Test-Assignment Problem

Murilo Antunes Goedert
Engenharia de Software UDESC
Ibirama/SC - Brasil
murilogoedert@gmail.com

Victor Hugo Grabowski Beltramini
Engenharia de Software UDESC
Ibirama/SC - Brasil
vhbeltramini@gmail.com

RESUMO

A atribuição de testes é um problema de otimização complexo, consiste em encontrar a melhor maneira de distribuir um conjunto de diferentes testes dentro de uma sala de aula a fim de minimizar a probabilidade de ocorrer cola. O presente documento tem como objetivo resolver esse problema de otimização implementando um conjunto de metaheurísticas incluindo estratégias construtivas e por modificação, e realizando um estudo experimental a partir do resultado destas implemetações.

PALAVRAS CHAVE. Alocação de testes, Metaheurísticas, Problemas de otimização.

ABSTRACT

The test assignment is a complex optimization problem, it consists of finding the best way to distribute a set of different tests within a classroom in order to minimize the probability of cheating occurring. This document aims to solve this problem of optimization by implementing a set of meta-heuristics including constructive strategies and by modification, and carrying out an experimental study from the result of these implementations.

KEYWORDS. Test-Assignment. Metaheuristics. Optimization problems.

1. Introdução

A partir do momento que temos a aplicação de um conjunto de testes em um grupo de concorrentes queremos eliminar ao máximo a probabilidade de que possa ocorrer copia de respostas entre estes concorrentes pois isso estará afetando o resultado final dos melhores classificados, a partir disso que surge o problema de alocação de de testes. A probabilidade e facilidade de que se ocorra cola pode se dar a partir de alguns fatores, sendo eles por exemplo a proximidade entre as carteiras e o quão similares são as provas. Sendo assim, par cada par de carteiras pode ser, a partir desses fatores, definido um fator de probabilidade de cola, que é o produto da distância entre as carteiras e a similiaridade entre as provas aplicadas nestas.

Inicialmente podemos representar o problema usando grafos, cada vértice sendo uma carteira e cada aresta definindo que as carteiras representadas pelos vértices estão contectadas no problema, ou seja, devem ser consideradas no resultado da função objetivo. Os dados do problema então são: O conjuto de Testes T, o conjunto de carteiras D, a matriz de proximidades P e a matriz de simílaridades entre os testes Y, F o conjunto de carteiras que não possuem prova. Adicionalmente, como nem todas as carteiras encontram-se próximas o suficiente para serem consideradas, temos E como sendo o subconjunto de carteiras em D que estão próximas. O problema resume-se à:

$$\textbf{minimize} \quad \sum_{\{d,e\}\in E} p_{de} \sum_{t\in T}\sum_{u\in T} \gamma_{tu} x_{dt} x_{eu}, \qquad (1)$$

$$\textbf{subject to} \quad \sum_{t\in T} x_{dt} = 1, \qquad d \in D, \qquad (2)$$

$$\sum_{d\in D} x_{d0} = F, \qquad (3)$$

$$x_{dt} \in \{0,1\}, \qquad d \in D, t \in T. \qquad (4)$$

Este problema foi introduzido inicialmente por Duives [Jelle Duives, 2013] para melhorar a distribuição de testes aplicados na Universidade de Bologna. A partir de seu estudo inicial foi demonstrado que o problema pode ser classificado como NP-Difícil, e o formulou como um programa quadrático binário não convexo. Em 2018, Souza [Marcelo de Souza, 2018] tambem abordou este mesmo problema em seu estudo utilizando heurísticas recombinatórias, seus resultados serão utilizados neste trabalho como comparação

2. Estudo inicial

Inicialmente foram implementados cinco algorítmos para geração das soluções, dois deles sendo heurísticas construtivas, comparadas então com duas buscas locais simples, com critério de parada de primeira melhora e melhor melhora, e por fim tambem foi implementada uma caminhada aleatória para comparação. As abreviações para os algoritmos são respectivamente H-I, H-II, BSPM, BSMM e CA.

