LVI SBPO SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL
DE 04 ATÉ 07 NOVEMBRO 2024
FORTALEZA | CEARÁ

UMA ABORDAGEM MULTI-OBJETIVO DO PROBLEMA DE ALOCAÇÃO DE SALAS DE AULA

Carlos Eduardo Souza, Bárbara Milagres, Pedro Silva
Departamento de Computação, Universidade Federal de Ouro Preto
Ouro Preto, MG, Brasil
{carlos.romaniello,barbara.leticia}@aluno.ufop.edu.br
silvap@ufop.edu.br

RESUMO

O presente trabalho tem como objetivo avaliar algoritmos que resolvam o Problema de Alocação de Salas (PAS) em cursos universitários. Este problema consiste em alocar turmas de disciplinas com horários predefinidos em salas de aula distribuídas em vários prédios, considerando as determinações da universidade. Dois algoritmos foram avaliados neste trabalho: (i) *Non-Dominated Sorting Genetic Algorithm* (NSGA-II) e (ii) *Multi-Objective Late Acceptance* (MOLA). A exploração do espaço de soluções no algoritmo MOLA é feita por meio de cinco estruturas de vizinhança. Os algoritmos foram avaliados usando dados reais e baseados nos dados do Instituto de Ciências Exatas e Biológicas da Universidade Federal de Ouro Preto. Além disso, os algoritmos foram comparados a uma solução exata construída por meio da estratégia de ϵ-restrito, com uma diferença média de 45,7% utilizando o MOLA e 28,5% utilizando o NSGA-II.

PALAVRAS CHAVE. Problema de alocação de salas. Metaheurísticas. ϵ-restrito.

TÓPICOS: Metaheurísticas, Apoio à Decisão Multicritério

ABSTRACT

The present work aims to evaluate algorithms designed to solve the Classroom Allocation Problem (CAP) in university courses. This problem involves assigning class groups with predefined schedules to classrooms distributed across various buildings, considering the university's regulations. Two algorithms were assessed in this study: (i) the Non-Dominated Sorting Genetic Algorithm (NSGA-II) and (ii) the Multi-Objective Late Acceptance (MOLA) algorithm. The solution space exploration in the MOLA algorithm is performed through five neighborhood structures. The algorithms were evaluated using real data based on information from the Institute of Exact and Biological Sciences of Federal University of Ouro Preto. Additionally, both algorithms were compared against an exact solution built through the ϵ-constraint strategy, with an average difference of 45.7% using MOLA and 28.5% using NSGA-II.

KEYWORDS. Classroom Allocation Problem. Metaheuristics. ϵ-constraint.

PAPER TOPICS: Metaheuristics, Multicriteria Decision Support

DOI: 10.59254/sbpo-2024-193484
https://proceedings.science/p/193484?lang=pt-br

1. Introdução

Semestralmente, inúmeras instituições de ensino superior precisam realizar a alocação de turmas de disciplinas em salas de aula obedecendo a uma série de regras e restrições, como por exemplo, a capacidade de uma sala suportar a demanda de uma turma. Essa tarefa, devido a sua complexidade, é um desafio para as instituições, pois há um elevado número de combinações possíveis para essas alocações, o que inviabiliza uma solução manual. Ademais, normalmente, uma solução manual não consegue contemplar todas as restrições impostas, o que pode gerar insatisfação por parte dos professores e alunos.

O problema de alocação de turmas possui várias etapas, desde a montagem dos horários escolares até a alocação final das salas. Para a montagem de sua grade, cada instituição deve se atentar com conflitos de horários e turmas, o que em si já é um problema complexo. Além disso, também é necessário evitar o conflito de horários entre professores para poder distribuir as aulas que eles ministram da melhor forma possível. Outra etapa para elaboração do problema é decidir se os alunos ou professores deverão mudar de sala durante o dia e essa decisão depende das preferências e características de cada instituição. Sendo assim, com tudo isso já decidido, a distribuição de aulas previamente definidas com horários estabelecidos, atentando-se a diversas particularidades relacionadas ao espaço físico, possibilidade de acesso, infraestrutura e recursos necessários são fatores que caracterizam o Problema de Alocação de Salas (PAS) [Schaerf, 1999].

O PAS é um problema da classe NP-difícil [Even et al., 1975; Carter e Tovey, 1992], por isso ele tem sido normalmente resolvido por meio de técnicas heurísticas, como as metaheurísticas. Sendo assim, vários trabalhos foram apresentados para a solução do PAS por meio dessas técnicas, como o LAHC (*Late Acceptance Hill-Climbing*) [Burke e Bykov, 2017], Busca Tabu [Subramanian et al., 2011] e *Simulated Annealing* [Beyrouthy et al., 2006]. Esses trabalhos utilizam de uma única função objetivo para determinar a qualidade das soluções encontradas por eles e, por isso, esses algoritmos são classificados como mono-objetivo.

Mesmo que o valor da solução fornecida pelos algoritmos citados anteriormente seja usado para avaliar a qualidade das soluções geradas, muitas vezes não reflete todas as necessidades de uma instituição e torna difícil a avaliação individual de cada restrição. Com isso em vista, algoritmos multi-objetivo (otimização de dois ou mais objetivos simultaneamente que normalmente são conflitantes) também foram utilizados na literatura para resolver esse problema, como por exemplo o *Non-Dominated Sorting Genetic Algorithm* (NSGA-II) [Deb et al., 2002], *Strength Pareto Evolutionary Algorithm* (SPEA) [Zitzler e Thiele, 1998] e *Multiobjective Simulated Annealing* (MOSA) [Ulungu et al., 1999]. Como os algoritmos são multi-objetivo, várias soluções são geradas com valores variados para cada objetivo, o que facilita na hora da escolha da solução pela instituição. Nenhuma solução é melhor em todos os objetivos do que outra solução.

