

Um Algoritmo de Enumeração Aplicado ao Problema de Distribuição de Socorro Médico

Tiago Araújo Neves

Universidade Federal Fluminense - UFF

Av. dos Trabalhadores, 420 - Vila Santa Cecília, Volta Redonda - RJ, 27255-125
tneves@id.uff.br

Raila de Medeiros Melo

Universidade Federal Fluminense - UFF

Av. dos Trabalhadores, 420 - Vila Santa Cecília, Volta Redonda - RJ, 27255-125
raila.medeiros@id.uff.br

Patricia Mendonça Maia Bernardes

Universidade Federal Fluminense - UFF

Av. dos Trabalhadores, 420 - Vila Santa Cecília, Volta Redonda - RJ, 27255-125
pbernardes@id.uff.br

RESUMO

Situações de desastre causam imensos prejuízos e trazem grande sofrimento à população afetada. No contexto das Cidades Inteligentes, é fundamental que os governos tenham planos para situações deste tipo e que estes planos sejam sólidos em termos de embasamento e efetividade. Ferramentas computacionais podem ajudar a validar e embasar os planos de ação nestas situações, tornando o uso destas técnicas imprescindível para Cidades Inteligentes. Este trabalho apresenta a aplicação de uma técnica de resolução exata para um problema de otimização ligado às situações de desastre que visa direcionar médicos para localidades onde suas especialidades médicas sejam necessárias, buscando reduzir o atraso nos atendimentos das pessoas afetadas pelo desastre. A técnica utilizada foi incorporada em um protótipo de *framework* e os resultados preliminares foram comparados com um dos melhores resolvidores do mercado, mostrando que a abordagem proposta consegue provar a otimalidade das soluções em ambientes computacionais com memória restrita.

PALAVRAS CHAVE. Distribuição de Socorro Médico, Cidades Inteligentes, Método exato.
Tópicos: Cidades Inteligentes, PO na saúde, Otimização Combinatória.

ABSTRACT

Disaster situations cause immense damage and bring great suffering to the affected population. In the context of Smart Cities, it is essential that the government provides effective and solid plans for situations like these. Computational tools can help to validate and substantiate action plans in these situations, making the use of these techniques essential for Smart Cities. This work presents the application of an exact technique to an optimization problem related to disaster situations. The objective is to create routes for doctors based on the demand for their medical specialties, minimizing the delays in care for people affected by the disaster. A framework prototype is created to implement used technique and preliminary results were compared with one of the best solvers on the market. The results show that the proposed approach can prove the solutions' optimality in restricted memory computational environments.

KEYWORDS. Medical Assistance Distribution, Smart Cities, Exact approach.

Paper Topics: Smart Cities, OR in Health, Combinatorial Optimization.

1. Introdução

É notório o dano causado por desastres ocorridos nos últimos anos. No Brasil, desastres causados por eventos de natureza climática como os ocorridos em São Paulo, em 2023, e neste ano no Rio Grande do Sul, ou mesmo de outra natureza como o ocorrido em Brumadinho, em 2019, causam imensos prejuízos, um número grande de mortes, deixam diversas pessoas desabrigadas e trazem profundo sofrimento à população local. De acordo com [CRED, 2024], desde 1990 houve mais de quinze mil desastres naturais em todo o mundo, atingindo cerca de oito bilhões de pessoas, gerando milhões de mortes e um prejuízo aproximado de cinco bilhões de dólares.

A preparação dos governos, e principalmente das cidades, no que diz respeito ao planejamento de ações preventivas e de resposta aos desastres é de extrema relevância, e o uso de modelos computacionais, matemáticos, físicos e de simulação tem ganhado importância neste contexto. Neste sentido, a discussão sobre o uso de técnicas computacionais para prevenção, resposta e reconstrução em situações de desastre também pode ser feita dentro do contexto das Cidades Inteligentes (CI). Cerca de 80% do Produto Nacional Bruto mundial é gerado em áreas urbanas, e metade da população global já reside em cidades, número esse que está projetado para chegar a cerca de 5 bilhões de pessoas até 2030 [Melo et al., 2023].

Normalmente, o foco das CIs é aprimorar a eficiência e o padrão de vida dos habitantes por meio de uma administração eficaz dos recursos, implementação de soluções sustentáveis e oferta de serviços inteligentes de alta qualidade e acessíveis, abordando problemas urbanos recorrentes por meio de ferramentas ligadas às tecnologias de informação [Caragliu et al., 2011]. A organização de uma CI engloba várias dimensões: economia, controle ambiental, mobilidade, governança, eficiência energética, serviços de saúde, entre outras [Muhammad et al., 2017]. No entanto, em função do impacto causado, situações de calamidade não podem ser ignoradas.

Este trabalho expõe resultados preliminares obtidos na continuação do estudo exposto em [Melo et al., 2023], que trata de um problema de otimização na fase de resposta, em situação pós-desastre. Em [Melo et al., 2023], os autores propuseram um modelo matemático e um algoritmo genético para o Problema de Distribuição de Socorro Médico (PDSM), que visa montar as rotas percorridas por médicos para atender os pacientes em abrigos em situação de desastres, considerando as demandas dos pacientes e as especialidades médicas. No estudo citado não foi possível encontrar as soluções ótimas para algumas instâncias usando o modelo matemático, o que motivou a concepção deste trabalho. Um protótipo de *framework* para construção de *softwares* de resolução de problemas combinatórios baseado em enumeração foi desenvolvido e aplicado ao PDSM. Os resultados preliminares mostram que a ferramenta possui bom desempenho, dadas as limitações do ambiente computacional utilizado.