Instance	Inst. Name	Desks	Tests	Empty	H-I	H-II	BSPM	BSMM	CA	Best Known
1	lab1_2x0	20	3	0	20,9	25,79	20,9	20,9	20,9	20,9
2	lab1_2x5	20	3	5	8,6	10,95	8,6	8,6	8,6	5,58
3	lab1_2x10	20	3	10	2,2	2,15	2,2	2,2	2,2	1,22
4	lab1_3x10	20	4	10	1,42	1,92	1,65	1,42	1,45	0,93
5	lab1_3x0	20	4	0	15,66	22,15	15,63	15,56	15,52	11,95
6	lab6b_3x0	47	4	0	55,83	78,81	55,79	55,52	55,32	43,83
7	lab6b_3x20	47	4	20	10,47	12,13	10,47	10,47	10,4	6,38
8	lab6b_2x10	47	3	10	37,25	39,74	37,25	37,25	37,25	24,84
9	lab6b_2x20	47	3	20	15,05	16	15,05	15,05	15,05	8,52
10	lab6b_2x0	47	3	0	75,15	83,3	75,15	75,15	75,15	53,72
11	lab3_3x0	79	4	0	80,97	112,78	80,77	80,7	80,7	64,4
12	lab3_3x10	79	4	10	52,26	63,39	52,26	51,76	51,76	38
13	lab3_2x20	79	4	20	48,85	47,27	48,85	48,65	47,12	21,94
14	lab3_2x10	79	3	10	75,95	79,15	76,05	75,95	74,8	50,26
15	lab2_3x10	60	4	10	32,62	42,66	32,51	32,51	32,51	23,18
16	lab2_3x0	60	4	0	56,35	66,25	56,31	56,31	56,11	42,93
17	lab2_3x20	60	4	20	17,25	20,7	16,69	16,69	16,69	10,7
18	lab2_2x20	60	3	20	25,45	25,1	25,45	25,45	25,3	14,11
19	lab2_2x10	60	3	10	45,9	49	45,9	45,9	45,75	29,95
20	lab2_2x0	60	3	0	75,15	86,25	75,15	75,15	75,15	54,03
Avarage					37,66	44,27	37,63	37,56	37,39	26,37

A tabela exibe o resultado dos algoritmos citados para 20 instâncias, comparando com o melhor valor conhecido para a instância.

3. Implementação dos algoritmos escolhidos

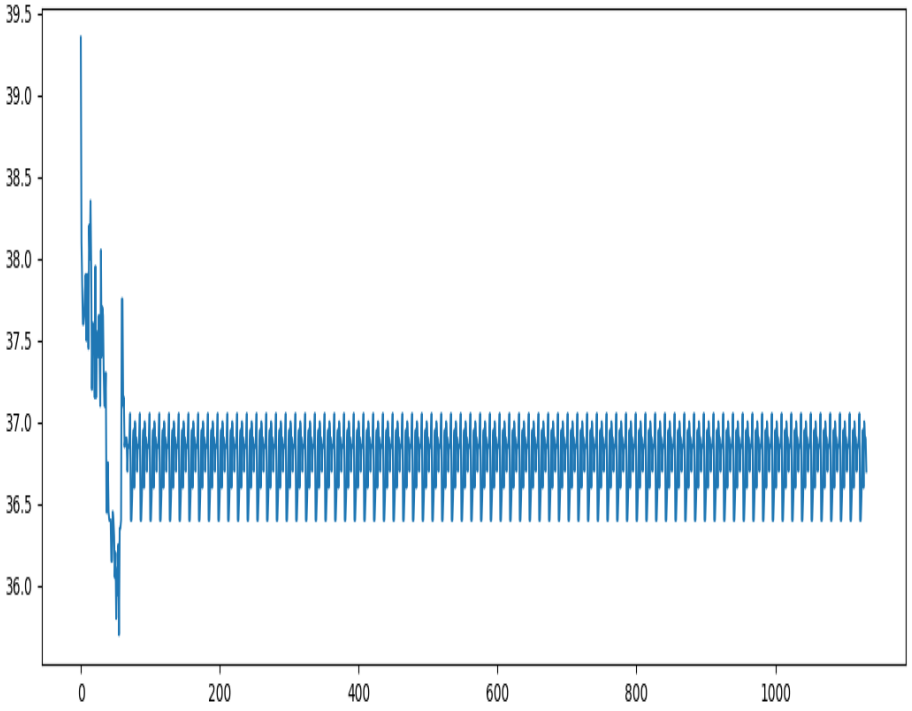
Para implementação foram selecionados os seguintes algoritmos: **Busca local Randomizada**, **Busca Tabu**, **Reinício aleatório**, **Construção repetida** e **Algoritmo semi-guloso (Guloso-K)**, os mesmos foram implementados de forma a permitir reutilização / combinação entre eles.