O objetivo deste trabalho é empregar dois algoritmos multi-objetivo (NSGA-II e *Multi-Objective Late Acceptance* - MOLA) para o contexto de alocação de salas do Instituto de Ciências Exatas e Biológicas (ICEB) da Universidade Federal de Ouro Preto (UFOP). O NSGA-II é uma adaptação de Deb et al. [2002] e o MOLA baseia-se no trabalho de Vancroonenburg e Wauters [2013] adaptado para utilizar estruturas de vizinhança para exploração do espaço de solução. Além disso, os resultados de NSGA-II e MOLA são comparados com a solução encontrada pela estratégia ϵ-restrito. Utilizou-se estruturas de vizinhanças para realizar operações e mutações durante sua execução.

O restante deste trabalho está dividido da seguinte forma: na Seção 2 são apresentados os trabalhos relacionados; a abordagem proposta e os resultados obtidos são apresentados, respectivamente, nas Seções 3 e 4; por fim, as conclusões e trabalhos futuros estão na Seção 5.

2. Trabalhos relacionados

Ao longo dos anos vários pesquisadores propuseram e estudaram diversos algoritmos e formas de se resolver o problema de alocação de salas, alguns de forma mais genérica como, por

DOI: 10.59254/sbpo-2024-193484
https://proceedings.science/p/193484?lang=pt-br

exemplo, Even et al. [1975] que faz a análise dos algoritmos utilizados na época e outros de forma específica buscando tratar as individualidades de suas realidades como, por exemplo, Al-Yakoob e Sherali [2006] que utilizou políticas e regras da universidade do Kuwait e Kripka e Kripka [2012] que focou sua pesquisa para a realidade da Universidade de Passo Fundo (UPF).

Um dos trabalhos que também focaram em um contexto específico de uma universidade é o trabalho de Nascimento et al. [2005] que utilizou três instâncias específicas: uma fictícia, uma da universidade de Lavras (UFLA) e outra do Instituto de Ciências Exatas e Biológicas (ICEB) da Universidade Federal de Ouro Preto (UFOP). Nesse trabalho ele resolve o PAS de forma mono-objetivo por meio do algoritmo *Simulated Annealing* [Aarts et al., 2005]. Outro trabalho relacionado ao PAS do ICEB-UFOP é o proposto por Souza et al. [2002] que utiliza o *Variable Neighborhood Search* (VNS) [Mladenović e Hansen, 1997]. Neste trabalho, o problema também é resolvido de forma mono-objetivo utilizando uma função objetivo para mensurar a qualidade da solução.

Outra estratégia para resolver o problema do PAS é por meio de programação linear inteira empregada por Chaves e Queiroz [2023]. Os autores conseguiram com poucas horas obter uma solução ótima com 98,8% das disciplinas alocadas para o contexto da Universidade Federal de Catalão (UFCAT).

Por ser um problema NP-difícil, de acordo com Even et al. [1975], conforme esse problema cresce vai ficando cada vez mais difícil de encontrar a melhor solução para o problema. Além disso é possível que vários critérios sejam avaliados pela universidade ou instituição de ensino que enfrenta esse problema. Sendo assim uma abordagem multi-objetivo para o PAS é capaz de gerar várias soluções para o mesmo problema com valores variados para cada um dos objetivos o que fornece mais possibilidades para a instituição de ensino.

3. Abordagem proposta

Para o problema de alocação de salas, no contexto do Instituto de Ciências Exatas e Biológicas da Universidade Federal de Ouro Preto, as aulas são feitas em 36 salas divididas em dois prédios com 16 horários diários divididos em três turnos (manhã, tarde e noite). Uma aula consiste em uma turma (um conjunto de alunos), professor e disciplinas, além de possuir um conjunto de dias e horários pré-definidos. Algumas restrições podem ser aplicadas aos encontros, como: (i) cada sala pode alocar apenas um encontro em um período de tempo e (ii) cada encontro somente pode ser alocado a uma sala em cada período de tempo.

As soluções geradas por meio dos algoritmos propostos neste trabalho tiveram a finalidade de melhorar a atual utilização das salas por uma universidade. Para tal, foram levadas em consideração três características que têm como objetivo criar uma solução mais adequada para alunos e professores, as quais envolvem os seguintes critérios: (i) evitar alocar encontros de pequena demanda em salas com alta capacidade, (ii) tentar alocar o máximo de encontros possíveis e, (iii) evitar alocar encontros em salas que possuem capacidade menor que a demanda.

Dessa forma, pode-se modelar o problema para atender os objetivos especificados de estudo e análise. Para isso, serão considerados três objetivos: Ociosidade (Idl ou *Idleness*): caso mais da metade da sala esteja com vagas não utilizadas essa quantidade é contabilizada; Desalocação (Dea ou *Deallocates*): a quantidade de alunos que estão desalocados; Em pé (Sta ou *Standing*): caso a sala não comporte a quantidade de alunos do encontro, essa diferença é contabilizada.

