Evoluindo os pesos de uma Rede Neural Artificial uma abordagem Multiobjetivo

Andrei Strickler, Aurora Pozo

***Resumo*—Este artigo compara o treinamento de um Perceptron de Múltiplas Camadas usando Algoritmos Evolutivos Multiobjetivo (MOEA). Dois MOEAs foram usados: Speed constraint MultiObjective Particle Swarm Optimization (SMPSO) e o Algoritmo Evolutivo Multiobjetivo baseado em decomposição com Alocação de Recursos Dinâmicos (MOEA/D-DRA). Algumas combinações de valores para os parâmetros desses algoritmos foram aplicados para o problema de classificação. Para comparar os resultados dos algoritmos aplicou-se o Hypervolume como um indicador de qualidade, e a partir da média deste indicador foram realizados os testes estatísticos de Kruskal Willis para comparar os resultados obtidos.**

***Keywords*—Algoritmos Multi-objetivos, SMPSO, MOEA/D-DRA, Redes Neurais Artificiais, Problema de Classificação**

I. Introdução

As Redes Neurais Artificiais (RNA) são técnicas computacionais, que apresentam um modelo inspirado na estrutura neural de organismos inteligentes e que adquirem conhecimento através da experiência [1]. Dentre os diferentes tipos de RNA, a do tipo Perceptron de Múltiplas Camadas (MLP - *Multilayer Perceptron*) é a mais utilizada. Uma RNA do tipo MLP é constituída principalmente por um conjunto de unidades sensoriais (formando a camada inicial de neurônios), uma ou mais camadas ocultas e uma camada de saída de nós computacionais (neurônios). O sinal de entrada se propaga para frente através da rede, camada por camada [5]. Antes de ser efetivamente utilizada, uma MLP deve ser treinada, ou seja, ter seus pesos sinápticos ajustados para otimizar o processo de classificação, e para isso utilizam-se algoritmos de treinamento.

Em uma MLP normalmente se utiliza o algoritmo *backpropagation* para o treinamento dos pesos das conexões [9]. Embora a boa reputação, o *backpropagation* tem algumas desvantagens. Por usar o método do gradiente descendente, a desempenho do treinamento está associado diretamente com a superfície de erro do problema, que pode conter mínimos locais, fazendo com que o método não ache o mínimo global.

Por outro lado, os algoritmos de otimização global usam estratégias heurísticas para tentar escapar de mínimos locais [10]. Os Algoritmos Evolutivos (AE) podem ajudar a evitar o problema de convergência para mínimos locais e explorar técnicas de otimização global no treinamento de MLP.

As Redes Neurais Artificiais Evolutivas (RNAE) referem-se a uma classe especial de RNA, na qual a evolução é uma forma de adaptação em conjunto com o aprendizado. Podem ser consideradas como sistemas capazes de trocar suas arquiteturas e aprender regras [12].

Algoritmos Evolutivos, têm seu mecanismo inspirado no processo de evolução natural. Algumas ideias fundamentais da evolução, tais como recombinação e seleção são utilizadas na construção de algoritmos robustos e que requerem um mínimo de informações sobre o problema [3].

No grupo de algoritmos evolutivos pode-se apontar os Algoritmos Genéticos (GA - *Genetic Algorithms*), Evolução Diferencial (DE - *Differential Evolution*) [11], Otimização por nuvem de partículas (SMPSO - *Speed constrained Multi-objective Particle Swarm Optimization*) [6], e Algoritmo Evolutivo Multi objetivo baseado em Decomposição (MOEA/D - *Multiobjective Evolutionary Algorithm Based on Decomposition*) [13], e uma estratégia a qual utiliza alocação de recursos dinâmicos (*Dynamical Resource Allocation* - MOEA/D-DRA) [14] capazes de efetuar a tarefa de treinamento da RNA.

O algoritmo de Otimização por Enxame de Partículas multi-objetivo (SMPSO) é uma técnica de otimização baseada em enxames de partículas, deselvolvida por Kennedy e Eberhart (1995). Kennedy e Eberhart [6] tinham como objetivo simular o comportamento de bandos de aves na procura por comida.

No algoritmo SMPSO [8], utilizado no presente trabalho, cada partícula possui a capacidade de se recordar da melhor posição em que já esteve. Essa recordação é chamada de componente pessoal. Existe também a componente social, que é a capacidade de recordar a melhor posição onde o enxame já esteve. A junção destas duas componentes faz com que os pássaros do enxame procurem e encontrem alimento.

