PROJETO FASE II

Inteligência Computacional 2023/2024

Licenciatura em Engenharia Informática

Dinis Meireles de Sousa Falcão / a2020130403@isec.pt

Kevin Fernando Pereira Rodrigues / a2013010749@isec.pt

ÍNDICE

EM QUE CONSISTE A COMPUTAÇÃO SWARM?

A **Computação Swarm** referesse a um paradigma computacional inspirado no comportamento coletivo de organismos sociais, como enxames de insetos, cardumes de peixes, ou bandos de pássaros. Este paradigma procura modelar e simular o comportamento descentralizado e autónomo de entidades simples, chamadas de agentes, que interagem entre si e respondem ao ambiente de maneira coletiva. Algumas das suas principais características são:

- **Descentralização:** não há uma entidade central que coordena as ações de todos os agentes. Cada agente age de forma autônoma, respondendo às informações locais e interagindo com seus vizinhos;
- **Autonomia:** não há uma entidade controladora que dite diretamente o comportamento de cada agente;
- **Comportamento Emergente:** padrões complexos e comportamentos globais surgem naturalmente a partir do comportamento individual simples de cada agente;
- **Adaptação Dinâmica:** Os agentes podem ajustar o seu comportamento em tempo real com base em novas informações ou condições ambientais;
- **Robustez e Tolerância a Falhas:** se um agente falhar, outros podem continuar a realizar as tarefas necessárias.

Aplicações da **Computação Swarm** incluem otimização, controle de tráfego, busca e resgate, simulação social, inteligência artificial distribuída, entre outras. Algoritmos de otimização por enxame, como **Particle Swarm Otimization** (PSO) e **Ant Colony Otimization** (ACO), são exemplos comuns de técnicas baseadas nesse paradigma e que foram utilizadas nas aulas práticas de Inteligência Computacional.

COMO FUNCIONA O ALGORITMO SELECIONADO?

O **Artificial Bee Algorithm** (ABA) é uma meta-heurística baseada no comportamento das abelhas na natureza. Foi proposto como uma abordagem de otimização inspirada no comportamento de procura de recursos das abelhas. O algoritmo visa resolver problemas de otimização, encontrando soluções de alta qualidade para uma variedade de problemas.

O algoritmo mantém três tipos de abelhas virtuais, cada uma representando um papel específico no processo de busca:

- **Abelhas Empregadas (Employed Bees):** estas abelhas exploram regiões no espaço de busca e carregam informações sobre a qualidade das soluções encontradas;
- **Abelhas Observadoras (Onlooker Bees):** com base nas informações fornecidas pelas abelhas empregadas, as abelhas observadoras decidem quais regiões do espaço de busca devem ser exploradas. Elas escolhem as suas regiões de destino de acordo com a qualidade das soluções encontradas pelas abelhas empregadas;
- **Abelhas Scout (Scout Bees):** estas têm a tarefa de explorar aleatoriamente novas regiões do espaço de busca. Se uma abelha empregada ou observadora não encontrar uma solução melhor após um número específico de iterações, ela se torna uma abelha scout, reiniciando a exploração numa nova posição aleatória.

O **Artificial Bee Algorithm** opera em ciclos iterativos, onde as abelhas empregadas, observadoras e scouts trabalham juntas para melhorar as soluções ao longo do tempo. A qualidade das soluções é avaliada com base numa função objetivo específica associada ao problema de otimização em questão.

Este algoritmo tem sido aplicado com sucesso em diversas áreas, incluindo otimização de funções matemáticas, problemas de programação linear, **design de redes neurais**, entre outros. A sua eficácia está relacionada com a sua capacidade de explorar o espaço de busca de maneira eficiente, combinando exploração local e global para encontrar soluções de alta qualidade.

Comparando o **Artificial Bee Algorithm** com o **Particle Swarm Otimization**, conseguimos destacar as vantagens e desvantagens de cada um:

Vantagens do **Artificial Bee Algorithm**:

- **Exploração Eficiente:** o algoritmo é projetado para realizar uma exploração eficiente do espaço de busca, equilibrando a exploração local e global através das abelhas empregadas, observadoras e scouts;
- **Adaptação Dinâmica:** a capacidade das abelhas empregadas e observadoras de se adaptarem dinamicamente com base nas soluções encontradas contribui para uma exploração mais adaptativa;
- **Diversidade na Busca:** a presença de abelhas scouts assegura que o algoritmo continue explorando novas regiões, aumentando a diversidade da busca.

