دانشکده علوم ریاضی

گروه ریاضی کاربردی

پایان‌نامه کارشناسی ارشد

حل مسائل بهینه سازی به روش $GRASP$

سید احسان ترابی

استاد راهنما

دکتر جعفر فتحعلی

مرداد ۱۳۹۴

سپاس خدای را که سخنوران، در ستودن او بمانند و شمارندگان، شمردن نعمت‌های او ندانند و کوشندگان، حق او را گزاردن نتوانند.

تقدیم به پدر و مادر عزیزم

آنان که همواره بر کوتاهی و درشتی من، قلم عفو کشیده و کریمانه از کنار غفلت‌هایم گذشته اند و در تمام عرصه‌های زندگی یار و یاوری بی چشم داشت برای من بوده اند.

تقدیم به همسر مهربانم

که سایه مهربانیش سایه سار زندگیم می باشد، او که اسوه صبر و مقاومت بوده و مسیح وار با صبرش در تمامی لحظات رفیق راهم بوده. کسی که آفتاب را به زندگی‌ام هدیه نمود و فروزان زندگی‌ام شد و عاشقانه سوخت تا گرما بخش وجود و روشنگر راهم شود.

تقدیم به برادرم

که گنجینه دلگرمی، مهربانی و محبت است. کسی که با فداکاری و گذشت مرا در این امر خطیر همراهی و همیاری نمود.

سپاس گزاری...

با نام آنکه خلق کرد هستی را از رحمت بی انتهای خویش تا انسان در قدوم خلقتش به پهنای گسترده‌ای به بزرگی جهان خیمه زند و به او عقل داد تا عظمت خلقت را دریابد.
اکنون که به یاری خداوند متعال این دوره تحصیلی‌ام را به پایان رسانده‌ام، هر چند واژه‌ها را یارای آن نیست که لطف و محبت آنانی را که در تمام دوران زندگی‌ام جرعه نوش دریای مهر و محبتشان بوده‌ام به تصویر بکشم، اما به رسم ادب و احترام بوسه بر دستانشان زده و بر خود واجب می‌دانم زحمات بی‌شائبه پدر و مادر عزیزم، صبر و فداکاری همسر مهربانم و برادر نازنینم را که همواره راه گشای مشکلاتم در تمام مراحل زندگی بوده‌اند، ارج نهاده و مراتب تقدیر و تشکر قلبی و باطنی را از مهربانی‌های آنان ابراز دارم
با تمام وجود و در اوج فروتنی از مقام والا و شامخ استاد فرزانه جناب آقای دکتر جعفر فتحعلی که در نهایت صبر و بزرگواری تمامی سعی و تلاش خود را در جهت اعتلای واقعی ارزش‌های آموزشی در کالبد هدایت‌ها و رهنمودها نسبت به اینجانب مبذول فرمودند، کمال قدردانی را می‌نمایم و تشکر خالصانه از تمامی کسانی که مرا در به انجام رساندن این مهم یاری نموده‌اند، در برابر وجود گرامی‌شان زانوی ادب بر زمین می‌نهم.

سید احسان ترابی
مرداد ۱۳۹۴

تعهدنامه

اینجانب **سید احسان ترابی** دانشجوی کارشناسی ارشد رشته **ریاضی کاربردی** **دانشکده علوم ریاضی** دانشگاه شاهرود، نویسنده پایان‌نامه با عنوان **حل مسائل بهینه سازی به روش** $GRASP$ ، 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 شاهرود " یا " Shahrood University " 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.

سید احسان ترابی
مرداد ۱۳۹۴

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

حل مسائل بهینه سازی از جمله مسائل پر کاربرد و پر اهمیت در مسائل مهندسی و غیر مهندسی به شمار می‌آید. لذا حل اینگونه مسائل از اهمیت خاصی بر خوردار است. روش‌های مختلفی برای حل مسائل بهینه سازی ارائه شده است. الگوریتم‌های ابتکاری و فرا ابتکاری از جمله الگوریتم‌های حل اینگونه مسائل می‌باشند. الگوریتم‌هایی همچون جستجوی تابو ، جستجوی حریصانه ، الگوریتم‌های ژنتیک ، الگوریتم‌های اصلاح تکراری و جستجوی ابتکاری با حافظه محدود و ....

در این پایان نامه از یک روش مبتنی بر جستجوی حریصانه به نام روش $GRASP$ استفاده شده است. روش $GRASP$ نوعی جستجوی تصادفی حریصانه است که در حوضه مسائل بهینه سازی همچون علم مکان یابی و.... قرار دارد و دارای کاربردهای فراوانی در این علم میباشد. مزیت این روش نسبت به سایر روشهای حریصانه دیگر پیچیدگی زمانی کمتر با کیفیت بالای جواب برای مسأله را می‌توان ذکر کرد. در تمامی روش‌های ارائه شده برای حل و یافتن جواب بهینه در مسائل بهینه سازی تلاش شده است که زمان صرف شده کاهش و کیفیت و مطلوبیت جواب یا جواب ها افزایش یابد که در این خصوص روش $GRASP$ روشی با الگوریتمی بسیار قوی است که از لحاظ زمان اجرا و کیفیت جواب بدست آمده نسبت به روشهای دیگر بهتر و مفیدتر عمل می‌کند.

واژه نامه: الگوریتم‌های فراابتکاری، جستجوی حریصانه، جستجوی محلی، گرسپ، مسیرپیوستگی.

فهرست مطالب

لیست تصاویر ذ

لیست جداول ۱

۱ مقدمه و مفاهیم اولیه ۳

۱.۱ مقدمه ۳

۲.۱ الگوریتم‌های فراابتکاری در بهینه سازي ترکیبي ۴

۳.۱ تعاریف اولیه ۵

۴.۱ ویژگی‌های اساسی مشخص کننده فراابتکاری‌ها ۷

۵.۱ دسته بندی الگوریتم‌های فراابتکاری ۹

۶.۱ پیاده‌سازی الگوریتم‌های فراابتکاری ۱۰

۷.۱ انواع الگوریتم‌های بهینه سازی ۱۰

۲ روش فراابتکاری حریصانه ۱۳

۱.۲ مقدمه ۱۳

۲.۲ انواع جستجوی آگاهانه ۱۳

۱.۲.۲ الگوریتم نخست-بهترین ۱۳

۳.۲ معرفی روش فراابتکاری جستجوی حریصانه ۱۵

۱.۳.۲ ساختار روش حریصانه ۱۶

۲.۳.۲ ویژگی‌های جستجوی حریصانه ۱۷

۴.۲ کاربردهای روش حریصانه ۲۱

۱.۴.۲ مساله کوله پشتی ۲۱

۲.۴.۲ تولید درخت پوشای کمینه ۲۲

۳.۴.۲ محاسبه‌ی کوتاه ترین مسیر ۲۵

۴.۴.۲ کدگذاری و فشرده سازی اطلاعات ۲۷

۵.۲ روش‌های حریصانه برای حل مساله p-میانه ۲۷

۱.۵.۲ الگوریتم تصادفی ۲۹

۲.۵.۲ الگوریتم Rpg ۲۹
۳.۵.۲ الگوریتم $Rgreedy$ ۲۹
۴.۵.۲ الگوریتم $Pworst$ ۳۰
۵.۵.۲ الگوریتم $Sample$ ۳۰
۶.۲ مقایسه‌ی الگوریتم‌های سازنده روی مساله p-میانه ۳۰

