الگوریتم های حل مسئله ی زمان بندی کلاس های دانشگاهی

University course time-tabling problem (UCTP)

نویسنده اول : آرمیتا اصحاب یمین

Armita.ay1379@gmail.com

چکیده مطالب

زمان‌بندی دروس دانشگاهی یکی از مسائل مهم و وقت‌گیر است که هر دانشگاه در ابتدای ترم با آن درگیر است. این مشکل در دسته‌ی مسائل NP-hard قرار دارد و حل آن با الگوریتم‌های کلاسیک بسیار دشوار است. بنابراین، تکنیک‌های بهینه‌سازی برای حل آن‌ها و تولید راه‌حل‌های بهینه یا تقریبا بهینه به جای راه‌حل‌های دقیق استفاده می‌شود. الگوریتم‌های ژنتیک به خاطر ویژگی جستجوی چندجانبه‌شان، به‌عنوان رویکردی کارآمد برای حل این نوع مسائل در نظر گرفته می‌شوند. در این تحقیق در ابتدا درباره ی مقدمات حل این مسئله و متغیر های آن می پردازیم و با چندین راه حل متداول از جمله الگوریتم های کلونی مورچه ها، جستجو تابو و جستجو انیلینگ برای حل این مسئله آشنا می شویم. سپس به مقایسه ی آن الگوریتم ها می پردازیم و در آخر سه الگوریتم ژنتیک هیبریدی جدید برای حل مشکل زمان‌بندی دروس دانشگاهی (UCTP) پیشنهاد شده است: FGARI، FGASA و FGATS. در الگوریتم‌های پیشنهادی، از منطق فازی برای اندازه‌گیری نقض محدودیت‌های نرم در تابع تناسب استفاده شده است تا با عدم قطعیت و ابهام‌های ذاتی داده‌های واقعی کنار بیاییم. همچنین، جستجوی محلی تکراری تصادفی، آنیلینگ شبیه‌سازی شده و جستجوی تابو به‌ترتیب برای بهبود توانایی جستجوی اکتشافی و جلوگیری از قفل شدن الگوریتم ژنتیک در بهینه محلی اعمال شده‌اند. نتایج تجربی نشان می‌دهد که الگوریتم‌های پیشنهادی قادر به تولید نتایج امیدوارکننده برای UCTP هستند.

واژگان کلیدی: الگوریتم زمانبندی کلاس های دانشگاهی، جستجو محلی،جستجو آنیلینگ، جستجو تابو، جستجو ژنتیک ترکیبی

مقدمه

مسئله‌ی زمان‌بندی دروس دانشگاهی یکی از چالش‌های دشواری است که مؤسسات آموزشی با آن روبرو هستند. حل کردن یک مشکل زمان‌بندی دانشگاه به صورت دستی معمولاً نیاز به صرف زمان زیاد و منابع پرهزینه دارد . به منظور مدیریت پیچیدگی این مشکلات و فراهم کردن حمایت خودکار برای زمان‌بندی‌های انسانی، سال‌ها تحقیق در این زمینه انجام شده است. [1] مشکل زمان‌بندی دروس دانشگاهی شامل برنامه‌ریزی کلاس‌ها، دانش‌آموزان، معلمان و اتاق‌ها در تعداد مشخصی از زمان‌هاست، به گونه‌ای که یک مجموعه از محدودیت‌ها را برآورده کند، که اغلب حل این مشکل را در شرایط واقعی بسیار دشوار می‌کند. به طور خلاصه، دو نمونه بارز از مشکل زمان‌بندی دروس دانشگاهی وجود دارد: زمان‌بندی دروس مبتنی بر برنامه درسی و زمان‌بندی دروس پس از ثبت‌نام. [7]

هر دو نوع مشکل به طور مکرر در گذشته حل شده‌اند. در زمان‌بندی مبتنی بر برنامه درسی، تعارضات بین دروس توسط برنامه‌های درسی منتشر شده توسط دانشگاه تعیین می‌شود. تعارضات در زمان‌بندی پس از ثبت‌نام به طور مستقیم توسط دانشجویانی که به طور فردی در دروس خاص ثبت‌نام می‌کنند، مشخص می‌گردد. [2] [8] به دلیل تنوع مشکل، تنوع محدودیت‌ها و الزامات خاص هر دانشگاه، پیدا کردن یک راه‌حل عمومی و موثر برای زمان‌بندی بسیار دشوار است. هیچ الگوریتم چندجمله‌ای قطعی شناخته شده‌ای برای UCTP وجود ندارد، چون این یک مشکل بهینه‌سازی ترکیبی NP-hard است.

تنوع وسیعی از مقالات، از حوزه‌های تحقیقاتی عملیاتی و هوش مصنوعی، به مشکلات گسترده زمان‌بندی دانشگاهی پرداخته‌اند. تحقیقات اولیه زمان‌بندی بر روی هوریستیک‌های ترتیبی متمرکز بود که روشی ساده‌تر و راحت‌تر برای حل مشکلات رنگ‌آمیزی گراف‌ها به حساب می‌آمد. ایده اصلی این بود که رویدادها را یکی یکی زمان‌بندی کنند و از سخت‌ترین رویدادها شروع کنند. محققان روش‌های مختلفی برای زمان‌بندی ارائه داده‌اند که شامل روش‌های مبتنی بر محدودیت، رویکردهای مبتنی بر گراف، روش‌های مبتنی بر خوشه، رویکردهای مبتنی بر جمعیت، روش‌های متا-هوریستیک، رویکردهای چندمعیاری، رویکردهای هایپر-هوریستیک/خودسازگار، استدلال مبتنی بر مورد، و رویکردهای مبتنی بر دانش فازی می‌شود. نتیجه‌گیری شده است که GAs متداول نتایج خوبی در میان چندین رویکرد توسعه‌یافته برای UCTP به دست نمی‌آورند. بنابراین، GAs متداول نیاز به بهبود دارند تا UCTP را بهینه تر حل کنند. [8]

متغیر های مسئله:

توضیح مساله زمانبندی کلاس‌ها در این تحقیق بر اساس برنامه درسی است که معمولاً در ایران استفاده می‌شود. این مساله شامل موجودیت‌های زیر است:

روزها و زمان‌ها: تعداد مشخصی از روزهای کاری در هر هفته در نظر گرفته شده است. تعداد روزهای کاری ۵ فرض شده است. هر روز به یک تعداد ثابت زمان تقسیم می‌شود که برای همه روزها یکسان است.

در اینجا تعداد زمان‌های هر روز ۹ در نظر گرفته شده که هر یک از آن‌ها یک ساعت است. بنابراین، تعداد کل زمان‌ها ۴۵ است. زمان‌ها به ترتیب شماره‌گذاری شده‌اند از ۱ تا ۴۵، همان‌طور که در بسیاری از مطالعات استفاده شده است. زمان‌ها به‌صورت مجموعه $T = \{t1, t2, ..., t45\}$ نمایش داده می‌شوند.

اتاق‌ها: هر اتاق دارای ظرفیت است که به‌صورت تعداد صندلی‌های قابل استفاده و امکانات خاص بیان می‌شود.

برنامه‌های درسی: یک برنامه درسی مجموعه‌ای از دوره‌ها است که هر دو دوره در این مجموعه دانشجویان مشترک دارند. ویژگی اصلی دوره‌ها در یک برنامه درسی این است که نباید تداخل داشته باشند. [1]

اساتید: هر استاد برنامه زمانی خاصی برای حضور در دانشگاه دارد و در ارائه موضوعات خاص تخصص دارد.

دوره‌ها: هر دوره دارای یک بازه زمانی ثابت است و به موضوع خاصی مربوط می‌شود و نیاز دارد که در اتاقی با ظرفیت و امکانات خاص برگزار شود.

با توجه به موارد فوق، UCTP تخصیص زمان، استاد و اتاق به مجموعه‌ای از دوره‌ها است به گونه‌ای که تمام **محدودیت‌های سخت** مورد نظر رعایت شده و **محدودیت‌های نرم** تا حد امکان برآورده شوند.[3]

محدودیت‌های سختی که باید رعایت شوند تا برنامه زمانی قابل قبول باشد به شرح زیر است:

1. دوره‌ها در هر برنامه درسی نباید تداخل داشته باشند.
2. هر اتاق نباید در یک زمان خاص بیش از یک دوره داشته باشد.
3. هر استاد نباید در یک زمان خاص به بیش از یک اتاق اختصاص داده شود.
4. هر استاد تنها باید در روزهایی که در دانشگاه حضور دارد تدریس کند.
5. کلاسی که به یک دوره اختصاص داده شده باید دارای امکانات و ظرفیت مورد نیاز آن دوره باشد.

