Universidade do Estado do Rio de Janeiro
Centro de Tecnologia e Ciência
Faculdade de Engenharia

Raphael Luiz Gagliardi

Aplicação de Inteligência Computacional para a Solução de Problemas Inversos de Transferência Radiativa em Meios Participantes Unidimensionais

Rio de Janeiro
2010

Raphael Luiz Gagliardi

Aplicação de Inteligência Computacional para a Solução de Problemas Inversos de Transferência Radiativa em Meios Participantes Unidimensionais

Dissertação apresentada, como requisito parcial para aquisição de título de mestre, ao Programa de Pós Graduação em Engenharia Eletrônica da Universidade do Estado do Rio de Janeiro – UERJ.

Orientadores: Prof. Dr. Luiz Biondi Neto

Prof. Dr. Antônio José da Silva Neto

Rio de Janeiro

2010

Raphael Luiz Gagliardi

Aplicação de Inteligência Computacional para a Solução de Problemas Inversos de Transferência Radiativa em Meios Participantes Unidimensionais

Dissertação apresentada, como requisito parcial para aquisição de título de mestre, ao Programa de Pós Graduação em Engenharia Eletrônica da Universidade do Estado do Rio de Janeiro – UERJ.

Aprovado em: 28 de Março de 2010

Banca Examinadora:

Prof. Dr. Luiz Biondi Neto (Orientador)

Prof. Dr. Antônio José da Silva Neto (Co-orientador)

Prof. Dr. Francisco José da Cunha Pires Soeiro

Prof. Dr. Pedro Henrique Gouvêa Coelho

Prof. Dr. João Carlos Correia Baptista Soares de Mello

Rio de Janeiro

2010

DEDICATÓRIA

Dedico esta dissertação à Flavia, minha esposa que sempre esteve ao meu lado, me apoiando e permitindo que eu realizasse este trabalho. Em especial ao meu filho pela ausência do pai neste período.

A meus pais, Maria e Paulo que sempre me apoiaram ao longo de meus estudos e carreira, a minha mãe que se privou de uma vida para que seus filhos tivessem uma boa educação.

AGRADECIMENTOS

Gostaria de agradecer aos meus professores, Luiz Biondi e Silva Neto pelo apoio e por acreditar que poderia realizar este trabalho.

Ao professor Francisco Soeiro pelo apoio na realização deste trabalho.

Ao meu colega Mauro Gil pela ajuda e apoio na realização desta dissertação.

Aos meus amigos e colegas que me auxiliaram e incentivaram enquanto realizava esta dissertação.

A UERJ onde me formei em engenharia e me proporcionou a realização deste curso.

RESUMO

Esta pesquisa consiste na solução do problema inverso de transferência radiativa para um meio participante (emissor, absorvedor e/ou espalhador) homogêneo unidimensional em uma camada, usando-se a combinação de rede neural artificial (RNA) com técnicas de otimização.

A saída da RNA, devidamente treinada, apresenta os valores das propriedades radiativas [ω, τ_0, $\rho 1$ e $\rho 2$] que são otimizadas através das seguintes técnicas: Particle Collision Algorithm (PCA), Algoritmos Genéticos (AG), Greedy Randomized Adaptive Search Procedure (GRASP) e Busca Tabu (BT).

Os dados usados no treinamento da RNA são sintéticos, gerados através do problema direto sem a introdução de ruído.

Os resultados obtidos unicamente pela RNA, apresentam um erro médio percentual menor que 1,64%, seria satisfatório, todavia para o tratamento usando-se as quatro técnicas de otimização citadas anteriormente, os resultados tornaram-se ainda melhores com erros percentuais menores que 0,04%, especialmente quando a otimização é feita por AG.

Palavras-chave: Problema Inverso, Transferência Radiativa, Rede Neural, PCA, Busca Tabu, GRASP, Algoritmos Genéticos.

ABSTRACT

This research consists in the solution of the inverse problem of radiative transfer for a participating media (emmiting, absorbing and/or scattering) homogeneous one-dimensional in one layer, using the combination of artificial neural network (ANN), with optimization techniques.

The output of the ANN, properly trained presents the values of the radiative properties [w, to, p1 e p2] that are optimized through the following techniques: Particle Collision Algorithm (PCA), Genetic Algorithm (GA), Greedy Randomized Adaptive Search Procedure (GRASP) and Tabu Search (TS).

The data used in the training are synthetics, generated through the direct problem without the introduction of noise.

The results obtained by the (ANN) alone, presents an average percentage error minor than 1,64%, what it would be satisfying, however, for the treatment using the four techniques of optimization aforementioned, the results have become even better with percentage errors minor than 0,03%, especially when the optimization is made by the GA.

Keywords: Inverse Problem, Radiative Transfer, Neural Network, PCA, Tabu Search, GRASP, Genetic Algorithms.

LISTAS DE ILUSTRAÇÕES

LISTAS DE TABELAS

LISTAS DE ABREVIATURAS E SIGLAS

BT	Busca Tabu
FIFO	First In First Out
GA	Genetic Algorithm
GRASP	Greedy Randomized Adaptive Search Procedure
LRC	Lista Restrita de Candidatos
LT	Lista Tabu de Movimentos
MATLAB	Software utilizado como ferramenta
PCA	Particle Collision Algorithm
RN	Rede Neural
RNAs	Redes Neurais Artificiais

LISTAS DE SÍMBOLOS

F1 e F2	Intensidades das fontes externas de radiação
I	Intensidade da radiação eletromagnética
ka_i	Coeficiente de absorção
L	Limite inferior do parâmetro, valor mínimo que pode atingir
$P_{SCATTERING}$	Probabilidade de espalhamento
R	Valor gerado aleatoriamente entre 0 e 1
S*	Melhor solução do algoritmo Busca Tabu
U	Limite superior do parâmetro, valor máximo que pode atingir
Yn	Intensidades de radiação que deixam o meio
βi	Coeficiente de extinção total
M	Co-seno do ângulo polar
ρ_1 e ρ_2	Refletividades nas superfícies internas do meio
σsi	Coeficiente de espalhamento
τ_0	Espessura óptica
ω	Albedo de espalhamento

SUMÁRIO

INTRODUÇÃO

A análise de problemas inversos devido a sua necessidade de possuir conhecimentos de vários campos é classificada como ciência multidisciplinar (CAMPOS VELHO, 2001). Podemos encontrar várias aplicações como na engenharia mecânica com o objetivo de detectar falha em vigas de fundação elástica (ORBANICHA et. al., 2009), na engenharia agrícola com estimativas da difusividade da massa de grãos (BORGES; FENGLER; CERVI, 2009), na engenharia ótica com um método de tomografia para reconstruir as propriedades óticas de um meio, baseado-se na solução de um problema inverso para a equação de difusão (BAL; SCHOTLAND, 2009), na geofísica no estudo de problemas sísmicos (MITROFANOV et. al., 2009), na medicina como ferramenta no diagnostico do câncer de mama (IRISHINA, 2009), dentre outras.

A transferência radiativa em um meio participante (emissor, absorvedor e/ou espalhador) é um fenômeno que está associado a diversos estudos e aplicações: na astrofísica com a simulação tridimensional do efeito do anel em observações da dispersão da luz solar (WAGNER; BEIRLE; DEUTSCHMANN, 2009), na física atmosférica (JERG, 2006), na engenharia ambiental para estimar o teor de umidade do combustível na região mediterrânea da Espanha (YEBRA; CHUVIECO, 2009), sensoriamento remoto na avaliação de índices de vegetação e a produtividade de plantação de soja (GALVÃO; FORMAGGIO; BREUNIG, 2009), na tomografia ótica (TARVAINEN, et. al, 2008), dentre outras.

O interesse na análise e solução de problemas inversos em transferência radiativa vem crescendo nos últimos anos. Neste contexto, algumas técnicas já foram exploradas, tais como: Redes Neurais Artificiais (SOARES; SILVA NETO; SOEIRO, 2004), Recozimento Simulado (SOEIRO; BECCENERI; SILVA NETO, 2009), Comitês de Redes Neuro-Fuzzy combinado com Redes Neurais Artificiais em Cascata (GIL et. al, 2008), algoritmos baseados na distância de Bregman (BERROCAL TITO, 2006), otimização extrema generalizada (SOUZA et. al, 2007), otimização por enxame de partículas (BECCENERI et. al, 2006), dentre outras.

Neste cenário a relevância deste trabalho se encontrará na resolução do problema inverso de transferência radiativa utilizando técnicas combinadas de inteligência computacional e métodos de otimização, sendo eles:

- Rede Neural Artificial combinada com PCA

- Rede Neural Artificial combinada com AG
- Rede Neural Artificial combinada com GRASP
- Rede Neural Artificial combinada com Busca Tabu

Criando-se sistemas híbridos, existe uma expectativa de uma melhor qualidade nos resultados, pois tornam ferramentas eficientes e poderosas para estimativa de propriedades óticas em meios participantes.

Objetivo Específico

O objetivo central desta dissertação é a solução do problema inverso de transferência radiativa utilizando técnicas combinadas de inteligência computacionais e métodos de otimização. Os sistemas híbridos apresentados são: Rede Neural combinado com PCA, Rede Neural combinado com Algoritmos Genéticos, Rede Neural combinado com GRASP e Rede Neural combinado com Busca Tabu. Onde compara-se os resultados dos quatro sistemas híbridos com o resultado de uma única RNA.

Para atingir este objetivo, dividiu-se o trabalho em etapas:

- Pesquisar e aprofundar os conceitos da transferência radiativa e problema inverso;
- Modelar a RNA;
- Modelar os sistemas híbridos;
- Projetar os algoritmos PCA, Algoritmo Genético, GRASP e Busca Tabu;

Organização da Dissertação

Esta dissertação está dividida em cinco capítulos e seis apêndices. Os capítulos são: Introdução, Fundamentação Teórica, Modelagem, Resultados e Conclusão e Trabalhos Futuros. Os apêndices são: Códigos fontes do PCA, Códigos Fontes do AG, Código Fontes do GRASP, Códigos Fontes do Busca Tabu, Códigos Fontes Comuns e Fluxograma do Algoritmo Retropropagação.

Nesta introdução apresenta-se a Justificativa, o Objetivo e a Organização da Dissertação.

No segundo capítulo realiza-se uma revisão teórica dos assuntos: formulação matemática do problema direto e inverso de transferência radiativa, RNA, PCA, AG, GRASP, Busca Tabu e Sistemas Híbridos.

No terceiro capítulo apresentam-se as modelagens da geração dos dados utilizando o problema direto, da RNA e dos sistemas híbridos: RNA combinado com PCA, RNA combinado com AG, RNA combinado com GRASP e RNA combinado com Busca Tabu.

No quarto capítulo, são abordados os resultados dos casos estudados para os quatro métodos híbridos descritos anteriormente e no final é realizada uma comparação e uma breve explicação sobre os resultados.

No quinto capítulo, elabora-se a conclusão do trabalho, baseado no estudo e resultados analisados.

Os apêndices apresentam os códigos Matlab® desenvolvidos para realizar as tarefas descritas nesta dissertação e o fluxograma do algoritmo retropropagação.

1 FUNDAMENTAÇÃO TEÓRICA

1.1 Formulação matemática e solução do problema direto

A modelagem dos fenômenos que ocorrem quando a radiação eletromagnética interage com um meio participante, i.e. meio absorvedor, espalhador e emissor, é feita com a versão linear da equação de Boltzmann.

Considere um meio unidimensional, homogêneo, espalhador isotrópico, de espessura óptica τ_0, com superfícies refletoras difusas (refletividades iguais a ρ_1 e ρ_2), submetido à incidência de radiação externa, conforme a representação esquemática apresentada na Figura 1.

A formulação matemática do problema de interação da radiação com o meio é dada por (ÖZISIK, 1973):

$$\mu\frac{\partial I(\tau,\mu)}{\partial\tau} + I(\tau,\mu) = \frac{\omega}{2}\int_{-1}^{1} p(\mu,\mu')I(\tau,\mu')d\mu' \ em\ 0 < \tau < \tau_{0,} - 1 < \mu < 1 \quad (1)$$

$$I(0,\mu) = F_1 + 2\rho_1\int_0^1 \mu' I(0,-\mu')d\mu', \ \ \mu > 0 \quad (2)$$

$$I(\tau_0,\mu) = F_2 + 2\rho_2\int_0^1 \mu' I(\tau_0,\mu')d\mu', \ \ \mu < 0 \quad (3)$$

onde I é a intensidade da radiação eletromagnética, τ a variável óptica, μ o co-seno do ângulo do feixe de radiação com o eixo τ, ω o albedo de espalhamento simples, F1 e F2 as intensidades das fontes externas de radiação incidentes em τ =0 e τ = τ_0, (SOARES; SILVA NETO; SOEIRO, 2004).

Quando a geometria, as condições de contorno e as propriedades radiativas do meio são conhecidas, o problema de transferência radiativa formulado com a equação (1) pode ser resolvido diretamente, obtendo-se I(τ, μ) para todo τ e todo μ. Este é o problema direto de transferência radiativa em um meio homogêneo unidimensional.

A fim de resolver o problema direto proposto utiliza-se o método de Chandrasekhar, ordenadas discretas, no qual o domínio do ângulo polar é

diferenciado e o termo integral (de dispersão) no lado direito da equação (1) é aplicada uma quadratura de Gauss (CHANDRASEKHAR, 1960).

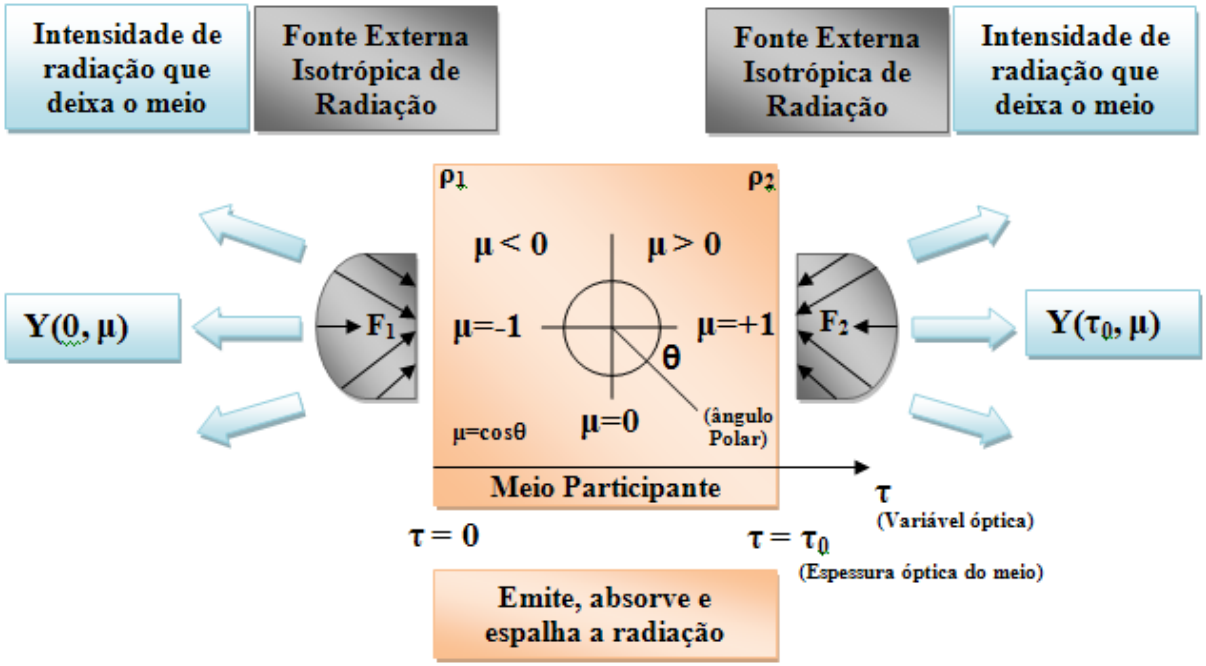

Figura 1: Meio Participante Homogêneo Unidimensional

1.2 Formulação do problema inverso

Conforme a representação feita na Figura 2(a), a partir do conhecimento dos fenômenos envolvidos em um determinado processo, sendo aqui considerado o problema de transferência radiativa, e do modelo matemático usado para descrevê-lo, incluindo a geometria do meio sob análise, e de uma causa conhecida, por exemplo, a intensidade da radiação incidente nas superfícies de contorno no meio participante, pode ser calculado o efeito, ou seja, a intensidade da radiação (radiância) em qualquer ponto do meio e em qualquer direção angular. A incógnita do problema direto de transferência radiativa, i.e. o efeito, é, portanto, a intensidade da radiação (SILVA NETO; CAMPOS VELHO, 2009, p. 26).

