

بهینه‌سازی برنامه‌ریزی هفتگی دروس دانشگاهی با روش‌های جست‌وجوی محلی

رامین باشی‌زاده^۱، مریم حسن‌زاده^۲

^۱ دانشجوی کارشناسی ارشد، گروه مهندسی کامپیوتر، دانشگاه شاهد، تهران، ایران

ramin.bashizade@gmail.com

^۲ استادیار، گروه مهندسی کامپیوتر، دانشگاه شاهد، تهران، ایران

hasanzadeh@shahed.ac.ir

چکیده: مسئله برنامه‌ریزی هفتگی دانشگاه، مسئله پیچیده‌ای است که حل آن به کمک رایانه، مدت‌هاست که زمینه فعالیت است. برای حل این مسئله، باید دروس را با توجه به محدودیت‌های سخت و نرم به زمان‌ها نسبت داد. محدودیت‌های سخت باید حتماً رعایت شوند (برخی از آن‌ها تحت شرایطی با هزینه‌ای بالا قابل نقض‌اند) و هدف، رعایت هرچه بیشتر محدودیت‌های نرم است. در این مقاله، کوشش شده با ارائه الگوریتم‌های جست‌وجوی محلی مناسب، یک برنامه هفتگی که محدودیت‌های سخت در آن رعایت شده، بهبود گردد. در واقع، ورودی روش پیشنهادی یک برنامه قابل قبول است که به وسیله یک الگوریتم جست‌وجوی خاص مسائل ارضای محدودیت به دست آمده است. این ورودی قابل قبول جهت نیل به سمت جواب بهینه به الگوریتم پیشنهادی ارائه می‌شود. نتایج نشان می‌دهد که روش پیشنهادی برای داده‌های واقعی در فضایی با ابعاد بالا و محدودیت‌های پیچیده، عملکرد بسیار خوبی دارد.

واژه‌های کلیدی: برنامه‌ریزی هفتگی دانشگاه، جست‌وجوی محلی، جدول زمان‌بندی.

۱. مقدمه

مسئله برنامه‌ریزی دانشگاهی به دو دسته برنامه‌ریزی برای امتحانات و برنامه‌ریزی هفتگی دروس تقسیم می‌شود. مقصود از برنامه‌ریزی در این مقاله، دسته دوم است. برنامه‌ریزی هفتگی بنا به تعریف کارتر و لاپورته (۱۹۹۸) [۱] عبارت از «یک مسئله مقاداردهی چند بعدی است که در آن، دانشجویان و استادان به دروس، گروه دروس و یا کلاس‌ها نسبت داده می‌شوند». در این میان محدودیت‌هایی به صورت طبیعی و یا با توجه به خواست کاربر وجود دارد که باید در هنگام حل مسئله در نظر گرفته شود.

به علت پیچیدگی و زمان‌بر بودن، و همچنین عدم رضایت‌بخشی برنامه‌ریزی دستی در برخی زمینه‌ها، سال‌هاست که به مسئله برنامه‌ریزی خودکار، توجه ویژه‌ای می‌شود. از سال ۱۹۶۲ که گاتلیب [۲]، نخستین مقاله را در این زمینه ارائه داد، مقالات بسیاری در کنفرانس‌ها و مجله‌ها منتشر شده است.

طی سال‌های اخیر نیز مقالات زیادی در این زمینه ارائه شده است. سوچا و دیگران (۲۰۰۲) [۳]، یک روش جست‌وجوی محلی و یک روش بر پایه گروه مورچه‌ها ارائه کردند. برک و سایرین (۲۰۰۳) [۴] نیز، یک روش فوق مکاشفه‌ای جست‌وجوی تابویی را معرفی کردند که هدف آن، تعمیم سیستم جست‌وجو بود و علاوه بر برنامه‌ریزی هفتگی، روی مسئله تنظیم وظایف پرستاران هم آزمایش شد. همچنین برک و دیگران (۲۰۰۶) [۵]، یک روش فوق مکاشفه‌ای گرافی ارائه کردند. عبدالله و سایرین (۲۰۰۵) [۶]، یک جست‌وجوی همسایگی متغیر با لیست ممنوعه ثابت را به کار گرفتند که ناخوشایندی همسایگی‌های بدون استفاده را افزایش می‌داد. ارنستو فر (۲۰۱۰) [۷]، یک الگوریتم ژنتیک هیبریدی را برای برنامه‌ریزی با در نظر گرفتن نفع دانشجویان ارائه کرد. دکتر منجمی و سایرین [۸] نیز با ایجاد تغییراتی در الگوریتم ژنتیک عادی، از آن برای حل این مسئله بهره بردند.

