Sobre o Problema do Escalonamento de Professores
Léo H.

1 Condições para a aplicação no Ensino Fundamental

1.1 Particularidades do Ensino Fundamental Brasileiro

1. A semana (um ciclo da agenda) será dividida em 5 dias, subdivididos em 5 períodos, sendo que entre o terceiro e o quarto há um intervalo.

2. É sempre pré-decidido quais períodos uma turma

1.2 "Hard Constraints"

1. Não há um mesmo período em que duas turmas tem aula com um mesmo professor.

2. Não há um mesmo período em que dois professores dão aula para uma mesma turma.

3. O número de vezes que um professor se encontra com uma turma é dado.

2 Desenvolvimento Diário

2.0.1 30 Oct 2019

Me parece que trabalhar na perspectiva do professor parece bem prudente. Ao invés de desenvolver na ordem Teacher - TeacherQuantity - Class, acredito que o contrário seja mais prudente. Como os horarios dos professores são infinitamente mais limitados, há muito maior possibilidade de propagação de constraint.

2.0.2 31 Oct 2019

Talvez trabalhar com os dois aspectos ao mesmo tempo seja mais proveitoso. Só ainda não sei como aproveitar isso.

2.0.3 21 Nov 2019

A ideia da Command Line Interface foi maaais ou meeenos colocada no lugar. O ponto é que podemos escolher duas coisas:

1. Fazer constraint propagation e sugestões pra quem tiver montando. E dai uma interface bem completinha ajudaria na solução do problema.

2. Tentar deduzir por heurística de uma vez só uma agenda. Tem menos chance de sucesso, no sentido que demora mais e dá menos controle para o usuário. Além disso, é pior para fazer um fact-check e descobrir se está boa a agenda.

O que eu vejo é que a primeira opção é melhor, mas menos comum na indústria. É só que o esforço teria que ser maior na parte de interface gráfica. Então se tivermos uma CLI robusta o suficiente, podemos ter uma GUI que só se utiliza de CLI.

Logo, para tudo funcionar bem, precisamos de funcionalidades como:
- ADD (professor, aluno, turma, disciplina, cronograma);
- "'SELECT (professor, aluno, turma, disciplina, cronograma);"'
- "'DELETE (professor, aluno, turma, disciplina, cronograma);"'
- "'UNDO (bem importante para desfazer merdas);"'

- “‘REDUCE cronograma;“‘

2.1 15 Dec 2019

Começando a pensar que é overkill fazer uma CLI inteira. Aprender com o Stockfish sobre a análise de movimentos futuros e notas:

- Transposition Table.
- Multicore.

De qualquer forma, acho prudente distinguir a parte lógica e a parte que ”chuta” do programa. O problema vai ser como elas se comunicam.

2.2 18 Dec 2019

Uma coisa que tem ocupado minha cabeça hoje é que se um professor precisa dar 5 aulas para uma mesma turma, isso gerará 5 meetings. E assim como no problema de 0 ¡= a ¡= b ¡= c ¡= d ¡= 5, a meeting 2 não pode acontecer antes da meeting 1 nem depois da meeting 3. Isso faz com que propagar constrains seja fundamentalmente mais fácil, já que há menos possibilidades.

O problema é que se eu tentar fazer isso e depois inserir uma meeting fixa, digamos, no período 2, qual vai ser a meeting que vai receber? Não pode haver uma ordem clara entre elas enquanto o usuário ainda puder digitar alterações.

É necessário então um algorítmo que primeiro faça uma cópia da lista de meetings, faça essa eliminação [...] a ¡= b [...], calcule os resultados disso (quem sabe até uma meeting fixa) e depois retorne para o estado desordenado.

TEOREMA: a função order_by_score_discrepancy nunca retornará um número menor que zero. PROVA: $max >= a_k$ para qualquer k. $max >= media$. $max \times n >= media \times n$ logo $max \times n - soma\_total >= 0$. A igualdade só ocorre quando $a_1 = a_2 = a_3 = ... = a_n$.

Order by score discrepancy operacional. Não bem testada, mas operacional. Como todo o programa envolve pequenas escolhas com grandes implicações, pensei que fazer essas escolhas de forma 100% aleatória não ajudaria em muita coisa. Além disso, queria fazê-las de uma forma econômica. Então pensei que fazer primeiro todas as que estivessem com o score_discrepancy mais alto seria mais prudente, já que essas envolvem menos opções.

Seria possível elimiar a função check_for_fixed_meetings? Ela é só uma versão mais eespecífica da nova de score discrepancy.

2.3 19 Dec 2019

Não esquecer de comentar na documentação que toda lista nesse programa é uma ”invalid-terminated list”. Que nem as null-terminated strings padrões do C.

TO DO: make a function like propagate_meeting_annihilation(), that works well on this case. explore_consequences() does not work well here. I don’t know fully well the consequences of it. One thing that I know it does is that if there is only one option remaining in the possible_periods now, it will collapse to period.

TO DO: figure out a way of rewarding the school preferences inside node-¿score. Ideally, it would be stricly in the order computed by score_order that we find the best timetables