دانشکده علوم ریاضی
گروه ریاضی کاربردی

پایان‌نامه

**برای دریافت درجه کارشناسی ارشد در رشته
ریاضی کاربردی، گرایش تحقیق در عملیات**

عنوان

مساله زمان‌بندی امتحانات

استاد راهنما

دکتر جعفر فتحعلی

استاد مشاور

دکتر محسن اصیلی

دانشجو

آیلر پورآدینه

۹۲/۹/۳۰

تقدیم به:

مادرم: که همواره مهربانی‌هایش بدرقه راهم بوده است.

پدرم: که رسم زندگانی‌ام آموخت.

سپاس‌گزاری...

امتنان و سپاس می‌گزارم تلاش‌ها، زحمات و راهنمایی‌های ارزشمند و صبورانه استاد فرزانه و گرانمایه‌ام، جناب دکتر جعفر فتحعلی را که با حمیت و جدیت مرا به دقت، اندیشه، درک و تعمق وامی‌داشتند.

آیلر پورآدینه
۹۲/۹/۳۰

تعهدنامه

اینجانب **آیلر پورآدینه** دانشجوی کارشناسی ارشد رشته **ریاضی کاربردی** **دانشکده علوم ریاضی** دانشگاه شاهرود، نویسنده پایان‌نامه با عنوان **مساله زمان‌بندی امتحانات** ، تحت راهنمایی **دکتر جعفر فتحعلی** متعهد می‌شوم:

- تحقیقات در این پایان‌نامه توسط اینجانب انجام شده است و از صحت و اصالت برخوردار است.
- در استفاده از نتایج پژوهش‌های دیگر پژوهش‌گران، به مرجع مورد استفاده استناد شده است.
- مطالب این پایان‌نامه، تا کنون توسط خود، یا فرد دیگری برای دریافت هیچ نوع مدرک یا امتیازی در هیچ‌جا ارایه نشده است.
- حقوق معنوی این اثر، به دانشگاه شاهرود متعلق دارد، و مقالات مستخرج با نام " دانشگاه صنعتی شاهرود " یا " Technology University of Shahrood " به چاپ خواهد رسید.
- حقوق معنوی تمام افرادی که در به‌دست آوردن نتایج اصلی پایان‌نامه تاثیرگذار بوده‌اند، در مقالات مستخرج از پایان‌نامه رعایت می‌گردد.
- در تمام مراحل انجام این پایان‌نامه، در مواردی که از موجود زنده (یا بافت‌های آنها) استفاده شده است، ضوابط و اصول اخلاقی رعایت شده است.
- در تمام مراحل انجام این پایان‌نامه، در مواردی که به حوزه اطلاعات شخصی افراد دسترسی یافته (یا استفاده) شده است، اصل رازداری و اصول اخلاق انسانی رعایت شده است.

آیلر پورآدینه
۹۲/۹/۳۰

مالکیت نتایج و حق نشر

- تمام حقوق معنوی این اثر و محصولات آن (مقالات مستخرج، کتاب، برنامه‌های رایانه‌ای، نرم‌افزارها و تجهیزات ساخته شده) متعلق به دانشگاه شاهرود می‌باشد. این مطلب باید به نحو مقتضی، در تولیدات علمی مربوطه ذکر شود.
- استفاده از اطلاعات و نتایج موجود در این پایان‌نامه بدون ذکر منبع مجاز نمی‌باشد.

چکیده

در بسیاری از مراکز آموزشی، از جمله دانشگاه‌ها و مدارس، تهیه جدول زمان‌بندی امری معمول می‌باشد. زمان‌بندی امتحانات درسی یک گروه آموزشی در دانشگاه‌ها و موسسات آموزشی یکی از مشکلات پیش روی برنامه نویسان این موسسات می‌باشد. در این پایان‌نامه دو روش برای حل مساله زمان‌بندی امتحانات بیان می‌شود. روش اول بر مبنای الگوریتم ژنتیک است و روش دوم جست‌وجوی ممنوع است. در هر دو روش راهکارهایی جهت بهبود بخشیدن به این دو روش ارایه شده است. در این پایان نامه تلاش می‌کنیم تداخل‌ها را کاهش داده و ماکزیمم زمان مطالعه را برای دانشجو پیشنهاد کنیم. برنامه برای هر دو روش با استفاده از نرم افزار برنامه نویسی متلب ۲۰۱۰ نوشته شده است و سپس نتایج عددی به‌دست آمده از این دو روش با هم مقایسه می‌شوند.

دانشگاه صنعتی شاهرود
مدیریت تحصیلات تکمیلی
فرم شماره (۶)

باسمه تعالی

شماره:
تاریخ:
ویرایش:

فرم صورت جلسه دفاع از پایان نامه تحصیلی دوره کارشناسی ارشد

با تأییدات خداوند متعال و با استعانت از حضرت ولی عصر (عج) نتیجه ارزیابی جلسه دفاع از پایان نامه کارشناسی ارشد خانم / آقای **آیلر پورآدینه** رشته *ریاضی کاربردی* گرایش **تحقیق در عملیات** تحت عنوان **مساله‌ی زمان‌بندی امتحانات** که در تاریخ **۹۲/۹/۳۰** با حضور هیأت محترم داوران در دانشگاه صنعتی شاهرود برگزار گردید به شرح ذیل اعلام می گردد:

قبول (با درجه : ~~بسیار~~ امتیاز ۱۸/۳۵ ☐ دفاع مجدد ☐ مردود ☐

۱ـ عالی (۲۰ ـ ۱۹)
۲ـ بسیار خوب (۱۸/۹۹ ـ ۱۸)
۳ـ خوب (۱۷/۹۹ ـ۱۶)
۴ـ قابل قبول (۱۵/۹۹ ـ ۱۴)
۵- نمره کمتر از ۱۴ غیر قابل قبول

عضو هیأت داوران	نام ونام خانوادگی	مرتبهٔ علمی	امضاء
۱ـ استادراهنما	دکتر جعفر فتحعلی	دانشیار	
۲ـ استاد مشاور	دکتر محسن اصیلی	استادیار	
۳ـ نماینده شورای تحصیلات تکمیلی	دکتر احمد زیره	دانشیار	
۴ـ استاد ممتحن	دکتر علیرضا ناظمی	استادیار	
۵ ـ استاد ممتحن	دکتر براتاله غزنوی	استادیار	

رئیس دانشکده : دکتر احمد زیره امضاء

لیست مقالات مستخرج از پایان‌نامه

۱. الگوریتم ژنتیک و جست‌وجوی ممنوع برای مساله جدول زمان‌بندی امتحانات

فهرست مطالب

فصل ۱

مفاهیم زمان‌بندی

۱.۱ مقدمه

با پیشرفت تکنولوژی و ماشینی شدن محیط‌های تولید و رقابت‌های موجود در بازار نیازمند مدیریت و برنامه ریزی صحیح در زمینه تولید هستیم. بنابراین مساله زمان‌بندی در عصر حاضر از اهمیت خاصی برخوردار است. زمان‌بندی شامل برنامه‌ریزی، اولویت‌دهی و چیدمان فعالیت‌هایی است که انجام دادن آن‌ها نیازمند ترتیب می‌باشد. در حقیقت، زمان‌بندی ابزاری در جهت بهینه‌سازی استفاده از منابع و امکانات در دسترس است و موجب افزایش بازدهی و بهره‌برداری مناسب از ظرفیت‌ها، کاهش زمان مورد نیاز به منظور تکمیل کارها و در نهایت سودآوری می‌شود.
با توجه به این موضوع که زمان در دنیای کنونی یک محدودیت به‌شمار می‌رود، زمان‌بندی موثر منابع نظیر ماشین آلات، نیروی انسانی و مواد، یک ضرورت در دنیای رقابت امروزی می‌باشد. در برخی از کتاب‌ها زمان‌بندی را هنر اولویت‌دهی و چیدمان فعالیت‌ها برای برآورده کردن نیازمندی‌ها، محدودیت‌ها و اهداف مشخص تلقی می‌کنند. زمان‌بندی جمع‌آوری قواعد، مدل‌ها و روش‌ها جهت تصمیم‌گیری و تعیین یک برنامه‌ی زمانی است. مسائل زمان‌بندی را می‌توان به دو گروه تصمیمات تخصیصی و تصمیمات ترتیبی تقسیم نمود.

۲.۱ تعاریف و تاریخچه

مساله زمان‌بندی از جمله مسائلی است که در زمینه‌های گوناگون، از قبیل آموزش، حمل‌ونقل و ...بکار برده می‌شود. از کاربردهای آن می‌توان موارد زیر را ذکر کرد:

- زمان‌بندی شیفت کاری پرستاران یک بیمارستان
- زمان‌بندی حرکت قطارها و هواپیماها
- زمان‌بندی تولید کارگاهی

- زمان‌بندی ارایه دروس و امتحانات برای مراکز آموزشی و ...

مسایل زمان‌بندی حدودا از سال ۱۹۶۰، توجه محققین را به خود جلب کرده است [۶]. در این زمینه کارهای بسیاری انجام شده است. یکی از حالت‌های مهم مساله زمان‌بندی، زمان‌بندی دروس و امتحانات مراکز آموزشی است. مساله زمان‌بندی امتحانات یک مساله NP -سخت می‌باشد.
مسائل جدول زمان‌بندی دانشگاهی در حالت کلی به دو دسته عمده تقسیم می‌شوند:

- **مسایل جدول زمان‌بندی امتحانات**: اکثر موسسات آموزشی باید مجموعه‌ای از امتحانات را در پایان هر ترم یا هر سال تحصیلی زمان‌بندی کنند. به شکل ساده‌تر این مساله می‌تواند به‌عنوان یک مساله‌ی تخصیص مطرح شود که هدف تخصیص یک مجموعه از امتحانات به تعداد ثابتی از بازه‌های زمانی می‌باشد، به‌طوری که هیچ دانشجویی بیش از یک امتحان در یک زمان خاص نداشته باشد. از طرفی یکسری محدودیت‌ها و اهداف دیگر نیز باید رعایت شوند که با توجه به مراکز آموزشی، این اهداف متفاوت هستند.

- **مسایل جدول زمان‌بندی دروس**: جدول زمان‌بندی دروس درگیر این مساله می‌باشد که دانشجویان، متقاضی یکسری دروس (دروس تئوری، سمینار، آزمایشگاه، دروس خودآموز و ...) می‌باشند که هدف مینیمم کردن ناسازگاری بین آن‌ها و برقراری اهداف و محدودیت‌هایی است که برای موسسه آموزشی مربوطه مطلوب است.

مساله جدول زمان‌بندی امتحانات، برنامه‌ریزی امتحانات یک مجموعه دروس دانشگاهی را بررسی می‌کند. اغلب به‌صورت دستی یا با گرفتن جدول زمان‌بندی سال‌های قبل و اصلاح آن برای سال جدید این کار انجام می‌شود. مساله زمان‌بندی امتحانات عبارت است از زمان‌بندی یک مجموعه از امتحانات دروس دانشگاهی به‌گونه‌ای که از همزمان بودن امتحانات دروس مشترکی که دانشجویان دارند اجتناب می‌کند.
مشکلاتی در تعیین جدول‌های زمان‌بندی مناسب امتحانات برای موسسه‌های تحصیلی وجود دارد. هر چه موسسه‌ها دانشجویان بیشتری را در درس‌های مختلف که تداخل دارند نام نویسی می‌کنند این مشکلات بیشتر می‌شود. یک مساله زمان‌بندی کلی شامل تخصیص یک مجموعه از وقایع (امتحانات دروس، مسابقات ورزشی، جلسات و ...) به تعداد محدودی از بازه‌های زمانی است که در مجموعه‌ای از محدودیت‌ها صدق کنند.
در مساله زمان‌بندی امتحانات، باید برای هر امتحان یک بازه با در نظر گرفتن یک سری محدودیت‌ها قرار داده شود. در حالت کلی مساله زمان‌بندی، به‌عنوان یک مساله‌ی تخصیص در نظر گرفته می‌شود. تعریف مساله تخصیص به‌صورت زیر می‌باشد.

مساله تخصیص

مساله تخصیص یکی از پرکاربردترین مسایل در زمینه تحقیق در عملیات است که گونه‌های مختلفی از آن ارایه شده است. قطعی نبودن داده‌های دنیای واقعی موجب می‌شود تا مسایل کلاسیک در عمل کاربرد چندانی نداشته و مسایل تخصیص تصادفی طراحی شوند.
فرض کنید می‌خواهیم m شغل را به n فرد تخصیص دهیم. اگر شغل j به فرد iام اختصاص یابد هزینه

c_{ij} را متحمل می‌شویم. هر شغل تنها به یک نفر اختصاص داده می‌شود. هدف تخصیص شغل‌ها به افراد به‌گونه‌ای است که مجموع کل هزینه‌ها کمینه شود. برای مدل کردن مساله فرض کنید؛

$$x_{ij} = \begin{cases} ۱ & \text{اگر شغل j به فرد i تخصیص یابد} \\ ۰ & \text{.در غیر این‌صورت} \end{cases}$$

صورت کلی مدل مساله به‌صورت زیر است:

$$\begin{aligned} Min \quad & \sum_{i=۱}^{n}\sum_{j=۱}^{m} c_{ij}x_{ij} \\ & \sum_{i=۱}^{n} x_{ij} = ۱ \qquad j = ۱,\dots,m \\ & \sum_{j=۱}^{m} x_{ij} = ۱ \qquad i = ۱,\dots,n \\ & x_{ij} \in \{۰,۱\} \qquad i = ۱,\dots,n \quad , \quad j = ۱,\dots,m \end{aligned} \tag{۱.۱}$$

در رابطه (۱.۱) محدودیت اول نشان می‌دهد که شغل jام باید تنها به یک فرد اختصاص یابد و محدودیت دوم تضمین می‌کند که هر فرد تنها یک شغل داشته باشد.

مساله تخصیص در بسیاری زمینه‌ها کاربرد دارد از جمله تخصیص کارمندان به وظایف، ماشین‌ها به کارها و غیره. همچنین حالت‌هایی وجود دارد که مساله تخصیص به‌عنوان زیر مساله در روش‌هایی برای حل مسایل ترکیبی مختلط گوناگون استفاده می‌شود.

از کارهای انجام شده در زمینه مساله زمان‌بندی امتحانات می‌توان به موارد متعدد اشاره نمود. از جمله مقالاتی که در آن از الگوریتم جست‌وجوی ممنوع برای حل مساله جدول زمان‌بندی امتحانات استفاده شده است، مقاله گاسپرو [۱] و اسکارف[۲] [۵]است که با جست‌وجوی ممنوع جواب را برای ۱۳ مثال واقعی تولید کرده‌اند. داده ورودی برای جست‌وجو گرافی است که تداخل امتحانات دانشجویان را نشان می‌دهد، وزن هر گره تعداد دانشجویانی که در امتحان نام نویسی کرده را نشان می‌دهد و وزن یال‌ها تعداد دانشجویان مشترک در هر دو امتحان را مشخص می‌کند. فضای جست‌وجو شامل رنگ‌آمیزی شدنی و نشدنی گراف است. در مقاله [۵] هدف این است که زمان اجرای الگوریتم را کاهش دهند، جست‌وجوی ممنوع با یک جواب اولیه که با الگوریتم حریصانه تولید شده شروع می‌شود.

آلوارز و همکاران [۳] [۱۴] از جمله کسانی هستند که در این زمینه کار کرده‌اند. آن‌ها الگوریتم جدیدی برای مساله زمان‌بندی امتحانات بیان کرده و برای دانشگاه اسپانیا بکار برده‌اند. ترکیبی از ابتکارات براساس جست‌وجوی ممنوع می‌باشد، ابتدا یک جواب که دانشجو دو امتحان همزمان نداشته باشد یافته و سپس با قرار دادن فاصله بین بازه امتحانات جواب را بهبود داده‌اند. نتایج حاصل از مثال واقعی بیانگر کارایی این الگوریتم است و ثابت شده است که این کارایی ناشی از جست‌وجوی ممنوع به‌کار

[۱] Gaspero
[۲] Schaerf
[۳] Alvarez

رفته در آن می‌باشد.
داسکالاکی و همکاران [۷]، یک مدل برنامه‌ریزی عدد صحیح را برای مساله جدول زمان‌بندی دروس دانشگاهی بیان می‌کنند. این مدل شامل محدودیت‌هایی می‌باشد که در اکثر مراکز آموزشی مورد نیاز است. آن‌ها میزان برآورده نشدن محدودیت‌های مورد نظر را مینیمم می‌کنند. بدین صورت که برای نقض شدن محدودیت‌ها ضرایب جریمه در نظر گرفته‌اند و هدف مینیمم کردن این ضرایب می‌باشد. مدل مربوطه با یک نرم‌افزار $CPLEX۵/۱$ برای یک دپارتمان نسبتا بزرگ حل شده است.
همچنین بلیگیانیس و همکاران [۱] برای طراحی جداول زمان‌بندی دبیرستان‌ها درکشور یونان، الگوریتمی بر مبنای محاسبات تکاملی پیشنهاد و پیاده‌سازی کرده‌اند. نتایج عملی کار ایشان روی تعداد زیادی از دبیرستان‌های یونان و در مقایسه با چند الگوریتم دیگر مطلوب گزارش شده است. به‌طور مشابه، پیلای و بن‌ژاف [۱۳] نیز در تحقیق خود از یک الگوریتم ژنتیک مطلع و هوشمند برای طراحی جداول زمان‌بندی امتحانات سود برده‌اند. دیدگاه آنان یک دیدگاه دو مرحله‌ای بوده است که در مرحله اول محدودیت‌های سخت و در مرحله دوم محدودیت‌های نرم به چالش کشیده می‌شوند. در این تحقیق به‌طور خاص مزایای الگوریتم ژنتیک آشکار شده است و نتایج امیدوار کننده‌ای از آن گزارش شده است.
کارتر [۴] [۳] در روش پیشنهادی خود مساله را به بخش‌هایی تقسیم کرده و از یک الگوریتم حریصانه برای تخصیص بازه‌های زمانی و از یک الگوریتم مبتنی بر تابع لاگرانژ، برای تخصیص کلاس‌های درس استفاده می‌کند.
ناجی عظیمی [۱۲] چهار روش ابتکاری ژنتیک، جست‌وجوی ممنوع، کلونی مورچگان و شبیه‌سازی تبریدی و سه روش ترکیبی از آن‌ها را برای حل مساله جدول زمان‌بندی امتحانات استفاده کرده است.
کوت و همکاران [۴] الگوریتم ارزیابی چند هدفه را برای مساله زمان‌بندی امتحانات بکار برده‌اند.
میرحسنی [۱۱] نشان داد برنامه‌ریزی عدد صحیح را می‌توان برای اصلاح برنامه‌ریزی امتحانات و توسعه آن استفاده کرد.

۳.۱ توصیف مساله

مساله جدول زمان‌بندی امتحانات شامل تخصیص یک مجموعه از امتحانات به تعداد محدودی بازه زمانی است که باید در مجموعه‌ای از محدودیت‌ها صدق کند. این محدودیت‌ها بر دو نوع هستند که در زیر بیان می‌کنیم:

- **محدودیت سخت**
  محدودیتی است که نمی‌تواند تحت هیچ شرایطی نقض شود و حتما باید رعایت شود. به‌عنوان مثال یک شخص نمی‌تواند به دو رخ داد در زمان یکسان رسیدگی کند. جواب‌هایی که این محدودیت‌ها را نقض نمی‌کنند جواب‌های امکانپذیر یا شدنی گفته می‌شوند.

- **محدودیت نرم**

[۴] Carter

محدودیت‌هایی که لزوما نباید رعایت شوند اما نقض آن‌ها باید مینیمم شود محدودیت‌های نرم هستند. در حالت‌هایی در جهان واقعی نمی‌توان جوابی پیدا کرد که در همه شرایط محدودیت‌های نرم صدق کند.

محدودیت‌های سخت و نرم که در مقالات مختلف استفاده شده در زیر آورده شده است. محدودیت اول به‌عنوان محدودیت سخت است و سه محدودیت دیگر محدودیت‌های نرم می‌باشند:

- دو امتحان دانشجو همزمان نباشد.
- دو امتحان دانشجو در یک روز نباشد.
- دو امتحان دانشجو در دو روز متوالی نباشد.
- تعداد صندلی موجود در بازه زمانی نباید از تعداد دانشجویانی که در آن بازه امتحان دارند کمتر باشد.

