XV SIMPÓSIO Programas de Mestrado Profissional 11 a 12 de novembro de 2020

Desafios de uma sociedade digital nos Sistemas Produtivos e na Educação

Centro Paula Souza

GOVERNO DO ESTADO DE SÃO PAULO

Otimização de Grades Horárias com Algoritmos Genéticos

Cassiano Henrique Martini[1], Adaní Cusin Sacilotti [2]

Resumo - A Inteligência Artificial (IA) tem conquistado cada vez mais relevância na solução de problemas da sociedade. Um destes problemas com potencial para sua aplicação é o da otimização de Grades Horárias por meio do uso de Algoritmos Genéticos, um desdobramento do campo da IA. O objetivo desta pesquisa foi realizar um estudo bibliográfico sobre Algoritmos Genéticos e aplicar as informações levantadas no desenvolvimento de um modelo que otimizasse as soluções de um caso real. Ao final da pesquisa foi observado que a utilização de Algoritmos Genéticos possui implementação simples, porém requer cuidados, e se aplicados a problemas específicos podem gerar soluções que otimizem o aproveitamento de recursos nos mais variados contextos.

Palavras-chave: Inteligência Artificial, Algoritmos Genéticos, Grades Horárias.

Abstract - Artificial Intelligence has gained more and more relevance in solving society's problems. One of these problems with potential for its application is the optimization of Timetables using Genetic Algorithms, an area of the AI field. The objective of this research was to carry out a bibliographic study on Genetic Algorithms and apply the information collected to the development of a model that optimized the solutions of a real case. At the end of the research it can be observed that the use of Genetic Algorithms has a simple implementation, but requires care, and if applied to specific problems, they can generate solutions that optimize the use of resources in the most varied contexts.

Keywords: Artificial Intelligence, Genetic Algorithms, Timetables.

1. Introdução

[1] Fatec Jundiaí – cassianomartini@hotmail.com
[2] Fatec Jundiaí – adani.sacilotti@fatec.sp.gov.br

A Inteligência Artificial, ramo da ciência da computação, tem conquistado cada vez mais relevância na solução de problemas da sociedade. Os avanços tecnológicos que seu estudo e exploração tem trazido, nos remete a uma nova revolução para a forma como nos relacionamos com a informação e lidamos com problemas complexos.

Um destes exemplos, com potencial para aplicação da Inteligência Artificial está na otimização de Grades Horárias, problema conhecido como Turma-Professor em que indivíduos de um mesmo ambiente precisam compartilhar recursos limitados por meio da divisão de seu uso entre horários e espaços específicos, como proposto por PASSOS et al. (2014).

A utilização de Algoritmos Genéticos (HOLLAND, 1975), recurso de uma das áreas da Computação Evolucionária ligada à Inteligência Artificial, tem se mostrado promissora para lidar com problemas do tipo Turma-Professor, tido como NP-Completo (RUSSEL; NORVIG, 2013) e, demonstra, segundo bibliografia relacionada, um bom potencial para minimizar os recorrentes conflitos de otimização combinatória, típicos da construção destas grades.

Utilizando a linguagem de programação Python e uma ferramenta de código aberto para computação científica, foi construído um Modelo de Algoritmo Genético (LINDEN, 2012) seguindo os métodos encontrados nas pesquisas de documentações realizadas com objetivo de ser aplicado a um problema de construção de Grade Horária em uma escola de período integral da cidade de Louveira, onde vinte e uma turmas de ensino fundamental necessitavam compartilhar recursos limitados entre si de acordo com um cronograma.

Ao final da pesquisa, uma visão mais abrangente sobre Algoritmos Genéticos pode ser fundamentada e um maior conhecimento sobre classes de problemas, incluído o problema Turma-Professor, pode ser alcançado, contribuindo não somente com novas formas de se pensar a organização e divisão de recursos em espaços acadêmicos, mas com uma visão ampliada a todos os ambientes em que gestores e coordenadores necessitavam aprimorar o aproveitamento de recursos e reduzir custos.

2. Referencial Teórico

Os Algoritmos Genéticos (HOLLAND, 1975), assim como a Programação Genética (FOGEL, 1965) e as Estratégias Evolucionárias (RECHENBERG, 1965) correspondem à área da Computação Evolucionária, ramo da Inteligência Artificial que se baseia na Teoria da Evolução das Espécies proposta por Charles Darwin (1859).

