Algoritmo genético para construção de horários

Acácia dos Campos da Terra[1], **João Pedro Winckler Bernardi**[1]

[1]Universidade Federal da Fronteira Sul (UFFS)
Caixa Postal 181 – 89802-112 – Chapecó – SC – Brasil

terra.acacia@gmail.com, jpwbernardi@hotmail.com

1. Introdução

Determinar a grade de horários do semestre de uma universidade pode parecer muito simples, mas dadas as restrições dos professores, sejam elas legais ou simplesmente suas preferências de horários, essa tarefa se mostra realmente complexa. Neste relatório, apresentamos o desenvolvimento de uma solução dada por um programa onde, dada a entrada com as informações de um curso, é utilizado um algoritmo genético para encontrar o horário que melhor se encaixe dentro das restrições dadas. O algoritmo genético funciona da seguinte forma: há uma população, onde são selecionados os melhores indivíduos nela para reproduzirem, os quais irão formar uma nova população, o processo se repete até que seja encontrado o indivíduo ideal ou que o tempo de execução do programa termine. Como em qualquer reprodução, há riscos de haver mutação num novo indivíduo. No caso deste algoritmo, o indivíduo ideal é aquele com a melhor grade de horários possível.

O trabalho foi proposto na disciplina de Inteligência Artificial, com o objetivo de colocar em prática os estudos feitos sobre algoritmos genéticos. O problema dado (gerar a grade de horários do curso) é composto por professores e suas preferências de horários, por disciplinas e seus respectivos semestres e informações como horários de aula, salas de aula e códigos das disciplinas. Sendo assim, é exigida uma entrada para o programa que contenha:

- Número N de professores, seguido de N nomes de professores, estando cada professor associado a um número M de horários, seguido de M horários os quais o professor prefere não ministrar aulas
- Número N de semestres, seguido de N códigos de semestres, estando cada semestre associado a um o número de sala K e a um número M de quantidade de horários, seguido de M horários disponíveis
- Número N de disciplinas , seguido de N linhas contendo o código da disciplina, o número de períodos a serem ocupados por aquela disciplina, o código do semestre e o nome do professor que irá ministrar a aula

Após passar pelo algoritmo elaborado, o programa irá retornar um arquivo *csv* contendo a grade com os horários já montada, de forma a atender da melhor forma possível as restrições legais (onde o mesmo professor não pode ministrar aula no último horário da noite e novamente no primeiro horário da manhã do dia seguinte, por exemplo) e as restrições pessoais (as preferências dos professores).

2. Algoritmo Genético Adaptado

O algoritmo genético precisa ser adaptado para a solução do problema proposto, então o primeiro passo para adaptá-lo é determinar o que é um indivíduo, ou melhor, o que o

indivíduo possui(seus genes). Os genes de um individuo são: um conjunto das matérias que já constam em seu horário, um vetor de matrizes, onde cada matriz representa a grade de uma turma diferente do curso, um inteiro que conta quantas falhas há na sua grade e outro inteiro indicando se o individuo está vivo. Entende-se por falha quando um professor é determinado a ministrar aula num horário que ele não gostaria, e por vivo quando não há nenhuma falha que infrinja alguma restrição legal. A grade de uma turma é representada conforme a imagem abaixo:

	SEG	TER	QUA	QUI	SEX
MAT	0	1	2	3	4
	5	6	7	8	9
VESP	10	11	12	13	14
	15	16	17	18	19
NOT	20	21	22	23	24
	25	26	27	28	29

Figura 1. Grade de turmas

O Algoritmo 1 mostra como o processo ocorre. Seja P o conjunto correspondente a população, enquanto não houver extrapolado o tempo máximo definido, tomamos $|P|$ pares de indivíduos da população para criar uma nova população, todos descendentes das populações passadas. Quando é obtido o indivíduo ideal, o mesmo é devolvido. Inicialmente, todos os indivíduos não possuem nenhuma matéria e tem as suas grades vazias.

Na reprodução, sejam a e b os indivíduos envolvidos e, sem perda de generalidade, assuma que a possui menos erros, logo é o mais propício para perpetuar seus genes. Assim, o indivíduo gerado possuirá a mesma grade de a, porém terá as matérias que estão em b e não estão em a adicionadas. Em todo novo indivíduo gerado, existe uma chance de ocorrer uma mutação, que é responsável por adicionar uma matéria aleatoriamente no novo indivíduo.