مساله زمان‌بندی امتحانات شامل هر دو محدودیت سخت و نرم است، اگر به یک زمان‌بندی شدنی برسیم به این معنی است که همه محدودیت‌های سخت رعایت شده است. همچنین رعایت محدودیت‌های نرم کیفیت جدول زمان‌بندی امتحانات را بهبود می‌دهد. محدودیت نرم در مساله زمان‌بندی امتحانات میزان زمان مطالعه هر دانشجو بین امتحانات را اندازه‌گیری می‌کند. جدول‌زمان‌بندی امتحانات باید ماکزیمم وقت آزاد برای مطالعه امتحانات پشت‌سر هم را پیشنهاد کند.
ویژگی‌هایی که باید جدول زمان‌بندی امتحانات داشته باشد به‌صورت زیر بیان شده است:

- برای هر درس فقط یک بازه امتحانی وجود داشته باشد.
- دانشجو در هر روز باید یک امتحان داشته باشد و نمی‌تواند امتحانات متوالی داشته باشد.
- تعداد بازه‌ها در جدول زمان‌بندی امتحانات می‌تواند تغییر کند.
- در هر اتاق بیش از یک امتحان وجود دارد.

در مساله زمان‌بندی امتحانات هدف تخصیص یک دسته بازه زمانی به مجموعه‌ای از امتحانات دروس می‌باشد به‌طوری که یک مجموعه از محدودیت‌ها رعایت شوند. مجموعه امتحانات، مجموعه بازه‌های زمانی، مجموعه دانشجویان و مجموعه نام نویسی دانشجویان برای امتحانات را در نظر می‌گیریم. مساله، تخصیص امتحانات به بازه‌های زمانی است که باید در مجموعه محدودیت‌ها صدق کند. محدودیت‌های در نظر گرفته شده در [۵] به‌صورت زیر است:
دو ماتریس در [۵] تعریف شده که به صورت زیر هستند:
یک ماتریس نام‌نویسی $C_{n\times q}$ وجود دارد که درایه‌هایش نشان دهنده این است که دانشجو در کدام امتحان حضور دارد.

$$c_{ij} = \begin{cases} ۱ & \text{اگر دانشجوی } s_j \text{ در امتحان } e_i \text{ حضور داشته باشد} \\ ۰ & \text{در غیراینصورت} \end{cases}$$

همچنین تخصیص با ماتریس باینری $Y_{n\times p}$ نشان داده می‌شود که درایه‌هایش نشان دهنده این است که امتحان e_i در بازه K ام می‌باشد.

$$y_{ik} = \begin{cases} ۱ & \text{اگر امتحان } e_i \text{ به بازه } K \text{ ام تخصیص داده شود} \\ ۰ & \text{در غیراینصورت} \end{cases}$$

آرایه L_p تعداد صندلی‌های موجود رانشان می‌دهد. برای هر بازه زمانی k، مقدار l_k کران بالای تعداد کل دانشجویانی است که در بازه زمانی k امتحان دارند. فرمول ریاضی متناظر با محدودیت‌ها به صورت زیر است:

$$\begin{aligned} \sum_{k=۱}^{p} y_{ik} &= ۱ \qquad (i = ۱, \ldots, n) \\ \sum_{h=۱}^{q} y_{ik} y_{jk} c_{ih} c_{jh} &\le ۱ \qquad (k = ۱, \ldots, p); (i, j = ۱, \ldots, n; i \neq j) \\ \sum_{i=۱}^{n} \sum_{h=۱}^{q} c_{ih} y_{ik} &\le l_k \qquad (k = ۱, \ldots, p) \\ y_{ik} &\in \{۰, ۱\} \qquad (i = ۱, \ldots, n); (k = ۱, \ldots, p) \end{aligned} \tag{۲.۱}$$

محدودیت اول تضمین می‌کند که هر درس تنها به یک بازه زمانی تخصیص داده شود، محدودیت دوم نشان می‌دهد که دانشجو نمی‌تواند دو امتحان همزمان داشته باشد و محدودیت سوم تعداد کل صندلی‌های موجود در آن بازه زمانی را تعیین می‌کند. در ادامه نسخه پیشنهادی توسط کارتر و همکاران [۲] را بررسی می‌کنیم که اساس آن بر تداخل مرتبه اول و تداخل مرتبه دوم استوار است .
تداخل مرتبه اول وقتی حاصل می‌شود که یک دانشجو دو امتحانش در بازه زمانی یکسان برنامه‌ریزی شود. در حالی‌که تداخل مرتبه دوم این است که دو امتحان دانشجو در بازه‌های زمانی نزدیک به یکدیگر باشد. تداخل مرتبه اول به‌عنوان محدودیت سخت و تداخل مرتبه دوم به‌عنوان محدودیت نرم مدل‌بندی می‌شود. بنابراین محدودیت‌ها به‌صورت زیر است:

- دانشجو نباید دو امتحانش در یک بازه زمانی یکسان باشد.
- دو امتحان یک دانشجو در بازه‌های زمانی نزدیک به یکدیگر نباشد.

رعایت محدودیت اول برای شدنی بودن مساله ضروری می‌باشد و رعایت محدودیت دوم میزان مطلوبیت جدول حاصل را بیان می‌کند. محدودیت‌های سخت و نرم با قرار دادن ضریب هزینه در تابع هدف مدل می‌شوند. مدل ریاضی تابع هدف را رابطه (۳.۱) نشان می‌دهد. با توجه به بازه زمانی t_i که برای امتحان i در نظر می‌گیریم مقدار تابع هدف را به‌صورت زیر محاسبه می‌کنیم:

$$F = \frac{(f_۱ + f_۲)}{M} \tag{۳.۱}$$

که در آن

$$f_۱ = \sum_{i=۱}^{N-۱} \sum_{j=i+۱}^{N} C_{ij} W(i,j) \quad \text{که} \quad W(i,j) = \begin{cases} ۲^{۵-t_i-t_j} & ۱ \le |t_i - t_j| \le ۵ \\ ۰ & \text{در غیر اینصورت} \end{cases}$$

$$f_۲ = \sum_{i=۱}^{N-۱} \sum_{j=i+۱}^{N} C_{ij} W'(i,j) \quad \text{که} \quad W'(i,j) = \begin{cases} ۱۰۰۰ & t_i = t_j \\ ۰ & \text{در غیر اینصورت} \end{cases}$$

نمادهای مورد استفاده در جدول (۱.۱) آمده است.

جدول ۱.۱: نمادها

تعداد دانشجویان	M
تعداد امتحانات	N
تعداد دانشجویان مشترک در هر دو امتحان i و j	$C(i,j)$
بازه امتحان i	t_i

در رابطه (۳.۱) $W(i,j)$ هزینه نسبت داده شده به محدودیت نرم است که در آن i,j امتحان i ام و j ام می‌باشند. کاهش هزینه به‌طور لگاریتمی از ۱۶ به ۱ است به‌این‌صورت که اگر فاصله بازه دو امتحان ۱ باشد هزینه تخصیص یافته به آن ۱۶ می‌شود و اگر فاصله بازه دو امتحان ۲ باشد هزینه ۸ و به همین ترتیب اگر فاصله دو بازه ۵ باشد هزینه ۱ می‌شود.

در رابطه (۳.۱) $W'(i,j)$ هزینه نسبت داده شده به محدودیت سخت است که مقدار آن ۱۰۰۰ است.

c_{ij} تعداد دانشجویان مشترک بین هر دو امتحان i و j می‌باشد. درایه‌های ماتریس تداخل c_{ij}ها می‌باشند که به‌طور تصادفی تولید می‌شوند. ماتریس تداخل ماتریس مربعی متقارن است که تعداد سطرها و ستون‌ها برابر تعداد امتحانات است.

در تابع برازش، هزینه محدودیت‌ها در تعداد دانشجویان مشترک در هر دو امتحان ضرب می‌شود. اگر تعداد دانشجویان مشترک در هر دو امتحان زیاد باشد و دو امتحان در بازه زمانی یکسان باشند در این‌صورت مقدار تابع برازش مقدار زیادی است. زیرا هزینه ۱۰۰۰ در تعداد دانشجویان مشترک در هر دو امتحان ضرب می‌شود. ولی هدف این است که هزینه را مینیمم کند. بنابراین از این‌گونه جواب‌ها صرف‌نظر می‌کنیم.

تابع F در بالا بیان شده در هر دو الگوریتم ژنتیک و جست‌وجوی ممنوع به ترتیب به‌عنوان تابع برازندگی و تابع هدف مورد استفاده قرار گرفته است، که در فصل‌های بعد به آن‌ها می‌پردازیم.

۴.۱ بهبود جدول زمان‌بندی امتحانات با استفاده از برنامه‌ریزی عدد صحیح

در مقاله [۱۱] یک برنامه ریزی برای امتحانات به‌منظور فراهم کردن زمان مطالعه بهتر بین امتحانات برای دانشجویان ارایه شده است و نشان داده شده چطور می‌توان برنامه‌ریزی عدد صحیح را برای این مسائل استفاده کرد. محدودیت‌هایی برای کیفیت جدول زمان‌بندی امتحان و برآورده کردن تمام نیازهایی که در بسیاری موسسات آموزشی وجود دارد آورده شده است. مدل مساله با توجه به محدودیت‌ها نوشته شده است و با استفاده از نرم افزار (MPL) برکامپیوتر پنتیوم سه ۷۰۰ MHz حل شده است. در این بخش به بیان این مدل می‌پردازیم.

۱.۴.۱ توصیف مساله

مجموعه دروس و یک امتحان برای هر درس را در نظر می‌گیریم. گروهی از امتحانات وجود دارد که هر امتحان از گروه شامل دانشجویانی است که در تعدادی امتحان نام نویسی کرده‌اند. در [۱۱] جدول زمان‌بندی امتحان شدنی برای تمام دروس برطبق جدول نیمسال تحصیلی که تمام دانشجویان باید آن را در نظر بگیرند، تنظیم می‌شود. دانشجویانی که فارغ‌التحصیل می‌شوند و نیمسال تحصیلی پایانی آن‌ها است باید در تمام دروس باقیمانده خود نام‌نویسی کنند. تحت این شرایط جدول زمان‌بندی امتحان نیاز به اصلاح دارد. بنابراین مدلی برای تعیین جدول زمان‌بندی امتحان طراحی شده که جدول زمان‌بندی امتحان که بهترین زمان مطالعه بین امتحانات را به دانشجویان پیشنهاد می‌دهد را فراهم می‌کند.

۲.۴.۱ مدل

در [۱۱] هدف اصلی تنظیم جدول زمان‌بندی امتحان مورد نیاز است. برای بهتر شدن جدول زمان‌بندی امتحانات، سه نوع تداخل امتحانی ممکن برای دانشجو مورد بررسی قرار می‌گیرد که این سه نوع تداخل به‌صورت زیر هستند:

۱. تداخل نوع اول : دو امتحان دانشجو در بازه زمانی یکسان نباشد.

۲. تداخل نوع دوم : دو امتحان دانشجو در یک روز نباشد.

۳. تداخل نوع سوم : دو امتحان دانشجو در دو روز متوالی نباشد.

معمولا دانشجویان تداخل نوع سوم را به تداخل نوع دوم و تداخل نوع دوم را به تداخل نوع اول ترجیح می‌دهند. تداخل نوع اول محدودیت سخت است و تداخل‌های نوع دوم و سوم به‌عنوان محدودیت‌های نرم در نظر گرفته شده‌اند. تابع هدف برای مینیمم کردن تعداد تداخل‌های مختلف نوع اول، دوم و سوم در نظر گرفته شده است.
برای مدل کردن مساله مجموعه‌های زیر را در نظر بگیرید:

- $S = \{۱, \ldots, s\}$ مجموعه دانشجویان با درس‌های مختلف.
- $C = \{۱, \ldots, c\}$ مجموعه دروس.
- $D = \{۱, \ldots, d\}$ مجموعه روزها.
- $T = \{۱, \ldots, t\}$ مجموعه بازه‌های زمانی یا جلسه در هر روز.
- Q_s تعداد دانشجویان با مجموعه دروس یکسان.

همچنین فرض کنید پارامترهای زیر را داریم:

- GS_c گروهی از دانشجویان که در درس c نام‌نویسی کرده‌اند.
- GC_s گروهی از دروس که دانشجویان نام‌نویسی کرده‌اند.

متغیرهای مساله به‌صورت زیر تعریف می‌شوند:

- $$X_{cdt} = \begin{cases} ۱ & \text{اگر امتحان درس c در روز d و بازه زمانی t باشد} \\ ۰ & \text{در غیراینصورت} \end{cases}$$
- $$Y_{sdt} = \begin{cases} ۱ & \text{اگر دانشجوی s در روز d و بازه زمانی t امتحان داشته باشد} \\ ۰ & \text{در غیر اینصورت} \end{cases}$$
- U_{sdt} تعداد دروس امتحانی مجازی که دانشجوی s در روز d و بازه زمانی t دارد. به‌عنوان مثال اگر $U_{sdt} = ۲$ باشد در این‌صورت امتحان دو درس دانشجوی s در زمان یکسان است.
- V_{sd} تعداد دروس امتحانی مجازی که دانشجوی s در روز d می‌تواند داشته باشد.
- W_{sd} تعداد دروس امتحانی مجاز که دانشجوی s در روز d و روز بعد آن می‌تواند داشته باشد.

مولفه‌های تابع هدف برای مساله عبارتند از:

- $\sum Q_s U_{sdt}$ تعداد دانشجویانی که تداخل نوع اول را در امتحانات دارند.
- $\sum Q_s V_{sd}$ تعداد دانشجویانی که تداخل نوع دوم را در امتحانات دارند.
- $\sum Q_s W_{sd}$ تعداد دانشجویانی که تداخل نوع سوم را در امتحانات دارند.

بنابراین تابع هدف به‌صورت زیر خواهد بود:

$$MinZ = K \sum_{s,d,t} Q_s U_{sdt} + L \sum_{s,d} Q_s V_{sd} + M \sum_{s,d} Q_s W_{s,d}.$$

در این تابع هدف تداخل‌های نوع اول، دوم و سوم به ترتیب با مقدارهای ثابت K، L و M که در آن $M \ll L \ll K$ فرمول‌بندی می‌شوند. این مقادیر ثابت جریمه‌ای است که برای تداخل‌ها در نظر گرفته شده است. چون مساله مینیمم سازی است می‌خواهیم که تابع هدف کمترین مقدار را داشته باشد، برای این منظور باید تک‌تک مولفه‌ها در تابع هدف کمترین مقدار را داشته باشند. جریمه K در مولفه $\sum_{s,d,t} Q_s U_{sdt}$ ضرب می‌شود و مقدار بزرگتری نسبت به جریمه های دیگر برای آن در نظر گرفته‌ایم. با توجه به این که تداخل نوع اول محدودیت سخت است رعایت آن الزامی است و جریمه بزرگی برای آن قرار داده شده است. برای اینکه تابع هدف کمترین مقدار را داشته باشد، باید تعداد دانشجویانی که تداخل نوع اول را در امتحانات دارند کوچک باشد زیرا K عدد بزرگی است اگر $\sum_{s,d,t} Q_s U_{sdt}$ مقدار بزرگی باشد مقدار تابع هدف مینیمم نمی‌شود. جریمه‌های K، L و M میزان اولویت رعایت محدودیت‌ها را مشخص می‌کند، هر چقدر جریمه کوچکتر باشد بدین معنی است که اولویت برای رعایت محدودیت مربوط پایین‌تر است. تابع هدف بهتر می‌شود وقتی‌که U_{sdt} ، V_{sd} و W_{sd} کوچکتر باشند و با توجه به محدودیت‌ها مقادیر زیادی نمی‌گیرند.

محدودیت‌های مساله به‌صورت زیر هستند:

$$\sum_{d,t} X_{cdt} = ۱ \qquad \forall c.$$

این محدودیت‌ها تضمین می‌کند که امتحان هر درس تنها به یک بازه زمانی تخصیص داده شود.

$$X_{cdt} \leq Y_{sdt} \qquad \forall c, d, t \qquad \forall s \in GS_c.$$

در محدودیت فوق اگر $X_{cdt} = ۱$ پس به‌ازای هر دانشجو که درس c را دارد $Y_{sdt} = ۱$ و در غیر این‌صورت یعنی اگر امتحان c در روز d و بازه زمانی t نباشد، $Y_{sdt} = ۰$ یعنی دانشجوی s که امتحان c را داشته، در روز d و بازه زمانی t امتحانی ندارد. شرایطی که در بالا گفته شد رابطه بین امتحانات و دانشجویان را بیان می‌کند.

$$\sum_{c \in GC_s} X_{cdt} \leq U_{sdt} + ۱ \qquad \forall s, d, t \qquad (۴.۱)$$

رابطه (۴.۱) تعداد دروس مجازی که دانشجوی s در روز d و بازه زمانی t دارد را مشخص می‌کند و این دروس از مجموعه دروسی است که دانشجوی s در آن‌ها نام‌نویسی کرده است. با توجه به این که جریمه بزرگی برای U_{sdt} در تابع هدف در نظر گرفته شده است بهتر است تعداد دروس مجازی که دانشجوی s در روز d و بازه زمانی t دارد کوچک باشد زیرا می‌خواهیم تابع هدف کمترین مقدار را داشته باشد.

محدودیت‌های زیر تعداد دانشجویانی که تداخل نوع دوم را در امتحانات دارند مشخص می‌کند.

$$\sum_{t} Y_{sdt} \leq V_{sd} + ۱ \qquad \forall s, d.$$

محدودیت‌های زیر تعداد دانشجویانی که تداخل نوع سوم را در امتحانات دارند مشخص می‌کند.

$$\sum_{t} Y_{sdt} + \sum_{t} Y_{s,d+۱,t} \leq W_{s,d} + ۱ \qquad \forall s, d.$$

از این رو بطور عملی تعداد تداخل‌های نوع اول هرگز بیشتر از یک نمی‌باشد بنابراین رابطه (۵.۱) را داریم:

$$U_{sdt} \leq ۱ \qquad (۵.۱)$$

بنابراین کران بالا برای این مجموعه متغیرها را بررسی می‌کنیم. همچنین تمام متغیرها نامنفی فرض شده‌اند.

برنامه جدول زمان‌بندی امتحان با مدل بالا با استفاده از نرم افزار (MPL) حل شده است . زمان اجرا بر مجموعه داده‌هایی واقعی از دانشگاه صنعتی شاهرود (که حدود ۳۲۴ امتحان در نیمسال تحصیلی، ۳۰ جلسه و براساس ۴۰۰۰ نام‌نویسی) کمتر از ۲ دقیقه برای مساله است. مقدارهای مختلفی برای این سه پارامتر آزمایش شده و در بعضی حالت‌ها زمان زیادی برای حل مساله صرف شده است.

فصل ۲

الگوریتم ژنتیک

۱.۲ مقدمه

محققان با مشاهده سازگاری، بقا، خود ترمیمی، هدایت، تولیدمثل و دیگر خصوصیات سیستم‌های طبیعی و این نکته که طبیعت مسائل خود را چگونه حل می‌کند به فکر تقلید از روش‌های طبیعی در حل مسائل سخت و طراحی سیستم‌ها افتاده‌اند. ایده اصلی الگوریتم ژنتیک هم بر اساس فکر و بر پایه تئوری تکامل داروین [۱] شکل گرفت.

در کنار روش‌های جستجوی فراابتکاری [۲]، روش‌های جستجوی دیگری نیز وجود دارند که به الگوریتم تکاملی [۳] معروفند. بر اساس نظریه داروین نسل‌هایی که از ویژگی‌ها و خصوصیات برتری نسبت به نسل‌های دیگر برخوردار هستند شانس بیشتری برای بقا و تکثیر خواهند داشت و ویژگی‌ها و خصوصیات برتر آن‌ها به نسل بعدی آن‌ها منتقل خواهد شد. همچنین بخش دوم نظریه داروین بیان می‌کند هنگام تکثیر یک فرزند، به تصادف رویدادهایی اتفاق می‌افتد که موجب تغییر خصوصیات فرزند بعد از جهش می‌شود و در صورتی‌که این تغییر فایده‌ای برای فرزند جدید داشته باشد موجب افزایش احتمال بقای این فرزند خواهد شد. در محاسبات کامپیوتری، براساس این نظریه داروین، روش‌هایی برای مسایل بهینه‌سازی ارایه شده است که همه این روش‌ها از پردازش تکاملی در طبیعت نشات گرفته‌اند. به این روش‌های جستجو، الگوریتم‌های جستجوی تکاملی می‌گوییم. الگوریتم ژنتیک یک الگوریتم تکاملی است. الگوریتم‌های تکاملی مانند الگوریتم‌های فراابتکاری، به شکل تصادفی اما هدایت شده در فضای نمونه مساله حرکت می‌کنند. با توجه به نظریه داروین، پیاده‌سازی یک الگوریتم تکاملی طی مراحل زیر انجام می‌پذیرد [۱۶].