۳ جستجوی محلی ۳۳
۱.۳ مقدمه ۳۳
۲.۳ جستجوی محلی ۳۳
۳.۳ ساختار الگوریتم جستجوی محلی ۳۴
۴.۳ زمینه های الگوریتم جستجوی محلی ۳۵
۵.۳ فضای واقعی جستجو ۳۵
۶.۳ کاربرد الگوریتم‌های جستجوی محلی ۳۶
۱.۶.۳ مسأله پوشش رأسی ۳۶
۲.۶.۳ مسأله فروشنده دوره گرد ۳۶
۳.۶.۳ مسأله رضایت بولی ۳۶
۴.۶.۳ مسأله زمان بندی پرستار ۳۶
۵.۶.۳ مسأله مکان یابی ۳۶

۴ روش جستجوي تطابقی تصادفی حریصانه ۳۹
۱.۴ مقدمه ۳۹
۲.۴ تاریخچه روش $GRASP$ ۳۹
۳.۴ ساختار روش $GRASP$ ۴۰
۴.۴ کاربرد های روش $GRASP$ ۴۲
۵.۴ فاز ساخت روش فرا ابتکاری $GRASP$ ۴۳
۶.۴ ساخت لیست اعضای داوطلب محدود RCL ۴۴
۷.۴ رفتار پارامتر α ۴۵

۵ مسیر پیوستگی ۵۱
۱.۵ مقدمه ۵۱
۲.۵ ساختار روش مسیر پیوستگی ۵۲
۳.۵ شبه کد های مربوط به روش مسیر پیوستگی ۵۲
۴.۵ استراتژی های مختلف مسیر پیوستگی ۵۳

آ نمایش کدهای استفاده شده ۵۷
آ.۱ شبه کد گرسپ با مسیر پیوستگی برای مسائل حداقل سازی ۵۷

آ.۲ شبه کد روش مسیر پیوستگی ۵۸
آ.۳ پالایش و تعمیم شبه کد فاز ساخت ۵۸
آ.۴ شبه کد فاز ساخت روش GRASP ۵۹
آ.۵ شبه کد GRASP برای مسائل مینیمم سازی ۵۹
آ.۶ شبه کد روش فرا ابتکاری GRASP ۵۹

مراجع ۶۱

واژه‌نامه فارسی به انگلیسی ۶۷

واژه‌نامه انگلیسی به فارسی ۶۹

لیست تصاویر

۱.۱ طبقه‌بندی انواع روش‌های بهینه سازی ۴

۱.۲ بازگرداندن ۳۶ سنت پول باقیمانده به خریدار به روش حریصانه با توجه به سکه‌های موجود. سکه‌های موجود از چپ به راست : ۲۵ سنتی ، ۱۰ سنتی ، ۱۰ سنتی ، ۵ سنتی ، ۱ سنتی ، ۱ سنتی ۱۹

۲.۲ بازگرداندن ۱۶ سنت پول باقیمانده به خریدار به روش حریصانه با توجه به سکه‌های موجود. سکه‌های موجود از چپ به راست : ۱۲ سنتی ، ۱۰ سنتی ، ۵ سنتی ، ۴ سکه ۱ سنتی ۲۰

۳.۲ گراف مربوط به احداث خط راه آهن شهری ۲۳

۴.۲ گراف‌های مربوط به استفاده الگوریتم کراسکال برای تشکیل درخت پوشای کمینه . ۲۴

۱.۴ توزیع مقادیر جواب فاز ساخت برای ارزش های مختلف پارامتر α از تابع RCL . ۴۶

۲.۴ : توزیع مقادیر جواب فاز جستجوی محلی برای ارزش های مختلف پارامتر α از تابع RCL ۴۷

۳.۴ انحراف استاندارد مقادیر یافت شده، بهترین و میانگین جواب و مجموع زمان سپری شده ۴۸

۱.۵ توزیع های زمانی مقادیر جواب برای $GRASP, GPRF, GPRB, GPRFB$ روی مثال $FR۷۵۰a$ ۵۵

۲.۵ توزیع های زمانی مقادیر جواب برای $GRASP, GPRF, GPRB, GPRFB$ روی مثال att ۵۵

۳.۵ : توزیع های زمانی مقدار جواب هدف برای $GPRB$ روی مثال att ۵۶

لیست جداول

۱.۲ نتایج بکارگیری الگوریتم‌های سازنده و بناکننده حریصانه روی مساله p–میانه . . . ۳۱

۱.۳ ماترس هزینه‌های سفر مساله مکان یابی سرویس دهنده‌ها ۳۷

۱.۴ ارزیابی و مقایسه استراتژی‌های مختلف مقادیر پارامتر α ۵۰

۱.۵ مقادیر جواب با زمان ثابت روی مثال att ۵۶

فصل ۱

مقدمه و مفاهیم اولیه

۱.۱ مقدمه

مسائل بهینه سازی [۱] از جمله مسائل پر کاربرد و پر اهمیت در مسائل مهندسی و غیر مهندسی به شمار می‌آید. لذا حل اینگونه مسائل از اهمیت خاصی برخوردار است. روش‌های مختلفی برای حل مسائل بهینه سازی ارائه شده است. الگوریتم‌های ابتکاری و فراابتکاری[۲] [۳۷] از جمله الگوریتم‌های حل اینگونه مسائل می‌باشند. الگوریتم‌هایی همچون جستجوی تابو [۳] [۱، ۴۸]، جستجوی حریصانه [۴] [۱۲] ، الگوریتم‌های ژنتیک [۳، ۱۱] [۵]، الگوریتم‌های اصلاح تکراری و جستجوی ابتکاری با حافظه محدود و غیره جز الگوریتم‌های ابتکاری و فراابتکاری می‌باشند. در این پایان نامه از یک روش مبتنی بر جستجوی حریصانه به نام روش $GRASP$ (رویه جستجوی تطابقی تصادفی)حریصانه استفاده شده است. روش $GRASP$ نوعی جستجوی تصادفی [۶] حریصانه است که در حل دسته‌ای از مسائل بهینه سازی[۹، ۱۳] همچون مکان یابی [۷] بکار می‌رود و دارای کاربردهای فراوانی در این علم می‌باشد. به عنوان مثال می‌توانیم حالت خاصی از مسائل بهینه سازی p – میانه و یا p – مرکز در نظر بگیریم که با استفاده از روش $GRASP$ براحتی قابل حل می‌باشد. مزیت این روش نسبت به سایر روش‌های حریصانه دیگر پیچیدگی زمانی کمتر با کیفیت بالای جواب برای مسأله می‌توان ذکر کرد. شکل (۱.۱) طبقه بندی انواع گوناگون روش‌های بهینه سازی را برای مسائل مختلف نشان می‌دهد. در تمامی روشهای ارائه شده برای حل و یافتن جواب بهینه در مسائل بهینه سازی تلاش شده است که زمان صرف شده کاهش و کیفیت و مطلوبیت جواب یا جواب‌ها افزایش یابد که در این خصوص روش $GRASP$، روشی

