باسمه تعالی

دانشگاه صنعتی اصفهان

دانشکده برق و کامپیوتر

# حل مسئله جدول زمانی در دانشگاه‌ها با یک روش خوب

درس:

هوش مصنوعی

استاد درس:

دکتر حکیم داودی

نام دانشجو:

حوری دهش (۹۸۲۱۴۱۳)

شهریور ۱۴۰۳

# فهرست

## مقدمه

در دنیای مدرن آموزش عالی، برنامه‌ریزی بهینه زمان‌بندی کلاس‌ها و تخصیص منابع دانشگاهی، به یکی از چالش‌های اساسی تبدیل شده است. زمان‌بندی موثر کلاس‌ها نه تنها بر کیفیت تجربه آموزشی و رضایت دانشجویان و اساتید تاثیر می‌گذارد، بلکه می‌تواند به کاهش مشکلاتی نظیر تداخل‌های زمانی و بار اضافی بر روی کارکنان کمک کند. با توجه به پیچیدگی‌های مرتبط، از جمله هماهنگی میان برنامه‌های مختلف و محدودیت‌های مکانی و زمانی، استفاده از روش‌های پیشرفته و الگوریتم‌های پیچیده برای حل این مسائل ضروری است تا به بهره‌وری و کارایی بالاتری در نهادهای آموزشی دست یابیم.

## توضیح مسئله

مسئله Timetabling کلاس‌ها در دانشگاه‌ها به طور کلی به تعیین بهترین برنامه زمانی برای برگزاری کلاس‌ها و تخصیص منابع آموزشی اشاره دارد. این مسئله به دلیل پیچیدگی‌های مربوط به مدیریت زمان، منابع و نیازهای مختلف آموزشی، شامل چندین چالش و ویژگی کلیدی است که باید به طور موثر مدیریت شوند. این ویژگی و چالش‌ها عبارتند از:

۱. ساختار دوره‌های آموزشی دانشگاه‌ها:

دوره‌های آموزشی به صورت سلسله‌مراتبی مدل‌سازی می‌شوند. این بدین معناست که دوره‌های آموزشی به شکل لایه‌لایه و با روابط والد-فرزند سازماندهی می‌شوند. برای مثال:

- دوره‌های اصلی (Parent Courses): این دوره‌ها به عنوان دوره‌های بزرگ‌تر و بالادست در نظر گرفته می‌شوند و می‌توانند چندین زیرمجموعه یا پیکربندی داشته باشند. برای مثال درس "مبانی برنامه‌نویسی کامپیوتر" یک دوره اصلی است که شامل چندین بخش است.
- پیکربندی‌ها (Sub-courses): این‌ها واحدهای فرعی هستند که تحت دوره‌های اصلی قرار می‌گیرند و بخش‌های مختلف یک دوره را تشکیل می‌دهند. برای مثال درس "مبانی برنامه‌نویسی کامپیوتر" شامل یک بخش تئوری و یک بخش آزمایشگاه است.

کلاس‌ها باید به گونه‌ای زمان‌بندی شوند که در اتاق‌های مشخص و در زمان‌های مناسب برگزار شوند. به عنوان مثال کلاس تئوری "مبانی برنامه‌نویسی" ممکن است در یک روز از هفته و کلاس آزمایشگاه آن در روز دیگری برگزار شود. این کلاس‌ها می‌توانند در روزهای مختلف یا در دوره‌های زمانی خاصی از ترم برنامه‌ریزی شوند. پس ساختار سلسله مراتبی به دانشگاه کمک می‌کند تا برنامه‌ریزی بهتری برای تخصیص منابع (مانند اتاق‌ها و زمان‌بندی کلاس‌ها) انجام دهد و کلاس‌ها را به صورت بهینه سازماندهی کند.

۲. ویژگی‌های زمانی و مکانی کلاس‌ها:

زمان‌بندی کلاس‌ها باید انعطاف‌پذیر باشد تا امکان برگزاری آن‌ها در زمان‌های مختلف روزها و هفته‌ها فراهم شود. همچنین، مدت زمان کلاس‌ها می‌تواند در طول هفته و در ساعات مختلف به صورت دقیق (دقیقه‌ای) تنظیم شود تا نیازهای آموزشی متنوع برآورده شود.

۳. ویژگی‌های اتاق‌ها و محدودیت‌ها:

اتاق‌ها دارای ظرفیت‌های مشخص هستند و ممکن است در بعضی بازه‌های زمانی خاص در دسترس نباشند. همچنین، زمان‌های خاصی برای جابجایی بین اتاق‌ها وجود دارد که بر اساس فاصله‌های فیزیکی بین اتاق‌ها تعیین می‌شود. هر

زمان و اتاق برای کلاس ممکن است جریمه‌هایی داشته باشد که این جریمه‌ها به کیفیت یا بدی انتخاب زمان و مکان برای کلاس‌ها بستگی دارند.

۴. محدودیت‌های تخصیص منابع و جریمه‌ها:

محدودیت‌ها به دو دسته سخت (hard) و نرم (soft) تقسیم می‌شوند:

- محدودیت‌های سخت (Hard Constraints): این محدودیت‌ها باید به طور قطعی رعایت شوند. از جمله این محدودیت‌ها می‌توان به موارد زیر اشاره کرد:
  - تداخل زمانی کلاس‌ها: کلاس‌ها نباید در زمان‌های همپوشان قرار گیرند.
  - نیاز به جابه‌جایی بین کلاس‌ها: دانشجویان نباید مجبور به جابه‌جایی‌های طولانی بین کلاس‌ها شوند.
  - زمان‌های خالی بیش از حد: باید از ایجاد زمان‌های خالی طولانی بین کلاس‌ها جلوگیری شود.
- محدودیت‌های نرم (Soft Constraints): این محدودیت‌ها باید تا حد امکان به طور کامل رعایت شوند با این حال، اگر نتوان به طور کامل به آنها پایبند بود ممکن است با جریمه‌ها یا پیامدهایی همراه باشد. به عبارت دیگر رعایت این محدودیت‌ها مهم است و در صورت عدم رعایت، پیامدهای منفی ممکن است به وجود بیاید. این محدودیت‌ها شامل موارد زیر هستند:
  - حداکثر تعداد استراحت‌ها: تعداد استراحت‌های مجاز بین کلاس‌ها.
  - حداکثر زمان بدون استراحت: مدت زمان طولانی که بدون استراحت طی می‌شود.
  - تعداد روزهای برگزاری کلاس‌ها: تعداد روزهایی که کلاس‌ها باید در آن‌ها برگزار شوند.

جریمه‌ها، به هزینه‌ها یا عواقب منفی ناشی از عدم رعایت محدودیت‌های نرم اشاره دارند. این جریمه‌ها می‌توانند به صورت نقاط منفی، هزینه‌های مالی، یا دیگر معیارهای کیفیت اندازه‌گیری شوند. هدف از کاهش جریمه‌ها، بهبود کیفیت زمان‌بندی و کاهش مشکلاتی است که به دلیل نقض محدودیت‌های نرم پیش می‌آید.

۵. معیارهای بهینه‌سازی:

هدف نهایی بهینه‌سازی این است که جریمه‌ها را برای زمان‌بندی و تخصیص اتاق‌ها کاهش دهیم و همچنین مشکلات ناشی از عدم رعایت محدودیت‌های نرم زمان‌بندی را کاهش دهیم. به عبارت دیگر ما می‌خواهیم مطمئن شویم که کلاس‌ها به شکلی برنامه‌ریزی شوند که به بهترین نحو از اتاق‌ها استفاده شود و زمان‌بندی بهینه باشد و تعداد تداخل‌های دانشجویی کاهش یابد.

محدودیت‌های نرم زمان‌بندی به مواردی اشاره دارند که باید رعایت شوند تا برنامه‌ریزی کلاس‌ها بهتر شود اما اگر نتوان به طور کامل به آن‌ها پایبند بود ممکن است مشکلاتی پیش بیاید.

تداخل‌های دانشجویی به وضعیتی اشاره دارند که دانشجویان نمی‌توانند به تمام کلاس‌های خود که در زمان‌های همپوشان (همزمان) ثبت شده‌اند، شرکت کنند. این مشکلات می‌توانند به دلیل تداخل زمانی بین کلاس‌ها یا فاصله‌های زیاد بین اتاق‌ها ایجاد شوند که باعث می‌شود دانشجویان نتوانند به راحتی بین کلاس‌ها جابجا شوند.

## هدف مسئله

هدف اصلی این مسئله، یافتن و پیشنهاد یک روش بهینه برای حل مشکل زمان‌بندی کلاس‌ها و تخصیص منابع در دانشگاه‌ها است. این مسئله پیچیده شامل نیاز به تخصیص بهینه منابع، مدیریت تداخل‌ها و برآورده کردن نیازهای مختلف ذینفعان است.

مدیریت تداخل‌ها به معنای شناسایی و حل مشکلاتی است که ممکن است زمانی پیش بیاید که دو یا چند کلاس در زمان‌های مشابه یا با منابع محدود (مانند اتاق‌ها یا اساتید مشترک) برنامه‌ریزی شده باشند. هدف این است که مشکلاتی مانند همپوشانی زمانی کلاس‌ها یا ناتوانی دانشجویان در جابجایی بین کلاس‌ها به درستی مدیریت شود.

بنابراین لازم است روشی پیشنهاد شود که بتواند به بهینه‌سازی زمان‌بندی کمک کرده و کیفیت برنامه‌ریزی را بهبود بخشد. روش پیشنهادی باید قادر به حل مسائل مختلف از جمله توزیع بهینه زمان کلاس‌ها، تخصیص مناسب اساتید و مدیریت منابع دانشگاهی باشد.