Essa teoria propõe o método de seleção natural, no qual as características dos seres vivos são passadas a seus descendentes por meio da hereditariedade genética. Indivíduos mais bem adaptados ao ambiente em que vivem conseguem se reproduzir e passar seus genes para uma próxima geração, enquanto indivíduos menos adaptados morrem sem passar suas características adiante (DARWIN, 1859).

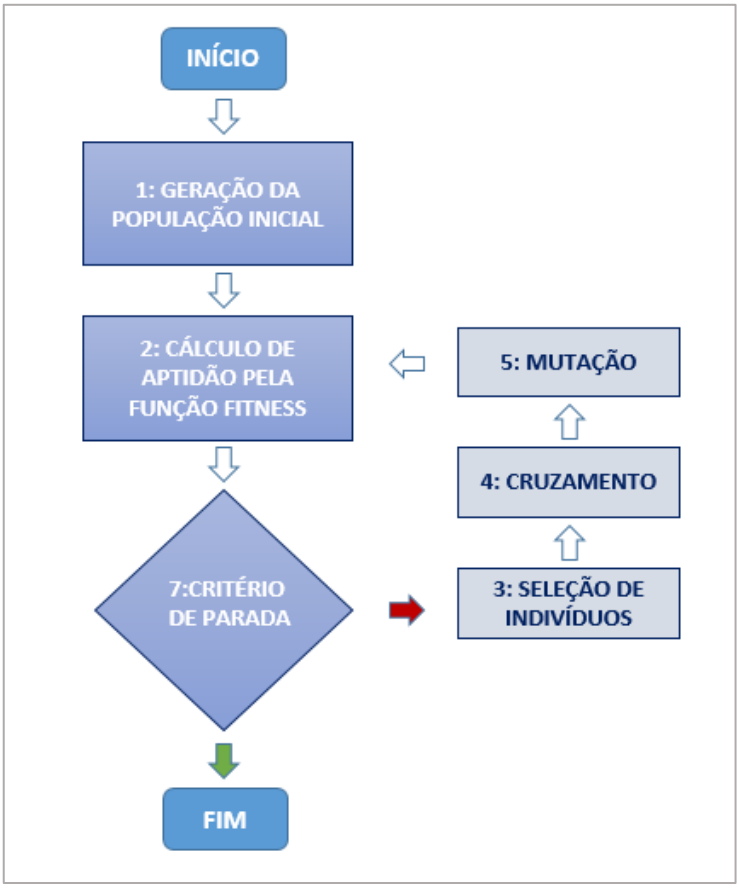
Em Algoritmos Genéticos (HOLLAND, 1975), faz-se um paralelo com a Teoria da Evolução de Darwin (DARWIN, 1859), em que um indivíduo corresponde a uma solução possível para um determinado problema (UNIVERSIDADE DE SÃO PAULO, 2009). Uma população representa vários indivíduos ou as várias soluções possíveis para um problema. Os genes ou cromossomos são os espaços mutáveis

dentro destes indivíduos e os alelos correspondem aos valores inseridos nesses genes. O genoma, por sua vez, corresponde ao conjunto de genes de um mesmo indivíduo (LINDEN, 2012).

De acordo com Ziviani (2007), “um algoritmo pode ser visto como uma sequência de ações executáveis para a obtenção de uma solução para determinado tipo de problema”. Em contrapartida, os Algoritmos Genéticos, propostos inicialmente por John Holland (1975), são classificados na computação como métodos heurísticos para otimização de problemas de classe NP-Completos. Tais métodos não buscam a solução ótima para um problema como algoritmos tradicionais, mas sim, uma solução aproximada para problemas classificados como intratáveis em tempo polinomial (ZIVIANI, 2007).

No estudo de algoritmos, a complexidade de problemas é dividida em classes para estimar a sua capacidade de resolução. Um problema é considerado intratável em “tempo polinomial”, por exemplo, quando seu tempo de execução e quantidade de etapas necessárias para encontrar uma solução crescem exponencialmente em cada etapa (ZIVIANI, 2007). Problemas Turma-Professor são conhecidos por pertencer a classe de problemas NP-Completos, o que significa dizer que soluções são possíveis, porém, não há garantias de que soluções ótimas serão encontradas por algum algoritmo conhecido, tornando a aproximação de resultados uma melhor resposta a eles. A Figura 1 apresenta um fluxograma de um algoritmo genético padrão.

Figura 1 – Fluxograma de um algoritmo genético padrão.