Em um ambiente de desastre, é mais importante encontrar soluções de boa qualidade para os problemas em um curto período de tempo. Este tipo de situação exige respostas rápidas e, neste sentido, o uso de técnicas heurísticas é frequentemente mais adequado, vide os resultados apresentados em [Melo et al., 2023]. Contudo, encontrar as soluções ótimas para estes problemas também é relevante para medir a eficiência dos algoritmos heurísticos (para decidir qual algoritmo será usado em uma situação real de desastre) e para obter estratégias para serem incorporadas em novos algoritmos. Neste sentido, o uso de algoritmos exatos para testes pode ser considerado como parte da preparação para desastres, o que motiva seu estudo e aplicação.

O artigo está organizado da seguinte forma: a Seção 2 apresenta a definição do problema e a literatura relacionada. A Seção 3 expõe a formulação matemática para o problema. A Seção 4 descreve a proposta deste trabalho. Os resultados parciais estão expostos na Seção 5. Já as conclusões parciais e os trabalhos futuros são apresentados na Seção 6.

2. Descrição do Problema e Literatura Relacionada

As atuações em Logística Humanitária (LH) podem ser melhor compreendidas através das fases dos desastres. Conforme Hezam et al. [2021] são quatro fases divididas em duas categorias temporais, pré-desastre e pós-desastre. Na categoria Pré-desastre tem-se as fases de preparação e mitigação. Na fase de preparação, é esperado que os governantes adotem abordagens proativas para garantir uma estrutura adequada para o enfrentamento de desastres, e que as pessoas recebam orientações (treinamento) para melhor enfrentar a situação. Na fase de mitigação, espera-se reduzir as vulnerabilidades das áreas que podem ser afetadas, objetivando minimizar a perda de vidas e danos a propriedades. As fases restantes, resposta e recuperação, pertencem à categoria pós-desastre. Na fase de resposta, são postas em prática as ações previstas na fase de preparação, e também são tomadas decisões críticas para situações não planejadas. Uma abordagem eficiente nesta fase pode minimizar a perda de vidas e os prejuízos econômicos. Já a fase de recuperação possui o foco de tratar os danos causados, visando restabelecer a infraestrutura e dar solução definitiva para situação das pessoas afetadas pelo desastre. Nesta fase também é avaliada a eficácia das medidas planejadas, mostrando que as fases estão correlacionadas de modo cíclico.

A LH tem três grandes focos: 1) localização de instalações; 2) distribuição de socorro/alívio; e 3) evacuação em massa. O PDSM está ligado à distribuição de socorro/alívio, que é feita na fase de resposta ao desastre. Entretanto, o foco deste trabalho está ligado à fase de preparação, onde busca-se escolher as técnicas e as estratégias que serão usadas em caso de desastre.

Existem trabalhos ligados ao tema de distribuição de alívio [Hezam et al., 2021] que focam na distribuição de itens (remédios, comida, água, material de higiene, etc), contudo, este não é o foco do problema abordado neste trabalho.

No que tange a distribuição de socorro médico, o trabalho de Melo et al. [2023] e a revisão literária de Hezam et al. [2021] apontam que apenas dois estudos utilizam modelos determinísticos, que são Wang et al. [2018] e Lei et al. [2015], o que mostra que este tópico ainda não recebeu muita atenção da literatura. Em Wang et al. [2018] os autores desenvolveram um dos primeiros estudos sobre agendamento da equipe médica (roteirização) e distribuição de ajuda humanitária como variáveis de decisão no campo da logística humanitária. No problema analisado, as equipes de atendimento médico são enviadas e os suprimentos de socorro são distribuídos entre os pontos de demanda. Os autores desenvolveram um modelo de programação inteira mista e uma meta-heurística híbrida de dois estágios para resolver o problema. Em Lei et al. [2015] é estudado um problema de agendamento de operações emergenciais que tem como objetivo minimizar o atraso total das operações a serem realizadas, sujeitas à disponibilidade de equipes médicas e dos suprimentos médicos. Uma heurística multiperíodo baseada em programação matemática é utilizada. Os autores citam que, durante o episódio do furacão Katrina em 2005, os governos federais e locais lançaram seus sistemas de resposta emergenciais para ajudar as pessoas afetadas, porém o gargalo mais crítico percebido foi a falta de disponibilidade de enfermeiros e devido a isso os atendimentos não podiam ser realizados.

Mais recentemente, em Melo et al. [2023] foram propostos uma formulação matemática, baseada em Programação Linear Inteira (PLI), e um algoritmo genético de chaves aleatórias viciadas para o PDSM, que é o foco deste trabalho. O estudo analisa a distribuição de socorro considerando as especialidades de cada médico e as demandas por estas especialidades. De acordo com o conhecimento dos autores, este é o único trabalho da literatura que leva este aspecto em consideração.

A descrição formal do PDSM pode ser feita do seguinte modo: em uma situação de desastre há diversos pacientes que necessitam de ajuda médica. Cada paciente possui demanda por uma especialidade e uma posição geográfica específica. Além disso, cada paciente possui um tempo limite para o início do atendimento, assim como um tempo estimado para a conclusão de seu aten-