محدودیت‌های نرمی که باید رعایت شوند تا برنامه زمانی به عنوان برنامه‌ای با کیفیت بالا در نظر گرفته شود به شرح زیر است:

1. هر دوره به استادی اختصاص داده می‌شود که در زمینه تخصص او باشد.
2. برنامه زمانی هر استاد باید مشابه برنامه زمانی ارائه شده توسط استاد باشد.

3. حداقل و حداکثر تعداد ساعات حضور هر دانشجو در روز باید رعایت شود.

4. سخنرانی‌های مربوط به یک برنامه درسی باید به یکدیگر نزدیک باشند.

5. کلاسی در آخرین زمان‌های یک روز برنامه‌ریزی نشده باشد.

۶. حداقل و حداکثر تعداد ساعات حضور هر استاد تأمین می‌شود.

۷. حضور دانشجو در ساعات متوالی در هر روز.

نکته: تعریف دقیق محدودیت های سخت و نرم به دانشجویان، اساتید و سازمان آموزشی مورد نظر بستگی دارد. [7]

الگوریتم های کاربردی:

جستجو Annealing:

یکی از مؤثرترین تکنیک‌ها برای حل مسائل زمان‌بندی، شبیه‌سازی انجمادی (SA) است که یک متاهیوریستیک احتمالی برای نزدیک شدن به نقطه ی بهینه کلی یک تابع خاص است.

آنیلینگ شبیه‌سازی شده الهام‌گرفته از فرایند آنیلینگ در متالورژی است، جایی که خنک کردن کنترل‌شده یک ماده منجر به رسیدن به حالت انرژی حداقلی می‌شود. الگوریتم SA به‌طور تصادفی فضای راه‌حل را بررسی کرده و هم به بهبودها و هم برخی از راه‌حل‌های «بدتر» را قبول می‌کند تا از مینیمم‌های محلی فرار کند. این قابلیت به SA اجازه می‌دهد تا به‌طور مؤثری در چشم‌اندازهای بهینه‌سازی پیچیده و چند قله‌ای حرکت کند و آن را برای مشکل زمان‌بندی مناسب می‌سازد .

الگوریتم SA در مراحل اصلی زیر عمل می‌کند :

۱.ابتدا: با یک راه‌حل تصادفی آغاز کنید و دمای اولیه (T) را تنظیم کنید که پذیرش راه‌حل‌های بدتر را کنترل می‌کند .

۲.برای هر تکرار :

- یک راه‌حل جدید با ایجاد یک تغییر کوچک در راه‌حل فعلی تولید کنید که به آن اختلال می‌گویند .
- انرژی (یا هزینه) راه‌حل جدید را محاسبه کنید .

۳.برنامه خنک‌سازی: به تدریج دما را طبق یک برنامه خنک‌سازی از پیش تعریف‌شده که می‌تواند خطی، نمایی یا لگاریتمی باشد، کاهش دهید .

۴.پایان: الگوریتم زمانی متوقف می‌شود که معیاری از پیش تعیین‌شده برآورده شده باشد، چه محدودیت زمانی، کیفیت هدف راه‌حل، یا پس از تعداد مشخصی از تکرارها.

برنامه خنک‌کنندگی برای عملکرد الگوریتم SA بسیار مهم است. یک برنامه خنک‌کنندگی آهسته اجازه می‌دهد تا فضای حل به طور کامل‌تری جستجو شود، در حالی که یک برنامه سریع می‌تواند به سرعت به حداقل محلی نزدیک شود. شایان به ذکر است که در این الگوریتم دما به صورت دائم در حال کمتر شدن است، بدین معنی ست که الگوریتم ما هر چه می گذرد حریصانه تر می شود و در انتخاب حالات ممکن دقت بیشتری به خرج می دهد و سختگیری بیشتری دارد. استراتژی‌های معمول شامل :

- خنک‌کنندگی خطی: کاهش دما به مقدار ثابت .
- خنک‌کنندگی نمایی: ضرب دما در یک عامل ثابت $(0 < \alpha < 1)$ بعد از هر تکرار .
- خنک‌کنندگی لگاریتمی: کاهش دما بر اساس لگاریتم تعداد تکرارها .

هنگام به کارگیری SA برای زمان‌بندی، اصلاحات خاصی لازم است :

1.کدگذاری راه حل: راه حل‌ها می‌توانند به عنوان آرایه‌هایی نمایش داده شوند که هر عنصر آن مربوط به یک درس تخصیص‌یافته به یک اتاق و زمان خاص است .

2.جستجوی همسایگی: تولید راه حل‌های همسایه با تعویض دروس یا تخصیص مجدد آن‌ها به زمان‌های خالی یا اتاق‌های مختلف .

3.تابع هزینه: تعریف تابع هزینه که نقض محدودیت‌های سخت و نرم را جریمه می‌کند. هدف کاهش این هزینه است .

پیاده‌سازی الگوریتم :

1. تولید راه حل اولیه: استفاده از یک روش حریص یا تصادفی برای ایجاد جدول زمان‌بندی اولیه .
2. بهبود تکراری: اعمال الگوریتم SA و اصلاح تدریجی راه حل .
3. نظارت بر محدودیت‌ها: به طور مداوم نقض محدودیت‌ها را پیگیری کرده و تابع هزینه را بر این اساس به‌روزرسانی کنید .

مزایای شبیه‌سازی تنش:

- انعطاف‌پذیری SA :می‌تواند با محدودیت‌ها و الزامات مختلف سازگار شود .
- قابلیت گسترش: الگوریتم به طور کارآمد داده‌های بزرگتر را که معمولاً در محیط‌های دانشگاهی وجود دارد، مدیریت می‌کند .
- سادگی پیاده‌سازی: پیاده‌سازی SA از نظر مفهومی نسبت به دیگر تکنیک‌های بهینه‌سازی ساده‌تر است .

چالش‌ها:

- تنظیم پارامترها: انتخاب برنامه‌کاری خنک‌کننده و دمای اولیه می‌تواند به طور قابل‌توجهی بر عملکرد تأثیر بگذارد .
- زمان همگرایی SA :ممکن است نیاز به تنظیم دقیق داشته باشد تا توازنی بین اکتشاف و استفاده بهینه برقرار کند .

شبیه‌سازی تنش یک چارچوب قوی، قابل‌انعطاف و کارآمد برای مسائل زمان‌بندی دروس دانشگاهی فراهم می‌کند. توانایی پیمایش در فضاهای حل پیچیده با ریسک پذیرش موقتی راه‌حل‌های بدتر، امکان کشف زمان‌بندی‌های بهینه یا نزدیک به بهینه را فراهم می‌کند. در حالی که چالش‌ها وجود دارد، به‌ویژه در مورد تنظیم پارامترها و ارزیابی کیفیت راه‌حل، مزایای نشان داده شده در کاربردهای عملی نشان می‌دهد که SA یک استراتژی امیدوارکننده برای این حوزه است.

Pseudocode for University Time Tabling using Simulated Annealing

```
C = Current = C initial
for t = t max to t min
Eç = E(c)
En = E(n)
ΔE = En - Eç
if (ΔE>0)
    C = N
else if ( e^((ΔE/T) )>rand(0,1))
   C = N
```

Epoch

جستجو تابو(Tabu search):

جستجوی تابو، یک الگوریتم پیشرفته بهینه‌سازی متاهیورستیک است که به خاطر توانایی‌اش در پیمایش فضاهای بزرگ ،حل و فرار از بهینه‌های محلی معروف است .[9]

مسئله برنامه‌ریزی دروس دانشگاه را می‌توان به‌طور رسمی به شکل زیر تعریف کرد:

- ورودی‌ها:
  - مجموعه‌ای از دروس، که هر کدام به یک گروه از دانشجویان و یک عضو هیئت‌علمی مرتبط است.
  - فهرستی از زمان‌های در دسترس و کلاس‌ها.
- خروجی‌ها:
  - تخصیص دروس به زمان‌های مشخص و اتاق‌ها.