Fonte: Elaborado pelos autores.

Segundo Ricardo Linden (2012), um algoritmo genético segue as seguintes instruções: 1) Gerar uma população composta por indivíduos com características aleatórias; 2) Avaliar os indivíduos gerados quanto à sua capacidade em prover boas soluções ao problema por meio de uma função conhecida como Função Fitness; 3) Selecionar os indivíduos melhor pontuados pela Função Fitness; 4) Cruzar os genes dos indivíduos selecionados para gerar novos indivíduos em um processo denominado Crossover; 5) Executar a mutação de alguns alelos de uma parcela da nova população criada; 6) Retornar os novos indivíduos à Função Fitness, estabelecendo uma estrutura de repetição até que: 7) Um critério de parada seja atingido. Essa estrutura, também é descrita em trabalhos apontados por BARSANULFO et al. (2016) e PASSOS et al. (2014).

3. Método

Esta pesquisa, de base bibliográfica, buscou levantar informações pertinentes à utilização de Algoritmos Genéticos e do Problema Turma-Professor propondo, aplicar as informações levantadas em um estudo de caso real passível de sua utilização.

Partindo do estudo de caso, foi realizada uma entrevista com diretores de uma escola de ensino fundamental de período integral localizada no município de Louveira/SP a fim de caracterizar o problema Turma-Professor relatado no ambiente por meio do levantamento de requisitos (PRESSMAN, 2016) necessários para a criação de uma grade de horários de atividades escolares otimizada com Algoritmos Genéticos.

3.1. Estudo de Caso: Grade Horária

Foram levantados os seguintes requisitos para modelagem de uma grade horária satisfatória, desprezando-se requisitos secundários menos importantes, buscando-se uma melhora objetiva aos problemas: 1) Turmas devem fazer apenas um horário de aula de especialidade, sendo aula de Inglês, Informática, Educação Física e Arte dentro do período de uma semana; 2) Turmas realizam no máximo duas atividades especiais no mesmo dia; 3) Todas as turmas devem fazer aulas normais quando não estiverem em atividades especiais; 4) As turmas não devem compartilhar atividade em um mesmo horário.

Como resultado da entrevista, três problemas principais foram identificados a respeito da criação de Grades Horárias nesta escola e que também estão presentes nos principais problemas de otimização combinatória do tipo NP-Completos: 1º - A criação de Grades Horárias é um trabalho que demanda muito tempo, em alguns casos, semanas ou meses para ser concluída. 2º - Muitas vezes as grades criadas não dão conta de contemplar todas as necessidades das turmas, sendo necessário uma reorganização de toda a grade e, frequentemente, uma mudança da atividade de uma turma acaba por modificar também a de outra turma. 3º - A demora por um arranjo adequado de atividades na grade escolar acaba por desperdiçar um tempo precioso do ano letivo, gerando sobrecarga em professores e coordenadores, bem como baixo aproveitamento de atividades por parte dos alunos.

Caracterizado o problema Turma-Professor, foi feita a tentativa de modelagem de um indivíduo utilizando a linguagem de programação Python, para dar os primeiros passos na criação de um Algoritmo Genético, com o qual fosse possível relacionar um indivíduo às atribuições de turmas e aulas de especialidades que constituiria o modelo de solução para o problema. A linguagem Python foi escolhida por ser uma das mais eficazes voltadas à manipulação de grandes volumes de dados disponíveis e utilizadas atualmente.

A melhor representação encontrada se deu por uma matriz de 50 linhas por 84 colunas. Todas as atividades, como aulas normais ou especialidades, possuem 50 minutos de duração cada, mas, levando-se em conta as trocas entre salas e o deslocamento de alunos ou professores, achou-se melhor definir como unidade de tempo, o período de 1 hora.

Um dia escolar de período integral possui 10 horas. Cada linha da matriz elaborada representa um horário único durante a semana, sendo a primeira linha referente à primeira hora da segunda-feira e a última linha representando a última hora da sexta-feira. As colunas representam o binômio turma/especialidade.

Observando-se que existem 21 turmas e fora definido apenas 4 aulas de especialidades para essas turmas, encontrou-se um total de 84 colunas representando todas as aulas possíveis para todas as turmas dispostas em uma grade, na qual um valor qualquer preenchido em algum ponto desta matriz representaria uma coordenada de atribuição para a turma, especialidade e horário. Foi inicializado no Algoritmo Genético uma população de 20 indivíduos com valores aleatórios entre 0 e 1 para seus genes, cada espaço desta matriz representa uma atribuição de aula, caso o valor seja igual a 1.