dimento, ambos em função da gravidade do caso (pacientes mais graves possuem um limite de atendimento menor e demandam mais tempo de cuidado). As informações dos pacientes são passadas para uma central de informações (normalmente uma base de operações ou um centro médico), que é responsável por tratar os dados e estabelecer a ordem dos atendimentos considerando os recursos humanos disponíveis. No processo de roteirização, a central de informações deve considerar a quantidade de médicos, as especialidades (ortopedia, pediatria, etc) de cada médico, o tempo máximo de uma rota (para evitar a exaustão da equipe médica), o tempo de descanso entre rotas (este tempo pode variar em função de condições individuais de cada médico, por exemplo idade, restrições físicas, cuidados especiais, etc) e o tempo de deslocamento entre cada par de localidades. Considerando estes dados, a central deve determinar a rota a ser seguida por cada médico e enviar aos pontos de demanda os horários estimados para os atendimentos. Todas as demandas devem ser atendidas e o atraso global da operação, usado como função objetivo, deve ser minimizado. Assim, o problema abordado otimiza o atendimento médico observando as demandas por especialidades e priorizando os pacientes em estado crítico, enquanto garante que a equipe médica não seja exaurida. Trata-se de um problema de suma importância em situações de desastre, uma vez que múltiplas pessoas necessitam de auxílio nestas situações e os gestores devem tomar decisões rapidamente, visando preservar vidas e minimizar perdas. Portanto, a criação de ferramentas robustas e confiáveis para ajudar na tomada de decisão nestes momentos críticos é extremamente relevante.

Em Melo et al. [2023], os autores propuseram uma formulação matemática capaz de resolver instâncias pequenas do problema, contudo, o desempenho não se repetiu para as instâncias maiores. Apesar de encontrar soluções viáveis, a otimalidade não foi provada para instâncias com mais de 10 pacientes. Estes resultados motivaram a criação de um algoritmo enumerativo alternativo para o PDSM. Algoritmos enumerativos para otimização são bem conhecidos e suas aplicações têm sido feitas com sucesso. Para casos mais gerais, os algoritmos do tipo *Branch-And-X* [Zionts, 1974] (como o *Branch-And-Bound* e *Branch-And-Cut* por exemplo) têm sido aplicados na resolução de diversos problemas, e são a base dos resolvidores de PLI disponíveis no mercado. Entretanto, estes algoritmos apresentam grande consumo de memória em alguns casos e também estão sujeitos a erros de truncamento e instabilidade numérica pelo fato de trabalharem no espaço contínuo. O grande consumo de memória é uma das causas (talvez a principal) da formulação proposta por [Melo et al., 2023] não ter conseguido provar o ótimo para as instâncias mencionadas.

Neste sentido, para evitar os problemas apresentados pelos algoritmos *Branch-And-X*, alguns autores elaboram algoritmos enumerativos no espaço discreto. Em Ghaffarinasab e Atayi [2018] os autores propõem dois algoritmos de enumeração para um Problema de Interdição de *Hubs* com Fortificação. Neste problema existem p *hubs* (centrais de comunicação) das quais se sabe que r serão atacados e inutilizados. Há recursos para proteger um total de q *hubs*, de modo que quando protegido, um *hub* não pode ser destruído. O objetivo é escolher quais *hubs* devem ser protegidos de forma que, após o ataque, o prejuízo ao sistema de comunicação seja minimizado. Os autores propõem uma formulação baseada em PLI, e dois algoritmos de enumeração, um explícito, onde todas as possibilidades são geradas e analisadas, e um implícito, onde algumas combinações não são analisadas por questões de qualidade de solução ou por questões de inviabilidade. Deve-se ressaltar que os algoritmos do tipo *Branch-And-X*, usados para resolver as formulações matemáticas, são do tipo implícito, e portanto não fazem enumerações desnecessárias. Os resultados mostram que os algoritmos de enumeração propostos são competitivos quando comparados a um resolvidor de propósito geral (que usa um algoritmo *Branch-And-X*) em casos mais simples. Já para casos mais complexos, o resolvidor acaba por consumir toda a memória disponível enquanto os algoritmos de enumeração continuam sendo capazes de fornecer as soluções ótimas.

Em Neves et al. [2012], os autores criaram um algoritmo de enumeração para um pro-

blema de alocação ligado ao gerenciamento de redes de distribuição de conteúdo. O objetivo do algoritmo era prover soluções exatas em situações onde a formulação baseada em PLI desenvolvida não era bem sucedida. Os resultados mostram que, para instâncias menores, a formulação consegue provar os valores ótimos em menos tempo. Contudo, para instâncias maiores o algoritmo de enumeração é a melhor opção.

Em Smith et al. [2012] os autores propõem um algoritmo de enumeração implícito para encontrar a decomposição ótima de Grafos em Ramos. De acordo com os autores, este problema possui relevância pelo fato de que algumas técnicas para resolução de problemas da classe NP-Difícil usam decomposição em ramos. Foram feitos testes computacionais em uma gama de instâncias da literatura. Os resultados obtidos foram comparados com outro algoritmo exato, mostrando que a técnica utilizada obteve resultados competitivos com os melhores da literatura.

Portanto, como exposto, pode-se verificar que algoritmos de enumeração podem obter bons resultados quando aplicados de modo coerente. Este fato encorajou a elaboração da abordagem proposta neste trabalho que será descrita mais adiante no texto. A próxima seção apresenta o modelo matemático do problema tratado, usado para comparação com o algoritmo proposto.

3. Modelagem Matemática

Esta seção apresenta uma formulação matemática para o PDSM proposta em [Melo et al., 2023] e aqui reproduzida por razões didáticas.

O modelo assume as seguintes premissas: 1) cada paciente está em uma localidade distinta. Caso exista mais de um paciente em uma mesma localidade geográfica, eles devem ser modelados como vértices diferentes com tempo de deslocamento nulo entre si; 2) No caso de um paciente com múltiplas demandas (demanda por múltiplas especialidades médicas), transforma-se este paciente em múltiplos pacientes de demanda única com tempo de deslocamento zero entre si; e 3) cada rota começa e termina no centro médico.