---

[۱] Optimization

[۲] Metaheuristic

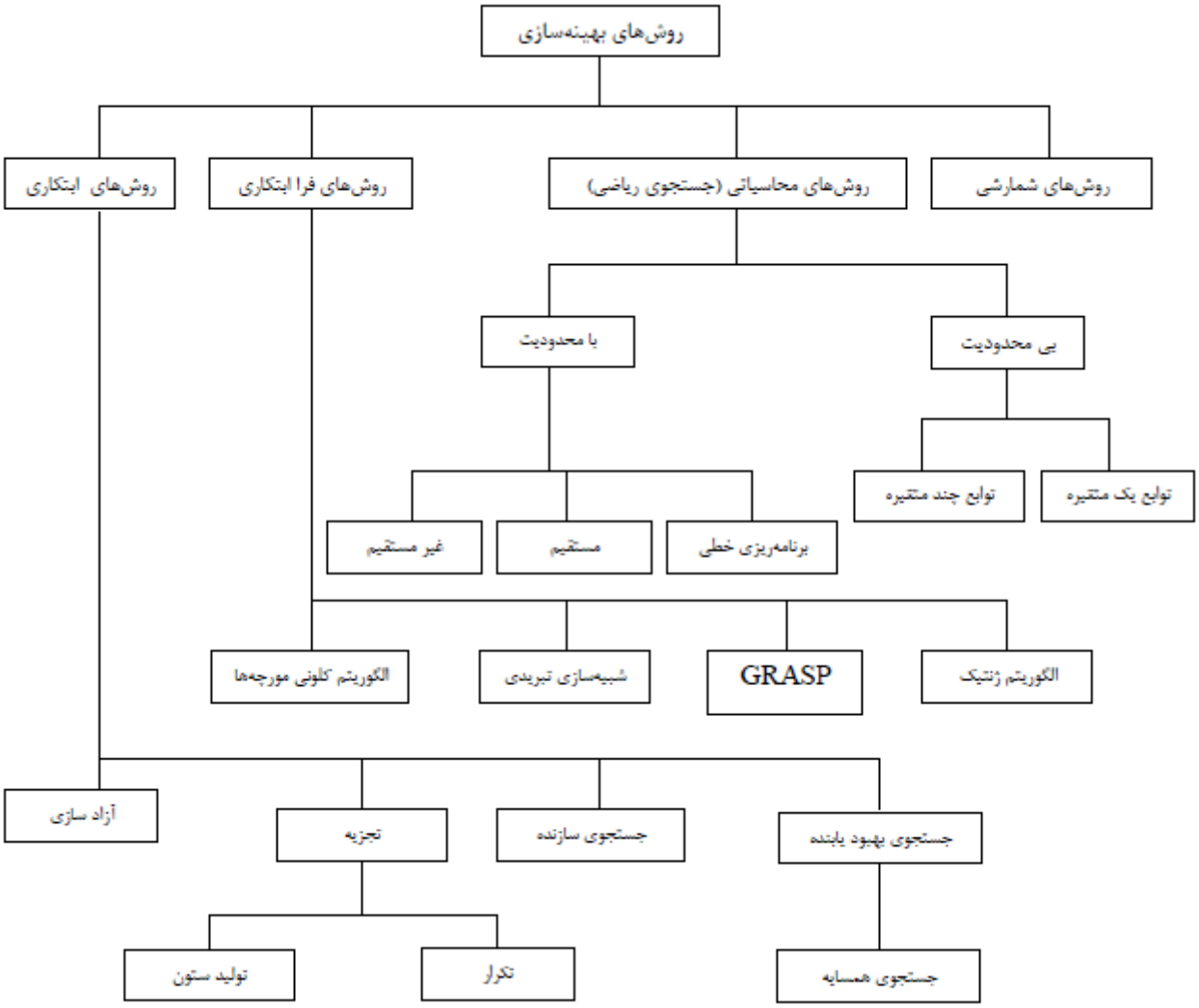
[۳] Tabu Search

[۴] Greedy Search

[۵] Genetic Algorithm

[۶] Random Search

[۷] Location



شکل ۱.۱: طبقه‌بندی انواع روش‌های بهینه سازی

با الگوریتمی بسیار قوی است که از لحاظ زمان اجرا و کیفیت جواب بدست آمده نسبت به سایر روش‌ها بهتر و مفیدتر عمل می‌کند.

۲.۱ الگوریتم‌های فراابتکاری در بهینه سازي ترکیبي

موضوع توابع فراابتکاري براي حل مسائل بهینه سازي ترکیبي زمینه‌ای تحقیقاتي است که با سرعت در حال رشد می‌باشد [۱۳]. این به دلیل اهمیت مسائل بهینه سازي ترکیبي در دنیاي صنعت و علم است. در اینجا مروري بر روش‌هاي فراابتکاري از دیدگاه مفهومي ارائه مي‌شود. مؤلفه‌ها و مفاهیم متفاوتي که در فراابتکارات مختلف براي تحلیل شباهت ها و تفاوت ها به کار مي‌روند، مطرح مي‌گردند. دو مفهوم بسیار مهم در فراابتکارات، متمرکزسازي [۸] و متنوع سازي [۹] است. این دو مفهوم، رفتار تابع فراابتکاري

[۸] Intensification

را تعیین مي كنند. اين دو از جهتي متناقض و از جهتي تكميل كننده هستند. با طرح مزايا و معايب رويكردهاي فراابتكاري متفاوت، اهميت تركيب فراابتكارات همچنين اجتماع فراابتكارات و روش‌هاي ديگر بهينه سازي مشخص مي‌گردد.

۳.۱ تعاریف اولیه

روش‌های حل بسياري از مسائل بهينه سازي عملي و نظري يا تئوری، شامل جستجوي بهترين پيكربندي، براي مجموعه‌ای ازمتغيرها تا رسيدن به هدف هاوجود دارند. به طور كلی اين مسائل به طور طبيعي ، به دو بخش مجزا تقسيم بندی مي‌شوند: مسائلي كه جواب آن‌ها با متغيرهاي داراي مقادير حقيقي كدگذاري شده و مسائلی كه با متغيرهاي گسسته كدگذاري شده‌اند. در ميان دسته دوم، كلاسي از مسائل ، به نام مسائل بهينه سازي تركيبي[۱۰] وجود دارد. در مسائل بهينه سازي تركيبي، ما در يک مجموعه متناهي[۱۱] يا نامتناهي [۱۲] شمارا به دنبال يک شئ مي‌گرديم، مانند يک عدد صحيح، زير مجموعه، جايگشت يا ساختار گراف [۲][۱۳].

تعریف ۱.۳.۱. يک مسأله بهينه سازي تركيبي $p = (S, f)$ شامل قسمت های زير است :

- مجموعه‌اي از متغيرها[۱۴] به صورت $X = \{x_1, x_2, \ldots, x_n\}$
- دامنه‌های متغيرها به صورت $D = \{D_1, D_2, \ldots, D_n\}$
- محدوديت بين متغيرها
- تابع هدف f كه بايد حداقل شود كه: $f : D_1, D_2, \ldots, D_n \to \mathbb{R}+$

مجموعه كل انتساب هاي ممكن به شكل زير است كه در آن S مجموعه كل جواب هاست:

$$S = \{s = \{(x_1, v_1), \ldots, (x_n, v_n)\} | v_i \in D_i, \quad s \text{ در محدوديت‌ها صدق كند}\}$$

معمولا Sرا فضاي جواب می‌نامند به نحوي كه هر عنصر از اين مجموعه می‌تواند يک جواب كانديد باشد.

براي حل يک مسأله بهينه سازي تركيبي، مي‌توان يک جواب $s^* \in S$ با كمترين مقدار تابع هدف را يافت كه براي هر s، $f(s^*) \leq f(s)$. در اين حالت، s^* جواب بهينه سراسري از (S, f) است و مجموعه S^* (كه زير مجموعه S مي‌باشد)، مجموعه جواب‌هاي بهينه سراسري[۱۵] ناميده مي‌شود.