- کدگذاری[۴] و نحوه نمایش،

---

[۱] Darwin

[۲] Meta heuristic

[۳] Evolutionary algorithm

[۴] Encoding

- تولید جمعیت اولیه،
- محاسبه میزان بهینگی هر یک از افراد جمعیت (تابع شایستگی [۵])
- انتخاب
- تولید نسل [۶]
- جهش [۷]

۲.۲ الگوریتم تکاملی

یک الگوریتم تکاملی، الگوریتمی احتمالی بر مبنای جمعیت‌هایی از جواب‌های شدنی است. فرم کلی یک الگوریتم تکاملی به‌صورت زیر است:

۱.

$$۰ \longmapsto t$$

۲. جمعیت $p(۰)$ را به‌صورت تصادفی مقدار دهی اولیه کنید.

۳. تمام افراد جمعیت $p(۰)$ را با استفاده از تابع هدف داده شده f، ارزیابی کنید.

۴. مراحل زیر را تا زمانی‌که به شرط توقف برسید، انجام دهید:

(آ)

$$t + ۱ \longmapsto t.$$

(ب) افرادی را از جمعیت قبلی $p(t-۱)$ انتخاب کنید و برای تولید جمعیت $p(t)$ از آن‌ها استفاده کنید.

(ج) جمعیت $p(t)$ راارزیابی کرده و شرط توقف را بررسی کنید.

۵. بهترین فرد را انتخاب کنید.

معمولا هر یک از متغیرهای تابع هدف از بازه مخصوص خود انتخاب می‌شوند. بنابراین بازه تابع هدف چند بعدی است. الگوریتم‌های مختلفی در مرحله انتخاب و ترکیب وجود دارند که بر مبنای آن‌ها جمعیت جدید از جمعیت قدیم ساخته می‌شود.

---

[۵] Fitness

[۶] Reproduction

[۷] Mutation

۳.۲ الگوریتم ژنتیک

الگوریتم ژنتیک یکی از مهم‌ترین الگوریتم‌های جستجوگر ابتکاری است که از آن برای بهینه‌سازی توابع مختلف استفاده می‌شود. الگوریتم ژنتیک جزیی از محاسبات تکاملی است که خود جزیی از هوش مصنوعی می‌باشد.

۱.۳.۲ کاربردهای الگوریتم ژنتیک

الگوریتم ژنتیک در فرآیند حل مساله احتیاجی به اطلاعات خاص درباره مساله ندارد، فقط مقادیر عددی تابع هدف را لازم دارد. بنابراین زمینه استفاده و اعمال آن محدودیتی ندارد. چند مورد از کاربردهای الگوریتم ژنتیک به‌صورت زیر است:

- بهینه‌سازی توابع
- نحوه توزیع و دستیابی به منابع غذایی در بیولوژی
- برنامه‌ریزی دروس
- زمان‌بندی خطوط تولید
- رنگ‌آمیزی گراف
- حل معادلات غیرخطی سطوح هم پتانسیل در فیزیک
- شبیه‌سازی تغییر عادت‌های رفتاری و تنظیم مدل مهاجرت در علوم اجتماعی

کاربردها در مهندسی

- استخراج و بهینه‌سازی قواعد فازی
- تعیین توپولوژی و آموزش شبکه‌های عصبی
- طراحی مسیر و حل معادلات سینماتیک معکوس
- طراحی بهینه مولکول‌های دارویی
- شبیه‌سازی سیستم ایمنی بدن
- حل مسایل ترکیباتی بهینه‌سازی سخت مانند فروشنده دوره‌گرد

۲.۳.۲ واژگان ژنتیک

در الگوریتم ژنتیک واژه‌هایی استفاده می‌شود که به‌صورت زیر تعریف می‌گردند.

- **ژن**[۸] : ژن‌ها بلوک‌های DNA یا اسید نوکلییک هستند که یک خصیصه (مثل رنگ چشم) را کد می‌کند. آن‌ها عناصر تشکیل دهنده یک کروموزم هستند که معمولا عدد یا کلمه یا حتی دستورات برنامه نویسی هستند.
- **کروموزم**[۹] : یک رشته طولانی به هم پیچیده از ژن‌ها یا همان رشته DNA است که اطلاعات ساختاری و رفتاری موجودات را در خود به صورت کد شده داراست. در حل مسائل به رمز در آمده یا کد شده یک جواب یا قسمتی از جواب را کروموزم گویند.
- **آلل** [۱۰] : به مجموعه‌های ممکن برای یک خصیصه آلل می‌گویند.
- **مکان** : هر ژن در کروموزم موقعیت خاص خودش را دارد به این موقعیت ژن در کروموزم مکان می‌گویند.
- **فنو تایپ** : بعد از تکامل همان خصوصیات فیزیکی فرزند می‌باشد.
- **جمعیت** : مجموعه کروموزم‌ها را جمعیت می‌گویند.

در الگوریتم ژنتیک اندازه جمعیت در تمام مراحل یکسان است و نباید حتما در هر مرحله الگوریتم جواب‌ها بهبود یابد.

۳.۳.۲ فضای جستجو

وقتی به دنبال جواب مساله‌ای هستیم بهترین جواب‌ها را در مجموعه جواب‌های شدنی جستجو می‌کنیم. فضای تمام جواب‌های شدنی فضای جستجو نامیده می‌شود هر نقطه در فضای جستجو یک جواب شدنی را نشان می‌دهد. هر جواب شدنی می‌تواند براساس ارزش و مطلوبیتش برای مساله درجه‌بندی شود. بهترین جواب از بین مجموعه جواب‌های شدنی با یک نقطه در فضای جستجو نشان داده می‌شود.

۴.۳.۲ مشخصات بیولوژیکی

بدن تمام موجودات زنده از سلول تشکیل شده است. در هر سلول مجموعه‌ای از موجودات هم‌شکل به‌نام کروموزم وجود دارند. کروموزم‌ها، رشته‌های DNA می‌باشند که هر یک متشکل از تعدادی ژن هستند. هر ژن یک خصیصه را کد می‌کند. به‌عنوان مثال رنگ چشم می‌تواند یک خصیصه باشد. مجموعه کامل همه کروموزم‌ها ژنوم [۱۱] نامیده می‌شود.

---

[۸] Gene

[۹] Chromosome

[۱۰] Allel

[۱۱] Genome

۴	۳	۲	۶	۸	۷	۵	۱

شکل ۱.۲: نمونه کدگذاری مساله‌ی هشت وزیر

۵.۳.۲ کدگزاری ونحوه نمایش

الگوریتم ژنتیک به‌جای اینکه برروی متغیرها و پارامترهای مساله کار کند، با شکل کد شده آن سروکار دارد. به عنوان مثال در مساله هشت وزیر هدف مشخص کردن چیدمانی از هشت وزیر در صفحه شطرنج است به نحوی که هیچ یک همدیگر را تهدید نکنند، یک روش کدگذاری جواب در این مساله می‌تواند استفاده از یک آرایه هشت عنصری باشد مانند شکل (۱.۲) که هر عنصر از آرایه نشان دهنده‌ی سطری است که وزیر در آن قرار دارد.

با توجه به مقادیر آرایه در می‌یابیم که عدد ۴ وزیر در ستون اول و سطر چهارم، عدد ۳ وزیر در ستون دوم و سطر سوم، عدد ۲ وزیر در ستون سوم و سطر دوم و ...از صفحه شطرنج قرار گرفته است.
بعضی کدگذاری‌های مختلف در الگوریتم ژنتیک به‌صورت زیر است:

کدگذاری دودویی:

این نوع کد گذاری متداول‌ترین نوع کدگذاری است. در این نوع کدگذاری هر کروموزم رشته‌ای از صفر و یک می‌باشد و هر ژن از کروموزم مقدار ۰ یا ۱ می‌گیرد و موجب تشکیل یک رشته دودویی در کروموزم می‌شود. کدگذاری دودویی می‌تواند حالت‌های زیادی را پوشش دهد و به راحتی با عملگرهای بیتی در زبان‌های برنامه‌سازی پیاده‌سازی شود. به‌عنوان مثال، فرض کنید یک تابع سه متغیره داریم و می‌خواهیم مقادیر این سه متغیر را طوری انتخاب کنیم که مقدار این تابع مینیمم شود. فرض کنید هر متغیر یک مقدار اعشاری ۴ بایتی (۳۲ بیتی) باشد. در صورتی که بخواهیم از روش نمایش دودویی برای این سه متغیر استفاده کنیم، باید یک رشته دودویی به طول ۹۶ بیت تشکیل دهیم.
کدگذاری بالا دارای خصوصیات زیر است :

۱. هر دقتی از متغیرها را می‌توان با تغییر اندازه طول رشته‌ها بدست آورد.

۲. متغیرهای مختلف می‌توانند دارای دقت‌های متفاوتی باشند.

۳. متغیرها می‌توانند مثبت یا منفی باشند.

۴. کران‌ها بطور دقیق قابل دسترسی هستند.

کدگذاری حقیقی:

زمانی که متغیرهای ما مقادیر پیوسته دارند، مناسب‌ترین روش نمایش یک جواب کاندید، استفاده از یک رشته اعداد حقیقی است. در زمان پیاده‌سازی الگوریتم در کامپیوتر، دقت این اعداد حقیقی به چند رقم اعشار محدود می‌شود؛ به این دلیل، این کدگذاری را کدگذاری اعشاری نیز می‌نامند. کدگذاری برای

A : ۱ ۳ ۵ ۲ ۴

B : ۱ ۲ ۵ ۴ ۳

شکل ۲.۲: کدگذاری جایگشتی

یک جواب با k ژن، بردار $< x_1, \ldots, x_k >$ است که در آن $x_i \in R$ [۸].
این روش کد کردن جواب دارای مزایای فراوانی است، مثلا در الگوریتم ژنتیک چون انتخاب براساس تابع برازندگی است هر عملگر انتخاب که برروی کدگذاری دودویی کار می‌کند می‌تواند برای کدگذاری حقیقی هم استفاده شود. مشکل استفاده از این نوع کدگذار در تعریف عملگرهای جهش و ادغام است. مساله اصلی در اینجا تعریف مناسب این دو عملگر است.

کدگذاری جایگشتی:

این نوع کدگذاری عموما برای حل مسایل بهینه‌سازی ترکیباتی مثل زمان‌بندی خودروها، مساله فروشنده دوره‌گرد و ...بکار می‌رود. در اینجا هر کروموزم شامل اعدادی است که معمولا ترتیب خاصی را بیان می‌کنند. به‌عنوان مثال در فروشنده دوره‌گرد معمولا ترتیب شهرها به صورت یک کروموزم کد می‌شود. اگر فروشنده‌دوره‌گرد باید از ۵ شهر عبور کند دو جواب با کدگذاری جایگشتی می‌تواند به‌صورت A و B باشد که در شکل (۲.۲) آمده است. به‌عنوان مثال در A فروشنده ابتدا از شهر ۱ عبور می‌کند سپس به شهر ۳ رفته به همین ترتیب از شهرهای ۵ و ۲ گذشته و در نهایت به شهر ۴ رسیده است.

به‌خاطر ساختار خاص کروموزم‌ها در این نوع کدگذاری لازم به تعریف عملگرهای جهش و ادغام مربوط به این کدگذاری است.

کدگذاری درختی:

کدگذاری درختی در الگوریتم ژنتیک به‌کار می‌رود. در کدگذاری درختی هر کروموزم یک درخت از اشیایی نظیر توابع یا دستورها در زبان برنامه‌نویسی می‌باشند. مثالی از کدگذاری درختی در زیر آمده است:

کروموزم	$(+x(/۵ * y))$	$(x * y+)$
کد باز شده	$x + ۵ * y$	$x + y$

۶.۳.۲ تولید جمعیت اولیه

الگوریتم ژنتیک کار خود را با تولید جمعیت اولیه‌ای از کروموزم‌ها آغاز می‌کند و سپس در یک حلقه به‌طور مکرر تعدادی از کروموزم‌های برتر نسل فعلی را انتخاب کرده و سپس نسل جدیدی از این کروموزم‌ها را تولید می‌کند. همانطور که دیدیم هر کروموزم نشان دهنده‌ی یک عنصر از فضای نمونه مساله است بنابراین منظور از تولید جمعیت اولیه، تولید تعدادی جواب برای مساله خواهد بود. تولید جواب‌های اولیه نیز به دو صورت ابتکاری و تصادفی انجام می‌پذیرد.
به‌عنوان مثال در مساله هشت وزیر می‌توانیم به شکل تصادفی وزیرها را در خانه‌های صفحه شطرنج قرار

دهیم و یا در مساله فروشنده دوره‌گرد می‌توانیم ترتیب ملاقات شهرها را هم به شکل تصادفی هم به شکل ابتکاری انتخاب کنیم.
نکته مهم هنگام تولید جمعیت اولیه آن است که هنگام تولید جواب برای مساله (تصادفی یا ابتکاری) جواب تولید شده باید به شکل کروموزمی که در مرحله قبل تعیین کردیم، کدگذاری شده و به مجموعه جمعیت اضافه شود. بنابراین باید از تولید جواب‌هایی که موجب نقض نحوه کدگذاری تعیین شده برای کروموزم شود اجتناب کنیم. تعداد کروموزم‌هایی که برای جمعیت اولیه تولید می‌کنیم به شکل تجربی حاصل می‌شود. اگرچه افزایش اندازه جمعیت موجب افزایش قدرت محاسباتی الگوریتم می‌شود، اما به‌طور چشم‌گیری مدت زمان اجرای الگوریتم را افزایش می‌دهد همچنین در برخی موارد بزرگی جمعیت موجب توقف زودرس الگوریتم می‌شود که منجر به تولید جواب‌های بهینه محلی خواهد شد، در حالی که هدف پیدا کردن بهینه سراسری است. در مقابل، جمعیت با اندازه کوچک نیز توانایی چندانی به خصوص در مسائل پیچیده برای حل مساله نخواهد داشت.

۷.۳.۲ تابع برازندگی

الگوریتم ژنتیک نیز مانند الگوریتم‌های فراابتکاری در هر بار تکرار به‌تدریج خود را به نقطه‌ای که بهینه سراسری مساله است نزدیک می‌کند. همچنین گذر از نمونه‌ای به نمونه دیگر براساس هزینه هر نمونه انجام می‌پذیرد. در روش‌های جستجوی فراابتکاری تخمین هزینه هر نمونه با استفاده از تابع هدف انجام می‌گیرد. در الگوریتم ژنتیک تابع برازندگی معمولا تابع هدف یا تابعی یکنوا از تابع هدف قرار می‌گیرد ولی در مواردی هم با گذاشتن جریمه یا تشویق در تابع برازندگی جواب‌ها را به سمت خاصی از فضای جستجو رهنمون می‌کند.
برای حل مسائل مینیمم‌سازی در الگوریتم ژنتیک می‌توانیم با تبدیل آن‌ها به فرم ماکزیمم سازی تبدیل کنیم. براساس نظریه داروین نسل‌هایی که از ویژگی‌ها و خصوصیات برتری نسبت به نسل‌های دیگر برخوردارند شانس بیشتری برای بقا و تکثیر خواهند داشت. در واقع تابع برازندگی برتری میان یک نمونه نسبت به نمونه دیگر را تخمین می‌زند و موجب می‌شود که هنگام تکثیر نمونه‌هایی به نسل بعد منتقل شوند، که شایستگی بیشتری داشته باشند.

۸.۳.۲ اعمال ژنتیک

اعمال ژنتیک برای انتخاب والدین، ایجاد فرزندان و گزینش نسل جدید می‌باشند. انتخاب و معرفی عملگرهای ژنتیک برای پیاده‌سازی بر روی مساله خاص یکی از مهمترین مباحث مربوط به الگوریتم ژنتیک است. انتخاب صحیح عملگر باعث قدرت و سرعت الگوریتم ژنتیک می‌شود.

عملگر انتخاب

وظیفه اصلی عملگر انتخاب، هدایت الگوریتم به نواحی امید بخش فضای جواب است. چون در الگوریتم ژنتیک اندازه جمعیت ثابت است، عملگر انتخاب با

- شناسایی جواب‌های خوب
- ساختن کپی از آن‌ها
- حذف جواب‌های بد

علاوه بر حفظ اندازه جمعیت باعث تکثیر و افزوده شدن جواب‌های خوب در جمعیت می‌شود. روش‌های مختلفی برای انتخاب کروموزم‌ها و ادغام آن‌ها وجوددارند که مهمترین این روش‌ها عبارتند از:

- روش تصادفی [۱۲]
- روش تورنمنت [۱۳]
- روش چرخ‌رولت [۱۴]
- روش رتبه‌ای [۱۵]
- روش بولتزمن [۱۶]
- روش حالت‌پایدار [۱۷]

روش تصادفی:
ساده‌ترین روش انتخاب کروموزم‌ها این است که بدون در نظر گرفتن میزان بهینگی کروموزم‌ها، آن‌ها را به تصادف انتخاب کرده و به حوضچه ژنتیک منتقل کنیم. پیاده‌سازی این روش بسیار ساده بوده ولی در مقابل از کارایی بسیار کمی برخوردار است و همانطور که می‌بینیم با اصول کلی انتخاب کروموزم‌ها که در ابتدای این بخش مطرح کردیم در تناقض است. اما با این حال می‌توان در برخی از مسائل، بخشی از عمل انتخاب را به شکل تصادفی انجام داد، چرا که در این روش کروموزم خوب و بد هر دو به یک اندازه شانس انتخاب شدن دارند. باید توجه کنیم که شرکت دادن یک کروموزم بد نیز در برخی موارد می‌تواند موجب تولید کروموزم بهتری می‌شود.

روش تورنمنت :
در روش انتخاب تورنمنت هر بار که می‌خواهیم کروموزمی را از جمعیت فعلی انتخاب کنیم، چند کروموزم از جمعیت را به تصادف انتخاب کرده و از میان آن‌ها کروموزم بهتر را به حوضچه ژنتیک منتقل می‌کنیم و این عمل تا پر شدن کامل حوضچه ژنتیک ادامه می‌یابد. انتخاب تورنمنت نسبت به سایر عملگرهای انتخاب دارای همگرایی بهتر و پیچیدگی کمتر می‌باشد.

روش چرخ رولت :

[۱۲] Random
[۱۳] Tournament
[۱۴] Rolette wheel
[۱۵] Rank
[۱۶] Boltzman
[۱۷] Steady state

اصول کلی انتخاب در این روش به این صورت است که در گام اول به هر یک از کروموزم‌های جمعیت فعلی احتمالی را نسبت می‌دهیم که شانس انتخاب آن کروموزم برای انتقال به حوضچه ژنتیک را نشان می‌دهد.
تابع توزیع احتمال نسبی براساس میزان برازندگی هر کروموزم و با توجه به مجموع مقدار تابع هدف همه کروموزم‌ها، احتمال را به هر یک از آن‌ها نسبت می‌دهد. یعنی به کروموزم i احتمال p_i را به صورت زیر نسبت می‌دهیم:

$$p_i = \frac{F_i}{\Sigma_{j=1}^{n} F_j}$$

که در آن n اندازه جمعیت، F_i مقدار تابع برازندگی کروموزم i ام و $\Sigma_{j=1}^{n} F_j$ مجموع مقادیر تابع شایستگی کل کروموزم‌ها است. توجه کنید مجموع احتمال همه کروموزم‌ها باید برابر۱ شود.
در گام دوم یک عدد تصادفی بین صفر و یک تولید می‌شود و توزیع تجمعی احتمال هر یک از کروموزم‌ها محاسبه می‌شود. به این صورت که مقدار احتمال هر کروموزم با مقدار احتمال کروموزم‌های قبل از آن جمع می‌شود. سپس کروموزم‌ها از اول بررسی شده و اولین کروموزمی که توزیع تجمعی احتمال آن بیشتر یا مساوی عدد تولید شده باشد، برای انتقال به حوضچه انتخاب می‌شود. در بعضی الگوریتم‌ها روش چرخ‌رولت به صورت زیر است:

۱. مقدار تابع شایستگی همه کروموزم‌ها با هم جمع می‌شود و حاصل S نامیده می‌شود.

۲. عدد تصادفی r در بازه $(0, S)$ در نظر گرفته می‌شود .

۳. اولین کروموزمی که مجموع مقادیر تابع بهینگی آن کروموزم با کروموزم‌های قبل از آن از r بیشتر شود انتخاب می‌شود. البته گام اول برای هر جمعیت فقط یک بار اجرا می‌شود.

