Teaching-Learning-Based Optimization no Treinamento de Redes Neurais Artificiais para Problemas de Classificação

Cristiano M. Garcia, Marcos D. Catalano, Eduardo A. Soares, Bruno H. G. Barbosa

Departamento de Engenharia - Universidade Federal de Lavras (UFLA)
cristianooo@gmail.com, diego.catalano@live.com, edu.soares999@gmail.com, brunohb@deg.ufla.br

Resumo Redes Neurais Artificiais (RNA) são técnicas comumente utilizadas em tarefas como predição, regressão e classificação. O treinamento destas redes costuma ser feito pelo algoritmo de retropropagação do erro, ou *backpropagation*. Diversos trabalhos na literatura sugerem a utilização de algoritmos de otimização baseados em populações, como o algoritmo PSO (*Particle Swarmn Optimization*), para o treinamento de RNAs. Em 2011, o algoritmo *Teaching-Learning Based Optimization* (TLBO) foi proposto, baseado em interações em sala de aula. O objetivo deste trabalho foi comparar o TLBO com outros algoritmos de otimização já consolidados na tarefa de treinar RNA para classificação. TLBO obteve resultados interessantes comparado a seus concorrentes utilizando as bases *Iris*[1], *Wine*[2] e *Ovarian Cancer*[3], podendo ser uma alternativa promissora no treinamento de RNAs.

1 Introdução

Redes Neurais Artificiais (RNA) são técnicas de aprendizado de máquina baseadas na disposição de células neurais de animais e são muito utilizadas em tarefas como reconhecimento de padrões, predição, regressão, entre outras, nas mais variadas áreas [1,2].

Os algoritmos de treinamento de RNAs baseados na retropropagação do erro, ou *backpropagation* [3], não garantem o encontro do mínimo global. Outras alternativas para treinamento são as metaheurísticas como o *Particle Swarm Optimization* (PSO) [4] e os

[1] http://archive.ics.uci.edu/ml/datasets/Iris
[2] http://archive.ics.uci.edu/ml/datasets/Wine
[3] https://home.ccr.cancer.gov/ncifdaproteomics/ppatterns.asp

Algoritmos Genéticos (GAs) [5], que podem convergir a um resultado aceitável com poucas iterações [6].

As metaheurísticas são procedimentos empregados na otimização em diversos campos de estudo, como engenharia, transporte e logística, para minimizar ou maximizar uma função ou modelo matemático através de experimentação de valores dentro de um conjunto factível do espaço de soluções. Muitas delas se baseiam em comportamentos encontrados na natureza e interações sociais.

Existem várias propostas na literatura no que tange o treinamento de RNAs com metaheurísticas. Por exemplo, em [7], o PSO é comparado com o *backpropagation* e conclui-se que o PSO é mais interessante que o *backpropagation* quando a aplicação necessita de aprendizado rápido. Em [8], foi apresentada uma comparação entre Evolução Diferencial (DE) e algoritmos baseados em *backpropagation*, como Levenberg-Marquadt e quasi-Newton, onde foi constatado que o DE possui desempenho comparável à tais algoritmos.

Além do PSO e DE, várias outras metaheurísticas vêm sendo propostas para os mais diversos fins, como os algoritmos *Artificial Bee Colony* (ABC) [9], o *Biogeography-based Optimization* (BBO) [10], o *Firefly Algorithm* [11], o *Teaching-Learning based Optimization* (TLBO) [12], entre outros.

Dentre esses algoritmos, pode-se destacar o TLBO, algoritmo de otimização baseado em interações professor-estudantes em sala de aula. Este algoritmo vem sendo empregado em diferentes áreas de pesquisa. Por exemplo, em [13,14] são utilizados métodos de otimização multi-objetivo baseado no TLBO. Em [15] é proposta uma abordagem *fuzzy* adaptativa utilizando TLBO para o problema de geração de testes *t-way*. Em [16] é utilizado um híbrido *fuzzy*-TLBO-PSO para selecionar genes associados ao câncer de mama.

Especificamente sobre treinamento de RNAs, o TLBO foi empregado tanto em problemas de regressão como classificação. Em [17], uma RNA é treinada com o TLBO pra realizar a tarefa de predição do consumo de energia elétrica na Turquia, em comparação com o algoritmo *backpropagation*, onde a RNA treinada com TLBO obteve melhores resultados. Em [18] foi treinada uma RNA com uma modificação do TLBO [19] para predição de chuvas no estado de Andhra Pradesh, na Índia. Os resultados da RNA treinada com a versão modificada do TLBO foram considerados melhores do que o da RNA

treinada com *backpropagation*. Em [20], é comparado o treinamento de RNA para estimativa da pressão mínima de miscibilidade de óleo de gás carbônico para recuperação de óleo melhorado. Foram utilizados os algoritmos *Cuckoo Optimization* e o TLBO. Como resultado, a RNA treinada com TLBO gerou um modelo mais robusto.