Vantagens do **Particle Swarm Optimization**:

- **Simplicidade:** conhecido pela sua simplicidade conceptual e implementação, facilitando a compreensão e aplicação numa variedade de problemas;
- **Rápida Convergência:** pode convergir rapidamente para soluções de alta qualidade, especialmente em problemas que possuem uma única solução ótima;
- **Robustez:** muitas vezes robusto e eficaz em uma ampla gama de problemas de otimização, tornando-o uma escolha atraente para problemas diversos.

Desvantagens do **Artificial Bee Algorithm**:

- **Parâmetros sensíveis:** o desempenho pode ser sensível à escolha dos parâmetros, como o número de abelhas e outros critérios;
- **Convergência Lenta:** pode ter uma convergência mais lenta em comparação com outros algoritmos, especialmente quando a exploração local é mais intensiva.

Desvantagens do **Particle Swarm Optimization**:

- **Exploração Limitada:** dificuldade em lidar com problemas que exigem uma exploração mais ampla do espaço de busca, especialmente quando há múltiplos ótimos locais;
- **Suscetível a Mínimos Locais:** problemas com múltiplos mínimos locais, o PSO pode ficar preso em soluções subótimas se não houver estratégias eficazes de exploração;
- **Dificuldade em Problemas Dinâmicos:** enfrentar desafios em problemas dinâmicos nos quais as condições do ambiente ou a função objetivo mudam ao longo do tempo.

Apesar de ambos terem as suas vantagens e desvantagens, a sua escolha dependerá das características do problema e dos requisitos específicos de otimização.

APLICAR E ILUSTRAR O ALGORITMO PARA OTIMIZAÇÃO

Utilização de uma função "benchmark" – função **Ackley** para as **dimensões 2** e **3**. Após a aplicação do algoritmo, obtemos os seguintes resultados, para os seguintes parâmetros:

Utilização do **Particle Swarm Optimization**:

- **Dimensão 2, 10 abelhas, 100 iterações:**

  Best Value (2D): [5.856455978363597e-11, 2.6546192876166378e-11]

- **Dimensão 2, 10 abelhas, 500 iterações:**

  Best Value (2D): [-1.2880686817778088e-15, 1.2160479882367236e-15]

- **Dimensão 2, 10 abelhas, 1000 iterações:**

  Best Value (2D): [3.2376437936552357e-16, -1.4907220965317917e-16]

- **Dimensão 2, 20 abelhas, 100 iterações:**

  Best Value (2D): [4.096818997326619e-16, 1.7279410103442546e-15]

- **Dimensão 2, 20 abelhas, 500 iterações:**

  Best Value (2D): [-6.153204861309716e-17, 1.2384786536676443e-16]

- **Dimensão 2, 20 abelhas, 1000 iterações:**

  Best Value (2D): [6.649548832856181e-16, -1.6695317252880003e-15]

- **Dimensão 2, 30 abelhas, 100 iterações:**

  Best Value (2D): [1.6392877242491925e-12, 3.882509655329556e-12]

- **Dimensão 2, 30 abelhas, 500 iterações:**

  Best Value (2D): [-1.8246952430665198e-17, -1.589343252627192e-16]

- **Dimensão 2, 30 abelhas, 1000 iterações:**

  Best Value (2D): [2.086642199148853e-16, -3.180658097134193e-17]

- **Dimensão 3, 10 abelhas, 100 iterações:**

Best Value (3D): [-7.652331391671728e-12, -3.706032107079239e-11, -8.385834866147239e-10]

- **Dimensão 3, 10 abelhas, 500 iterações:**

Best Value (3D): [2.390233570642506e-07, -0.9392013464754835, -1.6172657153162124e-06]

- **Dimensão 3, 10 abelhas, 1000 iterações:**

Best Value (3D): [-9.36408270302882e-16, -1.4021055176285652e-15, 1.2860119258155073e-15]

- **Dimensão 3, 20 abelhas, 100 iterações:**

Best Value (3D): [-4.8515003600055126e-11, -2.136014989558976e-11, 1.9780005400249897e-11]

- **Dimensão 3, 20 abelhas, 500 iterações:**

Best Value (3D): [-2.654597196259228e-16, 7.796467251070024e-17, -1.4821879137292834e-15]

