Algoritmo Genético e Busca em Feixe Aplicado ao Problema Combinatório da Mochila

Everaldo R. S. Junior [1], **Matheus S. P. Lima**[2]

[1]Departamento de Informática – Universidade Tecnológica Federal do Paraná (UTFPR)
Curitiba – PR – Brazil

everaldojunior@alunos.utfpr.edu.br, matheus.pedrozo@gmail.com

Abstract. *This paper describes the process of formalizing and modeling the implementation of a solution for the combinatorial optimization problem, know as the "backpack problem", using the genetic algorithm and beam search technics.*

Resumo. *Este artigo tem como objetivo descrever, formalizar e modelar o processo de implementação de uma solução para o prolema de otimização combinatória da "Mochila"através do uso das técnicas de algoritmos genéticos e buscas em feixe.*

1. Problema da mochila

O problema da mochila é um problema de otimização combinatória: Dado um conjunto de itens, cada um com peso e valor atrelado, deve-se determinar a quantidade de cada item a ser incluído na mochila, de modo que o peso total é menor ou igual à um peso limite, e o valor total é o maior possível.

Um dos problemas mais comuns a ser resolvido é o "problema da mochila 0/1"onde o número de cópias x_i de cada tipo é restringido a zero ou um. Dado um conjunto n de itens numerados de 1 até n, cada um com um peso respectivo w_i e um valor v_i, junto à um peso máximo W,

$$\text{maximize } \sum_{i=1}^{n} v_i x_i \text{ sujeito a, } \sum_{i=1}^{n} w_i x_i \leqslant W \text{ onde } x_i \in \{0, 1\}$$

2. Modelagem

Com a escolha do tipo de problema a ser modelado, o primeiro passo a ser tomado é a escolha de linguagem usada para a implementação e modo de representação dos conjuntos de dados. Como linguagem de programação foi escolhido o *javascript* por ser uma ferramenta de fácil configuração e uso, além da experiência dos integrantes com o recurso.

2.1. Indivíduos

Nessa sessão é descrito como acontece a criação e qual a estrutura dos indivíduos (estados) utilizados para a execução do Algoritmo Genético e Busca em Feixe Local

2.2. Itens possíveis

Para a criação dos itens candidatos a entrar na mochila são ciadas funções que populam uma lista, em forma de vetor, de itens. Cada item recebe de forma aleatória e única um peso (*Weight*), valor (*Value*) e identificador. A tabela 1 apresenta a maneira na qual os dados de valor e peso estão organizados.

Items	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Value (v)	2	12	14	4	1	9	4	2	8	5	8	1	1	6	3
Weight (W)	18	13	7	13	4	14	20	15	15	10	20	8	17	8	20

Tabela 1. Itens possíveis nas mochilas

2.3. População Inicial

Os algoritmos são iniciados a partir da geração de uma população composto por *n* indíviduos, que tem a seguinte estrutura:

Individual	Type
id	String
totalValue	int
totalWeight	int
backpack	0|1[ ]

Tabela 2. Estrutura do Indivíduo

Para a primeira geração, a mochila será preenchida de forma aleatória, cada índice *n* no campo *backpack*, corresponde ao mesmo índice n na lista de Itens criados anteriormente, sendo 0 para não presente e 1 para presente.

2.4. Função Fitness

Durante a geração da população, cada indivíduo é avaliado segundo uma função de *fitness*, que no caso do problema foi modelado da seguinte maneira: Se o campo *totalWeigth* for maior que a capacidade máxima da mochila, avaliamos o campo *fitness* como 0, caso seja maior ou igual, atribuímos o mesmo valor do campo *totalValue* para o campo *fitness*.

3. Algoritmo genético

Um Algoritmos genético é uma variante de busca em feixe estocástica, baseado na teoria da Evolução de Charles Darwin [Peter Norvig 2009], onde estados (indivíduos) são selecionados, reproduzem, e por fim estados sucessores são gerados pela combinação de outros dois estados. De forma breve, um estado pode ser chamado de indivíduo e são representados em uma estrutura que é chamada de cromossomo, que por fim é uma estrutura de dado onde as funções de seleção, reprodução e mutação operam, cada estado desse cromossomo que pode ser invidualizado é chamado de alelo. [3]

3.1. Modelagem do algoritmo genético para o Problema da Mochila

3.1.1. Seleção

A estratégia adotada para a seleção dos indivíduos, foi a do torneio. Onde dois indivíduos da população são escolhidos de forma aleatória, e o indivíduo com o maior *fitness* é separado para o processo de reprodução.

3.1.2. Reprodução

Na etapa de reprodução dois indivíduos do grupo previamente selecionados são escolhidos aleatoriamente. Um número aleatório entre 0 e 1 é escolhido e se ele for maior que a variável responsável pela chance do cruzamento dos cromossomos acontecerem, então um novo individuo é gerado tendo o campo *backpack* preenchido de forma aleatória entre os alelos dos pais geradores. Caso a reprodução não aconteça, então o novo individuo gerado terá o campo *backpack* igual ao do pai gerador.