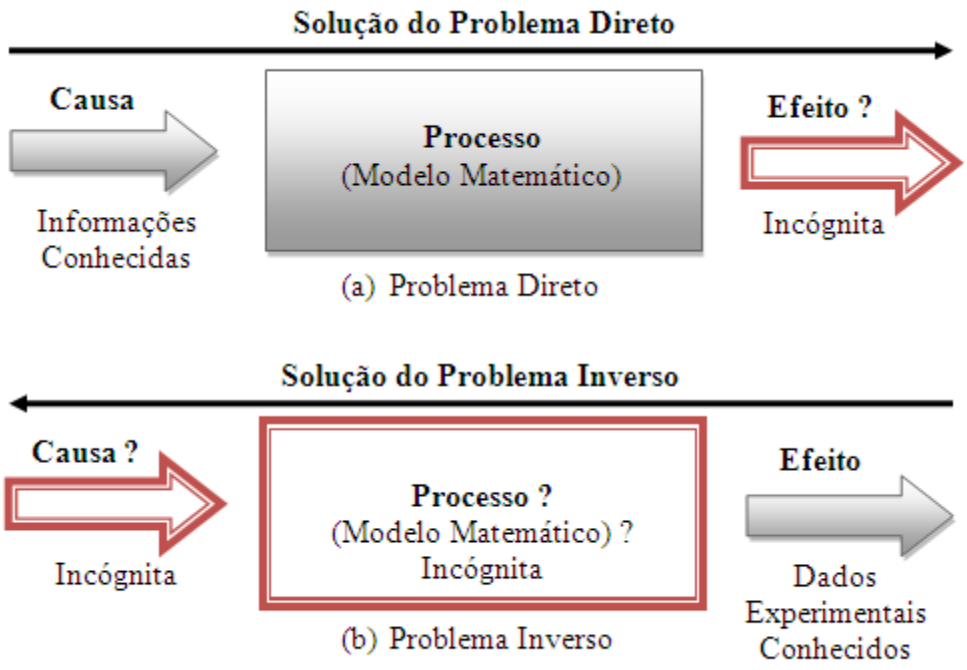

Figura 2: Representação Esquemática do Problema Direto (a) e do Problema Inverso (b)

Considere agora que a condição de contorno, a intensidade de uma ou das duas fontes externas de radiação, e/ou as propriedades radiativas do meio, os coeficientes de absorção e de espalhamento e as refletividades difusas, não são conhecidos. Porém, estão disponíveis valores experimentais da intensidade da radiação, Y, medidos externamente ao meio conforme representado na Figura 1.

Será possível então determinar as incógnitas do problema a partir destes dados experimentais? Reformulando essa pergunta, será possível determinar a causa, as intensidades da radiação externa incidente, e/ou o que se desconhece do processo que está sendo investigado, as propriedades radiativas a partir dos valores medidos experimentalmente da radiação que emerge do meio?

Este é o Problema Inverso (PI), representado esquematicamente na Figura 2(b) e que para o nosso problema podemos observar na figura 3. A matriz Y representa as intensidades que deixam o meio, medidos experimentalmente e a Z representa o valor das incógnitas a serem determinadas, i.e. propriedades radiativas do meio.

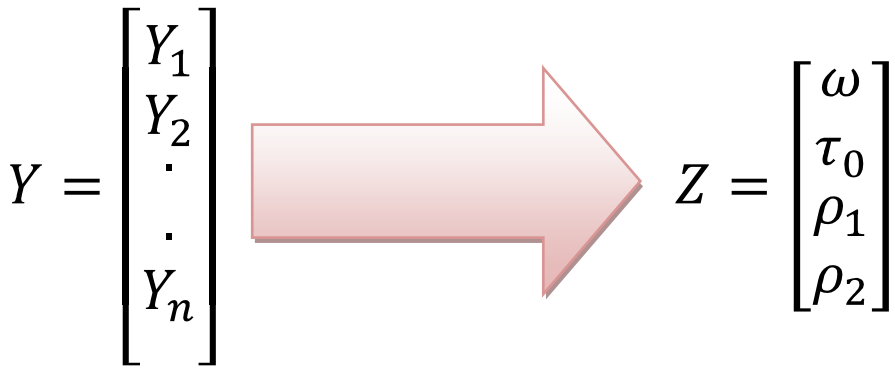

Figura 3: Problema Inverso de Transferência Radiativa

1.3 Rede neural

Uma rede neural é um processador paralelo distribuído constituído de unidades de processamento simples, que têm a propensão natural para armazenar conhecimento experimental e torná-lo disponível para uso (HAYKIN, 2001). Ela se assemelha ao cérebro em dois aspectos:

1) O conhecimento é adquirido pela rede a partir de seu ambiente através de um processo de aprendizagem.
2) Forças de conexão entre neurônios, conhecidas como pesos sinápticos, são utilizados para armazenar o conhecimento adquirido.

O cérebro pode ser modelado como um computador (sistema de processamento de informações) altamente complexo, não linear e paralelo (HAYKIN, 2001). O cérebro humano contém cerca de 100 bilhões de neurônios, cada um destes com até 10.000 conexões sinápticas.

Um neurônio é uma unidade de processamento de informação que é fundamental para a operação de uma rede neural. A figura 4 mostra o modelo de um neurônio, que forma a base para o projeto de redes neurais artificiais. Aqui nós identificamos três elementos básicos do modelo neuronal:

1) Um conjunto de sinapses ou elos de conexão;
2) Um somador para somar os sinais de entrada;

3) Uma função de ativação para restringir a amplitude da saída de um neurônio;

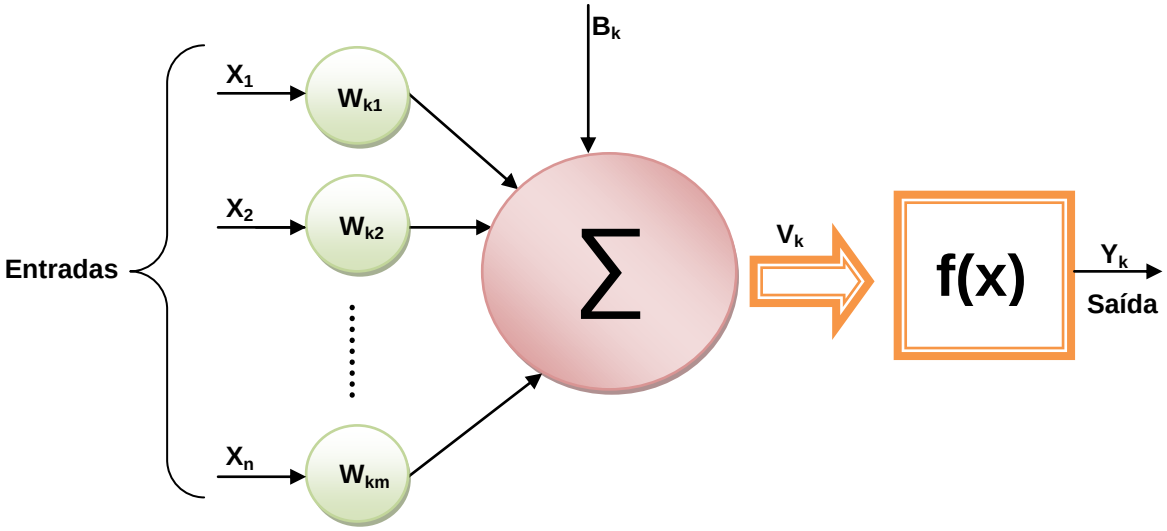

Figura 4: Modelo de um neurônio artificial k

Em termos matemáticos, podemos descrever um neurônio k escrevendo o seguinte par de equações:

$$u_k = \sum_{j=1}^{m} W_{kj}.X_j \tag{4}$$

$$Y_k = f(u_k + B_k) \tag{5}$$

onde X_1, X_2, ..., X_m são os sinais de entrada; W_{k1}, W_{k2}, ..., W_{km} são os pesos sinápticos do neurônio k; u_k é a saída do combinador linear devido aos sinais de entrada; B_k é o bias; f(x) é a função de ativação; e Y_k é o sinal de saída do neurônio.

A maneira pela qual os neurônios de uma rede neural estão estruturados está intimamente ligada com o algoritmo de aprendizagem usado para treinar a rede. Em geral podemos identificar três classes de arquiteturas de rede fundamentalmente diferentes:

1) Redes Alimentadas Adiante com Camada Única

   Em uma rede neural em camadas, os neurônios estão organizados na forma de camadas. Na forma mais simples de uma rede em camadas, temos uma camada de entrada de nós de fonte que se projeta sobre uma

camada de saída e neurônios (nós computacionais), mas não vice-versa. Em outras palavras, esta rede é estritamente do tipo alimentada adiante ou acíclica. Ela é ilustrada na Figura 5 para o caso de quatro nós tanto na camada de entrada como na de saída. Esta rede é chamada de rede de camada única, sendo que a designação “camada única” se refere à camada de saída de nós computacionais (neurônios). Não contamos a camada de entrada de nós de fonte porque lá não é realizada qualquer computação.

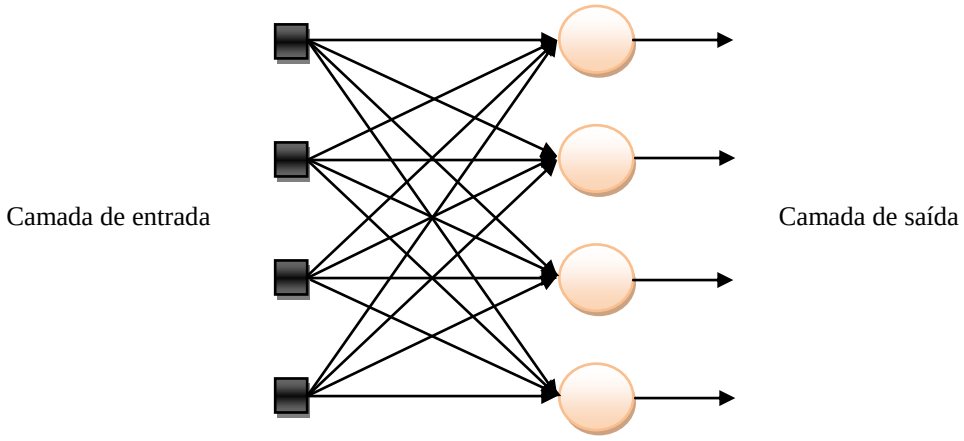

Figura 5: Redes alimentadas adiante com uma única camada de neurônios

2) Redes Alimentadas Diretamente Com Múltiplas Camadas

A segunda classe de uma rede neural alimentada adiante ou *feed-forward* se distingue pela presença de uma ou mais *camadas ocultas*, cujos nós computacionais são chamados correspondentemente de *neurônios ocultos* ou *unidades ocultas*. A função dos neurônios ocultos é intervir entre a camada externa e a saída da rede de uma maneira útil. Adicionando-se uma ou mais camadas ocultas, torna-se a rede capaz de extrair estatísticas de ordem elevada. A habilidade de os neurônios ocultos extraírem estatísticas de ordem elevada é particularmente valiosa quando o tamanho da camada de entrada é grande (HAYKIN, 2001).

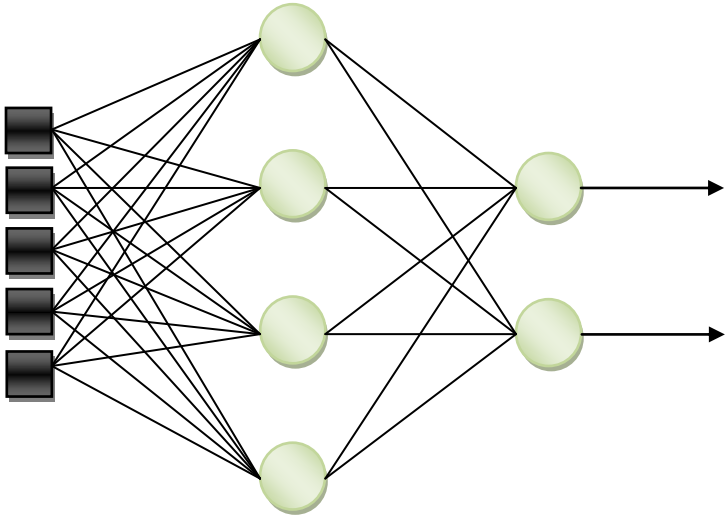

Figura 6: Rede alimentada adiante totalmente conectada com uma camada oculta e uma camada de saída

3) Redes Recorrentes

Uma rede neural recorrente se distingue de uma rede neural alimentada adiante por ter pelo menos um laço de realimentação. Uma rede recorrente pode consistir, por exemplo, de uma única camada de neurônios com cada neurônio alimentando seu sinal de saída de volta para as entradas de todos os outros neurônios, como ilustrado no grafo arquitetural da Figura 7.

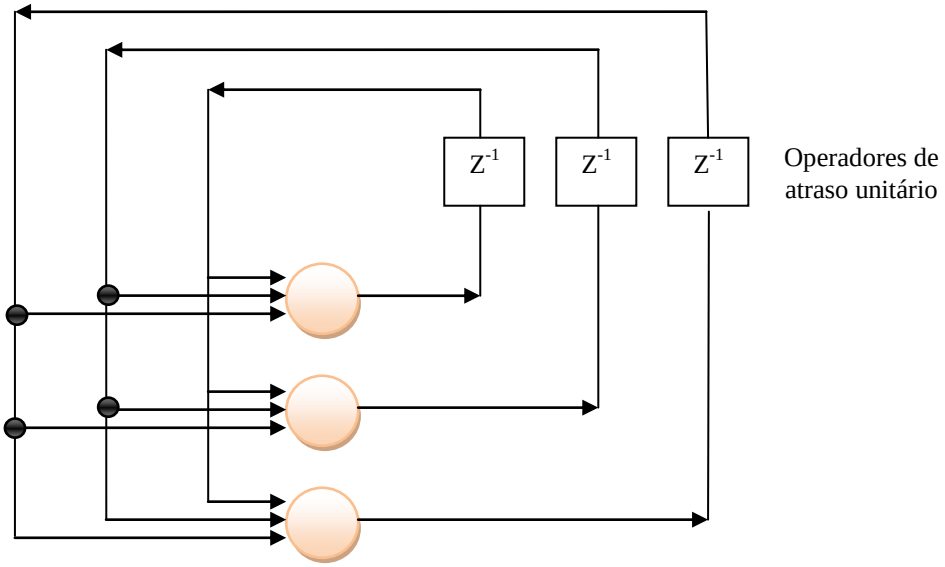

Figura 7: Rede recorrente sem laços de auto-alimentação e sem neurônios ocultos

na estrutura representada nesta figura, não há laços de auto-realimentação na rede; auto-realimentação se refere a uma situação onde a saída de um neurônio é realimentada para a sua própria entrada. A rede recorrente ilustrada na figura 7 também não tem neurônios ocultos (HAYKIN, 2001).

Na Figura 8, ilustra-se outra classe de redes recorrentes com neurônios ocultos. As conexões de realimentação mostradas na Figura 8 se originam dos neurônios ocultos, bem como dos neurônios de saída.

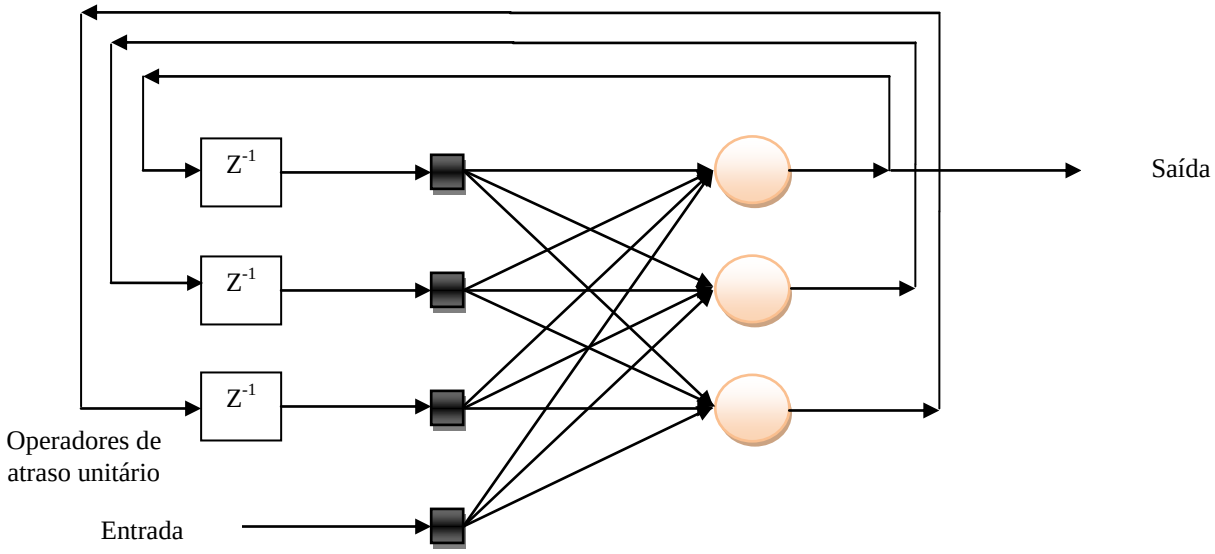

Figura 8: Rede recorrente com neurônios ocultos

O processamento de uma rede neural pode ser dividido em duas fases:

1) Treinamento
2) Processamento

A propriedade que é de importância primordial para uma rede neural é a sua habilidade de aprender a partir de seu ambiente e de melhorar o seu desempenho através da aprendizagem. A melhoria do desempenho ocorre com o tempo de acordo com alguma medida preestabelecida. Uma rede neural aprende acerca do seu ambiente através de um processo interativo de ajustes aplicados a seus pesos sinápticos e níveis de bias. Idealmente a rede se torna mais instruída sobre o seu ambiente após cada iteração do processo de aprendizagem (HAYKIN, 2001).