Seja $G(N, A)$ um grafo direcionado onde N é o conjunto de vértices e A é o conjunto de arestas. Seja $N = \{o, d, 1, 2, 3, \dots, n\}$ o conjunto de localidades, sendo que o e d representam o centro médico, mas o marca o início de uma rota e d marca o fim). As arestas $(i, j) \in A$ representam o tempo de viagem entre uma localidade (paciente) i e uma localidade j .

Sejam também $N' = N \setminus \{o, d\}$; $A = \{(i, j) : i, j \in N, i \neq j\}$, o conjunto de arestas; $K = \{1, \dots, k\}$, o conjunto de médicos; $P = \{1, \dots, p\}$, o conjunto de especialidade médica; $R = \{1, \dots, r\}$, o conjunto de rotas de cada médico; B_i , o tempo limite para início do atendimento do paciente i (função da gravidade do caso); T_{ij} , o tempo de deslocamento entre as localidades onde estão os pacientes i e j ; S_i , o tempo de atendimento do paciente i (tempo estimado em função da especialidade requerida); C_k , o tempo necessário para descanso entre viagens para o médico k ; Q_k , o limite de tempo para rotas do médico k ; H_{kp} uma constante que assume 1 se o médico k possui a especialidade p e 0 caso contrário (c.c.); D_{ip} uma constante que assume 1 se o paciente i demanda a especialidade p e 0, c.c. ; Sejam também as seguintes variáveis: Y_{kri} , o atraso no atendimento do paciente i na r -ésima rota do médico k ; W_{kri} , o tempo de início do atendimento do paciente i na r -ésima rota do médico k ; e X_{krij} a variável binária que assume 1 se o paciente j é atendido imediatamente após o paciente i na r -ésima rota do médico k , e 0 caso contrário.

Uma possível formulação matemática para o PDSM é apresentada em (1) – (12).

$$\text{Min } Z = \sum_{i=1}^{N'} \sum_{k=1}^K \sum_{r=1}^R Y_{kri} \quad (1)$$

s.a.

$$Y_{kri} \geq W_{kri} - B_i, \quad \forall i \in N', \forall k \in K, \forall r \in R; \quad (2)$$

$$W_{kri} + S_i + T_{i,j} - W_{krj} \leq (1 - X_{krij})M, \quad \forall k \in K, \forall r \in R, \quad \forall i, j \in N, i \neq j; \quad (3)$$

$$W_{k(r+1)o} \geq W_{krd} + C_k, \quad \forall k \in K, \forall r \in \{1, \dots, (|R| - 1)\}; \quad (4)$$

$$W_{krd} - W_{kro} \leq Q_k, \quad \forall k \in K, \forall r \in R; \quad (5)$$

$$W_{kro} \leq W_{krd}, \quad \forall k \in K, \forall r \in R; \quad (6)$$

$$\sum_{j=1}^{N \setminus \{d\}} X_{krji} - \sum_{j=1}^{N \setminus \{o\}} X_{krij} = 0, \quad \forall k \in K, \forall r \in R, \forall i \in N'; \quad (7)$$

$$\sum_{k=1}^K \sum_{r=1}^R \sum_{j=1}^{N \setminus \{d\}} X_{krji} = 1, \quad \forall i \in N'; \quad (8)$$

$$\sum_{j=1}^{N \setminus \{o\}} X_{kroj} \leq 1, \quad \forall k \in K, \forall r \in R; \quad (9)$$

$$\sum_{j=1}^{N'} X_{kroj} = \sum_{j=1}^{N'} X_{krjd}, \quad \forall k \in K, \forall r \in R; \quad (10)$$

$$X_{krij} = 0, \quad \forall i \in N', \forall p \in P, \forall k \in \{z : z \in K, D_{ip} \neq 0, H_{kp} = 0, \forall j \in N, \forall r \in R\}; \quad (11)$$

$$Y_{kri}, W_{kri}, X_{krij} \geq 0 \text{ e } Y_{kri}, W_{kri}, X_{krij} \in \mathbb{Z}, \quad \forall i, j \in N, \forall p \in P, \forall k \in K, \forall r \in R. \quad (12)$$

A função objetivo (1) visa minimizar os atrasos em todas as localidades. As restrições (2) são responsáveis pelo cálculo do atraso. Já as restrições (3) certificam que, se o paciente j é atendido pelo médico k em sua r -ésima viagem imediatamente após o paciente i , então o tempo de início do atendimento de j deve ser igual ao tempo de início do atendimento de i , somado com o tempo de atendimento de i e com o tempo de deslocamento entre i e j . Já as restrições (4) garantem que o médico k só pode começar uma nova rota após o descanso obrigatório. As restrições (5) e (6) garantem que o tempo limite das rotas seja respeitado e que o tempo de chegada ao destino seja maior (ou igual no caso de rota vazia) que o tempo de partida na origem. O conjunto de restrições (7) trata o equilíbrio de fluxo. As restrições (8) garantem que todos os pacientes são visitados uma única vez, enquanto as restrições (9) limitam o número de partidas por rota a partir da origem. Já as restrições (10) certificam que se uma viagem foi iniciada, ela deve ser finalizada. As restrições (11) garantem que um paciente não seja atendido por um médico sem a especialidade requerida. Por fim, as restrições (12) tratam a não negatividade e integralidade do problema.

Os resultados obtidos por esta formulação foram comparados com os resultados do método proposto, visando validar o protótipo construído. Uma descrição mais detalhada do método proposto e os resultados obtidos estão descritos mais adiante no texto.