جستجوی تابو از یک روش جستجوی محلی به همراه یک ساختار حافظه به نام "فهرست تابو" استفاده می‌کند تا از بازگشت به راه‌حل‌های قبلاً دیده شده جلوگیری کند. مراحل اصلی رویکرد جستجوی تابو برای برنامه‌ریزی دروس به شرح زیر است:

1. مرحله آغازین: تولید یک راه‌حل اولیه قابل قبول به‌صورت تصادفی یا از طریق روش‌های ابتکاری.
2. اکتشاف همسایه: تعریف یک ساختار همسایه که اجازه می‌دهد حرکات محلی انجام شود، مانند تعویض زمان‌های کلاس بین دروس یا تغییر تخصیص کلاس‌ها.
3. ارزیابی حرکت: ارزیابی تأثیر هر حرکت ممکن بر یک تابع هدف که کیفیت راه‌حل را کمی‌سازی می‌کند (مثلاً، تعداد تداخل‌ها، کل نارضایتی).
4. لیست تابو: یک لیست تابو پیاده‌سازی کنید تا حرکات اخیر را به خاطر بسپارد. یک حرکت به مدت تعیین‌شده‌ای «تابو» محسوب می‌شود.
5. انتخاب بهترین حرکت: از حرکات کاندید، بهترین حرکت غیر تابو یا بهترین حرکت تابو را انتخاب کنید اگر هیچ حرکت غیر تابویی بهبود در راه‌حل ایجاد نکند.
6. معیارهای توقف: معیارهای توقفی مثل حداکثر تعداد تکرار یا محدودیت زمانی تعریف کنید. [10]

پیاده‌سازی نیاز به تصمیم‌گیری در مورد اجزای مختلف دارد:

• تابع هدف: یک تابع هدف امیدوارکننده شامل جریمه‌هایی برای تضادهای زمان‌بندی (دسترس‌پذیری مدرسان، تداخل دانش‌آموزان) و نقض محدودیت‌ها (ظرفیت اتاق‌ها) است.

• مدیریت لیست تابو: اندازه و ساختار لیست تابو نیاز به تنظیم دارد، تا کارایی حافظه و امکان جستجو را متعادل کند.

• اندازه همسایگی: تنظیم اندازه همسایگی می‌تواند تأثیر محسوسی بر عملکرد داشته باشد؛ همسایگی بزرگ‌تر ممکن است به کاوش بهتر منتهی شود اما زمان محاسباتی را افزایش دهد. [9]

در اینجا به شبه کدی از این جستجو میپردازیم:

```
Initialize currentSolution with a feasible timetable
Initialize bestSolution with currentSolution
Initialize tabuList as empty
Set tabuTenure to a fixed number

WHILE stoppingCriteriaNotMet
   Generate a list of candidateSolutions by modifying currentSolution
   FOR each candidate in candidateSolutions
      IF candidate is not in tabuList AND candidate is better than currentSolution
         Update currentSolution to candidate
         IF candidate is better than bestSolution
            Update bestSolution to candidate
         END IF
      END IF
   END FOR

   Add currentSolution to tabuList
   IF size of tabuList exceeds tabuTenure
      Remove oldest entry from tabuList
   END IF
END WHILE
Return bestSolution as the optimal timetable
```

در اینجا مراحل حل مسئله‌ی زمان‌بندی دانشگاه با استفاده از جستجوی تابو به زبان فارسی توضیح داده شده است:

شروع با راه‌حل فعلی و بهترین راه‌حل: با یک جدول زمانی قابل قبول شروع کنید و این را به عنوان راه‌حل فعلی و بهترین راه‌حل اولیه در نظر بگیرید.

ایجاد لیست تابو: یک لیست تابو خالی ایجاد کنید که تغییرات اخیر را ذخیره می‌کند تا از بازگشت به همان راه‌حل‌ها جلوگیری شود.

تولید راه‌حل‌های کاندید: با تغییرات کوچک در راه‌حل فعلی، لیستی از راه‌حل‌های کاندید ایجاد کنید.

ارزیابی و به‌روزرسانی: هر کاندید را بررسی کنید. اگر راه‌حل کاندید در لیست تابو نیست و از راه‌حل فعلی بهتر است:

راه‌حل فعلی را با کاندید به‌روز کنید.

اگر کاندید بهتر از بهترین راه‌حل است، بهترین راه‌حل را نیز به‌روز کنید.

مدیریت لیست تابو: راه‌حل فعلی را به لیست تابو اضافه کنید. اگر اندازه لیست تابو از حد مجاز (تابو تنور) بیشتر شد، قدیمی‌ترین ورودی را حذف کنید.

تکرار مراحل: تا زمانی که معیار توقف (مثل تعداد معین تکرارها یا عدم بهبود) برآورده شود، این مراحل را تکرار کنید.

این روش به شما کمک می‌کند تا با ایجاد تغییرات کوچک و جلوگیری از بازگشت به راه‌حل‌های قبلی، به یک جدول زمانی بهینه دست یابید.

الگوریتم رضایت محدودیت:

زمان‌بندی یک فعالیت اساسی در مؤسسات آموزشی است که شامل برنامه‌ریزی دروس با رعایت محدودیت‌های مختلف می‌شود. استفاده از مسائل رضایت محدودیت (CSP) یک رویکرد ساختاریافته برای مقابله با این مشکل پیچیده بهینه‌سازی فراهم می‌کند. در این قسمت به بررسی فرموله کردن زمان‌بندی دروس دانشگاهی به عنوان یک CSP می‌پردازد، محدودیت‌های مختلف موجود را توضیح می‌دهیم، تکنیک‌های رایج برای حل این مشکلات را مورد بحث قرار می‌دهد و پیاده‌سازی راه‌حل‌های CSP را در زمینه زمان‌بندی دانشگاهی نشان می‌دهد .

زمان‌بندی یک عملکرد حیاتی در دانشگاه‌ها است که بر چندین ذینفع از جمله دانشجویان، اساتید و مدیران تأثیر می‌گذارد. هدف این است که برنامه‌هایی برای دروس ایجاد شود که حداکثر استفاده از منابع را به همراه داشته باشد و در عین حال به مجموعه‌ای از محدودیت‌ها پاسخ دهد. پیچیدگی این مشکل با افزایش تعداد دروس، منابع و نیازهای متنوع شرکت‌کنندگان افزایش می‌یابد.

شناسایی زمان‌بندی به عنوان یک مسئله رضایت محدودیت (CSP) امکان رویکردی سیستماتیک برای فرموله کردن و حل آن را فراهم می‌کند[14] .

درک مسائل رضایت محدودیت (CSP) :

یک CSP شامل مجموعه‌ای از متغیرها است که هر کدام می‌توانند مقادیری از یک دامنه خاص بگیرند و مجموعه‌ای از محدودیت‌ها که مشخص می‌کنند کدام ترکیب‌های مقادیر قابل قبول هستند. به‌طور رسمی، یک CSP می‌تواند به‌صورت زیر تعریف شود :

- متغیرها $(X = \{X_1, X_2, ..., X_n\})$
- دامنه‌ها (D_i) :دامنه متغیر (X_i) است یعنی، مقادیر ممکن که (X_i) می‌تواند به خود بگیرد
- محدودیت‌ها $(C = \{C_1, C_2, ..., C_m\})$:نمایانگر محدودیت‌ها یا نیازهایی هستند که باید برآورده شوند .

در زمینه زمان‌بندی دروس دانشگاهی، این عناصر به‌صورت زیر شناسایی می‌شوند :

متغیرها :

·دوره‌ها: هر دوره نمایانگر یک متغیر است. به عنوان مثال، اگر یک دانشگاه سه دوره ارائه دهد، متغیرها می‌توانند(C_1) ، (C_2) و (C_3) باشند .

دامنه‌ها :

·زمان‌ها و مکان‌ها: هر دوره می‌تواند به زمان‌های مختلف و کلاس‌های خاص تخصیص یابد. دامنه‌های هر متغیر شامل همه زمان‌ها و مکان‌های مجاز است .

محدودیت‌ها :

محدودیت‌ها در زمان‌بندی دانشگاهی را می‌توان به چند دسته تقسیم کرد :

1 .محدودیت‌های سخت: این‌ها باید به طور دقیق رعایت شوند و نمی‌توانند نقض شوند .

oهیچ دو دوره‌ای نباید در همان زمان برای همان گروه از دانش‌آموزان قرار بگیرند .

oیک اتاق نمی‌تواند بیشتر از ظرفیت خود دانش‌آموزان را پذیرش کند .

2 .محدودیت‌های نرم: این‌ها مطلوب هستند اما در صورت لزوم می‌توانند تا حدی نقض شوند .

- ترجیح به برخی زمان‌ها که توسط دانش‌آموزان یا اساتید درخواست شده است .
- کاهش فاصله‌های زمانی بین کلاس‌های متوالی برای دانش‌آموزان .

مسئله زمانبندی را فرمول‌بندی کردن:

برای فرمول‌بندی مؤثر زمانبندی دوره‌های دانشگاهی به‌عنوان یک مسئله محدودیت‌دار(CSP) ، ابتدا باید دوره‌های خاص، زمان‌های در دسترس و کلاس‌های درس مشخص شوند. مرحله بعدی این است که تعاریف واضحی برای دامنه‌ها و محدودیت‌های مربوطه تعیین کنیم[14].

هدف این است که دوره‌ها را به زمان‌های مشخص و اتاق‌ها تخصیص دهیم و در عین حال تمام محدودیت‌های سخت را برآورده کرده و حداکثر رضایت از محدودیت‌های نرم را جلب کنیم.

