Uma Abordagem Multiobjetivo para o Problema do Escalonamento de Médicos

Lucas Machado Cid[1]*, Mário César San Felice[1], Pedro H. Del Bianco Hokama[2]

[1]Departamento de Computação – UFSCar – São Carlos – SP – Brazil

[2]Instituto de Matemática e Computação – UNIFEI – Itajubá – MG – Brazil

lucas.cid@estudante.ufscar.br, felice@ufscar.br, hokama@unifei.edu.br

Abstract. *The Physician Rostering Problem (PRP) seeks to assign shifts to physicians, so that all hospital demands are met, no physician is overloaded, and the scheduling is as pleasant as possible for them. Constraint Programming (CP) is a paradigm for solving combinatorial problems, which combines techniques from Artificial Intelligence, Theory of Computation and Operations Research. This work uses CP to solve the PRP considering demands from a real hospital, while addresses the quality and variety of solutions from the perspective of multiobjective optimization in order to assist the decision maker. Satisfactory results have been achieved for instances with up to 40 physicians and a 30-day planning horizon.*

Resumo. *No problema do escalonamento de médicos (PRP, do inglês Physician Rostering Problem) busca-se atribuir turnos para médicos, de forma que todas as demandas do hospital sejam atendidas, nenhum médico fique sobrecarregado, e o escalonamento fique o mais agradável possível para estes. Programação por Restrições (CP, do inglês Constraint Programming) é um paradigma para resolução de problemas combinatórios, que combina técnicas de Inteligência Artificial, Teoria da Computação e Pesquisa Operacional. Este trabalho utiliza CP para resolver o PRP considerando demandas de um hospital real, enquanto aborda a qualidade e variedade das soluções pela ótica da otimização multiobjetivo para melhor auxiliar o tomador de decisão. Foram obtidos resultados satisfatórios com até 40 médicos num horizonte de planejamento de 30 dias.*

1. Introdução

O paradigma de Programação por Restrições [Dechter 2003, Rossi et al. 2006] estuda problemas envolvendo a atribuição de valores para variáveis de decisão a fim de satisfazer certas restrições. Como exemplo, temos o problema de atribuir turnos de trabalho aos profissionais de um hospital, de forma que as necessidades do hospital (todos os turnos com certo número de funcionários e com alguns profissionais mais experientes, por exemplo), e condições de trabalho (como ninguém ter que trabalhar o dia inteiro) sejam atendidas. Além disso, se possível, também deseja-se atender restrições que deixem a situação mais confortável para os profissionais (um exemplo seria um profissional que prefere não trabalhar em turnos noturnos, se possível). Este problema é abordado tanto pelo problema

*processo nº 2021/10996-2, Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP)

do Escalonamento de Médicos (PRP, do inglês Physician Rostering Problem) quanto no problema do Escalonamento de Enfermeiras (NSP, do inglês Nurse Scheduling Problem), sendo o primeiro destes o foco deste trabalho.

A relevância do PRP deriva diretamente do cenário que o motivou, por permitir melhorar as condições de trabalho em instituições de saúde e ao mesmo tempo permitir que essas instituições aloquem de melhor forma seus recursos humanos. O PRP corresponde a uma família de problemas que apresenta grande variedade de restrições, já que as exigências de escalonamento mudam significativamente quando consideramos diferentes cenários de aplicação. A maior parte dos problemas desta família é NP-difícil, não sendo possível resolvê-los de forma ótima em tempo polinomial a menos que P=NP [Bovet and Crescenzi 1994]. Portanto, diversas abordagens inventivas são utilizadas para encontrar soluções de boa qualidade e em tempo razoável para estes problemas.

O survey de Erhard, Schoenfelder, Fügener, Brunner [Erhard et al. 2018] revisa os principais trabalhos existentes na área do Escalonamento de Médicos e explica as diversas peculiaridades do problema, além de analisar e comparar abordagens diferentes para sua resolução. Por exemplo, em relação aos turnos, podemos considerá-los como fixos (e predefinidos) ou adaptáveis. O mais comum é que os turnos sejam fixos, geralmente de 8 horas cada, abrangendo assim todo o dia. Também é comum que cada turno tenha 6 horas de duração, com exceção do turno noturno, que se estende por 12 horas seguidas. Já os turnos adaptáveis dão mais opções ao escalonamento por permitirem a atribuição de médicos para trabalhar em turnos que variam em duração e horário de início, como é o caso no artigo de Brunner et al. [Brunner et al. 2009].

Um diferencial do PRP é que as soluções, além de respeitar as peculiaridades e normas do hospital, podem ser melhoradas ao favorecer as preferências individuais dos médicos e ao criar um cronograma justo. Para este objetivo, as restrições são divididas em dois conjuntos: restrições rígidas (imprescindíveis para que a solução seja viável) e restrições flexíveis (não interferem diretamente em uma solução ser ou não viável, mas sim na qualidade da mesma). Um hospital pode, por exemplo, definir que é preferível que um médico não trabalhe dois finais de semanas seguidos, ou que não trabalhe mais do que 4 turnos noturnos por semana. Essas restrições podem ou não ser atendidas, mas se forem, melhoram a qualidade das soluções. A questão que surge daqui é justamente como definir o quanto elas melhoram a qualidade das soluções. Poderíamos ter uma solução que respeita a restrição de não trabalhar finais de semana seguidos e ter outra solução que respeita a restrição de não trabalhar mais do que 4 turnos noturnos por semana. Nesse caso, qual é a melhor solução?