Figura 1: Algoritmo de Enumeração por Ramos

Algoritmo 1 Enumerar(x :xs, soluçãoIncumbente, ramo)

```

1: se ( $x$  é nulo) então
2:   se ( $\text{custo}(\text{ramo}) < \text{custo}(\text{soluçãoIncumbente})$  & ramo é viável) então
3:     atualizar soluçãoIncumbente
4:   fim se
5: senão
6:   se ( $\text{custo}(\text{ramo}) \leq \text{custo}(\text{soluçãoIncumbente})$  & ramo é viável) então
7:      $r0 = \text{ramo} + \text{criarRamo}(x=0)$ 
8:     Enumerar(  $x$ s, soluçãoIncumbente,  $r0$ )
9:      $r1 = \text{ramo} + \text{criarRamo}(x=1)$ 
10:    Enumerar(  $x$ s, melhorSolução,  $r1$ )
11:   fim se
12: fim se

```

Fonte: Autores

4. Algoritmo Enumerativo

A solução proposta é um *framework*, aqui referido como SEF, e tem como base um algoritmo de enumeração do tipo implícito. Um ponto a ser ressaltado é que a intenção da ferramenta proposta não é competir com os resolvedores de PLI nos casos onde eles são efetivos. O objetivo é tentar fornecer uma ferramenta capaz de abordar os casos onde estes resolvedores apresentam dificuldade, seja por questões de tempo computacional ou por questões de consumo de memória.

Uma primeira versão deste algoritmo, chamado de SEA, foi proposta para um problema de alocação aplicado no contexto de redes de distribuição de conteúdos [Neves et al., 2012]. Em Melo et al. [2023] o PDSM foi modelado matematicamente mas não foi possível provar a otimalidade para todas as instâncias utilizando PLI. Devido a este fato, os autores começaram a trabalhar em uma versão do SEA para o PDSM, com o objetivo de usá-lo nas instâncias que ainda não tinham a otimalidade provada. Durante o desenvolvimento, percebeu-se que o algoritmo poderia ser generalizado para vários problemas, o que motivou a construção de uma nova versão do SEA e sua implementação no SEF.

O algoritmo proposto constrói as soluções incrementalmente, atribuindo valores para as variáveis binárias, de modo que, para atingir uma solução completa, várias soluções parciais são analisadas. No estado atual, a aplicação do algoritmo ainda é restrita a casos que obedeçam alguns requisitos: 1) o problema deve ser modelado na forma de variáveis binárias; 2) decisões sobre valores de variáveis tomadas posteriormente não podem desfazer inviabilidades causadas por decisões anteriores; e 3) decisões tomadas posteriormente não podem alterar o custo de decisões anteriores na função objetivo. A Figura 1 apresenta de forma simplificada a ideia base do SEA para um problema de minimização.

A estratégia consiste em descrever o espaço de soluções como uma árvore binária de decisões e explorar a árvore para encontrar o ótimo global. A árvore em questão representa o espaço discreto de decisões, o que elimina problemas de instabilidade numérica e de truncamentos existente em alguns algoritmos que utilizam árvores para descrever o espaço contínuo, como por exemplo o *branch-and-bound* [Zionts, 1974]. Um ponto chave para esta estratégia é como explorar a árvore de maneira eficiente. Para isso, utiliza-se mecanismos de poda, que impedem a exploração de ramos que gerem soluções inviáveis ou que não irão produzir soluções de boa qualidade.

A árvore que descreve o espaço de soluções representa possíveis valores das variáveis binárias. Esta árvore é binária e completa com altura igual ao número de variáveis binárias a serem enumeradas, denominado l , e 2^l folhas. Uma solução completa para problema de otimização é obtida percorrendo um ramo da árvore, que consiste no caminho entre a raiz e uma folha. Isto significa que para atingir uma solução completa para o problema, l decisões binárias precisam ser

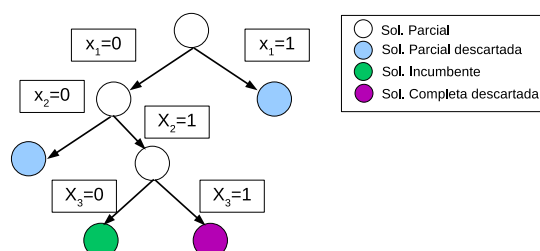
tomadas. A solução ótima do problema pode ser obtida por meio de uma busca em todos os ramos da árvore. Contudo, esta árvore pode se tornar demasiadamente grande mesmo para instâncias de pequeno porte, e por esta razão, construir toda a árvore não é razoável. Para explorar a estrutura de um modo mais eficiente, optou-se por construir a árvore sob demanda, o que significa que apenas um ramo é utilizado por vez. Essa característica faz com que a demanda do algoritmo por memória seja pequena, uma vez que o algoritmo trabalha com apenas um ramo por vez e cada ramo tem tamanho máximo l , a complexidade de espaço requerida se torna $O(l)$. A estratégia de exploração da árvore, mostrada na Figura 1 usa os seguintes passos: 1) Criar uma lista contendo as variáveis a serem enumeradas. 2) Suponha que a notação $x : xs$ representa uma lista com nó cabeça x e cauda xs . Seja $Enumerar(x : xs, soluçãoIncumbente, ramo)$ uma função recursiva que recebe um ramo, a solução incumbente (melhor solução encontrada até o momento) e uma lista de variáveis. Esta função remove o elemento na cabeça da lista (x), liga este elemento em $ramo$ criando dois novos ramos, chamados de $r0$ e $r1$, que representam, respectivamente, a atribuição do valor 0 para variável x e a atribuição do valor 1 para variável x . A função $Enumerar$ então é invocada recursivamente para ambos os ramos, usando a lista xs e a solução incumbente como parâmetros ($Enumerar(xs, soluçãoIncumbente, r0)$ e $f(xs, soluçãoIncumbente, r1)$). A função $Enumerar$ encerra o processo quando a lista recebida como parâmetro é vazia (primeiro elemento nulo), ou quando alguma inviabilidade é encontrada.

