Resolução do problema da mochila compartimentada usando Inspyred

Robson Liesner
Instituto Federal Catarinense - IFC
Videira, SC - Brasil
Email: robsonlljr@gmail.com

Resumo—O inspyred é uma estrutura de código aberto gratuita para a criação de algoritmos de inteligência computacional inspirados biologicamente em Python, incluindo computação evolutiva, inteligência de enxame e imunocomputação. Além disso, o inspyred fornece versões canônicas fáceis de usar de muitos algoritmos bioinspirados para usuários que não precisam de muita personalização. Com o intuito de utilizar as funcionalidades do framework Inspyred e tudo que foi aprendido durante as aulas, foi reproduzido o problema da mochila compartimentada desenvolvido no artigo de Fabiano do Prado Marques que desenvolveu um estudo com os diversos tipos diferentes do problema da mochila.

I. Introdução

O problema da mochila consiste em alcançar o maior número de itens dentro de uma mochila onde o peso total dos itens inseridos na mochila não ultrapasse o máximo permitido, é um problema de otimização combinatória clássico da computação, estudado desde os anos 80. Nos dias de hoje é possível encontrar diversas soluções propostas para o problema, desde algoritmos simples a algoritmos complexos [1]. Com isso, no artigo que foi levado como base foi estudado o problema de otimização combinatorial conhecido por Problema da Mochila Compartimentada, que é uma extensão do clássico Problema da Mochila. O problema da mochila compartimentada consiste em determinar as capacidades adequadas de vários compartimentos que podem vir a ser alocados em uma mochila e como esses compartimentos devem ser carregados, respeitando as restrições de capacidades dos compartimentos e da mochila. Busca-se maximizar o valor de utilidade total. Segundo o autor o problema é muito pouco estudado na literatura, apesar de surgir naturalmente em aplicações práticas.

II. Inspyred Library

A biblioteca inspyred cresceu a partir dos insights do livro de Ken de Jong “Evolutionary Computation: A Unifled Approach”. O objetivo da biblioteca, como foi dito anteriormente, é separar a computação específica do problema da computação específica do algoritmo. Qualquer algoritmo bioinspirado tem pelo menos dois aspectos que são inteiramente específicos do problema: qual a aparência das soluções para o problema e como essas soluções são avaliadas. Esses componentes certamente mudarão de problema para problema. Por exemplo, um problema de otimização do volume de uma caixa pode representar soluções como lista de três elementos de valores reais para comprimento, largura e altura, respectivamente. Em contraste, um problema que lida com a otimização de um conjunto de regras para escapar de um labirinto pode representar soluções como uma lista de par de elementos.

O princípio central de design para o inspyred é separar componentes específicos do problema dos componentes específicos do algoritmo de uma maneira limpa, de modo a tornar os algoritmos o mais geral possível em uma variedade de problemas diferentes.

```
from random import Random
from time import time
import inspyred

def main(prng=None, display=False):
    if prng is None:
        prng = Random()
        prng.seed(time())

    problem = inspyred.benchmarks.Binary(inspyred.benchmarks.Schwefel(2),
                                         dimension_bits=30)
    ea = inspyred.ec.GA(prng)
    ea.terminator = inspyred.ec.terminators.evaluation_termination
    final_pop = ea.evolve(generator=problem.generator,
                          evaluator=problem.evaluator,
                          pop_size=100,
                          maximize=problem.maximize,
                          bounder=problem.bounder,
                          max_evaluations=30000,
                          num_elites=1)

    if display:
        best = max(final_pop)
        print('Best Solution: \n{0}'.format(str(best)))
    return ea

if __name__ == '__main__':
    main(display=True)
```

Figura 1. ilustração da estrutura do inspyred

A biblioteca inspyred vê os cálculos evolutivos como sendo compostos das seguintes partes:

Componentes específicos do problema:

- Um gerador que define como as soluções são criadas;
- um avaliador que define como os valores de aptidão são calculados para soluções;

Operadores evolutivos específicos de algoritmo:

- Um observador que define como o usuário pode monitorar o estado da evolução;
- Um terminador que determina se a evolução deve terminar;
- Um seletor que determina quais indivíduos devem se tornar pais;
- Um variador que determina como a prole é criada a partir de indivíduos existentes;
- Um substituto que determina quais indivíduos devem sobreviver na próxima geração;
- Um migrador que define como as soluções são transferidas entre diferentes populações;
- Um arquivador que define como as soluções existentes são armazenadas fora da população atual;

III. Problema da Mochila Compartimentada

A. Formulação do Problema

Basicamente, o problema da mochila é definida do seguinte modo: Suponha que um alpinista deve carregar sua mochila selecionando itens, dentre vários disponíveis, e considerando a capacidade da mochila. A cada item é atribuído um valor de utilidade e o alpinista deve selecioná-los buscando maximizar o valor de utilidade total. Até este ponto, o problema coincide com a formulação clássica do problema da Mochila. Além disso, muitos itens são de classes distintas (por exemplo: classe 1, classe 2 e classe 3) e devem ser divididos entre os compartimentos da mochila. Os compartimentos da mochila são flexíveis, pois os itens de cada classe são de diferentes tamanhos. Entretanto, as capacidades dos compartimentos são limitadas a inferior e superiormente, caso estes sejam criados, digamos por: Peso mínimo e Peso máximo. Cada compartimento incluído na mochila reproduz uma perda da capacidade da mesma, isto é, 3 compartimentos forem utilizados, então a capacidade disponível na mochila será de capacidade total - capacidade atual. É importante ressaltar que vários compartimentos, de capacidades diferentes, podem ser criados para carregar itens de uma mesma classe, em que itens de uma mesma classe podem ser alocados em mais em um compartimento.

B. Resolução do Problema

Considerando os seguintes conjunto de dados:

- Peso máximo no compartimento: 456
- Peso mínimo no compartimento: 156
- Peso suportado pela mochila: 1368
- Existem apenas 3 compartimentos na mochila que são o: Traseiro, Frontal e Lateral;
- Respeita os dados contidos na tabela de cada classe de itens, como mostra na imagem:

Dados das Classes de Itens:

Classe 1	Custo = 1.50		Classe 2		Custo = 0.50		Classe 3		Custo = 2.50	
Itens	1	2	3	4	5	6	7	8	9	10
Peso	55	58	57	54	56	50	52	54	55	58
Valor	28.00	30.00	7.00	26.00	30.00	27.00	8.00	29.00	10.00	35.00
Demanda	15	16	12	14	20	13	14	10	12	11

Dados dos Itens Livres:

Classe Livre		Custo = 0.00			
11	12	13	14	15	16
53	56	55	58	60	51
23.00	12.00	21.00	9.00	26.00	33.00
21	23	26	21	17	15

Figura 2. Dados de cada classe e item

- Classe livre: essa classe é caracterizada por itens especiais do alpinista e não possuem custo de volume na mochila;

Identificar a natureza do problema de otimização:

- Máximo de lucro que pode levar na mochila pela escolha certa das cargas disponível

Identificar o conjunto de variáveis para o problema:

- Compartimentos: Traseiro (T), Frontal (F) e Lateral L;
- Classe 1 :
  - C1_1 = c1_T1, c1_F1, c1_L1;
  - C1_2 = c1_T2, c1_F2, c1_L2;
- Classe 2:
  - C2_1 = c2_T1, c2_F1, c2_L1;
  - C2_2 = c2_T2, c2_F2, c2_L2;
  - C2_3 = c2_T3, c2_F3, c2_L3;
  - C2_4 = c2_T4, c2_F4, c2_L4;
- Classe 3:
  - C3_1 = c3_T1, c3_F1, c3_L1,
  - C3_2 = c3_T2, c3_F2, c3_L2,
  - C3_3 = c3_T3, c3_F3, c3_L3,
  - C3_4 = c3_T4, c3_F4, c3_L4;
- Classe 4:

- C4_1 = c4_T1, c4_F1, c4_L1,
- C4_2 = c4_T2, c4_F2, c4_L2,
- C4_3 = c4_T3, c4_F3, c4_L3,
- C4_4 = c4_T4, c4_F4, c4_L4,
- C4_5 = c4_T5, c4_F5, c4_L5,
- C4_6 = c4_T6, c4_F6, c4_L6;

Demanda limite de cada item:

- C1_1 <=15**;**
- C1_2 <=16;
- C2_1 <=12;
- C2_2 <= 14;
- C2_3 <= 20;
- C2_4 <= 13;
- C3_1 <= 14;
- C3_2 <= 10;
- C3_3 <= 12;
- C3_4 <= 11;
- C4_1 <= 21;
- C4_2 <= 23;
- C4_3 <= 26;
- C4_4 <= 21;
- C4_5 <= 17;
- C4_6 <= 15;

Identifique e formule matematicamente as restrições do problema:

- Soma dos pesos das cargas traseira que pode ser aceita na parte traseira da mochila;
- Soma dos pesos das cargas lateral que pode ser aceita na parte frontal da mochila;
- Soma dos pesos das cargas lateral que pode ser aceita na parte lateral da mochila;
- Soma de todas os itens da classe 1 menos o volume aceito na mochila;
- Soma de todas os itens da classe 2 menos o volume aceito na mochila;
- Soma de todas os itens da classe 3 menos o volume aceito na mochila;
- Soma de todos os itens possíveis no compartimento traseiro da mochila dividido pela soma total de todas as cargas na mochila, menos o peso máximo do compartimento;
- Soma de todos os itens possíveis no compartimento frontal da mochila dividido pela soma total de todas as cargas na mochila, menos o peso máximo do compartimento;
- Soma de todos os itens possíveis no compartimento lateral da mochila dividido pela soma total de todas as cargas na mochila, menos o peso máximo do compartimento;
- Restrição de demanda de cada item para que não ultrapasse a quantidade disponível;

Identificar a Função Objetivo:

- Lucro(C1_1, C1_2, C2_1, C2_2, C2_3, C2_4, C3_1, C3_2, C3_3, C3_4, C4_1, C4_2, C4_3, C4_4, C4_5, C4_6) = (C1_1 * 28) + (C1_2 * 30) + (C2_1 * 7) + (C2_2 * 26) + (C2_3 * 30) + (C2_4 * 27) + (C3_1 * 8) + (C3_2 * 29) + (C3_3 * 10) + (C3_4 * 35) + (C4_1 * 23) + (C4_2 * 12) + (C4_3 * 21) + (C4_4 * 9) + (C4_5 * 26) + (C4_6 * 33)

Desenvolvedor a função de fitness adequada:

- Função Fitmess = Função Lucro / Estimativa grosseira da função lucro;
- Função Fitness = Função Lucro / 5637;

IV. Testes e Resultados

Inicialmente foi testado com 10 mil gerações que o inspyred iria percorrer e depois de inúmeros testes e codificação das restrições do problema verifiquei que era necessário apenas 5 mil gerações para chegar em um resultado otimizado.

```
final_pop = ea.evolve(generator=gera_populacao,
                      evaluator=avaliacao,
                      pop_size=1000,
                      maximize=True,
                      bounder=ec.Bounder(0, 456),
                      max_generations=5000,
                      num_inputs=48,
                      crossover_points=1,
                      mutation_rate=0.25,
                      num_elites=1,
                      num_selected=48,
                      tournament_size=48)
```

Figura 3. valores inseridos para determinação do inspyred

Como mostrado na imagem, ficou definido apenas 0.25 de mutação e limite de Bounder de 456 que é o limite suportado em cada compartimento da mochila.

