به نام خدا

# حل مسئله جدول زمانی در دانشگاه‌ها با یک روش خوب

درس:
هوش مصنوعی

استاد درس:
دکتر حکیم داودی

نام دانشجو:
حوری دهش

شهریور ۱۴۰۳

# 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 اصلی (Parent Courses)
- پیکربندی‌ها (Sub-courses)

۲. ویژگی‌های زمانی و مکانی کلاس‌ها

۳. ویژگی‌های اتاق‌ها و محدودیت‌ها

# توضیح مسئله

۴. محدودیت‌های تخصیص منابع و جریمه‌ها

- محدودیت‌های سخت
  - 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
  - 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.

۵. معیارهای بهینه‌سازی

# 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**

- فراهم آوردن مجموعه داده استاندارد
- ارزیابی الگوریتم‌ها
- تشویق به نوآوری

**توصیف نمونه‌های استفاده شده در مسابقه**

در مسابقه ITC 2019، ۳۰ نمونه آزمون (benchmark instances) منتشر شد که شامل:

- Early
- middle
- late

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.

**تحلیل دقیق‌تر:**

- Early Instances

نمونه‌ها در اوایل مسابقه

مشکلات کوچکتر با تعداد کلاس‌ها، دانشجوها و اتاق‌های کمتر

شروع و آشنایی با مسئله

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.

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.

- Middle Instances

  مسئله پیچیده‌تر

  اندازه نمونه‌ها افزایش یافته

  تعداد بیشتری کلاس و دانشجو

  محدودیت‌های بیشتر

  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

  نمونه‌های نهایی و پیچیده‌ترین نمونه‌ها

  منتشر شده در انتهای مسابقه

  معمولا بسیار بزرگ هستند و شامل تعداد زیادی دانشجو، کلاس و اتاق

  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
- نوع محدودیت‌ها

**ویژگی‌های نمونه‌های استفاده شده در مسابقه**

- اندازه مسئله
- تقاضاهای دانشجویان
- محدودیت‌های توزیع
- دامنه متغیرها
- الگوهای زمانی (زمان‌ها)
- بهره‌برداری از اتاق‌ها
- وزن‌های بهینه‌سازی

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.

**به عنوان مثال می‌خواهیم نمونه agh-ggis-spr17 از دسته Early را تحلیل کنیم.**

```
-<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>
```

```
-<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"/>
```

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

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

```
- <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>
```

# 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.

**مراحل چک کردن و امتیازدهی به صورت زیر است:**

۱. تایید رعایت محدودیت‌های سخت

۲. بررسی محدودیت‌های نرم

۳. محاسبه جریمه‌ها

۴. مقایسه با سایر تیم‌ها

# نحوه ارزیابی راه‌حل تیم‌ها

۵. اختصاص امتیازات

- مراحل ارزیابی
  - تقسیم امتیازات بر اساس نمونه‌های early، middle، late
  - اعمال امتیازات برای تیم‌های برتر
  - تقسیم امتیاز در صورت مساوی بودن
  - محاسبه مجموع امتیازات برای هر تیم
  - حالت تساوی در مجموع امتیازات

| Position | Instance | | |
|---|---|---|---|
| | 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

# نحوه ارزیابی راه‌حل تیم‌ها

۶. نتایج نهایی

## 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 |

# نحوه ارزیابی راه‌حل تیم‌ها

۶. ادامه نتایج نهایی

## 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 روش تحلیل 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.

- 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
  - Make

- دستور مورد نیاز برای تولید خروجی روی نمونه‌های مسابقه:

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

# اجرا و روش تحلیل تیم انتخاب شده

## خروجی نمونه agh-ggis-spr17.xml

```
-<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"/>
```

# اجرا و روش تحلیل تیم انتخاب شده

این تیم از کدهای ابزار Timetablingبه 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.

# 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.

# الگوریتم Simulated Annealing

اصول پایه‌ای Simulated Annealing

- الهام از فیزیک
- ابتدا دما بالا
- کاهش دما
- پذیرش راه حل های بد
- تابع پذیرش
- تکرار

# تحلیل روش

سه نوع جریمه برای ارزیابی وضعیت راه‌حل در نظر گرفته شده است:

- جریمه نرم

جریمه نرم، محدودیت‌هایی است که رعایت آن‌ها الزامی نیست اما بهتر است رعایت شوند تا کیفیت کلی برنامه بهبود یابد.

- جریمه سخت
  - تداخل بین دو کلاس، ۱ امتیاز جریمه سخت می‌دهد.
  - 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.
  - عدم برآورده شدن یک محدودیت ضروری، امتیازهای جریمه سختی معادل با جریمه نرم آن محدودیت (در صورتی که ضروری نبود) می‌دهد.

- جریمه بیش از ظرفیت کلاس‌ها

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.

# تحلیل روش

- 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.

  الگوریتم دو لیست از جهش‌ها را نگه می‌دارد:

  - جهش‌های قابل قبول
  - جهش‌های غیرقابل قبول

- تابع Cooling Function

  این تابع کنترل می‌کند که چگونه دما در طول زمان کاهش می‌یابد

- تابع Evaluation Function:

  این تابع برای محاسبه یک امتیاز یا جریمه برای یک راه‌حل استفاده می‌شوند.

# شرایط 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:

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

تفاوت انرژی یا $\Delta E$:

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

# الگوریتم Focused Search

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

۱. فرار از کمینه‌های محلی

مزیت نسبت به جستجوی کامل یا جستجوی محلی ساده

۲. مدیریت پیچیدگی محاسباتی

مزیت نسبت به روش‌های جستجوی کامل یا الگوریتم‌های سنتی

۳. برخورد موثر با محدودیت‌های سخت

مزیت نسبت به الگوریتم‌های مبتنی بر جستجوی ساده

۴. تطبیق‌پذیری و انعطاف‌پذیری

مزیت نسبت به روش‌های دیگر

۵. کارایی جستجو

مزیت نسبت به الگوریتم‌های Random Search

# منابع

https://www.itc2019.org/home

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