---

[۹] Diversification

[۱۰] Combimatorial Optimization

[۱۱] Finite Set

[۱۲] Infinite

[۱۳] Graph

[۱۴] Variable

[۱۵] Global

به عنوان مثال‌هايي از مسائل بهينه سازي تركيبي، مسأله فروشنده دوره گرد[۱۶] [۱۴] مسأله انتساب درجه دوم[۱۷] ، مسائل جدول زماني و زمان بندي را مي‌توان نام برد.

با توجه به اهميت عملي مسائل بهينه سازي تركيبي ، الگوريتم‌هاي زيادي براي مهاركردن[۱۸] آن ها، توسعه يافته‌اند. اين الگوريتم‌ها مي‌توانند به صورت كامل يا تقريبي دسته بندي شوند. الگوريتم‌هاي كامل، تضمين مي‌كنند براي هر نمونه اندازه متناهي از مسأله بهينه سازي تركيبي ، جواب بهينه اي در زمان محدود يافت خواهد شد. هنوز، براي مسائل بهينه سازي تركيبي كه NP - سخت هستند الگوريتمي با زمان چند جمله اي وجود ندارد. پس روش‌هاي كامل ممكن است در بدترين حالت، نياز به زمان محاسبه نمايي داشته باشند. اين مسأله اغلب منجر به زمان‌هاي محاسبه خيلي طولاني براي اهداف عملي مي‌گردد. بنابراين استفاده از روش‌هاي تقريبي در طي ۳۰ سال گذشته بيشتر مورد توجه قرار گرفت. در روش‌هاي تخميني، ضمانت يافتن جواب بهينه، قرباني جستجوي جواب‌هاي خوب در زمان هاي بسيار كوتاه مي‌شود.

در ميان روش‌هاي تقريبي پايه، اغلب بين روش‌هاي سازنده[۱۹] و جستجوي محلي [۲۲، ۲۳] تفاوت قائل مي‌شويم. الگوريتم‌هاي سازنده با اضافه كردن اجزايي به يک جواب جزئي تهي اوليه، جواب‌هايي را از ابتدا توليد مي‌كنند تا وقتي كه جواب كامل شود. آنها، نوعا سريع‌ترين روش هاي تقريبي هستند. با اين حال، آن ها اغلب در مقايسه با الگوريتم‌هاي جستجوي محلي، جواب‌هايي با كيفيت پايين تر را برمي‌گردانند. الگوريتم‌هاي جستجوي محلي از جواب اوليه‌اي شروع مي‌كنند و به طور تكراري براي جايگزيني جواب فعلي با جواب بهتري در همسايگي فعلي تلاش مي‌كنند. همسايگي به شكل زير تعريف مي‌شود.

تعريف ۱.۳.۲. [۶۳] ساختار همسايگي تابعي است مانند : $N : S \to \mathbb{R}+$ كه به هر $s \in S$ مجموعه‌اي از همسايگي‌هاي S $N(s) \subseteq S$ نسبت مي‌دهد. $N(s)$ همسايگي S گفته مي‌شود.

معرفي ساختار همسايگي، ما را قادر به تعريف مفهوم جواب‌هاي كمينه محلي مي‌سازد.

تعريف ۱.۳.۳. جواب كمينه محلي (بهينه محلي) با توجه به ساختار همسايگي N، جوابي مانند $\tilde{s}$ است به نحوي كه به ازاي هر $s \in N(\tilde{s})$، $f(\tilde{s}) \leq f(s)$. $\tilde{s}$ را يک جواب كمينه محلي قوي [۲۰] مي‌ناميم اگر به ازاي هر $s \in N(\tilde{s})$، $f(\tilde{s}) < f(s)$ باشد.

در طي ۲۰ سال گذشته، نوع جديدي از الگوريتم تقريبي به وجود آمد كه اساسا تلاش مي‌كند روش‌هاي ابتكاري پايه را با هدف جستجوي كارا و موثر فضاي جستجو در چارچوب‌هاي سطح بالاتر تركيب كند. امروزه اين روش‌ها، فراابتكاري ناميده مي‌شوند. لغت فرا ابتكاري ابتدا توسط گلاوور [۴۹] ارائه شد كه از دو كلمه يوناني تشكيل شده است. ابتكاري[۲۱] به معناي يافتن و پيشوند فرا [۲۲] به معناي وراي، در

[۱۶] Travelling Salesman Problem
[۱۷] Quadratic Assignment Problem
[۱۸] Tackle
[۱۹] Constructive
[۲۰] Strict
[۲۱] Heuristic
[۲۲] Meta

سطحي بالاتر می‌باشد. قبل از توافق وسيع بر روي اين لغت، از عبارت ابتكارات جديد و تازه [۲۳] استفاده مي‌شد. اين رده از الگوريتم‌ها شامل بهينه سازي گروه مورچگان ACO[۲۴]، محاسبه تكاملي [۲۵] شامل الگوريتم‌هاي ژنتيک [۱۱] ، جستجوي محلي تكراري [۲۶] شبيه سازي تبريدی [۲۷] و جستجوي ممنوع مي‌باشد. تا به حال تعريف پذيرفته شده مشتركي براي فراابتكار ارائه نشده است. فقط در چند سال اخير بعضي محققين در اين زمينه، تعاريفي پيشنهاد كرده‌اند. مانند موارد زير:
فراابتكار به صورت فرآيند توليد تكراري تعريف مي‌شود كه يک تابع ابتكاري فرعي را با تركيب هوشمند مفاهيم متفاوت براي كاوش و استخراج فضاي جستجو، هدايت مي‌كند و در آن استراتژي‌هاي يادگيري براي اطلاعات ساختاري، جهت يافتن جواب‌هاي كاراي نزديک به بهينه، مورد استفاده قرار مي گيرند.
فراابتكار يک فرآيند اصلي [۲۸] تكراري است كه عمليات توابع ابتكاري فرعي را براي توليد كاراي جواب‌هاي با كيفيت بالا هدايت مي‌كند. اين فرآيند يک روش واحد كامل (يا ناكامل) يا مجموعه‌اي از جواب‌ها در هر تكرار را به كار مي‌گيرد. توابع ابتكاري فرعي مي‌توانند رويه‌هاي سطح بالا (يا پايين) در يک جستجوي محلي ساده يا فقط يک روش ساختاري باشند.
بنابراين فراابتكار، مجموعه مفاهيمي است كه مي تواند جهت تعريف روش‌هاي ابتكاري براي مجموعه بزرگي از مسائل متفاوت مورد استفاده قرار گيرد. به بيان ديگر، فراابتكار را مي‌توان به عنوان يک چارچوب الگوريتمي كلي در نظر گرفت كه مي‌تواند براي مسائل بهينه سازي مختلف، با اصلاحات نسبتا اندكي براي تطابق با مسأله خاص، به كار رود.

۴.۱ ویژگی‌های اساسی مشخص کننده فراابتکاری‌ها

روش‌های فراابتکاری دارای ویژگی‌های زیر هستند:

- فراابتكاری ها، استراتژي هايي براي راهنمايي فرآيند جستجو هستند.
- هدف، كاوش كاراي فضاي جستجو براي يافتن جواب‌هاي (نزديک به) بهينه است.
- تكنيک‌هاي شركت كننده در الگوريتم‌هاي فراابتكاري، در محدوده رويه‌هاي ساده جستجوي محلي تا فرايندهاي يادگيري پيچيده قرار مي‌گيرند.
- الگوريتم‌هاي فراابتكاري، تقريبي و عمدتا غيرقطعي [۲۹] هستند.