## ITC 2019

مسابقه ITC 2019 (International Timetabling Competition 2019) به‌طور خاص به مسئله Timetabling کلاس‌ها در دانشگاه‌ها پرداخته و به بررسی روش‌های نوآورانه برای بهینه‌سازی این فرآیند کمک کرده است. این مسابقه، بستری را فراهم کرده که پژوهشگران و متخصصان از روش‌ها و الگوریتم‌های مختلف برای حل مسائل پیچیده زمان‌بندی استفاده کنند و نتایج آنها را مقایسه کنند.

### نقش ITC 2019 در پیشبرد پژوهش‌ها و توسعه الگوریتم‌های Timetabling کلاس‌ها

مسابقه ITC 2019 به رفع مسئله Timetabling کلاس‌ها از طریق فراهم آوردن یک مجموعه استاندارد از چالش‌ها و داده‌های آزمایشی کمک کرده است. این مسابقه از طریق چندین اقدام کلیدی به این هدف دست یافته است:

- فراهم آوردن مجموعه داده استاندارد: به شرکت‌کنندگان نمونه‌های واقعی و شبیه‌سازی شده از مسائل Timetabling ارائه داده تا راه‌حل‌های پیشنهادی را بر اساس این داده‌ها ارزیابی کنند.
- ارزیابی الگوریتم‌ها: با استفاده از معیارهای ارزیابی دقیق، الگوریتم‌های مختلف Timetabling را از نظر عملکرد و کارایی مورد بررسی قرار داد.
- تشویق به نوآوری: ایجاد فضایی برای رقابت و تبادل نظر میان پژوهشگران، منجر به توسعه روش‌های جدید و بهبود الگوریتم‌های موجود شد.

### توصیف نمونه‌های استفاده شده در مسابقه

در مسابقه ITC 2019، ۳۰ نمونه آزمون (benchmark instances) منتشر شد که شامل نمونه‌های early، middle و late بودند. این نمونه‌ها به بررسی و ارزیابی الگوریتم‌های زمان‌بندی کمک کردند. هر نمونه آزمون ویژگی‌های خاصی دارد که به بررسی ابعاد مختلف مسئله زمان‌بندی کمک می‌کند. این ویژگی‌ها شامل اندازه مسئله (تعداد کلاس‌ها، دانشجویان و اتاق‌ها)، تقاضاهای دانشجویان، الگوهای تاریخ و زمان و.. می‌شود.

دلیل اصلی در تقسیم‌بندی ۳۰ نمونه به دسته‌های early، middle و late بخاطر ترتیب زمانی انتشار این نمونه‌ها در مسابقه و ویژگی‌های خاص هر دسته است. هر دسته از نمونه ها با توجه به پیچیدگی و داده‌های متفاوت از جمله تعداد دانشجویان، کلاس‌ها، محدودیت‌ها و جزئیات دیگر تعریف می‌شود.

- دلیل تقسیم‌بندی:
  - Early Instances: این دسته شامل نمونه‌هایی است که در اوایل مسابقه منتشر شده‌اند. این نمونه‌ها معمولا شامل مشکلات کوچک‌تر با تعداد کلاس‌ها، دانشجوها و اتاق‌های کمتر هستند. این نمونه‌ها بیشتر برای شروع و آشنایی با مسئله استفاده می‌شوند. به عنوان مثال نمونه agh-fis-spr17 که در این دسته قرار دارد دارای ۷ روز، ۲۸۸ زمان‌بندی در روز (یعنی در هر روز ۲۸۸ بازه زمانی مختلف برای برگزاری کلاس‌ها و دیگر فعالیت‌ها وجود دارد) و ۱۶ هفته است و دارای تعداد نسبتا کمی از کلاس‌ها و اتاق‌هاست.
    برای درک بهتر ۲۸۸ زمان‌بندی در روز می‌توان این مثال رو زد:
    تصور کنید که شما در یک دانشگاه باید کلاس‌ها را برای یک روز برنامه‌ریزی کنید. روز تحصیلی شما به ۲۸۸ بخش کوچک تقسیم شده است. هر بخش یک بازه زمانی مشخص را پوشش می‌دهد.
    فرض کنید که روز تحصیلی از ساعت ۸ صبح تا ۸ شب است. این زمان معادل ۱۲ ساعت است. اگر روز به ۲۸۸ بازه زمانی تقسیم شده باشد، باید طول هر بازه زمانی را محاسبه کنیم:
    ۱۲ ساعت = ۷۲۰ دقیقه
    ۷۲۰ دقیقه / ۲۸۸ بازه = ۲.۵ دقیقه برای هر بازه زمانی
    یعنی هر بازه زمانی ۲.۵ دقیقه طول دارد.
    مثال واقعی‌تر:
    تصور کنید که شما یک کلاس ۱ ساعته دارید. در این تقسیم‌بندی، این کلاس باید در ۲۴ بازه زمانی متوالی قرار گیرد (زیرا ۱ ساعت = ۶۰ دقیقه و ۶۰ دقیقه / ۲.۵ دقیقه = ۲۴ بازه)
    اگر یک کلاس دیگر ۴۵ دقیقه طول می‌کشد، این کلاس در ۱۸ بازه زمانی متوالی قرار می‌گیرد (زیرا ۴۵ دقیقه / ۲.۵ دقیقه = ۱۸ بازه)
    پس تقسیم روز به ۲۸۸ بازه زمانی به این معنی است که هر روز به بخش‌های بسیار کوچکی تقسیم می‌شود تا زمان‌بندی دقیق‌تری برای کلاس‌ها و استفاده بهینه از اتاق‌ها فراهم شود و همینطور مطمئن شد که هیچ دو کلاسی همزمان در یک اتاق برگزار نمی‌شود. این کار به ما کمک می‌کند که برنامه‌ریزی دقیق‌تری داشته باشیم و از تداخل‌های زمانی جلوگیری کنیم.
  - Middle Instances: در این دسته مسئله پیچیده‌تر شده و اندازه نمونه‌ها و محدودیت‌ها افزایش یافته است. در این نمونه‌ها معمولا تعداد بیشتری از کلاس‌ها و دانشجویان وجود دارد و همچنین محدودیت‌های بیشتری اعمال می‌شود. به عنوان مثال نمونه yach-fal17 که در این دسته قرار دارد، دارای ۷ روز و ۲۸۸ زمان‌بندی در روز و تعداد زیادی محدودیت است.
  - Late Instances: این دسته شامل نمونه‌های نهایی و پیچیده‌ترین نمونه‌ها است که در انتهای مسابقه منتشر شده‌اند. این نمونه‌ها معمولا بسیار بزرگ هستند و شامل تعداد زیادی دانشجو، کلاس و اتاق می‌باشند و محدودیت‌های زیادی دارند. به عنوان مثال نمونه agh-fal17 دارای ۷ روز، ۲۸۸ زمان‌بندی در روز و ۱۸ هفته است و شامل محدودیت‌های پیچیده‌تر و تعداد بیشتری از اتاق‌هاست.
- تفاوت‌های کلی بین نمونه‌ها:
  - اندازه و پیچیدگی: در هر نمونه، اندازه نمونه، تعداد کلاس‌ها، دانشجویان و اتاق‌ها افزایش یافته است. همچنین محدودیت‌های بیشتری در نمونه‌های Middle و Late اعمال شده است.

- نوع محدودیت‌ها: در نمونه‌های Early، محدودیت‌ها ساده‌تر هستند ولی در نمونه‌های Middle و Late، محدودیت‌های پیچیده‌تر و متنوع‌تری مانند پراکندگی حضور دانشجویان در کلاس‌ها و محدودیت‌های زمانی اضافه شده است.

**ویژگی‌های نمونه‌های استفاده شده در مسابقه**

- اندازه مسئله: نمونه‌ها از اندازه‌های مختلفی برخوردار هستند، از مشکلات کوچک با تعداد کلاس‌ها و اتاق‌های کم تا مشکلات بزرگ با هزاران کلاس و دانشجو.
- تقاضاهای دانشجویان: داده‌های مربوط به تقاضاهای دانشجویان ممکن است از منابع مختلف جمع‌آوری شده باشد، شامل ثبت‌نام‌های قبلی یا تقاضاهای بر اساس برنامه درسی.
- محدودیت‌های توزیع: برخی از نمونه‌ها محدودیت‌های توزیع متعددی دارند که می‌تواند تاثیر زیادی بر پیچیدگی حل مسئله داشته باشد.
- دامنه متغیرها: شامل زمان‌های در دسترس برای کلاس‌ها و تعداد اتاق‌های قابل استفاده است که می‌تواند بر سختی حل مسئله تاثیر بگذارد.
- الگوهای زمانی (زمان‌ها): شامل تعداد هفته‌ها، مدت زمان هر جلسه، تعداد روزهای برگزاری کلاس‌ها و چگونگی توزیع کلاس‌ها در طول ترم.
- بهره‌برداری از اتاق‌ها: به میزان استفاده از اتاق‌ها و چالش‌های ناشی از استفاده زیاد یا کم از اتاق‌ها اشاره دارد.
- وزن‌های بهینه‌سازی: اهمیت معیارهای بهینه‌سازی مختلف مانند ترجیحات زمانی، ترجیحات اتاق‌ها و تداخل‌های دانشجویی را نشان می‌دهد که بر حل مسئله تاثیر می‌گذارند.

