X, UX e RBFNX estocástico são empregados. O crossover é aplicado da mesma maneira descrita na Seção IV, mas utilizando a RBF Gaussiana ao invés da RBF q-Gaussiana com q variável;
- **AG com dqRBFNX+2X+UX**, i.e., no qual 2X, UX e qRBFNX determinístico são empregados, conforme descrito na Seção IV;
- **AG com sqRBFNX+2X+UX**, i.e., no qual 2X, UX e qRBFNX estocástico são empregados, conforme descrito na Seção IV.

Em todos os experimentos, os parâmetros dos AGs foram: $|\mathbf{P}| = 50$, $p_c = 0.6$, taxa de mutação igual a $1/N$ e 3 indíviduos no grupo da seleção por torneio. O número de execuções foi 50, sendo que cada AG foi executado durante $50N$ gerações. A fim de controlar o nível de diversidade da população, todas as soluções da população, exceto a melhor, foram substituídas por soluções aleatórias toda vez que a melhor solução convergiu para um ótimo local. Considera-se que o algoritmo converge para um ótimo local quando o fitness da melhor solução não muda durante $10 + \lceil \frac{N}{|\mathbf{P}|} \rceil$ gerações.

As seguintes medidas foram usadas para testar o desempenho dos operadores de crossover:

- ***successful recombination rate***: dada pelo número de recombinações com sucesso dividido pelo número de recombinações realizadas durante a execução. Um recombinação com sucesso é contada toda vez que o descendente é melhor do que os dois pais, ou seja, o descendente tem melhor fitness que os pais. Usamos essa taxa quando comparamos dqRBFNX, ou sqRBFNX, com UX e 2X nas experiências com um mesmo AG. Também usamos essa taxa quando comparamos AGs com diferentes estratégias de crossover;
- ***improvement rate***: dada pelo número de vezes que recombinação melhorou o melhor fitness (da população anterior) dividido pelo número de recombinações realizadas durante a execução;
- ***best fitness***: dado pelo melhor fitness encontrado durante a execução;
- ***time***: dado pelo tempo da execução do AG em segundos.

O teste de Wilcoxon (pareado), com nível de significância $\alpha = 0.05$, foi usado para comparar estatisticamente os resultados.

B. Resultados Experimentais

A Tabela I mostra a comparação da *successful recombination rate* para os AGs com sRBFXN+2X+UX (proposto em [10]) e com sqRBFNX+2X+UX (proposto aqui). Experimentos (não mostrados aqui) indicam que RBFNX com RBFs Cauchy e multiquadrática inversa também resultam em melhores resultados para *successful recombination rate*, quando comparadas com o RBFNX com RBF Gaussiana. As RBFs Cauchy e multiquadrática inversa tem caudas mais longas que a RBF Gaussiana (Figura 1). Em média, os valores de q empregados no qRBFNX são maiores do que 1; a RBF Gaussiana é reproduzida pela RBF q-Gaussiana quando $q = 1$. Portanto, os resultados indicam que os melhores resultados do sqRBFNX (Tabela I) podem ser explicados pelo utilização, em média, de caudas mais longa para a RBF. Vale dizer que, em geral, a recombinação qRBFNX, aplicada da maneira descrita neste trabalho, não resultou em melhores resultados que a utilização da recombinação RBFNX com RBFs Cauchy e multiquadrática inversa.

Tabela I
MEDIANA PARA *successful recombination rate* NOS EXPERIMENTOS DOS AGS COM SRBFXN+2X+UX [10] E SQRBFXN+2X+UX. NO PRIMEIRO (SRBFXN), A RBFN COM FUNÇÃO GAUSSIANA É EMPREGADA, ENQUANTO QUE A FUNÇÃO Q-GAUSSIANA COM q VARIÁVEL É EMPREGADA NO SEGUNDO (SQRBFNX). OS SÍMBOLOS '$=$', '$-$' E '$+$' RESPECTIVAMENTE INDICAM QUE OS RESULTADOS APRESENTADOS NA ÚLTIMA COLUNA SÃO IGUAIS, PIORES OU MELHORES QUE OS RESULTADOS APRESENTADOS NA RESPECTIVA COLUNA. O SÍMBOLO 's' INDICA QUE OS RESULTADOS SÃO ESTATISTICAMENTE SIGNIFICANTES. OS MELHORES RESULTADOS SÃO DESTACADOS EM NEGRITO.