4. Mix de abordagens

Após a implentação das abordagens anteriores nesta seção foram criados novos algoritmos onde misturamos mais de uma abordagem a fim de buscar uma melhor combinação para resolver o problema. Um exemplo de onde esta mescla pode ser benéfica é na busca tabu visto que utiliza de uma solução aleatória para como seu estado incial e que caso essa solução inicial seja boa vinda de alguma outra abordagem a busca terá a aportunidade de melhorar ainda mais este resultado. Sendo assim criamos os seguintes algoritmos: **Auto-tabu (Tabu + controle automático da tenure e aplicação de perturbações)**, **Tabu + Guloso-K**, **Tabu + Construção repetida**, **Reinício aleatório + Guloso-K**, **Construção repetida + Guloso-K** e **Construção repetida + Perturbação**.

5. O Algoritmo Auto-tabu

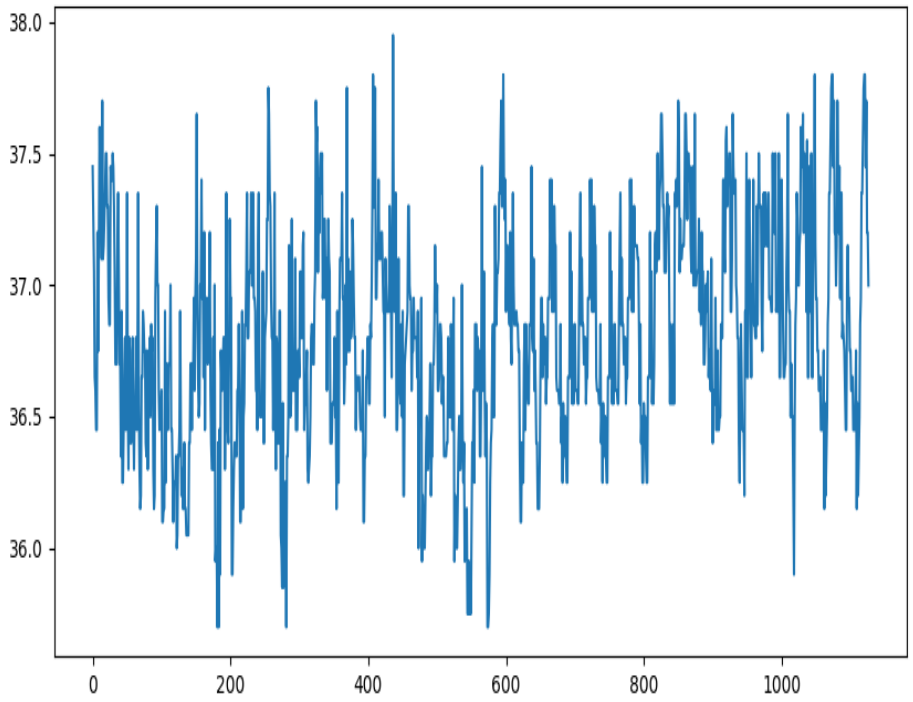
Durante o desenvolvimento dos algoritmos, em especial na busca Tabu, notamos que em algus casos o algoritmo acabava ciclando em alguma região da busca onde tinhamos que ficar alterando a tenure até encontrar um valor de iterações proibidas que fazia a execução sair do ciclo. O problema pode ser observado claramente no gráfico abaixo, onde é mostrada a evolução do valor da função objetivo (eixo y) pelo número de iterações (eixo x).