روش رتبه‌ای:
در روش چرخ رولت اگر مقادیر تابع شایستگی خیلی با هم تفاوت داشته باشند، دچار مشکل خواهیم شد. مثلا اگر بهترین مقدار تابع شایستگی یک کروموزم ۹۰ درصد باشد، در حدود ۹۰ درصد از شکل (چرخ رولت) را اشغال خواهد کرد. بنابراین سایر کروموزم‌ها شانس کمتری برای انتخاب شدن خواهند داشت. در این روش ابتدا کروموزم‌ها را مرتب کرده و با توجه به میزان بهینگی آن‌ها رتبه‌ای را به هر یک از کروموزها نسبت می‌دهیم. به عنوان مثال ممکن است بدترین کروموزم رتبه صفر و بهترین کروموزم رتبه $n-1$ به خود بگیرد که n نشان دهنده اندازه جمعیت فعلی است.
نحوه رتبه‌بندی کروموزم‌ها نیز توسط طراح الگوریتم تعیین می‌شود. پس از رتبه‌بندی کروموزم‌ها احتمال انتخاب هر یک از آن‌ها را محاسبه می‌کنیم. روش محاسبه احتمال انتخاب کروموزم را نیز طراح الگوریتم ژنتیک تعیین می‌کند. نکته مهم در طراحی روش محاسبه‌ی احتمال این است که انتساب احتمال به کروموزها باید براساس رتبه کروموزم‌ها انجام گرفته و محاسبه‌ی احتمال به‌شکلی انجام شود که حتی در صورت زیاد بودن واریانس بهینگی جمعیت فعلی، تعادلی در احتمال انتخاب کروموزم‌های خوب و بد ایجاد کند. تاثیر استفاده از تابع احتمال رتبه‌ای و نسبی بر روی داده‌هایی است که مقدار شایستگی بهترین و بدترین کروموزم با هم اختلاف زیادی دارند.

هنگامی که واریانس شایستگی جمعیت زیاد باشد، استفاده از تابع احتمال نسبی موجب خواهد شد که حوضچه ژنتیک تنها از کروموزم‌های بهتر تشکیل شود، ولی در مقابل استفاده از احتمال رتبه‌ای موجب می‌شود که تعادلی نسبی بین کروموزم‌ها برای انتخاب شدن به وجود آید. پس از آن که احتمال انتخاب کروموزم‌ها تعیین گردید، یک عدد تصادفی بین صفر و یک تولید می‌شود و توزیع تجمعی احتمال هر یک ار کروموزم‌ها مانند قبل محاسبه می‌شود. سپس کروموزم‌ها از اول بررسی شده و اولین کروموزمی که توزیع تجمعی احتمال آن بیشتر یا مساوی عدد تولید شده باشد، برای انتقال به حوضچه ژنتیک انتخاب می‌شود. این روش انتخاب همگرایی کندتری دارد زیرا در این روش بهترین کروموزم‌ها با بدترین کروموزم‌ها تفاوت چندانی ندارد.

روش بولتزمن:

ایده این روش از حرارت دادن فلزات نشات گرفته است (مانند روش شبیه‌سازی تبرید)، در این روش ابتدا دمای بالایی را برای شروع انتخاب کرده و رفته رفته از مقدار دما می‌کاهیم. دمای بالا بدین معنی است که در ابتدای اجرای الگوریتم کروموزم‌های بهینه و غیربهینه از شانس تقریبا یکسانی برای انتقال به حوضچه ژنتیک برخوردارند. این در حالی است که رفته رفته و با کاهش دما احتمال انتخاب کروموزم‌های بهتر افزایش و شانس انتخاب کروموزم‌های غیر بهینه کاهش می‌یابد.

پس از انتخاب کروموزم‌ها و انتقال آن‌ها به حوضچه ژنتیک، دو مرحله دیگر نیز نمودی از نظریه داروین هستند. این دو مرحله عبارتند از ادغام و جهش که در ادامه به بررسی آن‌ها می‌پردازیم.

عملگر ادغام

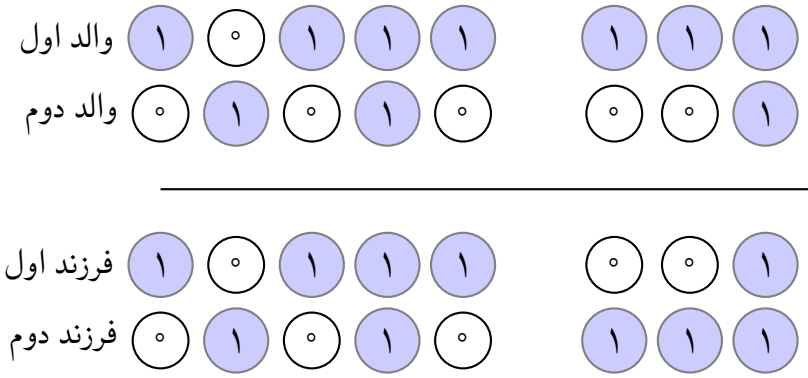
بخشی از فرایند تکامل در طبیعت بدین صورت است که کروموزم‌هایی از والدین انتخاب شده و با هم ترکیب می‌شوند. ما نیز در مرحله‌ی انتخاب، کروموزم‌هایی را برای ترکیب به حوضچه ژنتیک منتقل کردیم. حال باید کروموزم‌هایی از این حوضچه انتخاب کرده و با هم ادغام کنیم. به همین منظور در این بخش روش‌هایی را برای ادغام کروموزم‌ها بررسی خواهیم کرد. توجه شود که در برخی از مسائل و با توجه به روش کدگذاری کروموزم‌ها، عمل ادغام کروموزم‌ها موجب نقض برخی از شرایط مساله می‌گردد. بنابراین در چنین مواردی باید با استفاده از روش‌های که طراحی می‌کنیم به اصلاح کروموزم‌ها بپردازیم. احتمال وقوع ادغام با p_c نمایش داده می‌شود. نکته مهم در عمل ادغام این است که p_c، نقش مهم را در سرعت و بهبود الگوریتم ژنتیک دارد. اگر احتمال ادغام بالا باشد ترکیبات مختلف در جمعیت زیاد می‌شود ولی احتمال خراب شدن جواب‌های خوب و رفتن به نواحی غیر ضروری هم افزایش می‌یابد.

عملگرهای ادغام مختلفی وجود دارد که در اینجا به تعداد محدودی اشاره می‌کنیم. انواع عملگر ادغام عبارتند از:

- ادغام تک نقطه‌ای [۱۸]
- ادغام چند نقطه‌ای [۱۹]

[۱۸] Single-sight crossover

[۱۹] Multi-point crossover



شکل ۳.۲: ادغام تک نقطه‌ای

- ادغام یکنواخت [۲۰]
- ادغام نگه‌دارنده اولویت [۲۱]

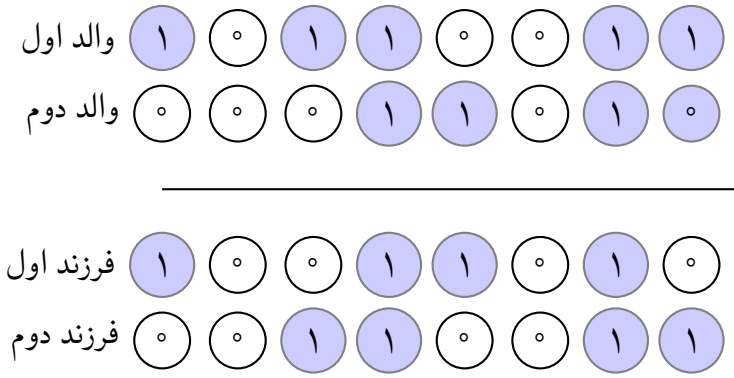
ادغام تک نقطه‌ای:
در ادغام تک نقطه‌ای، نقطه‌ای از کروموزم را به تصادف انتخاب کرده و سپس برای تولید دو کروموزم فرزند، ژن‌های بعد از این نقطه را در دو والد با هم جابجا می‌کنیم. نقطه انتخاب شده را محور[۲۲] می‌نامیم شکل (۳.۲) را ببینید.

ادغام چند نقطه‌ای:
در ادغام چند نقطه‌ای، m نقطه به‌صورت تصادفی و بدون تکرار تعیین شده و به‌صورت صعودی مرتب می‌شوند. سپس متغیرهای بین هر دو نقطه یکی در میان بین دو والد جابجا می‌شوند تا به این ترتیب دو فرزند جدید تولید شوند. هنگامی که تعداد نقاط زوج است، قسمت اول و آخر کروموزم‌های فرزند از یک والد مشترک ارث برده‌اند.

ادغام یکنواخت:
ادغام یکنواخت، حالت توسعه یافته‌ی روش چند نقطه‌ای است. در این روش هر بیت براساس یک احتمال پنجاه درصدی از والدها انتخاب شده و جابجا می‌شوند. شکل (۴.۲) را ببینید.

در روشی دیگر، ادغام یکنواخت توسط الگوی از پیش تعیین شده به نام ماسک [۲۳] صورت می‌گیرد. بدین ترتیب که برای تولید فرزند اول ژن‌های متناظر با مقدار ۱ در ماسک از والد اول و ژن‌های متناظر با صفر

[۲۰] Uniform crossover
[۲۱] Precedence preservative crossover
[۲۲] Pivot
[۲۳] Mask



شکل ۴.۲: ادغام یکنواخت

شکل ۵.۲: ادغام نگه‌دارنده اولویت

از والد دوم گرفته می‌شود. برای تولید فرزند دوم جای والدین عوض می‌شود. مقدار ماسک به‌صورت تصادفی برای هر جفت از والدها انتخاب می‌شود. از آن‌جایی که مقدار ماسک به‌صورت تصادفی برای هرجفت از والدها تولید می‌شود، در نتیجه فرزندان ترکیبی از ژن‌های والد خود خواهند بود. تعداد نقاط موثر (نقاطی در ماسک که مقدارشان با ژن بعدشان متفاوت است) ثابت نبوده و به‌صورت میانگین در حدود $\frac{L}{2}$ است که L طول کروموزم است.

ادغام نگه‌دارنده اولویت:

ادغام نگه‌دارنده اولویت (PPX) اغلب برای مسائل حمل‌ونقل و زمان‌بندی استفاده می‌شود. در این روش عملگر روی ارتباط اولویت‌های ژن‌ها عمل می‌کند (به‌عنوان مثال مساله خط تعادل مونتاژ U شکل). (PPX) برای یک مساله شامل شش ژن به‌صورت زیر است که در شکل (۵.۲) نشان داده شده است[۱۶].

- یک بردار به طول ژن‌های والدین به تصادف با عناصر ۱ و ۲ در نظر بگیرید. این بردار برای هر

ژن مشخص می‌کند که فرزند از والد اول یا والد دوم به ارث می‌برد.

- ابتدا یک فرزند خالی را در نظر بگیرید.

تا زمانی‌که کروموزم‌های والد تهی و کروموزم فرزند کامل شود، مراحل زیر را تکرار کنید.

- ژن سمت چپ را در یکی از دو والد مطابق با ترتیب والدین در بردار مشخص شده انتخاب کنید.
- ژن انتخاب شده را در هر دو والد حذف کنید.
- ژن انتخاب شده را به فرزند اضافه کنید.

عملگر جهش

نظریه داروین بیان می‌کند که پس از تولید فرزند، ممکن است در برخی از این کروموزم‌ها جهش‌هایی به تصادف روی دهد که موجب بهتر شدن یا بدتر شدن آن شوند. ما نیز در استفاده از الگوریتم‌های ژنتیک از این خصوصیت استفاده کرده و جهش‌هایی را در جواب‌های تولید شده برای مساله ایجاد می‌کنیم. این روش نیز بسته به ابتکار طراح الگوریتم به روش‌های گوناگونی پیاده‌سازی می‌شود. احتمال وقوع جهش در یک کروموزم را نرخ جهش می‌گویند و با P_M نشان می‌دهند. انواع عملگر جهش عبارتند از:

- یکنواخت [۲۴]
- جابجایی[۲۵]
- وارونگی [۲۶]
- جایگذاری[۲۷]

به‌طور کلی نمی‌توان گفت کدام جهش بهتر است. با توجه به نوع مساله جهش مناسب انتخاب می‌شود.

جهش یکنواخت:

در این عملگر یک ژن از کروموزم به‌صورت تصادفی انتخاب شده و مقدار آن به مقدار تصادفی دیگری تغییر می‌یابد یعنی یک عدد تصادفی بین$[۱, L]$ که L طول کروموزم می‌باشد انتخاب شده و ژن موجود در آن کروموزم تغییر می‌کند به‌عنوان مثال فرض‌کنید:

فرزند : ۰۱۱۰۱۰۱۰۰۱۰

عدد تصادفی تولید شده پنج باشد. بنابراین ژن موجود در مکان پنج عوض می‌شود. یعنی یک تبدیل به صفر شده در نتیجه کروموزم به صورت زیر در می‌آید:

---

[۲۴] Uniform
[۲۵] Swap
[۲۶] Inversion
[۲۷] Insert

فرزند بعد از جهش : ۰۱۱۰۰۰۱۰۰۱۰

جهش جابجایی :
در این روش دو عدد تصادفی انتخاب می‌کنیم سپس ژن‌ها در این دو مکان با یکدیگر تعویض می‌شوند به‌عنوان مثال اگر دو عدد تصادفی ۳ و ۶ باشند آنگاه ژن‌ها در مکان ۳ و ۶ را با یکدیگر تعویض می‌نماییم و داریم:

فرزند : ۶۱۸۲۵۷۴۳
فرزند بعد از جهش: ۶۱۷۲۵۸۴۳

جهش وارونگی:
در این روش ابتدا دو عدد تصادفی را بین $[۱, L]$ که L طول کروموزم می‌باشد، انتخاب کرده و کروموزم به سه بخش تقسیم می‌شود. حال قسمت وسط برعکس می‌گردد. به‌عنوان مثال دو عدد تصادفی ۲ و ۵ باشند بعد از جهش فرزند به صورت زیر در می‌آید:

فرزند : ۳۴/۱۷۵/۸۲۶
فرزند بعد ازجهش : ۳۴/۵۷۱/۸۲۶

جهش جایگذاری:
در این روش ژن به صورت تصادفی انتخاب شده و جای آن عوض می‌شود فرض کنید:

فرزند : ۳۴۲۶۵۷۱۸

و عدد تصادفی بدست آمده برای مشخص کردن ژن جهت جابجایی برابر ۶ بوده و همچنین برای محل جدید آن نیز مکان بین ۳ و ۴ انتخاب شده باشد در این‌صورت فرزند بعد از جهش به‌صورت زیر در می‌آید:

فرزند بعد از جهش : ۳۴۲۷۶۵۱۸

۹.۳.۲ شرط توقف

یکی دیگر از عوامل در بهبود الگوریتم ژنتیک برای پیاده‌سازی به روی یک مساله خاص شرط توقف است. عمده‌ترین شروط توقف را می‌توان به‌صورت زیر فهرست کرد.

۱. رسیدن به جوابی که دارای صلاحیت (برازندگی) بیشتر از مقدار تعیین شده باشد.

۲. تولید تعداد معلوم نسل

۳. رسیدن به جمعیتی که برازندگی آن‌ها تقریبا یکی باشد یا تغییرات محسوس در جمعیت صورت نگیرد و یا واریانس برازندگی جواب‌ها از مقدار مشخص کمتر گردد.

۱۰.۳.۲ مزایای الگوریتم ژنتیک

۱. الگوریتم ژنتیک با یک مجموعه از جواب‌ها که همان جمعیت است شروع به جست‌وجو می‌کند. بدین ترتیب به‌جای یافتن نقطه مناسب، محدوده‌های مناسبی در فضای جست‌وجو شناسایی می‌گردد. در نتیجه همزمان با ایجاد شانس بیشتر برای متغیرهای شایسته‌تر، جست‌وجوی سایر مناطق فضای جست‌وجو با شایستگی کمتر نادیده گرفته نمی‌شود. لذا احتمال یافتن نقطه بهینه سراسری افزایش می‌یابد. این ویژگی بخصوص برای توابع با تغییرات ناگهانی و دارای چند نقطه بهینه موضعی مناسب است. یکی از مهم‌ترین تفاوت‌های الگوریتم ژنتیک با الگوریتم‌های کلاسیک هم در این امتیاز است.

۲. در الگوریتم ژنتیک با تابع برازندگی سروکار داریم که معمولا از تابع هدف استفاده می‌شود. بنابراین فقط اطلاعاتی درباره خود تابع هدف نیاز داریم نه به مشتق و ...درباره آن. به همین دلیل به راحتی می‌توان از این روش در بهینه‌سازی توابع گسسته و یا توابع مشتق‌ناپذیر و یا توابع با تغییرات ناگهانی و چندین جواب بهینه موضعی استفاده کرد.

۳. استفاده از قواعد احتمالی و ابتکاری به‌جای قواعد قطعی یکی دیگر از امتیازات الگوریتم ژنتیک است. زیرا این امتیاز سبب دوری از بهینه‌های موضعی می‌شود. البته استفاده الگوریتم ژنتیک از قواعد احتمالی به معنای یک جست‌وجوی صرفا تصادفی نیست، بلکه این الگوریتم از انتخاب تصادفی به‌عنوان ابزاری برای هدایت عمل جست‌وجو استفاده می‌کند. توجه کنید که الگوریتم ژنتیک فقط به نسل قبل وابسته است، پس می‌توان آن را نوعی فرآیند مارکوف دانست. (فرآیند مارکوف فرآیندی است که به مرحله قبل خود وابسته است)

۴. برتری دیگر الگوریتم ژنتیک موازی بودن آن می‌باشد، از آن جایی که الگوریتم ژنتیک چندین نقطه شروع دارد در یک لحظه می‌تواند فضای مساله را از چند جهت مختلف جست‌وجو کند. امکان پیاده‌سازی این الگوریتم روی ماشین‌های موازی وجود دارد که باعث افزایش سرعت حل مسایل با پیشرفت علم کامپیوتر شده است.

۵. چون پیچیدگی زمانی مسایل بزرگ تابع نمایی است و جزء مسایل NP-سخت محسوب می‌شود به همین دلیل از الگوریتم‌های فراابتکاری مانند ژنتیک استفاده می‌شود.

۱۱.۳.۲ معایب الگوریتم ژنتیک

مشکل اصلی الگوریتم ژنتیک علی‌رغم سادگی پیاده‌سازی، هزینه اجرای آن است. اغلب حل یک مساله نیازمند تولید چندین نسل از کروموزم‌ها است و این مساله نیاز به زمان زیادی دارد (خصوصا اگر تعداد جمعیت اولیه زیاد باشد و نیز تابع هدف تابع پیچیده‌ای باشد). گاه پیش می‌آید که برای حل یک مساله به‌عنوان مثال یک پردازنده پنتیوم باید زمان طولانی برنامه را اجرا کند. طبیعی است که این زمان زیادی است برای حل یک مساله و همین امر گاهی استفاده از الگوریتم را با مشکل مواجه کند.

۴.۲ الگوریتم ژنتیک و مساله زمان‌بندی امتحانات

ناجی عظیمی [۱۲]، الگوریتم ژنتیک را برای مساله زمان‌بندی امتحانات به‌کار برده است. مطالب مطرح شده در این بخش برگرفته از [۱۲] است. مساله یک محدودیت سخت و یک محدودیت نرم دارد، مدل مساله رابطه (۳.۱) است که در فصل اول آمده است و هدف مینیمم کردن هزینه‌ها می‌باشد. جواب اولیه تصادفی تولید می‌شود و بعد از بدست‌آوردن چند جواب، مجموعه جواب‌ها را جمعیت اولیه در نظر می‌گیریم.

۱.۴.۲ پیاده‌سازی الگوریتم ژنتیک برای مساله زمان‌بندی امتحانات

چنانچه اندازه جمعیت خیلی بزرگ باشد، سرعت اجرای الگوریتم به شدت افزایش خواهد یافت از طرفی اگر اندازه جمعیت کوچک باشد مساله خیلی زود به جوابی که لزوما بهینه نیست همگرا خواهد شد. اندازه جمعیت حداکثر ۱۰۰ در نظر گرفته شده است. شرط خاتمه الگوریتم تکرار تعداد مشخص نسل (حداکثر تعداد نسل‌ها) می‌باشد. در این بخش به بیان مراحل مختلف این الگوریتم، برای مساله جدول زمان‌بندی امتحانات می‌پردازیم.

مرحله اول: کدگذاری

برای کدگذاری کروموزم‌ها از روش جایگشتی استفاده شده است. در کدگذاری جایگشتی اعداد صحیح نمایانگر ژن‌ها در کروموزم است. هر جواب با یک بردار نشان داده می‌شود که طول بردار تعداد امتحانات است و هر عنصر این بردار بازه تخصیص داده شده به امتحان است. هر رشته جواب معادل یک کروموزم است و هر کروموزم معادل یک جدول زمان‌بندی برای امتحانات فرض شده است که هر یک دارای برازندگی مربوط به خود می‌باشد. هر جدول زمان‌بندی امتحانات به‌عنوان یک ساختار خطی است که هر سلول ساختار خطی نشان دهنده یک امتحان در جدول زمان‌بندی امتحانات می‌باشد. در هر سلول یک مقدار صحیح می‌باشد که بازه زمانی تخصیص داده شده به امتحان را نشان می‌دهد. شکل زیر کروموزمی را نمایش می‌دهد که اعداد در هر سلول یک بازه زمانی است و به تعداد امتحانات سلول داریم. به‌عنوان مثال عدد ۲۰ در سلول اول بازه ۲۰ام تخصیص داده شده به امتحان اول را نشان می‌دهد. اگر نیاز باشد امتحانات از شنبه تا چهارشنبه هفته قرار داده شود، در هر روز چهار بازه زمانی داریم بنابراین ۲۰ بازه زمانی برای برنامه‌ریزی امتحانات در کل هفته داریم، به‌عنوان مثال عدد ۱ بازه زمانی ۱۰ – ۸ روز شنبه و عدد ۲ بازه زمانی ۱۲ – ۱۰ روز شنبه را نمایش می‌دهد و عدد ۷ بازه زمانی ۱۶-۱۴ روز یکشنبه است. اگر قرار باشد امتحانات در دو هفته برنامه‌ریزی شود با توجه به آنچه گفته شد کلا ۴۰ بازه برای تخصیص امتحانات داریم.