Existem diversos métodos para atender tais restrições flexíveis. Fügener, Brunner e Podtschaske [Fügener et al. 2015] utilizam um dos mais comuns que consiste em adicionar, quando houver a quebra de tais restrições, uma penalidade na função objetivo com peso equivalente à importância da restrição. Desse modo, ao minimizarmos a função objetivo, as soluções que sofrerem menos penalidades serão priorizadas. Este método pode, entretanto, esbarrar na dificuldade de encontrar pesos adequados para cada restrição flexível já que não existe nenhuma medida comumente aceita para compará-las.

Para lidar com essa dificuldade recorremos à área de Otimização Multiobjetivo [Coello et al. 2009]. Diferente de problemas com uma única função objetivo, nos

quais almejamos encontrar apenas a melhor solução, aqui temos várias funções objetivo para avaliar cada solução e nosso objetivo passa a ser encontrar soluções que compõe a Fronteira de Pareto. A Fronteira de Pareto é um conjunto de soluções em que as funções objetivo de cada uma delas não podem ser melhoradas sem piorar alguma outra e em que nenhuma solução domina a outra, sendo impossível compará-las sem utilizar algum critério subjetivo ou arbitrário. Por definição, para que uma solução x domine y, x deve ter, em um contexto de minimização, todas suas funções objetivo com valor menor ou igual a y e ao menos uma com valor estritamente menor. Formalmente, temos que $\forall i \in \{1, \ldots, n\} f_i(x) \leq f_i(y)$ e $\exists i \in \{1, \ldots, n\} f_i(x) < f_i(y)$.

2. Definindo o Problema do Escalonamento de Médicos

O Escalonamento de Médicos é um problema comum de ser encontrado em instituições de saúde e geralmente é difícil resolvê-lo manualmente. Por conta do grande numero de possibilidades, essa tarefa acaba consumindo muito tempo de um profissional altamente especializado e muitas vezes gera um resultado ruim. A automatização do escalonamento pode, portanto, melhorar a atribuição de médicos a turnos, além de diminuir o tempo gasto com a elaboração do mesmo.

O problema consiste na atribuição de turnos de trabalho para médicos de um Hospital em um determinado horizonte de planejamento. Essas atribuições, por sua vez, devem respeitar tanto as diretrizes do hospital como as regras contratuais definidas para cada médico. Essas são restrições rígidas, que definem se uma solução para o problema é ou não válida. O problema, entretanto, vai além de encontrar soluções válidas; é também desejável respeitar restrições flexíveis que capturam preferências pessoais dos funcionários e evitam que algumas situações ocorram. Peguemos como exemplo a situação em que um médico trabalha vários turnos seguidos; alguns hospitais podem permitir que isso ocorra, mas no geral preferem evitar. Nesse caso, é preciso priorizar as soluções que não infringem essas regras mais brandas.

Para o problema precisamos considerar o conjunto de dias para os quais desejamos montar o cronograma de trabalho. É preciso considerar também as diferentes áreas que podem existir dentro de um hospital, podendo ser necessárias quantidades diferentes de médicos para suprir a demanda em cada área. É até mesmo possível que apenas profissionais com alguma especialidade específica possam ser alocados em determinadas áreas, adicionando maior complexidade para a tarefa. O escalonamento é, portanto, a tarefa de atribuir médicos para trabalhar turnos nas diferentes áreas do hospital, de modo a criar um cronograma para todo o horizonte de planejamento definido.

2.1. Levantamento de Requisitos

O PRP define uma família de problemas semelhantes, porém distintos, pois cada hospital e conjunto de profissionais tem suas regras e preferências específicas. Assim, para completar a definição e modelagem do nosso PRP entramos em contato com profissionais de saúde envolvidos na formulação dessas escalas e levantamos requisitos típicos tanto dos hospitais quanto dos profissionais.

- **Cuidados com turnos noturnos.** O turno noturno é mais longo, cansativo e estressante. Por este motivo, deseja-se impor um número máximo de turnos noturnos que cada médico pode trabalhar por semana. Além do mais, cada médico deve ter um descanso de ao menos 12 horas caso tenha trabalhado em um turno noturno.

- **Distribuição das horas de trabalho.** É desejável que as horas de trabalho de um médico sejam distribuídas sem sobrecarregar dias ou semanas específicas.
- **Médicos não devem ser alocados em períodos em que eles não estão disponíveis.** Médicos podem ter restrições sobre dias em que eles têm compromissos e não devem ser alocados para trabalhar nesses dias pré-definidos.
- **Respeitar as horas definidas no contrato.** É desejável que os médicos trabalhem um total de horas próximo ao definido no contrato, reduzindo assim horas extras ou devidas.
- **O escalonamento deve ser o mais estável possível.** Isso significa que é desejável que os dias e turnos em que um médico é alocado não variem muito a cada semana.