[۲۳] Modern Heuristic
[۲۴] Ant Colony Optimization
[۲۵] Evolutionary Computation
[۲۶] Iterated Local Search
[۲۷] Simulated Annealing
[۲۸] Master
[۲۹] Non - Deterministic

- ممکن است مکانیزم هایي براي اجتناب از به دام افتادن در نواحي محدود [۳۰] فضاي جستجو به کار ببرند.
- فراابتکارات، مخصوص مسأله خاصي نيستند.
- ممکن است از دانش خاص دامنه [۳۱] به شکل توابع ابتکاري که با استراتژي‌هاي سطح بالاتر کنترل مي‌شوند، استفاده کنند.
- امروزه ، فراابتکارات پيشرفته تر، تجربه جستجو را براي راهنمايي جستجو به کار مي‌برند.

به طور کلي مي‌توان گفت فراابتکارات، استراتژي‌هاي سطح بالا براي کاوش فضاي جستجو با استفاده از روش‌هاي متفاوت هستند. مسأله مهم اين است که توازن پويايي بين متنوع سازي و متمرکزسازي وجود دارد. به طور کلي لغت متنوع‌سازي به کاوش فضاي جستجو اطلاق مي‌گردد در حالي که لغت متمرکزسازي به کاوش تجربه جستجوي جمع‌آوري شده گفته مي‌شود. اين لغات از زمينهء جستجوي ممنوع [۱، ۴۸] ناشي مي‌شوند و لازم است روشن شود که لغات کاوش [۳۲] و استخراج [۳۳] گاهي به جاي هم به کار مي‌روند، در واقع کلمات استخراج و کاوش، اغلب به استراتژي‌هاي نسبتا کوتاه مدت وابسته به تصادفي بودن اطلاق مي‌شود، در حالي که متنوع سازي و متمرکزسازي، به استراتژي‌هاي ميان مدت يا بلند مدت که مبني بر استفاده از حافظه هستند، گفته مي‌شود. به کاربردن لغات متنوع سازي و متمرکزسازي در معني اصلي‌شان، در کل زمينه فراابتکارات، پذيرفته تر است. از يک سو به دليل شناسايي سريع مناطقي از فضاي جستجوي داراي جواب‌هاي کيفيت بالا و از سوي ديگر براي هدر ندادن زمان طولاني در مناطقي از فضاي جستجو که قبلا کاوش گرديده و يا جواب کيفيت بالايي فراهم نکرده اند، توازن بين متنوع سازي و متمرکزسازي داراي اهميت بسيار است .

استراتژي‌هاي جستجوي داراي فراابتکارات متفاوت، وابستگي زيادي به فلسفه خود فراابتکارات دارند. فلسفه‌هاي متفاوت فراواني در فراابتکارات موجود دارد. بعضي به عنوان توسعه‌اي هوشمند از الگوريتم‌هاي جستجوي محلي در نظر گرفته مي‌شوند. هدف از اين نوع فراابتکار، فرار از کمينه محلي براي پيشروي در کاوش فضاي جستجو و حرکت براي يافتن کمينه محلي اميدوارکننده‌تر است. اين مورد در جستجوي ممنوع [۴۸] ، جستجوي محلي تکراري [۳۳] ، جستجوي همسايگي متغير [۴۵] ، $GRASP$ و شبيه سازي تبريدي به کار مي‌رود. اين فراابتکارات (که روش‌هاي خط سير [۳۴] ناميده مي‌شوند) بر روي يک يا چند ساختار همسايگي اعمال شده روي اعضاي (جواب‌هاي) فضاي جستجو، کار مي‌کنند.

در مورد الگوريتم‌هايي مثل گروه مورچه ها و محاسبه تکاملي، فلسفه متفاوتي وجود دارد. آن ها مؤلفه يادگيريي را به کار مي‌برند که به طور صريح يا ضمني سعي مي‌کند ارتباط [۳۵] بين متغيرهاي تصميم گيري

[۳۰] Confined
[۳۱] Domain-specific knowledge
[۳۲] Digging
[۳۳] Exploitation
[۳۴] Trajectory
[۳۵] Correlation

را براي شناخت نواحي با كيفيت در فضاي جستجو، ياد بگيرد. اين نوع فراابتكارات، تا اندازه اي، نمونه برداري را از فضاي جستجو انجام مي‌دهد. براي نمونه در محاسبه تكاملي، اين مسأله توسط تركيب مجدد جواب‌ها و در بهينه سازي گروه مورچه ها با نمونه برداري از فضاي جستجو در هر تكرار برحسب يک توزيع احتمالي انجام مي‌شود. الگوریتم‌های فراابتکاری یا فراتکاملی نوعی از الگوریتم‌های تقریبی هستند که برای یافتن پاسخ نزدیک به بهینه به کار می‌روند. روش‌ها و الگوریتم‌های بهینه‌سازی به دو دسته الگوریتم‌های دقیق و الگوریتم‌های تقریبی تقسیم‌بندی می‌شوند. الگوریتم‌های دقیق قادر به یافتن جواب بهینه به صورت دقیق هستند اما در مورد مسائل بهینه سازی سخت کارایی ندارند و زمان حل آنها در این مسائل به صورت نمایی افزایش می‌یابد. الگوریتم های تقریبی قادر به یافتن جواب‌های خوب (نزدیک به بهینه) در زمان حل کوتاه برای مسائل بهینه سازی سخت هستند. الگوریتم های تقریبی نیز به سه دسته الگوریتم‌های ابتکاری و فراابتکاری و فوق ابتکاری بخش بندی می‌شوند. دو مشکل اصلی الگوریتم‌های ابتکاری، قرار گرفتن آنها در بهینه‌های محلی، و ناتوانی آن‌ها برای کاربرد در مسائل گوناگون است. الگوریتم‌های فراابتکاری برای حل این مشکلات الگوریتم های ابتکاری ارائه شده‌اند. در واقع الگوریتم‌های فراابتکاری، یکی از انواع الگوریتم‌های بهینه‌سازی تقریبی هستند که دارای راه کارهای برون رفت از بهینه محلی می‌باشند و قابل کاربرد در طیف گسترده‌ای از مسائل هستند. رده‌های گوناگونی از این نوع الگوریتم در ده‌های اخیر توسعه یافته است.

۵.۱ دسته بندی الگوریتم‌های فراابتکاری

معیارهای مختلفی می‌تواند برای طبقه‌بندی الگوریتم های فراابتکاری استفاده شود:

۱. **مبتنی بر یک جواب و مبتنی بر جمعیت** [۳۶] : الگوریتم های مبتنی بر یک جواب در حین فرایند جستجو یک جواب را تغییر می‌دهند، در حالی که در الگوریتم‌های مبتنی بر جمعیت در حین جستجو، یک جمعیت از جواب‌ها در نظر گرفته می‌شوند. از الگوریتم‌های متداول فراابتکاری مبتنی بر یک جواب می‌توان الگوریتم جستجوی ممنوعه و الگوریتم تبرید شبیه‌سازی شده و الگوریتم حریصانه و روش $GRASP$ را نام برد. روش‌هاي مبتني بر جمعيت ، در هر تكرار [۳۷] به جاي يک جواب ، مجموعه اي (جمعيتي) از جواب ها را مورد بحث قرار مي‌دهند. كارايي نهايي به شدت وابسته به روشي است كه روي جمعيت اعمال مي‌گردد. از الگوریتم‌های شناخته شده فراابتکاری بر پایه جمعیت می‌توان الگوریتم‌های تکاملی (الگوریتم ژنتیک، برنامه‌ریزی ژنتیک، ...)، بهینه‌سازی کلونی مورچگان ، کلونی زنبورها ، روش بهینه‌سازی ازدحام ذرات، الگوریتم رقابت استعماری ، و الگوریتم چکه آب های هوشمند را نام برد. دو روش مبتني بر جمعيت كه بيشتر مورد مطالعه قرار گرفته اند، محاسبه تكاملي و بهينه سازي كلوني مورچه (ACO) مي‌باشد. در الگوریتم‌هاي محاسبه تكاملي، با عمليات تركيب مجدد و جهش [۳۸] جمعيتي از افراد اصلاح

[۳۶] Population

[۳۷] Iteration

[۳۸] Mutation

مي‌گردند و در بهينه سازي كلوني مورچه گروهي از مورچه‌هاي مصنوعي براي ساخت جواب به كار مي‌روند كه با رديابي فرمون [۳۹] و اطلاعات تابع ابتكاري هدايت مي‌شوند.