O algoritmo SMPSO enquadra-se nos algoritmos evolutivos. Esta classificação é justificada por Kennedy e Eberhart [6]. O SMPSO é dependente de processos estocásticos e o ajuste dos componentes melhor individual e melhor global assemelham-se ao operador de cruzamento, bem como o conceito de aptidão (*fitness*), utilizados nos algoritmos genéticos.

A utilização do algoritmo SMPSO para esse trabalho, dá-se pelo fato deste ser rápido e eficaz no trabalho de evolução de uma população em relação a outros algoritmos. E a utilização do algoritmo MOEA/D-DRA está relacionado ao fato deste ser um dos algoritmos recentes e com resultados promissores.

Multiobjective Evolutionary Algorithm based on Decomposition (MOEA/D) é um algoritmo evolutivo para otimização multi-objetivo usando a ideia de decomposição [13]. O MOEA/D decompõe o problema com muito objetivos em subproblemas de otimização escalar. Assim, o algoritmo resolve esses subproblemas simultaneamente pela evolução de uma população de soluções. Em cada geração, a população está composta pela melhor solução até então encontrada (desde o início da execução do algoritmo) para cada subproblema.

As relações entre os subproblemas são definidos com base na distâncias entre seus vetores de ponderação. Cada

subproblema é otimizado pelo MOEA/D usando informações a partir de seus subproblemas vizinhos [13].

O algoritmo MOEA/D-DRA [14] utiliza dos mesmos conceitos do MOEA/D [13] porém a estratégia de decomposição em subproblemas é modificada em relação a quantidade de memória computacional reservada para cada subproblema. Pois é identificado em cada subproblema a utilidade que este terá em relação aos demais, e em decorrência dessa utilidade é reservado o espaço de memória na execução.

Neste trabalho, serão empregados os algoritmos SMPSO [8] e MOEA/D-DRA [14] para o treinamento de MLPs para dois objetivos, sensibilidade e especificidade, através da evolução de uma população de indivíduos, na qual cada indivíduo representa uma possível solução ao problema de classificação.

Este artigo está estruturado da seguinte forma: a Seção II apresenta os conceitos básicos de RNA (subseção II-A), Algoritmos Evolutivos (subseção II-B), SMPSO (subseção II-B1), MOEA/D-DRA (subseção II-B2), Hipervolume (subseção II-C), e o problema de classificação (subseção II-D); na Seção III é exposto o método proposto utilizado neste trabalho; na Seção IV é descrita a configuração dos experimentos e descreve os resultados encontrados; e por fim, a Seção V contém a conclusão e trabalhos futuros.

II. Conceitos Básicos

Nesta seção iremos abordar cada um dos métodos estudados para a formação de uma MLP que terá realizado o trabalho de ajuste de pesos tanto pelo algoritmo SMPSO como pelo MOEA/D-DRA, ambos para os mesmos objetivos.

A. RNA

Segundo Haykin [5], os estudos sobre Redes Neurais têm sido motivados desde o começo pelo reconhecimento de que o cérebro humano processa informações de uma forma inteiramente diferente do computador digital convencional. O cérebro é um computador altamente complexo, não-linear e paralelo. Ele tem a capacidade de organizar seus constituintes estruturais, conhecidos por neurônios, de forma a realizar certos processamentos, como reconhecimento de padrões, percepção e controle motor, muito mais rapidamente que o mais rápido computador digital existente.

Uma Rede Neural biológica tem dois componentes essenciais: os neurônios e suas ligações (sinapses). Para que isso funcione, sinais são captados através das inúmeras partes sensíveis do corpo humano e enviados para o sistema nervoso central, que é composto por uma rede de neurônios interligados [5]. Na Figura 1 tem-se a ilustração de uma RNA do tipo MLP.

As RNAs, inspiradas nas Redes Neurais biológicas, buscam retratar um modelo inspirado na estrutura neural de organismos inteligentes e, através de métodos computacionais desenvolvidos, adquirir conhecimento de acordo com as experiências. Sendo característica mais importante de uma RNA é a forma como ocorre o seu processo de aprendizado. Para Haykin, a aprendizagem é definida por um processo pelo qual os parâmetros livres de uma rede neural são adaptados através de um processo de estimulação pelo ambiente em que está inserida. Para isso, existem outros modelos de aprendizado como: correção de erros, baseado em memória, Hebbiana.