3. Modelando o PRP

A partir dos requisitos levantados montamos modelos matemáticos que visam atender tais necessidades. Formalmente, no PRP, temos um número P de médicos disponíveis no hospital, um número D de dias (geralmente totalizando um mês) que estão incluídos no horizonte de planejamento, um número S de turnos (geralmente três) por dia a serem trabalhados no hospital, um número A de diferentes áreas que precisam de atenção na unidade de saúde e uma função $\mathrm{V} : \mathrm{D} \times \mathrm{S} \times \mathrm{A} \to \mathbb{N}$ que indica o número de vagas que devem ser preenchidas pelos médicos em um determinado dia, turno e área do hospital. Para qualquer $\mathrm{K} \in \mathbb{N}$ seja $[\mathrm{K}] = \{1, 2, \ldots, \mathrm{K}\}$. Para cada d em D, s em S, a em A e v em $[\mathrm{V}(d, s, a)]$, temos uma variável $\mathrm{W}_{d,s,a,v}$ que indica o médico que irá atuar na vaga v da área a durante o turno s do dia d.

Cada turno pode ter um número diferente de horas a serem trabalhadas; para este propósito, usamos a função $\mathrm{H} : \mathrm{S} \to \mathbb{Z}^+$ que indica a duração de cada turno. Além disso, cada médico pode ter um número diferente de horas de trabalho por semana definido em seu contrato. Assim, definimos a função $\mathrm{WR} : \mathrm{P} \to \mathbb{Z}^+$ que indica quantas horas cada médico deve trabalhar por semana.

Como parte da entrada, temos o dia da semana em que se inicia o horizonte de planejamento. Considerando isso e o número de dias D, calculamos o número de semanas SM que são parte do horizonte de planejamento e, para cada $sm \in [\mathrm{SM}]$, um conjunto DS_{sm} que contém os dias de semana sm que fazem parte do nosso horizonte de planejamento. A função $F : \mathrm{P} \times \mathrm{D} \times \mathrm{S} \to \mathbb{N}$ também faz parte da entrada e indica se um médico não está disponível para trabalhar em determinado dia e turno.

3.1. Restrições Rígidas

$$\sum_{a \in \mathrm{A}} \sum_{v \in [\mathrm{V}(d,s,a)]} (\mathrm{W}_{d,s,a,v} = p) \leq 1 \qquad p \in \mathrm{P}, d \in \mathrm{D}, s \in \mathrm{S} \tag{1}$$

$$\sum_{d \in \mathrm{DS}_{sm}} \sum_{\substack{a \in \mathrm{A} \\ v \in [\mathrm{V}(d,s,a)]}} (\mathrm{W}_{d,S,a,v} = p) \leq \mathrm{MaxNS} \qquad p \in \mathrm{P}, sm \in \mathrm{SM} \tag{2}$$

$$\sum_{\substack{a \in \mathrm{A} \\ v \in [\mathrm{V}(d,s,a)]}} (\mathrm{W}_{d,S,a,v} = p) \geq 1 \Rightarrow \sum_{\substack{s \in [\mathrm{S}-1] \\ a \in \mathrm{A} \\ v \in [\mathrm{V}(d+1,s,a)]}} (\mathrm{W}_{d+1,s,a,v} = p) = 0 \qquad \begin{matrix} p \in \mathrm{P}, \\ d \in [\mathrm{D}-1] \end{matrix} \tag{3}$$

$$\sum_{i=d}^{d+6} \sum_{\substack{s\in \mathrm{S} \\ a\in \mathrm{A} \\ v\in[\mathrm{V}(d,s,a)]}} (\mathrm{W}_{i,s,a,v} = p)\mathrm{H}(s) \geq \mathrm{WR}(p) - \mathrm{LimiteInf} \qquad p \in \mathrm{P}, d \in [\mathrm{D}-6] \tag{4}$$

$$\sum_{i=d}^{d+6} \sum_{\substack{s\in \mathrm{S} \\ a\in \mathrm{A} \\ v\in[\mathrm{V}(d,s,a)]}} (\mathrm{W}_{i,s,a,v} = p)\mathrm{H}(s) \leq \mathrm{WR}(p) + \mathrm{LimiteSup} \qquad p \in \mathrm{P}, d \in [\mathrm{D}-6] \tag{5}$$

$$\mathrm{W}_{d,s,a,v} \neq p \qquad \begin{array}{r} p \in \mathrm{P}, d \in \mathrm{D}, s \in \mathrm{S}, \\ v \in [\mathrm{V}(d,s,a)], \\ a \in \mathrm{A}, f(p,d,s) = 1 \end{array} \tag{6}$$

$$\mathrm{W}_{d,s,a,v} < \mathrm{W}_{d,s,a,v+1} \qquad \begin{array}{r} d \in \mathrm{D}, s \in \mathrm{S}, \\ a \in \mathrm{A}, v \in [\mathrm{V}(d,s,a)-1] \end{array} \tag{7}$$