Existem diversas classificações e técnicas de treinamentos para RNA (HAYKIN, 2001), porém somente detalharemos o aprendizado supervisionado, utilizado nesta dissertação.

Aprendizagem supervisionada ou aprendizagem com um professor em termos conceituais, pode-se considerar o professor como tendo conhecimento sobre o ambiente, com este conhecimento sendo representado por um conjunto de *exemplos de entrada-saída*. A Figura 9 mostra um diagrama em blocos que ilustra esta forma de aprendizagem(HAYKIN, 2001).

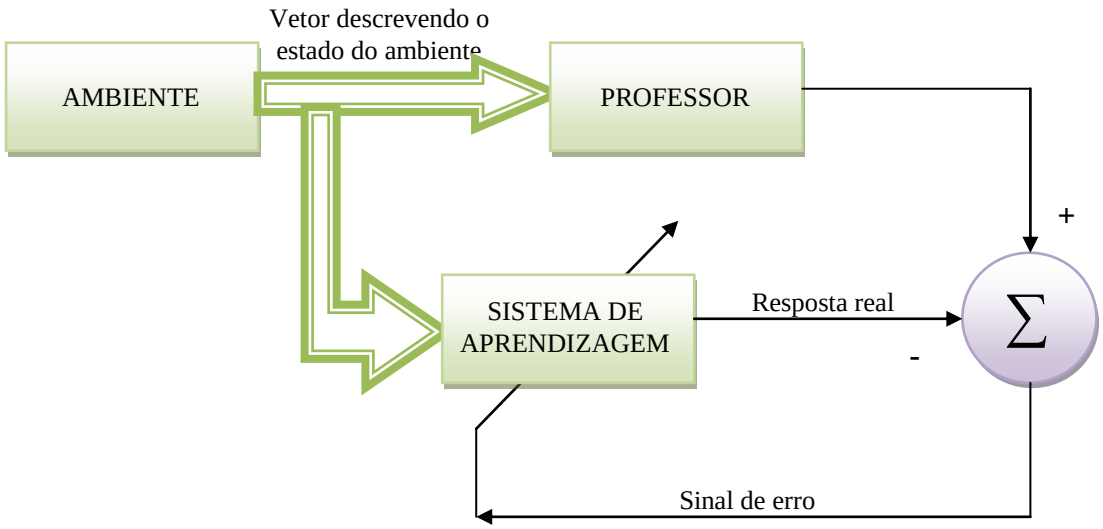

Figura 9: Diagramas em blocos da aprendizagem com um professor

Depois de treinada, a rede passa para a segunda fase, que é a fase de execução. Nesta etapa do processamento, são apresentados vetores de entrada à rede, também sob forma numérica, que normalmente não fizeram parte do treinamento (PASCHOALINO; LOUREIRO; SOARES DE MELLO; BIONDI NETO, 2007).

1.4 PCA

O algoritmo de colisão de partículas foi inspirado no algoritmo de recozimento simulado, seguindo uma analogia com a colisão de partículas nucleares, este foi desenvolvido por Wagner Sacco (SACCO; OLIVEIRA, 2005).

Este algoritmo, denominado algoritmo de colisão de partículas, tem como vantagem o fato de não necessitar da especificação de parâmetros além daqueles que definem o número de iterações.

O PCA é inspirado no espalhamento de uma partícula nuclear incidente (onde ela é espalhada pelo núcleo-alvo) e pela absorção (onde ela é absorvida pelo núcleo-alvo). Nesta modelagem, a partícula que atinge um núcleo com baixo valor da função objetivo é absorvida. Em contrapartida, uma partícula que atinge um núcleo com alto valor da função objetivo é espalhada para outra região.

Isso permite que o espaço de busca do problema seja amplamente percorrido e que as regiões mais promissoras sejam exploradas através de eventos sucessivos de espalhamento e absorção.

Primeiramente, uma solução inicial é determinada e em seguida esta solução é modificada através de uma perturbação estocástica. As qualidades dessas possíveis soluções são comparadas e então é decidido pela manutenção ou alteração da solução atual por outra solução potencial.

Se a qualidade desta nova solução for melhor do que aquela da solução antiga, então a partícula é absorvida e ocorre a exploração das vizinhanças para que seja encontrado como resultado, uma solução ainda melhor. Na Exploração é realizada uma busca local, gerando pequenas perturbações estocásticas na solução dentro de um processo iterativo.

Se a qualidade da nova solução for pior do que aquela da solução antiga, então a partícula será espalhada. A probabilidade de espalhamento pscattering é inversamente proporcional a sua qualidade, uma vez que uma partícula de menor qualidade terá maior probabilidade de ser espalhada.

Deste modo, o PCA também pode ser considerado um algoritmo do tipo Metropolis, uma solução pode ser aceita, com determinada probabilidade, mesmo sendo sua qualidade menor do que aquela da solução antiga. Tal flexibilidade pode evitar a convergência para ótimos locais. (SACCO; KNUPP; LUZ; SILVA NETO, p. 80).

A seguir, o fluxograma do PCA é representado na Figura 10. Logo abaixo segue um breve comentário sobre cada etapa estabelecida:

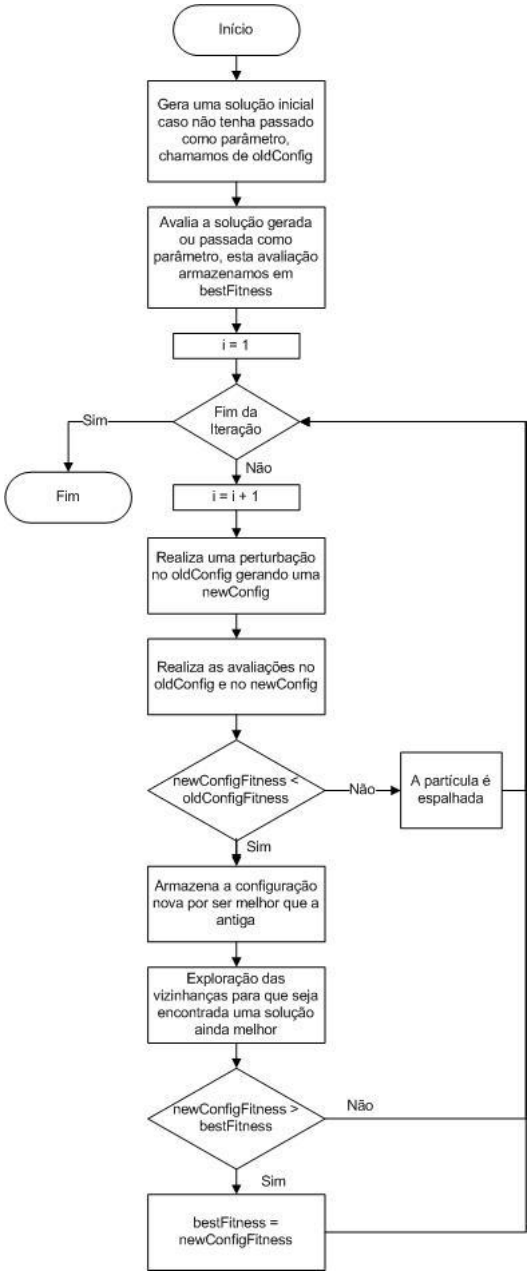

Figura 10: Fluxograma do algoritmo PCA

Passo 1: Gera uma solução inicial chamada oldConfig, caso não tenha passado como parâmetro.

Passo 2: Avalia a solução gerada ou passada como parâmetro, esta avaliação é armazenada em bestFitness.

Passo 3: Realiza uma perturbação no oldConfig gerando uma newConfig:

$$NewConfig = OldConfig + \begin{bmatrix} (U - OldConfig).r \\ -(OldConfig - L).(1 - r) \end{bmatrix} \quad (6)$$

$$NewConfig = L, se\ NewConfig < L \quad (7)$$

$$NewConfig = U, se\ NewConfig > U \quad (8)$$

U→ limite superior do parâmetro, valor máximo que pode atingir

L→ limite inferior do parâmetro, valor mínimo que pode atingir

R→ valor gerado aleatoriamente entre 0 e 1

Passo 4: Realiza as avaliações no oldConfig e no newConfig.

Passo 5: Caso a nova solução seja melhor que a antiga ele armazena a configuração nova e realiza a exploração nas vizinhanças com uma pequena perturbação. Após este trabalho é realizado um comparativo, verificando se a solução atual seria melhor que a antiga.

Passo 6: A partícula é espalhada: Caso a nova solução não seja melhor que a antiga, a partícula é espalhada, para isso será calculado a probabilidade de espalhamento $p_{scattering}$.

$$p_{scattering} = 1 - \frac{F(BestConfig)}{F(NewConfig)} \quad (9)$$

Gere um número aleatório $\mathbf{r}$. Se $\mathbf{p_{scattering}}$ for maior que $\mathbf{r}$, então OldConfig recebe NewConfig e realiza a exploração, caso contrário, volte para o inicio do algoritmo gerando uma nova solução;

1.5 Algoritmo genético

Algoritmos genéticos são inspirados no principio da evolução das espécies com base filosófica na teoria de Darwin e nos estudos da engenharia genética. São algoritmos probabilísticos, heurísticos que fornecem um mecanismo de busca paralela baseado na sobrevivência dos mais aptos e na reprodução.

O algoritmo genético possui alguns tópicos importantes: montagem do cromossoma, população inicial, avaliação, seleção, cruzamento e mutação.

A montagem do cromossoma é um dos principais itens a serem definidos, a partir deste define-se a população inicial, avaliação, seleção, cruzamento e mutação. A montagem do cromossoma basicamente consiste em como traduzir a informação do problema em uma maneira de ser tratada pelo computador, poderá ser realizado de diversas maneiras: cadeia de caracteres, números binários e etc. A representação deve ser o mais simples possível. Não existe uma regra, cada caso deve ser analisado separadamente e para isto nada melhor que o especialista que conhece bem o problema e algoritmo genético.

Para a realização da busca será necessário uma população inicial, esta população inicial pode já existir ou poderá ser gerada. Em caso de uma população inicial já existir, deverá ser realizado a busca nesta população. Caso esta população não exista será necessário a geração da população Os critérios de geração da população inicial devem visar à regra do negocio.

O algoritmo genético consiste em evoluir uma população para uma população melhor, para isto é necessário realizar uma avaliação do individuo, para que o mesmo tenha sua probabilidade de ser escolhido para realizar o cruzamento é preciso qualificá-lo numericamente. Esta avaliação consiste em minimizar ou maximizar uma função.

Em cada geração, avaliam-se todos os indivíduos e realiza-se a seleção dos melhores indivíduos para que um novo cruzamento seja realizado. O método muito utilizado para seleção é a roleta, onde todos os indivíduos têm chance de ser selecionados, porém os que têm melhor aptidão têm mais chance de serem selecionados.

Após selecionar os dois indivíduos, realiza-se o cruzamento, ou crossover. Este cruzamento deverá ser a combinação dos dois indivíduos gerando um terceiro, esta combinação pode ser realizada de diversas formas, exemplo: operação aritmética (soma, subtração, média) ou escolher um ponto de corte nos pais e copiar parte dos pais para os descendentes.

Existe um último processo e não menos importante: a mutação. Com ela serão evitados mínimos locais. Existem muitas formas de se realizar esse processo, porém o conceito seria sempre o mesmo. Selecionar alguns indivíduos e alterar parte do cromossoma, realizando assim a mutação.

A seguir, o fluxograma do AG é representado na Figura 11. Logo abaixo segue um breve comentário sobre cada etapa estabelecida:

Figura 11: Representação do Algoritmo Genético

Passo 1: Gera uma população inicial, caso não tenha recebido como parâmetro;

Passo 2: Avalia a adaptabilidade de cada indivíduo da população;

Passo 3: Seleciona os indivíduos para o cruzamento;

Passo 4: Realiza o cruzamento;

Passo 5: Realiza a mutação;

Passo 6: Verificar o critério de parada, casos sejam satisfeitas, termina-se, caso contrário vá ao passo 2;

1.6 Busca tabu

A Busca Tabu é um algoritmo para resolver problemas de otimização combinatória que compreende em buscar melhores soluções vizinhas aleatórias através do uso de uma estrutura de memória que ajuda a não ocorrer ótimos locais, sendo assim possibilitando mover-se de uma solução a outra, por todo conjunto de soluções.

A Busca Tabu foi primeiramente sugerida por: Glover, F. (1986) “Future paths for integer programming and links to artificial intelligence”, Computers & Operations Research, Vol 13, pp. 533-549.

Este algoritmo caracteriza-se por uma memória que é permitida armazenar os últimos movimentos realizados e que pode ser utilizada para recordar aqueles movimentos que fazem cair em soluções já exploradas. Esta memória serviria para impedir a evolução a partir destas soluções.

A característica importante deste algoritmo é a construção de uma lista tabu de movimentos (LT): compreende os movimentos que não são permitidos (movimentos tabus). A razão que leva a montar esta lista é a necessidade de excluir movimentos que possam nos levar a algum ponto anteriormente explorado em iterações.

Um movimento só se torna tabu durante um determinado período, sendo assim esta lista é uma lista cíclica. A primeira a entrar é a primeira a sair, *First In First Out* (FIFO), e cada vez que é realizado um movimento, este movimento é inserido na lista, sendo assim, o movimento mais velho é eliminado.

As condições tabus não são invioláveis. Quando um movimento tabu proporciona uma solução melhor que qualquer outra previamente encontrada, sua condição tabu pode ser desfeita.

Figura 12: Representação do Algoritmo Busca Tabu

Abaixo serão mostradas as principais características do algoritmo busca tabu:

a) Geração da solução inicial

   A solução inicial dependerá do algoritmo especifico que vai gerar a solução inicial, usualmente esta geração é realizada de forma aleatória.

b) Vizinhança

A busca tabu trabalha com vizinhanças. A geração das soluções vizinhas, classificando cada uma destas soluções e ficando com a melhor que não pertença ao grupo de soluções tabus.

c) Tamanho da Lista Tabu

O valor do tamanho da lista tabu deve respeitar o tamanho do problema. Este valor deve ser bem escolhido, realizado uma análise. Para alcançar boas soluções este parâmetro é muito importante.

d) Critério de Aspiração

Como descrito anteriormente, as condições tabus não são invioláveis. Quando um movimento tabu proporciona uma solução melhor que qualquer outra previamente encontrada, sua condição tabu pode ser desfeita. O critério que permite essa ação chama-se Critério de Aspiração. Existem diferente critérios de aspiração São eles:

- Aspiração por *default*. Se todos os elementos são classificados como elementos tabus, foi escolhido o elemento que seja menos tabu.

- Aspiração por objetivo: Quando um elemento proporciona um melhor custo, se aceita o elemento como solução.

- Aspiração por Busca Direcionada: Um movimento tabu torna-se admissível se a direção da busca (melhorando ou não melhorando) não é alterada.

e) Intensificação

Seu objetivo é concentrar as busca em regiões promissoras de seu espaço. Uma boa forma de intensificar é levando em consideração o histórico.

f) Diversificação

Seu objetivo é encaminhar a busca para novas regiões do espaço de busca. Uma maneira eficiente de diversificar é encorajar a busca de soluções ainda não exploradas.

g) Critério de Parada

O critério de parada pode ser por número de iterações, por tempo determinado, um exemplo: depois de x iterações sem melhorar o custo.