۲. **الهام گرفته از طبیعت و بدون الهام از طبیعت**: بسیاری از الگوریتم‌های فراابتکاری از طبیعت الهام گرفته شده‌اند، در این میان برخی از الگوریتم‌های فراابتکاری نیز از طبیعت الهام گرفته نشده‌اند.

۳. **با حافظه و بدون حافظه**: برخی از الگوریتم‌های فراابتکاری فاقد حافظه می‌باشند، به این معنا که، این نوع الگوریتم‌ها از اطلاعات بدست آمده در حین جستجو استفاده نمی‌کنند (به طور مثال شبیه سازی تبریدی). این در حالی است که در برخی از الگوریتم‌های فراابتکاری نظیر جستجوی ممنوعه از حافظه استفاده می‌کنند. این حافظه اطلاعات بدست آمده در حین جستجو را در خود ذخیره می‌کند.

۴. **قطعی و احتمالی**: یک الگوریتم فراابتکاری قطعی نظیر جستجوی ممنوعه، مسأله را با استفاده از تصمیمات قطعی حل می‌کند. اما در الگوریتم‌های فراابتکاری [۳۷] احتمالی نظیر شبیه سازی تبرید ، یک سری قوانین احتمالی در حین جستجو مورد استفاده قرار می‌گیرد.

۶.۱ پیاده‌سازی الگوریتم‌های فراابتکاری

فرایند طراحی و پیاده‌سازی الگوریتم‌های فراابتکاری دارای سه مرحله‌ی متوالی است که هر کدام از آن‌ها دارای گام‌های مختلفی هستند. در هر گام فعالیت‌هایی باید انجام شود تا آن گام کامل شود. مرحله‌ی ۱ آماده‌سازی است که در آن باید شناخت دقیقی از مسأله‌ای که می‌خواهیم حل کنیم بدست آوریم، و اهداف طراحی الگوریتم فراابتکاری برای آن باید با توجه به روش‌های حل موجود برای این مسأله به طور واضح و شفاف مشخص شود. مرحله‌ی بعدی، ساخت نام دارد. مهمترین اهداف این مرحله انتخاب روند حل، تعریف معیارهای اندازه گیری عملکرد، و طراحی الگوریتم برای استراتژی حل انتخابی می‌باشد. آخرین مرحله پیاده‌سازی است که در آن پیاده‌سازی الگوریتم طراحی شده در مرحله‌ی قبل، شامل تنظیم پارامترها، تحلیل عملکرد، و در نهایت تدوین و تهیه گزارش نتایج باید انجام شود. در زیر انواع الگوریتم‌های روش بهینه سازی بیان شده است که روش‌هایجستجوی محلی و روش جستجوی حریصانه مد نظر ما می باشد.

۷.۱ انواع الگوریتم‌های بهینه سازی

بعضی از الگوریتم‌های بهینه سازی عبارتند از:
الگوریتم بهینه‌سازی دومرحله‌ای ازدحام ذرات، الگوریتم شبیه‌سازی تبریدی، الگوریتم تپه [۴۰] ، الگوریتم تپه‌نوردی [۴۱] ، الگوریتم تقسیم و حل، الگوریتم تکاملی [۴۲] ، الگوریتم جستجوی محلی (بهینه‌سازی)،

[۳۹] Pheromone trails
[۴۰] Hill Algorithm
[۴۱] Hill Climbed Algorithm

الگوریتم جستجوی ممنوعه، الگوریتم جهش ترکیبی قورباغه، الگوریتم چکه آب های هوشمند، الگوریتم حریصانه، الگوریتم ژنتیک [۳، ۱۱] ، الگوریتم غیر مرکب، الگوریتم کارمارکار [۴۳] [۳۷] ، الگوریتم کلونی مورچگان، الگوریتم لونبرگ–مارکارد [۴۴] ، الگوریتم ناسینوف، الگوریتم‌های فراابتکاری [۲۹، ۳۷]، برنامه‌ریزی پویا، برنامه‌سازی تکاملی، پوشش مجموعه [۴۵] [۵۳] ، روش بهینه‌سازی ازدحام ذرات، روش پنالتی [۴۶] [۴۱]، زمان‌بندی (رایانه)، شاخه و حد، ضرب زنجیره‌ای ماتریس، مسأله زیرآرایه بیشینه، هرس آلفا بتا و هوش ازدحامی که در این پایان نامه از دو الگوریتم حریصانه و الگوریتم جستجوی محلی بیشتر استفاده خواهد شد و در فصول بعدی به طور کامل به چگونگی عملکرد این دو الگوریتم می پردازیم.

[۴۲] Evolution Algorithm
[۴۳] Karmakar Algorithm
[۴۴] Lanberg - Markard Algorithm
[۴۵] Covering Set
[۴۶] Penalti Precedure

فصل ۲

روش فراابتکاری حریصانه

۱.۲ مقدمه

استراتژي‌هاي جستجوي آگاهانه يا مكاشفه‌اي [۱] ، از دانش مسأله استفاده می‌كند و در انتخاب جواب يا گره، جواب يا گرهی را انتخاب می‌كنند كه شانس رسيدن به هدف در آن بيشتر باشد يا به نظر آيد كه به هدف نزديک تر است. براي اينكه تخمين بزنيم كه جواب يا گره چقدر به هدف نزديک تر است از تابع ارزيابی استفاده می‌كنيم. اين تابع هزينه رسيدن به جواب هدف يا گره هدف را تخمين می‌زند و به عبارت ديگر ميزان مفيد بودن جواب فعلی را باز می‌گرداند. در جستجوي آگاهانه در رابطه با هزينه رسيدن به هدف، در اختيار عامل قرار داده می‌شود.

۲.۲ انواع جستجوی آگاهانه

الگوريتم‌های مختلفی در جستجوی آگاهانه يا مكاشفه ای با سازوكارهای خاص و گوناگونی وجود دارد كه از جمله اين الگوريتم‌ها می توان از الگوريتم نخست-بهترين [۲] ، جستجوی پرتو [۳] ، جستجو A^*، جستجوی IDA^* [۴]، جستجوی SMA^* [۵] و الگوريتم تپه نوردی نام برد.