Na restrição (1) garantimos que um médico nunca poderá ocupar duas vagas ao mesmo tempo. A restrição (2) define que um médico pode trabalhar no máximo MaxNS número de turnos noturnos por semana. Na restrição (3) lidamos com a situação em que um médico trabalhou no turno da noite em um determinado dia. Nesse caso, ele não poderá trabalhar os próximos dois turnos do dia seguinte. As restrições (4) e (5) garantem que todos os médicos, em quaisquer séries de 7 dias consecutivos, trabalharão o número de horas estipulado em seus contratos. $\mathrm{LimiteInf}$ e $\mathrm{LimiteSup}$ são constantes cujos valores podem ser definidos para relaxar essa restrição. A restrição (6) garante que um médico não será alocado para trabalhar nos períodos em que não estiver disponível. Finalmente, a restrição (7) é usada para remover a simetria. Pela natureza do problema, as soluções em que um médico alterna entre as vagas disponíveis são consideradas soluções diferentes, mesmo que na prática sejam as mesmas. Essa restrição introduz uma ordem de atribuição dos médicos às vagas: deve ser feita em ordem crescente, de modo que o identificador do médico que ocupa uma vaga de determinado dia e área seja menor que o identificador do médico que ocupa a vaga seguinte.

3.2. Restrições Flexíveis

Atualmente, modelamos quatro restrições flexíveis como funções objetivo para capturar diferentes aspectos de qualidade de uma solução.

A função objetivo (8) penaliza a instabilidade no cronograma. Ou seja, se um médico está alocado para um determinado dia e turno em uma semana, mas não trabalha nesse mesmo período nas demais semanas. Neste caso, a função objetivo é penalizada. Este função objetivo visa melhorar a qualidade do trabalho dos profissionais de saúde, minimizando os efeitos negativos de um cronograma instável. Aqui usamos SM_d como o número de semanas em que o dia da semana d aparece no horizonte de planejamento. Para esta função objetivo, para cada $p \in \mathrm{P}$, $d \in \{1, \ldots, 7\}$ e $s \in \mathrm{S}$, definimos uma variável $\mathrm{Y}_{p,d,s}$ que indica quantas semanas o médico p trabalhou no turno s do dia da semana d.

$$\mathrm{Y}_{p,d,s} = \sum_{i=0}^{\mathrm{SM_d}-1} \sum_{\substack{a\in \mathrm{A} \\ v\in[\mathrm{V}(d,s,a)]}} \mathrm{W}_{d+i*7,s,a,v} = p$$

Dessa forma, podemos modelar a seguinte função objetivo

$$\min \sqrt{\sum_{p \in \mathrm{P}} \left(\sum_{d=1}^{7} \sum_{s \in \mathrm{S}} (\mathrm{Y}_{p,d,s} \geq 1) * (\mathrm{SM_d} - \mathrm{Y}_{p,d,s}) \right)^2} \tag{8}$$

A função objetivo (9) penaliza a atribuição de cargas diárias que ultrapassem 12 horas, ja que é indesejável que médicos trabalhem longos turnos sem descanso. A penalidade é proporcional ao número de horas de trabalho em excesso. Para esta função objetivo, para cada $p \in \mathrm{P}$ e $d \in \mathrm{D}$, definimos uma variável $\mathrm{X}_{p,d}$ cujo valor é a quantidade de horas que o médico p trabalhou no dia d.

$$\mathrm{X}_{p,d} = \sum_{\substack{s \in \mathrm{S} \\ a \in \mathrm{A}}} \sum_{v \in [\mathrm{V}(d,s,a)]} (\mathrm{W}_{d,s,a,v} = p) * \mathrm{H}(s)$$

Então, modelamos a seguinte função objetivo

$$\min \sqrt{\sum_{p \in \mathrm{P}} \left(\sum_{d \in \mathrm{D}} (\mathrm{X}_{p,d} > 12) * (12 - \mathrm{X}_{p,d}) \right)^2} \tag{9}$$

A função objetivo (10) penaliza horas trabalhadas a mais ou a menos durante o horizonte de planejamento. Essa penalidade leva em consideração quantas horas devem ser trabalhadas de acordo com o contrato do profissional.

$$\min \sqrt{\sum_{p \in \mathrm{P}} \left(\sum_{\substack{d \in \mathrm{D} \\ s \in \mathrm{S}}} \sum_{\substack{a \in \mathrm{A} \\ v \in [\mathrm{V}(d,s,a)]}} ((\mathrm{W}_{d,s,a,v} = p) * \mathrm{H}(s)) - ((\mathrm{D}/7) * \mathrm{WR}(p)) \right)^2} \tag{10}$$

A função objetivo (11) é semelhante às restrições rígidas (4) e (5). Ela penaliza cargas de trabalho maiores ou menores do que a carga horária semanal definida no contrato do médico.

$$\min \frac{\sqrt{\sum_{p \in \mathrm{P}} \left(\sum_{d \in \mathrm{D}} \left| \left(\left(\sum_{i=0}^{6} \sum_{s \in \mathrm{S}} \sum_{\substack{a \in \mathrm{A} \\ v \in [\mathrm{V}(d,s,a)]}} (\mathrm{W}_{d+i,s,a,v} = p) * \mathrm{H}(s) \right) - \mathrm{WR}(p) \right) \right| \right)^2}}{7} \tag{11}$$

Pelo modo como inicialmente as penalidades estavam sendo calculadas nas funções objetivo, não havia nada que impedisse que um determinado médico fosse sobrecarregado com situações indesejadas enquanto outros se vissem completamente livres dessas situações. Como alternativa, ao invés de somar as penalidades para cada médico, optou-se pela soma do quadrado da penalidade. Desse modo, as penalidades serão cada vez maiores quando acumuladas por um médico e menores quando distribuídas ao longo da força de trabalho do hospital. Essa alteração aumentou a justiça no escalonamento impedindo que o cronograma se mostre muito bom para alguns médicos, mas sob o custo de sobrecarregar um ou mais poucos médicos.