Resultados obtidos na classe 1:

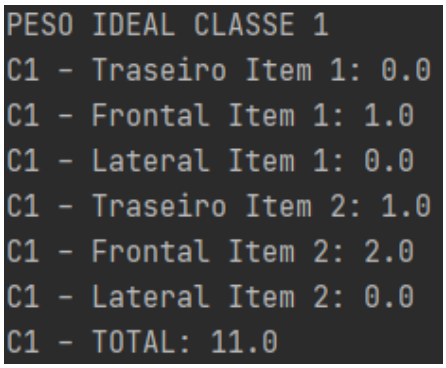
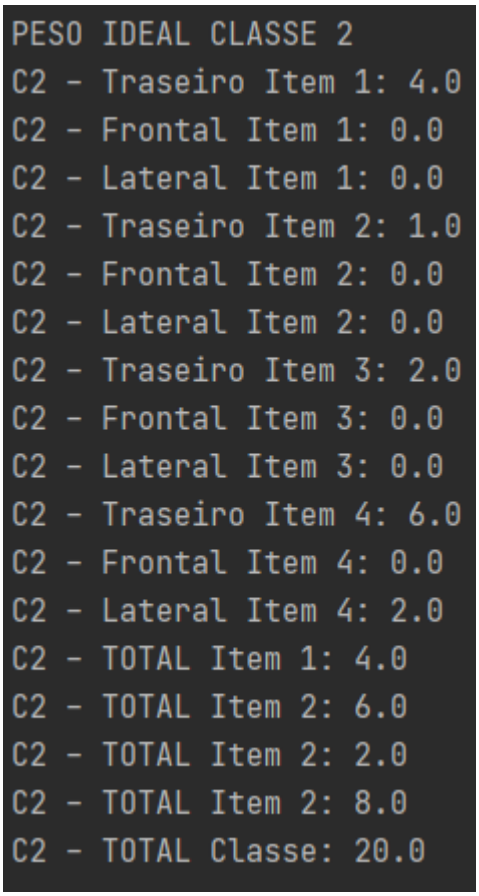


Figura 3. Quantidade de Itens da classe 1

Resultados obtidos na classe 2:



. Figura 4. Quantidade de itens da classe 2

Resultados obtidos na classe 3:

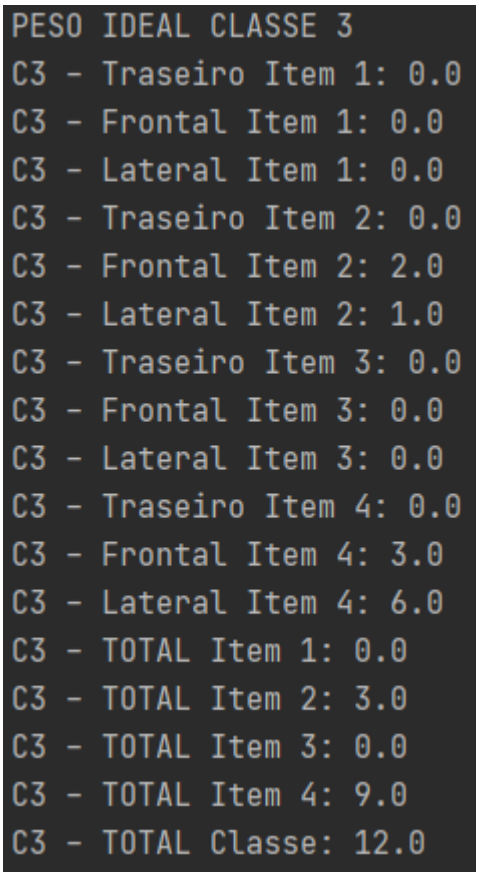


Figura 5. Quantidade de itens da classe 3

Resultados obtidos na classe 4:

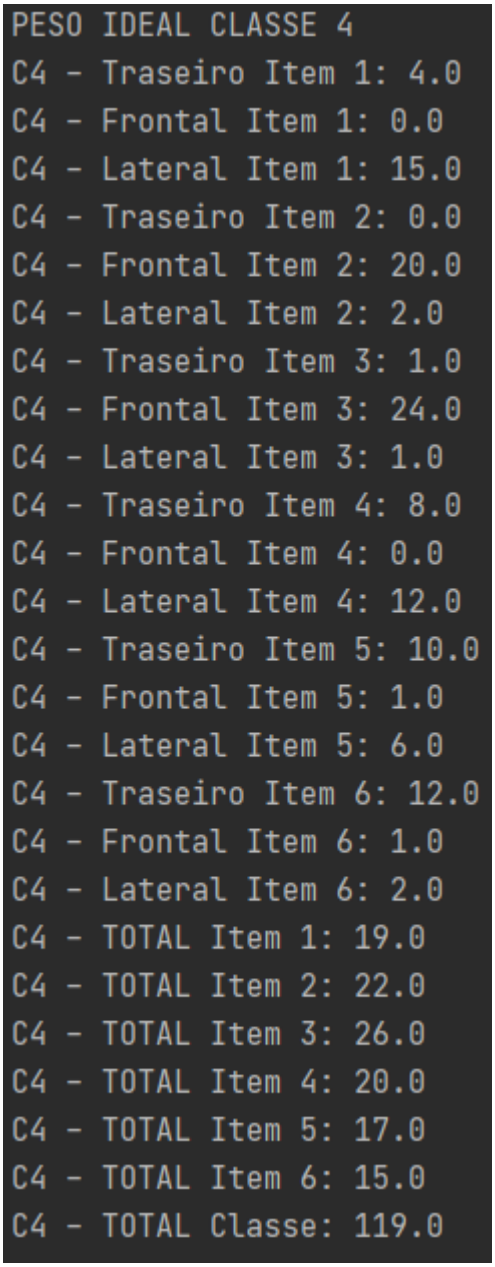


Figura 6. Quantidade de itens da classe 4

Valores obtidos em cada classe:

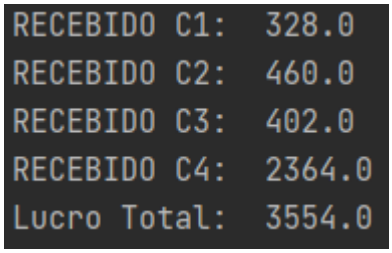


Figura 7. Lucro recebido

V. Conclusão

Concluo que todos os objetivos que foram descritos no início do trabalho foram concluídos, após o estudo geral do problema da mochila compartimentada apresentado no artigo de Fabiano de Prado Marques. Seguindo o roteiro de resolução de problema ensinado durante as aulas, foi definido as variáveis necessárias, codificar as restrições que são descritas no problema, criar a função objetivo levando em fato obter o lucro máximo e assim determinar a função fitness do problema.

Referências

[1] A. G. Silva, “Aplicação de algoritmos gené
ticos.” Online; acesso em 10 julho, 2021,
http://www.inf.ufsc.br/alexandre.goncalves.silva/courses/14s2/ine5633/sli
des/aulaAG_*aplicacoes.pdf.*

[2] INITIATIVE, Inspired Intelligence. Inspyred. Disponível em: https://pythonhosted.org/inspyred/examples.html#genetic-algorithm. Acesso em: 20 jul. 2021.

[3] MARQUES, Fabiano do Prado. O problema da mochila compartimentada*. 2000. 119 f. Tese (Doutorado) - Curso de Ciência da Computação e Matemática Computacional, Usp, São Carlos, 2000. Disponível em: https://www.teses.usp.br/teses/disponiveis/55/55134/tde-14012005-165350/publico/pmc.pdf. Acesso em: 16 jul. 2021.