۱.۲.۲ الگوريتم نخست-بهترين

اين استراتژي را در قالب يک گراف و يا يک درخت بيان می‌كنيم. در يک درخت، زماني كه گره‌ها مرتب مي‌شوند، گره‌اي كه بهترين ارزيابي را داشته باشد، قبل از ديگر گره‌ها بسط داده مي‌شود كه هدف

[۱] Revelation

[۲] Best – First Search

[۳] Beam Search

[۴] Intractive Deeping A

[۵] Simplified Memory Bounded A

یافتن راه‌حل‌هاي کم‌هزینه است، این الگوریتم‌ها عموماً از تعدادي معیار تخمین براي هزینه راه‌حل‌ها استفاده مي‌کنند و سعي بر حداقل کردن آن‌ها دارند. در این الگوریتم‌ها یک تابع ارزیابی تولید می‌شود که توضیحاتی در مورد مطلوب بودن یا نبودن بسط یک گره را ارائه می‌دهد. در این روش در هر مرتبه، گرهی که بهترین ارزیابی را داشته باشد ابتدا بسط داده می‌شود. به عبارت دیگر گرهی انتخاب می‌شود که تابع ارزیابی، بهترین مقدار را برای آن برگرداند. برحسب اینکه تابع ارزیابی چگونه پیاده سازی شود. روش‌های گوناگونی از الگوریتم اول بهترین خواهیم داشت. جستجوی اول بهترین دو نوع کلی دارد که در یکی تابع ارزیابی، هزینه رسیدن از گره فعلی به سمت هدف را حداقل می‌کند و در دومی هزینه کل مسیر از گره شروع تا هدف حداقل می‌شود.

۱.۱.۲.۲ جستجوی حریصانه [۱۲]

جستجوی حریصانه یکی از روش‌های جستجوی نخست- بهترین است. در این روش هدف به حداقل رساندن هزینه رسیدن به هدف با استفاده از تابع تخمین می‌باشد. تابع ارزیابی که هزینه رسیدن از یک حالت (حالت فعلی) به حالت هدف را تخمین می‌زند تابع ابتکاری [۶] نام دارد و با حرف H نمایش داده می شود. به عنوان مثال برای گراف ها داریم که $H(n)$ می تواند هزینه تخمین زده شده از ارزان‌ترین مسیر از گره n به گره هدف باشد یا تابع ارزیابی گره n می تواند تابعی باشد که فاصله گره n تا هدف را تخمین می زند. توابع ابتکاری، معمولی‌ترین شکل رساندن اطلاعات اضافی مسأله به الگوریتم جستجو هستند. ایده اصلی روش حریصانه این است که از یک مجموعه اولیه مانند مجموعه تهی شروع می کند و در هر گام یک مؤلفه از جواب را که نسبت به بقیه بهتر است انتخاب کرده و به جواب اضافه می کند. به بحث روش جستجوی حریصانه به طور مفصل در بخش (۳.۲) پرداخته شده است.

۲.۱.۲.۲ جستجوی پرتوی

دقیقاً همانند جستجوی حریصانه‌است با این تفاوت که در لیست اولیه آن حداکثر k عضو وجود دارد. یعنی فقط k تا از بهترین گره‌ها کاندید برای توسعه دادن هستند و با بقیه از فهرست خارج می‌شوند. فضای حالت در این روش نسبت به روش حریصانه کاهش می‌یابد اما ممکن است که جواب را از فهرست خارج کنیم این روش کامل و بهینه نیست.

۳.۱.۲.۲ جستجوی A^* [۱۳]

در جستجوی حریصانه با انتخاب تابع تخمین $H(n)$ که هزینه تخمینی رسیدن از گره فعلی به گره هدف [۷] بود سعی می‌کردیم که سریع تر به سمت هدف حرکت نماییم و همچنین فضای حالت را کاهش دهیم اما این روش نه کامل بود و نه بهینه. از طرفی در جستجو با هزینه یکسان با انتخاب تابع $g(n)$ که هزینه

[۶] Heuristic Function

[۷] Target

واقعی مسیر از ریشه [۸] تا گره n بود در پی یافتن مسیری [۹] با حداقل هزینه بود که روشی بود کامل [۱۰] و بهینه اما می‌توانست بسیار زمان بر بوده و در برخی موارد بی فایده باشد. برای دستیابی به مزایای هر دو جستجو از ترکیب این دو روش تحت عنوان جستجوی A^* استفاده می‌کنیم که تابع ارزیابی آن به صورت زیر است:

$$f(n)H(n) + g(n) \qquad (۱.۲)$$

$H(n)$: هزینه تخمینی برای رسیدن از گره n از ارزانترین راه به هدف

$g(n)$: هزینه رسیدن از گره ریشه به گره n

۳.۲ معرفی روش فراابتکاری جستجوی حریصانه

شیوه این روش از شخصیت معروف اسکروج [۱۱] گرفته شده است. شاید اسکروج حریص ترین شخصی باشد که تا به حال دیده شده است. او به جای آن که به گذشته و آینده فکر کند، تنها انگیزه هر روز او به دست آوردن طلای بیشتر بود. الگوریتم حریصانه نیز مانند شیوه اسکروج می‌باشد بدین صورت که عناصر داده‌ای را به طور متوالی گرفته و از بین آنها بدون توجه به انتخاب‌های قبلی یا بعدی، بهترین را بر اساس برخی معیارهای خاص انتخاب می‌کند. البته این عقیده که به علت برخی خصوصیات منفی اسکروج، مانند حرص، نباید از الگوریتم‌های حریص استفاده کرد، اشتباه است چرا که این الگوریتم ها، جواب‌هایی بسیار ساده و کارا تولید می‌کنند. روش حریصانه یکی از روش‌های مشهور و پرکاربرد طراحی الگوریتم‌ها است که با ساختاری ساده در حل بسیاری از مسائل استفاده می‌شود. این روش اغلب در حل مسائل بهینه‌سازی استفاده شده و در پاره‌ای مواقع جایگزین مناسبی برای روش‌هایی مانند برنامه‌ریزی پویا است. در حالت کلی این روش سرعت و مرتبه‌ی اجرایی بهتری نسبت به روش‌های مشابه خود دارد؛ اما متناسب با مسأله ممکن است به یک جواب بهینه‌ی سراسری ختم نشود. در روش حریصانه رسیدن به هدف در هر گام مستقل از گام قبلی و بعدی است. یعنی در هر مرحله برای رسیدن به هدف نهایی، مستقل از این که در مراحل قبلی چه انتخاب‌هایی صورت گرفته و انتخاب فعلی ممکن است چه انتخاب‌هایی در پی داشته باشد، انتخابی که در ظاهر بهترین انتخاب ممکن است صورت می‌پذیرد. به همین دلیل است که به این روش، روش حریصانه گفته می‌شود.

الگوریتم‌های حریصانه مانند برنامه نویسی پویا، اغلب برای مسائل بهینه سازی بکار می‌روند، با این تفاوت که استفاده از روش حریصانه بسیار راحت است. در برنامه نویسی پویا از خاصیت بازگشتی جهت تقسیم یک نمونه به نمونه های کوچکتر استفاده می‌شود در حالی که در الگوریتم حریصانه هیچ تقسیمی به نمونه‌های کوچکتر صورت نمی‌گیرد. یک الگوریتم حریصانه برای تولید جواب از دنباله‌ای عناصر انتخابی استفاده می‌کند که هر یک از آن ها در آن لحظه، بهترین انتخاب به نظر می رسند. یعنی هر انتخاب، به طور محلی بهینه است و انتظار می رود که بتوان یک جواب بهینه نهایی بدست آورد.