Na sequência, foi criada a Função Fitness responsável por avaliar a aptidão de cada indivíduo gerado. Como critério de avaliação dos indivíduos cada turma da escola teria apenas 4 aulas de especialidades distintas no decorrer da semana.

Definiu-se também que cada turma não poderia ter mais do que duas aulas de especialidades em um único dia da semana. A Função Fitness definida avalia se cada coluna da matriz que compõe um indivíduo contém apenas um valor igual a 1, por representar uma atribuição de determinado binômio a uma hora específica da semana.

Em seguida, a Função Fitness avalia as linhas da matriz em blocos de 21 genes, cada bloco podendo conter apenas um valor igual a 1. Todos os indivíduos de uma mesma geração são submetidos à avaliação da Função Fitness e, para cada um, é calculado um percentual de acerto das linhas e colunas.

Por último foi calculado o percentual geral de acertos obtidos pelo genoma do indivíduo que recebe uma nota de aptidão para solução do problema proposto que terá impacto na etapa de seleção seguinte. A Figura 2 demonstra a codificação desenvolvida para a Função Fitness.

Foram definidos dois meios de seleção de indivíduos: roleta 1 e roleta 2. Na roleta 1, um *array* de caracteres teve seus espaços divididos e preenchidos de acordo com o peso direto de cada indivíduo de uma mesma geração. Quanto maior a aptidão de um indivíduo, segundo retorno da Função Fitness, maior o espaço ocupado por ele neste *array*. Na sequência, efetua-se um sorteio dentro deste *array* e os indivíduos que o compõem em maior quantidade acabam por serem selecionados mais facilmente para dar origem a uma próxima geração de indivíduos, mais aptos a gerar soluções melhores ao problema.

Essa solução se mostrou muito eficaz quando há uma amostragem de indivíduos com pontuações de avaliação muito distantes entre si, porém, acaba por

gerar máximas locais, ou seja, o afunilamento de soluções que não são as melhores para o problema, fazendo com que novas populações convirjam para a estagnação dos indivíduos em soluções com baixa eficácia.

Figura 2 - Código da função fitness.

```
    #Calcula média de pontos por coluna
    cont = 0 #contador de linhas
    notas_coluna = 0.0

    for n in range(83):
        for m in range(49):
            if matriz[m][n] != 1:
                cont=cont+1
        #calcula a média da coluna e guarda
        if cont==50:
            notas_coluna=notas_coluna+98.0
        else:
            notas_coluna=notas_coluna+((cont*100)/49)
        cont=0
    notas_coluna=notas_coluna/84

    #Média de pontos por linha
    bloco=0
    media_bloco=0.0

    for m in range(49):
        for n in range(83):
            if matriz[m][n] != 1:
                cont=cont+1
            bloco=bloco+1
            if bloco == 21:
                if cont==21:
                    media_bloco=media_bloco+95.0
                else:
                    media_bloco=media_bloco+((cont*100)/20)
                bloco=0
                cont=0
    media_bloco=media_bloco/(50*4)
    aptidao = (media_bloco+notas_coluna)/2
    return(aptidao)
```

Fonte: Elaborado pelos autores.

Para resolver o problema, por meio do cálculo do desvio padrão da amostragem de pontuações de uma geração presente, uma segunda roleta entra em ação caso este desvio torne-se expressivo na amostragem.

A roleta 2 ranqueia os indivíduos de uma geração e atribui a cada ranqueado um peso pré-definido dentro de um *array* de 210 caracteres, tendo peso 1 para o último ranqueado e peso 20 para o melhor colocado. A fim de impulsionar a seleção dos mais aptos, a roleta 2 prioriza ainda os 3 melhores colocados do ranque gerado. Na sequência, efetua-se aleatoriamente um sorteio dentro desse *array*, da mesma forma da roleta 1, selecionando assim os indivíduos mais abundantes e melhores, ainda que muito semelhantes em aptidão.

Na sequência, ocorre um processo conhecido como Crossover. Cada indivíduo selecionado tem seu genoma dividido na décima quinta linha da matriz. A primeira parte desta cisão é combinada com a segunda, resultante da divisão de algum outro indivíduo. Essa combinação dará origem a um novo indivíduo que será incluso na geração seguinte.