بسیاری از مقالات دیگر نیز در این زمینه منتشر شده‌اند که دارای رویکردهای متفاوت از قبیل شبیه‌سازی حرارت، گروه مورچه‌ها، الگوریتم ژنتیک و... هستند که هر یک برای مسئله‌هایی با اندازه‌های متفاوت مناسب است؛ برای مثال، راه حل‌های

جست‌وجوی محلی برای مسائل با اندازه کوچک و متوسط کارا هستند، ولی برای مسائل بزرگ گروه مورچه‌ها و روش فوق مکاشفه‌ای گرافی نتایج مناسبی دارند [۹]. اغلب روش‌های ارائه شده بر روی داده‌های آزمون استاندارد تمرکز داشته و ارزیابی شده‌اند که پیچیدگی‌های یک محیط آموزشی واقعی را ندارند؛ پیچیدگی‌هایی که به عنوان نمونه حاصل از تعداد گروه‌های متعدد برای ارائه یک درس واحد، دروس مشترک بین رشته‌گرایش‌های مختلف و پیچیدگی محدودیت‌ها از نظر کم بودن تعداد دامنه متغیرهای آن‌هاست.

هدف این مقاله، بهبودسازی سیستم برنامه‌ریزی هفتگی دانشگاه با شرایط پیچیده واقعی با به کارگیری مناسب سه الگوریتم جست‌وجوی محلی موفق و متداول، و مقایسه نتایج آن‌ها با یکدیگر است. در تمامی الگوریتم‌ها، ورودی برنامه‌ای قابل قبول است که تمامی محدودیت‌های سخت سطح صفر و بخشی از محدودیت‌های سخت سطح یک [۱۰] در آن رعایت شده و هدف الگوریتم‌ها به حداقل رساندن نقض محدودیت‌های نرم [۱۰] است. لازم به ذکر است که در این مقاله، این برنامه قابل قبول توسط یک الگوریتم جست‌وجوی خاص مسائل ارضای محدودیت [۱۱] در تلفیق با الگوریتم ژنتیک به دست آمده؛ البته بحث مقاله مستقل از این موضوع است و هر الگوریتم مناسب دیگری نیز می‌تواند ورودی را مهیا کند.

در قسمت دوم مقاله، به معرفی سیستم پیشنهادی برای پیاده‌سازی برنامه هفتگی دانشگاه پرداخته شده است. قسمت سوم به توضیح الگوریتم‌های به کار رفته برای بهبودسازی سیستم برنامه‌ریزی هفتگی اختصاص دارد. نتایج حاصل از اجرای الگوریتم‌ها، ارزیابی و مقایسه آن‌ها نیز در قسمت چهارم آمده است. در قسمت پنجم نیز جمع‌بندی و پیشنهادهایی برای ادامه کار مطرح شده است.

۲. سیستم پیشنهادی

سیستمی که در این مقاله برای طراحی برنامه هفتگی به کار رفته، شامل موجودیت‌های زمان، استاد و درس است. روزهای هفته به تعدادی زمان برای ارائه دروس تقسیم می‌شوند. هر استاد، دروسی را ارائه می‌دهد و زمان‌های مشخصی در دانشگاه

است. این محدودیت‌ها به دو دسته محدودیت‌های سطح دو و اولویت‌ها دسته‌بندی می‌شوند.

محدودیت‌های سطح دو مربوط به دروسی می‌شوند که ترجیح بر این است که دانشجو بتواند آن‌ها را به طور هم‌زمان اخذ کند (دروس پیش‌نیاز و پس‌نیازی که گروه به دانشجو اجازه می‌دهد در صورت قبول نشدن در درس پیش‌نیاز، هر دو را در یک ترم اخذ کند).