- **Dimensão 3, 20 abelhas, 1000 iterações:**

Best Value (3D): [-1.502910762373696e-16, 2.8486247094089334e-17, 9.986564252866852e-17]

- **Dimensão 3, 30 abelhas, 100 iterações:**

Best Value (3D): [3.98777039164199e-11, 1.2424714586886277e-10, -6.095465652433762e-11]

- **Dimensão 3, 30 abelhas, 500 iterações:**

Best Value (3D): [-2.0365821309734173e-15, 2.0285579711594462e-16, 7.270390594856231e-16]

- **Dimensão 3, 30 abelhas, 1000 iterações:**

Best Value (3D): [-1.5689351285423033e-16, -1.4078002295431425e-16, 1.5733094121800051e-15]

Utilização do **Artificial Bee Algorithm**:

- **Dimensão 2, 10 abelhas, 100 iterações:**

Best Value (2D): [-0.02249126180786693, 0.9769652739642734]

- **Dimensão 2, 10 abelhas, 500 iterações:**

Best Value (2D): [-1.2753932923859965, -1.9996459900851833]

- **Dimensão 2, 10 abelhas, 1000 iterações:**

  Best Value (2D): [-0.0160334773878847, 0.06301623776436135]

- **Dimensão 2, 20 abelhas, 100 iterações:**

  Best Value (2D): [7.85503912129448e-06, -1.1219920165443933e-05]

- **Dimensão 2, 20 abelhas, 500 iterações:**

  Best Value (2D): [2.1991987454980017e-08, -3.500965288815962e-08]

- **Dimensão 2, 20 abelhas, 1000 iterações:**

  Best Value (2D): [-1.2565813391194223e-07, 1.7967136732169822e-08]

- **Dimensão 2, 30 abelhas, 100 iterações:**

  Best Value (2D): [-2.5747281459537998e-08, 8.637678228102488e-08]

- **Dimensão 2, 30 abelhas, 500 iterações:**

  Best Value (2D): [-2.443639401297953e-06, -2.3721612628080866e-07]

- **Dimensão 2, 30 abelhas, 1000 iterações:**

  Best Value (2D): [-3.4294032263347637e-11, -4.546453578057263e-12]

- **Dimensão 3, 10 abelhas, 100 iterações:**

  Best Value (3D): [-0.9504123802214975, -0.06495056016012633, 1.407648388817473]

- **Dimensão 3, 10 abelhas, 500 iterações:**

  Best Value (3D): [1.6890024980434815, 1.7072384222746835, -0.9644158991981875]

- **Dimensão 3, 10 abelhas, 1000 iterações:**

  Best Value (3D): [-0.9262781831250715, 2.561650780558278, -0.9023044325222056]

- **Dimensão 3, 20 abelhas, 100 iterações:**

  Best Value (3D): [-0.02075086913238773, -0.015083480598648315, 0.014992665847080788]

- **Dimensão 3, 20 abelhas, 500 iterações:**

  Best Value (3D): [-4.723254614529575e-08, -5.201817572653998e-06, 2.66821384625816e-06]

- **Dimensão 3, 20 abelhas, 1000 iterações:**

Best Value (3D): [6.045192517527838e-06, -2.7107166444835207e-06, -2.428972381808234e-06]

- **Dimensão 3, 30 abelhas, 100 iterações:**

Best Value (3D): [-1.700989530873905e-05, -0.000857725909110788, 0.0002112020282893853]

- **Dimensão 3, 30 abelhas, 500 iterações:**

Best Value (3D): [-4.175181822145396e-06, 2.90353573793868e-06, 2.7257084152280297e-07]

- **Dimensão 3, 30 abelhas, 1000 iterações:**

Best Value (3D): [-5.8516574557714e-08, -9.640283492304845e-09, -6.565353137189966e-08]

O melhor resultado seria aquele que tem o menor valor absoluto de "Best Value (2D/3D)", pois isso indica a solução mais próxima do mínimo global da **função de Ackley**. Sendo assim, os melhores resultados são:

- **Particule Swarm Optimization:**
    - **Dimensão 2, 10 abelhas, 1000 iterações** (melhor para 2D);
    - **Dimensão 3, 10 abelhas, 1000 iterações** (melhor para 3D).
- **Artificial Bee Algorithm:**
    - **Dimensão 2, 10 abelhas, 1000 iterações** (melhor para 2D);
    - **Dimensão 3, 10 abelhas, 1000 iterações** (melhor para 3D).
- **Entre ambas:**
    - **Dimensão 2, 10 abelhas, 1000 iterações** (Particle Swarm Optimization)
    - **Dimensão 3, 10 abelhas, 1000 iterações** (Particle Swarm Optimization)

Sendo assim, de acordo com os resultados obtidos, o Agoritmo de Otimização que obteve melhores resultados foi o **Particule Swarm Optimization**.