Este processo ocorre para toda a população de indivíduos, gerando assim uma nova população com o mesmo número de indivíduos da primeira geração. Optou-se por utilizar um mecanismo conhecido como elitismo na modelagem do algoritmo para que as melhores soluções encontradas fossem mantidas nas gerações seguintes, otimizando assim as próximas combinações genéticas.

Foi definido na codificação que apenas 30% da população sofreria alguma mutação de seus genes. Para isso, definiu-se uma taxa de mutação que incidirá na

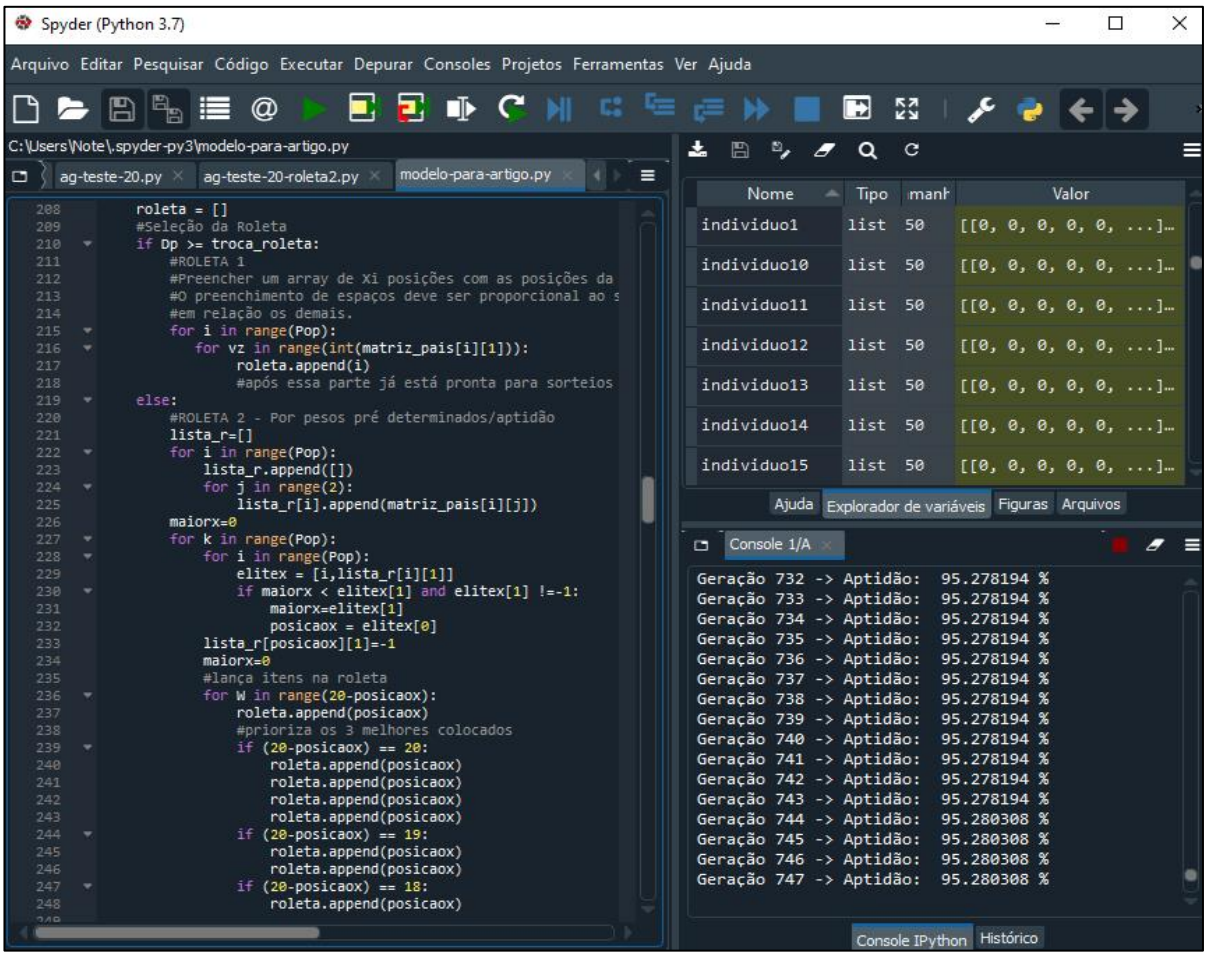
troca genética desses 30%. Nesse processo, alguns genes são selecionados e o alelo de seus genes trocado aleatoriamente, dando origem, assim, a novas características de uma parte da população seguinte.