۲۰	۳	۱۸	.....	۷	۱۱	۶

مرحله دوم: تولید مثل

در این مرحله برای تمام کروموزم‌ها مقدار برازندگی از جمعیت را محاسبه کرده که هر چقدر مقدار برازندگی کروموزم کمتر باشد آن کروموزم مطلوب‌تر می‌باشد. سپس میانگین برازندگی جمعیت را محاسبه می‌کنیم.

کروموزم‌هایی از جمعیت را که مقدار برازندگی آن از میانگین برازندگی جمعیت کمتر است انتخاب می‌کنیم، کروموزم‌های انتخاب شده در محل جدیدی جمع‌آوری می‌کنیم که به آن حوضچه ژنتیک گفته می‌شود. والدین را از حوضچه ژنتیک برای ادغام انتخاب می‌کنیم. کروموزم‌های انتخاب شده را به حوضچه ژنتیک انتقال می‌دهیم.

مرحله سوم: عملگرانتخاب

انتخاب کروموزم‌ها از حوضچه ژنتیک که در مرحله قبل بیان شده است به روش‌های گوناگونی انجام می‌پذیرد. در این پژوهش با استفاده از روش چرخ رولت دو والد را از حوضچه ژنتیک انتخاب می‌کنیم. این مرحله شامل چندین گام است که به‌صورت زیر می‌باشد:

گام اول: ابتدا کروموزم‌ها را براساس کاهش برازندگی نسبی مرتب می‌کنیم. بدین ترتیب ابتدا آن‌هایی که برازندگی نسبی بیشتری دارند انتخاب می‌شوند که بعد از ادغام بهبود یابند و مقدار برازندگی نسبی آن‌ها کاهش یابد زیرا در این مساله هدف کاهش مقدار برازندگی کروموزم‌ها است.

گام دوم: عدد تصادفی r را بین صفر و یک انتخاب می‌کنیم.

گام سوم: برای $i = ۰$ تا $(N - ۱)$ که N اندازه جمعیت است اعمال زیر را انجام می‌دهیم:

- مقدار برازندگی هر کروموزم را بر مجموع برازندگی همه کروموزم‌ها تقسیم می‌کنیم که برازندگی نسبی برای هر کروموزم می‌باشد. حال مجموع برازندگی نسبی کروموزم‌ها را تا کروموزم i محاسبه می‌کنیم. به این‌صورت که مقدار احتمال هر کروموزم با مقدار احتمال کروموزم‌های قبل از آن جمع می‌شود و مقدار حاصل را با sum نشان می‌دهیم.
- اگر $(sum - ۱)$ کمتر یا مساوی عدد تصادفی r باشد کروموزم i را انتخاب کرده و از حلقه خارج می‌شویم.
- در غیر این‌صورت i را افزایش داده و با کروموزم بعدی ادامه می‌دهیم.

مرحله چهارم: عملگر ادغام

عملگر ادغام با احتمال $P_c = ۰/۸$ انجام می‌شود. دو والد را که در مرحله قبل انتخاب شده است به روش ادغام تک نقطه‌ای ترکیب می‌کنیم و دو فرزند تولید می‌شود.

مرحله پنجم: عملگر جهش

عملگر جهش با احتمال $P_M = ۰/۰۲$ انجام می‌شود. یک فرزند را به تصادف انتخاب کرده و مقدار یک ژن از کروموزم فرزند را به تصادف تغییر می‌دهیم. در این عملکرد یک امتحان را به تصادف انتخاب کرده‌ایم و بازه زمانی آن را با یک بازه تصادفی تغییر می‌دهیم.

مرحله ششم: انتخاب بهترین فرد

در این مرحله مقدار برازندگی فرزندان تولید شده و افراد در جمعیت را به صورت نزولی مرتب نموده و کروموزم یا فرد با کمترین مقدار برازندگی به‌عنوان بهترین جواب در حافظه‌ای نگهداری می‌شود. زمانی‌که شرط توقف برقرار شد همانطور که گفته شد برای هر نسل بهترین فرد را انتخاب نموده‌ایم بهترین فرد تولید شده در میان تمام نسل‌ها جواب مورد نظر می‌باشد.

توصیف کلی الگوریتم ژنتیک در مقاله [۱۲]

- جمعیت اولیه از N کروموزم تولید کنید.
- تا زمانی‌که تعداد نسل‌ها در شرط توقف صدق نمی‌کند کارهای زیر را انجام دهید.

۱. شروع

۲. مقدار برازندگی برای هر فرد را محاسبه کنید.

۳. تولید مثل را انجام دهید و کروموزم‌ها را بعد از تولید مثل به حوضچه ژنتیک انتقال دهید.

۴. دو والد از حوضچه ژنتیک با استفاده از روش چرخ رولت انتخاب کنید.

۵. دو والد انتخاب شده را با احتمال P_c به روش تک نقطه‌ای ترکیب کنید.

۶. جهش با احتمال P_M را بر فرزندان انجام دهید.

- پایان.

پارامترها

پارامتر N نشان دهنده اندازه جمعیت می‌باشد، اندازه جمعیت در تمام مراحل اجرای الگوریتم مقدار ثابتی دارد. مقدارهای مختلف برای آن آزمایش شده است و این مقادیر در جدول (۱.۲) آمده است. بیشترین مقداری که برای اندازه جمعیت در نظر گرفته شده ۲۰۰ است ولی مناسب نیست زیرا زمان زیادی برای اجرا صرف می‌شود و کوچکترین مقداری که برای اندازه جمعیت در نظر گرفته شده ۱۰ است که این تعداد خیلی کم است زیرا سبب همگرایی زودرس می‌شود. اندازه جمعیت ۱۰۰ در نظر گرفته شده است. پارامتر P_M و P_c نشان دهنده احتمال نسبی جهش و ترکیب می‌باشند. مقادیر مختلفی به آن‌ها تخصیص داده شده است که در جدول (۱.۲) آورده شده است. با این مقادیر الگوریتم انجام شده است و مقدارهایی که جواب بهتری داشته برای این دو پارامتر یافت شده است. بهترین مقدار پارامترها در جدول (۲.۲) آمده است.

جدول ۱.۲: مقادیر پارامترها

پارامتر	مقادیر			
N	۱۰	۵۰	۱۰۰	۲۰۰
P_M		۰/۰۲	۰/۱	۰/۵
P_c		۰/۵	۰/۸	۱

جدول ۲.۲: بهترین پارامترها

پارامتر	مقدار
N	۱۰۰
P_M	۰/۰۲
P_c	۰/۸

۲.۴.۲ الگوریتم پیشنهادی

الگوریتم ژنتیک روش بهینه‌سازی کارآمدی است که خصایص مثبت روش‌های تصادفی و حریصانه را تا حد زیادی در خود جمع کرده است. لیکن همواره می‌توان با توجه به خصوصیات مساله تغییراتی را در یک الگوریتم ژنتیک استاندارد پیشنهاد نمود. در این بخش سعی می‌کنیم با تغییراتی الگوریتم را بهبود بخشیم.

جهش:

اپراتور جهش از یک‌دستی جامعه و افتادن در نقاط بهینه محلی جلوگیری می‌کند. در روش پیشنهادی ما به‌جای اپراتور جهش تصادفی در [۱۲] ایده جهش ابتکاری را مورد استفاده قرار داده‌ایم:

جهش از طریق جابجایی تعدادی از ژن‌ها (بازه‌ها) محقق می‌شود. به این صورت که دو عدد تصادفی بین بازه $[۱, N]$ که N طول کروموزم‌ها (تعداد امتحانات) می‌باشد انتخاب کرده و کروموزم به سه قسمت مساوی تقسیم می‌شود. حال قسمت وسط عکس می‌شود. با این ابتکار تخصیص چند بازه به امتحانات را تغییر می‌دهیم. این جهش نسبت به جهش قبلی بهتر بوده وباعث بهبود جواب شده است.

ماتریس تداخل ماتریس متقارن مربعی است که تعداد سطرها و ستون‌ها برابر تعداد امتحانات است و درایه موجود در سطر i و ستون j در ماتریس تداخل تعداد دانشجویان مشترک در هر دو امتحان i و j را نشان می‌دهد. ماتریس تداخل در مقاله [۱۲] تصادفی تولید شده است، در روش پیشنهادی ما ماتریس تداخل را با استفاده از پیش‌ثبت نام دانشجویان تولید کرده‌ایم به این‌صورت که داده‌هایی که هر دانشجو در کدام امتحانات نام‌نویسی کرده است را از ورودی گرفته بعد از انجام اعمالی، ماتریس تداخل را تولید می‌کنیم، که این کار تا حد قابل توجهی مقدار تابع برازش را کاهش می‌دهد.

۳.۴.۲ مثال پیاده‌سازی زمانبندی امتحانات برای الگوریتم ژنتیک

مجموعه امتحانات، مجموعه بازه‌های زمانی، مجموعه دانشجویان و نام نویسی دانشجویان برای امتحانات را در نظر می‌گیریم. داده‌های لازم را در جدول (۳.۲) آورده شده است. ستون دوم بازه‌هایی که امتحانات باید در آن‌ها برگزار شوند می‌باشد و ستون سوم دانشجویان هستند که باید با توجه به پیش‌ثبت نام آن‌ها امتحانات برنامه‌ریزی شود.

جدول ۳.۲: مشخصات مجموعه داده‌ها

دانشجویان	بازه‌های زمانی	دروس امتحانی
ب	شنبه ساعت ۱۸- ۱۶	شیمی
آ	شنبه ساعت ۱۶-۱۴	فیزیک
پ	شنبه ساعت ۱۲ – ۱۰	ریاضی
ج	شنبه ساعت ۱۰ – ۸	فارسی
		ورزش

نام نویسی دانشجویان در جدول (۴.۲) آمده است. به‌عنوان مثال در ستون اول دروس امتحانی را که دانشجوی آ ثبت نام کرده قرار داده شده و دانشجوی آ در دروس شیمی، ورزش، ریاضی و فارسی ثبت نام کرده است.

جدول ۴.۲: نام نویسی دانشجویان

ج	پ	ب	آ
شیمی	ورزش	شیمی	شیمی
فارسی	شیمی	فیزیک	ورزش
ورزش	فیزیک	ریاضی	ریاضی
	ریاضی		فارسی
	فارسی		

با استفاده از پیش‌ثبت نام دانشجویان ماتریس تداخل را بدست می‌آوریم که ماتریس متقارن از مرتبه ۵ می‌باشد زیرا ۵ امتحان داریم. به‌عنوان مثال عدد ۳ در سطر اول و ستون اول روی قطر اصلی تعداد دانشجویان که در امتحان اول نام نویسی کرده می‌باشد و عدد ۲ در سطر اول و ستون دوم تعداد دانشجویان مشترک در امتحان اول و دوم را نشان می‌دهد. ماتریس تداخل به‌صورت زیر است.

$$\begin{bmatrix} ۳ & ۲ & ۳ & ۲ & ۲ \\ ۲ & ۲ & ۲ & ۱ & ۱ \\ ۳ & ۲ & ۴ & ۳ & ۳ \\ ۲ & ۱ & ۳ & ۳ & ۳ \\ ۲ & ۱ & ۳ & ۳ & ۳ \end{bmatrix}$$

حال به‌صورت تصادفی ۱۰ (اندازه جمعیت) جواب اولیه تولید می‌کنیم. جمعیت اولیه که مجموعه جواب‌های اولیه می‌باشند و مقدار برازندگی‌ها که با استفاده از رابطه (۳.۱) بدست آورده شده در جدول (۵.۲) آمده است. اعداد صحیح ۱ تا ۴ بازه روز شنبه ساعت ۱۰ – ۸ وبه همین ترتیب تا بازه روز شنبه ساعت ۱۸-۱۶ را نشان می‌دهند. به‌عنوان مثال جواب ۱ که در سطر دوم جدول (۵.۲) قرار دارد

نشان دهنده یک برنامه‌ریزی برای امتحانات است و مقدار برازندگی آن نیز آورده شده است. در جواب ۱ امتحان ریاضی در روزشنبه ساعت ۱۲ – ۱۰ می‌باشد، امتحان فیزیک روز شنبه ساعت ۱۸-۱۶، امتحان شیمی در روز شنبه ساعت ۱۲ – ۱۰، امتحان فارسی روز شنبه ساعت ۱۲ – ۱۰ و امتحان ورزش روز شنبه ساعت ۱۸-۱۶ می‌باشد و سایر جواب‌ها نیز به همین صورت می‌باشند.

جدول ۵.۲: جمعیت اولیه و مقدار برازندگی‌ها

	ریاضی	فیزیک	شیمی	فارسی	ورزش	مقادیر برازندگی
جواب ۱	۲	۴	۲	۲	۳	$F = ۲۰۴۶$
جواب ۲	۲	۳	۴	۴	۱	$F = ۷۹۶$
جواب ۳	۴	۴	۴	۴	۲	$F = ۳۲۶۴$
جواب ۴	۴	۳	۲	۱	۴	$F = ۵۵۱$
جواب ۵	۱	۲	۲	۲	۲	$F = ۳۲۸۶$
جواب ۶	۳	۲	۳	۴	۴	$F = ۱۵۶۰$
جواب ۷	۳	۳	۳	۱	۴	$F = ۱۷۸۹$
جواب ۸	۳	۲	۴	۴	۲	$F = ۱۰۵۴$
جواب ۹	۳	۱	۴	۲	۳	$F = ۵۲۶$
جواب ۱۰	۱	۳	۲	۳	۲	$F = ۱۰۶۴$

کروموزمی که کمترین مقدار برازندگی دارد را به‌عنوان بهترین جواب در x^* ذخیره می‌شود.

x^*	۳	۱	۴	۲	۳

حال میانگین برازندگی‌هایی را که در جدول (۵.۲) آورده شده بدست آورده سپس کروموزم‌هایی را که مقدار برازندگی آن‌ها از میانگین بدست آمده کمتر هستند در حوضچه ژنتیک قرار داده و والدین را از میان آن‌ها انتخاب می‌کنیم. جواب‌های انتخابی در این مرحله در جدول (۶.۲) آورده شده است.

$$\overline{F} = ۱۵۹۳/۴$$

حال با استفاده از روش چرخ رولت که در بخش قبل توضیح داده‌ایم، دو والد را برای ادغام انتخاب می‌کنیم. به این صورت که ابتدا به ترتیب کاهش برازندگی نسبی کروموزم‌ها را مرتب نموده و ابتدا کروموزمی که بدتر است را انتخاب نموده با انجام این کار جواب‌های بدتر را ابتدا بهبود می‌دهیم. والدین انتخاب شده در جدول زیر آورده شده است.

والد اول	۳	۲	۳	۴	۴	$F = ۱۵۶۰$
والد دوم	۳	۲	۴	۴	۲	$F = ۱۰۵۴$

جدول ۶.۲: حوضچه ژنتیک

مقادیر برازندگی	ورزش	فارسی	شیمی	فیزیک	ریاضی
$F = ۷۹۶$	۱	۴	۴	۳	۲
$F = ۵۵۱$	۴	۱	۲	۳	۴
$F = ۱۵۶۰$	۴	۴	۳	۲	۳
$F = ۱۰۵۴$	۲	۴	۴	۲	۳
$F = ۵۲۶$	۳	۲	۴	۱	۳
$F = ۱۰۶۴$	۲	۳	۲	۳	۱

در این قسمت بر روی دو والد انتخاب شده با احتمال $P_c = ۰/۸$ عمل ادغام تک نقطه انجام می‌شود. فرزندان تولید شده در جدول زیر آورده شده است.

فرزند اول	۳	۲	۳	۴	۲	$F = ۱۰۶۴$
فرزند دوم	۳	۲	۴	۴	۴	$F = ۷۹۸$

یک فرزند را به تصادف انتخاب نموده که فرزند دوم می‌باشد و با احتمال $P_M = ۰/۰۲$ جهش بر آن انجام می‌شود، در بخش قبل جهش مورد استفاده را بیان کرده‌ایم. فرزند بعد از جهش به‌صورت زیر است.

۴	۱	۴	۲	۳	$F = ۷۹۸$

فرزندان را جایگزین کروموزم‌هایی از جمعیت می‌کنیم که در جمعیت بدترین هستند و از فرزندان نیز بدتر هستند. جمعیت جدید جمعیت اولیه برای نسل بعد در نظر گرفته می‌شود. x^* تغییر نمی‌کند. شرط توقف تعداد تکرار مشخص از نسل است بنابراین این کار برای ۵ نسل انجام می‌شود و جواب نهایی به‌صورت زیر است.

x^*	۳	۱	۴	۲	۳	$F = ۵۲۶$

بنابراین جدول زمان‌بندی امتحانات به‌صورت زیر است.

ورزش	فارسی	شیمی	فیزیک	ریاضی
شنبه ۱۶-۱۴	شنبه ۱۲ – ۱۰	شنبه ۱۸-۱۶	شنبه ۱۰ – ۸	شنبه ۱۶-۱۴

برنامه الگوریتم ژنتیک پیشنهادی و ژنتیک ارایه شده در [۱۲] که با استفاده از زبان برنامه نویسی متلب ۲۰۱۰ نوشته شده است، برروی ۱۰ داده آزمایش انجام شده است، اندازه جمعیت و تعداد نسل‌ها ۵۰ و ۱۰۰ در نظر گرفته شده است. مشخصات مجموعه داده‌ها در جدول (۷.۲) آورده شده است.

جدول ۷.۲: مشخصات مجموعه داده‌ها

شماره	تعداد امتحانات	تعداد بازه‌ها	تعداد دانشجویان
۱	۲۰	۱۵	۱۰۰
۲	۴۰	۲۰	۳۰۰
۳	۲۶	۱۵	۳۰۰
۴	۲۶	۲۰	۲۰۰
۵	۲۶	۱۰	۳۰۰
۶	۳۰	۱۵	۲۵۰
۷	۲۶	۲۴	۱۵۰
۸	۶۰	۳۲	۴۰۰
۹	۵۰	۲۴	۸۰۰
۱۰	۴۴	۲۶	۴۰۰

در روش ژنتیک با ماتریس حاصل از پیش‌ثبت نام مقدار تابع برازش تا حد قابل توجهی کمتر است نسبت به حالتی که ماتریس تداخل تصادفی تولید شود زیرا در ماتریس با استفاده از پیش‌ثبت نام درایه‌های بدست آمده با استفاده از نام نویسی دانشجویان است و اعداد کوچکتر می‌باشد ولی در ماتریس تداخل تصادفی درایه‌ها تصادفی تولید می‌شوند و ممکن است اعداد بزرگ باشند. همچنین جهش معکوس استفاده شده باعث بهبود قابل توجهی در مقدار تابع برازندگی گردیده است زیرا به طور همزمان بازه چند امتحان را عوض کرده‌ایم ولی در جهش یک نقطه فقط بازه یک امتحان تعویض می‌گردد و ممکن است با یک جهش جواب خوبی حاصل نشود. زمان اجرای برنامه تقریبا در تمام حالت‌ها یکسان است. نتایج عددی تابع برازندگی و زمان اجرای برنامه برای جهش تک نقطه‌ای و ماتریس تداخل با استفاده از پیش ثبت نام $(GA۳)$ ، الگوریتم ژنتیک با جهش معکوس و ماتریس تداخل با استفاده از پیش ثبت نام $(GA۴)$، الگوریتم ژنتیک با جهش معکوس و ماتریس تداخل تصادفی $(GA۲)$، الگوریتم ژنتیک با جهش تک نقطه و ماتریس تداخل تصادفی $(GA۱)$ به‌ترتیب در جدول (۸.۲) و (۹.۲) آورده شده است.