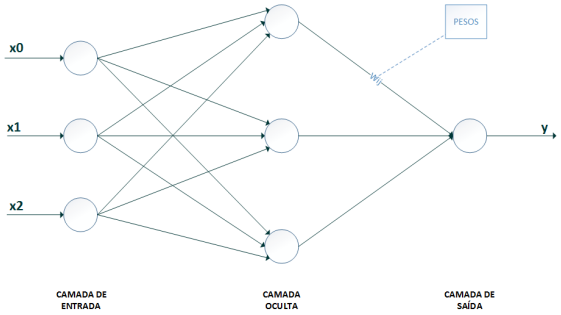

Figura 1. Representação de uma RNA - MLP

As redes MLP têm sido aplicadas com sucesso em uma variedade de áreas, desempenhando tarefas como classificação e reconhecimento de padrões, e controle e processamento de sinais [7]. Uma MLP é constituída por um conjunto de neurônios, os quais formam a camada de entrada da rede, uma ou mais camadas ocultas, e uma camada de saída. Essas camadas são interligadas por conexões, essas por sua vez possuem um peso sináptico como atributo. O peso de cada conexão, é utilizado em funções para atribuir um valor de entrada aos neurônio adjacente ao de origem dessa conexão.

As RNAs Evolutivas (RNAEs) referem-se a uma classe de RNAs na qual a evolução (dos AE) é uma forma de adaptação em conjunto com o aprendizado. Os AEs podem ser usados para executar várias tarefas, tais como treinamento de pesos de conexão, adaptação de regras de aprendizado, seleção de características de entrada.

B. AE

Para [12], os AEs podem ser utilizados na evolução global, para encontrar um conjunto de pesos de conexões ótimos (ou próximos), e sem cálculo do gradiente. O valor do erro, pode ser definido com base nas necessidades específicas da tarefa a ser executada. Um fator muito utilizado na formulação da função de erro é a diferença, chamada de erro, entre a saída desejada e a saída atual.

Diversos algoritmos evolucionários podem ser utilizados na evolução de pesos de uma MLP, assim como AG e DE, SMPSO e MOEA/D-DRA estão inclusos nesses.

1) SMPSO: O algoritmo 1 (SMPSO) possui três conceitos para a atualização das variáveis, de posição e velocidade, sendo essas as variáveis que deslocam as partículas pelo espaço de busca. O primeiro conceito é o da inércia, o qual nos assegura que tal partícula irá seguir a mesma direção e com a mesma velocidade para a próxima iteração. O segundo conceito é o do conhecimento pessoal, o qual nos garante que na próxima iteração a velocidade terá o incentivo do conhecimento adquirido, a melhor posição em que esta partícula já esteve, ao longo das iterações passadas. Por fim, o terceiro conceito, que se diz respeito ao conhecimento social, o qual proporciona com que todas partículas sejam incentivadas a seguir para a melhor posição já encontrada por toda a população ao longo das iterações passadas.

Esses conceitos, estão fortemente relacionados com dois temas fundamentais em AEs, a diversificação e a intensificação da população no espaço de busca. Tal que o conceito da inércia, age sobre a diversificação, pois a mesma direção e velocidade,

faria com que esta partícula buscasse por uma região, do espaço de busca, ainda não descoberta pela população. E os conceitos, pessoal e social, agem sobre a intensificação da busca pela melhor solução no espaço.

Nas Equações 1 e 2, são apresentados os cálculos realizados para a atualização das variáveis de velocidade e posição, respectivamente, de cada partícula. Na 1 são exigidas variáveis de velocidade v_i^t (termo de inércia), melhor posição pessoal $pBest_i^t$ (termo pessoal) e melhor posição global $gBest_i^t$ (termo global). Para a próxima geração, é necessário guardar a melhor posição no espaço de busca em que essa partícula já esteve (pBest - conceito pessoal), e o outro fator, é retenção da melhor posição global, dentre todas as partículas (gBest - conceito social).

Esses termos são utilizados no cálculo para obtenção da diferença entre esses termos (pessoal e social) com a posição atual da partícula. As diferenças são multiplicadas por constantes que influenciam em qual será a trajetória da população, e por fim, somadas ao conceito da inércia, (velocidade atual).

$$v_i^{t+1} = \overbrace{\omega \cdot v_i^t}^{inercia} + \overbrace{c_1 \cdot r_1^t(pBest_i^t - p_i^t)}^{pessoal} + \overbrace{c_2 \cdot r_2^t(gBest_i^t - p_i^t)}^{social} \quad (1)$$

$$p_i^{t+1} = p_i^t + v_i^{t+1} \quad (2)$$

Dessa forma, cada partícula do algoritmo SMPSO [8] representa uma potencial solução para o problema. Cada partícula é composta basicamente por: um vetor que representa sua localização no espaço de pesquisa (x); um vetor que representa sua velocidade (v); a aptidão da posição onde a partícula se encontra (p); um vetor que representa a melhor posição onde a partícula já esteve (p_b); um vetor que representa a melhor posição onde o enxame já esteve (g_b).