O processo de avaliação de indivíduos – seleção, cruzamento e mutação – ocorre até que um critério de parada seja alcançado. Definiu-se que o critério de parada ocorreria de duas formas: por número de gerações produzidas ou quando algum dos indivíduos atingisse um percentual esperado de satisfação para a solução do problema Turma-Professor.

O código foi escrito de modo a permitir que vários ajustes pudessem ser feitos antes de sua execução, como proposto por SILVA et al. (2014), como controle da taxa de mutação de indivíduos, da taxa de atribuição de alelos, do número de gerações mínima e da taxa de sucesso dos indivíduo. Fora incluído ainda o controle do desvio padrão da amostragem para operar a troca entre roletas de seleção de indivíduos.

Foi utilizado para realizar os testes o software IDE de código aberto Spyder, destinado a programação científica com o uso da linguagem Python em sua versão 3.7. Na Figura 3 é possível visualizar o IDE Spyder executando o algoritmo genético criado para o problema Turma-Professor do caso estudado, tendo à esquerda da imagem um exemplo das roletas utilizadas pra seleção de indivíduos e no canto inferior direito, as amostragens de aptidão por geração.

Figura 3 – Programa Spyder executando o algoritmo genético desenvolvido.



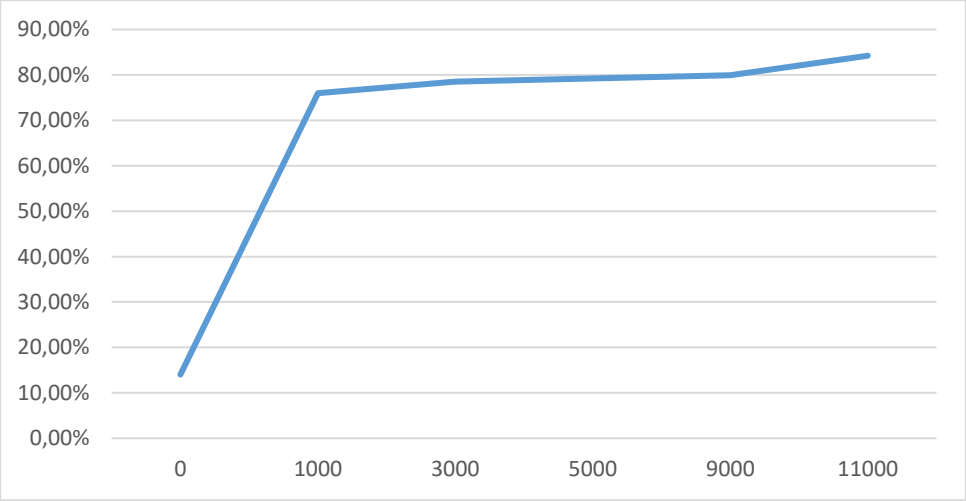
Fonte: Elaborado pelos autores.

4. Resultados e Discussão

Nas simulações realizadas, duas situações tiveram maior destaque quanto a evolução de seus indivíduos. A primeira ocorreu ao inicializar os genes de 20 indivíduos em uma faixa máxima de aleatoriedade de preenchimento de genes, entre 0 e 4200 alelos, com taxa de mutação de 1,5% para 30% da população e troca de meio de seleção dos mais aptos, da roleta 1 para roleta 2, quando o desvio padrão da amostragem de aptidão de uma geração fosse menor ou igual a 0,5.

Entre as gerações 0 e 1000 houve um crescimento de 54,11% de aptidão das soluções, saltando da primeira geração gerada aleatoriamente com 14,03% de aptidão para indivíduos com 68,14% de aptidão de solução para o problema Turma-Professor estudado. Já nas gerações entre 1000 e 3000 pode ser verificado um crescimento mais discreto, de apenas 7,87%. Entre 3000 e 11000 gerações, o crescimento foi de apenas 5,2%, totalizando uma aptidão de 81,21% na solução do problema como mostra o Gráfico 1 a seguir:

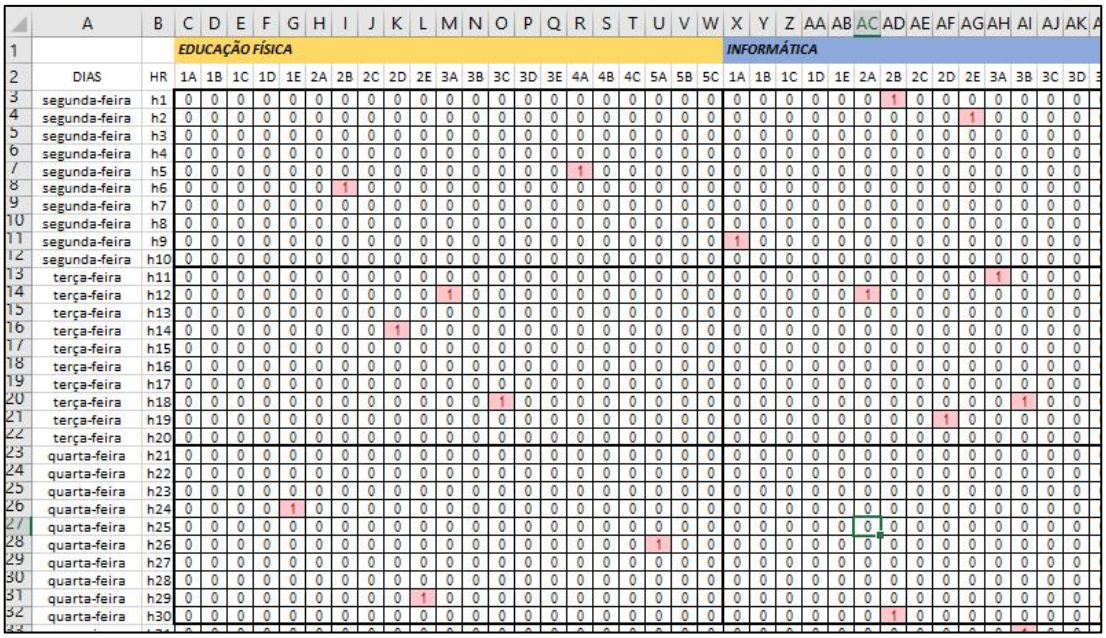
Gráfico 1: Evolução da aptidão em 81,21%.



Fonte: Elaborado pelos autores.

Em um segundo teste, mudando apenas a taxa de preenchimento aleatório dos genes para uma proporção mais próxima do que se estimou ser ideal ao problema estudado, entre 1 e 8 genes preenchidos aleatoriamente em cada linha da matriz que compõe o indivíduo, obteve-se logo na primeira geração um percentual de aptidão de 95,24%. Após 30000 gerações e o período de uma hora decorrida, o percentual de evolução havia crescido discretamente apenas 0,05%, retornando ao final dos testes uma aptidão de 95,29%. Mediante o resultado satisfatório da aproximação da solução do problema, os resultados foram exportados para uma planilha eletrônica para tornar gráfica a representação das atribuições de aulas. Parte desta planilha pode ser vista na Figura 4 a seguir:

Figura 4: Plotagem de parte da solução proposta pelo algoritmo.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI	AJ	AK
1			EDUCAÇÃO FÍSICA																					INFORMÁTICA														
2	DIAS	HR	1A	1B	1C	1D	1E	2A	2B	2C	2D	2E	3A	3B	3C	3D	3E	4A	4B	4C	5A	5B	5C	1A	1B	1C	1D	1E	2A	2B	2C	2D	2E	3A	3B	3C	3D
3	segunda-feira	h1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
4	segunda-feira	h2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
5	segunda-feira	h3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	segunda-feira	h4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	segunda-feira	h5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	segunda-feira	h6	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	segunda-feira	h7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	segunda-feira	h8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	segunda-feira	h9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
12	segunda-feira	h10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	terça-feira	h11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
14	terça-feira	h12	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
15	terça-feira	h13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	terça-feira	h14	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	terça-feira	h15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	terça-feira	h16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	terça-feira	h17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	terça-feira	h18	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
21	terça-feira	h19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
22	terça-feira	h20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
23	quarta-feira	h21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	quarta-feira	h22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	quarta-feira	h23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	quarta-feira	h24	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	quarta-feira	h25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	quarta-feira	h26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
29	quarta-feira	h27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	quarta-feira	h28	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	quarta-feira	h29	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	quarta-feira	h30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Fonte: Elaborado pelos autores.

5. Considerações finais

Constatou-se que a modelagem da solução em forma de indivíduo constituiu o grande desafio na implantação de Algoritmos Genéticos, que apesar de possuírem um fluxograma de execução bastante simples, oferecem razoável dificuldade em sua concepção.