Para facilitar a modelagem do problema, torna-se necessário representar o caso onde um encontro não é alocado em nenhuma sala. Uma forma de resolver isso é utilizar o conceito de sala virtual que representa uma sala inexistente que "aloca" todos os encontros que não estão alocados em nenhuma sala na solução.

O problema pode ser matematicamente definido por:

Variáveis de decisão:

DOI: 10.59254/sbpo-2024-193484
https://proceedings.science/p/193484?lang=pt-br

X_{esh}: matriz de decisão binária de $|E| \times |S^+| \times |H|$ que indica que o encontro e está alocado na sala s no horário h;

Funções objetivo:

$$Z_1 = min \sum_{i \in E} f_{Dea}(i,j), \forall j \in S^+ \tag{1}$$

$$Z_2 = min \sum_{i \in E} f_{Idl}(i,j), \forall j \in S^+ \tag{2}$$

$$Z_3 = min \sum_{i \in E} f_{Sta}(i,j), \forall j \in S^+ \tag{3}$$

sujeitos a:

$$f_{Dea}(i,j) = \begin{cases} D_i, \text{se } j \in S_v \\ 0, \text{se } j \notin S_v \end{cases} \tag{4}$$

$$f_{Idl}(i,j) = \begin{cases} C_j - D_i & \text{, se } j \in S \wedge D_i < \frac{C_j}{2} \\ 0 & \text{, caso contrário} \end{cases} \tag{5}$$

$$f_{Sta}(i,j) = \begin{cases} D_i - C_j & \text{, se } j \in S \wedge D_i > C_j \\ 0 & \text{, caso contrário} \end{cases} \tag{6}$$

Restrições:

$$\sum_{i \in E} X_{ijk} = 1, \forall j \in S^+, \forall k \in H_i^+ \tag{7}$$

$$\sum_{j \in S} X_{ijk} \leq 1, \forall i \in E, \forall k \in H \tag{8}$$

sendo que E representa conjunto de encontros; H_e é o conjunto dos horários disponíveis; H^+ corresponde ao conjunto dos horários de cada encontro; S é conjunto de salas; S_v representa conjunto da sala virtual; S^+ é $S \cup S_v$; D_i refere-se à demanda do encontro i, $i \in E$; e, por fim, C_j é a capacidade da sala j, $j \in S$.

De forma resumida, o modelo tem como objetivo otimizar as funções presentes nas Equações (1), (2) e (3) que significam, respectivamente, a quantidade de alunos alocados em uma sala virtual S_v, que representa alunos que não foram alocados em nenhuma sala, a quantidade de vagas ociosas em uma sala que possui um encontro alocado e a quantidade de vagas que excederam a capacidade da sala. Para fazer isso, utiliza-se uma matriz de decisão X_{esh} que indicará quando um encontro e $(e \in E)$ for alocado em uma sala s $(s \in S^+)$ contendo o valor 1 ou indicando que um encontro não está alocado em uma sala contendo o valor 0. E por fim, ele deverá respeitar as restrições das Equações (7) e (8) que significam, respectivamente, que todo encontro deverá ser alocado em alguma sala do conjunto S^+ e que cada sala pode ter no máximo um encontro alocado por horário.

3.1. Organização dos dados

Para melhor visualização do problema, os encontros foram separados por dia - de segunda a sábado - sendo que cada dia também contém uma matriz de tamanho *quantidade de horários* $\times$ *quantidade de salas*, que armazena as informações do encontro alocado naquela célula e o status da célula (alocado, não alocado ou reservado).

Os dados utilizados representam a alocação utilizada pela própria universidade, porém para fins de análise, foram geradas mais instâncias fictícias aleatórias que utilizam os dados originais como base.

DOI: 10.59254/sbpo-2024-193484
https://proceedings.science/p/193484?lang=pt-br

3.2. Estrutura de Vizinhança

Para percorrer o espaço de soluções foram implementados cinco movimentos **que são utilizados por todos os algoritmos deste trabalho**:

- **Movimento trocar:** Para o movimento trocar, o algoritmo procura aleatoriamente dois encontros para tentar fazer uma troca. Para ocorrer uma troca, os encontros devem ocorrer no mesmo dia, estar alocado e ter os mesmos horários de aula.

- **Movimento deslocar:** Para o movimento deslocar, o algoritmo procura, aleatoriamente, uma sala para tentar deslocar o encontro escolhido. Para ocorrer um deslocamento, a sala deve estar situada no mesmo dia que o encontro, não possuir nenhum outro encontro alocado nos horários do encontro escolhido e não estar reservada pela instituição para outras finalidades.

- **Movimento substituir:** Para o movimento substituir, o algoritmo procura aleatoriamente um encontro desalocado para tentar substituir um encontro que está alocado. Para ocorrer uma substituição, os encontros devem ocorrer no mesmo dia, um deles estar alocado e o outro não e devem ter os mesmos horários de aula.

- **Movimento alocar:** Para o movimento alocar, é necessário escolher um encontro que ainda não está alocado. Para ocorrer uma alocação o algoritmo procura um encontro que deve estar desalocado, uma sala aleatória que deve estar vazia e o encontro e a sala devem pertencer ao mesmo dia.

- **Movimento desalocar:** Para o movimento desalocar, é necessário encontrar um encontro que já está alocado. Para ocorrer uma desalocação, o algoritmo procura um encontro que deve estar alocado em uma sala aleatória.