Para melhor exemplificar e compreender o que foi realizado no algoritmo acima, será realizado um passo a passo:

Passo 1: Geração da Vizinhança não permitindo vizinhos que estejam na lista tabu;

Passo 2: Escolha da melhor solução dentre todos os vizinhos, a avaliação consiste em minimizar ou maximizar a função objetivo;

Passo 3: Uma fase de melhorias, cujo objetivo é encontrar uma solução ótima local.

Passo 4: Atualiza S* caso a solução encontrada seja melhor que a anterior

Passo 5: Inclui o item na lista tabu

Passo 6: Verifica se está na ultima iteração, caso isso seja verdade, pára o processamento.

1.7 GRASP

O GRASP foi desenvolvido em 1995 por Tom Feo e Maurício Resende. O significado da sigla GRASP é Greedy Randomized Adaptative Search Procedure ou procedimento de busca gulosa adaptativa aleatória.

Este algoritmo é baseado em iterações, onde cada iteração possui três fases:

Fase 1: Fase de construção, na qual uma solução viável é construída.

Fase 2: Escolha aleatória do elemento que passará a fase da busca local.

Fase 3: Uma fase de melhorias, cujo objetivo é encontrar uma solução ótima local.

Figura 13: Representação do Algoritmo GRASP

O diferencial deste algoritmo esta na fase de construção monta-se um conjunto de candidatos de forma gulosa ou gananciosa, em outras palavras, considerando os melhores elementos, esta lista pode ser chamada de LRC (Lista Restrita de Candidatos). (ALVARENGA; ROCHA, 2006)

A escolha do candidato que participará da busca local é realizado de forma aleatória. Na fase de busca local a vizinhança dessa solução, ou seja, soluções semelhantes a ela são investigadas até que se encontre a melhor localmente.

Este algoritmo é adaptativo devido o elemento selecionado na iteração x foi decorrente da construção da LRC considerando o elemento da iteração x-1.

A melhor solução encontrada ao longo de todas as iterações é retornada como resultado. (ALVARENGA; ROCHA, 2006)

No GRASP básico não existe fuga de ótimo local como existe em outros métodos meta-heuristicos. Neste método, a idéia é começar com uma solução inicial e percorrer um único caminho dentro da vizinhança explorável, considerando em cada iteração o elemento da iteração anterior.

1.8 Sistemas híbridos

Todo sistema que usa mais de uma técnica (redes neurais, sistemas fuzzy, algoritmos genéticos, regressão, sistemas especialistas, técnicas de agrupamento, etc) para a solução de um problema de modelagem é considerado um sistema híbrido.

Os sistemas híbridos podem ser classificados da seguinte forma:

- Híbrido Sequencial (hibridização fraca) - Um subsistema A com paradigma A atua como entrada de um subsistema B com paradigma B.
- Híbrido Auxiliar (hibridização média) - Um subsistema B com paradigma B é chamado pelo subsistema A com paradigma A, que realiza algum serviço auxiliar.
- Híbrido Incorporado (hibridização forte) - Uma combinação entre os dois paradigmas.

2 MODELAGEM

2.1 Geração dos dados

Os dados experimentais utilizados nesta dissertação para a solução do problema inverso são sintéticos gerados a partir do método das ordenadas discretas. Consideram-se radiação incidente isotrópica, paredes refletoras difusas, espalhamento isotrópico, sem termo fonte, sem ruído, na resolução do problema direto, em outras palavras, a partir das propriedades radiativas calcularam-se as intensidades que deixam o meio.

Para geração dos dados utilizando o problema direto descrito no item 1.1, foi projetada uma interface web que está hospedada no endereço www.transferenciaradiativa.com.br. Nesta página é permitida à geração dos dados sintéticos, para utilizar esta funcionalidade será necessário o preenchimento de alguns campos:

1) Quantidade de Padrões: Quantidade de padrões que serão gerados, i.e., o número de conjuntos de propriedades mapeando intensidades que deixam o meio.
2) Quantidade de Entradas: Este campo não pode ser alterado sempre sendo quatro, representa as propriedades radiativas (ω, τ_0, ρ_1 e ρ_2).
3) Quantidade de Saídas: Quantidade de Intensidades que deixam o meio, Y.
4) Albedo de Espalhamento: Define os valores para o albedo de espalhamento, ω. Caso selecione aleatório, o sistema atribuirá um valor aleatório para o albedo, respeitando a faixa de mínimo e máximo escolhida pelo usuário. Caso selecione fixo, poderá colocar um valor fixo para este campo. O albedo de espalhamento somente pode variar entre 0 a 1.
5) Espessura óptica: Define os valores para a espessura óptica, τ_0. Caso selecione aleatório, o sistema atribuirá um valor aleatório para a espessura, respeitando a faixa de mínimo e máximo escolhida pelo usuário. Caso selecione fixo, poderá colocar um valor fixo para este campo.
6) Reflectividade 1 e 2: Define os valores para as reflectividades 1 e 2, ρ_1 e ρ_2. Caso selecione aleatório, o sistema atribuirá um valor aleatório para a

reflectividade, respeitando a faixa de mínimo e máximo escolhida pelo usuário. Caso selecione fixo, poderá colocar um valor fixo para este campo. O albedo de espalhamento somente pode variar entre 0 a 1.

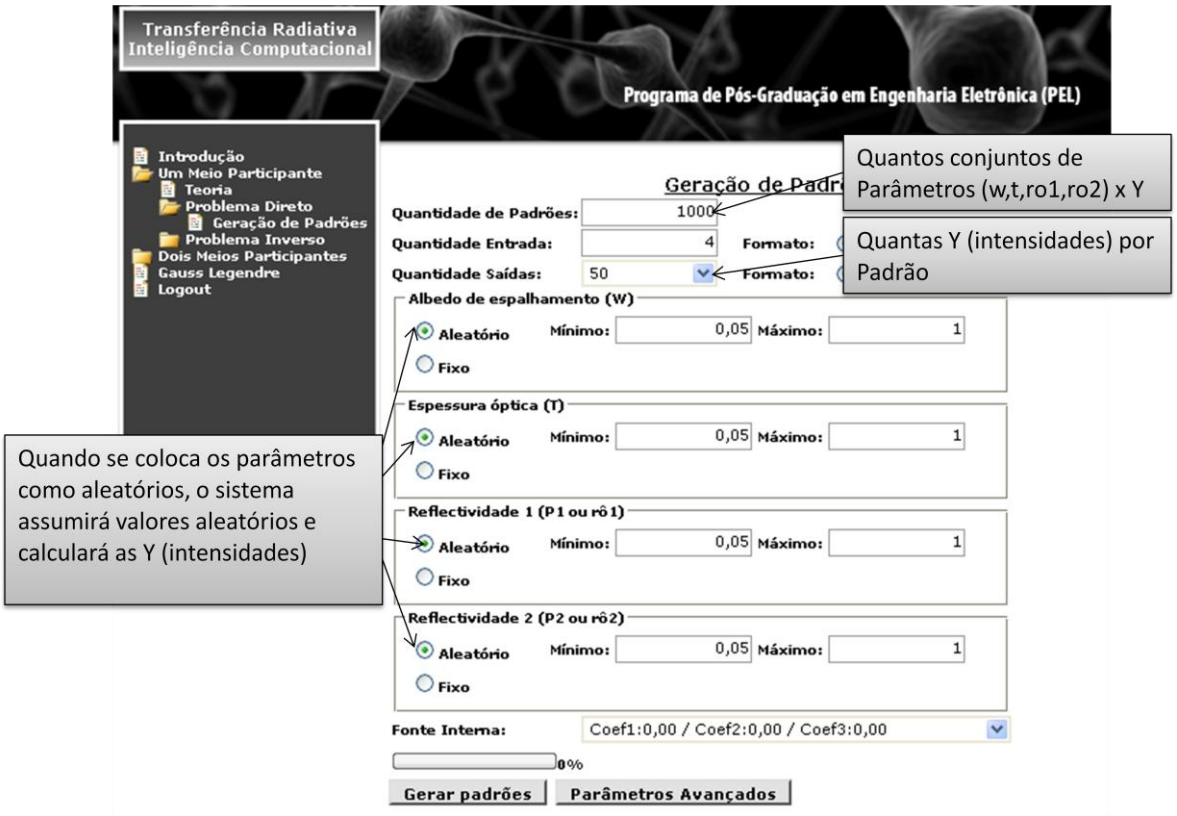

Figura 14: Página web onde foi gerada os dados sintéticos

Para o caso estudado na presente dissertação foi gerado 1000 conjuntos de propriedades radiativas (ω, τ_0, ρ_1 e ρ_2) ou quantidade de padrões, aleatórias variando cada propriedade radiativa com mínimo de 0,05 e com máximo de 1 e calculado as 50 intensidades que deixam o meio ou quantidade de saídas. Veja na figura 15 este mapeamento.

Logo, no presente problema inverso estão mapeando 50 intensidades que deixam o meio, Y, em busca das quatro propriedades radiativas (ω, τ_0, ρ_1 e ρ_2).

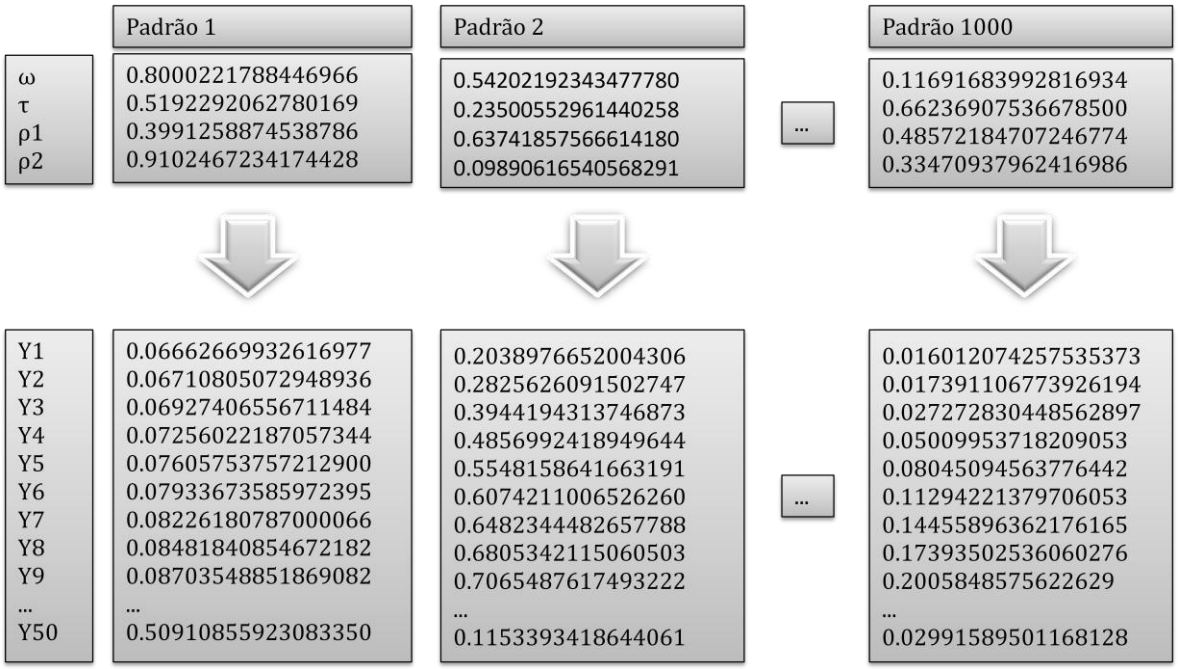

Figura 15: Mapeamento de entradas com saídas, geradas pelo problema direto

2.2 Função objetivo

Como descrito na introdução, o objetivo central desta dissertação é a solução do problema inverso de transferência radiativa utilizando técnicas combinadas de inteligência computacionais e métodos de otimização. Os métodos de otimização utilizados nesta dissertação buscam minimizar a função objetivo dada pela soma dos resíduos quadrados,

$$\boldsymbol{Q(Z)} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{Nd}} [\boldsymbol{I_i(Z)} - \boldsymbol{Y_i}]^{\boldsymbol{2}} \tag{10}$$

onde $\boldsymbol{I_i}$ e $\boldsymbol{Y_i}$ correspondem respectivamente ao valor calculado e ao valor medido experimentalmente da intensidade da radiação para a mesma posição e Nd é o numero total de dados experimentais.

2.3 Rede neural

No presente estudo utilizou uma rede neural *feed-forward*, com cinqüenta entradas, que mapeiam quatro saídas conforme a figura 16, onde estas cinqüenta entradas são as intensidades da radiação e as quatro saídas são as propriedades radiativas ω, $\tau_{0,}$ ρ_1 e ρ_2.

Na elaboração do projeto da RNA ideal para solucionar um determinado problema, uma das tarefas mais desafiante é determinar o número de elementos de processamento da camada oculta, bem como o número de camadas ocultas. Não existem regras para isto, em nosso caso realizou-se várias tentativas utilizando a faixa entre 5 a 60 neurônios, onde uma única camada escondida com 40 neurônios apresentou uma melhor generalização.

Para função de ativação foi utilizado tangente hiperbólica. Os dados foram gerados conforme o item 2.1, sendo gerado um grupo de 1.000 padrões, isto é 1.000 conjuntos de 50 intensidades mapeando 4 propriedades radiativas, que serão utilizados para treinar a rede. O software utilizado foi o toolbox de redes neurais do Matlab®.

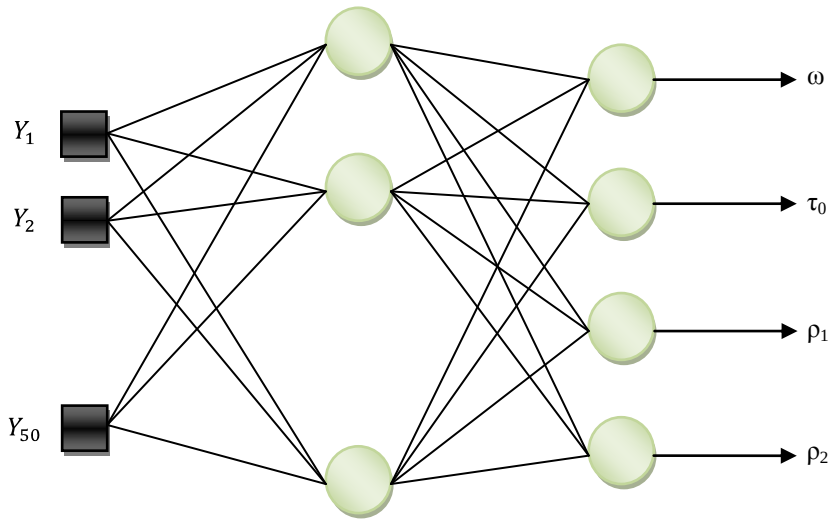

Figura 16: Rede Neural aplicada a resolver o problema inverso de transferência radiativa

Para o treinamento foi separado 1000 padrões com as quatro propriedades radiativas (ω, $\tau_{0,}$ ρ_1 e ρ_2), utilizado o algoritmo *Levenberg-Marquardt*

backpropagation ou retropropagação com tolerância de erro de $1e^{-9}$, porém somente chegou a $2{,}35e^{-5}$, com um total de 210 épocas. A partir dos dados mencionados, foram realizados 100 treinamentos para escolher a melhor rede, considerando a que generalizava melhor.

Figura 17: Treinamento da Rede Neural aplicada a resolver o problema inverso de transferência radiativa

O critério adotado de parada do treinamento da RNA foi o monitoramento do poder de generalização utilizando a validação cruzada, onde se divide os 1000 padrões em 60% para treinamento, 20% para validação e 20% para teste, podemos verificar isto na figura 18 com o treinamento da RNA utilizando a ferramenta Matlab®.