<u>ویژگی‌های خاص هر دانشگاه:</u>

- دانشگاه Masaryk: نمونه‌های مختلف از دانشگاه Masaryk شامل مسائل مربوط به کلاس‌های منظم، آموزش از راه دور و ترکیب آن‌ها با یکدیگر هستند.
- دانشگاه Purdue: نمونه‌ها شامل مسائل مربوط به کلاس‌های بزرگ و مشکلات چندین دپارتمان است.
- دانشگاه AGH: نمونه‌ها از مشکلات بزرگ با تعداد زیاد کلاس‌ها و محدودیت‌های خاص است.
- دانشگاه‌های دیگر: نمونه‌های دانشگاه‌های مختلف دیگر نیز شامل ویژگی‌های خاصی هستند: مانند مشکل در زمان‌بندی کلاس‌ها یا محدودیت‌های ویژه.

فرمت نمونه‌های ورودی به صورت XML است و این نمونه‌ها شامل تنظیمات برای یک سیستم زمان‌بندی و تخصیص منابع می‌باشند. این تنظیمات شامل جزئیات دروس، اتاق‌ها، کلاس‌ها، زمان‌بندی‌ها و نیازهای دانشجویان است. هدف از استفاده از این نمونه‌ها، تنظیم پارامترهای مختلف برای ایجاد یک برنامه زمانی بهینه و منطبق با محدودیت‌ها و الزامات متنوع است.

تصویرهای زیر بخشی از نمونه agh-ggis-spr17 از دسته Early را نشان می‌دهند:

```
- <problem name="agh-ggis-spr17" nrDays="7" slotsPerDay="288" nrWeeks="16">
    <optimization time="4" room="1" distribution="15" student="5"/>
  - <rooms>
    - <room id="1" capacity="180">
        <travel room="3" value="1"/>
        <travel room="4" value="1"/>
        <travel room="5" value="1"/>
        <travel room="7" value="1"/>
        <travel room="8" value="1"/>
        <travel room="10" value="1"/>
        <travel room="11" value="1"/>
        <travel room="12" value="1"/>
        <travel room="13" value="1"/>
        <travel room="14" value="1"/>
        <travel room="15" value="1"/>
        <travel room="16" value="1"/>
        <travel room="28" value="1"/>
        <travel room="32" value="1"/>
        <travel room="35" value="1"/>
        <travel room="44" value="1"/>
        <unavailable days="1000000" start="0" length="288" weeks="1111111111111111"/>
        <unavailable days="0100000" start="0" length="204" weeks="1111111111111111"/>
        <unavailable days="0100000" start="246" length="42" weeks="1111111111111111"/>
        <unavailable days="0010000" start="0" length="288" weeks="1111111111111111"/>
        <unavailable days="0001000" start="0" length="204" weeks="1111111111111111"/>
        <unavailable days="0001000" start="246" length="42" weeks="1111111111111111"/>
        <unavailable days="0000100" start="0" length="288" weeks="1111111111111111"/>
        <unavailable days="0000010" start="246" length="42" weeks="1111111111111111"/>
        <unavailable days="0000001" start="246" length="42" weeks="1111111111111111"/>
      </room>
```

تحلیل این نمونه:

مشخصات کلی مسئله:

- name: agh-ggis-spr17، که به نظر می‌رسد مربوط به یک ترم بهار در سال ۲۰۱۷ است.
- nrDays (تعداد روزها): ۷ روز
- slotsPerDay (تعداد بازه‌ها در روز): ۲۸۸ بازه زمانی در هر روز.
- nrWeeks (تعداد هفته‌ها): ۱۶ هفته یعنی برنامه‌ریزی برای کل ترم که ۱۶ هفته به طول می‌انجامد.

بخش بهینه‌سازی یا optimization:

این بخش به تعیین اولویت‌ها و اهداف مختلفی که باید در فرآیند زمان‌بندی و تخصیص منابع بهینه شوند، می‌پردازد. این بخش مشخص می‌کند که به چه میزان و در چه جنبه‌هایی باید بهینه‌سازی انجام شود. در اینجا، عددها نمایانگر میزان اهمیت یا وزن هر هدف بهینه‌سازی هستند:

- time=4: این مقدار نشان‌دهنده اهمیت یا وزن بهینه‌سازی زمان‌بندی است. مقدار ۴ به این معنی است که زمان‌بندی و تخصیص زمان‌ها برای کلاس‌ها و جلسات نسبتا اهمیت زیادی دارد و باید به خوبی بهینه‌سازی شود.
- room=1: این مقدار نشان‌دهنده اهمیت بهینه‌سازی تخصیص اتاق‌ها است. مقدار ۱ به این معنی است که تخصیص اتاق‌ها کمترین اهمیت را نسبت به دیگر اهداف بهینه‌سازی دارد یعنی در مقایسه با زمان‌بندی و توزیع کلاس‌ها، توجه کمتری به بهینه‌سازی اتاق‌ها داده شده است.
- distribution=15: این مقدار بهینه‌سازی تخصیص مناسب کلاس‌ها به زمان‌های مختلف و اتاق‌های موجود را نشان می‌دهد. مقدار ۱۵ به این معنی است که باید توجه زیادی به نحوه تقسیم و برنامه‌ریزی کلاس‌ها بین زمان‌های مختلف و اتاق‌ها داده شود تا بهترین نتیجه حاصل شود.
- student=5: این مقدار نشان‌دهنده اهمیت بهینه‌سازی تخصیص دانشجویان به کلاس‌ها است. مقدار ۵ به این معنی است که توجه به نیازهای دانشجویان و نحوه تخصیص آن‌ها به کلاس‌ها نسبتا مهم است ولی نسبت به زمان‌بندی و توزیع کلاس‌ها اهمیت کمتری دارد.

بخش اتاق‌ها:

- یک اتاق با id=1 و ظرفیت ۱۸۰ نفر وجود دارد.
- در بخش travel، نشان داده شده که اتاق ۱ با ۱۷ اتاق دیگر (با شناسه‌های ۳، ۴، ۵، ...، ۴۴) فاصله‌ای برابر با ۱ واحد دارد. این ممکن است به معنای فاصله فیزیکی یا زمان سفر بین اتاق‌ها باشد.

بخش Unavailable:

در این بخش، زمان‌هایی که اتاق در دسترس نیست مشخص شده است:

- days رشته‌ای ۷ رقمی که هر رقم نشان‌دهنده یک روز از هفته است (از شنبه تا جمعه). "۱" یعنی آن روز در دسترس نیست و "۰" یعنی در دسترس است.
- start: نقطه شروع بازه زمانی که اتاق در دسترس نیست.
- length: طول بازه زمانی عدم دسترسی به اتاق.
- weeks: رشته‌ای ۱۶ یا ۱۸ رقمی که نشان می‌دهد در کدام هفته‌ها این عدم دسترسی وجود دارد (هر رقم نماینده یک هفته است). "۱" یعنی آن هفته در دسترس نیست و "۰" یعنی در دسترس است.

برای مثال:

unavailable days=1000000 یعنی اتاق در روز اول در دسترس نیست.

start=0 length=288 یعنی اتاق در کل روز اول (۲۸۸ بازه زمانی) در دسترس نیست.

weeks=1111111111111111 یعنی این عدم دسترسی در تمام ۱۶ هفته‌ی ترم ادامه دارد.

```
- <course id="10">
  - <config id="10">
    - <subpart id="23">
      - <class id="77" limit="120">
          <room id="10" penalty="0"/>
          <room id="23" penalty="0"/>
          <room id="24" penalty="0"/>
          <room id="29" penalty="4"/>
          <room id="42" penalty="0"/>
          <time days="0010000" start="96" length="18" weeks="0101010010010101" penalty="0"/>
          <time days="0010000" start="96" length="18" weeks="1010101000101010" penalty="0"/>
          <time days="0001000" start="96" length="18" weeks="0101010010101010" penalty="0"/>
          <time days="0001000" start="96" length="18" weeks="1010100101010100" penalty="0"/>
          <time days="0010000" start="106" length="18" weeks="0101010010010101" penalty="0"/>
          <time days="0010000" start="106" length="18" weeks="1010101000101010" penalty="0"/>
          <time days="0001000" start="106" length="18" weeks="0101010010101010" penalty="0"/>
          <time days="0001000" start="106" length="18" weeks="1010100101010100" penalty="0"/>
          <time days="0010000" start="116" length="18" weeks="0101010010010101" penalty="0"/>
          <time days="0010000" start="116" length="18" weeks="1010101000101010" penalty="0"/>
          <time days="0001000" start="116" length="18" weeks="0101010010101010" penalty="0"/>
          <time days="0001000" start="116" length="18" weeks="1010100101010100" penalty="0"/>
          <time days="0010000" start="126" length="18" weeks="0101010010010101" penalty="0"/>
          <time days="0010000" start="126" length="18" weeks="1010101000101010" penalty="0"/>
          <time days="0001000" start="126" length="18" weeks="0101010010101010" penalty="0"/>
          <time days="0001000" start="126" length="18" weeks="1010100101010100" penalty="0"/>
          <time days="0010000" start="136" length="18" weeks="0101010010010101" penalty="0"/>
          <time days="0010000" start="136" length="18" weeks="1010101000101010" penalty="0"/>
          <time days="0001000" start="136" length="18" weeks="0101010010101010" penalty="0"/>
          <time days="0001000" start="136" length="18" weeks="1010100101010100" penalty="0"/>
          <time days="0010000" start="146" length="18" weeks="0101010010010101" penalty="0"/>
          <time days="0010000" start="146" length="18" weeks="1010101000101010" penalty="0"/>
          <time days="0001000" start="146" length="18" weeks="0101010010101010" penalty="0"/>
          <time days="0001000" start="146" length="18" weeks="1010100101010100" penalty="0"/>
```

course id=10: این بخش نشان‌دهنده یک درس است که شناسه آن ۱۰ است.

config id=10: این قسمت پیکربندی مربوط به این درس را مشخص می‌کند. پیکربندی ممکن است شامل زیرمجموعه‌هایی از کلاس‌ها باشد.