4. Implementação

Nessa seção são relatados detalhes de implementação do software cujo código pode ser encontrado no GitHub[1]. O modelo proposto foi implementado usando linguagem C++ e o resolvedor CPLEX CP Optimizer 12.10, sendo que primeiro foram implementadas as restrições rígidas apresentadas na Subseção 3.1. Na Subseção 4.1 é detalhada a criação e utilização de um gerador de instâncias para o problema. Na Subseção 4.2 relatamos a implementação das Restrições Flexíveis e decisões tomadas para atacar o caráter multi-objetivo do problema.

4.1. Geração de Instâncias

Com base em informações obtidas no levantamento de requisitos para o problema foi desenvolvido um gerador de instâncias, uma ferramenta importante para avaliar a eficácia e a eficiência do modelo implementado. As instâncias geradas pelo algoritmo permitem avaliar o desempenho do modelo em diferentes cenários, considerando o número de médicos e carga horária de cada um deles, bem como o número de áreas e vagas disponíveis. O gerador de instâncias recebe como entrada um arquivo de configurações, no qual são definidos os parâmetros que irão nortear a geração dos dados. Os parâmetros definidos são:

- **Horas.** Contém uma lista com as possíveis durações das jornadas de trabalho dos médicos. Nesse caso, são permitidas jornadas de 18, 24, 30 e 36 horas.
- **Número de médicos.** Contém uma lista com o número de médicos que podem ser gerados em cada instância. Nesse caso, serão geradas instâncias com 10, 20, 30 ou 40 médicos.
- **Vagas por área.** Contém uma lista com o número de vagas disponíveis para cada área de especialidade. Nesse caso, há áreas com 1, 2 e 3 vagas disponíveis.
- **Número máximo de áreas.** Define a quantidade máxima de áreas que uma instância pode ter. Nesse caso, as instâncias geradas terão no máximo 4 áreas.

O algoritmo utiliza uma biblioteca de geração de números aleatórios para escolher a carga horária de cada médico, selecionando também aleatoriamente a quantidade de médicos que serão criados para cada uma das instâncias. A geração de instâncias para áreas segue um modelo parecido. Com o número máximo de áreas e o número de vagas possíveis para cada área definidos na configuração, o algoritmo seleciona aleatoriamente um número de áreas para cada instância (que não extrapole o número máximo pré-definido) e em seguida seleciona o número de vagas para cada uma das áreas. Por fim uma combinação de áreas com o número de vagas correspondente é gerado. Finalizada a geração de médicos e áreas, é possível permutar as entidades a fim de gerar instâncias distintas.

4.2. Implementação de Restrições Flexíveis

Em seguida foi iniciada a implementação das restrições flexíveis apresentadas na Subseção 3.2. A princípio as penalidades geradas pelas funções objetivo que representam cada restrição foram somadas com pesos iguais em uma função a ser minimizada. Assim, quanto maior o valor das penalidades mencionadas, maior o valor da função objetivo e, consequentemente, pior a solução. O programa então devolve a solução encontrada com menor valor para a função objetivo.

[1] `https://github.com/Lucas-Cid/physician-roastering`

Essa abordagem, apesar de funcional, esbarra na dificuldade de se encontrar pesos para os valores de penalidades distintos de cada uma das restrições, isto é, definir o grau de importância de cada uma delas. Uma solução $\mathrm{S1}$ pode se sair muito bem em uma restrição $\mathrm{R1}$, mas ignorar completamente uma restrição $\mathrm{R2}$. Uma outra solução $\mathrm{S2}$ faz o oposto: se sai muito bem em $\mathrm{R2}$ mas ignora $\mathrm{R1}$. Desse modo, comparar $\mathrm{S1}$ e $\mathrm{S2}$ não faz sentido, já que elas se sobressaem em quesitos diferentes.

Seguindo a abordagem da Otimização Multiobjetivo, é importante considerar as funções objetivo independentemente. Entretanto, o CP Otimizer tem a limitação de permitir a minimização de apenas uma função objetivo. Como solução, optou-se então pela implementação de um algoritmo de geração de pesos, apresentado no Algoritmo 1, responsável por variar e atribuir pesos a cada uma das funções objetivo. Para cada conjunto de pesos gerado é feita uma chamada do programa, gerando uma solução. Assim, é

Algoritmo 1: Gerador de Pesos

Entrada: Dimensão DIM de pesos
Saída: Conjunto de pesos P, cada item do conjunto com dimensao DIM

```
Crie um vetor conjuntoInicial;
para i = 1 até DIM faça
    Crie um vetor T com DIM itens;
    Inicialize T com 0's;
    T_i ← 1;
    Insira T em conjuntoInicial;
    Insira T em P;
Insira conjuntoInicial na fila;
enquanto Critério de parada não atingido faça
    Retire um esquema de pesos E da fila, digamos [p_1, ..., p_DIM];
    novoPeso ← (p_1 + p_2 + ··· + p_DIM)/DIM;
    Insira na fila combinações de tamanho DIM entre E e novoPeso;
    Insira novoPeso em P;
devolva P;
```

possível a obtenção de um conjunto de soluções que priorizam quesitos diferentes, dependendo dos pesos utilizados. Vale destacar que este algoritmo escolhe pesos de modo homogêneo pelo espaço de soluções, visando obter soluções com pontos fortes e fracos distintos.