تکنیک‌های حلCSP ها:

چندین تکنیک وجود دارد که می‌توانند برای حلCSP ها استفاده شوند، از جمله:

1. الگوریتم‌های برگشت به عقب(backtracking): این روش آزمایش و خطا به‌دنبال یک راه‌حل به‌طور تدریجی می‌گردد و وقتی به یک تخصیص نامعتبر می‌رسد، به عقب برمی‌گردد.

2. انتشار محدودیت: تکنیک‌هایی مانند سازگاری قوس می‌توانند به کاهش دامنه متغیرها کمک کنند و مقادیری را که منجر به نقض می‌شود، حذف کنند.

3. جستجوی محلی: روش‌های دوراندیش مانند الگوریتم‌های ژنتیک، آنیلینگ شبیه‌سازی شده یا جستجوی تابو می‌توانند فضاهای حل را به‌طور کارآمد کاوش کنند، به‌ویژه برای نمونه‌های بزرگ.

4. برنامه‌نویسی عدد صحیح: فرمول‌بندی‌های برنامه‌نویسی ریاضی می‌توانند برای یافتن راه‌حل‌های بهینه با تعریف تابع هدف و محدودیت‌ها به‌صورت معادلات خطی استفاده شوند.

چندین دانشگاه به راهکارهای مبتنی بر CSP برای حل مشکلات برنامه‌ریزی خود روی آورده‌اند که منجر به بهبود قابل توجهی در کارایی برنامه‌ریزی شده است. به عنوان مثال، محققان دانشگاه ملبورن یک چارچوب CSP پیاده‌سازی کردند که از استراتژی‌های ترکیبی استفاده می‌کند و تکنیک‌های برگشت به عقب و جستجوی محلی را ترکیب می‌کند و این امر باعث کاهش 30 درصدی تضادهای برنامه‌ریزی شده است. [15]

فرموله‌سازی برنامه‌زمانبندی دروس دانشگاهی به عنوان یک مشکل رضایت محدودیت، مسیر روشنی برای حل کارآمد یک چالش پیچیده برنامه‌ریزی ارائه می‌دهد. با درک ساختارCSP ها و استفاده از تکنیک‌های مناسب، مؤسسات آموزشی می‌توانند برنامه‌های بهینه‌شده‌ای توسعه دهند که نیازهای ذینفعان مختلف را در نظر بگیرد. کارهای آینده باید بر ادغام داده‌های زمان واقعی و الگوریتم‌های پیشرفته یادگیری ماشین تمرکز کنند تا استحکام و سازگاری راهکارهای برنامه‌ریزی را بیشتر ارتقا دهند.

الگوریتم کولنی مورچه ها (ACO):

به بررسی کاربرد بهینه‌سازی کلونی مورچه‌ها(ACO) ، یک الگوریتم الهام‌گرفته از طبیعت بر اساس رفتار جستجوی غذا در مورچه‌ها، می‌پردازیم تا به مسأله زمان‌بندی دوره‌های دانشگاهی پاسخ دهد. روش‌های بهینه‌سازی سنتی معمولاً با پیچیدگی و مقیاس مسأله دست و پنجه نرم می‌کنند و نیاز به بررسی رویکردهای هنجاری و فرامکانی را ضروری می‌سازند. در میان این روش‌ها، بهینه‌سازی کلونی مورچه‌ها (ACO) به عنوان یک تکنیک قدرتمند برای حل چنین مسأله‌های ترکیبی ظهور کرده است[13] .

زمان‌بندی به عنوان یک مسأله تخصیص منابع تعریف می‌شود که اهداف آن اغلب شامل حداقل کردن تضادها، حداکثر کردن استفاده از منابع و برآورده کردن ترجیحات مختلف ذینفعان است. این مسأله می‌تواند به طور رسمی با یک تَرتیب توصیف شود :

- (C):مجموعه دوره‌ها
- (T):مجموعه زمان‌های خالی
- (R):مجموعه اتاق‌ها
- (F):مجموعه اعضای هیئت علمی
- محدودیت‌ها: سخت (باید رعایت شود) و نرم (ترجیحات)

بهینه‌سازی کلونی مورچه‌ها که در دهه 1990 توسط مارکو دوریگو توسعه یافته، شیوه‌ای را تقلید می‌کند که مورچه‌ها غذا پیدا می‌کنند و از فرومون‌ها به عنوان یک روش ارتباطی استفاده می‌کنند. یک کلونی از مورچه‌های مصنوعی با پیمایش فضای راه‌حل، راه‌حل‌ها را جستجو می‌کند و فرومون‌هایی را deposit می‌کند که احتمال انتخاب مسیرهای بعدی را تحت تأثیر قرار می‌دهد. این فرآیند تکراری برای حل مسائل مختلف بهینه‌سازی، از جمله مسیریابی، زمان‌بندی و برنامه‌ریزی مؤثر بوده است[13] .

الگوریتم ACO شامل مراحل کلیدی زیر است :

1. شروع: تعریف پارامترهایی مانند تعداد مورچه‌ها، سطح فرومون‌ها و اطلاعات کلیدی بر اساس دانش حوزه .

2. ساخت راه‌حل: هر مورچه با انتخاب تصادفی دوره‌ها، زمان‌ها و اتاق‌ها بر اساس مسیرهای فرومونی و اطلاعات کلیدی، یک راه‌حل می‌سازد .

3. به‌روزرسانی فرومون: به‌روزرسانی سطح فرومون‌ها بر اساس کیفیت راه‌حل‌ها، با تشویق به جست‌وجوی مسیرهای امیدوارکننده و اجازه تبخیر فرومون‌های قدیمی .

4. فرآیند تکراری: ساخت راه‌حل و به‌روزرسانی فرومون را تا زمان رسیدن به یک معیار توقف تکرار کنید (برای مثال، حداکثر تعداد تکرار یا همگرایی راه‌حل‌ها) .

برای سازگار کردن ACO با زمان‌بندی دوره‌های دانشگاهی، چند تغییر لازم است :

- نمایش راه‌حل: راه‌حل‌ها می‌توانند به‌صورت یک ماتریس نشان داده شوند که در آن سطرها زمان‌ها و ستون‌ها کلاس‌ها را نشان می‌دهد. هر خانه نشان‌دهنده دوره‌ای است که برای آن ترکیب زمان و اتاق اختصاص داده شده است

.

- اطلاعات کلیدی: شامل اطلاعاتی مانند تعداد دانشجویان ثبت‌نام‌شده در دوره‌ها و ترجیحات اساتید برای راهنمایی جست‌وجوی مورچه‌ها .

- مدیریت محدودیت‌ها: محدودیت‌های سخت را به‌طور مستقیم در فرآیند ساخت راه‌حل وارد کرده و محدودیت‌های نرم را در حین به‌روزرسانی فرومون ارزیابی کنید.

نتایج نشان داده است که ACO به‌طور مداوم جدول‌های زمانی با تضادهای کمتر نسبت به روش‌های سنتی مانند الگوریتم‌های ژنتیکی و آنیل‌سازی شبیه‌سازی شده تولید کرده است. علاوه بر این، کارایی محاسباتی ACO با افزایش تکرارها بهبود میابد.

بهینه‌سازی کلونی مورچه‌ها به‌عنوان روشی قوی و انعطاف‌پذیر برای حل مشکلات زمان‌بندی دروس دانشگاهی باثبات است. طبیعت سازگار آن به آن اجازه می‌دهد تا با توجه به محدودیت‌ها و الزامات منحصر به فرد هر دانشگاه شخصی‌سازی شود.

کارهای آینده بر بهینه‌سازی بیشتر الگوریتم از طریق پردازش موازی و ترکیب آن با سایر روش‌های متاهیوریستیک تمرکز خواهد کرد تا کیفیت راه‌حل و کارایی محاسباتی را افزایش دهد.

شبه کدی از این الگوریتم برای حل مسئله UCTP :

```
Initialize pheromoneMatrix with initial pheromone values
Initialize bestSolution to an empty solution
Set parameters alpha, beta, evaporationRate, and numberOfAnts

REPEAT until stoppingCriteriaMet
    FOR each ant in numberOfAnts
        Initialize ant's solution as empty
        WHILE ant's solution is incomplete
            Choose next component based on pheromoneMatrix and heuristic information
            Add component to ant's solution
        END WHILE
        Evaluate ant's solution
        Update pheromoneMatrix with ant's solution if it's better than bestSolution
        IF ant's solution is better than bestSolution
            Update bestSolution to ant's solution
        END IF
    END FOR
    Evaporate pheromones in pheromoneMatrix
    Apply pheromone updates based on the ants' solutions
END REPEAT
Return bestSolution as the optimal timetable
```

الگوریتم های ژنتیک ترکیبی:

الگوریتم‌های ژنتیکی متعارف نتایج خوبی در میان تعدادی از رویکردهای توسعه یافته برای UCTP ارائه نمی‌دهند. بنابراین، الگوریتم‌های متعارف نیاز به بهبود دارند تا مسئله UCTP را حل کنند. الگوریتم‌های مبتنی بر جمعیت، به‌ویژه الگوریتم‌های ژنتیکی، در سال‌های اخیر رایج‌ترین راه‌حل برای UCTP بوده‌اند. [1]

بنابراین، در این قسمت سه الگوریتم مبتنی بر الگوریتم ژنتیکی ارائه شده است FGARI ، FGATS و FGASA .