پیکربندی به معنای تنظیمات مربوط به یک درس خاص است. برای هر درس، ممکن است چندین حالت یا تنظیمات مختلف وجود داشته باشد. هر پیکربندی شامل اطلاعاتی مثل کلاس‌ها، اتاق‌هایی که درس در آن‌ها برگزار می‌شود، زمان‌های برگزاری کلاس‌ها و سایر جزئیات است. هر درس ممکن است به دلایل مختلف (مثل تقسیم دانشجویان به گروه‌های مختلف یا برگزاری کلاس در زمان‌های مختلف) چندین پیکربندی داشته باشد. برای مثال یک درس ممکن است همزمان در چندین کلاس مختلف با زیرگروه‌های دانشجویان مختلف تدریس شود و هر یک از این کلاس‌ها در زمان و اتاق متفاوت برگزار شود. این تنوع اطلاعات و تنظیمات می‌تواند در قالب پیکربندی‌های مختلف برای یک درس خاص نمایش داده شود.

مثال: فرض کنید شما یک درس به نام ریاضیات دارید. برای این درس، ممکن است چندین پیکربندی وجود داشته باشد. مثلا:

- یک پیکربندی برای کلاس‌های نظری
- یک پیکربندی دیگر برای کلاس‌های عملی

در اینجا config id=10 به یکی از این پیکربندی‌ها اشاره دارد مثلا پیکربندی مربوط به کلاس‌های نظری ریاضیات.

subpart id=23: این زیرمجموعه‌ای از کلاس‌های مربوط به این درس است. هر درس ممکن است چندین کلاس یا بخش داشته باشد.

برای مثال: برای درس ریاضیات می‌توان زیرمجموعه‌های مختلفی از کلاس‌ها وجود داشته باشد:

- زیرمجموعه ۱: کلاس‌های نظری (به عنوان مثال کلاس‌های صبح روزهای شنبه و دوشنبه)
- زیرمجموعه ۲: کلاس‌های عملی (مثلا کلاس‌های بعدازظهر روزهای سه‌شنبه و چهارشنبه)

در این مثال subpart id=23 به یکی از این زیرمجموعه‌ها اشاره دارد مثلا کلاس‌های نظری که در چندین زمان و اتاق مختلف برگزار می‌شوند.

class id=77  limit=120: این بخش مربوط به یک کلاس خاص با شناسه ۷۷ است که حداکثر ظرفیت ۱۲۰ نفر را دارد.

room id="..." penalty="...": این‌ها اتاق‌هایی هستند که این کلاس می‌تواند در آنها برگزار شود. هر اتاق یک شناسه و یک مقدار جریمه (penalty) دارد. جریمه نشان می‌دهد که استفاده از یک اتاق خاص ممکن است چقدر غیرمطلوب باشد. اگر جریمه صفر باشد یعنی اتاق برای کلاس مناسب است اما اگر مقدار آن بیشتر از صفر باشد (مثلا اتاق ۲۹ با جریمه ۴)، نشان‌دهنده محدودیت‌هایی در استفاده از آن اتاق است.

جریمه ۴ برای اتاق ۲۹ نشان می‌دهد که استفاده از این اتاق کم‌تر مطلوب است (نسبت به اتاق‌هایی که جریمه آن‌ها صفر است). به عبارت دیگر، اگر سیستم اتاق دیگری بدون جریمه یا با جریمه کمتر پیدا کند، ترجیحا آن را انتخاب می‌کند اما اگر مجبور باشد از اتاق ۲۹ استفاده کند، هزینه‌ای به برنامه افزوده می‌شود.

این جریمه‌ها می‌توانند به دلایلی مثل اندازه کوچک‌تر اتاق، دوری از مکان اصلی یا در دسترس نبودن اتاق در ساعات خاص تعیین شوند.

time: این بخش مربوط به زمان‌هایی است که این کلاس می‌تواند برگزار شود. برای هر کلاس، چندین زمان ممکن تعریف شده است:

days=0010000: این کد روز هفته‌ای را که کلاس برگزار می‌شود نشان می‌دهد. برای مثال، ۰۰۱۰۰۰۰ نشان می‌دهد که کلاس در روز سوم برگزار می‌شود.

start=96: این مقدار نشان‌دهنده زمان شروع کلاس است به عبارت دیگر کلاس از چه زمانی آغاز می‌شود. معمولا این مقدار به دقیقه از شروع روز اشاره دارد. به این معنی که start=96 به این معناست که کلاس ۹۶ دقیقه بعد از شروع روز آغاز می‌شود.

برای مثال، اگر روز آموزشی از ساعت ۸ صبح شروع شود و ۹۶ دقیقه به آن اضافه کنیم کلاس در ساعت ۹:۳۶ صبح شروع خواهد شد.

length=18: این نشان می‌دهد که کلاس چند واحد زمانی طول می‌کشد به عبارت دیگر به طول مدت کلاس اشاره دارد. این واحدهای زمانی می‌توانند دقیقه، ساعت یا دیگر واحدهای زمانی باشند در بسیاری از سیستم‌ها، واحد زمانی معمولا دقیقه است. بنابراین، اگر length=18 به معنی ۱۸ دقیقه باشد، این بدین معناست که کلاس به مدت ۱۸ دقیقه طول می‌کشد و اگر واحد زمانی ساعت باشد، طول کلاس ۱۸ ساعت خواهد بود.

weeks: این کد نشان می‌دهد که کلاس در چه هفته‌هایی از ترم برگزار می‌شود. برای مثال "۰۱۰۱۰۱۰۰۱۰۰۱۰۱۰۱" نشان می‌دهد که کلاس در هفته‌های خاصی از ترم برگزار خواهد شد و عدد ۱ به معنای برگزاری کلاس در آن هفته و ۰ به معنای برگزار نشدن است.

penalty=0: این جریمه‌ای است که برای برگزاری کلاس در این زمان خاص اعمال می‌شود. اگر جریمه صفر باشد زمان مناسب است.

```
−<class id="86" limit="130" room="false">
    <time days="0100000" start="204" length="36" weeks="1111111010111111" penalty="0"/>
  </class>
```

class id=86: این شناسه مربوط به کلاس شماره ۸۶ است. هر کلاس یک شناسه منحصربه‌فرد دارد که در سیستم برای شناسایی آن استفاده می‌شود.

limit=130: این نشان‌دهنده حداکثر تعداد دانشجویانی است که می‌توانند در این کلاس شرکت کنند. در اینجا، حداکثر تعداد ۱۳۰ نفر است.

room=false: این نشان می‌دهد که برای این کلاس هنوز هیچ اتاقی اختصاص داده نشده است یا ممکن است این کلاس نیازی به اتاق نداشته باشد (مثلا یک کلاس آنلاین).

time: اطلاعات مربوط به زمان و روزهایی که این کلاس برگزار می‌شود (مابقی اطلاعاتش مثل قبلی می‌باشد.)

```
−<distribution type="SameAttendees" required="true">
    <class id="1292"/>
    <class id="1291"/>
    <class id="1290"/>
    <class id="1289"/>
  </distribution>
```

این بخش از نمونه مربوط به "قید کلاس‌ها" است و نوع خاصی از محدودیت‌ها یا تنظیمات مربوط به کلاس‌ها را نشان می‌دهد.

distribution type=SameAttendees: این قسمت محدودیت کلاس‌ها را تعیین می‌کند. در اینجا، نوع SameAttendees انتخاب شده است که به معنای این است که همه دانشجویان یکسان (حاضرین مشترک) باید در این کلاس‌ها شرکت کنند یعنی دانشجویانی که در یک کلاس مشخص شرکت می‌کنند باید در سایر کلاس‌های ذکرشده نیز شرکت کنند.

required=true: این نشان می‌دهد که رعایت این محدودیت اجباری است. اگر مقدار true باشد یعنی این قید باید حتما رعایت شود و همه کلاس‌های مشخص‌شده باید دانشجویان مشترک داشته باشند.

class id=1292 و class id=1291 و class id=1290 و class id=1289: این‌ها کلاس‌های مشخص‌شده در این قید هستند. هر کلاس با شناسه منحصربه‌فردی (مانند ۱۲۹۲، ۱۲۹۱ و ...) مشخص شده است. این بخش به سیستم می‌گوید که کلاس‌های با شناسه ۱۲۹۲، ۱۲۹۱، ۱۲۹۰، و ۱۲۸۹ باید دانشجویان یکسانی داشته باشند و همه آن‌ها باید این قید را رعایت کنند.