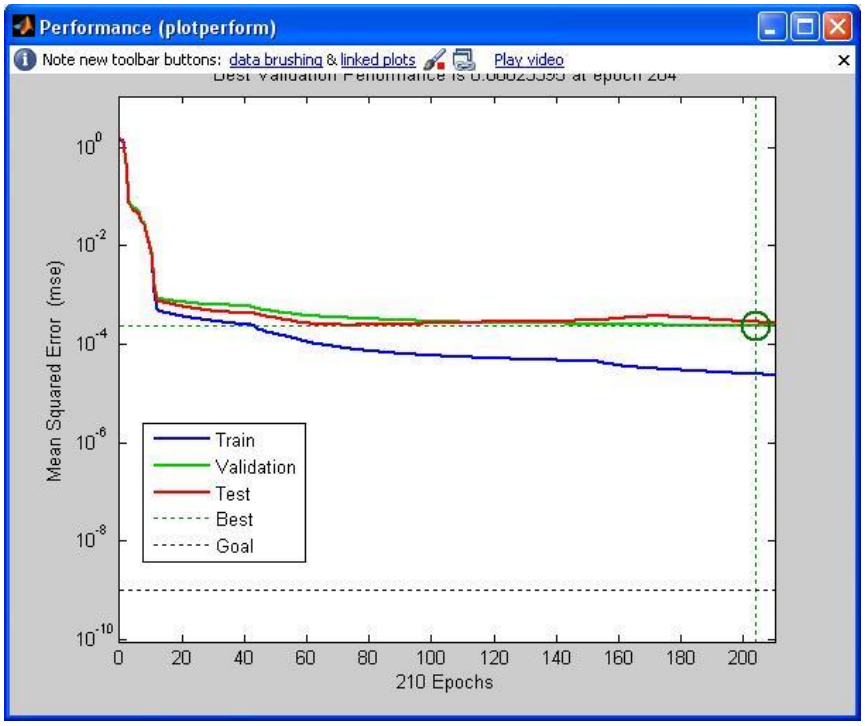

Figura 18: Treinamento da RNA

Na figura 19 apresenta-se a regressão linear da rede neural utilizando a ferramenta Matlab®:

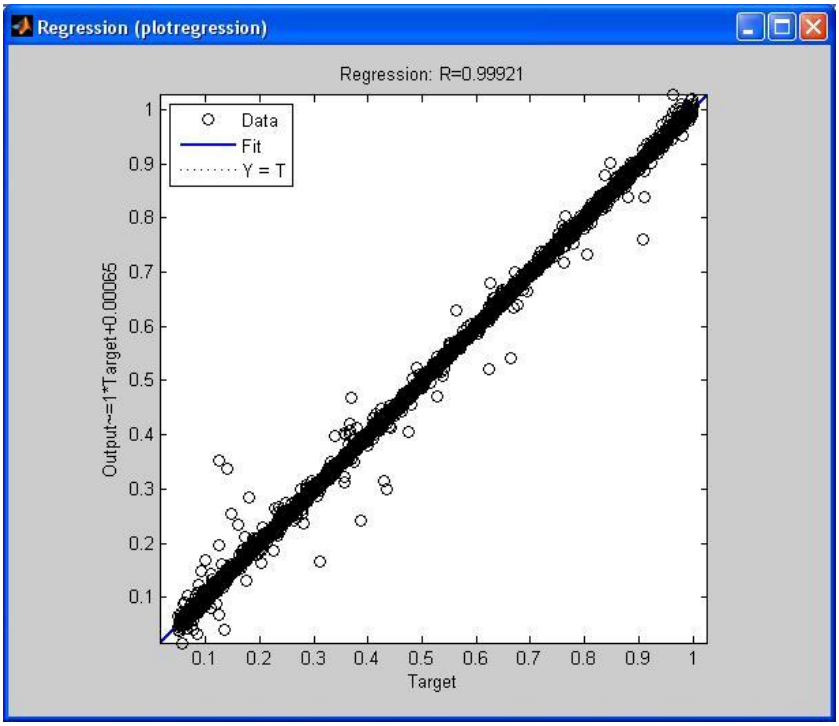

Figura 19: Regressão Linear da RNA

2.4 Hibridização: RNA combinada com PCA

Neste item utilizou-se uma solução híbrida, Rede Neural combinada com PCA, para resolver o problema inverso descrito no item 1.2.

A Rede Neural foi utilizada para identificar as quatro propriedades radiativas conforme visto anteriormente no item 2.3. Conforme descrito no objetivo de nossa dissertação, para a otimização das propriedades radiativas foi utilizado o PCA, veja na figura 20 a arquitetura da solução.

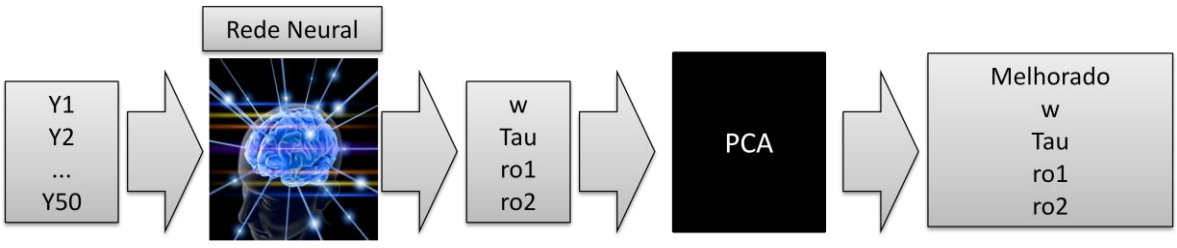

Figura 20: Arquitetura da Solução RN combinada com PCA

O Algoritmo PCA utiliza uma solução inicial, em nosso algoritmo chamamos de oldConfig e definiu-se o formato de um vetor, $[\omega\ \tau_0\ \rho_1\ \rho_2]$. Os valores das propriedades radiativas que compõe este vetor foram retirados da saída da RNA.

O PCA é um método de otimização e como tal necessita de uma avaliação numérica, conforme no item 2.2, será a realizado a minimização da função objetivo, equação (10).

O número de intensidade que deixam o meio, isto é Nd, foi considerado o valor de 50 e para o número de iterações foi considerado o valor de 100.

2.5 Hibridização: RNA combinada com Algoritmo Genético

Neste item utilizou-se uma solução híbrida, Rede Neural combinada com Algoritmo Genético, para resolver o problema inverso descrito no item 1.2.

A Rede Neural foi utilizada para identificar as quatro propriedades radiativas conforme visto anteriormente no item 2.3. Conforme descrito no objetivo de nossa dissertação, para a otimização das propriedades radiativas foi utilizado o algoritmo genético, veja na figura 21 a arquitetura da solução.

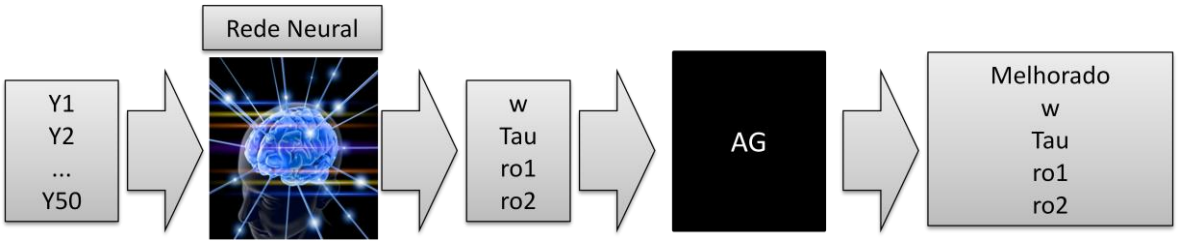

Figura 21: Arquitetura da Solução RN combinada com AG

Para a montagem do cromossoma utilizou-se um vetor com as propriedades radiativas a serem melhorados [ω, τ_0, ρ_1 e ρ_2].

Para a população inicial foi gerado 100 indivíduos onde seu cromossoma é formado pelas quatro propriedades radiativas a serem melhoradas, a geração das propriedades radiativas foi variando aleatoriamente entre 2% a 40% do valor entregue pela rede neural.

O AG é um método de otimização e como tal necessita de uma avaliação numérica, conforme o item 2.2, será a realizado a minimização da função objetivo, equação (10).

O método de seleção utilizado foi o roleta e foi utilizado o elitismo onde separa-se até 10% dos melhores indivíduos para passar para a geração seguinte, este valor de 0% a 10% é aleatório.

Para o cruzamento realizou-se a média entre as propriedades radiativas desses dois indivíduos gerando assim um terceiro indivíduo.

Selecionou-se entre 10% a 20% dos indivíduos, isto é 10 a 20 indivíduos, para realizar a mutação, esta mutação é realizada variando de 0% a 60% os valores das propriedades radiativas.

2.6 Hibridização: RNA combinada com Busca Tabu

Neste item utilizou-se uma solução híbrida, Rede Neural combinada com Busca Tabu, para resolver o problema descrito no item 1.2.

A Rede Neural foi utilizada para identificar as quatro propriedades radiativas conforme visto anteriormente no item 2.3. Conforme descrito no objetivo de nossa dissertação, para a otimização das propriedades radiativas foi utilizado o algoritmo Busca Tabu, veja na figura 22 a arquitetura da solução.

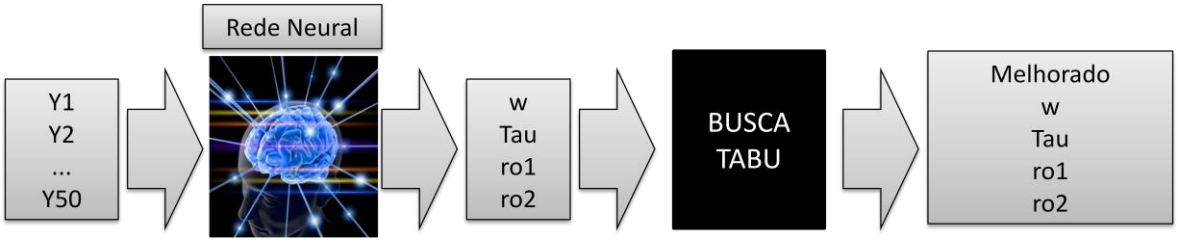

Figura 22: Arquitetura da Solução RN combinada com BUSCA TABU

A estrutura do elemento ou vizinho a ser otimizado é um vetor [ω, τ_0, ρ_1 e ρ_2].

Para vizinhança gerou-se 100 elementos, a geração desta vizinhança foi baseada na saída da RNA. A cada iteração seleciona-se o melhor vizinho, variando x% cada propriedade radiativa do vetor vizinho, onde x será um valor aleatório de 0

a 30. Dos 100 vizinhos gerados, 50 são variados para baixo (s^* - %x) e 50 são variadas para cima (s^* + %x).

Para concretizar a classificação de cada elemento, foi considerado o número de dados experimentais de 50 intensidades, isto é Nd é igual a 50. A Busca Tabu é um método de otimização e como tal necessita de uma avaliação numérica, conforme o item 2.2, será a realizado a minimização da função objetivo, equação (10).

Para preencher a busca tabu, foi escolhida uma faixa de cada parâmetro, logo foram armazenados para cada iteração, os pares ωmin e ωmax, $\tau_0$min e $\tau_0$max, $\rho_1$min e $\rho_1$max e $\rho_2$min e $\rho_2$max. Logo se o vizinho gerado estiver dentro das faixas, não será utilizado, respeitando a busca tabu, é claro, levando em consideração os critérios de aspiração, onde caso seja encontrado um elemento melhor que o encontrado até o presente momento, irá considerá-lo. Ponderando um tamanho 10 elementos para lista tabu, foram levados em conta 10% do tamanho da vizinhança.

2.7 Hibridização: RNA combinada com GRASP

Neste item utilizou-se uma solução híbrida, Rede Neural combinada com GRASP, para resolver o problema descrito no item 1.2.

A Rede Neural foi utilizada para identificar as quatro propriedades radiativas conforme visto anteriormente no item 2.3. Conforme descrito no objetivo de nossa dissertação, para a otimização das propriedades radiativas foi utilizado o algoritmo GRASP, veja na figura 23 a arquitetura da solução.

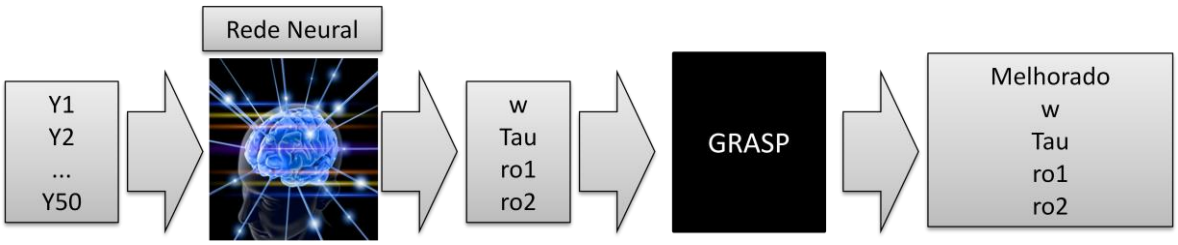

Figura 23: Arquitetura da Solução RN combinada com GRASP

Para a fase de construção, gerou uma lista variando o valor passado pela rede neural em x% e conseqüentemente, gerou uma lista de 100 elementos.

Para concretizar a classificação de cada elemento, foi considerado o número de dados experimentais de 50 intensidades, isto é Nd é igual a 50. GRASP é um método de otimização e como tal necessita de uma avaliação numérica, conforme no item 2.2, será a realizado a minimização da função objetivo, equação (10).

Ordenou-se a lista em ordem crescente, onde os primeiros elementos têm mais chance de passar para a próxima fase, já que este algoritmo é guloso.

Aleatoriamente escolheu entre os x primeiros elementos da LC, o elemento que passará para próxima fase, a fase da otimização local.

Na fase de otimização local, realizou uma busca pela vizinhança do valor fornecido na fase de construção, esta busca acontece realizando uma mudança aleatória de 2 a 20% em cada uma das quatro propriedades radiativas, tentando explorar cada vizinhança possível.

O elemento que obtiver o menor custo na função avaliação passa para a próxima fase, após todas as iterações, o melhor elemento ou que obteve o menor custo é o elemento que servirá como resultado.

3 Resultados

Neste capitulo apresentam-se os resultados encontrados, primeiramente utilizando a RNA única para resolver o problema inverso e em seguida com os sistemas híbridos onde utilizam as combinações da RNA com as técnicas de otimização.

3.1 Rede neural única

Abaixo serão apresentados os resultados encontrados na saída da RNA.

O erro médio percentual consiste no resultado da média do erro percentual das quatro propriedades radiativas, em seguida é realizado o acúmulo deste valor em uma variável, que posteriormente é dividida por 1000, que é a quantidade de padrões utilizados na RNA.

$$\boldsymbol{ErroMédioPercentual} = \frac{\sum_{\mathbf{0}}^{\mathbf{1000}} \frac{\boldsymbol{Erro1} + \boldsymbol{Erro2} + \boldsymbol{Erro3} + \boldsymbol{Erro4}}{\mathbf{4}}}{\mathbf{1000}} \qquad (\mathbf{10})$$

$$\boldsymbol{Erro1,2,3,4} = \mathbf{100} * \left(\frac{\boldsymbol{ABS(OUT - Target)}}{\boldsymbol{Target}}\right)$$

Os erros Erro1, Erro2, Erro3 e Erro4 são respectivamente os erros percentuais das propriedades radiativas ω, τ_0, ρ_1 e ρ_2. OUT é o valor da saída da RNA e o Target é o resultado esperado.

A tabela 1 apresenta as quantidades de erros em cada faixa de percentual, isto é, dos 1000 padrões utilizados, 60,3% dos padrões ficaram com erro menor que 1%, 95,5% dos padrões ficaram com erro menor que 5% e somente 0,4% dos padrões ficaram com erro acima de 25%.

Erro Médio Percentual	1,64%
Quantidade de erros < 1%	603
Quantidade de erros < 5%	954
Quantidade de erro < 10%	980
Quantidade de erro > 25%	4

Tabela 1: Tabela Com os Resultados da Rede Neural

Com o objetivo de melhorar a estimativa dessas propriedades radiativas (ω, τ_0, ρ_1 e ρ_2), selecionou os 2 melhores resultados da rede neural para serem otimizados, utilizando as técnicas PCA, Algoritmo Genético, Busca Tabu e GRASP.

Caso 1	ω	τ_0	ρ_1	ρ_2
Valor Esperado	0,23503	0,71344	0,62959	0,84789
Saída da RNA	0,23517	0,71262	0,62959	0,84912
Erro Percentual	0,0616%	0,1151%	0,0007%	0,1457%

Tabela 2: Tabela com o caso 1 das propriedades radiativas da Rede Neural

Caso 2	ω	τ_0	ρ_1	ρ_2
Valor Esperado	0,52558	0,26537	0,56653	0,74467
Saída da RNA	0,52531	0,26592	0,56710	0,74532
Erro Percentual	0,0512%	0,2081%	0,1005%	0,0873%

Tabela 3: Tabela com o caso 2 das propriedades radiativas da Rede Neural

O uso de redes neurais artificiais para estimativa do valor inicial funcionou muito bem para o estudo de transferência radiativa, apresentando um erro médio percentual de 1,64. Para continuar este trabalho, ou seja, melhorar o resultado das propriedades radiativas pelos outros métodos, escolheu os casos 1 e 2, estes casos apresentaram o menor erro percentual.

3.2 Hibridização: RNA combinada com PCA

Agora serão apresentados os resultados para a utilização da hibridização combinando RNA com PCA para resolver o problema inverso descrito no item 1.2.