Modelo	N	K	sRBFNX	sqRBFNX
random	100	2	0.117774(s+)	**0.119378**
		5	0.102970(s+)	**0.103586**
	300	2	0.084560(s+)	**0.085406**
		5	0.077251(+)	**0.077499**
	500	2	0.067483(s+)	**0.067661**
		5	0.061909(+)	**0.061968**
adjacent	100	2	0.115920(s+)	**0.118055**
		5	0.101965(+)	**0.102533**
	300	2	0.082956(s+)	**0.083730**
		5	0.076739(s+)	**0.077095**
	500	2	0.066066(s+)	**0.066409**
		5	0.061576(+)	**0.061742**

A seguir, resultados da comparação do qRBFNX com UX e 2X é apresentado. A Figura 2 mostra a comparação da *successful recombination rate* na primeira execução do AG em experimentos com $N = 100$ e $K = 2$. Nestes experimentos, UX, 2X, e dqRBFNX (ou sqRBFNX) são aplicados com igual probabilidade de ocorrência ($\frac{1}{3}$); já no sqRBFXN+2X+UX (ou dqRBFNX+2X+UX), cujos experimentos são descritos a seguir e anteriormente, sqRBFNX (ou dqRBFNX) é aplicado com probabilidade 0.9, enquanto que 2X e UX são aplicados cada um com probalilidade 0.05 (ver Seção IV). A figura mostra que tanto sqRBFNX como dqRBFNX apresentam melhores resultados médios da taxa de sucesso quando comparados com UX e 2X.

Tais observações são confirmadas nos resultados mostrados nas tabelas II e III para experimentos com todas as execuções e para diferentes valores de N e K. A recombinação dqRBFNX apresenta melhores taxas de recombinação com sucesso que UX e 2X para todos os casos. Já sqRBFNX apresenta melhores resultados, mas apenas para o modelo aleatório. Vale observar que UX apresenta resultados melhores que 2X para os experimentos com o modelo aleatório, enquanto que 2X supera UX nos experimentos com o modelo adjacente. Este é um resultado esperado já que 2X preserva ligações entre variáveis de decisão que estão próximas no vetor solução; no modelo adjacente, a avaliação da subfunção $f_i(.)$ na Eq. 10 depende das variáveis de decisão com índices próximos a i. Os resultados indicam que a qRBFNX explora relações entre variavéis de decisão presentes em casos de recombinação com sucesso obtidas no passado para criar as máscaras de decisão dependentes das informações dos pais.

Tabela II
MEDIANA PARA *successful recombination rate* NOS EXPERIMENTOS EM QUE UX, 2X, E DQRBFNX SÃO APLICADOS COM IGUAL PROBABILIDADE DE OCORRÊNCIA.

Modelo	N	K	2X	UX	dqRBFNX
random	100	2	0.0843(s+)	0.0932(s+)	**0.1328**
		5	0.0766(s+)	0.0835(s+)	**0.1108**
	300	2	0.0674(s+)	0.0716(s+)	**0.0901**
		5	0.0622(s+)	0.0657(s+)	**0.0813**
	500	2	0.0553(s+)	0.0584(s+)	**0.0706**
		5	0.0508(s+)	0.0535(s+)	**0.0653**
adjacent	100	2	0.0915(s+)	0.0862(s+)	**0.1281**
		5	0.0919(s+)	0.0744(s+)	**0.1037**
	300	2	0.0721(s+)	0.0665(s+)	**0.0882**
		5	0.0734(s+)	0.0594(s+)	**0.0783**
	500	2	0.0590(s+)	0.0538(s+)	**0.0692**
		5	0.0600(s+)	0.0488(s+)	**0.0629**

Os resultados das comparações entre os AG com diferentes estratégias de recombinação são apresentados nas tabelas IV-VII. A Tabela IV mostra que os resultados de *successful recombination rate* para o AG com sqRBFNX+2X+UX, ou dqRBFNX+2X+UX, são estatisticamente superiores quando comparados com os resultados do AG com 2X+UX. Na comparação entre sqRBFNX+2X+UX e dqRBFNX+2X+UX, o primeiro apresenta resultados superiores. As mesmas observações valem para os resultados de *improvement rate* apresentados na Tabela V, com a exceção dos resultados no experimento com o modelo adjacente para a comparação

Tabela III
MEDIANA PARA *successful recombination rate* NOS EXPERIMENTOS EM QUE UX, 2X, E SQRBFNX SÃO APLICADOS COM IGUAL PROBABILIDADE DE OCORRÊNCIA.