A solução para este problema foi inicialmente, criar uma modificação no funcionamento básico da busca tabu, permitindo um conjunto adicional de parametrizações alem da tenure, tais parametrizações são:

- **initialTernure**: Valor inicial da ”Tabu tenure”
- **tenureIncrement**: Incremento da Tenure a cada **cycleRange**.
- **cycleRange**: Ciclo (Em iterações) onde caso não hája melhora neste período, é incrementada a tenure em **tenureIncrement** unidades.
- **cutRange**: Altura de corte, caso a solução atual ultrapasse **cutRange**% de aumento em relação a melhor conhecida, a tenure é zerada, a solução retorna a melhor conhecida e é aplicada uma mutação aleatória na mesma.

Com esta abordagem, pode-se verificar uma melhora no resultado final obtido, e tambem a eliminação da cliclagem que ocorria, como se verifica no gráfico abaixo:



Após a criação dos algoritmos, foram feitos testes com cinco instâncias representativas, obtendo o desvio relativo em comparação ao melhor resultado conhecido, os dados foram os seguintes:

Algorithm	Instance 5	Instance 8	Instance 11	Instance 14	Instance 16	Avarage
TS+	0,3	0,46	0,24	0,47	0,29	0,352
TS + Greedy-K	0,27	0,54	0,27	0,44	0,3	0,364
TS + RC	0,26	0,44	0,27	0,43	0,29	**0,338**
RR + Greedy-K	0,29	0,49	0,28	0,45	0,3	0,362
RC + Greedy-K	0,29	0,46	0,27	0,45	0,3	0,354
RC + Perturbation	0,28	0,46	0,27	0,45	0,3	0,352

Verificando os resultados, chegamos a conclusão que o melhor algoritmo foi o que combina Busca Tabu com a construção repetida, obtendo um desvio padrão médio de 0,338.

6. Testes finais

Após termos definido qual o melhor algoritmo criado, fizemos uma bateria de testes com este algoritmo em todas as 20 instâncias iniciais, e obtemos os seguintes resultados:

Instance	Inst. Name	Desks	Tests	Empty	TS-RC (value)	Best Known	RD
1	lab1_2x0	20	3	0	20,9	20,9	0
2	lab1_2x5	20	3	5	8,1	5,58	0,45
3	lab1_2x10	20	3	10	1,84	1,22	0,51
4	lab1_3x10	20	4	10	1,32	0,93	0,41
5	lab1_3x0	20	4	0	15,15	11,95	0,26
6	lab6b_3x0	47	4	0	54,39	43,83	0,24
7	lab6b_3x20	47	4	20	9,04	6,38	0,41
8	lab6b_2x10	47	3	10	35,94	24,84	0,44
9	lab6b_2x20	47	3	20	13,25	8,52	0,55
10	lab6b_2x0	47	3	0	73,24	53,72	0,36
11	lab3_3x0	79	4	0	82,1	64,4	0,27
12	lab3_3x10	79	4	10	51,86	38	0,36
13	lab3_2x20	79	4	20	44,17	21,94	1,01
14	lab3_2x10	79	3	10	72.1	50,26	0,43
15	lab2_3x10	60	4	10	31,4	23,18	0,35
16	lab2_3x0	60	4	0	55,42	42,93	0,29
17	lab2_3x20	60	4	20	15,66	10,7	0,46
18	lab2_2x20	60	3	20	22,1	14,11	0,56
19	lab2_2x10	60	3	10	44	29,95	0,46
20	lab2_2x0	60	3	0	74,24	54,03	0,37
Avarage					**34,43**	**26,37**	**0,41**

7. Conclusão

Apesar de termos bons resultados durante o desenvolvimento dos algorimos híbridos se comparados aos desenvolvidos inicialmente, as soluções obtidas ainda estão um pouco longe das melhores conhecidas, já que atingimos o melhor valor conhecido em apenas uma das 20 instâncias selecionadas (lab1_2x0). Verificamos que para o problema de alocação de testes, a facilidade com que se aproxima das melhores soluções varia muito de instância para instância, tendo que em alguns casos, criar adaptações ou configurações para explorar regiões diferentes da topologia de busca.

Referências

Jelle Duives, E. M., Andrea Lodi (2013). Test-assignment: a quadratic coloring problem. *Journal of Heuristics*, 19:5–44.

Marcelo de Souza, M. R. (2018). An automatically designed recombination heuristic for the test-assignment problem. Acessado: 2022-07-19.