همچنین اولویت‌هایی وجود دارند که عدم رعایت آن‌ها برای برنامه تولیدشده، ناخوشایندی کمی به همراه دارد. این اولویت‌ها می‌تواند مربوط به دروس دو ترم متوالی در نمودار درسی، فشردگی برنامه دانشجویان در طول هفته و مواردی این‌گونه باشد.

۳-۲. تابع برازش

هدف از معرفی محدودیت‌ها، امکان ارزیابی برنامه‌های تولیدشده است. حال برای عملی کردن این هدف، از تابع برازش استفاده می‌کنیم. مقدار این تابع، ناخوشایندی برنامه را نشان می‌دهد. تابع برازش با تشخیص نقض محدودیت‌ها (به غیر از محدودیت‌های سطح صفر) و جمع کردن ناخوشایندی هر کدام از آن‌ها، مقداری را برمی‌گرداند که به صورت نسبی نشان‌دهنده میزان ناخوشایندی برنامه است. به این معنی که این مقدار برای مقایسه دو برنامه با هم مناسب است، ولی برای کاربرد، مفهوم خاصی ندارد.

باید این نکته را در نظر داشت که الگوریتم‌های پیاده‌سازی شده در این مقاله، هیچ‌گاه یک برنامه معتبر را (که محدودیت‌های سطح صفر در آن رعایت شده‌اند) به برنامه نامعتبر تبدیل نمی‌کنند، لذا از بررسی محدودیت‌های سطح صفر در تابع برازش اجتناب شده است.

۳. معرفی الگوریتم‌ها

در این مقاله، برای حل مسئله برنامه‌ریزی هفتگی آموزشی، از سه الگوریتم جست‌وجوی محلی استفاده شده است. همان‌طور که در مقدمه اشاره شد، در شماره چهار به بررسی و مقایسه این سیستم‌ها پرداخته شده است.

حضور دارد. هر درس هم با توجه به تعداد واحدهایی که دارد، به تعدادی زمان نیاز دارد. برای تنظیم یک برنامه هفتگی با در دست داشتن این سه موجودیت، برخی محدودیت‌های بدیهی وجود دارد که از آن‌ها با عنوان محدودیت‌های سخت یاد می‌شود. برخی موارد دیگر هم ترجیح کاربر است که به آن‌ها محدودیت‌های نرم گفته می‌شود. در ادامه به معرفی این محدودیت‌ها پرداخته شده است.

البته جزئیات دقیق دسته‌بندی محدودیت‌ها خارج از موضوع این مقاله است و در [۱۰] آمده است. در اینجا تنها تعریف کلی از آن‌ها ارائه شده تا درک عملکرد الگوریتم‌های جست‌وجوی محلی آسان‌تر شود.

۱-۲. محدودیت‌های سخت

محدودیت‌های سخت به دسته‌ای از محدودیت‌ها گفته می‌شود که نقض آن‌ها منجر به نامعتبر شدن برنامه هفتگی، یا نامطلوب شدن آن به میزان زیاد می‌شود. این محدودیت‌ها خود به دو سطح صفر و یک تقسیم می‌شوند. محدودیت‌های سطح صفر عبارت‌اند از:

- یکسان بودن استاد دو درس مختلف در یک زمان؛
- ارائه درسی در زمانی غیر از زمان‌های حضور استاد آن؛
- تداخل زمانی دو درس هم‌مکان (مانند دو آزمایشگاه).

نقض این دسته از محدودیت‌های سخت موجب نامعتبر شدن برنامه می‌شود، اما دسته دیگری از محدودیت‌های سخت وجود دارند که نقض آن‌ها برنامه را نامعتبر نمی‌کند، بلکه باعث می‌شود ناخوشایندی زیادی به آن تحمیل شود. این محدودیت‌ها که محدودیت‌های سطح یک نامیده می‌شوند، شامل دروسی هستند که دانشجو باید قادر باشد آن‌ها را در یک ترم اخذ کند. نقض این محدودیت‌ها بیشترین ناخوشایندی را به برنامه تحمیل می‌کند.