پس این بخش از نمونه نشان می‌دهد که برای کلاس‌های با شناسه‌های ۱۲۹۲، ۱۲۹۱، ۱۲۹۰ و ۱۲۸۹، یک محدودیت وجود دارد که همه دانشجویان این کلاس‌ها باید مشترک باشند. همچنین رعایت این محدودیت اجباری است و نمی‌توان آن را نادیده گرفت.

```
-<student id="2116">
    <course id="128"/>
    <course id="129"/>
    <course id="131"/>
    <course id="133"/>
    <course id="135"/>
    <course id="137"/>
    <course id="138"/>
    <course id="124"/>
    <course id="126"/>
    <course id="142"/>
    <course id="271"/>
  </student>
```

این بخش از نمونه به توصیف یک دانشجو خاص و دوره‌های آموزشی که او در آن‌ها ثبت‌نام کرده است، می‌پردازد.

student id=2116: این خط نشان‌دهنده دانشجو با شناسه ۲۱۱۶ است. در این بخش تمام اطلاعات مربوط به این دانشجو قرار دارد.

course id=128 تا course id=271: این خطوط لیستی از دوره‌های آموزشی هستند که دانشجو با شناسه ۲۱۱۶ در آن‌ها ثبت‌نام کرده است. هر کد id به یک دوره آموزشی خاص اشاره دارد.

## نحوه ارزیابی راه‌حل تیم‌ها

راه‌حل‌های ارسال شده با استفاده از یک سیستم خودکار ارزیابی بررسی می‌شوند. این سیستم بر اساس مجموعه‌ای از محدودیت‌های سخت و نرم که در مسئله تعریف شده‌اند، راه‌حل‌ها را ارزیابی می‌کند. به طور کلی، مراحل چک کردن و امتیازدهی به این صورت است:

۱. تایید رعایت محدودیت‌های سخت:

ابتدا راه‌حل باید محدودیت‌های سخت را رعایت کند، که شامل مواردی مثل عدم تداخل زمانی کلاس‌ها، استفاده مناسب از فضاهای موجود (مانند اتاق‌ها) و جلوگیری از تداخل اساتید یا دانشجویان است. اگر راه‌حلی این محدودیت‌ها را رعایت نکند، جریمه سنگینی دریافت می‌کند.

۲. بررسی محدودیت‌های نرم:

سپس، محدودیت‌های نرم چک می‌شوند. این محدودیت‌ها مواردی هستند که بهتر است رعایت شوند، اما نقض آن‌ها تنها جریمه‌های کمتری دارد.

۳. محاسبه جریمه‌ها:

بر اساس تعداد و نوع محدودیت‌های نقض‌شده، جریمه‌هایی برای هر راه‌حل محاسبه می‌شود. هر چه جریمه‌ها کمتر باشند، راه‌حل بهینه‌تر است و امتیاز بهتری کسب می‌کند. سیستم ارزیابی خودکار بر اساس این جریمه‌ها، امتیازی را به هر راه‌حل اختصاص می‌دهد که بیانگر کیفیت آن است.

۴. مقایسه با سایر تیم‌ها:

راه‌حل‌های تیم‌ها بر اساس امتیازی که از سیستم ارزیابی دریافت می‌کنند، با یکدیگر مقایسه می‌شوند. به عنوان مثال تیمی که بهترین راه‌حل را ارائه دهد، رتبه اول را می‌گیرد. تیم‌هایی که راه‌حل‌های خوبی ارائه دهند ولی به پای تیم اول نرسند، در رتبه‌های بعدی قرار می‌گیرند.

۵. اختصاص امتیازات:

بر اساس جدول زیر، امتیازها بر اساس رتبه‌بندی راه‌حل‌ها در هر نمونه تعلق می‌گیرد.

| | Instance | | |
|---|---|---|---|
| Position | Early | Middle | Late |
| 1st | 10 | 15 | 25 |
| 2nd | 7 | 11 | 18 |
| 3rd | 5 | 8 | 15 |
| 4th | 3 | 6 | 12 |
| 5th | 2 | 4 | 10 |
| 6th | 1 | 3 | 8 |
| 7th | | 2 | 6 |
| 8th | | 1 | 4 |
| 9th | | | 2 |
| 10th | | | 1 |

**Table 1** Points awarded for an instance

مراحل ارزیابی:

- تقسیم امتیازات بر اساس نمونه‌های early، middle، late:

هر تیم بر اساس جایگاهی که در هر نمونه (early، middle، late) کسب می‌کند، امتیاز دریافت می‌کند. جدول بالا نشان می‌دهد که برای هر نمونه و هر جایگاه چند امتیاز تعلق می‌گیرد. برای مثال اگر یک تیم در نمونه early در جایگاه اول قرار بگیرد، ۱۰ امتیاز خواهد گرفت و در نمونه late، ۲۵ امتیاز.

- اعمال امتیازات برای تیم‌های برتر:

در هر نمونه، به شش تیم برتر (در نمونه‌های early)، هشت تیم برتر (در نمونه‌های middle)، و ده تیم برتر (در نمونه‌های late) امتیاز داده می‌شود. اگر تیمی هیچ راه‌حلی برای یک نمونه ندهد، امتیازی برای آن نمونه دریافت نمی‌کند.

- تقسیم امتیاز در صورت مساوی بودن:

اگر دو یا چند تیم در یک جایگاه برابر قرار بگیرند، امتیازات مربوط به آن جایگاه‌ها بین آن‌ها تقسیم می‌شود (و در صورت نیاز گرد می‌شود). برای مثال اگر دو تیم به طور مشترک در جایگاه دوم باشند، امتیازات جایگاه دوم و سوم با هم جمع شده و به طور مساوی بین آن دو تقسیم می‌شود.

- محاسبه مجموع امتیازات برای هر تیم:

امتیازات هر تیم در تمامی نمونه‌ها (early، middle، late) جمع می‌شود. بر اساس مجموع امتیازات، تیم‌ها رده‌بندی می‌شوند.

- حالت تساوی در مجموع امتیازات:

اگر تیم‌هایی مجموع امتیازات برابری داشته باشند، تساوی با استفاده از روش ترتیب لغوی (lexicographic ordering) شکسته می‌شود یعنی به تیمی که در نمونه‌های early یا با امتیازات بالاتر در دیگر نمونه‌ها عملکرد بهتری داشته باشد، جایگاه بالاتری داده می‌شود.

۶. نتایج نهایی

برای مثال برای نمونه agh-fis-spr17، تیم D.Holm & R.Mikkelsen توسط سیستم ارزیابی امتیاز ۳۰۸۱ را دریافت می‌کند. سپس با توجه به مقایسه این امتیاز با سایر تیم‌ها و برتری آن‌ها در این نمونه، بر اساس جدول بالا، ۱۰ امتیاز به آن‌ها اختصاص داده می‌شود.

Early Instances

| → Instance ↓ Author | agh-fis-spr17 | agh-ggis-spr17 | bet-fal17 | iku-fal17 | mary-spr17 | muni-fi-spr16 | muni-fsps-spr17 | muni-pdf-spr16c | pu-llr-spr17 | tg-fal17 | Total Points |
|---|---|---|---|---|---|---|---|---|---|---|---|
| D. Holm & | 3081 | 35808 | 290086 | 18968 | 14910 | 3756 | 868 | 36487 | 10038 | 4215 | |
| R. Mikkelsen | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 9 | 99 |
| Efstratios | 4557 | 36616 | 295427 | 26840 | 15021 | 3844 | 883 | 37487 | 13385 | 4215 | |
| Rappos | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 9 | 72 |
| Edon | 6799 | 77932 | 299205 | 50613 | 15894 | 5006 | 1938 | 58206 | 16874 | 8044 | |
| Gashi | 3 | 3 | 5 | 3 | 5 | 5 | 5 | 5 | 5 | 2 | 41 |
| Karim | 5709 | 56755 | 313812 | 44482 | 16698 | 5207 | 4135 | 77573 | 19231 | 7358 | |
| Er-rhaimini | 5 | 5 | 3 | 5 | 3 | 3 | 3 | 3 | 3 | 3 | 36 |
| Alexandre | 35139 | 194138 | | | 51147 | 19314 | 211142 | | 68003 | 6774 | |
| Lemos | 2 | 2 | | | 2 | 2 | 2 | | 2 | 5 | 17 |

### Middle Instances

| → Instance ↓ Author | agh-ggos-spr17 | agh-h-spr17 | lums-spr18 | muni-fi-spr17 | muni-fsps-spr17c | muni-pdf-spr16 | nbi-spr18 | pu-d5-spr17 | pu-proj-fal19 | yach-fal17 | Total Points |
|---|---|---|---|---|---|---|---|---|---|---|---|
| D. Holm & | 3055 | 23502 | 95 | 3825 | 2596 | 18151 | 18014 | 15910 | 148016 | 1239 | |
| R. Mikkelsen | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 15 | 150 |
| Efstratios | 6320 | 26159 | 114 | 4289 | 3303 | 24318 | 19055 | 18813 | 561194 | 1844 | |
| Rappos | 11 | 6 | 8 | 11 | 11 | 11 | 11 | 11 | 6 | 8 | 94 |
| Edon | 9666 | 25081 | 107 | 4692 | 9222 | 40074 | 26517 | 19440 | 237909 | 1727 | |
| Gashi | 6 | 11 | 11 | 8 | 8 | 6 | 8 | 8 | 8 | 11 | 85 |
| Karim | 7725 | 25745 | 178 | 5433 | 23520 | 38826 | 30309 | 20242 | 176039 | 3181 | |
| Er-rhaimini | 8 | 8 | 6 | 6 | 6 | 8 | 6 | 6 | 11 | 6 | 71 |
| Alexandre | 79745 | 55887 | 820 | 18080 | 618217 | 310994 | 49924 | | | 32198 | |
| Lemos | 4 | 4 | 4 | 4 | 4 | 4 | 4 | | | 4 | 32 |