Isso posto, conclui-se que o TLBO possui resultados interessantes comparados com algoritmos de mesma natureza. Na literatura, há trabalhos que apresentem o desempenho de RNAs treinadas por meio do TLBO para problemas de reconhecimento de padrões [21,22]. Porém, poucos trabalhos comparam seus métodos com mais de uma heurística. Portanto, este trabalho tem como objetivo treinar RNAs para realizar tarefas de classificação com o algoritmo TLBO, comparando seus desempenhos com outras cinco metaheurísticas: *Artificial Bee Colony*, *Biogeography Based Optimization*, *Differential Evolution*, *Firefly Algorithm* e *Particle Swarm Optimization*. A comparação entre as diversas técnicas de otimização visam então, determinar aquelas mais adequadas para a tarefa de classificação em base de dados de origens diversas. As diferentes técnicas são comparadas em termos de acurácia.

2 Materiais e Métodos

Este trabalho tem como objetivo comparar o TLBO com outros algoritmos de otimização, na tarefa de treinamento de RNA para classificação. Nesta seção são apresentados o algoritmo TLBO, a modelagem das RNAs para estimação de seus pesos pelas metaheurísticas, os parâmetros dos algoritmos implementados para comparação com o TLBO, as bases de dados utilizadas e, por fim, o método de avaliação e comparação dos algoritmos.

2.1 O Algoritmo TLBO

O TLBO é um método baseado em interações sociais, dividido em duas fases: fase de aprendizado a partir do professor (*teacher phase*) e fase de aprendizado entre os estudantes (*learner phase*).

Em cada iteração, na fase *teacher*, é selecionado um professor (*teacher*), que é o indivíduo com melhor aptidão. Após essa escolha, os indivíduos (estudantes) "aprendem" com o professor, se movendo na

sua direção, por meio do fator de aprendizado (ou *teaching factor*), calculado de forma aleatória no intervalo de 1 a 2.

Depois, na fase *learner*, cada indivíduo é comparado a um outro escolhido aleatoriamente, e um novo indivíduo é criado de forma a se mover na direção do melhor [12]. O pseudocódigo do TLBO é apresentado no Algoritmo 1. Como pode ser observado, trata-se de um algoritmo muito simples, em que os únicos parâmetros a serem ajustados são o número de iterações e o tamanho da população de indivíduos. Esta característica, em conjunto com seu bom desempenho na busca por soluções, faz com que este algoritmo seja uma boa opção no treinamento de RNAs.

2.2 Representação das RNAs

Cada RNA é representada por um indivíduo dentro dos algoritmos de otimização. O indivíduo é, portanto, formado por um vetor de valores reais que representam todas as sinapses da RNA. Os indivíduos foram modelados da seguinte forma: as primeiras posições são para os pesos provenientes das entradas, as posições posteriores são destinadas aos *bias* dos neurônios da primeira camada oculta. As posições seguintes são os valores de pesos dos neurônios da camada oculta para a camada seguinte (outra camada oculta ou de saída) e assim sucessivamente, sendo que e as últimas posições são os valores de *bias* da camada de saída. Um exemplo pode ser visto na Figura 1.

A avaliação dos indivíduos da população é realizada a partir do desempenho de classificação da RNA formada, considerando os erros obtidos em dados de treinamento. Assim, o objetivo dos algoritmos de otimização é a minimização do erro percentual de classificação da RNA implementada com seus valores.