در الگوریتم‌های پیشنهادی، منطق فازی برای اندازه‌گیری نقض محدودیت‌های نرم در تابع تناسب استفاده می‌شود تا با عدم قطعیت و ابهام ذاتی در داده‌های واقعی مقابله شود.[5]

همچنین، جستجوی محلی تکراری تصادفی، آنیلینگ شبیه‌سازی شده و جستجوی تابو به ترتیب برای بهبود قابلیت جستجوی اکتشافی و جلوگیری از به دام افتادن الگوریتم ژنتیکی در بهینه محلی اعمال می‌شود. چندین الگوریتم برای حل مشکلات زمان‌بندی پیشنهاد شده است. [6]

به‌طور کلی، دو نوع الگوریتم متا-هیوستیک وجود دارد. نوع اول الگوریتم‌های مبتنی بر جستجوی منطقه‌ای محلی و نوع دوم الگوریتم‌های مبتنی بر جمعیت هستند. هر نوع مزایا و معایب خاص خود را دارد.

الگوریتم‌های مبتنی بر منطقه محلی شاملSA ، جستجوی همسایگی بسیار بزرگ، TSو بسیاری دیگر هستند. معمولاً، الگوریتم‌های مبتنی بر منطقه محلی بیشتر بر روی بهره‌برداری تمرکز دارند تا اکتشاف، به این معنی که آن‌ها در یک جهت حرکت می‌کنند بدون اینکه یک اسکن وسیع‌تری از فضای جستجو انجام دهند.

الگوریتم‌های مبتنی بر جمعیت با تعدادی راه‌حل شروع می‌کنند و آن‌ها را تصفیه می‌کنند تا به راه‌حل‌های بهینه جهانی در کل فضای جستجو دست یابند. الگوریتم‌های مبتنی بر جمعیت که به‌طور معمول برای حل مشکلات زمان‌بندی استفاده می‌شوند، الگوریتم‌های تکاملی(EAs) ، بهینه‌سازی ازدحام ذرات ، بهینه‌سازی کلونی مورچه‌ها ، سیستم ایمنی مصنوعی و غیره هستند.

در سال‌های اخیر، چندین محقق ازGA ها برای UCTP استفاده کرده‌اند. آن‌ها کاراییGA ها را با استفاده از اپراتورهای ژنتیکی و تکنیک‌های اصلاح‌شده LS افزایش دادند. به‌طور کلی، زمانی که یک GA ساده اعمال می‌شود، ممکن است زمان‌بندی‌های غیرقانونی تولید کند که شامل تکرار و/یا رویدادهای گم‌شده است. کیفیت راه‌حل‌های تولیدشده توسط الگوریتم‌های مبتنی بر جمعیت ممکن است بهتر از الگوریتم‌های مبتنی بر منطقه محلی نباشد، عمدتاً به این دلیل که الگوریتم‌های مبتنی بر جمعیت بیشتر نگران اکتشاف هستند تا بهره‌برداری.

الگوریتم‌های مبتنی بر جمعیت راه‌حل‌ها را در کل فضای جستجو اسکن می‌کنند بدون اینکه بر روی افراد با تناسب خوب در یک جمعیت تمرکز کنند. علاوه بر این، الگوریتم‌های مبتنی بر جمعیت ممکن است با همگرایی زودرس مواجه شوند که ممکن است منجر به گرفتار شدن آن‌ها در بهینه‌های محلی شود. دیگر عیب این الگوریتم‌ها نیاز به زمان بیشتر است.

با این حال، الگوریتم‌های ژنتیکی مزایای متعددی در مقایسه با سایر تکنیک‌های بهینه‌سازی دارند. به‌عنوان مثال، الگوریتم‌های ژنتیکی می‌توانند یک جستجوی چندجانبه با استفاده از مجموعه‌ای از راه‌حل‌های کاندید انجام دهند.

ترکیب‌های مختلفی از الگوریتم‌های محلی و جهانی برای حل مسائل در ادبیات مربوط به زمانبندی گزارش شده است. علاوه بر این، به‌طور فزاینده‌ای به‌این نکته توجه می شود که الگوریتم‌های تکاملی بدون گنجاندن دانش ویژه مساله، به خوبی الگوریتم‌های مبتنی بر برنامه‌نویسی ریاضی در برخی کلاس‌های مسائل زمانبندی عمل نمی‌کنند.

در این قسمت، ما می‌خواهیم ویژگی‌های خوب الگوریتم‌های مبتنی بر مناطق محلی و جهانی را برای حل مساله زمانبندی کلاس‌ها ترکیب کنیم.

ما تلاش می‌کنیم تعادلی بین قابلیت اکتشاف (بهبود جهانی) الگوریتم‌های ژنتیکی و قابلیت بهره‌برداری (بهبود محلی) روش‌های محلی ایجاد کنیم. علاوه بر این، یک ساختار داده حافظه خارجی معرفی می‌شود تا بخش‌هایی از راه‌حل‌های خوب قبلی را ذخیره کند و این بخش‌های ذخیره‌شده را به نسل‌های جدید دوباره معرفی کند تا بتواند الگوریتم‌های پیشنهادی را سریع‌تر به نقطه بهینه مساله زمانبندی کلاس‌ها هدایت کند.

الگوریتم‌های پیشنهادی :

در این تحقیق سه الگوریتم ژنتیکی ترکیبی به نام‌هایFGARI ، FGASAو FGATS پیشنهاد شده‌اند که ترکیبی از الگوریتم ژنتیکی، منطق فازی و الگوریتم‌های جستجوی محلی هستند. تفاوت این الگوریتم‌ها در الگوریتم جستجوی محلی آن‌هاست. در واقع، این الگوریتم‌ها الگوریتم‌های ژنتیکی اصلاح‌شده‌ای هستند که کد شبه‌عمومی آن‌ها در شکل ۱ نشان داده شده است .

```
Initialize population
Calculate fitness of all solutions
Sort population by fitness
While termination condition not reached do
   Select two parents from population by tournament selection with size 2
   Create child solution using crossover with a probability P_c
   Apply mutation with a probability P_m to child solution
   Apply Local Search to child solution
   Replace child solution with the worst member of the population
   Sort population by fitness
End while
The best solution achieved as output
```

ما از مدل الگوریتم ژنتیکی حالت پایدار استفاده می‌کنیم که بالاتر ذکر شده است، جایی که تنها یک راه‌حل فرزند با انتخاب، تقاطع و جهش در هر نسل تولید می‌شود. سپس فرزند با جستجوی محلی بهبود می‌یابد. در نهایت، بدترین عضو جمعیت با فرد جدید فرزند جایگزین می‌شود.

در این الگوریتم، یک کروموزوم به‌عنوان یک ماتریس 3 * NC نمایش داده می‌شود، که در آن NC تعداد دوره‌هاست. شاخص ستون‌ها شماره شناسایی یک دوره و محتوای ردیف‌های ماتریس شماره شناسایی استاد، زمان شروع و اتاق به ترتیب را نشان می‌دهد. شکل ۲ ساختار یک کروموزوم را نشان می‌دهد.

	C_1	C_2	C_3	...	C_{Nc}
Professor ID	1	3			1
Start Timeslot ID	23	12			30
Room ID	10	2			10

جمعیت اولیه به گونه‌ای تولید می‌شود که ویژگی‌های تصادفی راه‌حل‌ها حفظ شود و همه محدودیت‌های سخت نیز رعایت گردد .

برای این منظور، با استفاده از ورودی‌هایUCTP ، یک ماتریس استاد-درس، یک ماتریس درگیری-درس و یک ماتریس اتاق-درس، یک ماتریس استاد-زمان‌بندی و یک ماتریس اتاق-زمان‌بندی تولید می‌شود.

یک ماتریس استاد-درس یک ماتریس NP × NC است که هر عنصر در ماتریس با "۰"، "۱" یا "۲" نمایش داده می‌شود. مقدار "۰" نشان می‌دهد که استاد نمی‌تواند درس را تدریس کند. مقدار "۱" نشان می‌دهد که استاد می‌تواند درس را تدریس کند اما در آن درس تخصص ندارد. مقدار "۲" نشان می‌دهد که استاد در آن درس متخصص است .و NC به ترتیب نشان‌دهنده تعداد اساتید و تعداد دروس هستند .