### Late Instances

| → Instance ↓ Author | agh-fal17 | bet-spr18 | iku-spr18 | lums-fal17 | mary-fal18 | muni-fi-fal17 | muni-fspsx-fal17 | muni-pdfx-fal17 | pu-d9-fal19 | tg-spr18 | Total Points |
|---|---|---|---|---|---|---|---|---|---|---|---|
| D. Holm & | 186200 | 348589 | 25878 | 349 | 4423 | 2999 | 17074 | 117412 | 43006 | 12704 | |
| R. Mikkelsen | 15 | 25 | 25 | 25 | 25 | 25 | 25 | 25 | 25 | 25 | 240 |
| Efstratios | | 360057 | 36711 | 386 | 5637 | 3794 | 33001 | 151464 | 134009 | 12856 | |
| Rappos | | 18 | 18 | 18 | 18 | 18 | 18 | 18 | 12 | 18 | 156 |
| Edon | 184030 | 360437 | 85969 | 486 | 7199 | 4712 | 44059 | 170061 | 82757 | 15992 | |
| Gashi | 18 | 15 | 12 | 15 | 12 | 15 | 15 | 15 | 15 | 15 | 147 |
| Karim | 153236 | 373039 | 70932 | 558 | 6944 | 4820 | 104625 | 191887 | 70450 | 19738 | |
| Er-rhaimini | 25 | 12 | 15 | 12 | 15 | 12 | 12 | 12 | 18 | 12 | 145 |
| Alexandre | | | | 1151 | 44097 | | | | | 31900 | |
| Lemos | | | | 10 | 10 | | | | | 10 | 30 |

## Total Results

| Position | Author | Early | Middle | Late | Total |
|---|---|---|---|---|---|
| 1. | D. Holm & R. Mikkelsen | 99 | 150 | 240 | 489 |
| 2. | Efstratios Rappos | 72 | 94 | 156 | 322 |
| 3. | Edon Gashi | 41 | 85 | 147 | 273 |
| 4. | Karim Er-rhaimini | 36 | 71 | 145 | 252 |
| 5. | Alexandre Lemos | 17 | 32 | 30 | 79 |

## اجرا و تحلیل روش تیم انتخاب شده

از بین روش‌های برتر مسابقه، روی روش تیمی که در مسابقه مقام سوم را کسب کرده و ۲۷۳ امتیاز گرفته بودند، تمرکز کردم. کد این تیم را از مخزن گیت‌هاب آن‌ها برداشتم و سپس آن را بر روی ۳۰ نمونه موجود در سایت مسابقه اجرا کردم. خروجی‌های به‌دست آمده را در پوشه‌ای به نام output ذخیره کردم و ورودی‌ها (نمونه‌ها) را در پوشه‌ای به نام input قرار دادم. همچنین یک پوشه دیگری به نام test نیز ایجاد کردم که یک نمونه تست به همراه خروجی مربوطه است. کدهای اصلی برنامه نیز در پوشه‌ای به نام itc-2019 قرار دارند.

برای استفاده از کد این تیم و اجرای نمونه‌ها بر روی آن ابتدا باید ابزارهای زیر را بر روی سیستم خود نصب کرد:

- .NET Core 2.1 یا بالاتر: برای ساخت و اجرای این پروژه، نیاز به .NET Core 2.1 یا نسخه‌های جدیدتر داریم که می‌توان آن را از سایت رسمی مایکروسافت دانلود و نصب کرد.
- Make: برای استفاده از دستورات Makefile نیاز به ابزار Make داریم. این ابزار به صورت پیش‌فرض روی اکثر توزیع‌های لینوکسی نصب است. برای ویندوز و macOS، می‌توان Make را از طریق بسته‌های توسعه‌دهنده مانند Cygwin یا Homebrew یا MinGW نصب کرد.

در نهایت برای ساخت فایل باینری در ویندوز به پوشه‌ای که سورس کد و Makefile قرار دارد رفته و دستور make win را درون cmd می‌زنیم تا باینری ویندوز ساخته شود. بعد از اجرای این دستور، فایل‌های خروجی در پوشه bin/win-x64 قرار خواهند گرفت.

اکنون که ابزارهای مورد نیاز نصب شده‌اند می‌توان درون پوشه مربوطه به کد آن تیم، cmd را باز کرده و دستور زیر را درون آن وارد کرد تا راه‌حل این نمونه را برایمان تولید کند:

```
C:\Users\hoori\Desktop\hoori dahesh\itc-2019>run-win.cmd --instance agh-ggis-spr17.xml
```

جای نمونه agh-ggis-spr17.xml می‌توان هر نمونه دیگری را قرار داد.

خروجی بخشی از نمونه مورد نظر:

```
-<solution name="agh-ggis-spr17" runtime="8264" cores="1" technique="Simulated Annealing (seed 957867360)" author="upfiek" institution="University of Prishtina" country="Kosovo">
   <class id="1" days="0010000" start="116" weeks="0101010010010101" room="43"/>
   <class id="2" days="1000000" start="96" weeks="1111111010111111" room="43"/>
   <class id="3" days="0100000" start="96" weeks="1111111010111111" room="43"/>
   <class id="4" days="0010000" start="136" weeks="1111111010111111" room="43"/>
   <class id="5" days="0001000" start="96" weeks="1111110111111110" room="43"/>
   <class id="6" days="0000100" start="96" weeks="0111110111111111" room="43"/>
   <class id="7" days="0010000" start="96" weeks="1111111010111111" room="43"/>
   <class id="8" days="0010000" start="116" weeks="1010101000101010" room="43"/>
  -<class id="9" days="1000000" start="176" weeks="1111111010110011" room="29">
      <student id="377"/>
      <student id="378"/>
      <student id="379"/>
      <student id="380"/>
      <student id="381"/>
      <student id="382"/>
      <student id="383"/>
      <student id="384"/>
      <student id="385"/>
      <student id="386"/>
      <student id="387"/>
      <student id="388"/>
      <student id="389"/>
      <student id="390"/>
      <student id="391"/>
      <student id="392"/>
      <student id="393"/>
      <student id="394"/>
      <student id="395"/>
      <student id="396"/>
```

تحلیل این خروجی:

- تگ <solution>:
  - name=agh-ggis-spr17: نام نمونه.
  - runtime=8264: زمان اجرای الگوریتم در ثانیه (۸۲۶۴ ثانیه).
  - cores=1: تعداد هسته‌های پردازنده‌ای که در اجرای الگوریتم استفاده شده است.
  - technique=Simulated Annealing (seed 957867360): تکنیک مورد استفاده برای حل مسئله. در اینجا از الگوریتم Simulated Annealing با یک seed خاص استفاده شده است.
  - author=upfiek: نویسنده یا تیمی که این راه‌حل را تولید کرده است.
  - institution=University of Prishtina: موسسه‌ای که نویسنده به آن تعلق دارد.
  - country=Kosovo: کشور موسسه.
- تگ‌های <class>:

هر تگ <class> نمایانگر یک کلاس در برنامه زمان‌بندی است.

  - id=1: شناسه کلاس.

- days=0010000: روزهایی که کلاس در آن‌ها برگزار می‌شود (در اینجا، کلاس تنها در روز سوم برگزار می‌شود).
- start=116: زمان شروع کلاس در هر روز است و معمولا این مقدار به دقیقه از شروع روز اشاره دارد. به این معنی که start=116 به این معناست که کلاس ۱۱۶ دقیقه بعد از شروع روز آغاز می‌شود. برای مثال، اگر روز آموزشی از ساعت ۸ صبح شروع شود و ۱۱۶ دقیقه به آن اضافه کنیم کلاس در ساعت ۹:۵۶ صبح شروع خواهد شد.
- weeks=0101010010010101: الگوی هفته‌ها که نشان می‌دهد کلاس در کدام هفته‌ها برگزار می‌شود.
- room=43: شماره اتاقی که کلاس در آن برگزار می‌شود.

- تگ‌های <student>:

هر تگ <student> نمایانگر دانشجویان ثبت‌نام شده در کلاس هستند به عبارت دیگر هر تگ <student> یک دانشجو با id مشخص را نمایندگی می‌کند. این بخش نشان می‌دهد که دانشجویان با شناسه‌های ۳۷۷، ۳۷۸، .... در کلاس با شناسه ۹ ثبت‌نام کرده‌اند.

این مشخصات برای هر کلاس متفاوت است. برای مثال کلاس با id=9 در روزهای اول (days=1000000) و از دقیقه ۱۷۶ (start=176) و در اتاق شماره ۲۹ (room=29) برگزار می‌شود. علاوه بر این، لیستی از دانشجویان (<student>) که در این کلاس ثبت‌نام کرده‌اند، وجود دارد.

این تیم از کدهای ابزار Timetabling به عنوان پایه کار خود استفاده کرده و آن را بر اساس نیازهای خود سفارشی‌سازی کرده‌اند. آنها در روش پیشنهادی خود، از دو الگوریتم با نام‌های Simulated Annealing و Focused Search on Particular Constraints بهره برده‌اند.

الگوریتم اصلی مورد استفاده آن‌ها Simulated Annealing است، اما از نسخه اصلاح شده‌ای از این الگوریتم استفاده کرده‌اند که شامل تابع‌های Cooling Function و Evaluation Function است. همچنین این الگوریتم در هر دو ناحیه قابل قبول و غیرقابل قبول فضای راه‌حل جستجو می‌کند، که در ناحیه غیرقابل قبول از ترکیبی از جریمه تدریجی و جستجوی محدود شده بر روی محدودیت‌های سخت خاص استفاده می‌شود.

در ادامه، به توضیح دقیق‌تر کارهایی که انجام داده‌اند و توابعی که در این فرآیند به کار گرفته‌اند، پرداخته می‌شود.