Em cada partícula presente na população, é efetuado o cálculo da velocidade e posição. Posteriormente é realizada a mutação polinomial [3] sobre cada partícula, e por fim é avaliada. A cada geração, são conhecidas e armazenadas as partículas com melhor *fitness* no arquivo de líderes.

Algoritmo 1: Pseudocódigo do algoritmo SMPSO

```
Entrada: Tamanho do enxame;
Saída: "arquivoLíderes";
enxame ← inicializaAleatoriamente();
gen = 0;
while gen < máximo de gerações do
    ComputeVelocidade(); // Eq. 1
    AtualizaPosição(); // Eq. 2
    MutacaoPolinomial(); // Perturbação
    Avaliação(); // Fitness obtido
    arquivoLíderes ← AtualizaArquivoDeLíderes();
    AtualizaMemóriaDasPartículas(); // pBest e gBest
    gen++;
end
return arquivoLíderest;
```

2) MOEA/D-DRA: A decomposição é outra maneira de resolver um problema com muitos objetivos. O MOEA/D decompõe o problema de otimização multi-objetivo (MOP), para muitos problemas mono objetivo, construídos de forma que a solução ótima para cada um desses subproblemas é um ponto na fronteira de Pareto ótima do MOP original. Em seguida, usa-se um algoritmo de otimização mono objetivo para resolver simultaneamente os subproblemas [13].

Existem dois componentes principais de uma MOEA/D. O primeiro é o mecanismo para decompor o MOP em subproblemas. Tradicionalmente vetores de ponderação são gerados aleatoriamente e de cada um deles o seu objetivo é um único problema. Utilizando-se soma ponderada, como a abordagem de *Tchebycheff*.

O segundo elemento importante é a forma de resolver os subproblemas de forma individual, obtidos da decomposição do problema original. A abordagem original do MOEA/D usa pontos de cruzamento e mutação de Algoritmos Genéticos (AG) [13], enquanto outras abordagens usam diferentes operadores, por exemplo, da Evolução Diferencial (Differential Evolution - DE), sendo esse utilizado no MOEA/D-DRA [14].

As relações de vizinhança entre os objetivos dos subproblemas são definidas com base nas distâncias entre seus vetores de ponderação. Cada subproblema é otimizado por meio de informações dos seus subproblemas vizinhos. Os subproblemas, no entanto, podem ter diferentes gastos computacionais, por conseguinte, é muito difícil atribuir um limite de gasto computacional para problemas diferentes. No algoritmo 2 (MOEA/D-DRA) [14], são definidos e calculados valores de utilidade π^i para cada subproblema i. Assim, ocorre a distribuição de limite de gasto computacional (processamento de dados) que cada subproblema irá consumir, tendo como base o valor estimado de utilidade. O vetor de ponderação λ^i é definido por um conjunto de vetores de ponderação através dos mais próximos dentre os vários existentes $\lambda_1, ..., \lambda_N$. O ponto ideal ($z^*$) é inicializado como o valor mínimo de cada objetivo encontrado na população inicial. Importante salientar que o algoritmo MOEA/D-DRA utilizado no presente trabalho utiliza a função de decomposição *Tchebycheff* [14]. Usando essa função de decomposição, cada problema obtém o formato apresentado na Equação 3.

$$Min\ g^{te}(x \mid \lambda, z^*) = max_{1 \leq j \leq M}\ \{\lambda_j \mid f_j(x) - z_j^* \mid\} \quad (3)$$
$$subject\ to\ x\ \in\ \Omega$$

Tal que g^{te} é a função *Tchebycheff*, $f(x) = (f_1(x), ..., f_M(x))$ é o conjunto de funções que devem ser minimizados, e $\lambda = (\lambda_1, ..., \lambda_M)$ é o vetor de pesos.

Em cada geração g, o algoritmo MOEA/D-DRA mantém: uma população de N pontos $x^1, ..., x^N$, onde x^i é a solução atual para o *i-ésimo* subproblema; $\pi^1, ..., \pi^N$, onde π^i é a estimativa de gasto em processamento que o subproblema i leva para ser processado.

Sendo I um conjunto de indivíduos selecionados através de um torneio com 10 indivíduos e um conjunto de $N/5M$ indivíduos é selecionado baseado na utilidade π^i, . Para cada indivíduo $i \in I$ será gerado um indivíduo $filho$ através dos operadores de cruzamento (*crossover*) e mutação.