۲-۲. محدودیت‌های نرم

محدودیت‌های نرم، قابلیت نقض شدن دارند، ولی نقض هر کدام از آن‌ها میزان معینی ناخوشایندی را به برنامه تحمیل می‌کند که بسیار کمتر از ناخوشایندی نقض محدودیت‌های سطح یک

آرایه‌دما به کار رفته در این الگوریتم، با استفاده از خاصیت تابع لگاریتم، مقداره‌ی شده است. برای این کار از دو رابطه (۱) و (۲) استفاده شده است:

$$\text{Temperature}[i] = K * \log(\text{Temperature.Count} - i + 1) - \log \text{Temperature.Count} - i \quad (1)$$

$$K = \frac{1}{\log(\text{Temperature.Count} + 1) - \log \text{Temperature.Count} + 1} \quad (2)$$

در این دو رابطه، مقدار دما در خانه نام، آرایه‌دما را نشان می‌دهد و *Temperature.Count* بیانگر تعداد خانه‌های آرایه است.

```

employing a Set the initial solution CurSol by
constructive heuristic;
Calculate initial cost function f(CurSol);
Set best solution BestSol ← CurSol;
do while (not termination criteria)
  Apply neighbourhood structure on CurSol,
NextSol;
  Calculate cost function f(NextSol);
  if (f(NextSol) < f(CurSol))
    CurSol ← NextSol;
    BestSol ← NextSol;
  end if
else
   $\delta = f(\text{Sol}^*) - f(\text{Sol});$ 
  Generate RandNum, a random number in
  [0,1];
  if (RandNum <  $e^{-\delta/\text{Temperature}[i]}$ )
    CurSol ← NextSol;
  end if
  Decrease Iteration Counter;
end do
    
```

الگوریتم (۲): شبه‌کد الگوریتم شبیه‌سازی حرارت [۱۲]

شیوه کار در این الگوریتم به این نحو است که مابعد بهتر همواره پذیرفته می‌شود. مابعد بدتر با احتمالی متناسب با میزان بدتر بودنش و همچنین مقدار آرایه‌دما در آن تکرار از الگوریتم، مورد قبول قرار می‌گیرد. نحوه پذیرش مابعد بدتر به این صورت است که عددی تصادفی در بازه [۰، ۱] تولید می‌شود و در صورتی که این عدد بیشتر از $e^{-\delta/\text{Temperature}[i]}$ باشد، مابعد پذیرفته می‌شود. در طول این روند، همواره بهترین برنامه ذخیره می‌شود و در نهایت با پایان یافتن الگوریتم برگردانده می‌شود.

۱-۳. الگوریتم تپه‌نوردی اولین انتخاب

در الگوریتم تپه‌نوردی اولین انتخاب [۱۲]، شیوه کار بدین صورت است که به کمک یک ساختار همسایگی، مابعدهای حالت فعلی تا جایی محاسبه شده‌اند که حالتی بهتر به دست آید. این الگوریتم برای مسائلی با تعداد مابعدهای بسیار زیاد مناسب است. شبه‌کد به کار رفته در این مقاله در الگوریتم (۱) آمده است.

```

Set the initial solution CurSol by employing a
constructive heuristic;
Calculate initial cost function f(CurSol);
do while (not termination criteria)
  Apply neighborhood structure on CurSol,
NextSol;
  Calculate cost function
  f(NextSol);
  if (f(NextSol) < f(CurSol))
    CurSol ← NextSol;
  end if
end do
    
```

الگوریتم (۱): شبه‌کد الگوریتم تپه‌نوردی اولین انتخاب [۱۲]

همان‌طور که دیده می‌شود، ورودی یک برنامه قابل قبول است. در یک حلقه مابعدهای این برنامه تا جایی محاسبه می‌شود که به برنامه‌ای بهتر از برنامه فعلی برسیم تا جایگزین آن در نظر گرفته شود. میزان مطلوب بودن برنامه‌ها با استفاده از تابع برازش مشخص می‌شود که در اینجا از آن با *f* یاد شده است. این کار تا جایی ادامه پیدا می‌کند که میزان ناخوشایندی برنامه فعلی به صفر برسد و یا به این نتیجه برسیم که ادامه دادن الگوریتم به صرفه نیست. بدین معنی که تعداد مابعدهای محاسبه شده به قدری بالا برود که عملاً بتوان گفت برنامه بهتری قابل دست‌یابی نیست.