3.3. *Non-dominated Sorting Genetic Algorithm* II adaptado (NSGA-II)

O primeiro algoritmo implementado foi o algoritmo multi-objetivo baseado em algoritmos genéticos, o NSGA-II [Deb et al., 2002]. O principal objetivo é encontrar um conjunto de soluções não dominadas, onde nenhuma é melhor do que a outra em todos os objetivos. A concepção do algoritmo original foi seguida, porém algumas adaptações foram feitas para se adequar ao problema abordado em alguns passos:

- **Cruzamento (*Crossover*)**: o problema foi separado de acordo com os dias da semana, com isso a etapa do *crossover* é realizada em cada dia da semana. Cada *crossover* realizado terá duas soluções 'pais' e gerará duas soluções 'filhas'. Para cada encontro do dia é sorteado qual 'filho' ficará com a alocação de qual 'pai' sendo que as alocações se complementam. Caso aconteça de dois encontros herdados de pais diferentes estarem alocados na mesma sala e no mesmo horário, gerando conflito, o encontro do segundo pai fica desalocado.

- **Mutação (*Mutation*)**: nessa etapa existe uma probabilidade da mutação acontecer e a mutação é composta por um conjunto de 5 movimentos escolhidos aleatoriamente que utilizam o conceito das estruturas de vizinhança. Como a mutação é realizada utilizando os movimentos unitários descritos na Seção 3.2 não há como ocorrer inviabilidades nas alocações.

3.4. Multi-Objective *Late Acceptance* (MOLA)

O segundo algoritmo implementado é o *Multi-Objective Late Acceptance* (MOLA) proposto por Vancroonenburg e Wauters [2013] com pequenas adaptações para se adequar ao problema

DOI: 10.59254/sbpo-2024-193484
https://proceedings.science/p/193484?lang=pt-br

abordado. O algoritmo tem a mesma finalidade do NSGA-II, que é encontrar um conjunto de soluções não dominadas.

O algoritmo desenvolvido em [Vancroonenburg e Wauters, 2013] é uma adaptação ao algoritmo criado em [Burke e Bykov, 2012] que permite com que ele seja capaz de encontrar soluções para problemas multi-objetivo. A concepção original consistia em criar uma lista com várias soluções que possuíam seus respectivos valores da função objetivo analisada e a cada iteração era gerada uma solução nova a partir da solução atual e caso essa nova solução fosse melhor que a atual ou a respectiva solução na lista, ela era aceita e os valores da lista e da solução atual eram atualizados. Com isso, no decorrer das iterações as soluções da lista convergem e criam um parâmetro para as próximas soluções a serem analisadas.

Para este trabalho, a lógica se manteve, porém na etapa de análise da nova solução gerada, o valor da função objetivo não é mais relevante para definir se ela será ou não aceita. Ela será comparada com uma outra lista de soluções que armazena todas as soluções não dominadas encontradas até o momento pelo algoritmo. Caso ela não seja dominada por nenhuma delas ela é aceita e caso ela domine alguma solução dessa lista, essa solução dominada é removida.

3.5. Inicialização

Ambos os algoritmos usados neste trabalho (NSGA-II e MOLA) precisam de soluções iniciais e para isso foi desenvolvido um algoritmo gerador guloso de soluções. Para a construção da solução gulosa, primeiro ordena-se de forma decrescente os encontros de acordo com a sua demanda e as salas de acordo com a capacidade de cada sala. Após este processo, para cada encontro aloca-se a melhor opção disponível (aquela com o número de vagas disponível mais próximo da demanda necessária).

Para o caso do NSGA-II, a construção da população também é baseada no mesmo processo, a diferença está no último passo, onde para cada encontro aloca-se ou a melhor ou a primeira opção disponível dada uma probabilidade p. Para este trabalho, adotou-se uma probabilidade de 50%.

3.6. ϵ-restrito

Outra estratégia utilizada para comparação foi a técnica do ϵ-restrito que busca encontrar as soluções da fronteira de Pareto de um determinado problema. A estratégia utilizada consiste em transformar um problema multi-objetivo em um problema mono-objetivo que otimiza apenas um dos objetivos e criando restrições que representam os outros utilizando algoritmos exatos para encontrar a solução ótima. Com isso é possível encontrar a fronteira de Pareto como mencionado em [Haimes, 1971].

Para executar essa estratégia para o problema abordado nesse trabalho, o objetivo de "desalocação" (Dea) se tornou a única função objetivo, enquanto "ociosidade" (Idl) e "em pé" (Sta) foram transformados em restrições adicionais, onde ambos foram fixados como limitantes. Para isto, aplicou-se um método de programação inteira com o *solver* CBC *(Coin-or branch and cut)*. Para o problema em questão foi utilizado o seguinte modelo matemático:

$$\min \sum_{e \in E} \sum_{s \in Sa'} \sum_{h \in H_e} C_{es} * X_{esh} \tag{9}$$

Restrições:

$$\sum_{e \in E} X_{esh} = 1, \forall s \in Sa', \forall h \in H_e \tag{10}$$

$$\sum_{s \in Sa} X_{esh} \leq 1, \forall h \in H, \forall e \in E \tag{11}$$

$$X_{esh} = X_{esh'}, \forall e \in E, \forall s \in Sa', \forall (h, h') \in G_e \tag{12}$$