یک ماتریس درگیری-درس یک ماتریس NC × NC است که هر عنصر در ماتریس با "۰" یا "۱" نمایش داده می‌شود. مقدار "۰" نشان می‌دهد که دروس هیچ درگیری ندارند. مقدار "۱" نشان می‌دهد که دروس درگیری دارند .

یک ماتریس اتاق-درس یک ماتریس NR × NC است که هر عنصر در ماتریس با "۰" یا "۱" نمایش داده می‌شود. مقدار "۰" نشان می‌دهد که اتاق برای درس مناسب نیست. مقدار "۱" نشان می‌دهد که اتاق برای درس مناسب است .نشان‌دهنده تعداد اتاق‌ها است .

یک ماتریس استاد-زمان‌بندی یک ماتریس NP × 45 است که هر عنصر در ماتریس با "۰"، "۱" یا "۲" نمایش داده می‌شود. مقدار "۰" نشان می‌دهد که استاد در دانشگاه حاضر نیست.

ارزش "2" نشان می‌دهد که استاد در دانشگاه حضور دارد و تمایل به تدریس در آن زمان دارد. یک ماتریس اتاق-زمان، یک ماتریس NR × 45 است که هر عنصر در این ماتریس با "0" یا "1" نمایش داده می‌شود. ارزش "0" نشان می‌دهد که اتاق در زمان مشخص خالی است. ارزش "1" نشان می‌دهد که اتاق در آن زمان خالی نیست.

پس از تعریف این ماتریس‌ها، دوره‌ها بر اساس تعداد استادانی که می‌توانند آن‌ها را تدریس کنند، به ترتیب صعودی مرتب می‌شوند. سپس برای هر دوره به طور تصادفی یک استاد، زمان و اتاق انتخاب می‌شود، به گونه‌ای که تمام محدودیت‌های سخت رعایت شود و به شرح زیر است:

1 .بر اساس ماتریس استاد-دوره، یک استاد انتخاب کنید.

2 .یک زمان مناسب از بین زمان‌های استاد انتخاب شده در مرحله 1 با استفاده از ماتریس استاد-زمان انتخاب کنید.

3 .یک اتاق مناسب بر اساس ماتریس اتاق-زمان و ماتریس اتاق-دوره انتخاب کنید .

تابع تناسب (fitness function):

تناسب یک راه‌حل به رعایت محدودیت‌های سخت و نرم بستگی دارد. در این الگوریتم‌ها، جمعیت اولیه، عملگرهای ژنتیکی و الگوریتم‌های جستجوی محلی به گونه‌ای تعریف شده‌اند که تمام محدودیت‌های سخت تمام راه‌حل‌ها رعایت شود. بنابراین، تابع تناسب تنها به رعایت محدودیت‌های نرم بستگی دارد. به این ترتیب، تابع تناسب تنها به محدودیت‌های نرم پرداخته می‌شود. از سوی دیگر، محدودیت‌های نرم به نوعی کیفی هستند و اندازه‌گیری دقیق آن‌ها دشوار و مبهم است و بین آن‌ها روابط "اگر-آنگاه" وجود دارد که می‌تواند به راحتی قوانین فازی را توصیف کند[11].

سپس، ما از منطق فازی برای اندازه‌گیری محدودیت‌های نرم استفاده می‌کنیم و تابع عضویت مناسبی برای هر محدودیت نرم تعریف می‌کنیم. تابع تناسب به صورت زیر تعریف می‌شود:

$$Fitnessfunction(I) = \sum_{i=1}^{7} w_i \mu_{\widetilde{Soft_i}}$$

جایی که μS˜oftI نشان‌دهنده مقدار تابع عضویت محدودیت نرم i است که در بازه [0,1] قرار دارد و wi نشان‌دهنده وزن محدودیت نرم i است که در این مقاله برای تمام محدودیت‌های نرم 100 فرض شده است. بنابراین بدترین مقدار برای تناسب یک راه‌حل 700 است.

انتخاب (Selection):

در الگوریتم‌های پیشنهادی، از انتخاب تورنمنت استفاده می‌شود. در این روش، 2 راه‌حل به‌طور تصادفی با چرخ رولت انتخاب می‌شوند. سپس بهترین راه‌حل بین آن‌ها انتخاب می‌شود. فرآیند انتخاب در هر نسل دو بار برای انتخاب دو والد برای تولید مثل انجام می‌شود[12].

تقاطع (Crossover):

به‌طور کلی، نشان داده شده است که تقاطع یکنواخت برای بسیاری از مسائل به‌ویژه مسائل بهینه‌سازی عددی مؤثرتر است. در این مقاله از یک اپراتور تقاطع یکنواخت با احتمال PC استفاده می‌شود. در نهایت، اگر فرزند دارای نقض محدودیت‌های سخت باشد؛ ما از یک تابع تعمیر برای بهبود آن در صورت امکان استفاده می‌کنیم. در غیر این صورت، عملیات تقاطع تکرار می‌شود.

جهش (Mutation):

در این الگوریتم از جهش تصادفی با احتمال Pm استفاده می‌شود. این به‌طور تصادفی یک زمان مناسب را برای موضوع بر اساس ماتریس دوره-زمان انتخاب می‌کند. اگر دوره دارای نقض محدودیت‌های سخت باشد، ما از یک تابع تعمیر برای بهبود آن در صورت امکان استفاده می‌کنیم. در غیر این صورت، عملیات تقاطع تکرار می‌شود.

سه الگوریتم جستجوی محلی را ارائه کرده ایم که بر اساس آن‌ها سه الگوریتم ژنتیک هیبریدی پیشنهاد شده‌اند. هر سه الگوریتم جستجوی محلی بر اساس ساختارهای همسایگی زیر عمل می‌کنند:

N1: به‌طور تصادفی یک استاد انتخاب کنید و زمان دو دوره مرتبط با آن استاد را جابجا کنید به‌طوری‌که محدودیت‌های سخت نقض نشود.

N2: یک دوره واحد را به‌طور تصادفی انتخاب کنید و آن را به یک زمان قابل‌قبول تصادفی دیگر منتقل کنید.

N3: یک دوره را به صورت رندوم انتخاب کن و استاد درس را تغییر بده. اگر نیاز است بازه ی زمانی و اتاق را جابه جا کن.

N4: یک دوره را به صورت رندوم انتخاب کن ویک دوره دیگر که طول و موضوع یکسانی با آن دارد را انتخاب کن. بازه ی زمانی این دو کلاس را با یکدیگر عوض کن.[11]

جستجوی محلی تکراری تصادفی(randomized iterative search):

این الگوریتم در الگوریتم FGARI استفاده می‌شود. در هر تکرار این الگوریتم، فهرستی با K عنصر از ساختارهای همسایگی که در بالا ذکر شد، به‌طور تصادفی تولید می‌شود. تمام همسایگی‌ها بر روی راه‌حل اصلی اعمال می‌شوند و تناسب آن برای هر همسایگی

اندازه‌گیری می‌شود. بهترین راه‌حل با راه‌حل اصلی مقایسه می‌شود. اگر بهترین راه‌حل بهتر از راه‌حل اصلی بود، راه‌حل اصلی با بهترین راه‌حل جایگزین می‌شود. در غیر این صورت، راه‌حل اصلی با بهترین راه‌حل با احتمال بسیار کم جایگزین می‌شود تا از بهینه محلی جلوگیری شود. شبه کد این الگوریتم در شکل 3 نشان داده شده است.[4]

```
Calculate initial fitness for S, Fitness function(S)
Set best solution S_best ← S
While (not termination criterion)
    Create neighborhood structure list randomly, List_NS = {N_1, N_2, ..., N_k}
    For i=1: K, where K is the total number of neighborhood structures
        Apply neighborhood structure i to S, NewS_i
        Calculate fitness for NewS_i, Fitness function(NewS_i)
    End for;
    Identify the best solution among all the NewS_i wherei ∈ {1, ..., K}, NewS_Best
    If (Fitness function(NewS_Best) < Fitness function(S_best))
        S ← NewS_Best
        S_best ← NewS_Best
    Else
      δ = Fitness function(NewS_Best) − Fitness function(S)
      Generate a random number in [0,1], R
      If (R < e^−δ)
          S ← NewS_Best
      End if
    End if
End while
```

الگوریتم آنیلینگ شبیه‌سازی‌شده (simulated annealing search):

الگوریتم آنیلینگ شبیه‌سازی‌شده به‌عنوان الگوریتم جستجوی محلی در FGASA استفاده می‌شود. آنیلینگ شبیه‌سازی‌شده به پارامترهای خود بسیار حساس است و روش‌هایی که برای تعیین این پارامترها استفاده می‌شود، بسیار مهم است. برخی از مهم‌ترین این پارامترها دما اولیه، دمای نهایی و روش خنک‌سازی هستند.