۲-۳. الگوریتم شبیه‌سازی حرارت

الگوریتم شبیه‌سازی حرارت [۱۲] با دخیل کردن عامل دما در انتخاب مابعد، امکان انتخاب شدن مابعد ناخوشایندتر از حالت فعلی را می‌دهد. با این کار، این الگوریتم برخلاف الگوریتم تپه‌نوردی، عملکردی حریصانه ندارد و امکان فرار از ماکسیمم محلی را به وجود می‌آورد.

الگوریتم (۲)، شبه‌کد الگوریتم پیاده‌سازی شده در این مقاله است.

۳-۳. الگوریتم بهینه‌سازی تکراری تصادفی با همسایگی‌های ترکیبی

الگوریتم بهینه‌سازی تکراری تصادفی با همسایگی‌های ترکیبی [۹]، سومین الگوریتمی است که برای حل مسئله برنامه‌ریزی هفتگی دانشگاه به کار گرفته شده است. در این الگوریتم به جای یک تابع همسایگی، از چند تابع برای پیدا کردن مابعد استفاده می‌شود. به این شیوه، همسایگی‌های ترکیبی گفته می‌شود. این کار، فرار از ماکسیمم محلی یک تابع همسایگی را آسان‌تر می‌کند. این الگوریتم نیز همانند الگوریتم شبیه‌سازی حرارت، امکان پذیرش برنامه بدتر را با احتمالی ممکن می‌سازد. نحوه پذیرش همانند الگوریتم شبیه‌سازی حرارت است، با این تفاوت که در اینجا عامل دما در نظر گرفته نشده است. شبه کد این الگوریتم، مطابق الگوریتم (۳) است.

```

Set the initial solution CurSol by employing a
constructive heuristic;
Calculate initial cost function  $f(\text{CurSol})$ ;
Set best solution  $\text{Solbest} \leftarrow \text{CurSol}$ ;
do while (not termination criteria)
  for  $i = 1$  to  $K$  where  $K$  is the total number of
  neighbourhood structures
    Apply neighbourhood structure  $i$  on
     $\text{CurSol}$ ,  $\text{TempSol}[i]$ ;
    Calculate cost function
     $f(\text{TempSol}[i])$ ;
  end for
  Find the best solution among  $\text{TempSol}[i]$ 
  where  $i \in \{1, \dots, K\}$ 
  call new solution  $\text{Sol}^*$ ;
  if  $(f(\text{Sol}^*) < f(\text{Solbest}))$ 
     $\text{CurSol} \leftarrow \text{Sol}^*$ ;
     $\text{Solbest} \leftarrow \text{Sol}^*$ ;
  end if
  else
    Apply an exponential Monte Carlo
    where:
     $\delta = f(\text{Sol}^*) - f(\text{CurSol})$ ;
    Generate  $\text{RandNum}$ , a random
    number in  $[0, 1]$ ;
    if  $(\text{RandNum} < e^{-\delta})$ 
       $\text{CurSol} \leftarrow \text{Sol}^*$ ;
    end if
  end else
end do

```

الگوریتم (۳): شبه کد الگوریتم بهینه‌سازی تکراری تصادفی با همسایگی‌های ترکیبی [۱۲]

در این الگوریتم، در هر تکرار، مقدار ناخوشایندی برای تمام همسایگی‌ها محاسبه، و بهترین آن‌ها انتخاب می‌شود. از این مرحله به بعد، شیوه کار شبیه الگوریتم شبیه‌سازی حرارت است. توابع همسایگی که در این مقاله به کار رفته، عبارت‌اند از:

- انتخاب یک درس به صورت تصادفی و انتقال آن به زمانی دیگر؛
- انتخاب دو درس به صورت تصادفی و جابجایی زمان ارائه آن‌ها؛
- انتخاب ناخوشایندترین درس از میان ۱۰٪ دروس (که به صورت تصادفی انتخاب شده‌اند) و انتقال آن به بهترین زمان ممکن.

منظور از ناخوشایندترین درس، درسی است که بیشترین میزان ناخوشایندی را به برنامه تحمیل کرده است.