A partir do conjunto de soluções obtidas, são eliminadas as soluções dominadas, a fim de obter uma Fronteira de Pareto. No fim, temos um conjunto de soluções que respeitam todas as restrições rígidas e que priorizam em diferentes níveis os aspectos de justiça no escalonamento.

5. Estratégias para Melhoria da Busca por Soluções

Nessa seção são apresentadas estratégias para melhorar a busca por soluções variadas. Na Subseção 5.1 é explicado como a normalização dos valores das Funções Objetivo foi feita e quais as vantagens dessa normalização. Na Subseção 5.2 relatamos uma estratégia para evitar que o programa perca tempo buscando por uma solução dominada por outras soluções já encontradas.

5.1. Normalização pelos Valores Ideal e Nadir

Como as funções objetivo que representam cada uma das restrições tem maneiras diferentes de realizar o cálculo das penalidades, é comum que alguma função objetivo acabe tendo valores com maior magnitude que outras, o que pode refletir em maior importância nas funções objetivo combinadas a despeito dos pesos utilizados. Por este motivo, a normalização se apresenta como uma alternativa interessante para garantir que as funções objetivo sejam tratadas igualmente. Para este fim foi realizada a normalização pelos valores Ideal e Nadir. Formalmente, temos um número DIM de funções objetivo. O Algoritmo 1 implementado gerará inicialmente DIM conjuntos de pesos p_i de forma a ativar individualmente a i-ésima função objetivo, desativando todas as outras.

$$p_1 = [1, 0, 0, \ldots, 0], p_2 = [0, 1, 0, \ldots, 0], \ldots, p_{\mathrm{DIM}} = [0, 0, 0, \ldots, 1]$$

Sendo $\mathcal{S} = \{S_1, S_2, \ldots, S_{\mathrm{DIM}}\}$ o conjunto das DIM soluções obtidas inicialmente, e tomando S_i, a solução gerada utilizando os pesos p_i, podemos definir o valor ideal para a i-ésima função objetivo como

$$f_i^* = f_i(S_i)$$

Já o ponto Nadir será construído a partir dos piores valores para as funções objetivo das DIM soluções iniciais encontradas

$$f_i^N = \max_{\forall S \in \mathcal{S}} f_i(S)$$

O ponto Ideal e o ponto Nadir são, portanto $(f_1^*, \ldots, f_{\mathrm{DIM}}^*)$ e $(f_1^N, \ldots, f_{\mathrm{DIM}}^N)$, respectivamente. Por fim, dada uma solução S qualquer, para cada função objetivo f_i passamos a utilizar sua versão normalizada, que é calculada segundo a fórmula

$$\frac{f_i(S) - f_i^*}{f_i^N - f_i^*}$$

5.2. Busca por Soluções Não Dominadas

Com a utilização do Algoritmo 1, passamos a utilizar os pesos de modo a gerar, sequencialmente, soluções para o problema. Entretanto, o programa não tinha conhecimento algum sobre as soluções anteriores encontradas, fazendo com que ocasionalmente novas soluções dominadas ou repetidas fossem encontradas ao longo do processo. Para resolver este problema, o modelo passou a receber como entrada também um conjunto S com as soluções previamente encontradas. A partir desse conjunto, são acrescentadas novas restrições que excluem do espaço de busca soluções dominadas por aquelas em $\mathcal{S}$. Isto é, uma nova solução SN deve ter ao menos uma função objetivo com valor menor (considerando um contexto de minimização) quando comparada com todas as outras funções soluções encontradas. Assim, sendo DIM o número de funções objetivo, para cada solução $\mathrm{S} \in \mathcal{S}$ adicionamos a seguinte restrição ao modelo

$$\left(\sum_{i=1}^{\mathrm{DIM}} f_i(SN) < f_i(S) \right) \geq 1$$

6. Resultados Experimentais e Apresentação das Soluções

Utilizando as instâncias geradas a partir do script detalhado na Subseção 4.1, rodamos o programa de duas maneiras distintas. Na primeira delas, visando apenas encontrar uma solução viável, sem olhar para as funções objetivo. Já na segunda, buscamos a minimização das funções objetivo com pesos uniformes, e estabelecendo um tempo limite de 1 hora por instância. A comparação entre alguns destes resultados encontra-se na Tabela 1. As colunas da tabela são as seguintes:

- **Nome**: o nome da instância;
- **Médicos**: o número de médicos disponíveis;
- **Áreas**: o número de áreas que devem ser atendidas pelos médicos;
- **Vagas**: o número de vagas, por dia, que devem ser preenchidas por médicos;
- **fi**: a economia do recurso i obtida pela otimização (f_i^*) em relação à solução encontrada sem otimização (f_i'). A medida segue a expressão $f_i^* / f_i' - 1$.