DOI: 10.59254/sbpo-2024-193484
https://proceedings.science/p/193484?lang=pt-br

em que E representa conjunto de encontros; H é conjunto dos horários disponíveis; H_e corresponde ao conjunto de horários de um encontro específico; Sa é o conjunto de salas disponíveis; Sa' corresponde a Sa mais uma sala virtual para os encontros desalocados; X_{esh} representa a matriz binária tridimensional que assume o valor 0 quando o encontro e, na sala s, no horário h não está alocado e 1 quando está alocado; G_e faz referência ao conjunto dos horários do encontro e agrupados dois a dois (h, h'); e C_{es} é uma matriz bidimensional com o custo de alocação de cada encontro e em cada sala s respeitando a equação:

$$C_{es} = o_{es} + d_e + d_{es} \tag{13}$$

sendo que o_{es} corresponde à capacidade ociosa do encontro e na sala s; enquanto d_e e d_{es} representam a demanda não alocada do encontro e e a demanda alocada do encontro e que ultrapassa da capacidade da sala s, respectivamente.

De forma resumida, a função objetivo da Equação (9) visa minimizar o valor da solução considerando a variável de decisão X_{esh} e a matriz C_{es} que contém os valores calculados pela função na Equação (13). As restrições, representadas pelas funções das Equações (10), (11) e (12), significam, respectivamente, que todo encontro deve ser alocado em uma sala do conjunto Sa', que toda sala deve ter no máximo um encontro alocado por horário e que todos os encontros em todos os seus horários devem ser alocados na mesma sala.

Para implementação da estratégia do ϵ-restrito, primeiro, ambos os objetivos "ociosidade" (Idl) e "em pé" (Sta) foram restringidos ao maior valor encontrado na execução dos algoritmos NSGA-II e MOLA como um *upper bound* (limite superior) e aplicou-se a otimização mono-objetivo com o solver. Após isso, relaxa-se a restrição do objetivo de "ociosidade" em ϵ, isto é, diminui-se o limite superior, e executa-se o solver novamente. Após relaxar-se ao máximo o objetivo de "ociosidade" (diminuir a restrição ao máximo, praticamente a anulando), relaxa-se em ϵ a restrição do objetivo de "em pé" e restringe-se ao máximo o valor do objetivo de "ociosidade" novamente e todo o processo é reexecutado para Idl. Todo esse passo é executado até que não possa-se mais relaxar tanto os objetivos de "ociosidade" quanto de "em pé". Para este trabalho, a relaxação máxima ocorre quando o objetivo fica zero e ϵ foi definido empiricamente como 30 para ambos os objetivos que se tornaram restrições, o que corresponde a um valor entre 1% e 10% dos valores máximos encontrados para cada objetivo para cada uma das instâncias pelos algoritmos MOLA e NSGA-II.

3.7. Métrica de avaliação

A definição de uma métrica de qualidade de uma solução para um problema de otimização multi-objetivo é mais complexo do que para um problema mono-objetivo Zitzler et al. [1999]. Dentre as métricas existentes para isso, existe o hiper volume, o qual é calculado baseado na área do espaço de soluções encontradas e um ponto qualquer dominado por todas as soluções da fronteira de Pareto. Quanto maior o hiper volume, melhor é o conjunto de soluções encontrada pelo algoritmo. Essa métrica será utilizada para avaliar as soluções dos dois algoritmos desenvolvidos neste trabalho e da estratégia de ϵ-restrito.

4. Experimentos e Resultados

Os algoritmos NSGA-II e MOLA foram implementados na linguagem de programação *Python* versão 3.11.5 e a estratégia de ϵ-restrito foi desenvolvida utilizando a biblioteca Python-MIP[1] utilizando seu resolvedor padrão. A instância original foi coletada no período 2020.1 no instituto de uma universidade. Além dela foram utilizadas mais cinco instâncias geradas a partir da principal, conforme descrito na Subseção 4.1.

[1]Mais informações sobre em `https://www.python-mip.com`.

DOI: 10.59254/sbpo-2024-193484
https://proceedings.science/p/193484?lang=pt-br

Para cada um dos parâmetros de entrada, tanto o NSGA-II quanto o MOLA foram executados 5 vezes totalizando 60 execuções (5 por algoritmo para a instância principal e 5 por algoritmo pra cada uma das outras instâncias geradas). Para o algoritmo MOLA a condição de parada foi estabelecida como sendo 900 segundos de execução e para o NSGA-II foram 100 gerações com 100 indivíduos. O padrão de 900 segundos foi escolhido pois com esse tempo o algoritmo MOLA apresentava dificuldades de encontrar novas soluções não dominadas e este se assemelhava ao tempo do NSGA-II ser executado com 100 gerações e 100 indivíduos.

A máquina utilizada para os testes foi um computador com processador Intel i7-9750H 2.6Ghz, 16GB de memória RAM, sistema operacional *Windows*. Dada a característica estocástica do algoritmo, para cada combinação instância-algoritmo foi utilizada uma *seed* diferente de um a cinco para geração de números pseudoaleatórios.

4.1. Descrição das instâncias

Para executar os experimentos deste trabalho foi utilizada uma instância real referente aos dados do Instituto de Ciências Exatas e Biológicas (ICEB) da Universidade Federal de Ouro Preto. Essa instância contêm informações importantes para o problema como: salas disponíveis, horário de funcionamento dos prédios, os encontros a serem alocados, os professores da universidade, restrições e preferências dos encontros.