۴. ارزیابی

برای ارزیابی این الگوریتم‌ها از داده‌های واقعی یک نیم‌سال دانشکده فنی و مهندسی دانشگاه شاهد (نیم‌سال دوم سال تحصیلی ۱۳۸۸-۱۳۸۹) در دو رشته کامپیوتر-سخت‌افزار و برق-الکترونیک استفاده شده است. همچنین برنامه قابل قبول ورودی توسط یک الگوریتم ژنتیک [۱۳] تهیه شده است. در بخش ۳، سه الگوریتم جست‌وجوی محلی معرفی شد که برای ارزیابی آن‌ها با توابع همسایگی معرفی شده، می‌توان ۷ حالت را در نظر گرفت. هر کدام از الگوریتم‌های تپه‌نوردی اولین انتخاب و شبیه‌سازی حرارت با سه تابع همسایگی معرفی شده، و نیز الگوریتم بهینه‌سازی تکراری تصادفی با همسایگی‌های ترکیبی، این ۷ حالت را تشکیل می‌دهند. در جدول (۱) پارامترهای عمومی آزمون آمده است. تعداد محدودیت‌ها بیانگر مجموع محدودیت‌های تولید شده توسط برنامه و اولویت‌های وارد شده توسط کاربر است. تعداد متغیرها، مجموع تعداد ساعت‌های دروس را نشان می‌دهد. میانگین درجه آزادی، متوسط تعداد زمان‌هایی است که هر متغیر می‌توانسته با توجه به زمان‌های حضور استاد مربوطه ارائه شود. فضای حالت نیز حاصل ضرب تعداد زمان‌های مجاز متغیرهاست.

جدول (۲): مقدار ناخوشایندی محدودیت‌ها

ناخوشایندی	سطح	
۱۰۰۰۰۰۰	صفر	
۵۰۰۰۰	یک	
۱۰۰۰۰	دو	
۷	تداخل دروس با اختلاف سطح ۰	اولویت
۴	تداخل دروس با اختلاف سطح ۱	
۱	تداخل دروس با اختلاف سطح ۲	
۱	تداخل دروس با اختلاف سطح ۳	
۱۰۰۰۰	فشرده‌گی برنامه در ۴ روز	
۵۰۰۰۰	فشرده‌گی برنامه در ۵ روز	
۲۵۰۰۰۰	فشرده‌گی برنامه در ۶ روز	
۵	توالی دو ساعت یک درس	

جدول (۱): پارامترهای عمومی

تعداد محدودیت‌ها	تعداد متغیرها	میانگین درجه آزادی	فضای حالت
۱۱۴۳۶	۱۹۶	۳/۴۱۸	$۲/۵۰۶۳ \times ۱۰^{۱۹۳}$

جدول (۲)، ناخوشایندی حاصل از نقض هر محدودیت را نشان می‌دهد. منظور از سطح درس در این جدول، عمق آن در زیردرخت پیش‌نیازی خود است. اگر دانشجو باید برای اخذ یک درس، پیش‌نیاز آن را گذرانده باشد، سطح درس پیش‌نیاز، یکی کمتر از سطح درس مورد نظر است. سطح دروسی که هیچ پیش‌نیازی ندارد، صفر است و سطح دروس بعدی با توجه به آن‌ها به دست می‌آید. طبعاً اگر از دو مسیر به یک درس برسیم، بیشینه سطح دو مسیر به عنوان سطح آن درس در نظر گرفته می‌شود. علت در نظر گرفتن این اولویت، این مسئله است که دانشجویانی که از نمودار درسی عقب افتاده‌اند، بتوانند در نیم‌سال‌های جلوتر، خود را راحت‌تر به نمودار برسانند.