OTIMIZAÇÃO DE HÍPER-PARÂMETROS

De acordo com o enunciado do **Projeto Fase II**, são considerados híper-parâmetros: o número de camadas, **número de neurónios por camada**, **tipo de função de ativação**, **algoritmo de otimização**, coeficiente de aprendizagem e o tipo de regularização.

Para avaliar qual os melhores híper-parâmetros, utilizámos uma função **evaluate()** que retorna o **Accuracy** da rede neuronal com os híper-parâmetros selecionados através do **Particule Swarm Optimization** (**n = 10**, **function = evaluate**, **lb = lb**, **ub = ub**, **dimension = 2**, **iteration 10**). Assim sendo, decidimos otimizar a arquitetura da nossa rede neuronal, alterando os seguintes **hiper-parâmetros**, e obtendo os seguintes **resultados**:

- **Alteração do número de neurónios das camadas 1 e 2:**
  - Valores entre 50 e 200, para ambas as camadas.

```
Best Parameters - [147.95301086801769, 193.82215413677895]
Best Parameters:  (147, 193)
Accuracy on Test Data (after optimization): 55.58%
```

```
lb = [50, 50]
ub = [200, 200]
```

- **Alteração do número de neurónios da camada 1 e da função de ativação:**
  - Valores entre 50 e 200 para a camada 1;
  - Funções de ativação: 0 = logistic, 1 = tanh.

```
Best Parameters - [61.810762519457185, logistic]
Best Parameters:  (61, logistic)
Accuracy on Test Data (before optimization): 53.30%
```

```
lb = [50, 0]
ub = [200, 1]
```

- **Alteração dos coeficientes de aprendizagem:**
  - Valores entre 0.001 e 0.01;
  - Valores entre 0.1 e 0.9.

PROCESSO DEMASIADO LONGO.

```
lb = [0.001, 0.1]
ub = [0.01, 0.9]
```

Após a otimização da arquitetura da rede neuronal através do **Particule Swarm Optimization**, temos de aplicar o **Artificial Bee Algorithm** para fazer a comparação com o anterior e determinar a melhor configuração.

ERROR:

Ao tentar implementar o **Artificial Bee Algorithm** na nossa rede neuronal, obtivemos vários erros e acabámos por não conseguir resolvê-los no prazo da submissão da Fase II do Projeto de Inteligência Computacional.

- ValueError: 1.9213483146067414 is not in list
- ValueError: 2.011764705882353 is not in list
- ValueError: 1.9213483146067414 is not in list
- ValueError: 1.9000000000000001 is not in list

Todos estes erros foram encontrados em:

- File ~\anaconda3\Lib\site-packages\SwarmPackagePy\aba.py:49 in <listcomp>
  [fitness.index(x) for x in sort_fitness[:count[0]]]]

CONCLUSÃO E DISCUSSÃO DE RESULTADOS

Relembrando a Fase I do Projeto de Inteligência Computacional, foram obtidos os seguintes resultados para o Teste da rede neuronal criada:

- **Número de neurónios da camada 1**: 300
- **Accuracy**: 40.00%

Nesta Fase II do Projeto, e com um conhecimento da matéria alargado relativamente à fase anterior, obtivemos uma melhoria dos resultados, como seria de esperar.

- **Número de neurónios da camada 1**: 147
- **Número de neurónios da camada 2**: 193
- **Accuracy**: 55.58%

- **Número de neurónios da camada 1**: 94
- **Função de ativação**: logistic
- **Accuracy**: 53.30%

De acordo com os resultados obtidos, podemos concluir que houve uma melhoria significativa da eficácia desta rede neuronal, visto que houve um aumento de cerca de 14% em relação ao valor de Accuracy calculado na Fase I do Projeto.

FIM