Modelo	N	K	2X	UX	sqRBFNX
random	100	2	0.0914(s+)	0.1027(s+)	**0.1143**
		5	0.0831(s+)	0.0907(s+)	**0.0975**
	300	2	0.0707(s+)	0.0761(s+)	**0.0780**
		5	0.0661(s+)	0.0697(s+)	**0.0707**
	500	2	0.0576(s+)	0.0608(s+)	**0.0615**
		5	0.0535(s+)	0.0558(s+)	**0.0563**
adjacent	100	2	0.0990(s+)	0.0960(s+)	**0.1085**
		5	**0.0995**(s-)	0.0822(s+)	0.0896
	300	2	**0.0759**(s-)	0.0716(s+)	0.0735
		5	**0.0775**(s-)	0.0633(s+)	0.0648
	500	2	**0.0612**(s-)	0.0572(s+)	0.0580
		5	**0.0625**(s-)	0.0514(s+)	0.0519

entre 2X+UX e os operadores propostos aqui. Os melhores resultados de *improvement rate* são obtidos pelo PX que, além de explorar as interações entre as variáveis de decisão, retorna a solução parcial com melhor avaliação para cada componente recombinante. Entretanto, como já salientado, PX utiliza o VIG, que não é disponível facilmente em muitos problemas.

Apesar de gerar descendentes melhores que os pais (medido pela *successful recombination rate*) e indivíduos melhores do que o melhor da última população (medido pela *improvement rate*), os operadores propostos aqui não resultaram em melhores valores de *best fitness* quando comparados com UX+2X (Tabela VI). Em outras palavras, utilizar sqRBFNX+2X+UX ou dqRBFNX+2X+UX não resultou em melhor desempenho para o AG. Novamente aqui, o PX obteve os melhores resultados; para o modelo adjacente, os resultados do 2X+UX foram estatiscamente superiores aos dos operadores propostos. Os melhores resultados obtidos pelos operadores propostos ocorreram para o modelo aleatório quando N e K são grandes.

A Tabela VII mostra que o AG com 2X+UX apresenta os menores tempos de execução. Apesar de todos os operadores de crossover serem aplicados com a mesma complexidade de tempo, O(N), UX e 2X são mais simples que PX e qRBFNX. PX foi mais rápido que sqRBFNX+2X+UX quando $K = 2$, mas foi mais lento quando $K = 5$. Isso é explicado porque o VIG é mais denso no último caso, o que resulta em mais operações para PX. Um VIG mais denso também resulta em um tempo maior para avaliar a função de fitness. Além disso, resulta em um número médio menor de componentes recombinantes encontradas por PX, o que explica parcialmente os melhores resultados de *best fitness* obtidos por sqRBFNX+2X+UX quando este é comparado com PX no modelo aleatório com $K = 5$ (Tabela VI). É importante observar que o tempo para operação e treinamento da qRBFN foram considerados nos valores mostrados na Tabela VII. Portanto, pode-se afirmar que o impacto do treinamento da qRBFN no tempo de execução não foi significativamente alto.

VI. CONCLUSÕES

Propusemos aqui o uso da função de base radial q-Gaussiana em uma rede RBF utilizada para gerar as máscaras de recombinação utilizadas em AGs. Dada a informação de dois