در این مقاله، برای تعیین دمای اولیهT0 ، 100 راه‌حل جدید از طریق ساختارهای همسایگی تولید می‌شود و سپس حداکثر تفاوت بین تناسب دو راه‌حل متوالی به عنوان دمای اولیه در نظر گرفته می‌شود. دمای نهایی Tf برابر با 0.09 T0 فرض می‌شود. تابع خنک‌سازی طبق روشی که در پیشنهاد شده است به صورت زیر است: جایی که Tr دمای فعلی را نشان می‌دهد و β یک مقدار ثابت است که در این مقاله 0.1 فرض شده است.

$$T_{r+1} = \frac{T_r}{1 + \beta T_r}$$

این الگوریتم در هر دما یک بار تکرار می‌شود. کد شبه‌الگوریتم این الگوریتم در شکل 4 نشان داده شده است.

```
Calculate initial fitness for S, Fitness function(S)
Set best solution S_best ← S
Create a neighborhood structure list of 100 elements randomly, List_NS100
Create 100 solution using List_NS100, S_i, i = 1...100
CalculateFitness function(S_i), f_i, i = 1...100
T_0 = max(Δf_i)
T_f = 0.09T_0
T_r = T_0
While (not termination criterion)
    Create neighborhood structure list randomly, List_NS = {N_1, N_2, ..., N_k}
    For i=1: K, where K is the total number of neighborhood structures
        Apply neighborhood structure N_i to S, S_New
    End for;
    If (Fitness function(S_New) < Fitness function(S_best))
        S ← S_New
        S_best ← S_New
    Else
      δ = Fitness function(NewS_Best) − Fitness function(S)
      Generate a random number in [0, 1], R
      If (R < e^(−δ/T_r))
          S ← S_New
      End if
    End if
    T_r = T_r / (1 + βT_r)
End while
```

الگوریتم جستجوی تابو(Tabu search algorithm):

الگوریتم جستجوی تابو در FGATS استفاده می‌شود. لیست‌های همسایگی که به تازگی بازدید شده‌اند به لیست تابو (که دارای طول ثابت است) اضافه می‌شوند.

در این الگوریتم، یک لیست با K عنصر از ساختارهای همسایگی به صورت تصادفی تولید می‌شود. هر همسایگی در لیست به تعداد Lبار به راه‌حل اصلی اعمال می‌شود. سپس تناسب راه‌حل جدید اندازه‌گیری می‌شود. راه‌حل جدید با راه‌حل اصلی مقایسه می‌شود. اگر راه‌حل جدید بهتر از راه‌حل اصلی بود، راه‌حل اصلی با راه‌حل جدید جایگزین می‌شود.

در غیر این صورت، راه‌حل اصلی با راه‌حل جدید با احتمال بسیار کم جایگزین می‌شود تا از بهینه محلی جلوگیری شود. در نهایت، لیست همسایگی به لیست تابو اضافه می‌شود. [9] [10]

شبه کد این الگوریتم در شکل 5 نشان داده شده است.

```
Calculate initial fitness for S, Fitness function(S)
Set best solution S_best ← S
While (not termination criterion)
    Create neighborhood structure list randomly, List_NS = {N_1, N_2, ..., N_k}
    For i=1: K, where K is the total number of neighborhood structures
        Apply neighborhood structure N_i to S for L times, S_New
    End for;
    If (Fitness function(S_New) < Fitness function(S_best))
        S ← S_New
        S_best ← S_New
    Else
      δ = Fitness function(NewS_Best) − Fitness function(S)
      Generate a random number in [0, 1], R
      If (R < e^−δ)
          S ← S_New
      End if
    End if
     Remove the first item from tabu list if it is full
     Add List_NS to end of tabu list
End while
```

ارزیابی الگوریتم های موجود برای حل مسئله زمانبندی دانشگاهی:

رضایت محدودیت: (CSP)

CSPبا تعریف متغیرها، دامنه‌ها و محدودیت‌ها، به مسئله زمان‌بندی می‌پردازد. تکنیک‌ها شامل بازگشت، انتشار محدودیت و روش‌های انسجام قوس هستند .

مزایا :

- برای مشکلات کوچک تا متوسط موثر است .
- انعطاف‌پذیری در افزودن محدودیت‌ها .

معایب :

- عملکرد با افزایش پیچیدگی کاهش می‌یابد .
- راه‌حل‌ها همیشه بهینه نیستند .

الگوریتم‌های متاهیورستیک:

رویکردهای متاهیورستیک تعادلی بین کیفیت راه‌حل و کارایی محاسباتی ارائه می‌دهند و آنها را برای مشکلات بزرگ مقیاس زمان‌بندی مناسب‌تر می‌کند[5] .

- الگوریتم‌های ژنتیک: (GA)

الگوریتم‌های ژنتیک از اصول انتخاب طبیعی برای تکامل مجموعه‌ای از راه‌حل‌ها در طول نسل‌ها استفاده می‌کنند .

مزایا :

- برای مشکلات بزرگ و پیچیده موثر است .
- قادر به فرار از بهینه‌های محلی .

معایب :

- نیاز به تنظیم دقیق پارامترها دارد .
- تصادفی بودن ممکن است به نتایج ناسازگار منجر شود .

- الگوریتم تبرید شبیه‌سازی شده: (SA)

الگوریتم تبرید شبیه‌سازی شده روند خنک‌سازی فلزات را تقلید می‌کند تا از بهینه‌های محلی فرار کند به این طریق که به طور موقت راه‌حل‌های بدتر را مجاز می‌شمارد و باعث بررسی بیشتر فضای راه‌حل می‌شود .

مزایا :

- پیاده‌سازی آن ساده است .
- پتانسیل یافتن راه‌حل‌های نزدیک به بهینه .

معایب :

- همگرایی می‌تواند کند باشد .
- عملکرد به شدت به برنامه خنک‌سازی وابسته است .

- جستجوی تابو :

جستجوی تابو جستجوی محلی را با پیگیری راه‌حل‌های بازدید شده قبلی بهبود می‌بخشد تا از بازگشت به عقب جلوگیری کند .

مزایا :

- برای مشکلات بزرگ و پیچیده زمان‌بندی موثر است .
- می‌تواند راه‌حل‌های با کیفیت بالا تولید کند .

معایب:

- مدیریت حافظه می‌تواند چالش‌برانگیز باشد.
- ممکن است به منابع محاسباتی گسترده‌ای نیاز داشته باشد.

الگوریتم‌های ترکیبی:

الگوریتم‌های ترکیبی عناصری از تکنیک‌های مختلف را ترکیب می‌کنند تا عملکرد را بهبود بخشند.

- الگوریتم ژنتیک با جستجوی محلی(GALS)

این رویکرد، الگوریتم ژنتیک را با روش‌های جستجوی محلی ترکیب می‌کند تا راه‌حل‌ها را بهبود بخشد.

مزایا:

- کیفیت بهتر راه‌حل.
- حفظ قابلیت‌های اکتشافی الگوریتم ژنتیک در حالی که از بهبود جستجوی محلی بهره می‌برد.

معایب:

- پیچیدگی بیشتر در پیاده‌سازی.

- بهینه‌سازی کلونی مورچه‌ها:(ACO)

بر اساس رفتار مورچه‌ها در یافتن مسیر به غذا، ACOاز جمعیتی از مورچه‌های مصنوعی برای کاوش در فضای راه‌حل و به‌روزرسانی مسیرهای فرومون برای هدایت جستجو استفاده می‌کند.

مزایا:

- موثر برای محیط‌های پویا.
- تعادل قوی بین اکتشاف و بهره‌برداری.

معایب:

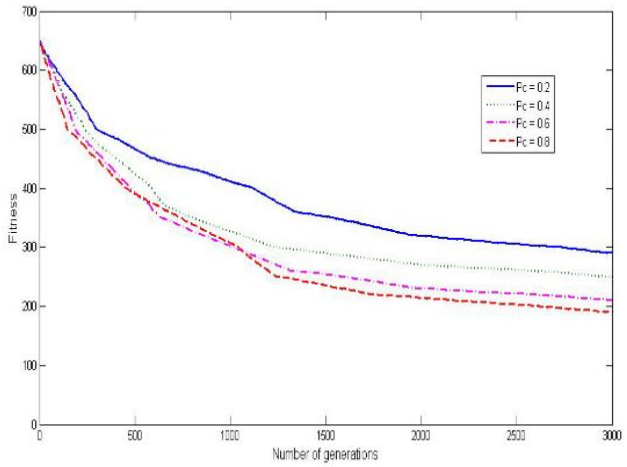
- هزینه محاسباتی بالا.
- حساس به تنظیمات پارامترها.