جدول ۸.۲: نتایج عددی تابع برازندگی

data	GA۱	GA۲	GA۳	GA۴
۱	۳۹۳٫۱۱	۱۸۴٫۰۳	۳۰۶٫۷۴	۲۹۴٫۶۲
۲	۳۸۲۰	۳۴۳۶	۵۵۳٫۹۳	۵۲۰٫۳۱
۳	۱۲۰۳	۱۱۹۵	۶۶۲٫۲۵	۶۶۲٫۲۵
۴	۶۳۶٫۷۳	۶۱۹٫۴۲۶	۳۸۲٫۶۰	۳۵۰
۵	۲۴۳۶	۲۶۰۰	۱۲۶۷	۱۲۱۹
۶	۱۶۹۲	۱۹۶۲	۶۸۴٫۵۸	۷۱۸٫۱۶
۷	۴۴۷٫۳۸	۴۰۵٫۴۵	۳۲۳٫۸۳	۲۳۰٫۷۲
۸	۵۵۴۷	۵۱۹۶	۳۸۸٫۷۳	۳۷۵٫۵۵
۹	۵۵۴۷	۵۱۹۶	۲۴۲٫۲۹	۲۲۷٫۷۵
۱۰	۳۱۵۵	۳۱۱۸	۲۶۰٫۳۷	۲۶۹٫۸۵

جدول ۹.۲: نتایج عددی زمان اجرا بر حسب ثانیه

data	GA۱	GA۲	GA۳	GA۴
۱	۸٫۵۸۶	۸٫۵۵۱	۸٫۷۵۵	۸٫۶۵۵
۲	۲۳٫۰۲۵	۲۲٫۱۷۶	۲۴٫۵۳۷	۲۴٫۳۴۳
۳	۲۴٫۷۲۷	۲۴٫۸۷۹	۲۵٫۵۵۰	۲۶٫۳۷۹
۴	۲۴٫۲۵۱	۲۴٫۵۵۶	۲۴٫۹۲۰	۲۵٫۶۷۱
۵	۱۲٫۲۶۷	۱۲٫۳۴۷	۱۲٫۲۸۷	۱۲٫۳۰۱
۶	۶٫۶۷۵	۶٫۵۸۸	۶٫۸۹۸	۶٫۷۶۷
۷	۵٫۳۸۶	۵٫۶۳۹	۵٫۶۳۵	۵٫۶۲۷
۸	۲۴٫۵۴۲	۲۳٫۵۴۲	۲۲٫۵۶۴	۲۲٫۴۵۲
۹	۱۷٫۶۶۳	۱۶٫۹۸۶	۱۸٫۲۷۷	۱۶٫۸۴۳
۱۰	۱۱٫۹۳۹	۱۱٫۹۵۷	۱۱٫۷۵۱	۱۲٫۱۷۵

فصل ۳

جست‌وجوی ممنوع

۱.۳ مقدمه

الگوریتم جست‌وجوی ممنوع [۱] یک الگوریتم بهینه‌سازی فراابتکاری است که برای اولین بار در سال ۱۹۸۶ توسط گلوور [۲] [۱۰] معرفی شد. در سال ۱۹۹۷، اولین کتابی که کاملا به جست‌وجوی ممنوعه اختصاص داشت توسط گلوور و لاگونا [۹]منتشر شد. واژه تابو از تنگان زبان مردم جزایر پلینزی در اقیانوس آرام گرفته شده است. این واژه به معنای شی مقدسی است که به‌دلیل قداست نباید آنرا لمس کرد. بر اساس واژنامه وبستر [۳]، امروزه این واژه در معنای ممنوعیت ایجاد شده به دلیل فرهنگ اجتماعی برای ایجاد اقدام حفاظتی یا ممنوعیت چیزی که دارای ریسک است، به‌کار می‌رود. معنای اخیر واژه تابو، با تکنیک جست‌وجوی ممنوعه کاملا سازگار است. ریسکی که در الگوریتم جست‌وجوی ممنوعه از آن اجتناب می‌شود، خطر مسیرهای نامناسب است.

یکی از موثرترین الگوریتم‌های حل برای مساله زمان‌بندی الگوریتم جست‌وجوی ممنوع است. کارایی روش جست‌وجوی ممنوع در مسایل بهینه‌سازی ترکیباتی اثبات شده است. الگوریتم جست‌وجوی ممنوع شامل مولفه‌های رایجی همچون لیست ممنوعه، حافظه چندتایی و مختلف، ساختارهای همسایگی و ...است. یکی از فاکتورهای مهم که تاثیر زیادی روی میزان کارایی الگوریتم دارد تعریف ساختار همسایگی مناسب با ساختار مساله می‌باشد. روش جست‌وجوی ممنوع مبتنی بر منع، با ایجاد تغییری کوچک در روش جست‌وجو همسایه به‌وجود می‌آید. هدف این روش آن است که بخش‌هایی از مجموعه جواب که پیش از این بررسی نشده است، مدنظر قرار گیرد. بدین منظور حرکت به جواب‌هایی که اخیرا جست‌وجو شده ممنوع خواهد بود. خاصیت اصلی جست‌وجوی ممنوع بر لیست ممنوع است که باعث گسترش مساله می‌شود.

---

[۱] Tabu search

[۲] Glover

[۳] Webster

۲.۳ جست‌وجوی ممنوع

ساختار کلی روش جستجوی ممنوع بدین صورت است که ابتدا یک جواب اولیه انتخاب می‌شود. سپس برای جواب مربوط براساس یک معیار خاص مجموعه‌ای از جواب‌های همسایه امکان‌پذیر در نظر گرفته می‌شود.
در گام بعد، پس از ارزیابی جواب‌های همسایه تعیین شده بهترین آن‌ها انتخاب می‌شود و جابجایی از جواب جاری به جواب همسایه انتخابی صورت می‌گیرد. این فرآیند به همین ترتیب تکرار می‌شود تا زمانیکه شرط خاتمه تحقق یابد.
در روش جست‌وجوی ممنوع فهرستی وجود دارد که جابجایی‌های منع شده را نگهداری می‌کند و به فهرست تابو معروف است و کاربرد اصلی آن پرهیز از همگرا شدن به جواب بهینه محلی است. به عبارت دیگر، به کمک فهرست ممنوع جابجایی به جواب‌هایی که اخیرا جست‌وجو شده‌اند ممنوع خواهد شد. فقط بخش‌هایی از مجموعه جواب که پیش از این مورد بررسی قرار نگرفته‌اند مد نظر خواهند بود. در واقع جابجایی از جواب جاری به جواب همسایه شدنی زمانی انجام می‌شود که در فهرست ممنوع قرار نداشته باشد. در غیر این‌صورت جواب همسایه دیگری که در ارزیابی جواب‌های همسایه در رده بعدی قرار گرفته است انتخاب شده و جابجایی به آن صورت می‌گیرد. الگوریتم معیاری به نام معیار آرمانی را چک خواهد کرد. براساس معیار آرمانی اگر جواب همسایه از بهترین جواب یافت شده تاکنون بهتر باشد، الگوریتم به آن حرکت خواهد کرد، حتی اگر آن جواب در فهرست ممنوع باشد.
در روش جست‌وجوی ممنوع بعد از هر جابجایی فهرست ممنوع به‌روز رسانی می‌شود به این معنا که جابجایی جدید به آن فهرست اضافه شده تا از بازگشت مجدد الگوریتم به آن جواب و ایجاد سیکل جلوگیری شود. یک جواب تا تعداد تکرار مشخصی در فهرست ممنوع قرار دارد. مدت زمانی که حرکت‌ها در فهرست ممنوعه قرار می‌گیرند توسط یک پارامتر که زمان ممنوعه [۴] نام دارد تعیین می‌شود. نحوه انتخاب با توجه به نوع مساله و شرایط می‌تواند متفاوت باشد.

۱.۲.۳ اجزای الگوریتم جست‌وجوی ممنوع

اجزای الگوریتم جست‌وجوی ممنوع عبارتند از:

- جواب اولیه [۵]
- مفهوم همسایگی [۶]
- لیست ممنوع [۷]

[۴] Tabu tenare
[۵] Initial solution
[۶] Neighborhood concept
[۷] Tabu list

- معیارهای آرمانی [۸]
- شروط توقف [۹]
- تقویت [۱۰] و تنوع [۱۱] در فضای جواب
- استراتژی فهرست کاندید
- حافظه‌ها در جست‌وجوی ممنوع

جواب اولیه

همان‌گونه که قبلا بیان کردیم، الگوریتم جست‌وجوی ممنوع برای شروع به یک جواب اولیه نیاز دارد که اگر این جواب اولیه، به صورت مناسب انتخاب نشده باشد، کارآیی الگوریتم پایین می‌آید. این یکی از ایرادهای این الگوریتم می‌باشد. جواب اولیه یا به‌صورت تصادفی یا با یک روش ابتکاری مانند روش حریصانه تولید می‌شود.

مفهوم همسایگی

بسیاری معتقدند یکی از خصوصیات بسیار مهم در الگوریتم جست‌وجوی ممنوع تعریف همسایگی مناسب است. از اولین مفاهیمی که در الگوریتم جست‌وجوی ممنوع می‌باید به آن پرداخت، مفهوم همسایگی است. در هر تکرار، انتقالی یا حرکتی بر روی جواب اعمال می‌شود مجموعه‌ای از جواب‌ها را در فضای جستجو تعریف می‌کند که جواب‌های همسایه گفته می‌شود. همسایگی، زیر مجموعه‌ای از فضای جواب است به این‌صورت که مجموعه جواب‌هایی که با استفاده از یک انتقال بدست می‌آید تعریف می‌شود. چنانچه از تعریف بر می‌آید ساختار همسایه می‌تواند حتی شامل تمامی فضای جواب نیز باشد. مشهورترین و ساده‌ترین روش‌هایی که برای ایجاد همسایگی در روش‌های پیشرفته ابتکاری استفاده می‌شوند عبارتند از جابجایی و تعویض. برای یک مساله خاص نوع حرکت تعریف شده نقشی اساسی در وسعت همسایگی به‌وجود آمده دارد.

لیست ممنوع

انتخاب نوع مناسب لیست ممنوع به مساله مورد بررسی بستگی دارد. در الگوریتم جست‌وجوی ممنوع، می‌توان تنها یک لیست ممنوعه برای ثبت حرکت‌های اخیر ایجاد نمود که این لیست از پدیدار شدن مجدد یک جواب در تعداد تکرارهای معین شده جلوگیری می‌نماید. معمولا اگر اندازه همسایگی به قدر

[۸] Aspiration criteria
[۹] Stopping criteria
[۱۰] Intensification
[۱۱] Diversification

کافی کوچک باشد به‌گونه‌ای که بتوان یک مشخصه اطلاعاتی از هر حرکت را (که در تعریف محدودیت ممنوع مورد استفاده قرار می‌گیرد) ذخیره کرد، ذخیره کردن شمارنده تکرار ارزشمند است، چرا که مشخص می‌سازد چه وقت محدودیت ممنوع مربوط به چنین مشخصه‌ای را می‌توان نادیده گرفت. این کار امکان تست ممنوع بودن حرکت در زمان ثابت را فراهم می‌سازد (با توجه به این که آیا این حرکت مشخصه ممنوع دارد یا نه). در این حالت حافظه مورد نیاز به اندازه همسایگی و نه به اندازه لیست ممنوع بستگی خواهد داشت.

به طور کلی لیست‌های ممنوعی که نسبت به حذف سیکل‌هایی با طول متناسب با اندازه لیست ممنوع تضمین می‌دهند، نتایج خوبی ارائه داده‌اند. در لیست‌های ممنوع مورد استفاده برای مسایل خیلی بزرگ (یا مسایل با تعاریف پیچیده برای مشخصه‌ها) مثل مساله فروشنده دوره‌گرد شاید امکان ذخیره یک آیتم اطلاعاتی برای هر حرکت وجود نداشته باشد. در این حالت مشخصه‌ها می‌توانند در لیست‌های ممنوع متوالی ذخیره شوند.

مفهوم اساسی در جست‌وجوی ممنوع مجاز دانستن جواب‌هایی است که در تابع هدف بهبود ایجاد نمی‌کنند، اما ممکن است ما را به سمت جواب سراسری راهنمایی کنند با این شرط که در لیست جواب‌های ممنوع قرار نداشته باشند. اما جست‌وجوی ممنوع برای استفاده از چنین راه حلی نیازمند آن است که از پدیده دور شونده که ناشی از بازگشت به جواب‌های پیشین است، جلوگیری کند. این حرکت‌های ممنوع برای مدتی معین در حافظه کوتاه مدت [۱۲] ذخیره می‌شوند که انجام مجموعه‌ای معین از حرکت‌ها را ممنوع می‌سازند. این مدت معین بنا بر الگوریتم حل، نوع مساله و ماهیت حرکت‌ها متغیر است. حافظه مورد استفاده برای نگهداری حرکت‌های ممنوع معمولا گردشی و دارای طول ثابت است. نشان داده شده است که لیست‌های ممنوعه‌ی با طول ثابت، نمی‌توانند همواره از ایجاد چرخه جلوگیری کنند و برخی از صاحب‌نظران تغییر طول لیست ممنوعه را در طول جست‌وجو پیشنهاد کرده‌اند. روش دیگر، تولید تصادفی مدت زمان‌های ممنوعه با بازه‌ای مشخص برای هر حرکت است. برای استفاده از این رویکرد، روشی متفاوت برای ذخیره‌سازی حرکت‌های ممنوعه مورد نیاز است که معمولا به‌صورت برچسب در یک آرایه ذخیره می‌شوند (این آرایه، معمولا تعداد تکرارهایی که هر کدام از حرکت‌ها، ممنوعه هستند را نیز ذخیره می‌کند).

به‌طور تجربی اندازه لیست‌های ممنوع که نتایج خوبی ارایه می‌کنند، اغلب با افزایش اندازه مساله رشد می‌کند. البته هیچ قاعده مشخصی (حتی به‌طور تجربی) نمی‌تواند اندازه مناسب لیست ممنوع برای مسایل مختلف را تعیین کند. این موضوع اساسا به خاطر این است که اندازه مناسب لیست بستگی به چگونگی محدودیت‌های ممنوع بکارگرفته شده دارد (محدودیت‌های قویتر عموما با اندازه‌های کوچک‌تر همراه هستند). راه تشخیص اندازه مناسب لیست برای یک مساله و انتخاب محدودیت‌های ممنوع، به‌طور ساده ایجاد تعادل بین امکان وقوع سریع سیکل (به خاطر کوچک بودن اندازه لیست) و بدتر شدن جواب (به خاطر بزرگ بودن اندازه لیست و در نتیجه منع شدن بسیاری از حرکت‌ها) است. بهترین اندازه در دامنه‌ای بین دو حد قرار می‌گیرد. در حقیقت بهترین رویکرد ، امکان تغییر اندازه لیست در این دامنه میانی است.

[۱۲] Short term memory

معیارهای آرمانی

هدف در این حالت انجام یک عمل ممنوع است که باعث ایجاد بهبود چشمگیری در نتیجه شود. تحت شرایطی بعضی از لیست‌های ممنوع بسیار قوی هستند. این شرایط ممکن است از حرکت‌های جذاب و خوب جلوگیری کنند و این در حالی است که ممکن است هیچ خطری جهت افتادن در دور وجود نداشته باشد و یا ممکن است این حرکت‌ها باعث بهبود کلی جست‌وجو شوند.
از این رو لازم است که از الگوریتمی استفاده کنیم تا ممنوع بودن را برای آن مورد خاص لغو کند و اجازه دهند آن متغیر وارد جست‌وجو شود. اگر هیچکدام از اعمال غیر ممنوع پیشرفت نداشته باشند در این حالت طبیعی است که عمل ممنوع انجام شود چون هدف بهینه‌سازی است. به این صورت که اگر عمل ممنوعی وجود داشته باشد که بهترین نتیجه ممکن را داشته باشد یعنی بهتر از همه اعمال غیر ممنوع باشد از آن جواب استفاده می‌شود. پذیرش این معیار را می‌توان احتمالی کرد که با یک احتمال انجام شود به‌عنوان مثال اگر ۲۰ درصد عمل ممنوع بهتر از اعمال غیر ممنوع باشد از این عمل ممنوع استفاده شود. معیار آرمانی با یک احتمال فعال می‌شود.
ساده‌ترین و پر استفاده‌ترین معیار آرمانی که تقریبا در تمام جست‌وجوهای ممنوع استفاده می‌شود، به این صورت است که یک حرکت ممنوع اگر به جوابی بهتر از جواب بهینه فعلی منجر گردد باید آن حرکت انجام شود یعنی ممنوعیت آن لغو می‌گردد (زیرا قبلا چنین جوابی مشاهده نشده است).

- معیار ۱: یک جواب باید از لیست ممنوع خارج شود اگر از هر جوابی که قبلا بدست آمده براساس مقدار تابع هدف بهتر باشد.
- معیار ۲: هرگاه ساختار روش و لیست‌های ممنوع در مرحله‌ای از فرآیند تکرار به‌گونه‌ای باشد که امکان هیچ گونه حرکتی وجود نداشته باشد در این صورت حرکتی که در صف خارج شدن از لیست ممنوع از همه به خروج نزدیک‌تر است انتخاب و ممنوعیت آن برداشته می‌شود.

شروط توقف

شروط توقف پر کاربرد در جست‌وجوی ممنوع عبارتند از :

۱. شرط توقف بعد از تعداد تکرار مشخص به‌عنوان مثال پس از ۵۰۰ یا ۱۰۰۰ تکرار.

۲. پس از تعدادی تکرار بدون اینکه جواب تابع هدف تغییر کند (این مورد بیشترین استفاده را تا به‌حال داشته است).

۳. وقتی که جواب به یک حد آستانه‌ای و مقدار از پیش تعیین شده برسد.

تقویت و تنوع در جواب

عبارت تقویت درجست‌وجو ممنوع به مکانیسمی اطلاق می‌شود که براساس آن جواب‌ها و حرکت‌هایی که باعث رسیدن به جواب‌های خوب شده‌اند، تقویت می‌شود. به‌عبارت بهتر، این استراتژی به بازگشت

به جواب‌های نخبه و جست‌وجوی بیشتر در محدوده آن‌ها اشاره دارد. از آنجا که جواب‌های نخبه برای مقایسه‌های بعدی مورد نیاز است، نحوه پیاده‌سازی این مفهوم شامل در نظر گرفتن یک حافظه‌ی مشخص برای این مجموعه از جواب‌ها است. استراتژی‌های تقویت به ابزاری برای مشخص کردن مجموعه‌ای از جواب‌های نخبه نیاز دارند تا پایه‌ای برای ترکیب خصوصیات خوب جهت ساخت جواب‌های تازه در دست داشته باشند. عضویت در این مجموعه از جواب‌ها اغلب با در نظر گرفتن یک آستانه برای مقدار تابع هدف انجام می‌پذیرد. تنوع در جست‌وجوی ممنوع یک مکانیسم است که کوشش می‌کند این مشکل را با انتقال جست‌وجو به ناحیه‌هایی که کمتر مورد بررسی قرار گرفته‌اند و یا نگرفته حل کند. دو روش عمده برای تنوع وجود دارد:

- شروع دوباره: جست‌وجو را از مناطقی که کمتر مورد بررسی قرار گرفته‌اند و یا از جواب‌های بهینه‌ای که قبلا پیدا کرده است دوباره شروع می‌کند (نقاط دیگر را داخل لیست ممنوع قرار می‌دهد).
- هدایت جست‌وجو: قرار دادن مکانیسمی در فرآیند جست‌وجو که کار رعایت پراکندگی را از همان ابتدا در دستور کار قرار می‌دهد.

یکی از مهمترین مشکلات مبتنی بر جست‌وجوی محلی گرایش این روش‌ها به محلی بودن است. آن‌ها تمایل دارند تا بیشتر در یک منطقه محدود به جست‌وجو بپردازند. از نکات منفی که می‌توان در این روش‌ها به آن رسید این است که هر چند ممکن است جواب‌های خوبی پیدا شوند اما ممکن است تمام ناحیه جواب بررسی نشود و ممکن است جواب بهینه‌ای پیدا شود که از جواب فعلی پیدا شده بسیار بهتر باشد.

استراتژی فهرست کاندید

در یک جست‌وجوی ممنوع عادی، برای حرکت از جواب فعلی به یک جواب همسایه، باید مقدار تابع هدف برای هر عنصر از همسایه‌ها ارزیابی شود. این کار می‌تواند از لحاظ محاسباتی بسیار هزینه‌بر باشد. روش دیگر، این است که به‌جای اینکه تمام همسایگی‌ها بررسی شود، تنها یک زیر مجموعه تصادفی از همسایگی‌ها در نظر گرفته شود، در نتیجه هزینه محاسباتی به‌طور قابل ملاحظه‌ای کاهش می‌یابد. انتخاب زیر مجموعه‌ای از جواب‌های همسایه به‌صورت تصادفی می‌تواند به‌عنوان یک مکانیزم ضد چرخه عمل کند. این کار اجازه می‌دهد که فهرست ممنوعه‌ی کوچکتری نسبت به کل همسایگی استفاده شود. البته باید در نظر داشت این کار یک عیب مهم دارد و آن احتمال از دست دادن جواب‌های خوب می‌باشد. بنابراین احتمال‌هایی را نیز می‌توان بکار برد تا معیارهای ممنوعه فعال شود.