A instância original possui 888 encontros, 36 salas e 16 horários no total. Já os dados de entrada que foram gerados utilizando a instância original como base possuem 1.000 encontros, 36 salas e 16 horários no total sendo que o valor da demanda dos encontros varia entre 10 e 70. Esse pequeno aumento no número de encontros e uma maior variedade e distribuição dos valores para suas demandas foi suficiente para tornar o problema mais complexo.

Para a geração dos dados foram consideradas as mesmas disciplinas, salas e horários dos dados originais, o que os difere da instância original são a quantidade de encontros e a demanda de cada nova instância. A criação desses novos dados de entrada teve como objetivo aumentar a dificuldade do problema para proporcionar uma análise mais criteriosa do desempenho dos algoritmo, uma vez que com uma maior quantidade de encontros para serem alocados mais combinações de soluções são possíveis. Além disso, a instância original gerava uma situação onde para todas as soluções o valor do objetivo *Em pé* (Idl) era 0 pois a demanda da maioria dos seus encontros era menor que a média de capacidade das salas.

4.2. Resultados obtidos

Após a execução de todos os algoritmos, a Tabela 1 e os gráficos das Figuras 1 e 2, foram gerados para facilitar a visualização dos resultados. Vale ressaltar que a fronteira de Pareto pode ser observada em ambos os gráficos das Figuras 1 e 2 através das soluções da estratégia ϵ-restrito que estão na cor preta. Ademais, os gráficos mostram a execução de todos os algoritmos para duas das instâncias utilizadas, sendo o primeiro para a instância original e o segundo para a primeira instância aleatória. Além disso, foi calculado o hiper volume com o ponto de referência $[2, 2, 2]$ e a quantidade de soluções geradas por cada algoritmo.

Para a instância original, todos os algoritmos conseguiram satisfazer completamente o objetivo "em pé" (Sta), o que acabou gerando um conjunto *flat* de soluções como é possível visualizar na Figura 1. O conjunto de soluções do NSGA-II cobriu uma área mais ampla do espaço de soluções do que o conjunto de soluções do MOLA além de ter gerado mais soluções que se aproximam da fronteira de Pareto.

Na primeira instância, a distribuição das soluções encontradas pelos algoritmos está mais distribuída no espaço de soluções, porém nenhum dos dois algoritmos conseguiu abranger uma área tão extensa quanto a fronteira de Pareto sendo o NSGA-II o que mais se aproximou, como pode ser visualizado na Figura 2.

DOI: 10.59254/sbpo-2024-193484
https://proceedings.science/p/193484?lang=pt-br

Tabela 1: Tabela de métricas para todos os algoritmos para cada instância. HV = Hiper Volume; *Gap* HV diferença em porcentagem entre o hiper volume da solução e da estratégia ϵ-restrito.

Instância	Algoritmo	HV	*Gap* HV (%)	Soluções geradas		
				Menor	Média	Maior
Instância original	MOLA	2,405939	30,6	20	32,0	46
	NSGA-II	2,708870	21,9	49	72,4	86
	ϵ-restrito	3,468343	0,0	6	6	6
Instância 1	MOLA	3,604383	47,8	11	35,0	72
	NSGA-II	5,281034	23,5	89	94,8	98
	ϵ-restrito	6,903395	0,0	187	187	187
Instância 2	MOLA	3,538751	48,1	35	50,4	70
	NSGA-II	4,591808	32,7	98	99,6	100
	ϵ-restrito	6,821391	0,0	181	181	181
Instância 3	MOLA	2,576858	59,4	24	37,8	48
	NSGA-II	3,794773	40,1	94	97,6	100
	ϵ-restrito	6,339259	0,0	307	307	307
Instância 4	MOLA	3,810017	43,2	28	52,0	80
	NSGA-II	4,937945	26,4	95	98,2	100
	ϵ-restrito	6,707994	0,0	210	210	210
Instância 5	MOLA	3,797787	45,1	17	54,0	86
	NSGA-II	5,276902	23,7	88	96,2	100
	ϵ-restrito	6,914846	0,0	238	238	238

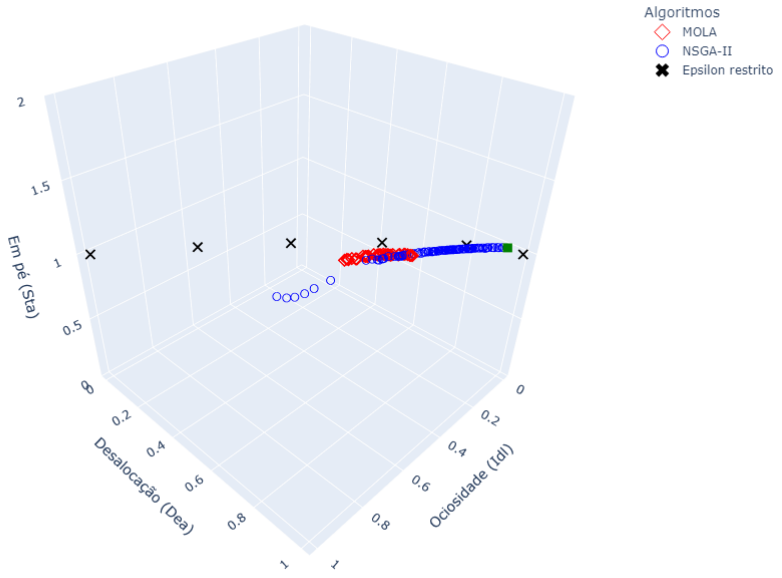

Figura 1: Gráfico contendo os resultados das execuções para a instância original.