2.3 Configuração dos algoritmos

Os códigos utilizados de todos os algoritmos utilizados neste trabalho foram obtidos do website YarPiz[4] e adaptados. Para a comparação com o TLBO, foram selecionados os algoritmos mencionados na Seção 1, com seus respectivos parâmetros:

[4] http://www.yarpiz.com/

Algorithm 1 Teaching-Learning-based Optimization

Require: nDimensões ← número de dimensões do problema a ser endereçado
Require: funçãoCusto ← função que calcula o custo, que direcionará os indivíduos ao longo das iterações
Require: numeroDeIndividuos ← número de indivíduos da população a ser inicializada

```
1: inicializa população pop aleatoriamente e calcula os custos de cada indivíduo
2: MelhorSolução ← null
3: for i ← 1 : nIteracoes do
4:     media ← media das posições dos indivíduos;
5:     seleciona teacher() {melhor indivíduo}
6:
7:     {início da fase teacher}
8:     for j ← 1 : numeroDeIndividuos do
9:         cria novo indivíduo novoIndividuo
10:        TF ← round(1 + rand(0, 1)) {TF = fator de aprendizagem (teaching factor)}
11:        novoIndividuo.Posição ← pop(j).Posição + rand() * (teacher.Posição - (TF * media))
12:        novoIndividuo.Custo = funçãoCusto(novoIndividuo.Posição)
13:
14:        if novoIndividuo.Custo < pop(j).Custo then
15:            pop(j) = novoIndividuo;
16:            if pop(j).Custo < MelhorSolução.Custo then
17:                MelhorSolução = pop(j)
18:            end if
19:        end if
20:    end for
21:    {fim da fase teacher}
22:
23:    {início da fase learner}
24:    for j ← 1 : numeroDeIndividuos do
25:        k ← número entre 1 e numeroDeIndividuos diferente de j
26:        if pop(k).Custo < pop(j).Custo then
27:            passo ← pop(j).Posição - pop(k).Posição
28:        else
29:            passo ← pop(k).Posição - pop(j).Posição
30:        end if
31:
32:        cria novo indivíduo novoIndividuo
33:        novoIndividuo.Posição ← pop(j).Posição + rand() * passo
34:        novoIndividuo.Custo = funçãoCusto(novoIndividuo.Posição)
35:
36:        if novoIndividuo.Custo < pop(j).Custo then
37:            pop(j) = novoIndividuo;
38:            if pop(j).Custo < MelhorSolução.Custo then
39:                MelhorSolução = pop(j)
40:            end if
41:        end if
42:    end for
43:    {fim da fase learner}
44: end for
```

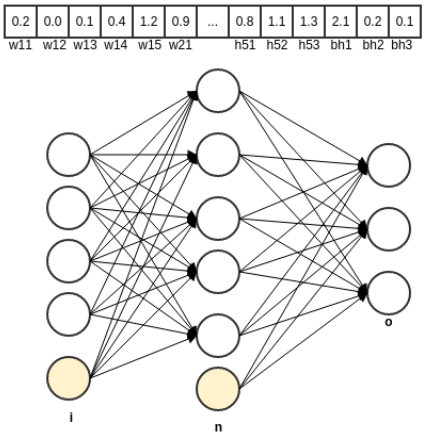

Figura 1. Representação de uma RNA por um vetor de números reais (indivíduo).

- *Particle Swarm Optimization* (PSO): c1 = 2 (coef. de aceleração do componente cognitivo); c2 = 2 (coef. de aceleração do componente social); w = 1 (coef. de inércia).
- *Biogeography-based Optimization* (BBO): kr = 0.2 (taxa da população a ser mantida de uma geração pra outra); pMutacao = 0.1 (taxa de mutação); sigma = 0.02 * (varmax - varmin) (componente da mutação); alpha = 0.9 (componente da migração).
- *Firefly Algorithm* (FA): gamma = 1 (coef. de absorção da luz); beta0 = 2 (coef. de atração); alpha = 0.2 (coef. de mutação); alphaDamp = 0.98 (taxa de amortecimento do coeficiente de mutação); delta = 0.05 * (varmax - varmin). (*range* de mutação uniforme).
- *Artificial Bee Colony* (ABC): L = (0.6 * nDimensões * tamanhoPopulação) (limite de abandono); a = 1 (coef. de aceleração).
- *Differential Evolution* (DE): pCR = 0.9 (probabilidade de *crossover*); beta = 0.5.

Todos os algoritmos, incluindo o TLBO, foram configurados com o mesmo tamanho de população (100 indivíduos), mesmo número de iterações (conforme apresentado na Tabela 1) e espaço de busca entre -500 e 500. A arquitetura da RNA foi definida com uma camada oculta, variando seu número de neurônios conforme apresentado na Tabela 1.

2.4 Bases de dados utilizadas

Foram utilizadas as bases de dados *Iris*[5], *Wine*[6] e *Ovarian Cancer*[7]. Os dados de entrada foram normalizados entre -1 e 1. Mais informações sobre as características das bases de dados e como elas afetam o problema podem ser encontradas na Tabela 1.