حافظه‌ها در جست‌وجوی ممنوع

در جست‌وجوی ممنوع دو نوع حافظه کوتاه مدت و بلند مدت مورد استفاده قرار می‌گیرد.

- حافظه کوتاه مدت : یکی از مشکلات روش‌های جست‌وجوی محلی، تکرار جست‌وجوهای قبلی می‌باشد. به عبارت دیگر، ممکن است یک الگوریتم جست‌وجوی محلی، یک نقطه جواب را بارها جست‌وجو نماید. در این حالت، الگوریتم در طی روند جست‌وجوی خود، چندین بار از یک نقطه عبور می‌نماید. محاسبه مجدد یک نقطه کاری بیهوده بوده و موجب می‌گردد که بخشی از جست‌وجوی الگوریتم بیهوده شده و کیفیت الگوریتم کاهش یابد. از طرفی، همان‌طور که قبلاً نیز شرح داده شد، این مشکل می‌تواند حتی منجر به افتادن الگوریتم در بهینه محلی شود.
یک راه برای جلوگیری از جست‌وجوی تکراری، ثبت کلیه جواب‌های جست‌وجو شده و مقایسه جواب جدید با جست‌وجوهای قبلی می‌باشد. با ادامه روند جست‌وجو، این لیست خیلی طولانی شده و نیاز به حافظه بزرگی دارد. از طرفی، مقایسه یک جواب جدید با لیست کلیه جست‌وجوهای قبلی، نیاز به زمان بسیار زیادی دارد و در مجموع این روش، روش بیهوده و زمان‌بری است. در این حالت، حجم حافظه مورد نیاز و زمان مورد نیاز برای مقایسه با جواب‌های قبلی، با تعداد جواب‌های گذر شده ارتباط خطی داشته و با افزایش تعداد جواب‌ها، این حجم و زمان مورد نیاز محاسبه نیز به شدت افزایش خواهد یافت.
می‌توان به جای کلیه نقاط عبوری، مجموعه‌ای از نقاطی را که به تازگی از آن‌ها گذر شده است، در حافظه سپرد و هر جواب جدید را با آن مقایسه نمود. این روش نیز کارا نیست، زیرا سپردن تعداد کمی نقطه (جواب) ممنوعیت بسیار کمی را برای جست‌وجوی همسایگی به وجود می‌آورد و امکان برگشت مجدد به بهینه محلی و جست‌وجوی نقاط قبلی زیاد است. در صورتی که طول لیست کوتاه باشد، به دلیل ممنوعیت کم، خیلی مؤثر نبوده و در صورتی که طول لیست زیاد باشد، به دلیل نیاز به حافظه بزرگ و از طرفی زمان طولانی مورد نیاز برای قیاس جواب جدید با جواب‌های قبلی، روشی ناکارآمد می‌باشد.

- حافظه بلند مدت : در مواردی که همسایه‌ها بر اساس یک مجموعه ثابت از حرکت‌ها، به عبارت دیگر نوع حرکت به جواب جاری وابسته نباشد، تعیین می‌شوند، آمار حرکت‌های انتخابی در طول جست‌وجو می‌تواند جالب توجه باشد. اگر بعضی از حرکت‌ها خیلی بیشتر از سایر حرکت‌ها تکرار شوند، ضروری است فرض کنیم که جست‌وجو با مشکلاتی در ارتباط با ایجاد گوناگونی در فضای جست‌وجو مواجه شده است و ممکن است جست‌وجو در یک دره پهناور گیر بیافتد. در صورتی که فضای جواب را دو بعدی فرض کنیم و مقدار تابع هدف را به صورت ارتفاع آن فرض کنیم، ممکن است مجموعه جواب شبیه به مجموعه‌ای از دره‌ها و کوه‌ها دیده شود. در عمل، مسائل خیلی زیادی وجود دارند که شامل دره‌های متعدد و پهناوری هستند. گیر افتادن در دره، زمانی اتفاق می‌افتد که دامنه حرکت‌ها نسبت به خود مجموعه جواب یا تابع هدف، بسیار کوچک باشد. در چنین مواقعی، استفاده از لیست ممنوع شامل تعداد کمی از معکوس حرکت‌هایی که اخیراً انجام شده است، به عنوان تنها ابزار برای جهت‌دهی به جست‌وجو، در اغلب موارد، برای جلوگیری از محدود شدن در بعضی از دره‌ها کافی نیست. این در حالی است که اگر هم تعدادحرکت‌های ممنوع را افزایش دهیم، ممکن است که جست‌وجو در دامنه‌ها انجام گردد و به دلیل حرکت‌های ممنوع فراوان عمق دره‌ها را پیدا نکند. به عبارت دیگر، با افزایش طول لیست ممنوع ممکن است که

الگوریتم، از یک دره خارج گردد و به دره دیگری وارد شود ولی در مجموع، امکان یافتن عمق دره‌ها کم است. در نتیجه، ضروری است که از مکانیزم‌های دیگری برای جهت‌دهی به جست‌وجو در بلند مدت استفاده نماییم.

به منظور اطمینان از وجود گوناگونی[۱۳] در طول جست‌وجو، بدون استفاده از تعداد زیادی از حرکت‌های ممنوع، یکی از تکنیک‌ها، استفاده از جریمه برای حرکت‌هایی است که خیلی بیشتر از بقیه اتفاق افتاده‌اند. روش‌های مختلفی را می‌توان برای تخصیص جریمه معرفی کرد. به طور مثال، ممنوعیت برای حرکت‌هایی که بیشتر از حد آستانه اتفاق افتاده‌اند یا افزایش جریمه‌ای متناسب با فرکانس هر حرکت، در زمان انتخاب حرکت‌ها، می‌تواند روش‌هایی از تخصیص جریمه باشد. در چنین مواقعی، جست‌وجو تمایل پیدا می‌کند که حرکت‌هایی را که کمتر استفاده شده‌اند با احتمال بیشتری مورد استفاده قرار دهد. این روش معمولاً به روش استفاده از حافظه مبتنی بر فرکانس[۱۴] مشهور است.

روش دیگری که ممکن است برای ایجاد حافظه بلند مدت استفاده گردد، اجبار برای انجام بعضی از حرکت‌ها[۱۵] می‌باشد. در این حالت، حرکت‌هایی که در تعداد زیادی از تکرارها، استفاده نشده‌اند، شناسایی شده و انجام آن‌ها، بدون توجه به کیفیت حرکت، اجباری می‌گردد. این روش به خروج از دره‌ها و ایجاد گوناگونی در جست‌وجو کمک می‌نماید.

مراحل الگوریتم جست‌وجوی ممنوع

۱. ایجاد جواب اولیه و ارزیابی آن.

۲. ایجاد فهرستی از عملیات مجاز (روش‌های تولید همسایه).

۳. تنظیم شمارنده همه عملیات برابر با صفر.

۴. انجام عملیات مجاز و غیر ممنوع (همه یا کسری از عملیات) و انتخاب بهترین جواب.

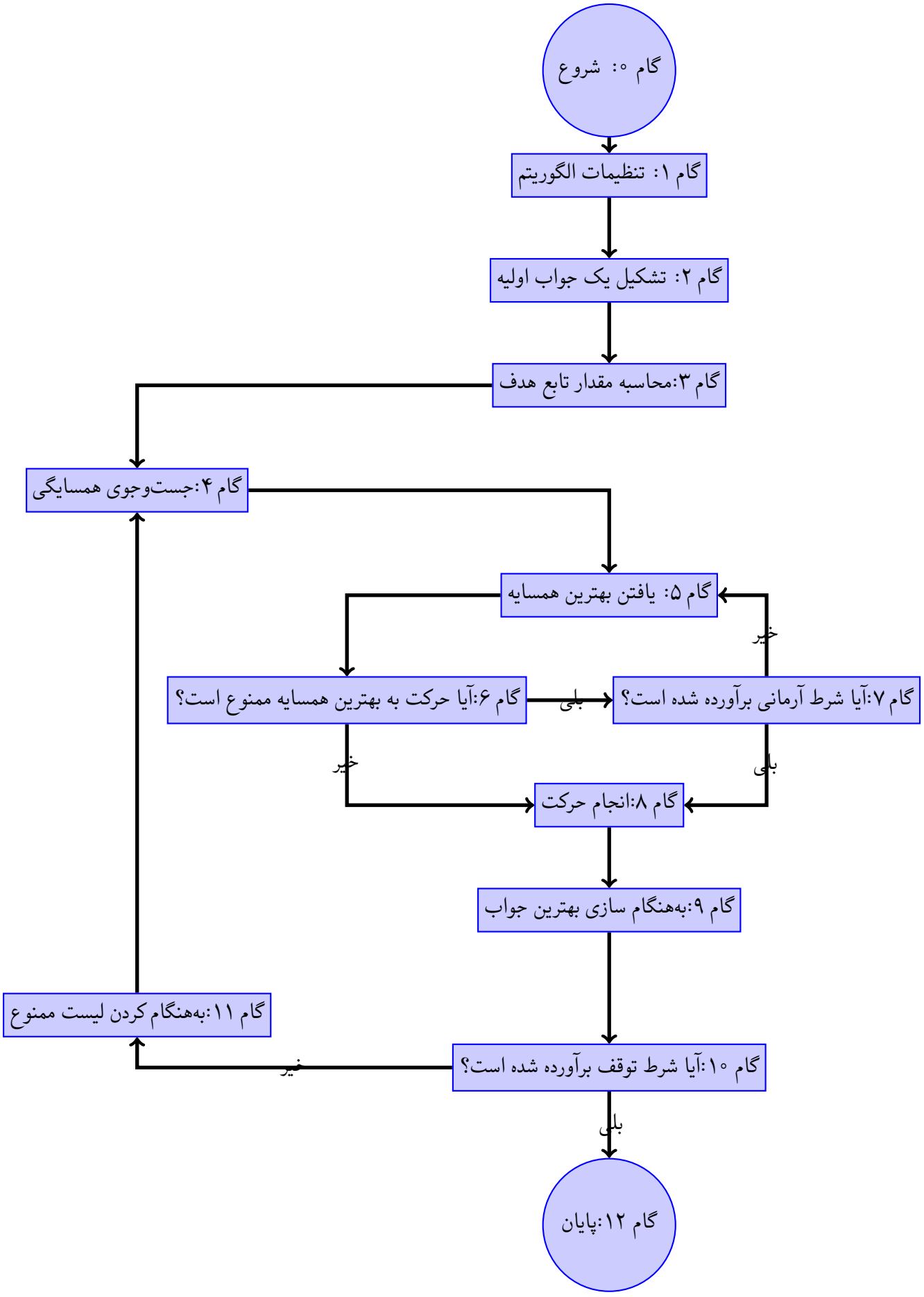
۵. اضافه کردن عملیات انجام شده به فهرست ممنوع.

۶. کم کردن یک واحد از مقدار شمارنده عناصر موجود در لیست ممنوع.

۷. به‌روز رسانی بهترین جواب یافته شده.

۸. بازگشت به مرحله ۴ وقتی شرایط خاتمه محقق نشده باشد.

برای اطلاعات مفصل‌تر، خواننده علاقمند می‌تواند به تحقیق گلوور و لاگونا ([۹]و [۱۵]) که در آن مروری بر جست‌وجوی ممنوع و کاربردهای آن بیان شده مراجعه نماید. شکل (۱.۳) مراحل اجرای الگوریتم جست‌وجوی ممنوع را نشان می‌دهد.



شکل ۱.۳: مراحل اجرای الگوریتم جست‌وجوی ممنوع

۳.۳ روش جست‌وجوی ممنوع برای حل مساله زمان‌بندی امتحانات

در مقاله ناجی عظیمی [۱۲] مساله زمان‌بندی امتحانات با استفاده از چهار روش ژنتیک، جست‌وجوی ممنوع، سرد شدن تدریجی و کلونی مورچگان مورد بررسی قرار گرفته است. در این بخش جست‌وجوی ممنوع ارایه شده در این مقاله را بیان می‌کنیم و در ادامه یک الگوریتم پیشنهادی جهت بهبود آن ارایه می‌کنیم.

۱.۳.۳ الگوریتم جست‌وجوی ممنوع ارائه شده در [۱۲]

جواب اولیه: منطقی است کیفیتی از جواب اولیه را انتظار داشته باشیم که بر جواب نهایی تاثیر دارد. در مقاله [۱۲] جواب اولیه تصادفی تولید میشود و روش‌های ابتکاری برای تولید آن استفاده نمی‌شود. این به ارزیابی روش‌های تحت مطالعه کمک خواهد کرد که تنها براساس شایستگی آن‌ها و مستقل از جواب اولیه می‌باشد.

نمایش جواب : هر جواب با یک بردار نشان داده می‌شود که طول بردار تعداد امتحانات است و هر عنصر این بردار بازه تخصیص داده شده به امتحان می‌باشد.

همسایگی جواب: همسایگی تعریف شده با تغییر تصادفی یک عنصر از بردار جواب بدست می‌آید یعنی به‌طور تصادفی بازه تخصیص داده شده به امتحان تغییر داده می‌شود. ممکن است همسایگی جواب یک جواب شدنی نباشد.

تابع هدف: تابع هدفی که برای ارزیابی جواب‌ها استفاده می‌شود رابطه (۳.۱) می‌باشد که در فصل اول آمده است و هدف مینیمم کردن هزینه است. ماتریس تداخل استفاده شده تصادفی بدست می‌آید که درایه سطر i و j تعداد دانشجویان مشترک در هر دو امتحان i و j را نشان می‌دهد. تعداد دانشجویان مشترک در هر دو امتحان در هزینه تعریف شده ضرب می‌شود.

همانطور که قبلا بیان شد جست‌وجوی ممنوع با یک جواب تصادفی شروع می‌شود و به یکی از همسایگی‌های خوب جاری حرکت می‌کند. هر وقت یک حرکت انجام می‌شود، جفت (امتحان و بازه) به لیست ممنوع که شامل حرکت‌های منع شده است اضافه می‌شود. به این معنا است که بازه امتحان تا زمانی که در لیست ممنوع است نمی‌تواند تغییر کند. از یک جواب داده شده، معمولا نمی‌توان تمام همسایگی‌های ممکن را بدست آورد. حرکت جدید که کاندید است در حقیقت جواب با بهترین همسایگی می‌باشد. اما اگر حرکت حاضر در لیست ممنوع باشد، تنها در صورتی حرکت جدید پذیرفته می‌شود که مقدار تابع هدف به سطح کمینه‌ای که تاکنون بدست آورده‌ایم برسد (سطح آرمانی) این فرآیند

[۱۳] Diversity

[۱۴] Frequency-based memory

[۱۵] Obligation to carry out move

تازمانیکه به شرط توقف برسیم تکرار می‌شود. شرط توقف این الگوریتم رسیدن به تعداد محدودی تکرار بین تکرار جاری و بهترین تکرار جواب بدست آمده می‌باشد.
الگوریتم جست‌و جوی ممنوع در [۱۲] به‌صورت زیر است.

- انتخاب جواب اولیه x_1 در X و $s := x_1$
- $nbiter := 0$ (تکرار جاری)، $bestiter := 0$ (تکراری که بهترین جواب یافت شده)، $bestsol := s$ (بهترین جواب) و $T := 0$ (لیست ممنوع).
- تابع آرمانی مقدار دهی اولیه شود.
- تا زمانی‌که $(f(s) > f^*)$ و $(nbiter - bestiter < nbmax)$ کارهای زیر را انجام دهید.

۱. $nbiter := nbiter + 1$
۲. مجموعه v^* از جواب‌های s_i که همسایگی هایی از جواب s هستند (بطوریکه ممنوع نباشند یا اگر ممنوع است سطح آرمانی رعایت شود) تولید کنید.
۳. یک جواب s^* از مجموعه v^* که کمترین f را داشته باشد انتخاب کنید.
۴. تابع آرمانی و لیست ممنوع را بروزرسانی کنید.
۵. اگر $f(s^*) < f(bestsol)$ آنگاه
$bestsol := s^*; bestiter := nbiter; s := s^*$
پایان.

مقادیر تخصیص داده شده به طول لیست ممنوع(L) که در [۱۲] استفاده شده در جدول (۱.۳) آمده است. به‌عنوان مثال اگر طول لیست ممنوع ۱۰ باشد یعنی تا ۱۰ تکرار نمی‌توان این حرکت را انجام داد و این مقدار کمی است برای طول لیست ممنوع زیرا بهتر است مقدار آن بیشتر باشد تا سایر حرکت‌ها نیز بررسی شود. پارامتر حافظه بلند مدت (G) استفاده نشده است. مقادیر حداکثر تعداد تکرار مجاز بدون بهبود در [۱۲] در جدول (۱.۳) آورده شده است.

جدول ۱.۳: طول لیست ممنوع

L	۱۰	۲۰	۳۰	$[N/3]$	$[N/2]$
حداکثر تعداد تکرار بدون بهبود		۵	۱۰	۲۰	۳۰

در [۱۲] از میان مقادیر پارامترها، مقادیر پارامترهایی که بهترین جواب را ارایه کرده‌اند برای جست‌وجوی ممنوع در جدول (۲.۳) آمده است.

جدول ۲.۳: بهترین مقادیر پارامترها

پارامتر	مقدار
طول لیست ممنوع	$[N/۳]$
حافظه بلند مدت	No
حداکثر تعداد تکرار بدون بهبود	۳۰

۲.۳.۳ الگوریتم پیشنهادی

برای پاسخگویی به نیازهای متغیر و متنوعی که در مسایل عملی وجود دارد الگوریتم بهینه‌سازی باید از انعطاف قابل توجهی برخوردار باشد. بسیاری از الگوریتم‌های توسعه یافته پژوهشگران برای مساله‌ای خاص طراحی شده‌اند.
همان‌گونه که در بخش قبل یادآوری شد، در میان روش‌های پیشرفته ابتکاری برای حل مسایل بهینه‌سازی، جست‌وجوی ممنوع و الگوریتم ژنتیک عملکرد بسیار مناسبی را از خود نشان داده‌اند. در این میان جست‌وجوی ممنوع دارای ویژگی‌هایی است که آن را از رویکرد دیگر متمایز کرده و کاندید مناسبی برای حل مسایل پیچیده می‌کند. سادگی، انعطاف‌پذیری، ثبات و کیفیت جواب‌ها ویژگی‌هایی است که توسط پژوهشگران بسیاری به‌عنوان ویژگی‌های شاخص جست‌وجوی ممنوع معرفی شده است. بر این اساس در طراحی الگوریتم پیشنهادی از چارچوب ارایه شده در جست‌وجوی ممنوع بهره گرفته شده است.
به‌صورت کلی می‌توان گفت که در چارچوب ارایه شده در جست‌وجوی ممنوع، به منظور ایجاد تنوع در فضای جست‌وجو دو نوع همسایگی بکار گرفته شده است. برای جلوگیری از قرار گرفتن در بهینه محلی، آخرین جواب‌های بدست آمده در برداری با عنوان لیست ممنوع ذخیره شده و انتخاب آن‌ها توسط الگوریتم برای مدتی ممنوع خواهد بود. الگوریتم جست‌وجوی ممنوع، برای مساله زمان‌بندی امتحانات با محدودیت‌هایی که در فصل اول گفته شده است را به‌صورت زیر بیان می‌کنیم:

- با جواب تصادفی که با یک بردار نشان داده می‌شود و طول بردار تعداد امتحانات است و هر عنصر این بردار بازه تخصیص داده شده به امتحان را نشان می‌دهد شروع می‌کنیم.

- برای بهبود جواب تا حد قابل توجهی از دو ابتکار زیر برای همسایگی استفاده می‌کنیم :
همسایگی معکوس[۱۶] : در این ابتکار بازه امتحانی برای امتحانات را دو به دو با هم جابجا می‌کنیم.
همسایگی معکوس[۱۷] : در ابتکار معکوس دو عدد تصادفی بین بازه $[۱, N]$ (Nتعداد امتحانات) انتخاب می‌کنیم. ترتیب بازه‌های تخصیص داده شده بین این دو عدد را معکوس می‌کنیم. با این ابتکار تخصیص چند بازه به امتحانات را تغییر می‌دهیم.

- هر همسایگی که بر جواب انجام می‌شود یک حرکت است و برای هر حرکت مقدار تابع برازش را محاسبه می‌کنیم.

[۱۶] Swap neighborhood
[۱۷] Inverse neighborhood

- سپس برای انتخاب جواب کاندید در تکرار بعد انتخاب تورنمنت را در نظر گرفته‌ایم که به‌صورت زیر می‌باشد:
انتخاب تورنمنت[۱۸] : جواب را با انتخاب تورنمنت با اندازه ۱۰۰ انتخاب می‌کنیم. به این صورت که از بین تمام جواب‌های ایجاد شده ۱۰۰ جواب تصادفی انتخاب کرده و از بین ۱۰۰ جواب بهترین را به‌عنوان جواب کاندید برای تکرار بعد انتخاب می‌کنیم.