Conforme apresentado na Tabela 1 e analisando as instâncias geradas, a área coberta pelo conjunto de soluções dos algoritmos NSGA-II e MOLA é mais restrita/menor do que a explorada pela abordagem ϵ-restrito, mas mesmo assim os algoritmos conseguiram gerar uma boa quantidade de

DOI: 10.59254/sbpo-2024-193484
https://proceedings.science/p/193484?lang=pt-br

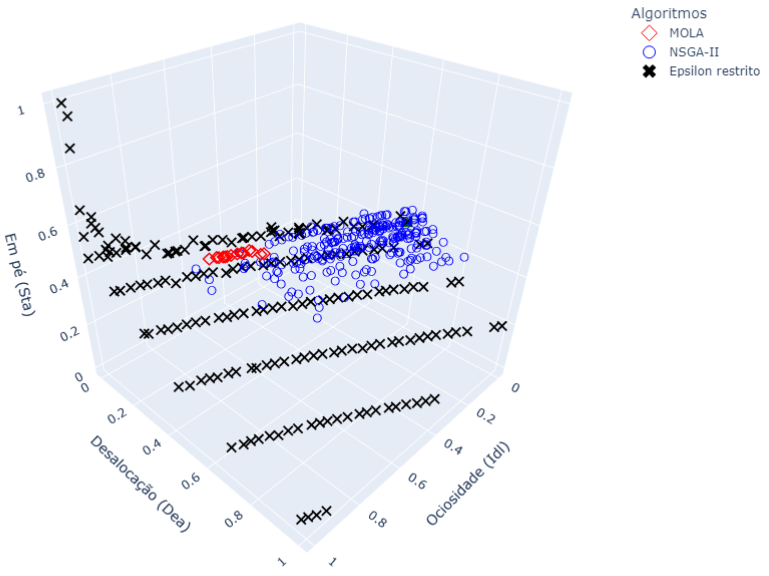

Figura 2: Gráfico contendo os resultados das execuções para a primeira instância.

soluções e exploraram as vizinhanças da melhor forma possível (vide coluna de soluções geradas). A maior discrepância entre as soluções encontradas pela estratégia ϵ-restrito e os outros dois algoritmos foi na terceira instância, onde houve um número significantemente maior de soluções reportadas. Já os algoritmos desenvolvidos não conseguiram explorar tão bem as soluções para esse problema ficando restritos em uma pequena área do espaço de soluções, sendo o MOLA o que teve mais dificuldades de exploração com esses dados de entrada. Para a quarta e quinta instâncias, os algoritmos mantiveram seus comportamentos, gerando uma quantidade boa de soluções e explorando uma parte do espaço de soluções.

Analisando os gráficos das Figuras 1 e 2, e a Tabela 1 é possível perceber que os algoritmos conseguiram gerar uma variedade de soluções para o problema. O NSGA-II conseguiu gerar uma quantidade de soluções maiores e que cobrem um espaço maior do espaço de soluções em relação ao MOLA, como pode ser observado pelos dados na Tabela 1 através dos valores de média, hiper volume e o *Gap* do hiper volume em relação a estratégia do ϵ-restrito. Isso demonstra a versatilidade do NSGA-II que é considerado um dos melhores algoritmos para resolução de problemas multi-objetivos [Zitzler et al., 2000].

Como o MOLA executa um movimento por iteração ele tem mais dificuldades de explorar o espaço de soluções e eventualmente de sair de um ótimo local o que acaba prejudicando sua cobertura do espaço de soluções. Já o NSGA-II, por gerar várias novas soluções por iteração e promover mudanças em algumas delas através da etapa de mutação, acaba por analisar uma quantidade maior de soluções do que o MOLA durante toda sua execução, facilitando assim a exploração do espaço de soluções.

Aliado ao fator sobre a dificuldade do MOLA de explorar uma quantidade maior de soluções, a solução inicial gerada pelo algoritmo guloso tem um grande impacto sobre qual área o MOLA irá percorrer. Outro detalhe é que para a instância original, todo o conjunto de soluções possui o mesmo valor para o objetivo "em pé" (Sta), 0. Para melhor visualização no gráfico, foi decidido que o maior valor encontrado para cada objetivo seria equivalente ao número 1, sendo assim, todas as soluções estão com o valor 1 para esse objetivo

Como esperado, os algoritmos implementados nesse trabalho abrangem uma área menor

DOI: 10.59254/sbpo-2024-193484
https://proceedings.science/p/193484?lang=pt-br

em relação as soluções do método do ϵ-restrito pelo fato de serem algoritmos meta-heurísticos que não necessariamente entregam soluções ótimas, mas sim soluções aproximadas. A desvantagem de utilizar esse método é seu tempo de processamento. Para o descobrimento da fronteira da Pareto foram necessárias em média doze horas de processamento utilizando valores altos para o parâmetro epsilon (ϵ) que chegou a ser cerca de 10% dos valores de um dos objetivos. Já os algoritmos multi-objetivos foram executados por apenas quinze minutos e conseguiram gerar um conjunto de soluções próximas à fronteira na maioria das vezes.

5. Considerações Finais

Os principais objetivos deste trabalho foram alcançados. As necessidades da instituição foram identificadas e levadas em consideração na resolução do problema, o modelo de entrada de dados foi proposto, as estruturas de vizinhança foram elaboradas, os algoritmos foram desenvolvidos e analisados.