A mutação polinomial [3], gera um $filho'$ a partir do indivíduo $filho$ formado pelo cruzamento, ou *crossover*. Neste trabalho foi utilizado o processo de *crossover* DE/rand/1/bin. No qual um indivíduo aleatório é selecionado como vetor alvo (x) e existe 1 par de indivíduos, também aleatoriamente

Algoritmo 2: Pseudocódigo do algoritmo MOEA/D-DRA

Entrada: tam. da populacao (N); nº de objetivos (M)
λ^i = geraVetoresDePeso(N);
$\lambda^i = (\lambda^i_1, ..., \lambda^i_M); i = 1, ..., N$
For $i = 1, ..., N$, definir o conjunto de índices vizinhos $B^i = \{i_1, ..., i_C\}$, onde $\{\lambda^{i_1}, ..., \lambda^{i_C}\}$ são C vetores de pesos mais próximos a λ^i (Distância euclidiana)
pop ← inicializaAleatoriamente();
Avaliar cada indivíduo $i \in$ pop e associar ao ser vetor λ^i;
Inicialize $z^* = (z^*_1, ..., z^*_M)$; $z^*_j = min_{1 \leq i \leq N} f_j(x^i)$
g = 1;
repita
  I = Selecionar usando 10-tournament utilizando (π^i);
  para cada *Individuo* $i \in I$ **faça**
    if $rand < \delta$ **then**
      escopo = B^i;
    else
      escopo = $\{1, ..., N\}$;
    end
    $filho$ = Crossover(DE/Rand/1/bin, i);
    $filho^{'}$ = MutacaoPolinomial($filho$);
    avaliar($filho^{'}$);
    atualiza z^*; $z^*_j = min(z^*_j, f_j(filho^{'}))$
    para cada *subproblema k (k = rand(escopo))* **faça**
      if *fitness*($filho$) < *fitness*(k) **then**
        substituir k por $filho$;
      end
    fim
    g++;
  fim
  computaUtilidade();
até $g >$ *maximo de avaliacoes*;

selecionados para calcular a mutação diferencial e por fim, o cruzamento binomial é utilizado (probabilidade de escolha de um componente do vetor pai ou do vetor gerado é dado pela distribuição binomial).

O próximo passo é a avaliação do indivíduo $filho$ em relação aos M objetivos e então é verificado se este irá substituir algum dos k indivíduos que pertencem a vizinhança de i, na próxima geração. Por fim, é atualizado o valor de utilidade π^i de cada subproblema i. Isto ocorrerá até que um número máximo de avaliações seja atingido.

C. Hipervolume

A comparação de desempenho de um ou vários métodos de otimização multi-objetivo é uma tarefa complexa. Duas metas da otimização multi-objetivo são: convergência e diversidade das soluções encontradas. Uma métrica bastante usada na avaliação de algoritmos multi-objetivos é o indicador de Hipervolume (HV). No HV, calcula-se o volume da região coberta entre os pontos das soluções na fronteira de Pareto P (soluções não-dominadas) e um ponto de referência W. Para cada solução $i \in P$, constitui-se um hipercubo, v_i com referência a um ponto W [15]. Este ponto de referência pode ser encontrado construindo um vetor com os piores valores da função objetivo. A união de todos os hipercubos encontrados é o resultado da métrica sendo que quanto maior o valor do HV, melhor. Um alto valor de HV indica que houve um elevado espalhamento entre as soluções de P e indica que houve também uma melhor convergência.

D. Problema de Classificação

A tarefa de classificação trata-se de separar conjuntos distintos de objetos, portanto identificar os objetos e distribuí-los em grupos previamente definidos. Os procedimentos de classificação estabelecem regras bem definidas, que podem ser usadas para designar os objetos seus respectivos grupos. Uma RNA que utiliza a abordagem de aprendizado supervisionado pode aprender a classificação de um determinado conjunto de dados, sendo que cada dado de entrada já está classificado, a RNA poderá aprender as regras existentes para cada grupo.

As RNAs podem ser utilizadas em problemas de classificação, pois possuem a capacidade de aprender e classificar os dados de entrada em grupos. Dessa forma, compreende-se que enquanto a execução avança, entradas semelhantes reforçarão cada vez mais uma certa configuração de neurônios. E para entradas diferentes, se aplicadas desde o início do processo de aprendizado provocarão neurônios diferentes. Portanto, pode-se dizer que para cada tipo de entrada se estabelecerá uma configuração de neurônios excitados. Entradas semelhantes provocarão configurações semelhantes, e estas configurações que irão permitir classificar as entradas.