Tabela 1. Testes e melhoria relativa

Nome	Médicos	Áreas	Vagas	f1	f2	f3	f4
a2p2	20	1	9	-15,95%	-29,29%	-17,93%	-42,57%
a5p2	20	2	12	-4,32%	-38,17%	-23,24%	-16,58%
a8p3	30	3	18	-9,60%	-9,30%	-30,19%	-23,89%
a12p3	30	4	18	-8,73%	-13,90%	-31,78%	-24,71%
Média				-9,65%	-22,67%	-25,78%	-26,94%

Uma vez que este trabalho tem como objetivo a utilização do programa em um hospital real, foi dada uma atenção especial à apresentação das soluções. Uma solução viável corresponde a um escalonamento que segue o modelo apresentado na Tabela 2.

Tabela 2. Exemplo de solução viável

	Manhã				Tarde				Noite			
Dias	*Área 1*		*Área 2*		*Área 1*		*Área 2*		*Área 1*		*Área 2*	
23/03/2022 (Qua)	Otavio	Mario	Alex	Pedro	Hugh	Mario	Gabriel	Pedro	Julia	Mario	Otavio	Gabriel
24/03/2022 (Qui)	Gustavo	Alex	Keira	Pedro	Lucas	Gustavo	Alex	Pedro	Lucas	Otavio	Will	Alex
25/03/2022 (Sex)	Gustavo	Pedro	Gabriel	Mario	George	Mario	Gabriel	Pedro	Hugh	Will	Angel	Keira
26/03/2022 (Sab)	Alex	Pedro	Clive	Julia	Otavio	Pedro	Alex	Mario	Vitor	Alex	Halle	Gustavo
27/03/2022 (Dom)	Otavio	Gabriel	Pedro	Mario	Gabriel	Mario	Otavio	Pedro	Halle	Lucas	Hugh	Clive
28/03/2022 (Seg)	Keira	Mario	Angel	Gustavo	Gustavo	Gabriel	Keira	Mario	Julia	Gustavo	Angel	Gabriel
29/03/2022 (Ter)	Vitor	Alex	Otavio	Pedro	Keira	Lucas	Alex	Pedro	George	Clive	Keira	Vitor

Por conta da abordagem multiobjetivo escolhida para atacar o problema, temos como resultado da execução um conjunto de soluções, cada qual favorecendo aspectos diferentes do problema. Esse conjunto é composto apenas por soluções não dominadas e, como já foi mencionado, não é possível compará-las diretamente. Por esse motivo, uma boa apresentação das soluções se faz necessária, a fim de expor as informações mais relevantes de forma objetiva e compreensível, permitindo que um profissional da área consiga selecionar uma das soluções dentre as presentes no conjunto.

Para facilitar a comparação das soluções de forma visual, foi criado um script em python que lê metadados (valores relacionados ao desempenho de cada função objetivo) das soluções geradas pelo programa principal (e gravadas em arquivos .csv) e as plota lado a lado em um gráfico de barras. A escolha do gráfico de barras se deu pela dificuldade

de apresentar conjuntamente soluções que variam em quatro aspectos distintos (isto é, quatro Funções Objetivo). Como exemplo temos o gráfico da Figura 1, que compara um conjunto de nove soluções.