Constatou-se que os algoritmos conseguem prover uma variedade de soluções não dominadas que podem ser analisadas para serem escolhidas pela instituição para aplicá-las durante o período de aulas. Porém, o algoritmo NSGA-II se mostrou melhor no quesito de soluções geradas e na cobertura do espaço de soluções geradas como exemplificado nos gráficos das Figuras 1 e 2 e na Tabela 1. Sendo assim o NSGA-II consegue entregar uma variedade maior de soluções para serem analisadas e então selecionadas pela instituição. Além disso essas soluções estão próximas da fronteira de Pareto gerada pelo método ϵ-restrito o que indica a qualidade das mesmas.

Para trabalhos futuros, um caminho a ser seguido é implementar abordagens para sair de ótimos locais com o algoritmo MOLA. Outra vertente que pode ser seguida é analisar como os dados de entrada influenciam na execução dos algoritmos e nas soluções, visto que, dependendo do problema, alguns objetivos podem não ser relevantes e o método de busca local pode ser ineficaz. Por fim, realizar uma análise do impacto da população inicial para os algoritmos.

Agradecimentos

Os autores gostariam de agradecer a Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG), a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) e a Universidade Federal de Ouro Preto (UFOP/PROPPI) por apoiar o desenvolvimento do presente estudo.

Referências

Aarts, E., Korst, J., e Michiels, W. (2005). Simulated annealing. Search methodologies: introductory tutorials in optimization and decision support techniques, p. 187–210.

Al-Yakoob, S. M. e Sherali, H. D. (2006). Mathematical programming models and algorithms for a class–faculty assignment problem. European Journal of Operational Research, 173(2):488–507.

Beyrouthy, C., Burke, E. K., Landa-Silva, J. D., McCollum, B., McMullan, P., e Parkes, A. J. (2006). Understanding the role of ufos within space exploitation. In Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2006), p. 359–362.

Burke, E. K. e Bykov, Y. (2012). The late acceptance hill-climbing heuristic. University of Stirling, Tech. Rep.

Burke, E. K. e Bykov, Y. (2017). The late acceptance hill-climbing heuristic. European Journal of Operational Research, 258(1):70–78.

DOI: 10.59254/sbpo-2024-193484
https://proceedings.science/p/193484?lang=pt-br

Carter, M. W. e Tovey, C. A. (1992). When is the classroom assignment problem hard? Operations Research, 40(1-supplement-1):S28–S39.

Chaves, M. F. d. C. e Queiroz, T. A. d. (2023). Problema de alocação de salas: Estudo de caso na universidade federal de catalão.

Deb, K., Pratap, A., Agarwal, S., e Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE transactions on evolutionary computation, 6(2):182–197.

Even, S., Itai, A., e Shamir, A. (1975). On the complexity of time table and multi-commodity flow problems. In 16th annual symposium on foundations of computer science (sfcs 1975), p. 184–193. IEEE.

Haimes, Y. (1971). On a bicriterion formulation of the problems of integrated system identification and system optimization. IEEE transactions on systems, man, and cybernetics, (3):296–297.

Kripka, R. M. L. e Kripka, M. (2012). Alocação de Salas Objetivando a Minimização de Deslocamentos dos Alunos pelo Campus Central da Universidade de Passo Fundo. In XXIV Congresso Nacional de Matemática Aplicada (CNMAC).

Mladenović, N. e Hansen, P. (1997). Variable neighborhood search. Computers & operations research, 24(11):1097–1100.

Nascimento, A. S., Sampaio, R. M., Alvarenga, G. B., et al. (2005). Uma aplicação de simulated annealing para o problema de alocação de salas. INFOCOMP Journal of Computer Science, 4(3): 59–66.

Schaerf, A. (1999). A survey of automated timetabling. Artificial intelligence review, 13:87–127.

Souza, M. J. F., Martins, A. X., Araújo, C., e Costa, F. W. A. (2002). Métodos de pesquisa em vizinhança variável aplicados ao problema de alocação de salas. XXII Encontro Nacional de Engenharia de Produção ENEGEP, Fortaleza, Brasil.

Subramanian, A., Medeiros, J. M. F., Cabral, L. F., e Souza, M. F. (2011). Aplicação da metaheurística busca tabu ao problema de alocação de aulas a salas em uma instituição universitária. Revista Produção Online, 11(1):54–75.

Ulungu, E. L., Teghem, J., Fortemps, P., e Tuyttens, D. (1999). Mosa method: a tool for solving multiobjective combinatorial optimization problems. Journal of multicriteria decision analysis, 8 (4):221.

Vancroonenburg, W. e Wauters, T. (2013). Extending the late acceptance metaheuristic for multi-objective optimization. In Proceedings of the 6th Multidisciplinary International Scheduling conference: Theory & Applications (MISTA2013).

Zitzler, E., Deb, K., e Thiele, L. (1999). Comparison of multiobjective evolutionary algorithms: Empirical results (revised version). TIK Report, 70.

Zitzler, E., Deb, K., e Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary computation, 8(2):173–195.

Zitzler, E. e Thiele, L. (1998). An evolutionary algorithm for multiobjective optimization: The strength pareto approach. TIK report, 43.

DOI: 10.59254/sbpo-2024-193484
https://proceedings.science/p/193484?lang=pt-br