Para a base de dados *Iris*, têm-se 4 atributos (entradas) e 3 classes. Dessa forma, a otimização dos pesos da RNA com estas características se torna um problema de 43 dimensões (pesos ou valores a serem otimizados): 4 (atributos) $\times$ 5 (neurônios na camada escondida) + 5 (neurônios na camada escondida) $\times$ 3 (neurônios de saída) + 5 (pesos de *bias* na camada escondida) + 3 (pesos de *bias* na camada escondida).

A base de dados *Wine* possui 13 entradas e 3 classes, fazendo com que o problema de otimização tenha uma dimensão de 88 e 173 para RNAs constituídas de 5 ou 10 neurônios escondidos, respectivamente.

Por fim, foi escolhido um problema de dimensão maior, a base de dados *Ovarian Cancer*, com 1032 valores a serem otimizados.

Base de Dados	Nº de neurônios da camada escondida	Nº de exemplos	Nº de atributos	Nº de classes	Dimensão total
Iris	5	150	4	3	43
Wine	5	178	13	3	88
Wine	10	178	13	3	173
Ovarian cancer	10	216	100	2	1032

Tabela 1. Informações sobre as bases de dados e complexidade dos problemas.

2.5 Método de avaliação

Foram realizadas 50 execuções independentes para cada algoritmo, com 25 gerações para a base de dados *Iris* e 35 gerações para as bases de dados *Wine* e *Ovarian Cancer*, utilizando o procedimento

[5] http://archive.ics.uci.edu/ml/datasets/Iris
[6] http://archive.ics.uci.edu/ml/datasets/Wine
[7] https://home.ccr.cancer.gov/ncifdaproteomics/ppatterns.asp

de validação cruzada *holdout validation*, onde 60% dos dados são utilizados para treinamento, 20% para validação e 20% para teste.

Por se tratar de uma tarefa de classificação, a métrica utilizada para comparar os algoritmos foi a porcentagem de erro em dados de teste, onde foram calculados os valores médios, mínimos e máximos. Sobre os erros de testes, também foi realizado o teste de Tukey HSD (*Honestly Significant Difference*) [23], com 95% de confiança.

3 Resultados e Discussão

A Tabela 2 apresenta os erros mínimos, médios e máximos em porcentagem obtidos em dados de testes das RNAs nas 50 execuções independentes, com a configuração de 5 e 10 neurônios em uma única camada escondida, para todas as bases de dados estudadas. Os melhores valores estão destacados em negrito. A escolha por 5 e 10 neurônios foi para aumentar a dificuldade dos experimentos, reduzindo a chance de *overtraining*.

Datasets/Algoritmos		ABC	BBO	DE	FA	PSO	TLBO
Iris (5)	Mín.	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	Méd.	6.33	5.87	5.20	4.40	7.33	**2.20**
	Máx.	16.67	20.0	16.67	13.3	26.67	**10.0**
Wine (5)	Mín.	2.86	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	Méd.	12.8	9.20	5.89	5.43	9.77	**1.49**
	Máx.	25.71	20.0	14.3	14.3	34.29	**8.57**
Wine (10)	Mín.	2.86	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	Méd.	13.89	7.83	7.49	6.86	7.83	**2.91**
	Máx.	34.29	22.86	20.0	20.0	25.71	**11.43**
Ovarian Cancer (10)	Mín.	4.65	4.65	**2.33**	**2.33**	**2.33**	**2.33**
	Méd.	13.02	11.3	8.51	10.51	14.28	**6.74**
	Máx.	25.58	25.58	18.6	23.26	30.23	**13.95**

Tabela 2. Erros mínimos, médios e máximos para todas as bases de dados avaliadas. Valores entre parênteses representam número de neurônios na camada escondida.

Como pode ser observado nos resultados para a base *Iris*, o TLBO obteve os melhores erros médio, mínimo e máximo. Foi realizado um teste de Tukey para avaliar o desempenho do TLBO em relação aos outros algoritmos e o resultado pode ser visto na Figura 2, comprovando que o TLBO obteve resultados estatisticamente melhores

(95% de confiança) que os algoritmos BBO, PSO, DE e ABC, obtendo resultados similares ao FA para essa base.

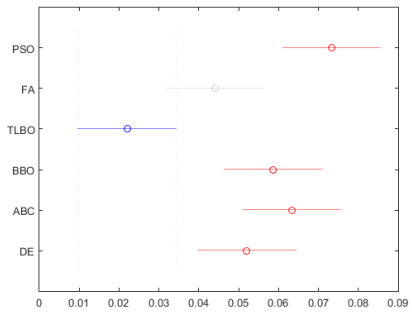

Figura 2. Teste de Tukey para a média dos erros de teste para a base de dados *Iris*.