Caso 1	ω	τ_0	ρ_1	ρ_2
Valor Esperado	0,23503	0,71344	0,62959	0,84789
Saída da RNA Única	0,23517	0,71262	0,62959	0,84912
RNA Combinada com PCA	0,23679	0,71155	0,63287	0,84845
Erro Percentual da RNA	0,0616%	0,1151%	0,0007%	0,1457%
Erro Percentual da RNA Combinada com PCA	0,7513%	0,2655%	0,5219%	0,0663%

Tabela 4: Tabela com o resultado RNA combinado com PCA para o caso 1

O método RNA combinado com o PCA piorou o erro médio percentual comparando com a saída da RNA única de 0,0808% para 0,4012%, considerando como referencial o valor esperado, isto é, valor gerado pelo problema direto.

Caso 2	ω	τ_0	ρ_1	ρ_2
Valor Esperado	0,52558	0,26537	0,56653	0,74467
Saída da RNA Única	0,52531	0,26592	0,56710	0,74532
RNA Combinada com PCA	0,52401	0,26631	0,56287	0,74307
Erro Percentual da RNA	0,0512%	0,2081%	0,1005%	0,0873%
Erro Percentual da RNA Combinada com PCA	0,2987%	0,3561%	0,6449%	0,2159%

Tabela 5: Tabela com o resultado RNA combinado com PCA para o caso 2

O método RNA combinado com o PCA piorou o erro médio percentual comparando com a saída da RNA única de 0,1118% para 0,3789%, considerando como referencial o valor esperado, isto é, valor gerado pelo problema direto.

O uso do algoritmo PCA para melhoria do valor entregue pela Rede Neural não se demonstrou um bom método para o estudo de transferência radiativa, nos dois casos ele piorou o erro percentual.

3.3 Hibridização: RNA combinada com algoritmo genético

Agora serão apresentados os resultados para a utilização da hibridização combinando RNA com AG para resolver o problema inverso descrito no item 1.2.

Caso 1	ω	τ_0	ρ_1	ρ_2
Valor Esperado	0,23503	0,71344	0,62959	0,84789
Saída da RNA Única	0,23517	0,71262	0,62959	0,84912
RNA Combinada com AG	0,23525	0,71352	0,62977	0,84790
Erro Percentual da RNA	0,0616%	0,1151%	0,0007%	0,1457%
Erro Percentual da RNA Combinada com AG	0,0961%	0,0108%	0,0290%	0,0011%

Tabela 6: Tabela com o resultado RNA combinado com AG para o caso 1

O método RNA combinado com o AG melhorou o erro médio percentual comparando com a saída da RNA única de 0,0808% para 0,0343%, considerando como referencial o valor esperado, isto é, valor gerado pelo problema direto.

Caso 2	ω	τ_0	ρ_1	ρ_2
Valor Esperado	0,52558	0,26537	0,56653	0,74467
Saída da RNA Única	0,52531	0,26592	0,56710	0,74532
RNA Combinada com AG	0,52567	0,26582	0,56577	0,74444
Erro Percentual da RNA	0,0512%	0,2081%	0,1005%	0,0873%
Erro Percentual da RNA Combinada com AG	0,0173 %	0,1686 %	0,1329 %	0,0315 %

Tabela 7: Tabela com o resultado RNA combinado com AG para o caso 2

O método RNA combinado com o AG melhorou o erro médio percentual comparando com a saída da RNA única de 0,1118% para 0,0876%, considerando como referencial o valor esperado, isto é, valor gerado pelo problema direto.

O uso do algoritmo genético para melhoria dos valores entregues pela Rede Neural demonstrou ser um bom método para melhoria das propriedades radiativas no estudo de transferência radiativa, nos dois casos ele melhorou o erro percentual.

3.4 Hibridização: RNA combinada com busca tabu

Agora serão apresentados os resultados para a utilização da hibridização combinando RNA com busca tabu para resolver o problema inverso descrito no item 1.2.

Caso 1	ω	τ_0	ρ_1	ρ_2
Valor Esperado	0,23503	0,71344	0,62959	0,84789
Saída da RNA Única	0,23517	0,71262	0,62959	0,84912
RNA Combinada com busca tabu	0,23515	0,71391	0,62940	0,84784
Erro Percentual da RNA	0,0616%	0,1151%	0,0007%	0,1457%
Erro Percentual da RNA Combinada com busca tabu	0,0519%	0,0655%	0,0297%	0,0056%

Tabela 8: Tabela com o resultado RNA combinado com busca tabu para o caso 1

O método RNA combinado com o busca tabu melhorou o erro médio percentual comparando com a saída da RNA única de 0,0808% para 0,0382%, considerando como referencial o valor esperado, isto é, valor gerado pelo problema direto.

Caso 2	ω	τ_0	ρ_1	ρ_2
Valor Esperado	0,52558	0,26537	0,56653	0,74467
Saída da RNA Única	0,52531	0,26592	0,56710	0,74532
RNA Combinada com busca tabu	0,52614	0,26607	0,56570	0,74436
Erro Percentual da RNA	0,0512%	0,2081%	0,1005%	0,0873%
Erro Percentual da RNA Combinada com busca tabu	0,1078%	0,2650%	0,1464%	0,0418%

Tabela 9: Tabela com o resultado RNA combinado com busca tabu para o caso 2

O método RNA combinado com o busca tabu piorou o erro médio percentual comparando com a saída da RNA única de 0,1118% para 0,1402%, considerando como referencial o valor esperado, isto é, valor gerado pelo problema direto.

O uso do algoritmo Busca Tabu para melhoria do valor entregue pela Rede Neural não demonstrou um bom método para o estudo de transferência radiativa, em um caso ele melhorou e em outro ele piorou.

3.5 Hibridização: RNA combinada com GRASP

Agora serão apresentados os resultados para a utilização da hibridização combinando RNA com GRASP para resolver o problema inverso descrito no item 1.2.

Caso 1	ω	τ_0	ρ_1	ρ_2
Valor Esperado	0,23503	0,71344	0,62959	0,84789
Saída da RNA Única	0,23517	0,71262	0,62959	0,84912
RNA Combinada com GRASP	0,23464	0,71291	0,62954	0,84792
Erro Percentual da RNA	0,0616%	0,1151%	0,0007%	0,1457%
Erro Percentual da RNA Combinada com GRASP	0,1662%	0,0744%	0,0080%	0,0038%

Tabela 10: Tabela com o resultado RNA combinado com GRASP para o caso 1

O método RNA combinado com o GRASP melhorou o erro médio percentual comparando com a saída da RNA única de 0,0808% para 0,0631%, considerando como referencial o valor esperado, isto é, valor gerado pelo problema direto.

Caso 2	ω	τ_0	ρ_1	ρ_2
Valor Esperado	0,52558	0,26537	0,56653	0,74467
Saída da RNA Única	0,52531	0,26592	0,56710	0,74532
RNA Combinada com GRASP	0,52446	0,26582	0,56365	0,74349
Erro Percentual da RNA	0,0512%	0,2081%	0,1005%	0,0873%
Erro Percentual da RNA Combinada com GRASP	0,2125%	0,1688%	0,5076%	0,1590%

Tabela 11: Tabela com o resultado RNA combinado com GRASP para o caso 2

O método RNA combinado com o GRASP piorou o erro médio percentual comparando com a saída da RNA única de 0,1118% para 0,2620%, considerando como referencial o valor esperado, isto é, valor gerado pelo problema direto.

O uso do algoritmo GRASP para melhoria do valor entregue pela Rede Neural não demonstrou ser um bom método para o estudo de transferência radiativa, em um caso ele melhorou e em outro ele piorou.

3.6 Comparativo entre os métodos

	RN	RN e PCA	RN e AG	RN e GRASP	RN e busca tabu
Caso 1	0,0808%	0,4012%	0,0343%	0,0631%	0,0382%
Caso 2	0,1118%	0,3789%	0,0876%	0,2620%	0,1402%

Tabela 12: Tabela Comparativa dos Percentuais de Erros de Todos os Métodos

O único método que otimizou os dois casos, isto é, diminuiu o erro médio para os dois casos, foi o AG. Os métodos GRASP e Busca Tabu apresentaram um bom resultado, porém no caso 2 não houve melhoria e o método que não apresentou otimização foi o PCA, nos dois casos apresentados eles piorou o resultado da Rede Neural.

4 CONCLUSÃO E TRABALHOS FUTUROS

Nesta dissertação apresentou-se diversas técnicas combinadas de inteligência computacional e métodos de otimização para a solução do problema inverso de transferência radiativa, considerando um meio unidimensional, homogêneo, espalhador isotrópico, de espessura óptica τ_0, com superfícies refletoras difusas (refletividades iguais a ρ_1 e ρ_2), submetidas à incidência de radiação externa conforme figura 1 do item 1.2.

A RNA única apresentou um resultado satisfatório, os resultados da RNA única já seriam suficientes para atingir os objetivos, onde apresentou um erro médio entre 1% a 2%.

O método combinando RNA com AG apresentou um excelente resultado para estimativa das propriedades radiativas. Foi necessário adaptar a geração da população inicial para a variação percentual próxima do valor do erro encontrado pela RNA. No AG adotou-se um cruzamento aritmético, isto é, uma média aritmética entre os dois indivíduos gerando um terceiro. Devido este fato o AG aproximou-se do resultado esperado melhor que os demais métodos, onde apresentou erros percentuais menores que 0,03%.

O método RNA combinado com GRASP demonstrou um resultado abaixo do esperado, em alguns casos não foi possível realizar a otimização porque encontraram mínimos locais, o GRASP tem como característica uma solução inicial gulosa e em seguida de uma busca local, onde estas características levam a produzir pouca variedade de solução e a mínimos locais.

O método RNA combinado com Busca Tabu demonstrou um resultado abaixo do esperado, em alguns casos não foi possível realizar a otimização. A lista tabu é implementada para fugir de mínimos locais, em alguns casos este processo não foi realizado com sucesso.

O método RNA combinado com PCA demonstrou um resultado abaixo dos demais para estimativa das propriedades radiativas, em todos os casos analisados não houve otimização, isto devido a encontrar mínimos locais onde a probabilidade de espalhamento não foi suficiente para buscar outros locais, onde melhores estimativas para os parâmetros poderiam ser encontrados.

Como trabalhos futuros poderiam ser desenvolvidas as mesmas técnicas para o problema inverso, considerando um meio participante, isto é, absorvedor, espalhador e emissor, unidimensional, composto por duas camadas de espessura L_1 e L_2, com paredes internas difusas, sujeito a incidência de radiação com intensidades F_1 e F_2. Diversas aplicações relevantes e atuais estão relacionadas a esta configuração sendo citados como exemplos: ótica hidrológica, sensoriamento remoto e modelos climáticos.

Outras técnicas poderiam ser testadas, tais como: recozimento simulado e colônia de formigas. Os métodos de otimização utilizados nesta dissertação poderiam ser utilizados únicamente, sem a utilização da Rede Neural.

Além destes futuros trabalhos, também serão implementadas estas técnicas na interface web www.transferenciaradiativa.com.br para disponibilizar para o meio acadêmico o estudo de transferência radiativa, problema inverso e as técnicas utilizadas nesta dissertação.

5 REFERÊNCIAS

ALVARENGA, Fabiano V. de; ROCHA, Marcelo L., Uma metaheurística grasp para o problema da Árvore geradora de custo mínimo com grupamentos utilizando grafos fuzzy. INFOCOMP Journal of Computer Science. 5(1):66–75, 2006.

BAL, Guillaume; SCHOTLAND, John C. Inverse scattering and acousto-optic imaging. Department of Applied Physics and Applied Mathematics, Columbia University, New York; Department of Bioengineering and Graduate Group in Applied Mathematics and Computational Science, University of Pennsylvania, Philadelphia, 2009.

BECCENERI, J. C.; STEPHANY, S.; CAMPOS VELHO, H. F.; SILVA NETO, A. J.. Solution of the inverse problem of radiative properties estimation with the particle swarm optimization technique, Proceedings of 14th Inverse Problems in Engineering Seminar, Ames, EUA, 2006.

BERROCAL TITO, M. J.. Aplicações de algoritmos baseados na distância de Bregman para a solução de problemas inversos em transferência radiativa. 2006. 136 f. Tese (Doutorado em Ciências em Engenharia Nuclar) - Universidade Federal do Rio de Janeiro, COPPE, 2006.

BIONDI NETO Luiz. Inteligência computacional aplicada a problemas inversos de transferência radiativa. II Workshop Problemas inversos e inteligência computacional, 2008.

BIONDI NETO, Luiz; COELHO, P. H. G.; VELLOSO, Maria Luiza Fernandes; SOARES DE MELLO, J.C.C.B; MEZA, Lidia Angulo. Monthly Flow Estimation Using Elman Neural Networks. In: Sixth International Conference on Enterprise Information Systems, 2004, Porto. Proceedings of the Sixth International Conference on Enterprise Information Systems. Porto : INSTICC - Institute for Systems and Technologies of Informatio0n, Control and Communication, v. 2. p. 153-158, 2004.

BIONDI NETO, Luiz ; COELHO, P. H. G. ; ANTUNES, Carlos A Henggeler de C . Identificação de Falhas em Transformadores de Energia, Usando-se Redes Neurais Artificiais. In: I CONGRESSO DE ESTATÍSTICA E INVESTIGAÇÃO OPERACIONAL DA GALIZA E NORTE DE PORTUGAL E VI CONGRESSO GALEGO DE ESTATÍSTICA E INVESTIGATIÓN DE OPERACIÓNS, 2005, Guimarâes. Actas do CEIO 2005. Azarém : Universidade do Minho, 2005.

BIONDI NETO Luiz; SOEIRO, FRANCISCO J.C.P.; SILVA NETO, Antônio José. Estimation Of The Absorption And Scattering Coefficients In Two-Layer Heterogeneous Participating Media With Committees Of Artificial Neural Networks, 8TH World Congress On Structural And Multidisciplinary Optimization, 2009.

BORGES, Pedro A. P.; FENGLER, Caroline; CERVI, Angéli. Estimativa da difusividade térmica de grãos de soja pelo método da compactação. Revista Brasileira de Engenharia Agrícola e Ambiental v.13, n.5, p.591–595, Campina Grande, PB, 2009.

BRAGA, Antônio de Pádua. Redes Neurais Artificiais. Rio de Janeiro: LTC, 2007.

CAMPOS VELHO, Haroldo Fraga, Problemas Inversos: Conceitos Básicos e Aplicações, Mini – Curso IV Encontro de Modelagem Computacional, 2001.

CHALHOUB, E. S. ; SILVA NETO, A. J. ; SOEIRO, F. J. C. P. . Estimation of Optical Thickness and Single Scattering Albedo with Artificial Neural Networks and a Monte Carlo Method. In: Inverse Problems, Desing and Optimization Symposium, 2007, Miami. Inverse Problems, Desing and Optimization Symposium (IPDO 2007), 2007. v. II. p. 576-583.

CHAVES, Antônio Augusto et al. Modelagens Exata E Heurística Para Resolução De Uma Generalização Do Problema Do Caixeiro Viajante. XXXXVI – SBPO. São João Del Rei - MG, Brasil, 2004.

COELHO, P. H. G. ; BIONDI NETO, Luiz ; GOMES, Eliane Gonçalves ; MEZA, Lidia Angulo . Representação em Espaço de Estado de Redes Neurais Recorrentes Complexas para Comunicações Móveis. Engevista (UFF), Niterói, v. 5, n. 8, p. 44-55, 2003.

COSTA, Felippe Pereira; FREITAS SOUZA, Marcone Jamilson, Programação de Horários em Escolas via GRASP e Busca Tabu. Monografia de graduação em Engenharia de Produção, Ouro Preto, MG, Brasil, 2003.

FEO T.A.; RESENDE M.G.C.. A probabilistic heuristic for a computationally diffcult set covering problem. Operations Research Letters, 8:67-71, 1989. T.A. Feo and M.G.C. Resende. Greedy randomized adaptive search procedures. Journal of Global Optimization, 6:109-133, 1995.

GALVÃO, Lênio Soares; FORMAGGIO, Antônio Roberto; BREUNIG, Fábio Marcelo. Relações entre índices de vegetação e produtividade de soja com dados de visada fora do nadir do sensor Hyperion/EO-1. Anais XIV Simpósio Brasileiro de Sensoriamento Remoto, Natal, Brasil, INPE, p. 1095-1102, 2009.

GIL, Mauro Cesar Cantarino et al. Aplicação de um Método de Comitês de Redes Neuro-Fuzzy Combinado com Redes Neurais Artificiais em Cascata para a Solução de Problemas Inversos em Transferência Radiativa. XI Encontro de Modelagem Computacional, Volta Redonda. Encontro de Modelagem Computacional, 2008.