2.1. Heurística

Uma heurística é considerada boa quando se aproxima melhor do resultado esperado. Como o desejado é que o horário respeite as restrições legais, as únicas falhas nos genes que se permite passar é quando o professor tem algum tipo de restrição no horário que foi designado a dar aula. Ao adicionar uma matéria na grade de um indivíduo, a primeira tentativa é inserir a matéria em um dos horários os quais o professor tem preferência, mas em caso de não ser possível atender a essa restrição, a matéria é colocada no primeiro horário vago da turma e é incrementado seu contador de erros. Pode ocorrer de, mesmo assim, não ser possível inseri-la naquele horário, e neste caso, o indivíduo é morto.

É com a seleção aleatória que é evitado que esses indivíduos perpetuem seus genes. Para a seleção, é utilizado o mesmo princípio que o escalonamento de processos por

Algoritmo 1: GENÉTICO

Entrada: $P : populacao$
Saída: Individuo ideal

enquanto $tempoDecorrido < tempoLimite$ **faça**
  $nP = \emptyset$
  para *i de 0 a* $|P|$ **faça**
    $a \leftarrow selecaoAleatoria(P)$
    $b \leftarrow selecaoAleatoria(P)$
    $z \leftarrow cruzamento(a, b)$
    $z \leftarrow mutacao(z)$
    se *Erros de z aceitáveis* ***E*** *z tem todas as matérias* **então**
      retorna z
    fim
    $nP \leftarrow nP \cup z$
  fim
  $P \leftarrow nP$
fim
retorna $melhorIndividuo(P)$

loteria, onde um indivíduo é escolhido através de um sorteio. As chances de um indivíduo ser sorteado se dão de acordo com a quantidade de erros que eles tem: quantos menos erros, maior é a chance de ser o escolhido. Os indivíduos mortos não são considerados para o sorteio e, quando identificados, eles passam por uma tentativa de ressuscitamento, gerando uma nova grade aleatória com as mesmas matérias, porém, mesmo com essa nova grade, o indivíduo pode permanecer morto. Se toda a população estiver morta, o processo é tentado novamente mais algumas vezes, definidas por uma constante no código. E, se ainda assim, não foi possível gerar nenhum indivíduo vivo, o programa termina informando que não foi possível obter uma grade de horários válidos.

3. Resultado

Como resultado do trabalho, foram produzidos dois programas. O primeiro($main.cpp$) implementa o algoritmo descrito nesse relatório, o segundo($csv.cpp$) implementa um código que dada a saída do primeiro gera um arquivo .csv com a grade do semestre. Para melhor resultado, o programa foi simplificado ao máximo e uma entrada bem construída pode gerar melhores resultados. Por exemplo, para adicionar uma matéria, o programa vai fazer a checagem conforme a ordem informada na entrada. Então, se o usuário preferir que as aulas sejam divididas durante a semana, ele deve informar a entrada conforme sua preferencia.

No programa $main.cpp$, deve-se tomar cuidado com as constantes definidas, pois elas devem ser alteradas dependendo da entrada e/ou preferências do usuário. A lista a seguir comenta brevemente sobre cada uma delas:

- MAX: Quantidade máxima de professores/turmas/disciplinas que vão constar na entrada;
- ERR: Quantidade máxima aceitável de erros numa grade;

- TOLERANCE: Quantas vezes será tentada ressuscitar toda uma população antes de terminar o programa;
- TAM: Tamanho da população;
- QTDBILHETES: Quantos bilhetes um individuo sem erros recebe para a seleção;
- QTDTURMAS: Quantas turmas existem na entrada. É imprescindível que esse valor esteja de acordo com a entrada;
- TIMEMAX: Quanto tempo o algoritmo funcionará;
- PROB: Probabilidade de mutação (1 a 100).

4. Conclusão

Embora tenha sido testado apenas com entradas pequenas, o programa se mostrou bastante eficiente. Um dos problemas identificados é o uso de inteiros no meio do código que deveriam ser trocados por constantes, se houvesse uma otimização. E, além disso, o programa não prioriza que as matérias estejam separadas em dias distintos. Portanto, se houvesse mais tempo, ou até mesmo uma continuação do trabalho, são pontos que seriam melhor considerados.