III. Método Proposto

Nesta seção será explicado o método proposto, no qual uma RNA do tipo MLP terá seus pesos evoluídos através dos algoritmos SMPSO e MOEA/D-DRA. A linguagem Java foi utilizada para implementar a RNA e implementar as instâncias de treinamento e teste da RNA, nos algoritmos SMPSO e MOEA/D-DRA executados a partir do framework JMetal.

A RNA foi codificada como um vetor de números reais. Desta forma, é possível determinar rapidamente quais são os pesos de entrada e saída de cada neurônio. O número de camadas da rede é fixo e consiste em uma camada de entrada, uma camada oculta e uma camada de saída. Deve-se ressaltar que esta MLP tem um valor extra, chamado de *bias* para cada neurônio da rede, com exceção da camada de entrada.

A camada de entrada recebe os valores de entrada de cada dado advindo da base. Cada neurônio das camadas, oculta e de saída, possuem apenas uma saída e várias entradas, conforme o número de neurônios na camada anterior [5].

Neste trabalho, o valor de saída de um neurônio é calculado através do somatório da multiplicação entre os seus respectivos valores de entrada e pesos de suas conexões, sendo que y_i é a saída do neurônio *i* , x_j é a j-ésima entrada para o neurônio, e w_{ij} é o peso da ligação entre os neurônios *i* e *j*, podendo ser visualizada na Equação 4 o resultante. Normalmente, *fi* é uma função não-linear: função degrau, sigmoide ou logística [2].

$$y_i = f_i \left(\sum_{j=1}^{n} (w_{ij} \cdot x_j) + bias \right) \quad (4)$$

Para a função de ativação *fi* , existem algumas formas que são mais utilizadas quando o ajuste de pesos para a MLP é

feito por AE, tais como a função degrau e a função sigmoide [2]. Para este trabalho, foi determinada a função sigmoide, que pode ser visualizada na Equação 5, tal que *out* é o resultado da função de ativação que assume o valor entre [0:1] e repassa este valor para y_i da Equação 4, sendo este a saída de cada neurônio. O resultado do somatório efetuado na Equação 4, é indicado como *net* na Equação 5.

$$out = \frac{1}{1 + e^{-net}} \tag{5}$$

Esse encadeamento de cálculos se sucederá até a camada final da rede, quando será comparado o valor de saída de cada neurônio da camada de saída ao de um valor já esperado, advindo dos valores de entrada. Por exemplo: para um dado de entrada a saída esperada é "negative", e um neurônio da camada de saída classificou como "positive", esse indivíduo (conjunto de pesos da RNA), sofrerá uma penalização em seu *fitness*, acrescentando a cada momento em que qualquer neurônio classifique um dado de entrada, diferentemente do resultado esperado.

Ao fim das execuções, obtemos o cálculo do *fitness* para cada objetivo, e essa dupla de valores é a resposta que os algoritmos SMPSO e MOEA/D-DRA esperam, para então poder dar continuidade ao seu trabalho de evoluir o conjunto de pesos. A partir disso, os algoritmos evolutivos aplicados nesse trabalho estão encarregados de gerar novos conjuntos de pesos, que maximizem os *fitness* nas iterações futuras.

Na Tabela I, tem-se o número de neurônios de cada camada para cada base de dados a ser testada. As bases de dados utilizadas foram retiradas de Frank & Asuncion [4]. Para o experimento foram utilizadas as seguintes bases:

1) Breast Cancer Wisconsin (Original) Data Set (Câncer);
2) Pima Indians Diabetes Data Set (Diabetes);
3) Glass Identification Data Set (Glass);
4) Statlog (Heart) Data Set (Heart).

Tabela I. NÚMERO DE NEURÔNIOS DE CADA CAMADA

Base	Atributos(Entrada)	Classes(Saída)	Ocultos
Câncer	9	2	5
Diabetes	8	2	10
Glass	9	7	10
Heart	13	2	5

O *fitness* de cada indivíduo (conjunto de pesos) é computado para cada objetivo. Tal que o primeiro é dado pelo conceito de sensibilidade daquele indivíduo em classificar corretamente os dados de entrada do tipo *"Positive"*, ou seja, um indivíduo mais apto é aquele que possui o maior nível de sensibilidade, devido a este objetivo ser da classe de maximização. Para o segundo objetivo, é computado o nível de especificidade dos indivíduos, tal que esse classifique corretamente os dados de entrada do tipo *"Negative"*.