**الگوریتم Simulated Annealing**

برای بهبود الگوریتم Hill-Climbing، یکی از روش‌های موثر استفاده از الگوریتم Simulated Annealing است. به عبارت دیگر، Simulated Annealing از Hill-Climbing توسعه یافته است. در این الگوریتم از یک حالت ابتدایی شروع کرده و همسایه‌های آن را بررسی می‌کنیم. هر همسایه یک heuristic نسبت به heuristic ما داره که اختلاف بین این heuristicها به عنوان ΔE شناخته می‌شود. اگر ΔE مثبت باشد، احتمالا heuristic بهتر شده است و همسایه را انتخاب می‌کنیم. اما اگر ΔE منفی باشد، همسایه را با احتمالی که به صورت $e^{\frac{-\Delta E}{T}}$ محاسبه می‌شود، انتخاب می‌کنیم. این احتمال به‌تدریج با کاهش دما (T) کاهش می‌یابد. دما (T) نمایانگر گذر زمان است و کاهش آن باعث می‌شود که انتخاب حالت‌های بدتر کمتر شود و الگوریتم به سمت بهینه شدن حرکت کند.

الگوریتم Simulated Annealing یک الگوریتم جستجو و بهینه‌سازی تصادفی است که برای حل مسائل بهینه‌سازی پیچیده و بزرگ استفاده می‌شود. این الگوریتم از فرآیندهای فیزیکی بازپخت فلزات الهام گرفته است.

اصول پایه‌ای Simulated Annealing:

۱. الهام از فیزیک: این الگوریتم از فرآیند فیزیکی بازپخت فلزات الهام گرفته است، که در آن فلزات به تدریج با دمای پایین‌تر سرد می‌شوند تا به حالت بهینه‌ای از ساختار بلوری برسند.
۲. ابتدا دما بالا: در الگوریتم‌های بهینه‌سازی، هدف پیدا کردن بهترین راه‌حل (یا مجموعه‌ای از راه‌حل‌ها) برای یک مسئله مشخص است. این راه‌حل می‌تواند در یک فضای بسیار بزرگ از همه راه‌حل‌های ممکن قرار داشته باشد. در این حالت، الگوریتم به گونه‌ای عمل می‌کند که به راحتی فضای جستجو را بررسی می‌کند، بدون اینکه فقط به راه‌حل‌های محلی بسنده کند و سعی می‌کند بهترین راه‌حل را در کل فضای جستجو پیدا کند.
۳. کاهش دما: در طول زمان، دما به تدریج کاهش می‌یابد. کاهش دما به‌صورت تدریجی موجب می‌شود که الگوریتم به تدریج بیشتر به سمت جستجوی محلی متمایل شود و از انتخاب‌های تصادفی کمتر استفاده کند.
۴. پذیرش راه حل های بد: در هر مرحله، الگوریتم ممکن است راه‌حل‌هایی را که از نظر کیفیت بدتر از راه‌حل فعلی هستند، با احتمال معین بپذیرد. این احتمال بر اساس تابع دما و تفاوت در کیفیت دو راه‌حل محاسبه می‌شود. این ویژگی به الگوریتم کمک می‌کند تا از مینیمم‌های محلی خارج شود و به جستجوی بهینه‌تری برسد به عبارت دیگر الگوریتم برای پیدا کردن بهترین راه‌حل (بهینه‌سازی) به کار می‌رود و هدفش این است که از گیر افتادن در مینیمم‌های محلی جلوگیری کند. در الگوریتم‌های معمولی بهینه‌سازی، تنها راه‌حل‌های بهتر پذیرفته می‌شوند اما در اینجا الگوریتم ممکن است بعضی از راه‌حل‌های بدتر از راه‌حل فعلی را هم با یک احتمال مشخص قبول کند. در ابتدا، دما بالا است و امکان پذیرش راه‌حل‌های بدتر بیشتر است. با گذشت زمان، دما کاهش می‌یابد و احتمال پذیرش راه‌حل‌های بدتر هم کاهش می‌یابد. راه‌حل‌های محلی (Local Minima) به نقاطی در فضای جستجو گفته می‌شود که از نظر کیفیت بهتر از راه‌حل‌های اطراف خود هستند، اما ممکن است بهترین راه‌حل ممکن نباشند. بنابراین، این ویژگی به الگوریتم اجازه می‌دهد که از گیر افتادن در این مینیمم‌های محلی جلوگیری کرده و فضای جستجو را به طور کامل‌تری بررسی کند تا به یک راه‌حل بهینه‌تر برسد.
۵. تابع پذیرش: تابع پذیرش، که معمولا تابع نمایی است، برای تصمیم‌گیری در مورد پذیرش یا رد یک راه‌حل جدید استفاده می‌شود.
۶. تکرار: الگوریتم به‌طور تکراری راه‌حل‌های جدیدی را تولید می‌کند، دما را کاهش می‌دهد و تصمیم می‌گیرد که آیا راه‌حل جدید را بپذیرد یا نه تا زمانی که دما به اندازه کافی پایین بیاید و فرآیند جستجو به پایان برسد.

## تحلیل روش

سه نوع جریمه برای ارزیابی وضعیت راه‌حل در نظر گرفته شده است: جریمه سخت، جریمه بیش از ظرفیت کلاس‌ها و جریمه نرم.

جریمه نرم، محدودیت‌هایی است که رعایت آن‌ها الزامی نیست اما بهتر است رعایت شوند تا کیفیت کلی برنامه بهبود یابد. این محدودیت‌ها برخلاف محدودیت‌های سخت که نقض آن‌ها قابل قبول نیست، انعطاف بیشتری دارند و اگر نقض شوند فقط

منجر به یک جریمه سبک‌تر (جریمه نرم) می‌شوند. برای مثال داشتن زمان استراحت بین کلاس‌ها یا تخصیص تعداد دانشجویان کمتر از ظرفیت اتاق، محدودیت‌های نرم هستند که نقض آن‌ها تاثیر مستقیم جدی ندارد اما رعایتشان مطلوب است. جریمه نرم معمولا به ازای هر نقض به‌طور جزئی محاسبه می‌شود.

جریمه سخت به صورت زیر است:

۱. تداخل بین دو کلاس، ۱ امتیاز جریمه سخت می‌دهد.
۲. تخصیص زمانی که با برنامه غیرقابل استفاده یک اتاق تداخل داشته باشد، ۱ امتیاز جریمه سخت می‌دهد یعنی اگر یک زمان خاص به اتاقی اختصاص داده شود که در آن زمان به دلایلی غیرقابل استفاده است (مثل در حال تعمیر بودن)، این تخصیص باعث جریمه شدن می‌شود.
۳. عدم برآورده شدن یک محدودیت ضروری، امتیازهای جریمه سختی معادل با جریمه نرم آن محدودیت (در صورتی که ضروری نبود) می‌دهد.

جریمه اضافه ظرفیت کلاس‌ها برابر با مجموع تمام ثبت‌نام‌های اضافی در کلاس‌ها است. ما این جریمه را به‌صورت جداگانه نگه می‌داریم زیرا به اندازه جریمه سخت محدودیت ندارد و راحت‌تر برآورده می شود.

تابع همسایگی:

این تابع کارش ایجاد تغییرات کوچک (جهش) در راه‌حل فعلی برای جستجوی راه‌حل‌های بهتر است.

جهش:

یک جهش به معنای تغییر کوچک در یک متغیر است. در اینجا، متغیرها مربوط به برنامه‌ریزی زمانی (زمان‌بندی کلاس‌ها، تخصیص اتاق‌ها و غیره) هستند. این تغییرات می‌توانند به صورت تغییر زمان کلاس، تغییر اتاق، یا تغییر پیکربندی کلاس یک دانشجو باشند.

الگوریتم دو لیست از جهش‌ها را نگه می‌دارد:

۱. جهش‌های قابل قبول: این‌ها جهش‌هایی هستند که ممکن است راه‌حل را بهبود دهند یا آن را در یک سطح قابل قبول نگه دارند.
۲. جهش‌های غیرقابل قبول: این جهش‌ها روی راه‌حل‌هایی اعمال می‌شوند که جریمه سخت غیرصفر دارند (یعنی راه‌حل‌هایی که مشکلات جدی دارند و نیاز به اصلاح فوری دارند). همچنین، این جهش‌ها شامل تغییراتی نمی‌شوند که دانشجویان را تحت تاثیر قرار دهند بلکه بر عناصر دیگری مانند زمان و اتاق تمرکز دارند.

تابع Cooling Function:

این تابع کنترل می‌کند که چگونه دما در طول زمان کاهش می‌یابد. این کاهش دما تعیین می‌کند که الگوریتم تا چه اندازه می‌تواند جهش‌های بزرگ (یعنی تغییرات ناگهانی یا بزرگتر) را بپذیرد. در دمای بالا، جهش‌های بدتر احتمالا پذیرفته می‌شوند اما با کاهش دما، فقط جهش‌های بهبود دهنده پذیرفته می‌شوند.

تابع Evaluation Function:

این تابع برای محاسبه یک امتیاز یا جریمه برای یک راه‌حل استفاده می‌شوند، که به الگوریتم کمک می‌کند تا بفهمد کدام راه‌حل‌ها بهتر هستند.

به طور کلی، این توابع به الگوریتم کمک می‌کنند تا میزان کیفیت یک راه‌حل را بر اساس دو نوع جریمه (سخت و نرم) ارزیابی کند. جریمه سخت نشان‌دهنده شکست در رعایت محدودیت‌های ضروری است و وزن بیشتری دارد، در حالی که جریمه نرم به محدودیت‌های ثانویه اشاره دارد و وزن کمتری دارد. نرمالایز کردن جریمه نرم نیز برای این است که این جریمه‌ها با توجه به بدترین حالت ممکن استانداردسازی شوند و مقایسه‌پذیری بهتری داشته باشند.