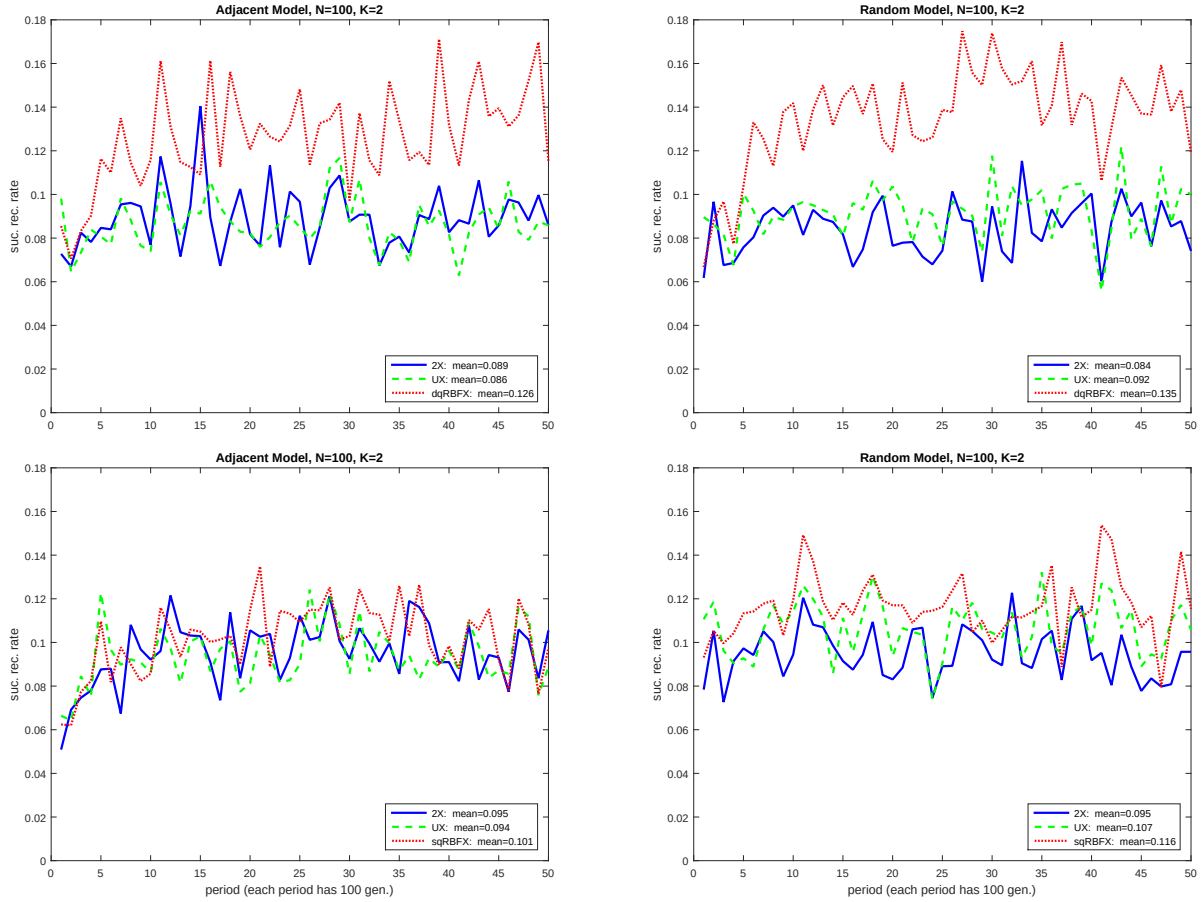

Figura 2. *Successful recombination rate* para diferentes tipos de crossover na primeira execução dos experimentos com $N = 100$ e $K = 2$. Nestes experimentos, UX, 2X, e dqRBFNX (ou sqRBFNX) são aplicados com igual probabilidade de ocorrência.

Tabela IV
MEDIANA PARA *successful recombination rate* PARA OS AGS COM DIFERENTES TIPOS DE CROSSOVER.

Modelo	N	K	2X+UX	PX	dqRBFNX+2X+UX	sqRBFNX+2X+UX
random	100	2	0.093588(s+)	0.096320(s+)	0.114264(s+)	**0.119378**
		5	0.083823(s+)	0.007556(s+)	0.099339(s+)	**0.103586**
	300	2	0.070119(s+)	**0.089586**(s-)	0.076635(s+)	0.085406
		5	0.063938(s+)	0.010588(s+)	0.075813(s+)	**0.077499**
	500	2	0.056428(s+)	**0.074264**(s-)	0.063719(s+)	0.067661
		5	0.052165(s+)	0.010711(s+)	**0.062071**(-)	0.061968
adjacent	100	2	0.092351(s+)	**0.123366**(s-)	0.116809(+)	0.118055
		5	0.084278(s+)	0.044552(s+)	0.101289(s+)	**0.102533**
	300	2	0.069742(s+)	**0.095268**(s-)	0.075412(s+)	0.083730
		5	0.065644(s+)	**0.077289**(-)	0.074064(s+)	0.077095
	500	2	0.056188(s+)	**0.077704**(s-)	0.061471(s+)	0.066409
		5	0.053314(s+)	**0.070528**(s-)	0.062298(-)	0.061742

Tabela V
MEDIANA PARA *improvement rate* PARA OS AGS COM DIFERENTES TIPOS DE CROSSOVER.