Pode-se notar nas Equações 6 e 7, as variáveis utilizadas como objetivos nesse trabalho, sensibilidade e especificidade, respectivamente. Tal que TP - *True Positive* - é o número de entradas positivas corretamente classificadas pela RNA; FN - *False Negative* - expressa o número de entradas negativas e classificadas como positivas; TN - *True Negative* - contabiliza o número de entradas negativas corretamente classificadas pela RNA; e FP - *False Positive* - é o número de entradas positivas classificadas como negativas pela RNA.

$$SEN = \left(\frac{TP}{TP + FN}\right) \tag{6}$$

Esses métodos tem como resultado um valor real entre [0:1], para ambos. Desse modo, esses métodos foram aplicados de forma que ocorra a maximização para ambos os objetivos. Tendo como ponto de máximo global o ponto [1.0,1.0].

$$SPE = \left(\frac{TN}{TN + FP}\right) \tag{7}$$

Desta forma, pode-se medir o quão boa é uma solução para o problema. Essa informação é fundamental para conduzir a evolução dos pesos da MLP através das iterações dos algoritmos SMPSO e MOEA/D-DRA. E por fim, obter os indivíduos que não são dominados para esse problema com a determinada configuração.

Para a análise dos resultados foi utilizado a média da taxa do *fitness* de cada objetivo, com 30 execuções para cada combinação de parâmetros. Os conjuntos de dados foram separados de acordo com a Tabela II.

Tabela II. SEPARAÇÃO DOS CONJUNTOS DE DADOS

Base	Treinamento	Teste	Total
Câncer	500	183	683
Pima Indians Diabetes	650	118	768
Glass Identification	170	44	214
Heart	220	50	270

Para este trabalho, foi utilizada a MLP com treinamento realizado pelos algoritmos: SMPSO e MOEA/D-DRA. Foi formulado um processo de treinamento com a evolução dos pesos das conexões pelos dois algoritmos, e ao fim realizou-se testes com entradas ainda não executadas pela MLP a fim de avaliar os algoritmos e combinações de parâmetros.

No término do processo evolutivo foi obtida a aproximação da fronteira de Pareto provida de cada algoritmo e avaliada no conjunto de testes.

IV. RESULTADOS

Para cada base de dados, foram geradas saídas em um padrão de arquivamento de acordo com os parâmetros variados, para o treinamento e teste da rede.

Foram realizados 30 execuções para cada conjunto de teste, sendo que cada conjunto de teste é determinado pela combinação de: (tamanho da população, número de avaliações), alterando valores nos parâmetros dos algoritmos, como tamanho da população: [50, 100], o número de iterações: [500, 1000]. Gerando as configurações a seguir: C1 com população de 50 indivíduos e máximo de iterações igual a 500; C2 com população de 50 indivíduos e máximo de iterações igual a 1000; C3 com população de 100 indivíduos e máximo de iterações igual a 500; C4 com população de 100 indivíduos e máximo de iterações igual a 1000.

Descritos na Tabela III, são apresentados os resultados da reprodução das instâncias de execuções descritas logo acima. Para efeito de comparação dos resultados entre os algoritmos executados. Na Tabela IV são expostos os erros e acertos das RNAs de melhor configuração, demarcadas em negrito na Tabela III, as quais obtiveram maior hipervolume em relação as outras configurações na mesma base e algoritmo.

Tabela III. RESULTADOS DE HIPERVOLUME EM CADA CONFIGURAÇÃO

Algoritmo	Base	Média HV C1 (D. Padrão)	Média HV C2 (D. Padrão)	Média HV C3 (D. Padrão)	Média HV C4 (D. Padrão)
SMPSO	Câncer	**0.99889** (0.002572)	0.99586 (0.008267)	0.99500 (0.013317)	0.99802 (0.005244)
	Diabetes	0.81054 (0.075324)	0.85761 (0.038138)	0.85706 (0.039694)	**0.85878** (0.050186)
	Glass	0.99573 (0.001395)	0.99035 (0.003285)	**0.99681** (5.8690E-4)	0.99672 (0.001294)
	Heart	0.64645 (0.037708)	0.64999 (0.032952)	0.66789 (0.032417)	**0.67879** (0.027469)
MOEA/D-DRA	Câncer	0.94478 (0.045548)	**0.97901** (0.031272)	0.97803 (0.032959)	0.94298 (0.041491)
	Diabetes	0.50173 (0.027114)	**0.62406** (0.041031)	0.60731 (0.038475)	0.50856 (0.037513)
	Glass	0.83237 (0.002012)	0.98992 (0.035803)	0.99217 (0.026678)	**0.99849** (0.002765)
	Heart	0.65033 (0.074096)	**0.72171** (0.073602)	0.71988 (0.069996)	0.69071 (0.079847)