Para a base de dados *Wine*, o TLBO novamente obteve as melhores médias, superando os outros algoritmos. Foi realizado outro teste de Tukey, que pode ser visualizado na Figura 3.

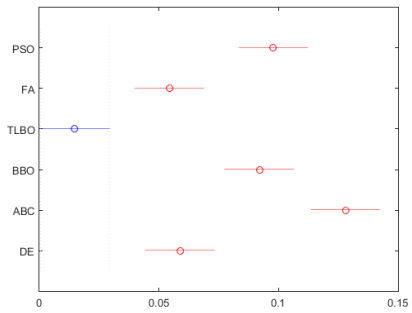

Figura 3. Teste de Tukey para a média dos erros de teste para a base de dados *Wine* com 5 neurônios.

Ainda para a base de dados *Wine*, o número de neurônios na camada escondida foi elevado pra 10. Pode-se inferir pela Tabela 2, que o TLBO obteve novamente as melhores médias. Na Figura 4, pode ser observado pelo Teste de Tukey que o desempenho do TLBO teve média significativamente menor que todos os outros algoritmos.

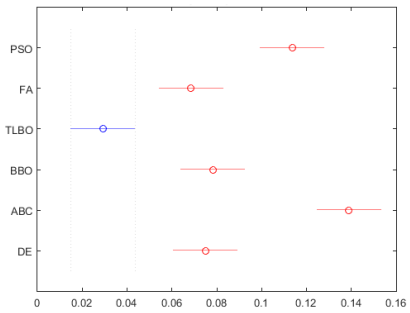

Figura 4. Teste de Tukey para a média dos erros de teste para a base de dados *Wine*.

Também foi realizada a comparação entre os algoritmos utilizando a base de dados *Ovarian Cancer*, com 10 neurônios na camada escondida. Pode-se observar pela Tabela 2 que quatro algoritmos atingiram o mesmo valor de erro mínimo pelo menos uma vez. O algoritmo TLBO novamente atingiu as melhores médias. Porém, no teste de Tukey para este problema (Figura 5), observa-se que o TLBO foi significativamente melhor que todos os outros algoritmos, com exceção do DE. É importante mencionar que TLBO obteve uma média melhor que o FA efetuando cerca de 1/3 da quantidade de avaliações que o FA precisou realizar, como pode ser visto na Figura 6. As medidas foram coletadas durante uma execução independente. O número de avaliações varia com o número de indivíduos da população e com o número de iterações do algoritmo. Como estes números foram fixados (população de 100 indivíduos e iterações iguais a 25 ou 35), o número de avaliações não varia de uma execução para outra.

4 Conclusão e Trabalhos Futuros

Neste trabalho, o objetivo foi comparar o algoritmo TLBO com os algoritmos *Particle Swarm Optimization*, *Biogeography-based Optimization*, *Firefly Algorithm*, *Differential Evolution* e *Artificial Bee Colony* para o treinamento de RNA na tarefa de classificação.

Os resultados apresentados na Seção 3 mostraram que o TLBO é uma alternativa promissora, tendo desempenhos médios superiores aos algoritmos PSO, DE, ABC, FA e BBO. Além disso o TLBO

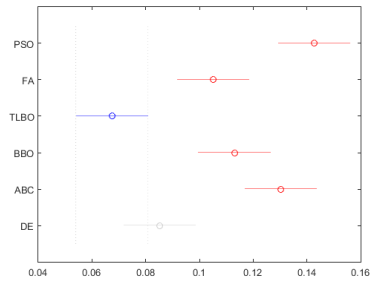

Figura 5. Teste de Tukey para a Base de Dados *Ovarian Cancer* com 10 neurônios na camada escondida.

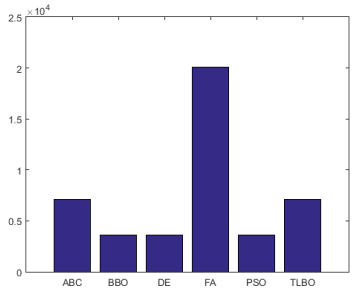

Figura 6. Avaliações por execução independente na base de dados *Ovarian Cancer*.

é de fácil implementação e exige a configuração de apenas dois parâmetros, o tamanho da população e o número de gerações.