شرط پذیرش در Simulated Annealing:

این الگوریتم از یک تابع به نام f_stun برای محاسبه تفاوت کیفیت بین راه‌حل فعلی و جدید استفاده می‌کند. این تفاوت به صورت انرژی یا $\Delta E$ محاسبه شده و برای تصمیم‌گیری در مورد پذیرش یا رد یک راه‌حل جدید به کار می‌رود. ثابت $\gamma$ نیز به منظور تنظیم حساسیت این تصمیم‌گیری استفاده می‌شود.

الگوریتم Simulated Annealing برای پذیرش یا رد یک راه‌حل جدید از یک شرط پذیرش استفاده می‌کند که بر اساس تفاوت بین مقدارهای تابع f_stun برای دو راه‌حل است. این شرط پذیرش تفاوت انرژی یا $\Delta E$ بین راه‌حل فعلی (s) و راه‌حل جدید (s') را بررسی می‌کند.

تابع f_stun:

این تابع برای محاسبه احتمالی استفاده می‌شود که به الگوریتم کمک می‌کند تا تصمیم بگیرد آیا یک راه‌حل جدید پذیرفته شود یا نه:

$$f_{\text{stun}}(x) = 1 - \exp\left[-\gamma(f(x) - f_0)\right]$$

f(x) نشان‌دهنده کیفیت راه‌حل x است.

$f_0$ کیفیت بهترین راه‌حل تاکنون را نشان می‌دهد.

$\gamma$ یک ثابت است که به صورت تجربی تعیین شده است و حساسیت تابع را تنظیم می‌کند.

exp تابع نمایی است که نشان‌دهنده کاهش احتمال پذیرش راه‌حل‌های بدتر با کاهش دما است.

تفاوت انرژی یا $\Delta E$:

$\Delta E$ که در معادله زیر، به صورت تفاوت بین مقدارهای f_stun برای راه‌حل فعلی و راه‌حل جدید محاسبه می‌شود:

$$\Delta E(s', s) = f_{\text{stun}}\left(\text{searchPenalty}(s')\right) - f_{\text{stun}}\left(\text{searchPenalty}(s)\right)$$

s نمایانگر راه‌حل فعلی و 's نمایانگر راه‌حل جدید است.

searchPenalty تابعی است که جریمه‌ی راه‌حل را محاسبه می‌کند (بر اساس جریمه‌های نرم و سخت که بالاتر گفته شد.)

این تفاوت انرژی به الگوریتم کمک می‌کند تا تصمیم بگیرد که آیا به سمت راه‌حل جدید حرکت کند یا خیر. اگر $\Delta E$ مثبت باشد، این به معنای بهبود است در غیر این صورت الگوریتم با توجه به تابع احتمالی f_stun تصمیم می‌گیرد که آیا جهش را بپذیرد یا نه.

الگوریتم Focused Search on Particular Constraints:

این الگوریتم به عنوان بخشی از فرآیند جریمه‌دهی و بهینه‌سازی استفاده می‌شود. وقتی که الگوریتم اصلی ( Simulated Annealing) به نتایج مطلوب نمی‌رسد یا به نظر می‌رسد در بهبود برخی از محدودیت‌های سخت (مانند تداخلات زمانی و مکانی) به مشکل برخورده است این الگوریتم به کار گرفته می‌شود. هدف آن ایجاد تغییرات تصادفی و بررسی بهبود در جریمه‌های متمرکز برای برخی از محدودیت‌هاست تا بتواند از دام کمینه‌های محلی خارج شود و راه‌حل بهتری را بیابد. به طور خلاصه، این الگوریتم نوعی تنگ کردن جستجو برای بهبود محدودیت‌های خاص است که در صورتی که به صورت مکرر با مشکل مواجه شوند، به کار می‌رود.

**مزیت‌های این دو الگوریتم**

الگوریتم‌های Simulated Annealing (SA) و Focused Search به دلیل قابلیت‌های فرار از کمینه‌های محلی، مدیریت پیچیدگی محاسباتی، برخورد با محدودیت‌های سخت، و انعطاف‌پذیری در مواجهه با مسائل مختلف، نسبت به روش‌های دیگر (مانند جستجوی کامل یا الگوریتم‌های ابتکاری) برتری دارند. این دو الگوریتم به‌ویژه در مسائل پیچیده مانند زمان‌بندی و مسائل بهینه‌سازی چندمحدودیتی که نیاز به جستجوی هوشمند و مدیریت هزینه‌های محاسباتی دارند، بسیار مؤثر هستند. مزیت‌های این دو الگوریتم به شرح زیر است:

۱. فرار از کمینه‌های محلی:
   - SA: برخلاف بسیاری از الگوریتم‌های جستجوی محلی (Local Search) که ممکن است در کمینه‌های محلی گرفتار شوند، SA این توانایی را دارد که با پذیرش موقت راه‌حل‌های با کیفیت پایین‌تر، از این کمینه‌ها عبور کند و فضای جستجوی گسترده‌تری را کاوش کند. این ویژگی به آن کمک می‌کند تا به راه‌حل‌های بهتر و حتی بهینه نزدیک شود.
   - مزیت نسبت به جستجوی کامل یا جستجوی محلی ساده: در جستجوی کامل یا جستجوی محلی، احتمال گیر افتادن در کمینه‌های محلی بالاست، در حالی که SA از این مشکل رها می‌شود.

۲. مدیریت پیچیدگی محاسباتی:
- SA: این الگوریتم به تدریج و بر اساس یک برنامه سرد کردن بهینه‌سازی می‌کند. برخلاف روش‌های جستجوی کامل که نیاز به ارزیابی تمام راه‌حل‌های ممکن دارند، SA می‌تواند با ارزیابی تعداد کمتری از راه‌حل‌ها به نتایج خوب برسد، که به صرفه‌جویی در زمان و منابع منجر می‌شود.
- Focused Search: این الگوریتم با تمرکز روی مناطق مشکل‌زا (مثل محدودیت‌های سخت) هزینه محاسباتی را بهینه می‌کند. به جای جستجوی کلی، به صورت هدفمند به بخش‌های مشکل‌زا می‌پردازد و زمان محاسباتی را کاهش می‌دهد.
- مزیت نسبت به روش‌های جستجوی کامل یا الگوریتم‌های سنتی: روش‌های سنتی زمان و منابع بیشتری برای کاوش کل فضای جستجو مصرف می‌کنند، در حالی که این دو الگوریتم به‌طور هوشمندانه پیچیدگی را مدیریت می‌کنند.

۳. برخورد موثر با محدودیت‌های سخت:
- SA: توانایی SA در حرکت بین راه‌حل‌های معتبر و نامعتبر و پذیرش تدریجی راه‌حل‌های بهتر، آن را برای مسائل با محدودیت‌های سخت مناسب می‌کند.
- Focused Search: این روش به صورت مستقیم بر روی محدودیت‌های سخت تمرکز می‌کند و مشکلاتی را که ممکن است در SA برطرف نشوند، بهینه می‌سازد. این ترکیب به حل بهتر مسائل پیچیده کمک می‌کند.
- مزیت نسبت به الگوریتم‌های مبتنی بر جستجوی ساده: در بسیاری از روش‌های جستجوی ساده، مدیریت محدودیت‌های سخت دشوار است و ممکن است به راه‌حل‌های نامعتبر ختم شود. اما این دو الگوریتم با تمرکز بر حل مشکلات خاص، محدودیت‌ها را به خوبی مدیریت می‌کنند.

۴. تطبیق‌پذیری و انعطاف‌پذیری:
- SA: این الگوریتم برای طیف گسترده‌ای از مسائل بهینه‌سازی قابل تنظیم است و به راحتی می‌تواند برای مسائل مختلف از جمله مسائل زمان‌بندی، مسائل گسسته، و مسائل پیوسته استفاده شود.
- Focused Search: این الگوریتم در شرایطی که SA به بهبود محدود می‌رسد یا با مشکل مواجه می‌شود، با تمرکز بر روی مناطق خاص مسئله، قابلیت سازگاری بالایی نشان می‌دهد و به الگوریتم اجازه می‌دهد که مسیر بهبود را ادامه دهد.
- مزیت نسبت به روش‌های دیگر: بسیاری از روش‌های دیگر برای مسائل خاص و با تنظیمات محدود کاربرد دارند، اما SA و Focused Search به دلیل انعطاف‌پذیری بالا در شرایط مختلف بهینه‌سازی مفیدترند.

۵. کارایی جستجو:
- SA: به دلیل سرد کردن تدریجی و جستجوی هوشمندانه، این روش جستجوی بی‌هدف را به حداقل می‌رساند و به سمت بهینه‌تر شدن راه‌حل‌ها هدایت می‌شود.
- Focused Search: با تمرکز بر روی مناطق خاص و محدود کردن جستجو به مناطق کلیدی، هزینه‌های محاسباتی را کاهش داده و جستجو را کارآمدتر می‌کند.
- مزیت نسبت به الگوریتم‌های جستجوی مبتنی بر تصادف (Random Search): روش‌های تصادفی معمولا بهینه‌سازی هدفمند ندارند، اما این دو روش به صورت هوشمندانه جستجو را بهینه می‌کنند.

## منابع

https://www.itc2019.org/home

https://link.springer.com/article/10.1007/s10951-023-00801-w