Apesar de uma busca ramo a ramo na árvore ser perfeitamente possível, pode levar muito tempo para explorar toda a estrutura. Devido a isto, mecanismos de poda são utilizados com o objetivo de evitar a exploração desnecessária de alguns ramos da árvore e assim reduzir o tempo computacional necessário para encontrar as soluções ótimas. A cada nova decisão tomada, são checadadas a viabilidade da solução e se vale a pena continuar enumerando as soluções a partir da decisão tomada. Se a solução parcial é inviável e o caso atende ao requisito 2, então nenhuma outra decisão tomada posteriormente tornará a solução corrente viável, logo, continuar enumerando as variáveis desta solução não irá gerar resultados válidos e portanto a recursão pode ser interrompida. De modo análogo, se a solução parcial possui um custo pior que a solução incumbente e o caso atende ao requisito 3, continuar enumerando as variáveis da solução corrente acarreta um desperdício de tempo, uma vez que nenhum resultado melhor que a solução incumbente será encontrado.

A Figura 2 exemplifica a árvore construída para um problema de minimização de três variáveis (x_1 , x_2 e x_3), com Função Objetivo (F.O.) dada por $z = 10x_1 + x_2 + x_3$ e uma restrição $x_1 + x_2 = 1$. Seguindo os passos do algoritmo da Figura 1, a primeira solução parcial seria encontrada com $x_1 = 0$. Ao tentar fazer $x_2 = 0$, o algoritmo gera uma solução inviável, e por isso a enumeração envereda pelo ramo onde $x_2 = 1$. Neste ponto, ao fazer $x_3 = 0$ a primeira solução completa é encontrada com $z = 1$. Ao verificar o ramo $x_3 = 1$, outra solução completa é encontrada com $z = 2$. Como este é um problema de minimização e já há uma solução completa de melhor qualidade esta nova solução é descartada. Por último, o algoritmo tenta enumerar o ramo onde $x_1 = 1$. Neste ponto, a solução parcial possui $z = 10$, e continuar a enumeração deste ramo é desnecessário, visto que as soluções completas produzidas por ele serão piores que a solução incumbente. Como não há mais possibilidades para enumerar, o algoritmo retorna como solução ótima $x_1 = 0$, $x_2 = 1$ e $x_3 = 0$.

O *framework* proposto permite incluir restrições de viabilidade de forma similar ao que é feito nos resolvidores de PLI existentes no mercado. Alguns mecanismos padronizados de restrições estão disponíveis, como por exemplo restrições que lidam com somatório de variáveis de decisão, restrições que lidam com valor de uma variável previamente conhecido, entre outros. Os mecanismos de restrição também podem ser criados pelo programador, dando mais flexibilidade à ferramenta. As inviabilidades também podem ser checadadas durante a avaliação da função objetivo,

Figura 2: Exemplo de Aplicação do Método



Fonte: Autores

o que flexibiliza ainda mais a análise de casos de inviabilidade.

Nos casos em que somente os mecanismos de restrição pré-disponíveis são utilizados, o programador só precisa criar um único componente de *software* para utilizar o *framework*, que é o componente de cálculo da função objetivo. Isso permite um rápido desenvolvimento de *software* para problemas distintos.

Outro potencial desta ferramenta está em sua flexibilidade em relação aos problemas que ela pode abordar. Como exemplo, pode-se citar os componentes de restrição. A PLI permite modelar somas de variáveis de decisão sem muitas dificuldades. Entretanto, ela não permite que variáveis de decisão sejam multiplicadas. Já o *framework* proposto permite a análise de multiplicação de variáveis na modelagem das restrições, assim como também possibilita esta multiplicação no cálculo da função objetivo. Isso permite que o *framework* proposto também possa ser usado para alguns casos de programação não linear, desde que eles respeitem os requisitos expostos no início desta seção. Como o SEF ainda é um protótipo, ele não foi testado para casos de programação não linear. Estudos para este fim serão objeto de trabalhos futuros.

Algumas observações são necessárias neste ponto. O *framework* permite o uso de *hot start* (uso de solução inicial fornecida por outro algoritmo), mas toma as devidas precauções quando a técnica não é utilizada. Também é preciso ressaltar que o algoritmo proposto está exposto de forma recursiva por questões didáticas. O objetivo é apresentar a ideia geral da proposta. O *framework* também permite paralelização. Na versão atual, o paralelismo é feito por *threads* de modo não automatizado, implicando que o programador deve decidir qual ramo da árvore será explorado por cada *thread*. A implementação de um mecanismo de paralelização automatizado será objeto de estudos futuros. Outro ponto a ser mencionado é que o *framework* permite concepção de métodos híbridos (integração com métodos heurísticos e outros métodos exatos). Testes usando o SEF para enumerar uma parte da solução e um resolvidor de PLI para calcular outra parte já estão em andamento. Os testes mostram que é possível inclusive distribuir as *threads* disponíveis de maneira flexível entre o *framework* e o resolvidor de PLI.

5. Resultados Preliminares

Esta seção apresenta os resultados obtidos com a aplicação do *framework* proposto ao PDSM. É importante ressaltar que estes são resultados preliminares, uma vez que o *framework* ainda está em desenvolvimento.