Como trabalhos futuros, são sugeridas: a comparação de TLBO com outros algoritmos e modificações de algoritmos já presentes neste artigo; e a comparação com algoritmos híbridos ou meméticos.

Agradecimentos

Agradecimentos à FAPEMIG pelo apoio financeiro.

Referências

1. CW Dawson and RL Wilby. Hydrological modelling using artificial neural networks. *Progress in physical Geography*, 25(1):80–108, 2001.
2. Mayank Kumar Gautam and Vinod Kumar Giri. An approach of neural network for electrocardiogram classification. *APTIKOM Journal on Computer Science and Information Technologies*, 1(3):115–123, 2016.
3. David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.
4. J Kennedy and R Eberhart. Particle swarm optimization. In *Proc. IEEE International Conf. on Neural Networks*. IEEE, 1995.
5. J. H. Holland. *Adaptation in Natural and Artificial System*. University of Michigan Press, 1975.
6. Jing-Ru Zhang, Jun Zhang, Tat-Ming Lok, and Michael R Lyu. A hybrid particle swarm optimization–back-propagation algorithm for feedforward neural network training. *Applied mathematics and computation*, 185(2):1026–1037, 2007.
7. Venu G Gudise and Ganesh K Venayagamoorthy. Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. In *Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE*, pages 110–117. IEEE, 2003.

8. Jarmo Ilonen, Joni-Kristian Kamarainen, and Jouni Lampinen. Differential evolution training algorithm for feed-forward neural networks. *Neural Processing Letters*, 17(1):93–105, 2003.
9. Dervis Karaboga and Bahriye Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of global optimization*, 39(3):459–471, 2007.
10. Dan Simon. Biogeography-based optimization. *IEEE transactions on evolutionary computation*, 12(6):702–713, 2008.
11. Xin-She Yang. Firefly algorithm, stochastic test functions and design optimisation. *International Journal of Bio-Inspired Computation*, 2(2):78–84, 2010.
12. Ravipudi V Rao, Vimal J Savsani, and DP Vakharia. Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. *Computer-Aided Design*, 43(3):303–315, 2011.
13. Vivek K Patel and Vimal J Savsani. A multi-objective improved teaching–learning based optimization algorithm (mo-itlbo). *Information Sciences*, 357:182–200, 2016.
14. R Venkata Rao, Dhiraj P Rai, and Joze Balic. A multi-objective algorithm for optimization of modern machining processes. *Engineering Applications of Artificial Intelligence*, 61:103–125, 2017.
15. Kamal Z Zamli, Fakhrud Din, Salmi Baharom, and Bestoun S Ahmed. Fuzzy adaptive teaching learning-based optimization strategy for the problem of generating mixed strength t-way test suites. *Engineering Applications of Artificial Intelligence*, 59:35–50, 2017.
16. Saleh Shahbeig, Mohammad Sadegh Helfroush, and Akbar Rahideh. A fuzzy multi-objective hybrid tlbo–pso approach to select the associated genes with breast cancer. *Signal Processing*, 131:58–65, 2017.
17. Ergun Uzlu, Murat Kankal, Adem Akpınar, and Tayfun Dede. Estimates of energy consumption in turkey using neural networks with the teaching–learning-based optimization algorithm. *Energy*, 75:295–303, 2014.
18. B Kavitha Rani, K Srinivas, and A Govardhan. Rainfall prediction with tlbo optimized ann. 2014.
19. Suresh Chandra Satapathy and Anima Naik. Modified teaching–learning-based optimization algorithm for global numerical optimization—a comparative study. *Swarm and Evolutionary Computation*, 16:28–37, 2014.
20. Abouzar Choubineh, Seyede Robab Mousavi, Masih Vafaee Ayouri, Masoud Ahmadinia, Dariush Choubineh, and Alireza Baghban. Estimation of the co2-oil minimum miscibility pressure for enhanced oil recovery. *Petroleum Science and Technology*, 34(22):1847–1854, 2016.
21. Suresh Chandra Satapathy, Anima Naik, and K Parvathi. Teaching learning based optimization for neural networks learning enhancement. In *SEMCCO*, pages 761–769. Springer, 2012.
22. Amaresh Sahu and Sabyasachi Pattnaik. Evolving neuro structure using adaptive pso and modified tlbo for classification. *Procedia Computer Science*, 92:450–454, 2016.
23. John Neter, Michael H Kutner, Christopher J Nachtsheim, and William Wasserman. *Applied linear statistical models*, volume 4. Irwin Chicago, 1996.