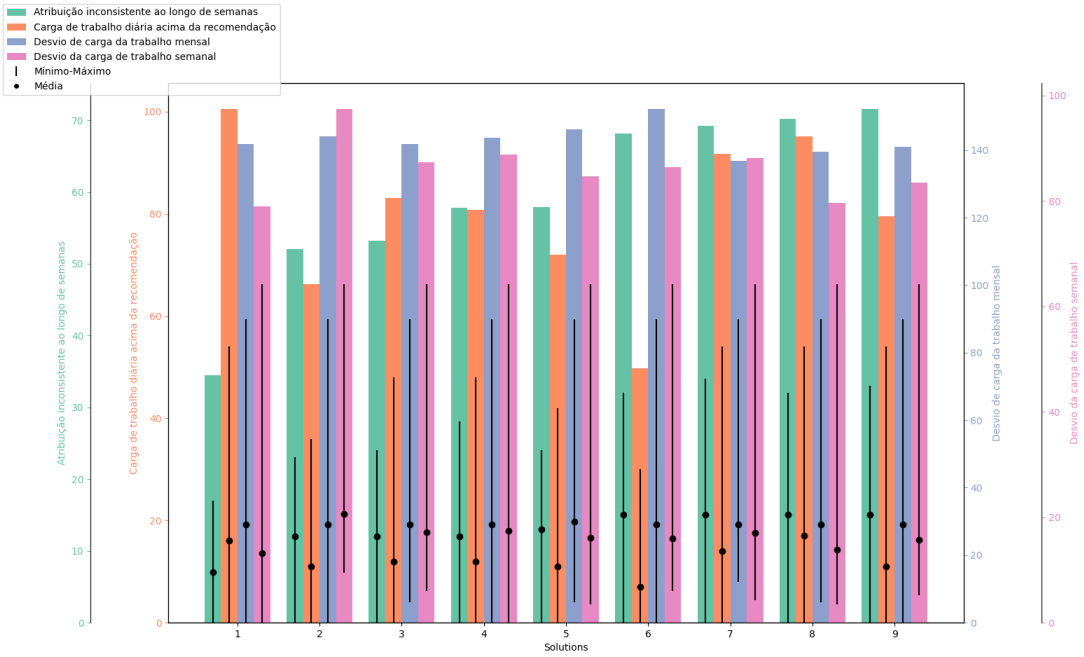

Figura 1. Comparação de soluções nos quatro quesitos analisados

Nesse gráfico, o eixo x representa as soluções encontradas a partir de 9 diferentes pesos gerados pelo Algoritmo 1. Aqui, cada solução é apresentada em blocos de quatro barras, com cada barra representando uma Função Objetivo (FO). Para a representação dos valores de cada barra, optamos pela utilização de quatro eixos y. Essa decisão se deu pelo fato de que cada FO tem sua própria escala (os valores normalizados são utilizados para a busca das soluções, mas na apresentação os valores reais são mostrados) e, caso apenas um eixo fosse utilizado, a visualização das FOs com valores em menor escala seria prejudicada. Dito isso, temos os eixos "Atribuição inconsistente ao longo de semanas", "Carga de trabalho diária acima da recomendação", "Desvio de carga da trabalho mensal" e "Desvio da carga de trabalho semanal" que representam valores para as FOs (8), (9), (10) e (11), respectivamente. No canto superior esquerdo da figura pode ser encontrada uma legenda que associa cada cor com uma FO.

Além do mais, se analisarmos as FOs modeladas na Seção 3.2, podemos observar que as funções são compostas pela somatória, para todos os médicos, de algum valor. Este valor pode ser, por exemplo, o número de horas que um médico trabalhou a menos ou a mais em um horizonte de planejamento (FO 10) ou o overtime total do profissional durante todo o cronograma (FO 9). Estas funções nada mais são do que a somatória de valores individuais para cada médico. Por este motivo, tais valores também são guardados durante a busca por soluções. Dessa forma, podemos apresentar também informações relevantes como mínimo, máximo e média dos valores individuais para cada FO apresentada. No gráfico, estas informações são apresentadas através de um segmento de reta vertical dentro de cada barra. O mínimo é o início do segmento, o máximo é o fim dele e a média é o círculo preto que o intercepta em algum ponto.

7. Conclusão

O escalonamento de médicos é um problema complexo que pode ser resolvido com a utilização da Programação por Restrições (CP), técnica que combina Inteligência Artificial, Teoria da Computação e Pesquisa Operacional. Este artigo apresenta o desenvolvimento de um software que utiliza CP para resolver o problema do escalonamento de médicos com requisitos de um hospital real, abordando a qualidade e variedade das soluções pela ótica da otimização multiobjetivo. O artigo também definiu o problema do escalonamento de médicos e suas principais restrições. Através da implementação de restrições flexíveis, o software pode penalizar soluções que infringem essas regras mais brandas, priorizando soluções que respeitem também as preferências pessoais e institucionais. O software conseguiu encontrar soluções satisfatórias para instâncias de tamanho razoável com 10, 20, 30 e 40 médicos, e um horizonte de planejamento de 30 dias.

Algumas melhorias podem ser implementadas para tornar o software ainda mais eficiente. Uma dessas melhorias é a criação de uma interface para a utilização do software, já que planilhas CSV não são práticas para muitos usuários. Além disso, nas conversas realizadas com médicos, evidenciou-se a necessidade de permitir que os profissionais informem os dias em que eles estarão ou não disponíveis. Portanto, a criação e implementação de uma notação flexível que permita médicos informarem os períodos para os quais poderão ser alocados pode facilitar a tarefa de escalonamento e torná-la ainda mais precisa.

Em conclusão, a solução proposta por meio do desenvolvimento do software utilizando CP é uma solução eficiente para o problema do escalonamento de médicos, e a implementação de melhorias pode torná-lo ainda mais prático e preciso. O software pode ajudar a diminuir o tempo gasto com a elaboração da escala das equipes de saúde em hospitais, que ainda é feita manualmente e custa várias horas de trabalho de profissionais qualificados, além de melhorar a qualidade das escalas ao considerar preferências pessoais e institucionais dos funcionários e do hospital.

Referências

Bovet, D. P. and Crescenzi, P. (1994). *Introduction to the Theory of Complexity*. Prentice Hall.

Brunner, J., Bard, J., and Kolisch, R. (2009). Flexible shift scheduling of physicians. *Health Care Management Science*, 12:285–305.

Coello, C., Dhaenens, C., and Jourdan, L. (2009). *Advances in Multi-Objective Nature Inspired Computing*, volume 272. Springer Berlin, Heidelberg.

Dechter, R. (2003). *Constraint Processing*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Erhard, M., Schoenfelder, J., Fügener, A., and Brunner, J. O. (2018). State of the art in physician scheduling. *European Journal of Operational Research*, 265(1):1–18.

Fügener, A., Brunner, J. O., and Podtschaske, A. (2015). Duty and workstation rostering considering preferences and fairness: a case study at a department of anaesthesiology. *International Journal of Production Research*, 53(24):7465–7487.

Rossi, F., Beek, P. v., and Walsh, T. (2006). *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. Elsevier Science Inc., USA.