جدول (۳): نتایج آزمون‌ها

الگوریتم	ناخوشایندی ورودی	تعداد تکرار	ناخوشایندی خروجی	زمان حل (ثانیه)
همسایگی‌های ترکیبی	۳۲۵۸۳۲۶	۵۰	۲۰۰۴۹۵۵	۱۲۸۱۵
تپه‌نوردی - انتقال درس		۵۰	۲۱۵۸۹۷۲	۹۵۴۹
تپه‌نوردی - جایجایی دو درس		۵۰	۳۱۵۸۲۶۸	۲۴۸۱
تپه‌نوردی - انتقال بدترین		۵۰	۱۵۵۶۹۷۷	۱۹۸۵۶
شبیه‌سازی حرارت - انتقال درس		۵۰	۲۴۰۷۲۸۱	۲۷۵۹
شبیه‌سازی حرارت - جایجایی دو درس		۵۰	۳۲۵۸۲۷۶	۳۲۶۷
شبیه‌سازی حرارت - انتقال بدترین		۵۰	۱۶۰۸۰۳۱	۹۲۳۸

ملاحظه می‌شود که بهترین عملکرد را الگوریتم تپه‌نوردی اولین انتخاب به همراه تابع همسایگی انتقال بدترین درس (از میان ۱۰٪ دروس که به صورت تصادفی انتخاب شده‌اند) به بهترین مکان داشته است؛ البته بیشترین زمان اجرا نیز مربوط به همین حالت است.

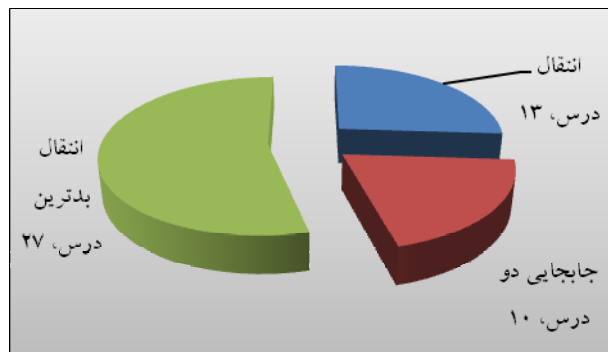
از طرفی، بهترین تابع همسایگی، انتقال بدترین درس (از میان ۱۰٪ دروس که به صورت تصادفی انتخاب شده‌اند) به بهترین مکان بوده است، زیرا اولاً در سه اجرای الگوریتم‌های تپه‌نوردی، اولین انتخاب و شبیه‌سازی حرارت با توابع همسایگی مختلف، بهترین عملکرد را در هر الگوریتم دارد؛ ثانیاً در شکل

در جدول (۳)، نتایج حاصل از اجرای ۷ حالت ذکرشده آمده است. توضیحی در مورد الگوریتم تپه‌نوردی اولین انتخاب، لازم به ذکر است. این الگوریتم در هر مرحله، حداکثر ۳۰ مابعد را تولید می‌کند و در صورتی که هیچ کدام از حالت فعلی بهتر نبودند، فرض می‌کند که دیگر امکان بهبود وجود ندارد. اما در صورتی که یکی از این مابعد‌ها از حالت فعلی بهتر باشد، آن را به عنوان حالت کنونی در نظر می‌گیرد و مجدداً تا ۳۰ مابعد را برای آن تولید می‌کند. این عمل حداکثر تا ۵۰ بار تکرار می‌شود؛ بنابراین، عدد ۵۰ در جدول، آشکارکننده این امر است.

عنوان ورودی برای این الگوریتم‌ها تهیه می‌شد که آن هم از طریق یک الگوریتم ژنتیک فراهم گردید. برای آزمایش روش‌ها و توابع همسایگی پیاده‌سازی شده، ۷ حالت به کار برده شد. نتایج، گویای این امر بود که بدون توجه به زمان اجرا، الگوریتم تپه‌نوردی اولین انتخاب به همراه تابع همسایگی انتقال بدترین درس (از میان ۱۰٪ دروس که به صورت تصادفی انتخاب شده‌اند) به بهترین مکان، بهترین نتیجه را داد؛ البته نتایج با بهینه‌سازی کد برنامه بهبود کلی می‌یابد. همچنین انتظار می‌رود با عوض شدن محدوده ناخوشایندی در رفتار الگوریتم‌ها تغییراتی مشاهده شود.

از مواردی که برای ادامه کار در این زمینه پیشنهاد می‌شود، ارتقای توابع همسایگی است؛ برای مثال، می‌توان چند درس را هم‌زمان جابجا کرد. یکی دیگر از کارهای انجام‌شده، به کارگیری جست‌وجوی ممنوعه است یعنی برای جلوگیری از اتلاف زمان و منابع در هنگام اجرای برنامه، همسایگی‌های بدون استفاده را تشخیص دهیم و آن‌ها را محاسبه نکنیم. از دیگر موارد مفید این است که برای افزایش سرعت اجرا در الگوریتم بهینه‌سازی تکراری تصادفی با همسایگی‌های ترکیبی، روی پردازش موازی برای محاسبه هم‌زمان توابع همسایگی کار شود.