3.1.3. Mutação

Durante a reprodução existe uma chance de uma mutação acontecer, sendo calculada da mesma forma da chance de cruzamento na etapa anterior. A mutação consiste em uma função que sorteia um índice aleatório no campo *backpack* e altera seu estado de 0 para 1 um e vice-versa.

3.1.4. Evolução e Condição de Parada

Após a criação da nova geração, o processo começa novamente a partir da etapa de seleção. O critério de parada para as iterações escolhido foi o de número de gerações.

4. Busca em Feixe Local

Assim como o Algoritmo genético, a Busca em Feixe Local é um algoritmo heurístico de busca [1], que pode ser considerado uma variante do algoritmo chamado Subida de Encosta, que tem como principal característica expandir (criando um feixe), os nós vizinhos mais promissores de acordo com a modelagem adotada até encontrar o valor desejado ou atingir um critério de parada [Peter Norvig 2009].

4.1. Modelagem da Busca em Feixe Local para o Problema da Mochila

4.1.1. Feixe inicial

Após a criação da população inicial descrita na sessão 2.3, *k* indivíduos aleatórios da população são selecionados, todos os sucessores (nesse caso vizinhos) dos estados selecionados são alocados em uma nova lista c.

4.1.2. Seleção

O algoritmo então seleciona os *k* melhores indivíduos dessa lista c de acordo com a função de *fitness*, definida na sessão 2.4 e o processo é repetido, gerando um novo feixe.

4.1.3. Condição de Parada

Assim como na modelagem do Algoritmo Genético, o critério de parada foi o número de gerações.

5. Metodologia

Os valores e pesos dos itens serão os definidos na sessão 2.1 Os demais parâmetros serão definidos e configurados como valores constantes em código de acordo com a tabela a seguir:

Paramêtros	Valor
PESO MAX MOCHILA	75
TAXA DE CRUZAMENTO	0.53
TAXA DE MUTAÇÃO	0.05
POPULAÇÃO INICIAL	1.000

Tabela 3. Parâmetros

Sendo o parâmetro POPULAÇÃO INICIAL e PESO MAX MOCHILA utilizados tanto para a execução do Algoritmo Genético quanto para a Busca em Feixe Local e os demais parâmetros (TAXA DE CRUZAMENTO e TAXA DE MUTAÇÃO) utilizados apenas para o Algoritmo Genético.

6. Resultados obtidos

Número de Gerações	10	100	1.000	10.000
Fitness médio	16.48	11.44	10.61	13.14
Melhor Fitness da geração	53	55	55	55
Melhor soluçao encontrada	Não	Sim	Sim	Sim

Tabela 4. Resultados (AG)

Número de Gerações	10	100	1.000	10.000
Fitness médio	22.11	44.23	49.28	46.02
Melhor Fitness da geração	26	50	53	55
Melhor soluçao encontrada	Não	Não	Não	Sim

Tabela 5. Resultados (LBS)

7. Conclusão

Com a execução dos dois algoritmos com diferentes cenários, algumas características de cada algoritmo heurístico ficaram evidentes. A Busca em Feixe Local foi capaz de encontrar a solução ótima, porém apenas em cenários com 10.000 gerações, mesmo sendo melhor nos quesitos *fitness* médio por geração e Melhor *Fitness* da geração (indivíduo) se comparado aos mesmos cenários aplicados ao Algoritmo genético, que por sua vez encontrou a solução ideal já no cenário com 100 gerações, porém com *Fitness*

médio por geração bem inferior se comparado com a Busca em Feixe Local. A modelagem do ambiente que optamos acaba favorecendo uma média baixa na execução do Algoritmo genético, já que atribuimos o mesmo valor de *fitness* para individuos que passam do limite da mochila por 1 ou 10 unidades por exemplo.

Já na busca em feixe local a lista da pontuação de de *fitness* dos individuos gerados não se deu de forma contínua, o que acaba dificultando a eficiência do algoritmo em achar um ótimo local.

O código fonte utilizado para a execução dos algoritmos com os diferentes cenários pode ser visto neste repositório do Github.

8. Divisão do Trabalho

Aluno	Atividade	Tempo
Everaldo Junior	Leitura do livro base Inteligência Artificial	4 horas
Everaldo Junior	Implementação	5 horas
Everaldo Junior	Escrita do artigo	3 horas
Matheus Lima	Leitura do livro base Inteligência Artificial	6 horas
Matheus Lima	Escrita do artigo	5 horas

Referências

[1] Algoritmos genéticos. `https://sites.icmc.usp.br/andre/research/genetic/`.

[3] Pacheco, marco. algoritmos geneticos: Principios e aplicaçoes. `http://www.inf.ufsc.br/~mauro.roisenberg/ine5377/Cursos-ICA/CE-intro_apost.pdf`.

[Peter Norvig 2009] Peter Norvig, S. R. (2009). *Inteligência Artificial*. Prentice Hall, 3th edition.