- شرط توقف نیز تعداد تکرارها است از رابطه زیر که در آن N تعداد امتحانات و P تعداد بازه‌ها می‌باشد بدست می‌آید.

$$\text{تعداد تکرار} = N \times P$$

هر حرکت که انجام می‌شود وارد لیست ممنوع می‌شود که شامل حرکت‌های ممنوع است. به این معنی که بازه امتحان نمی‌تواند تغییر کند تا زمانیکه طول ممنوع برای آن حرکت صفر شود.

- تابع هدف برای ارزیابی جواب‌ها رابطه (۳.۱) است و هدف مینیمم کردن هزینه است. ماتریس تداخل را تصادفی بدست می‌آوریم که در فصل اول توضیح داده شده است.

در الگوریتم پیشنهادی دو نوع همسایگی تعویض و معکوس برای جواب تعریف شده است با این کار تمام همسایگی‌های ممکن را تولید می‌کنیم و با قرار دادن شرایطی در برنامه نوشته شده همسایگی‌هایی که تکراری هستند بررسی نمی‌شوند. اما در روش ارایه شده در مقاله [۱۲] یک نوع همسایگی تعریف شده است که با تغییر تصادفی بازه تخصیص داده شده به امتحان همسایگی را بدست می‌آوریم. در این حالت ممکن است جواب خوب را از دست بدهیم.
در الگوریتم پیشنهادی از انتخاب تورنمنت استفاده شده است که در انتخاب تورنمنت همانطور که توضیح داده‌ایم ممکن است جواب خوبی یافت شود و یا عکس آن. اما زمان اجرای برنامه با این انتخاب تا حد قابل توجهی کاهش می‌یابد. انتخاب ساده در روش ارایه شده در مقاله [۱۲] استفاده شده است. زمان اجرای برنامه با این انتخاب بسیار زیاد است زیرا باید جواب با کمترین مقدار برازندگی را از میان تمام جواب‌ها انتخاب شود. در مقاله [۱۲] از معیار آرمانی استفاده شده که در صورت ممنوع بودن حرکت خوب آن‌را انجام می‌دهیم.

۳.۳.۳ مثال پیاده سازی زمان‌بندی امتحانات برای روش جست‌وجوی ممنوع

مجموعه امتحانات، مجموعه بازه‌های زمانی، مجموعه دانشجویان و نام نویسی دانشجویان برای امتحانات را در نظر می‌گیریم. داده‌های لازم در جدول (۳.۳) آورده شده است. ستون دوم بازه‌هایی که امتحانات باید در آن‌ها برگزار شوند می‌باشد و ستون سوم دانشجویان هستند که باید با توجه به پیش‌ثبت نام آن‌ها امتحانات برنامه‌ریزی شود.

جدول ۳.۳: مشخصات مجموعه داده‌ها

دانشجویان	بازه‌های زمانی	دروس امتحانی
ب	شنبه ساعت ۱۸- ۱۶	شیمی
آ	شنبه ساعت ۱۶-۱۴	فیزیک
پ	شنبه ساعت ۱۲ – ۱۰	ریاضی
ج	شنبه ساعت ۱۰ – ۸	فارسی
		ورزش

نام نویسی دانشجویان در جدول (۴.۳) آمده است. به‌عنوان مثال در ستون اول دروس امتحانی را که دانشجوی آ ثبت نام کرده قرار داده شده و دانشجوی آ در دروس شیمی، ورزش، ریاضی و فارسی ثبت نام کرده است.

جدول ۴.۳: نام نویسی دانشجویان

ج	پ	ب	آ
شیمی	ورزش	شیمی	شیمی
فارسی	شیمی	فیزیک	ورزش
ورزش	فیزیک	ریاضی	ریاضی
	ریاضی		فارسی
	فارسی		

با استفاده از پیش‌ثبت نام دانشجویان ماتریس تداخل را بدست می‌آوریم که ماتریس متقارن از مرتبه ۵ می‌باشد زیرا ۵ امتحان داریم. به‌عنوان مثال عدد ۳ در سطر اول و ستون اول روی قطر اصلی تعداد دانشجویان که در امتحان اول نام نویسی کرده می‌باشد و عدد ۲ در سطر اول و ستون دوم تعداد دانشجویان مشترک در امتحان اول و دوم را نشان می‌دهد. ماتریس تداخل به‌صورت زیر می‌باشد.

$$\begin{bmatrix} ۳ & ۲ & ۳ & ۲ & ۲ \\ ۲ & ۲ & ۲ & ۱ & ۱ \\ ۳ & ۲ & ۴ & ۳ & ۳ \\ ۲ & ۱ & ۳ & ۳ & ۳ \\ ۲ & ۱ & ۳ & ۳ & ۳ \end{bmatrix}$$

[۱۸] Tournament selection

تکرار اول

جواب اولیه را به‌صورت تصادفی بدست می‌آوریم. مقدار برازندگی جواب با استفاده از رابطه (۳.۱) که در فصل اول آورده شده محاسبه می‌شود. اعداد صحیح ۱ تا ۴ بازه روز شنبه ساعت ۱۰-۸ و به همین ترتیب تا بازه روز شنبه ساعت ۱۸-۱۶ را نشان می‌دهند.

مقدار برازندگی	ورزش	فارسی	شیمی	فیزیک	ریاضی	
$F = ۷۹۶$	۱	۴	۴	۳	۲	جواب اولیه(s)

در جواب اولیه امتحان ریاضی روز شنبه ساعت ۱۲ − ۱۰، امتحان فیزیک روز شنبه بازه ۱۶-۱۴، امتحان شیمی روز شنبه ساعت ۱۸-۱۶، امتحان فارسی روز شنبه ساعت ۱۸-۱۶ و امتحان ورزش روز شنبه ساعت ۱۰ − ۸ می‌باشد. جواب اولیه(s) را به‌عنوان بهترین جواب در نظر می‌گیریم $s^* = s$، $f^* = ۷۹۶$ و $L = ۲$(طول لیست ممنوع). با استفاده از همسایگی تعویض همسایگی‌ها را برای جواب اولیه تولید می‌کنیم که همسایگی‌های جواب و مقدار برازندگی آن‌ها در جدول (۵.۳) آمده است. در همسایگی تعویض بازه امتحانی برای امتحانات را دو به دو با هم جابجا می‌کنیم.

جدول ۵.۳: همسایگی‌های جواب و مقادیر برازندگی‌ها

مقادیر برازندگی	ورزش	فارسی	شیمی	فیزیک	ریاضی	
$F = ۷۹۸$	۱	۴	۴	۲	۳	همسایگی ۱
$F = ۵۵۱$	۱	۴	۲	۳	۴	همسایگی ۲
$F = ۷۹۹$	۱	۲	۴	۳	۴	همسایگی ۳
$F = ۷۹۵$	۲	۴	۴	۳	۱	همسایگی ۴
$F = ۳۰۸$	۱	۴	۳	۴	۲	همسایگی ۵
$F = ۵۵۲$	۱	۳	۴	۴	۲	همسایگی ۶
$F = ۸۰۵$	۳	۴	۴	۱	۲	همسایگی ۷
$F = ۷۹۶$	۴	۴	۱	۳	۲	همسایگی ۸
$F = ۷۹۴$	۴	۱	۴	۳	۲	همسایگی ۹

حال از بین همسایگی‌ها جوابی که کمترین مقدار برازندگی را دارد به‌عنوان جواب کاندید برای تکرار بعد در نظر می‌گیریم. بنابراین همسایگی ۵ جواب کاندید برای تکرار بعد می‌باشد و $f^* = ۳۰۸$ است. چون طول ممنوع ۲ است برای دو تکرار نمی‌توانیم این همسایگی را به‌عنوان جواب انتخاب کنیم. شرط توقف را تعداد تکرار مشخص در نظر گرفته‌ایم و این تعداد تکرار ۴ می‌باشد، تا ۴ تکرار این کار را انجام می‌دهیم. بعد از ۴ تکرار جواب نهایی $f^* = ۳۰۵$ می‌باشد. در جواب اولیه بازه ۴ به دو امتحان شیمی

و فارسی تخصیص داده شده است و چون تعداد دانشجویان مشترک در امتحان فارسی و شیمی ۳ است مقدار تابع برازندگی بیشتر می‌باشد. در جواب نهایی بازه ۴ به دو امتحان فیزیک و فارسی تخصیص داده شده است و تعداد دانشجویان مشترک در این دو امتحان ۱ است به همین دلیل مقدار تابع هدف کمتر می‌باشد.

s^*	۲	۴	۱	۴	۳	$F = ۳۰۵$

بنابراین جدول زمان‌بندی امتحانات به‌صورت زیر است.

ورزش	فارسی	شیمی	فیزیک	ریاضی
شنبه ۱۲ – ۱۰	شنبه ۱۸-۱۶	شنبه ۱۰ – ۸	شنبه ۱۸-۱۶	شنبه ساعت ۱۶-۱۴

برنامه الگوریتم پیشنهادی و برنامه روش جست‌وجوی ممنوع ارایه شده در [۱۲] را با استفاده از زبان برنامه نویسی متلب ۲۰۱۰ نوشته شده است. برروی ۱۰ داده آزمایش انجام شده است و مشخصات مجموعه داده‌ها در جدول (۶.۳) آورده شده است.

جدول ۶.۳: مشخصات مجموعه داده‌ها

شماره	تعداد امتحانات	تعداد بازه‌ها	تعداد دانشجویان
۱	۲۰	۱۵	۱۰۰
۲	۴۰	۲۰	۳۰۰
۳	۲۶	۱۵	۳۰۰
۴	۲۶	۲۰	۲۰۰
۵	۲۶	۱۰	۳۰۰
۶	۳۰	۱۵	۲۵۰
۷	۲۶	۲۴	۱۵۰
۸	۶۰	۳۲	۴۰۰
۹	۵۰	۲۴	۸۰۰
۱۰	۴۴	۲۶	۴۰۰

در روش جست‌وجوی ممنوع پیشنهادی با طول‌های ممنوع مختلف ۳۰، ۵۰ و ۱۰۰ آزمایش کرده مقدار تابع هدف در سه مورد هم متفاوت بوده است اما با قرار دادن طول ممنوع ۱۰۰ تعداد حالت‌هایی که مقدار

تابع هدف با این طول بهتر از بقیه باشد بیشتر است، با افزایش دادن طول ممنوع می‌توانیم جواب‌های بیشتری را بررسی کنیم و در هر تکرار ممکن است جواب بهتری از تکرار قبل یافت شود. همچنین روش جست‌وجوی ممنوع ارایه شده در [۱۲] با طول ممنوع ۱۰۰ انجام شده است. روش جست‌وجوی ممنوع پیشنهادی زمان اجرای کمتری نسبت به روش جست‌وجوی ممنوع ارایه شده در [۱۲] دارد . مقدار تابع هدف جست‌وجوی ممنوع پیشنهادی نسبت به مقدار تابع هدف در روش جست‌وجوی ممنوع ارایه شده در [۱۲] کمتر می‌باشد. در روش ارایه شده در [۱۲] همسایگی تعریف شده مناسب نیست به همین دلیل مقدار تابع هدف بیشتر می‌باشد. نتایج عددی از مقدار تابع هدف و زمان اجرا در روش جست‌وجوی ممنوع پیشنهادی $Tabu$۱ و الگوریتم جست‌وجوی ممنوع ارایه شده در [۱۲] $Tabu$۲ به ترتیب در جدول (۷.۳) و (۸.۳) آورده شده است. در نمونه شماره ۸ زمان اجرا در الگوریتم ارایه شده در [۱۲] طولانی است به همین دلیل از انجام آن صرف‌نظر کرده‌ایم.

جدول ۷.۳: نتایج عددی تابع هدف

data	Tabu۱	Tabu۲
۱	۲۳۴٫۸۱	۳۳۹٫۹۹
۲	۲۴۷۰	۲۲۰۳
۳	۵۸۰٫۹۲	۹۴۵٫۸۱
۴	۹۸۵٫۶۲	۱۳۷۸٫۵۱
۵	۴۴۹٫۴۱	۹۰۹٫۱۸
۶	۳۳۱٫۶۳	۶۴۹٫۳۶
۷	۳۶۶٫۸۵	۴۴۲٫۶۹
۸	۴۰۸٫۲۸	-
۹	۵۵۷٫۳۲	۱۰۷۱٫۶
۱۰	۳۱۳٫۳۲	۹۸۲٫۳

جدول ۸.۳: نتایج عددی زمان اجرا برحسب ثانیه

data	Tabu۱	Tabu۲
۱	۱۶٫۴۳۷	۳۳٫۷۰۲
۲	۶۵٫۹۰۱	۷۸۶
۳	۲۶٫۵۵۹	۱۱۱٫۷۰۲
۴	۳۱٫۲۳۷	۱۰۸٫۹۱۹
۵	۱۲٫۳۹۹	۱۳۷٫۵۶۲
۶	۲۳٫۸۱۹	۱۸۹٫۶۴
۷	۲۹٫۷۸۲	۱۲۴٫۲۱
۸	۲۱۵٫۷۳۱	-
۹	۱۱۴٫۹۸۴	۹۲۸٫۰۴
۱۰	۹۸٫۲۱۹	۸۴۵

پیوست آ

کد Matlab

آ.۱ کد ژنتیک

```
clc
clear
m=44;
n=50;
p=26;
s=400;
%POP=randint(n,m,[1,p])
load dade10.m
POP=dade10;
load exa10.m
X=exa10;
cont=0
tic
for i=1:50
[RE,fitness]=REP(X,POP,m,n,p,s);
o=randperm(100);
O=o(1);
if O<=80
far=crossover(RE,m,n,p,s)
end
mu=MUTATION(far,m,n,p,s)
```

```
%end
% X=ESH(s,m);
X;
fitnes=zeros(1,n);
nk=1;
for q=1:2
for j=1:m
t(j)=far(q,j);
end
for i=1:m-1
for j=i+1:m
if 1<= abs(t(i)-t(j))&& abs(t(i)-t(j))<=5
w(i,j)=2^(5- abs(t(i)-t(j)));
else if t(i)==t(j)
w(i,j)= 1000;
else
w(i,j)=0;
end
end
end
end
w;
sum=0;
for j=1:m-1
for k=j+1:m
sum=sum+X(j,k)*w(j,k);
end
end
sum;
fitnes(q)=sum/s;
dov(nk)=fitnes(q);
nk=nk+1;
end
fitness(n+1)=dov(1);
```

```
fitness(n+2)=dov(2);
for i=1:2
POP(n+i,:)=far(i,:);
end
H=fitness
so1=sort(H);
PO=zeros(n,m);
az=0;
for i=1:n
for ja=1:n+2
if so1(i)==H(ja)
H(ja)=0;
az=az+1;
PO(az,:)=POP(ja,:);
break
end
end
end
cont=cont+1;
best(cont)=so1(1);
BEST=best;
POP=PO
st=sort(BEST);
star=st(1)
end
    toc
```

آ.۲ کد جست‌وجوی ممنوع با انتخاب تورنمنت

```
clc;
clear;
m=44;
p=26;
```

```
s=400;
load exa10.m
es=exa10;
load java10.m
X=java10;
z=cost(es,X,m,p,s);
actionlist=creatpermactionlist(m);
naction=numel(actionlist)
N=naction
TL=100;
maxit=m*p
sol=empty_individual;
sol.position=X
sol.cost=cost(es,sol.position,m,p,s)
Bestsol=sol
bestsol=sol
bestcost=zeros(maxit,1)
tc=zeros(naction,1);
tic
for it=1:maxit
bestnewsol.cost=inf;
for i=1:100
A=randperm(N,1)
if tc(i)==0
newsol.position=action(sol.position,actionlist{A});
newsol.cost=cost(es,newsol.position,m,p,s);
newsol.ActionIndex=A;
if newsol.cost<=bestnewsol.cost
bestnewsol.position=newsol.position
bestnewsol.cost=newsol.cost
bestnewsol.ActionIndex=newsol.ActionIndex
end
end
end
```

```
sol= bestnewsol
if sol.cost<Bestsol.cost
Bestsol.cost=sol.cost;
end
for i=1:naction
if i==sol.ActionIndex
tc(i)=TL;
else
tc(i)=max(tc(i)-1,0);
end
end
tc
end
CC=Bestsol.cost
toc
```

آ.۳ کد تابع پیش ثبت‌نام

```
function B=ESH(s,m)
M=enrolment(s,m)
for i=1:s
N{i}=M(i,:);
end
for j=1:m
S{j}=[];
end
for i=1:s
for j=1:m
if sum(ismember(N{i},j),2)==1
S{j}=[S{j},i];
end
end
end
for j=1:m
```

```
disp(S{j});
end
for i=1:m
for j=1:m
c=intersect(S{i},S{j});
B(i,j)=numel(c);
end
end
B
end
```

مراجع

[1] Beligiannisa, G. Moschopoulosa, C. Kaperonisa, G. and Likothanassisa, D. *Applying evolutionary computation to the school timetabling problem*, Journal Of Computer and Operations Research, vol. 35, (2008), pp. 1265-1280.

[2] Carter, M. W. Laporte, G. and lee, S. Y. *Examination timetabling: algorithmic strategies and application* , Journal Of The Operational Research Society, vol. 47, (1996), pp. 373-383.

[3] Carter, M. *A comprehensive course timetabling and student scheduling system at the university of waterloo*, Lecture Notes in Computer Science, vol. 2079, (2001), pp. 64-82.

[4] Cote, P. Wong, T. and Sabourin, R. *Application of a hybrid multi-objective evoulutionary algorithm to the uncapacitated exam proximity problem*, in: practice and theory of timetabling V, 5th International Conference, PATA 2004, Pittsburgh, PA, USA, 18-20 Agust, vol. 3616, (2004), pp. 294-312.

[5] Di Gaspero, L. and A, Scharf. *Tabu search techniques for examination timetabling* , in: Selected papers from third International confrance on the theory and practice of automated timetabling, Lecture Notes in Computer Science, vol. 2079 , Springer, Berlin, (2000), pp. 104-117.

[6] Daskalaki, S. and Birbas, T. *Efficient solutions for a university timetabling problem through integer programing* , European Journal Of Operational Research, vol. 160, (2005), pp. 106-120.

[7] Daskalaki, S. Birbas, T. and Housos, E. *An integer programing formulation for a cases study in university timetabling* , European Journal Of Operation Research, vol. 153, (2004), pp. 117-135.

[8] Eiben, A. E. Smith, J. E. *Introuduction to evolutionary computiong*, Springer, 2003.

[9] Glover, F. and Laguna, M. *Tabu search*, Kluwer Academic Publisher, (1997).

[10] Glover, F. *The general employee scheduling problem: An integeration of managment science and artificiul intelligence*, Computers and Operations Research, vol. 15, (1986), pp. 563-593.

[11] Mir Hassani, S. A. *Improving paper spread in examination timetabling using integer programing* , Applied Mathmatics and Computation, vol. 179, (2006), pp. 702-706.

[12] Naji Azimi, Z. *Hybrid heuristics for examination timetabling problem*, Applied Mathmatics and Computation, vol. 163, (2005), pp. 705-733.

[13] pillay, N. and Banzhaf, W. *An informed genetic algorithm for the examination timetabling problem*, Applied Soft Computing, vol. 10, (2010), pp. 457-467.

[14] Ramon, A. V. Enrice, G. and Jose, M. T. *A Tabu search algorithm to schedule university examination* , Journal Of Question, vol. 21, (1997), pp. 201-215.

[15] Rosing, K. E. Revelle, C. S. Rolland, D. A. Schilling, D. A. and Current, J. R. *Heuristic concentration and tabu search: head to head comarison*, European Journal of Operational Research, vol. 104, (1998), pp. 93-99.

[16] Sivanandam, S. N. and Deepa, S. N. *Introduction to Genetic Algorirhms*, Springer-Verlage Berlin Heidelberg, (2008).

Aabstract

In most educational centers, such as universities and schools, providing timetabling is a common activity. Exams scheduling of a department in universities and educational institutes is one of the problems which programmers face it. In this thesis, two methods of solving the problem of scheduling for exams are stated. First approach is based on genetic algorithm and the second one is tabu search. In both methods some solutions for improvement of these approaches are mentioned. In this thesis we have tried to decrease interference between exams and suggested the maximum time of studying for students. Programs of both methods have been written by Matlab 2010 and the numerical results of two approaches have been compared.

Technology University of Shahrood
Faculty Of Mathematical Sciences

Dissertation Submitted in Partial Fulfillment of The Requirements For The Degree of Master of Science in

Applied Mathematics

Examination Timetabling Problem

Supervisor

dr.Jafr Fathali

Advisor

dr.Mohsen Asili

by

Ailar Pouradineh

2013/21/12