A busca pela codificação de um modelo-solução para o problema Turma-Professor, proposto no estudo de caso, evidenciou a necessidade dos embasamentos documentais referentes à sua construção, bem como um dedicado processo de levantamento de requisitos para a sua codificação. Pode ser observado que somente mediante a concepção de um modelo de indivíduo adequado foi possível desenvolver corretamente as demais partes do algoritmo.

Observou-se ainda que a utilização da linguagem Python aliada ao software Spyder tornou muito simples o processo de codificar o algoritmo e que, tanto a linguagem quanto o ambiente de desenvolvimento, oferecem considerável robustez e velocidade e, ainda, que programadores experientes na linguagem possam construir soluções mais elegantes, utilizando recursos avançados próprios desta linguagem, se aplicados a Algoritmos Genéticos.

A utilização de Algoritmos Genéticos proporcionou uma aproximação de soluções ótimas e demonstrou bom potencial para ser associada a área de planejamento estratégico de organizações, em Sistemas de Apoio a Decisão, Sistemas de Informação Executivos e Business Intelligence, por contemplar a habilidade de gerar previsões e estimativas acerca de situações complexas ou não estruturadas inerentes a estas áreas. O estudo do problema Turma-Professor e sua classe de problema NP-Completo, proporcionou também o contato com problemas similares de outras áreas, fora do ambiente escolar, trazendo uma forte perspectiva de utilização dos Algoritmos Genéticos a áreas como Logística, Transportes, Finanças e Engenharia Civil.

Tornou-se evidente, mediante a análise da solução proposta pelo algoritmo no estudo de caso que, apesar de não gerar soluções perfeitas, suas soluções contemplaram uma taxa de 95% de sucesso, podendo servir de guia para gestores e coordenadores na composição ou mesmo na atualização rápida de cronogramas e grades de horários.

Referências

BARSANULFO, A., MATTIOLI, F., CAMPOS, R. D., SAAD, E. F., ANDRADE, R. B. *Escalonamento de Horários Acadêmicos Utilizando Algoritmos Genéticos*. FACTHUS – Jornal de Engenharia, Tecnologia e Meio Ambiente., v. 1, n. 1, p. 27-31. 2016.

DARWIN, Charles. *On The Origin of Species by Means of Natural Selection*. London: J. Murray, 1859.

FOGEL, Lawrence J.; OWENS, Alvin J.; WALSH, Michael J. *Artificial Intelligence Through Simulated Evolution*. New York: John Wiley & Sons, 1966. 170 p.

HOLLAND, J. H. *Adaption in Natural and Artificial Systems*. The MIT Press: Cambridge, 1975. 228 p.

LINDEN, Ricardo. *Algoritmos Genéticos*. Rio de Janeiro: Ciência Moderna; 3ª Edição, 2012. 496 p.

PASSOS, A. R. F., DIAZ, E. Y. V., TAVERA, M. J. M., JEQUENESSE, P. M., MACHADO, M. N. *Uma Aplicação De Algoritmos Genéticos ao Problema de Timetabling*. Universidade Federal Fluminense – ENGEVISTA, v. 19, n. 4, p. 862-880. 2017.

PRESSMAN, Roger S.; MAXIM, Bruce R. *Engenharia de Software: uma abordagem profissional.* 8ª ed. São Paulo: McGraw-Hill, 2016. 968 p.

RECHENBERG, Inge. *Cybernetic solution path of an experimental problem.* Ministry of Aviation, Farnborough - Library Translation no. 1122, 1965.

RUSSEL, Stuart; NORVIG, Peter. *Inteligência Artificial.* 3ª ed. São Paulo, GEN LTC, 2013. 1016 p.

SILVA, A. B., BETEMPS, C. M., HEINEN, M. *Algoritmos Genéticos na obtenção de uma Grade de Horários com Múltiplos Cursos para uma Instituição de Ensino*. Encontro Anual de Tecnologia da Informação e Semana Acadêmica de Tecnologia da Informação., v. 4, n. 1, p. 239-246. 2014.

UNIVERSIDADE DE SÃO PAULO. Instituto de Ciências Matemática e da Computação. *Algoritmos Genéticos*. São Carlos, 2009. Disponível em: <https://sites.icmc.usp.br/andre/research/genetic>. Acesso em: 19 ago. 2019.

ZIVIANI, Nivio. Projeto de Algoritmos: com implementação em Java e C++. São Paulo: Cengage Learning, 2011. 621 p.