Tabela IV. ACERTOS E ERROS NA FASE DE TESTE

Algoritmo	Base	TP	FN	TN	FP
SMPSO	Câncer	81	3	280	2
	Diabetes	98	48	46	44
	Glass	7	3	70	8
	Heart	38	10	26	26
MOEA/D-DRA	Câncer	81	3	279	3
	Diabetes	82	64	55	35
	Glass	8	2	70	8
	Heart	23	25	44	8

Analisando a Tabela III com as médias de hipervolume de cada configuração, nota-se que o desempenho das RNAs que o algoritmo SMPSO obteve para as bases Câncer e Diabetes foi superior ao MOEA/D-DRA. Em contra-partida, as bases Glass e Heart obtiveram melhor desempenho com o algoritmo MOEA/D-DRA evoluindo as RNAs.

Por fim, foram executados os resultados estatísticos Kruskal Wallis a partir dos Hypervolumes encontrados em cada uma das 30 execuções de cada configuração, e chegou-se a conclusão que os algoritmos contém diferença estatística nas bases: Câncer, Diabetes e Heart. Na base Glass o resultado mostrou-se com igualdade estatística entre os dois algoritmos.

Para o caso em que houve diferença estatística, o algoritmo SMPSO foi melhor para as bases Câncer e Diabetes. E o algoritmo MOEA/D-DRA foi melhor para a base Heart. Portanto, não se pode afirmar que um algoritmo é melhor que o outro em relação ao conjunto das quatro bases testadas.

V. CONCLUSÃO

Baseado na análise dos resultados obtidos, pode-se notar que, os algoritmos fornecem bons resultados para algumas bases, enquanto em outras as soluções encontradas não foram boas. Um fator que aumenta as taxas de acerto da rede é dado pelos casos de entradas de treinamento e teste, podendo fazer com que uma RNA classifique somente um padrão, deixando de lado o restante. Isso faz com que a RNA se especialize em um padrão, e acabe informando um valor de sensibilidade ou especificidade muito alto, porém não condiz com a realidade, quando esta deveria classificar os dois padrões.

Para esse trabalho, notou-se que os algoritmos SMPSO e MOEA/D-DRA podem ser utilizados para esse tipo de aplicação, porém neste caso, alguns resultados apontam que RNAs treinadas por algoritmos multi-objetivos pode acabar se especializando em apenas um dos objetivos, deixando de lado o outro. Para trabalhos futuros pode-se corrigir as falhas citadas, além de:

- Alteração da técnica de evolução, para comparação com os resultados obtidos, com as mesmas bases;
- Executar uma maior diversidade de valores para os parâmetros dos algoritmos, assim como, incluir mais bases aos testes, para comparação de resultados;

REFERÊNCIAS

[1] F. M. De Azevedo, L. M. Brasil, and R. C. L. de Oliveira. *Redes neurais com aplicações em controle e em sistemas especialistas*. Visual Books, 2000.

[2] A. C. P. L. F. de Carvalho, A. C. B. Delbem, R. A. F. Romero, E. V. Simões, and G. P. Telles. Computação Bioinspirada. *Biologia*, 2004.

[3] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley-Interscience Series in Systems and Optimization. John Wiley & Sons, Chichester, 2001.

[4] A. Frank and A. Asuncion. UCI machine learning repository. *URL http://archive.ics.uci.edu/ml*, 10, 2010.

[5] S. Haykin. *Redes neurais*. Bookman, 2001.

[6] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948 vol.4, Nov 1995.

[7] G. Luger. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving (5th Edition)*. Pearson Addison Wesley, 2004.

[8] A. J. Nebro, J.J. Durillo, J. García-Nieto, C.A. Coello Coello, F. Luna, and E. Alba. Smpso: A new pso-based metaheuristic for multi-objective optimization. In *2009 IEEE Symposium on Computational Intelligence in Multicriteria Decision-Making (MCDM 2009)*, pages 66–73. IEEE Press, 2009.

[9] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.

[10] Y. Shang and B. Wah. Global optimization for neural network training. *Computer*, 29(3):45–54, Mar 1996.

[11] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, pages 341–359, 1997.

[12] X. Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, Sep 1999.

[13] Q. Zhang and H. Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *Evolutionary Computation, IEEE Transactions on*, 11(6):712–731, 2007.

[14] Q. Zhang, W. Liu, and H. Li. The performance of a new version of MOEA/D on CEC09 unconstrained MOP test instances. *IEEE Congress on Evolutionary Computation*, 1:203–208, 2009.

[15] E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca, and V.G. da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *Evolutionary Computation, IEEE Transactions on*, 7(2):117–132, April 2003.