ارزیابی سه الگوریتم ترکیبی پیشنهادی ژنتیک:

ارزیابی تأثیر پارامتر احتمال تقاطع بر عملکرد GA پیشنهادی:

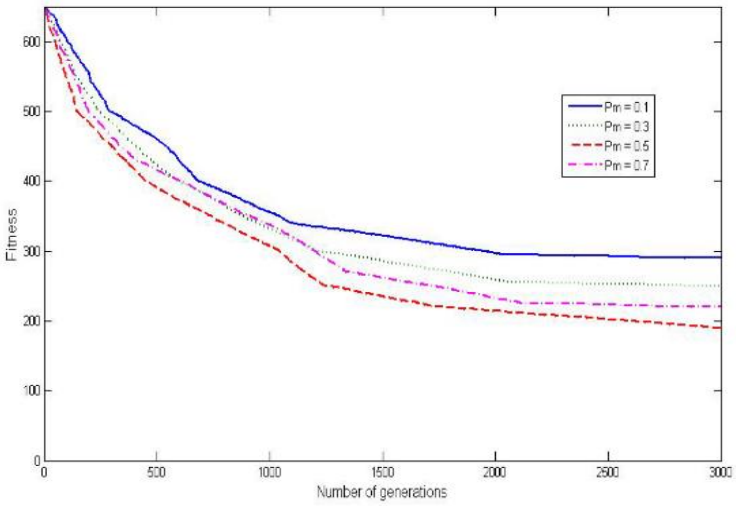
عملکرد الگوریتم ژنتیکی به شدت به پارامتر احتمال تقاطع (PC) حساس است. برای ارزیابی تأثیر این پارامتر، الگوریتم ژنتیکی پیشنهادی (بدون مرحله جستجوی محلی) با چهار مقدار مختلف (0.2، 0.4، 0.6 و 0.8) برای احتمال تقاطع اجرا می‌شود.

شکل زیر تأثیر تغییر PC بر GA را نشان می‌دهد. در شکل محور افقی تعداد نسل‌ها و محور عمودی تناسب بهترین راه‌حل را نشان می‌دهد. همان‌طور که در شکل مشاهده می‌شود، توانایی GA برای یافتن راه‌حل بهینه زمانی که مقدار PC از 0.2 به 0.8 افزایش می‌یابد، بهبود می‌یابد. این به این دلیل است که وقتی مقدار بالایی برای PC انتخاب می‌کنیم، احتمال تولید راه‌حل‌های جدید افزایش می‌یابد و جستجو وسیع‌تر می‌شود .



ارزیابی تأثیر پارامتر احتمال جهش بر GA پیشنهادی:

احتمال جهش (Pm) پارامتر مهم دیگری است که بر کارایی GA تأثیر می‌گذارد. شکل زیر رفتار GA را با مقادیر مختلف Pm نشان می‌دهد.



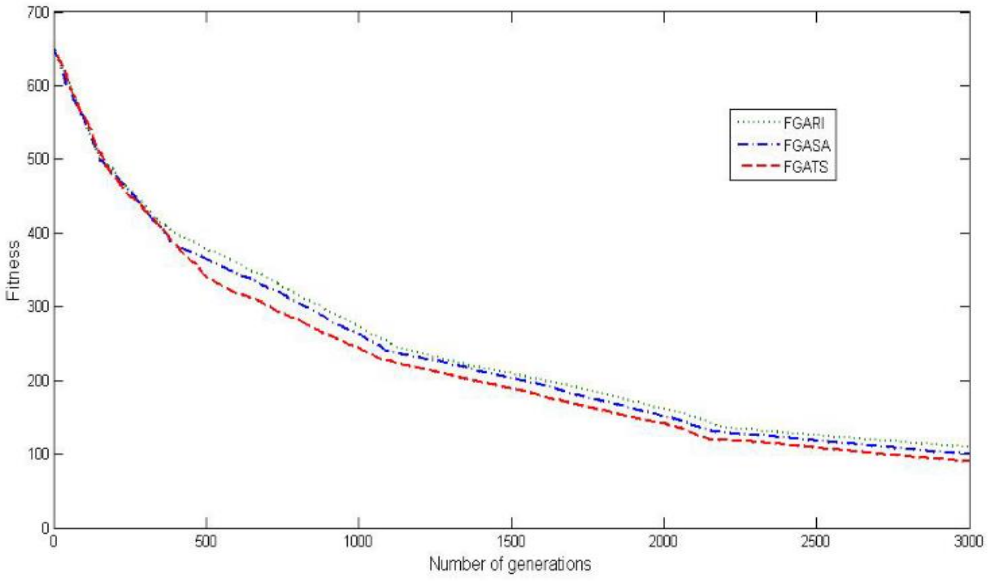
از شکل بالا می‌توان مشاهده کرد که وقتی مقدار Pm از 0.1 به 0.5 افزایش می‌یابد، عملکرد GA به دلیل افزایش احتمال تولید راه‌حل‌های جدید بهبود می‌یابد.

با این حال، زمانی که مقدار Pm بیشتر افزایش یابد، عملکرد GA کاهش می‌یابد. این به این دلیل است که مقدار بالای Pm باعث تغییر ناگهانی می‌شود و پس از چند نسل، GAممکن است در یک حالت زیر بهینه گرفتار شود و بنابراین نتواند به راه‌حل بهینه دست یابد.

مقایسه کارایی سه الگوریتم پیشنهادی:

شکل زیر کارایی سه الگوریتم پیشنهادی را نشان می‌دهد. همان‌طور که در قسمت قبل ذکر شده است، پایه سه الگوریتم یکسان است و تنها تفاوت در روش جستجوی محلی است که در آن‌ها استفاده می‌شود.

همان‌طور که در شکل زیر مشاهده می‌شود، FGATSبهترین عملکرد را دارد و FGARI بدترین عملکرد را دارد. با این حال، تفاوت عملکرد چندان قابل توجه نیست و هر سه الگوریتم توانایی قابل قبولی برای حل UCTP دارند.



همانطور که قبل تر ذکر شد، پایه این الگوریتم‌ها یکسان است و تنها تفاوت در روش جستجوی محلی است که در آن‌ها استفاده می‌شود.

با توجه به شبه‌الگوریتم سه الگوریتم جستجوی محلی که در شکل ها نشان داده شده‌اند، پیچیدگی زمانی آن‌ها یکسان است.

بنابراین، الگوریتم GA با TS می‌تواند راه‌حل بهینه را در زمان کمتری پیدا کند، به‌ویژه زمانی که مجموعه داده بزرگ باشد.

مراجع :

[1] Abdullah S. “Heuristic Approaches for University Timetabling Problems”. PhD thesis, School of Computer Science and Information Technology, The University of Nottingham, United Kingdom, 2006.

[2] Shahvali M., and et al. “A fuzzy genetic algorithm with local search for university course timetabling”, Proc. of ICMI2011, pp.250-254, 2011.

[3] Yang S, Jat S. N. “Genetic algorithms with guided and local search strategies for university course timetabling”. IEEE TSMC, vol. 41, NO. 1, January, 2011.

[4] Abdullah S. and et al, “Using a randomized iterative improvement algorithm with composite neighborhood structures”. Proc. of 6th ICMH, pp. 153–169, 2007.

[5] Chaudhuri A, De K. “Fuzzy Genetic Heuristic for University Course Timetable Problem”. IJASCA, Vol. 2, No. 1, March, 2010

[6] Meysam Shahvali Kohshori1and Mohammad Saniee Abadeh, “Hybrid Genetic Algorithms for University Course Timetabling”

[7] Mei Ching Chen, San nah sze , Say Leng Goh, Nasser R.Saber, “A survey of University course timetabling problem : Percpectives, Trends and opportunities”

[8] Achini kumari herath of university of Missisiippi, “Genetic algorithm for university course timetabling problem”

[9] White G, Xie B, Zonjic S., “Using tabu search with longer term memoryand relaxation to create examination timetables”. EJOR, Vol.153, No.16, pp.80-91, 2004

[10] Burke E. K. and et al, “A tabu-search hyper-heuristic for timetabling and rostering”. Journal of Heuristics. 9(6), pp 451-470, 2003

[11] Zervoudakis K, Stamatopoulos P., “A generic object-oriented constraint-based model for university course timetabling” . Proc of 3rdICPTAT, pp 28-47, 2001

[12] Apurva T Kanavade , Akshata D Kshirsagar , Sonali N Kokane, “Time table scheduling using genetic algorithm”

[13] Thatchai Thepphakorn , Pupong Pongcharoen ,Chris Hicks b ,” An ant colony based timetabling tool”

[14] Gadi Solotorevsky ,Ehud Gudes ,”solving a real life time tabling and transportation problem using distributed csp techniques”