Modelo	N	K	2X+UX	PX	dqRBFNX+2X+UX	sqRBFNX+2X+UX
random	100	2	0.000445(s+)	**0.000915**(s-)	0.000429(s+)	0.000459
		5	0.000326(+)	**0.000585**(s-)	0.000320(s+)	0.000340
	300	2	0.000501(+)	**0.000990**(s-)	0.000461(s+)	0.000527
		5	0.000359(s+)	**0.000763**(s-)	0.000346(s+)	0.000371
	500	2	0.000501(s+)	**0.000976**(s-)	0.000470(s+)	0.000523
		5	0.000369(s+)	**0.000851**(s-)	0.000344(s+)	0.000397
adjacent	100	2	0.000442(-)	**0.000837**(s-)	0.000391(s+)	0.000428
		5	0.000351(-)	**0.000657**(s-)	0.000320(s+)	0.000326
	300	2	0.000497(-)	**0.000636**(s-)	0.000446(s+)	0.000484
		5	0.000365(-)	**0.000780**(s-)	0.000315(s+)	0.000359
	500	2	0.000478(s+)	**0.000494**(-)	0.000440(s+)	0.000493
		5	0.000352(+)	**0.000776**(s-)	0.000327(s+)	0.000358

pais, a rede RBF gera uma máscara binária indicando de qual pai a solução descendente deve herdar cada variável de decisão. No método proposto (qRBFNX), as unidades radiais tem funções de ativação com diferentes formas, controladas pelo parâmetro q da função q-Gaussiana. Experimentos com o problema NK landscapes indicaram que a rede RBF com a função q-Gaussiana gerou mais recombinações com sucesso do que a rede RBF com a função Gaussiana [10]. As funções q-Gaussianas empregadas aqui tinham valores médios de q maiores do que 1 (quando $q = 1$, a função Gaussiana é reproduzida), gerando portanto funções de ativação com caudas mais longas quando comparadas com a função de ativação Gaussiana. A utilização de funções de ativação com caudas mais longas mostrou-se benéfica para a o crossover baseado em redes RBF (RBFNX). Resultados de experimentos (não mostrados aqui) com funções de ativação com caudas longas, e.g. Cauchy, confirmaram esta observação. Em geral, o método proposto aqui não resultou em melhores resultados

Tabela VI
MEDIANA PARA *best fitness* PARA OS AGS COM DIFERENTES TIPOS DE CROSSOVER.

Modelo	N	K	2X+UX	PX	dqRBFNX+2X+UX	sqRBFNX+2X+UX
random	100	2	0.7352(+)	**0.7383**(s-)	0.7333(+)	0.7379
		5	0.7553(-)	0.7483(+)	**0.7554**(-)	0.7535
	300	2	0.7343(-)	**0.7411**(s-)	0.7329(+)	0.7333
		5	0.7461(+)	0.7441(+)	0.7451(+)	**0.7474**
	500	2	0.7318(-)	**0.7384**(s-)	0.7312(+)	0.7317
		5	0.7422(s+)	0.7441(+)	0.7409(s+)	**0.7443**
adjacent	100	2	0.7471(s-)	**0.7512**(s-)	0.7436(-)	0.7425
		5	0.7593(s-)	**0.7653**(s-)	0.7512(s+)	0.7553
	300	2	0.7391(s-)	**0.7483**(s-)	0.7367(+)	0.7370
		5	0.7456(s-)	**0.7665**(s-)	0.7384(s+)	0.7410
	500	2	0.7361(s-)	**0.7483**(s-)	0.7328(s+)	0.7344
		5	0.7409(s-)	**0.7676**(s-)	0.7364(+)	0.7372

Tabela VII
MEDIANA PARA TEMPO DE EXECUÇÃO (EM SEGUNDOS) PARA OS AGS COM DIFERENTES TIPOS DE CROSSOVER.

Modelo	N	K	2X+UX	PX	dqRBFNX+2X+UX	sqRBFNX+2X+UX
random	100	2	**6.66**(s-)	14.55(s-)	19.93(s-)	20.45
		5	**14.31**(s-)	29.57(s+)	27.75(-)	27.82
	300	2	**58.91**(s-)	122.81(s-)	159.12(s-)	163.03
		5	**128.35**(s-)	251.38(s+)	231.60(-)	231.95
	500	2	**162.20**(s-)	331.87(s-)	420.54(s-)	430.41
		5	**355.76**(s-)	618.91(s+)	552.51(+)	550.25
adjacent	100	2	**6.67**(s-)	14.86(s-)	20.02(s-)	20.28
		5	**14.46**(s-)	29.17(s+)	27.61(s-)	27.90
	300	2	**59.33**(s-)	122.91(s-)	159.62(s-)	161.82
		5	**129.79**(s-)	248.53(s+)	229.97(-)	230.31
	500	2	**162.72**(s-)	332.84(s-)	413.40(-)	420.56
		5	**358.40**(s-)	464.04(s+)	420.99(+)	417.07

que a utilização da recombinação RBFNX com funções de base radial Cauchy e multiquadrática inversa. Assim, torna-se necessário no futuro investigar métodos para adaptação do parâmetro q durante o processo de otimização realizado pelo AG, a semelhança do que é feito para os centros das unidades radiais.