Para os testes computacionais foram usadas 9 instâncias de [Melo et al., 2023]. Todas com quatro médicos e número de pacientes entre 5 e 15. Três delas usam a Distribuição Normal para determinar as demandas dos pacientes por especialidades. Três usam a Distribuição uniforme e três usam a Distribuição Zipf. Para resolver a formulação matemática foi utilizado o resolvidor CPLEX versão 12.10 [IBM, 2019]. O ambiente computacional utilizado foi um computador com processador i7 de 12 núcleos com 3.7 GHz por núcleo, rodando Linux *kernel* 5.4. A memória

utilizada para os testes foi limitada a 8 GB. Apesar de ter 12 núcleos disponíveis, apenas 10 foram utilizados nos experimentos. O tempo limite para todos os experimentos foi 21600 segundos. É importante mencionar que este é o tempo máximo de início de uma enumeração para o software proposto, mas enumerações em andamento não são interrompidas.

A Tabela 1 mostra os resultados obtidos para instâncias de até 15 pacientes. A primeira coluna indica a instância tratada. Na segunda, terceira e quarta colunas estão respectivamente o valor da função objetivo, o tempo computacional (em segundos) e o *gap* (em porcentagem) obtidos pelo CPLEX *gap* 0 indica ótimo provado). Na quinta, sexta e sétima colunas estão respectivamente o valor da função objetivo, o tempo computacional (em segundos) e os *status* obtidos pelo SEF. O status indica se o ótimo foi provado ou não. Nos experimentos expostos na Tabela 1 o algoritmo genético de chaves aleatórias viciadas padrão Londe et al. [2024], exposto em [Melo et al., 2023] como BRKGA, foi usado como *hot start*. Os resultados do BRKGA estão expostos nas colunas oito e nove para melhor exposição dos resultados. É importante ressaltar que o uso de *hot start* não invalida a comparação entre os resultados das abordagens visto que o CPLEX também usa heurísticas para resolução de problemas inteiros [IBM, 2019], e que o BRKGA é um algoritmo geral, sem operadores específicos. Em negrito estão os melhores resultados conhecidos (colunas F.O.) e os casos em que a otimalidade foi provada (colunas Gap e Status).

Tabela 1: Resultados Computacionais

Instância	CPLEX			SEF			BRKGA	
	F.O.	Tempo (s)	Gap (%)	F.O.	Tempo (s)	Status	F.O.	Tempo (s)
normal-m4-p5	110	0.08	0	110	1.27	Prov.	120	1.02
uniforme-m4-p5	87	0.04	0	87	2.28	Prov.	87	2.00
zipf-m4-p5	140	0.09	0	140	1.29	Prov.	140	1.03
normal-m4-p10	288	3.01	0	288	9.85	Prov.	295	3.00
uniforme-m4-p10	288	2.81	0	288	8.84	Prov.	318	3.00
zipf-m4-p10	382	13.39	0	382	8.96	Prov.	387	3.03
normal-m4-p15	834	848.48	82.55	809	19043.08	Prov.	834	7.03
uniforme-m4-p15	742	3971.98	73.33	752	30229.85	Não Prov.	765	6.05
zipf-m4-p15	903	3207.38	81.69	903	23206.4	Não Prov.	927	6.02

Pelos resultados expostos na Tabela 1, para as instâncias com até 10 pacientes, a formulação matemática é a melhor opção, visto que ela consegue provar a otimalidade em um tempo inferior em 5 do 6 casos. Os dois algoritmos foram capazes de provar a otimalidade para estas instâncias.

Já para as instâncias de 15 pacientes, a solução ótima só foi provada em um único caso (normal-m4-p15) pelo método proposto. Para esta mesma instância, o CPLEX não foi capaz de encontrar a solução ótima, ficando 25 unidades de custo pior que a solução encontrada pelo método proposto. Para as outras duas instâncias, os dois algoritmos encontraram a mesma solução, mas nenhum deles foi capaz de provar a otimalidade.

Para estes resultados devem ser feitas algumas observações sobre o tempo computacional. Os valores apresentados na Tabela 1 referentes ao SEF são a soma dos tempos obtidos pelo algoritmo genético e pelo algoritmo de enumeração. Contudo, é preciso dizer que nas instâncias com 5 pacientes, o tempo do algoritmo genético supera em muito o tempo do algoritmo de enumeração. Como exemplo pode-se citar a instância normal-m4-p5 onde o algoritmo genético levou 1.02 segundos para concluir sua execução, enquanto o SEF levou 0.25 segundos. Estudos sobre a influência do algoritmo usado como *hot start*, visando mitigar esta questão serão objeto de trabalhos futuros. Já para o CPLEX, os tempos apresentados para as instâncias com gap maior que zero indicam que a execução foi encerrada por falta de memória disponível.

Figura 3: Captura de Tela Durante a Execução do SEF

%CPU	%MEM	TIME+	COMMAND
1001	0.0	1219:18	exeMR-M+
0.3	0.4	0:13.94	chrome
0.3	0.9	0:17.27	chrome
0.3	0.0	0:08.94	kworker+
0.0	0.0	0:02.83	systemd

Fonte: Autores

Os resultados para instâncias com até 10 pacientes mostram que o SEF tem potencial, uma vez que estes são resultados preliminares, e o desempenho do método ainda pode melhorar. Já os resultados para instâncias de 15 pacientes não confirmam este potencial, visto que, apesar do método proposto conseguir provar que a solução encontrada é ótima na instância normal-m4-p15, nenhum dos algoritmos conseguiu provar a otimalidade para a maioria dos casos.

Neste sentido foi proposto um segundo experimento, reduzindo o tamanho das instâncias utilizadas. Para cada instância de 15 pacientes o primeiro médico teve o primeiro paciente da primeira rota definido *a priori*. Para isso, uma das variáveis x da formulação teve seu valor definido via restrição. Para este experimento, a técnica de *hot start* não foi utilizada. Os resultados estão expostos na Tabela 2. As colunas possuem o mesmo significado da Tabela 1.