GLOVER, F. Future paths for integer programming and links to artificial intelligence”, Computers & Operations Research, Vol 13, pp. 533-549, 1986.

GRASP (algoritmo) - wikipédia, a enciclopédia livre, 2009. Disponível em: <http://pt.wikipedia.org/wiki/GRASP_(algoritmo)>. Acesso em: 10 ago. 2009.

HAYKIN, S., Redes Neurais: Princípios e Prática. 2.ª edição, Porto Alegre: Bookman, 2001.

IRISHINA, Natalia. Microwave medical imaging using level set techniques. 2009. 127 f. Tese (Doctorado en Ingeniería Matemática) - Departamento de Ciencia e Ingeniería de Materiales e Ingeniería Química, Universidad Carlos III de Madrid, Madrid, Spain, 2009.

JERG, Matthias Peter. Solar radiative transfer parameterizations for three-dimensional effects in cloudy atmospheres. 2006. 199 F. Dissertação (Faculty of Physics) - Faculty of Physics - Faculty of Physics, LMU München, Deutschland, 2006.

LEITE, MARCIO; ARROYO, José Elias Cláudio. Algoritmo busca tabu para a minimização do tempo de processamento e atrasos de entrega em sistemas de produção flowshop permutacional, XXVI ENEGEP - Fortaleza, CE, Brasil, 2006

LINDEN, Ricardo. Algoritmos Genéticos. 2. Ed. – Rio de Janeiro: Brasport, 2008. ISBN 978-85-7452-373-6

MITROFANOV, Georgy et al. Utilização das transformadas de laplace e fourier-bessel na modelagem de meios elásticos delgados. Revista Brasileira de Geofísica (2009) 27(2): 205-224. Sociedade Brasileira de Geofísica, 2009.

ORBANICHA, Claudio J. et al. Detección de fallas en vigas de fundación elástica mediante el metodo inverso. Asociación Argentina de Mecánica Computacional Vol XXVIII, págs. 613-631. Tandil, Argentina, 3-6 Noviembre 2009.

OZISIK, M. N. Radioative transferer and interaction with conduction and convection. New York, 1973.

PASCHOALINO, F.F.; LOUREIRO, T.Y.C.; SOARES DE MELLO, J.C.C.B.; BIONDI NETO, L. Previsão de demanda de energia elétrica no Brasil utilizando redes neurais de Elman. Anais do SPOLM, Rio de Janeiro, 2007.

REZENDE, Solange Oliveira. Sistemas Inteligentes – Fundamentos e Aplicações. Barueri, SP: Manole 2005. ISBN 85-204-1683-7.

SACCO, W. F.; KNUPP, Diego Campos; LUZ, Eduardo Fávero Pacheco da; SILVA NETO, Antônio José. Algoritmo de Colisão de Partículas (Particle Collision Algorithm). In: BECCENERI, José Carlos; SILVA NETO, Antônio José. Técnicas De Inteligência Computacional Inspiradas Na Natureza: Aplicação Em Problemas Inversos Em Transferência Radiativa. Sociedade Brasileira de Matemática Aplicada e Computacional, São Carlos – SP, Brasil, 2009.

SACCO, W.F.; OLIVEIRA, C. R. E. A New Stochastic Optimization Algorithm based on Particle Collisions, 2005 ANS Annual Meeting, Transactions of the American Nuclear Society, American Nuclear Society, San Diego, CA, 2005.

SILVA NETO, Antônio José. Problemas Inversos:Aplicações em Engenharia e Medicina - Um Breve Texto sobre Engenharia, Medicina, Matemática, Neurônios, Genes e Formigas. II Workshop Problemas inversos e inteligência computacional, Rio de Janeiro, Brasil, 2008.

SILVA NETO, Antônio José; CAMPOS VELHO, H. F.. Problemas Inversos em Transferência Radiativa - Uma Formulação Implícita. In: SILVA NETO, Antônio José; BECCENERI, J. C. Técnicas De Inteligência Computacional Inspiradas Na Natureza: Aplicação Em Problemas Inversos Em Transferência Radiativa. São Carlos: SBMAC, 2009. v. 41. 122 p.

SOARES, Patrícia O.; SILVA NETO, Antônio José; SOEIRO, Francisco J.C.P. Utilização de Redes Neurais para Identificação de Parâmetros em Transferência Radiativa, ABCM, 2004.

SOEIRO, Francisco J.C.P.; BECCENERI, José Carlos; SILVA NETO, Antônio José. Recozimento Simulado (Simulated Annealing). In: SILVA NETO, Antônio José; BECCENERI, J. C. Técnicas De Inteligência Computacional Inspiradas Na Natureza: Aplicação Em Problemas Inversos Em Transferência Radiativa. São Carlos: SBMAC, 2009. v. 41. 122 p.

SOUZA, F. L. de; SOEIRO, F. J. C. P; SILVA NETO, A. J.; RAMOS, F. M.. Application of the generalized extremal optimization algorithm to an inverse radiative transfer problem, Inverse Problems in Science and Engineering, v. 15, 7 (2007), 699-714.

T. TARVAINEN, et. al. Utilizing the Radiative Transfer Equation in Optical Tomography. PIERS ONLINE, VOL. 4, NO. 6, p. 655-660, 2008.

WAGNER, T.; BEIRLE, S.; DEUTSCHMANN, T.: Three-dimensional simulation of the Ring effect in observations of scattered sun light using Monte Carlo radiative transfer models, Atmos. Meas. Tech., 2, 113-124, 2009

YEBRA, Marta; CHUVIECO, Emilio. Linking ecological information and radiative transfer models to estimate fuel moisture content in the Mediterranean region of Spain: Solving the ill-posed inverse problem. Remote Sensing of Environment, Volume 113, Issue 11, Pages 2403-2411, 2009.

APÊNDICE A – Códigos Fontes do PCA

1) Função pca – Para melhoria dos parâmetros de Transferência Radiativa em uma camada:

```
function [oldConfig] = pca(Y, iteracao, superiorLimit, inferiorLimit, original, wOriginal, tauOriginal, ro1Original, ro2Original)
% Descrição: Função PCA para melhoria de parâmetros de transferencia radiativa 1 Camada.
% Parametros da Função:
% Y - Intensidades medidas experimentalmente
% iteracao - Quantidade de iterações
% superiorLimit - Valor máximo dos parâmetros
% inferiorLimit - Valor mínimo dos parâmetros
% original - Valores Possíveis 1 e 2,
% 1 - utiliza os parâmetros passados como parametros
% 2 - Gera os parâmetros aleatórios
% wOriginal, tauOriginal, ro1Original, ro2Original - Parâmetros passados,
% Retorna o resultado encontrado

   % Gera uma solução inicial caso não tenha passado como parâmetro
   if (original == 1)
      w=wOriginal;
      tau=tauOriginal;
      ro1=ro1Original;
      ro2=ro2Original;
   else
      w=rand();
      tau=rand();
      ro1=rand();
      ro2=rand();
   end
   oldConfig = [w,tau,ro1,ro2];

   % Avalia a solução gerada ou passada como parâmetro
   bestFitness=fitnessPCA(oldConfig, Y);

   for i=1: iteracao

      i
      % Realiza uma perturbação no oldConfig gerando uma newConfig
      newConfig = perturbation(oldConfig, superiorLimit, inferiorLimit);
      % Realiza uma perturbação no oldConfig gerando uma newConfig

      % Realiza as avaliações no oldConfig e no newConfig
      newConfigFitness = fitnessPCA(newConfig, Y);
      oldConfigFitness = fitnessPCA(oldConfig, Y);
```

```
        % Utilizado para debug
        sprintf('pca n w=%f tau=%f ro1=%f ro2=%f  fit=%f', newConfig(1), newConfig(2),
newConfig(3), newConfig(4), newConfigFitness)
        sprintf('pca o w=%f tau=%f ro1=%f ro2=%f  fit=%f', oldConfig(1), oldConfig(2),
oldConfig(3), oldConfig(4), oldConfigFitness)

        % Verifica se a avaliação nova é menor que a antiga
        if (newConfigFitness < oldConfigFitness)

            % Armazena a configuração nova por ser melhor que a antiga
            oldConfig = newConfig;

            % Exploração das vizinhanças para que seja encontrada uma solução melhor
            oldConfig = exploitation(iteracao, oldConfig, superiorLimit, inferiorLimit, Y);

            % Verifica se a avaliação nova é melhor que a melhor armazenada
            if (newConfigFitness > bestFitness)
                bestFitness = newConfigFitness;
            end

        else
            % A partícula é espalhada.
            % A probabilidade de espalhamento é inversamente proporcional a sua
qualidade
            oldConfig = scattering(iteracao, newConfig, oldConfig, Y, bestFitness,
original, wOriginal, tauOriginal, ro1Original, ro2Original, superiorLimit, inferiorLimit);
        end
    end

end
```

2) Função perturbation **-** Função que realiza uma perturbação no oldConfig gerando uma newConfig

```
function [newConfig] = perturbation(oldConfig, superiorLimit, inferiorLimit)
% Descrição: Realiza uma perturbação no oldConfig gerando uma newConfig
% Parâmetros da Função:
% oldConfig: Resultado antigo
% superiorLimit: Limite superior dos parâmetros
% inferiorLimit: Limite inferior dos parâmetros
% Retorna um novo resultado

    % Calcula a dimensão do oldConfig
    dimension = size(oldConfig, 2);

    % Para todos os parâmetros realiza o cálculo
    for i=1:dimension

        % Busca um valor randômico para realizar a perturbação
```

```
        valRand=rand();

        % Realiza a perturbação, variando o valor
        newConfig(i)=oldConfig(i) + ((superiorLimit(i) - oldConfig(i)) * valRand) -
((oldConfig(i) - inferiorLimit(i)) * (1-valRand));

        % Se ultrapassou um dos limites, considera o limite como novo valor
        if (newConfig(i) > superiorLimit(i))
            newConfig(i) = superiorLimit(i);
        else
            if (newConfig(i) < inferiorLimit(i))
                newConfig(i) = inferiorLimit(i);
            end
        end
    end
end
```

3) Função exploitation - Função que explora as vizinhanças para que seja encontrada uma solução ainda melhor

```
function [newConfig] = exploitation(iteracao, oldConfig, superiorLimit, inferiorLimit, Y)
% Descrição: Exploração das vizinhanças para que seja encontrada uma solução
ainda melhor
% Parâmetros da função:
% iteracao: Quantidade de iterações
% oldConfig: Resultado antigo
% superiorLimit: Limite superior dos parâmetros
% inferiorLimit: Limite inferior dos parâmetros
% Y - Intensidades medidas experimentalmente
% Retorna um novo resultado

    % Realiza um loop pela quantidade de iterações definida
    for n=1: iteracao

        % realiza uma pequena perturbação na solução antiga, gerando um nova
solução
        newConfig = smallPertubation(oldConfig, superiorLimit, inferiorLimit);

        % Avalia a solução antiga e a nova
        newFitness = fitnessPCA(newConfig, Y);
        oldFitness = fitnessPCA(oldConfig, Y);

        % Verifica se a antiga é melhor que a nova, se for fica com a antiga
        if (oldFitness < newFitness)
            newConfig = oldConfig;
        end
    end
```

```
end
```

4) Função smallPertubation **-** Função que gera uma pequena perturbação nos parâmetros

```
function [newConfig] = smallPertubation(oldConfig, superiorLimit, inferiorLimit)
%Descrição: Gera uma pequena pertubarção nos parâmetros
% Parâmetros:
% oldConfig: Resultado antigo
% superiorLimit: Limite superior dos parâmetros
% inferiorLimit: Limite inferior dos parâmetros
% Retorna um novo resultado

   % Calcula a dimensão, isto é, o numero de parâmetros
   dimension = size(oldConfig, 2);

   % Faz um loop variando cada parâmetro
   for i=1:dimension
      % Realiza uma pequena variação, aleatória
      r=rand();
      U = (1 + (0.2 * r)) * oldConfig(i);
      L = (1 + (-0.2 * r)) * oldConfig(i);

      newConfig(i)=oldConfig(i) + ((U - oldConfig(i)) * r) - ((oldConfig(i) - L) * (1-r));

      % Se ultrapassou um dos limites, considera o limite como novo valor
      if (newConfig(i) > superiorLimit(i))
         newConfig(i) = superiorLimit(i);
      end
      if (newConfig(i) < inferiorLimit(i))
         newConfig(i) = inferiorLimit(i);
      end

   end
end
```

5) Função scattering **-** Função que espalha a partícula, a probabilidade de espalhamento é inversamente proporcional a sua qualidade, ajuda a não cair em mínimos locais

```
function [oldConfig] = scattering(iteracao, newConfig, oldConfig, Y, bestFitness,
original, wOriginal, tauOriginal, ro1Original, ro2Original, superiorLimit, inferiorLimit)
% Descrição: A probabilidade de espalhamento é inversamente proporcional a sua
qualidade -------
% Parâmetros da função:
% iteracao: Quantidade de iterações
% newConfig: Resultado novo
% oldConfig: Resultado antigo
```

```
% Y: Intensidades medidas experimentalmente
% bestFitness: Melhor resultado até o momento
% original: Valores Possíveis 1 e 2,
% 1 - utiliza os parâmetros passados como parâmetro
% 2 - Gera os parâmetros aleatórios
% wOriginal, tauOriginal, ro1Original, ro2Original - Parâmetros passados,
% superiorLimit: Limite superior dos parâmetros
% inferiorLimit: Limite inferior dos parâmetros
% Retorna um novo resultado

    % Calcula a probabilidade de espalhamento
    newFitness = fitnessPCA(newConfig, Y);
    pScattering = 1 - (bestFitness / newFitness);

    % Gera um número aleatório de 0 a 1
    valRand=rand();

    % Se a probabilidade de espalhamento for maior o valor aleatório
    if (pScattering > valRand)
        % Explora a nova solução
        oldConfig = newConfig;
        oldConfig = exploitation(iteracao, oldConfig, superiorLimit, inferiorLimit, Y);
    else
        % Gera uma população inicial caso não tenha passado como parâmetro
        if (original == 1)
            w=wOriginal;
            tau=tauOriginal;
            ro1=ro1Original;
            ro2=ro2Original;
        else
            w=rand();
            tau=rand();
            ro1=rand();
            ro2=rand();
        end
        oldConfig = [w,tau,ro1,ro2];
    end
end
```

APÊNDICE B – Códigos Fontes do GA

1) Função gaTrans **-** Algoritmo genético para melhoria de parâmetros de Transferência Radiativa em uma camada

```
function [sm] = gaTrans(parametros, quantidadePopulacaoInicial, percentual,
generations, percentualMutacao, variacaoMutacao, Y)
% Descrição: Função algoritmo genético para melhoria de parâmetros de
Transferencia Radiativa 1 Camada.
% Parâmetros da Função:
% parametros: Parâmetros a serem melhorados
% quantidadePopulacaoInicial: Quantidade da população do algoritmo genético
% percentual: Percentual de variação da população inicial
% generations: Quantidade de Gerações ou iterações
% percentualMutacao: Percentual da população que será realizada a mutação
% variacaoMutacao: Variação percentual da mutação
% Y - Intensidades medidas experimentalmente
% Retorna os parâmetros da melhor solução

   %Gera população inicial Baseado nos parametros
   populacao = geraPopupacaoInicial(parametros, quantidadePopulacaoInicial,
percentual, Y);

   parametros(5) = fitnessTrans(parametros(1:4),       Y);
   sm = parametros;

   for n=1: generations

      n
      %Fitness dos elementos e ordena pela valor fitness
      populacao = fitnessGeral(quantidadePopulacaoInicial, populacao, Y);

      %Verifica se o melhor elemento da população é melhor que a melhor solução
      if (populacao(1, 1:5) < sm(5))
        sm = populacao(1, :);
      end

      %Realiza o calculo para a roleta
      total = 0;
      for m=1:quantidadePopulacaoInicial
         populacao(m, 6) = populacao(quantidadePopulacaoInicial, 5) - populacao(m,
5) + populacao(1, 5);
         total = total + populacao(m, 6);
      end

      %Separa os melhores para próxima geração
      melhores = round(quantidadePopulacaoInicial * 0.1);
      if melhores == 0
         melhores = 1;
```

```
        end
        populacaoNova(1:melhores, 1:4) = populacao(1:melhores, 1:4);

        %Depois de preparar tudo vamos realizar o crossover
        for m=melhores:quantidadePopulacaoInicial
            filho = crossover(total, populacao);
            populacaoNova(m, 1:4) = filho(1:4);
        end

        %Realizar a mutação
        populacaoNova = mutacao(populacaoNova, percentualMutacao,
variacaoMutacao);

        populacao = populacaoNova;
    end

    % Avalia a ultima população
    populacao = fitnessGeral(quantidadePopulacaoInicial, populacao, Y);

    %Verifica se o melhor elemento da população é melhor que a melhor solução
    if (populacao(1, 1:5) < sm(5))
      sm = populacao(1, :);
    end

end
```