(۱) نیز که تعداد انتخاب‌های توابع همسایگی در الگوریتم بهینه‌سازی تکراری تصادفی با همسایگی‌های ترکیبی آمده، این تابع بیشترین میزان انتخاب را دارد. از طرف دیگر، تابع همسایگی جابجایی دو درس، بدترین عملکرد را دارد که البته انتظار می‌رود توازن بیشتری در برنامه‌هایی با ناخوشایندی پایین‌تر برقرار باشد.



شکل (۱): تعداد استفاده از توابع همسایگی در الگوریتم همسایگی‌های ترکیبی

۵. نتیجه‌گیری

هدف این مقاله، بهینه‌سازی برنامه‌ریزی هفتگی دانشگاه با به کارگیری چند الگوریتم جست‌وجوی محلی متداول در شرایط پیچیده واقعی بود. برای این منظور، باید یک برنامه قابل قبول به

مراجع

- [1] Carter, MW, Laporte, G., *Recent developments in practical course timetabling*, Springer Lecture Notes in Computer Science, Vol. 1408, pp. 3-19, 1998.
- [2] Gotlieb, C.C., *The Construction of Class-Teacher timetables*, In Proceedings of IFIP Congress, North-Holland Pub. Co., Amsterdam, pp. 73-77, 1962.
- [3] Socha, K, Knowles, J, Samples, M., *A max-min ant system for the university course timetabling problem*, Springer Lecture Notes in Computer Science, Vol. 2563, pp. 1-13, 2002.
- [4] Burke, EK, Kendall, G, Soubeiga, E., *A tabu search hyperheuristic for timetabling and rostering*, Journal of Heuristics, Vol. 9, No. 6, pp. 451-470, 2003.
- [5] Burke, EK, McCollum, B, Meisels, A, Petrovic, S, Qu, R., *A Graph-Based Hyper-Heuristic for Educational Timetabling Problems*, European Journal of Operational Research, Vol. 176(1), pp. 177-192, 2007.
- [6] Abdullah, S, Burke, EK, McCollum, B., *An investigation of variable neighborhood search for university course timetabling*, Proceedings of The 2nd Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2005), pp. 413-427, New York, USA, July 18th-21st 2005.
- [7] Nunez, E.F.Q., *A Hybrid Genetic Algorithm for the Student-Aware University Course Timetabling Problem*, Honors Projects, 2010.

[۸] منجمی، سید امیرحسین، سولماز مسعودیان، افسانه استکی، و ناصر نعمت‌بخش، طراحی جدول زمان‌بندی خودکار برای دروس دانشگاهی با استفاده از الگوریتم‌های ژنتیک، نشریه علمی- پژوهشی فناوری آموزش، سال چهارم، جلد ۴، شماره ۲، زمستان ۱۳۸۸.

[9] Doerner, K.F, et al., *Using a Randomised Iterative Improvement Algorithm with Composite Neighbourhood Structures for the University Course Timetabling Problem*, *Metaheuristics*, Vol. 39(IV) pp. 153-169, 2007.

[۱۰] شریف‌نیا، شهرام، طراحی شیء‌گرای سیستم برنامه‌ریزی آموزشی، پایان‌نامه کارشناسی، دانشگاه شاهد، ۱۳۹۰.

[۱۱] رحیمی خرسند، جمال، استفاده از روش‌های حل مسائل ارضای محدودیت در برنامه‌ریزی آموزشی، پایان‌نامه کارشناسی، دانشگاه شاهد، ۱۳۹۰.

[12] Russell, S.J, Norvig, P., *Artificial Intelligence: A Modern Approach*, Prentice Hall, 2010.

[۱۳] آدینه‌زاده، فرانک، حل مسئله برنامه‌ریزی آموزشی به کمک الگوریتم ژنتیک، پایان‌نامه کارشناسی، دانشگاه شاهد، ۱۳۹۰.