Tabela 2: Resultados Computacionais para as Instâncias Reduzidas

Instância	CPLEX			SEF		
	F.O.	Tempo (s)	Gap (%)	F.O.	Tempo (s)	Status
normal-m4-p15r	915	605,15	72,72	915	1882,50	Prov.
uniforme-m4-p15r	813	3729,41	62,37	813	13247,73	Prov.
zipf-m4-p15r	1041	1835,82	74,98	1041	5939,70	Prov.

A Tabela 2 mostra que o SEF consegue provar a otimalidade em todos os casos, enquanto CPLEX é capaz de encontrar a solução ótima, mas não de provar a otimalidade devido ao consumo de memória. Já a eficiência no uso de memória do método proposto pode ser confirmada pela Figura 3, que mostra uma captura de tela do gerenciador *TOP* durante a execução do método.

É importante ressaltar, como pode ser visto na Figura 3, que o programa que implementa o método proposto, denominado exeMR.mep, mesmo com 10 *threads* executando simultaneamente, possui um consumo de memória muito baixo, mesmo após vários minutos de execução.

6. Conclusões Parciais e Trabalhos Futuros

Apesar de ser uma pesquisa ainda em desenvolvimento, o trabalho feito até o momento mostra que a pesquisa obteve sucesso em seus objetivos. Desenvolveu-se uma ferramenta que utiliza uma técnica de enumeração, capaz de atingir resultados competitivos com um dos melhores resolvidores do mercado para um problema de grande importância no âmbito das CIs. Enfatiza-se que competir com os resolvidores de PLI não era o objetivo inicial da pesquisa, sendo este buscar uma alternativa para casos onde tais resolvidores não obtivessem bons resultados. Os resultados obtidos, neste sentido, foram melhores que o esperado, visto que o SEF conseguiu resultados melhores que o CPLEX em alguns casos. Contudo, muitos testes ainda precisam ser feitos para validar a ferramenta, uma vez que ela ainda está em desenvolvimento.

Até o momento, pode-se afirmar que a técnica proposta possui baixa demanda por memória, grande flexibilidade em termos de modelagem de restrições para problemas de otimização e também grande potencial de paralelismo em relação ao uso de *threads*, além de ser livre de erros

de truncamento e de instabilidade numérica. O baixo consumo de memória era um dos objetivos principais da pesquisa, visto que computadores com grande volume de memória não costumam fazer parte do acervo disponível nas prefeituras da maioria das cidades. Neste sentido, elaborar algoritmos que possam ser executados em máquinas de menor poder computacional melhora o alcance de potenciais beneficiados da pesquisa. Existem também potenciais pouco explorados como o uso da ferramenta para problemas de otimização não lineares, compartilhamento das *threads* entre estratégias híbridas e paralelismo multimáquina. Contudo, muito ainda precisa ser feito para tornar a ferramenta robusta e utilizável por um usuário padrão.

Como trabalhos futuros, pode-se citar o aprimoramento dos mecanismos atuais do *framework*. Muitos dos mecanismos usados atualmente podem ser melhorados para redução de tempo computacional e redução do tempo de implementação de aplicações. A criação de um mecanismo de paralelização que utilize *threads* de modo independente do programador é o foco do trabalho neste momento. Também pode-se citar o uso do *framework* em outros problemas para validação.

Referências

- Caragliu, A., Bo, C. D., e Nijkamp, P. (2011). Smart cities in europe. *Journal of Urban Technology*, 18(2):65–82.
- CRED (2024). The International Disaster Database. <https://www.emdat.be>. Acessado em 05/2024.
- Ghaffarinasab, N. e Atayi, R. (2018). An implicit enumeration algorithm for the hub interdiction median problem with fortification. *European Journal of Operational Research*, 267:23–39.
- Hezam, I. M. et al. (2021). A systematic literature review on mathematical models of humanitarian logistics. *Symmetry*, 13(1).
- IBM (2019). IBM ILOG CPLEX Optimization Studio user's manual: version 12.10.
- Lei, L., Pinedo, M., Qi, L., Wang, S., e Yang, J. (2015). Personnel scheduling and supplies provisioning in emergency relief operations. *Annals of Operations Research*, 235(1):487–515.
- Londe, M. A., Pessoa, L. S., Andrade, C. E., e Resende, M. G. (2024). Biased random-key genetic algorithms: A review. *European Journal of Operational Research*.
- Melo, R. M., Neves, T. A., e Christo, E. S. (2023). Roteamento de médicos por especialidade em situação de pós-desastre através de um modelo heurístico e matemático. In *Anais do LV Simpósio Brasileiro de Pesquisa Operacional (SBPO 2023)*, Rio de Janeiro. SOBRAPO.
- Muhammad, G., Alsulaiman, M., Amin, S. U., Ghoneim, A., e Alhamid, M. F. (2017). A facial-expression monitoring system for improved healthcare in smart cities. *IEEE Access*, 5:10871–10881.
- Neves, T., Ochi, L., e Albuquerque, C. (2012). Soluções exatas para o problema de replicação e distribuição de requisições em redes de distribuição de conteúdos. In *Anais do 30º Simpósio Brasileiro de Redes de Computadores SBRC 2012*, Porto Alegre. SBC.
- Smith, J. C., Ulusal, E., e Hicks, I. V. (2012). A combinatorial optimization algorithm for solving the branchwidth problem. *Computational Optimization and Applications*, 51:1211–1229.
- Wang, S., Liu, F., Lian, L., Hong, Y., e Chen, H. (2018). Integrated post-disaster medical assistance team scheduling and relief supply distribution. *The Intern. Journal of Logistics Management*.
- Zionts, S. (1974). *Linear and Integer Programming*. Prentice-Hall.