2) Função crossover **-** Função que seleciona com método roleta dois indivíduos e realiza o cruzamento entre eles

```
function [filho] = crossover(total, populacao)
% Descrição: Seleciona com método roleta 2 indivíduos e realiza o cruzamento entre
dois elementos
% Parâmetros da Função:
% total: Parâmetro utilizado para seleção dos indivíduos pelo método da roleta
% populacao:População que será selecionado os dois indivíduos

    %Seleciona o primeiro pai
    roleta1 = total*rand();
    soma=0;
    quantidade = size(populacao, 1);
    for n=1:quantidade
        soma = soma + populacao(n, 6);
        if soma > roleta1
            pai1 = populacao(n, 1:6);
            break
        end
    end
```

```
    %Seleciona o segundo pai
    roleta2 = total*rand();
    soma=0;
    for n=1:quantidade
        soma = soma + populacao(n, 6);
        if soma > roleta2
            pai2 = populacao(n, 1:6);
            break
        end
    end

    %Realiza o cruzamento
    for m=1:4
        filho(m) = (pai1(m) + pai2(m)) / 2;
    end

end
```

3) Função mutacao **-** Função seleciona x% dos indivíduos aleatoriamente e realiza a mutação

```
function [populacao] = mutacao(populacao, percentualMutacao, variacaoMutacao)
% Descrição: Seleciona x% dos indivíduos aleatoriamente e realiza a mutação.
% Parâmetros da Função:
% populacao: População a ser realizado a mutação
% percentualMutacao: Percentual da população que será realizada a mutação
% variacaoMutacao: Variação percentual da mutação
% Retorna a população que foi realizado a mutação

    % Calcula a quantidade de indivíduos que vão ser mutados
    quantidade = size(populacao, 1);
    quantidadeMutacao = round((quantidade * percentualMutacao) / 100);
    if (quantidadeMutacao == 0)
        quantidadeMutacao = 1;
    end

    % Loop para cada individuo que será mutado
    for n=1:quantidadeMutacao

        % Seleciona aleatório o individuo
        escolha = round(quantidade*rand());
        if (escolha == 0)
            escolha = 1;
        end

        % Para cada parâmetro é realizada uma pequena variação no valor
        for m=1:4
            percentual = variacaoMutacao*rand();
```

```
            positivo = round(rand());
            if (positivo == 1)
                populacao(escolha, m) = populacao(escolha, m) + (populacao(escolha, m)
* (percentual/100));
            else
                populacao(escolha, m) = populacao(escolha, m) - (populacao(escolha, m)
* (percentual/100));
            end
            if (populacao(escolha, m) > 1)
                populacao(escolha, m) = 1;
            end
            if (populacao(escolha, m) < 0)
                populacao(escolha, m) = 0;
            end
        end
    end
end
```

APÊNDICE C – Códigos Fontes do GRASP

1) Função grasp **-** Função GRASP para melhoria de parâmetros de Transferência Radiativa uma camada

```
function [sm] = grasp(parametros, qtdeInicial, LRC, iteracao, percentualInicial, iteracaoBL, percentualBL, YOriginal)
% Descrição: Função GRASP para melhoria de parâmetros de Transferencia Radiativa 1 Camada.
% Parâmetros da Função:
% parametros - Parametros a serem melhorados
% qtdeInicial - quantidade inicial da populacao inicial
% LRC - Numero máximo de itens que podem ser escolhido da lista.Este  parâmetro
% define quanto gulosa será, se 1 escolhe sempre o primeiro elemento.
% iteracao - Quantidade de iterações
% percentualInicial - Variação percentual da população inicial
% iteracaoBL - Numero de iterações da busca local
% percentualBL - Variação percentual da busca local
% YOriginal - Intensidades medidas experimentalmente
% Retorna os parâmetros da melhor solução

   % Considera os parametros passado como melhor solução
   parametros(5) = fitnessTrans(parametros(1:4), Y);
   sm = parametros;

   for n=1: iteracao
      n
      % Gera a população inicial
      populacaoInicial = geraPopulacaoInicial(parametros, qtdeInicial, percentualInicial);

      % Avalia a população inicial
      populacaoInicial = fitnessGeral(qtdeInicial, populacaoInicial, YOriginal);

      % Separa a lista restrita de candidatos
      populacaoLRC = populacaoInicial(1:LRC, 1:5)

      % Calcula seleciona um elemento aleatório entre os candidatos da lista
      aleatorio = round(rand() * LRC)
      if (aleatorio == 0)
         aleatorio = 1;
      end
      elementoCandidato = populacaoLRC(aleatorio, 1:5)

      % Realiza uma busca local neste candidato
      elementoCandidatoMelhorado = buscaLocal(elementoCandidato, iteracaoBL, percentualBL, YOriginal)

      % Este candidato é a base para a próxima iteração
```

```
        parametros = elementoCandidatoMelhorado;

        %Verifica se o melhor elemento da solução é melhor que a melhor solução
        if (elementoCandidatoMelhorado(1, 1:5) < sm(5))
          sm = elementoCandidatoMelhorado(1, :);
        end
    end
end
```

APÊNDICE D – Códigos Fontes do Busca Tabu

1) Função buscaTabu **-** Função Busca Tabu para melhoria de parâmetros de Transferência Radiativa uma camada

```
function [sm] = buscaTabu(elemento, iteracao, qtdeVizinhos, variacaoPercentual,
YOriginal, tamLT)
% Descrição Busca Tabu para melhoria de parâmetros de Transferencia Radiativa 1
Camada.
% Parâmetros da Função:
% elemento: Parâmetros a serem melhorados
% iteracao: Quantidade de iterações
% qtdeVizinhos: Quantidade de vizinhos
% variacaoPercentual: Variação percentual na geração dos vizinhos
% YOriginal: Intensidades medidas experimentalmente
% tamLT: Tamanho da lista tabu

   %avalia a solução inicial
   elemento(5) = fitnessTrans(elemento(1:4), YOriginal);
   sm = elemento;

   % Inicializa variáveis
   iLT = 1;
   listaTabu = 0;

   for i=1:iteracao
      i
      %Gera qtdeVizinhos vizinhos, variando X% os valores de w,?0, ?1 e ?2
      %Onde X=valor aleatório de 0 a variacaoPercentual% positivo e negativo
      V = geraVizinhanca (elemento, qtdeVizinhos, variacaoPercentual, listaTabu,
sm, YOriginal);

      %Classifica e ordena a vizinhança
      V = fitnessGeral(qtdeVizinhos, V, YOriginal);
      assignin('base', strcat('V', int2str(i)), V);

      %Seleciona o melhor vizinho
      v = V(1, :);
      v = buscaLocal(v, YOriginal, 5, 20);

      %Verifica se o melhor elemento da solução é melhor que a melhor solução até
agora
      if (v(5) < sm(5))
        sm = v
        assignin('base', strcat('sm ', int2str(i)), sm);
      end

      % Este elemento é a base para a próxima iteração
      elemento = v;
```

```
        % Inclui o elemento na lista tabu
        listaTabu(iLT, 1:5) = v(1:5);

        % Verifica se a lista esta cheia, se sim reinicializa
        iLT = iLT + 1;
        if (iLT == tamLT)
            iLT = 1;
        end
    end
end
```

2) Função geraVizinhanca **–** Gera vizinhança considerando aspiração e verificando se esta na lista tabu

```
function [V] = geraVizinhanca(parametros, qtdeVizinhos, variacaoPercentual,
listaTabu, sm, YOriginal)
%Descrição: Gera vizinhança considerando aspiração e verificando se esta na lista
tabu
% Parâmetros da Função:
% parametros: Parâmetros para a base da vizinhança
% qtdeVizinhos: Quantidade de vizinhos
% variacaoPercentual: Variação percentual na geração dos vizinhos
% listaTabu: Lista com os elementos a serem considerados tabus
% sm: Melhor solução até agora
% YOriginal: Intensidades medidas experimentalmente

    %Inicializa variáveis
    positivo = 0;
    menor = 0.995;
    maior=1.005;
    gerarVizinho=true;
    n = 1;

    % Enquanto não gerar os vizinhos suficientes fica no loop
    while gerarVizinho

        %Gera um vizinho variando o parametros com o percentual randômico
        for m=1:4
            percentual = variacaoPercentual*rand();
            positivo = round(rand());
            if (positivo == 1)
                valorGerado(m) = parametros(m) - ((parametros(m) * percentual) / 100);
            else
                valorGerado(m) = parametros(m) + ((parametros(m) * percentual) / 100);
            end
            if (valorGerado(m) > 1)
                valorGerado(m) = 1;
```

```
            end
            if (valorGerado(m) < 0.05)
                valorGerado(m) = 0.05;
            end
        end

        %Avalia o valor gerado
        valorGerado(5) = fitnessTrans(valorGerado(1:4), YOriginal);

        %Se o valor não satisfaz o critério de aspiração por objetivo
        encontrou = 0;
        if (valorGerado(5) >= sm(5))
            %Verifica se esta na lista tabu
            if listaTabu ~= 0
                for p=1:size(listaTabu, 1)
                    if (valorGerado(1) > (listaTabu(p, 1) * menor) && valorGerado(1) <
(listaTabu(p, 1) * maior) && ...
                        valorGerado(2) > (listaTabu(p, 2) * menor) && valorGerado(2) <
(listaTabu(p, 2) * maior) && ...
                        valorGerado(3) > (listaTabu(p, 3) * menor) && valorGerado(3) <
(listaTabu(p, 3) * maior) && ...
                        valorGerado(4) > (listaTabu(p, 4) * menor) && valorGerado(4) <
(listaTabu(p, 4) * maior))
                        encontrou = 1;
                    end
                end
            end
        end

        if (encontrou == 0)
            V(n, 1:5) = valorGerado(1:5);
            n = n + 1;
        end
        if (n == qtdeVizinhos + 1)
            gerarVizinho=false;
        end
    end

end
```

APÊNDICE E – Códigos Fontes Comuns

1) Função fitnessGeral **-** Função que realiza a avaliação da população e ordena pela valor fitness

```
function [populacao] = fitnessGeral(populacao, Y)
% Descrição: Realiza a avaliação da população e ordena pela valor fitness
% Parâmetros da Função:
% populacao - População a ser avaliada e ordenada
% Y - Intensidades medidas experimentalmente
 %Retorna a população avaliada

   %Realiza a avaliação dos elementos da população e armazena na posição 5 do
próprio elemento da população
   janela = waitbar(0,'Avaliação');
   quantidade = size(populacao,1);
   for m=1:quantidade
      waitbar(m/quantidade, janela, m);
      populacao(m, 5) = fitnessTrans(populacao(m, 1:4), Y);
   end
   close(janela)

   %Ordena de acordo com fitness
    populacao = ordenaLista(populacao);
    populacao(1:quantidade, 1:5)
    assignin('base', 'populacao', populacao);
    refreshdata(populacao, 'base')

end
```

2) Função fitnessTrans – Função de avaliação

```
function [Q] = fitnessTrans(Z, Y)
% Descrição: Função de avaliação do PCA
% Parâmetros da Função:
% Z: Este parâmetro é um array que possui: z=[w,tau,ro1,ro2]
% Y - Intensidades medidas experimentalmente
% Retorna a avaliação

   Nd=size(Y, 1);
   [Ytemp] = odms(Z(1),Z(2),Z(3),Z(4),Nd,0,0,0);

   Q = 0;
   for i=1: Nd
      Q = Q + (Ytemp(i) - Y(i)).^2;
   end

end
```

3) Função geraPopulacaoInicial – Função que gera população inicial, variando x% para mais e para menos aleatoriamente

```
function [populacaoInicial] = geraPopulacaoInicial(parametros,
quantidadePopulacaoInicial, variacaoPercentual)
% Descrição: Gera população inicial, variando x% para mais e para menos
aleatoriamente
% Parâmetros da Função
% parametros - Parametros que serão variados criando a população inicial
% quantidadePopulacaoInicial - Quantidade de elementos a serem criados
% variacaoPercentual- Variação que sofre os parametros para gerar a população
inicial

   % Loop pela quantidade necessária da população
   for n=1: quantidadePopulacaoInicial

      % Para cada parâmetro realiza a variação
      for m=1:4

         %Gera um valor aleatório de 0 a variação percentual
         percentual = (variacaoPercentual)*rand();

         %Aleatoriamente decide se a variação será para mais ou menos
         positivo = round(rand());

         % Realiza a variação
         if (positivo == 1)
            valorGerado(m) = parametros(m) - ((parametros(m) * percentual) / 100);
         else
            valorGerado(m) = parametros(m) + ((parametros(m) * percentual) / 100);
         end

         % Se o valor passar o limite considera o limite
         if (valorGerado(m) > 1)
            valorGerado(m) = 1;
         end
         if (valorGerado(m) < 0.05)
            valorGerado(m) = 0.05;
         end

      end
      populacaoInicial(n, 1:4) = valorGerado(1:4);
   end
end
```

4) Função buscalocal – Realiza uma pequena variação nos parâmetros em busca de um ótimo local

```
function [parametrosMelhor] = buscaLocal(parametros, iteracao, variacaoPercentual, YOriginal)
% Descrição: Realiza uma pequena variação nos parâmetros em busca de um ótimo local
% Parâmetros da Função:
% parametros: Parâmetros que serão realizados a busca local
% iteracao; Quantas iterações serão realizadas a busca do ótimo local
% variacaoPercentual: Variação percentual que o parametros sofre
% YOriginal - Intensidades medidas experimentalmente

   %Considera os parametros de entrada o melhor
   parametrosMelhor = parametros;
   janela = waitbar(0,'Avaliacao');

   % Loop de acordo com a iteração
   for n=1: iteracao
      waitbar(n/iteracao, janela);

      %Como são quatro parâmetros vamos variar todos os conjuntos de possibilidades
      for m=1:15
         mensagem=strcat(int2str(n),'-',int2str(m));
         waitbar(n/iteracao, janela, mensagem);
         mNovo = num2str(dec2bin(m));
         if (length(dec2bin(m)) == 1)
            mNovo = strcat('000', num2str(dec2bin(m)));
         end
         if (length(dec2bin(m)) == 2)
            mNovo = strcat('00', num2str(dec2bin(m)));
         end
         if (length(dec2bin(m)) == 3)
            mNovo = strcat('0', num2str(dec2bin(m)));
         end
         parametrosNovo = parametrosMelhor;
         parametrosNovo2 = parametrosMelhor;
         if (mNovo(1) == '1')
            percentual = (variacaoPercentual)*rand();
            parametrosNovo(1) = limitarValor(parametrosMelhor(1) + (parametrosMelhor(1) * percentual/100));
            parametrosNovo2(1) = limitarValor(parametrosMelhor(1) - (parametrosMelhor(1) * percentual/100));
         end
         if (mNovo(2) == '1')
            percentual = (variacaoPercentual)*rand();
            parametrosNovo(2) = limitarValor(parametrosMelhor(2) + (parametrosMelhor(2) * percentual/100));
```

```
            parametrosNovo2(2) = limitarValor(parametrosMelhor(2) -
(parametrosMelhor(2) * percentual/100));
        end
        if (mNovo(3) == '1')
            percentual = (variacaoPercentual)*rand();
            parametrosNovo(3) = limitarValor(parametrosMelhor(3) +
(parametrosMelhor(3) * percentual/100));
            parametrosNovo2(3) = limitarValor(parametrosMelhor(3) -
(parametrosMelhor(3) * percentual/100));
        end
        if (mNovo(4) == '1')
            percentual = (variacaoPercentual)*rand();
            parametrosNovo(4) = limitarValor(parametrosMelhor(4) +
(parametrosMelhor(4) * percentual/100));
            parametrosNovo2(4) = limitarValor(parametrosMelhor(4) -
(parametrosMelhor(4) * percentual/100));
        end
        parametrosNovo(5) = fitnessTrans(parametrosNovo, YOriginal);
        parametrosNovo2(5) = fitnessTrans(parametrosNovo2, YOriginal);
        if (parametrosNovo(5) < parametrosMelhor(5))
            parametrosMelhor = parametrosNovo;
        end
        if (parametrosNovo2(5) < parametrosMelhor(5))
            parametrosMelhor = parametrosNovo2;
        end
      end
   end
   close(janela)

end
```

APÊNDICE F – Fluxograma do Algoritmo Retropropagação