Os experimentos também indicaram que qRBFNX foi capaz de gerar mais recombinações com sucesso do que UX e 2X. Um número maior de descendentes gerados pela recombinação que melhoraram o fitness da população anterior também foi observado. No entanto, isso não implicou em gerar melhor desempenho (fitness do melhor indivíduo). Este resultado pode ser explicado pelo fato de a qRBFNX ter gerado soluções similares, ao longo das gerações do AG, quando pais similares foram recombinados. Portanto, no futuro, é imperativo pesquisar novos meios de aumentar a diversidade das soluções geradas. Nesta linha, uma maneira de melhorar o desempenho do operador de crossover é utilizando novas formas de construção dos conjuntos de treinamento, de modo a aumentar a diversidade das soluções geradas. A aplicação do qRBFNX em outros problemas de otimização é outra possível linha de pesquisa futura.

AGRADECIMENTOS

Este trabalho foi parcialmente financiado por: i) Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), por meio dos processos 2013/07375-0 e 2015/06462-1; ii) Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), por meio do processo 305755/2018-8.

REFERÊNCIAS

[1] W. M. Spears, *Evolutionary Algorithms: the role of mutation and recombination*. Springer, 2000.

[2] R. Tinós, D. Whitley, and G. Ochoa, "A new generalized partition crossover for the traveling salesman problem: tunneling between local optima," *Evolutionary Computation*, pp. 1–31, 2019.

[3] A. V. Eremeev and Y. V. Kovalenko, "Genetic algorithm with optimal recombination for the asymmetric travelling salesman problem," in *Int. Conf. on Large-Scale Scientific Computing*, 2017, pp. 341–349.

[4] A. V. Eremeev and J. V. Kovalenko, "Optimal recombination in genetic algorithms for combinatorial optimization problems: Part ii," *Yugoslav Journal of Operations Research*, vol. 24, no. 2, pp. 165–186, 2014.

[5] D. Whitley, "Next generation genetic algorithms: A user's guide and tutorial," in *Handbook of Metaheuristics*. Springer, 2019, pp. 245–274.

[6] M. W. Hauschild and M. Pelikan, "Network crossover performance on NK landscapes and deceptive problems," in *Proc. of GECCO'2010*, 2010, pp. 713–720.

[7] R. Tinós, D. Whitley, and F. Chicano, "Partition crossover for pseudo-boolean optimization," in *Proc. of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII*, 2015, pp. 137–149.

[8] R. Tinós, L. Zhao, F. Chicano, and D. Whitley, "NK hybrid genetic algorithm for clustering," *IEEE Trans. on Evol. Comp.*, vol. 22, no. 5, pp. 748–761, 2018.

[9] W. Chen, D. Whitley, R. Tinós, and F. Chicano, "Tunneling between plateaus: improving on a state-of-the-art MAXSAT solver using partition crossover," in *Proc. of GECCO'2018*, 2018, pp. 921–928.

[10] R. Tinós, "An RBF network based crossover for pseudo-boolean optimization," in *To appear in the Proceeding of the 2019 Brazilian Conference on Intelligent Systems (BRACIS)*, 2019.

[11] R. Tinós and L. O. Murta Júnior, "Selection of radial basis functions via genetic algorithms in pattern recognition problems," in *10th Brazilian Symposium on Neural Networks*. IEEE, 2008, pp. 171–176.

[12] ——, "Use of the q-gaussian function in radial basis function networks," in *Foundations of Computational Intelligence Volume 5*. Springer, 2009, pp. 127–145.

[13] J. Moody and C. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computation*, vol. 1, pp. 281 – 294, 1989.

[14] S. Haykin, *Neural networks: a comprehensive foundation*, 2nd ed. Prentice Hall, 1999.

[15] C. Tsallis, "What are the number that experiments provide?" *Química Nova*, vol. 17, p. 468, 1994.

[16] T. Yamano, "Some properties of q-logarithm and q-exponential functions in tsallis statistics," *Physica A*, vol. 305, pp. 486–496, 2002.

[17] S. A. Kauffman, *The origins of order: Self-organization and selection in evolution*. Oxford University Press, 1993.