[۸] Tassel
[۹] Path
[۱۰] Perfect
[۱۱] Scrooge

البته جواب نهایی همیشه بهترین نیست. لذا برای یک الگوریتم معین بایستی تحقیق شود که آیا جواب نهایی همیشه بهینه است یا خیر؟ با یک مثال ساده، به معرفی الگوریتم می پردازیم. جو، کارمند بخش فروش یک فروشگاه است. اغلب با مشکل خرد کردن پول مشتریانش روبرو است. چراکه مشتریان معمولا نمی خواهند مقدار زیادی سکه دریافت کنند. مثلا اگر مشتری برای دریافت ۰/۸۷ دلار ۸۷ سکه یک پنی دریافت کند ممکن است عصبانی شود. بنابراین هدف او این است که نه تنها پول‌های باقیمانده مشتریان را به طور کامل پرداخت کند بلکه تعداد سکه‌های داده شده را به حداقل ممکن برساند. یک روش حریصانه برای حل چنین مسأله ای بصورت زیر است. جو باید مجموعه ای از سکه‌ها را تحویل مشتری دهد. در ابتدا سکه‌ای در این مجموعه وجود ندارد. جو به دنبال بزرگترین سکه می‌گردد با علم به این که معیار او برای انتخاب بهترین سکه (بهترین حالت محلی) ارزش سکه می‌باشد. این کار در یک الگوریتم حریصانه به روال انتخاب [۱۲] موسوم است. سپس بررسی می کند که اگر این سکه را به مجموعه پول خردها اضافه کند مقدار کل سکه‌ها از میزان بدهی او به مشتری تجاوز می کند یا خیر؟ این عمل را بررسی امکان سنجی [۱۳] در الگوریتم حریصانه گویند. اگر افزودن سکه موجب شود که مجموعه پول خردها از میزان بدهی او به مشتری بیشتر نشود آن سکه را اضافه می‌کند. سپس بررسی می‌کند که آیا ارزش کل سکه‌های مجموعه برابر میزان بدهی اوست یا خیر؟ به این عمل بررسی [۱۴] جواب در الگوریتم حریصانه گویند. اگر آنها برابر نبودند سکه دیگری را با استفاده از روال انتخاب می گیرد و روند فوق را تکرار می‌کند. این عمل تا آنجا تکرار می‌شود که ارزش سکه‌ها برابر میزان بدهی او به مشتری شود و یا این که سکه‌های صندوق تمام شده باشد که در حالت اخیر وی قادر نیست پول مشتری را به طور کامل پرداخت کند. زمانی که یک دزد عجول و حریص وارد خانه‌ای می‌شود، در مسیر حرکت خود هر وسیله و کالای با ارزشی را داخل کیسه می‌اندازد. وی در این حالت چندان توجهی نمی‌کند که چه اشیائی را قبلا برداشته و ممکن است در آینده چه اشیاء گرانبهاتری به دست آورد. او در هر گام تنها از بین اشیاء دم دست خود با ارزش‌ترین آن را انتخاب کرده و به وسایل قبلی اضافه می‌کند.

این روش کاربردهای عمومی دیگری نیز دارد. زمانی که در مقابل خرید از یک فروشگاه یک اسکناس تحویل فروشنده داده می‌شود، وی با یک حساب سرانگشتی سعی می‌کند با حداقل اسکناس‌ها و سکه‌های ممکن باقیمانده‌ی پول را تولید کرده و به خریدار تحویل دهد. این حساب سرانگشتی ممکن است روش حریصانه باشد.

۱.۳.۲ ساختار روش حریصانه

کلیت روش حریصانه در هر مرحله، انتخاب یک عنصر از عناصر موجود است. این عنصر قسمتی از جواب مسأله است که به مجموعه عناصر نهایی اضافه می‌شود. در طی این مسیر گام‌های زیر اتفاق می‌افتد.

۱. **روال انتخاب حریصانه:** در این گام یک عنصر برای اضافه شدن به مجموعه جواب انتخاب

[۱۲] Election

[۱۳] Feasibility Study

[۱۴] Survey

می‌شود. معیار یا روال انتخاب عنصر برای اضافه شدن، ارزش آن عنصر است. بسته به نوع مسأله، هر عنصر ارزشی دارد که با ارزشترین آنها انتخاب می‌شود.

۲. **امکان‌سنجی و افزودن:** پس از انتخاب یک عنصر به صورت حریصانه، باید بررسی شود که آیا امکان اضافه کردن آن به مجموعه جواب‌های قبلی وجود دارد یا نه. گاهی اضافه شدن عنصر یکی از شرایط اولیه‌ی مسأله را نقض می‌کند که باید به آن توجه نمود. اگر اضافه کردن این عنصر هیچ شرطی را نقض نکند، عنصر اضافه خواهد شد؛ وگرنه کنار گذاشته شده و بر اساس گام اول عنصر دیگری برای اضافه شدن انتخاب می‌شود. اگر گزینه‌ی دیگری برای انتخاب وجود نداشته باشد، اجرای الگوریتم به اتمام می‌رسد.

۳. **بررسی اتمام الگوریتم:** در هر مرحله پس از اتمام گام ۲ و اضافه شدن یک عنصر جدید به مجموعه جواب، باید بررسی کنیم که آیا به یک جواب مطلوب رسیده‌ایم یا نه؟ اگر نرسیده باشیم به گام اول رفته و چرخه را در مراحل بعدی ادامه می‌دهیم.

در زیر یک الگوریتم برای مسأله پول خرد این روال ارائه شده است:
در صورتی که تعدادی سکه و مثال حل نشده‌ای در اختیار داشته باشیم ؛
روال انتخاب : بزرگترین سکه باقیمانده را در اختیار می گیریم ؛
بررسی امکان سنجی : اگر مجموع مقدار سکه‌های دریافتی از مقدار بدهی مورد نظر فراتر باشد؛
سکه انتخاب شده را نمی‌پذیریم ؛
در غیر اینصورت
به مجموعه سکه های مجاز اضافه میشود ؛
بررسی جواب : اگر مجموع مقدار سکه ها برابر مقدار بدهی باشد ؛
جواب کامل و مسأله حل می‌شود ؛
به زبان ساده، در روش حریصانه طی هر مرحله یک عنصر به روش حریصانه به مجموعه جواب اضافه شده، شرط محدودیت‌ها بررسی شده و در صورت نبود مشکل، عنصر و عناصر بعدی به همین ترتیب به مجموعه جواب اضافه می‌شوند. در طی این گام‌ها اگر به یک شرط نهایی خاص برسیم، یا امکان انتخاب عنصر دیگری برای اضافه کردن به مجموعه جواب وجود نداشته باشد، الگوریتم پایان یافته و مجموعه جواب به دست آمده به عنوان جواب بهینه ارائه می‌شود. توجه داشته باشید که ممکن است بر اساس نوع مسأله، ترتیب اضافه شدن عناصر به مجموعه جواب اهمیت داشته باشد.

۲.۳.۲ ویژگی‌های جستجوی حریصانه

الگوریتم حریصانه یا آزمند شبیه روش‌های پویا اغلب برای حل مسائل بهینه سازی استفاده می‌شود. این الگوریتم بهترین انتخاب را با توجه به شرایط مسأله انجام می‌دهد به امید آنکه با ادامه‌ی همین روش بهینه سازی انجام شود . البته این نکته که بهترین انتخاب فعلی ما را به جواب بهینه می‌رساند را باید اثبات کرد . در شیوه حریصانه در هر مرحله عنصری که بر مبنای معیاری معین بهترین به نظر می‌رسد، بدون توجه به انتخاب هایی که قبلاً انجام شده یا در آینده انجام خواهد شد، انتخاب می‌شود. الگوریتم های

حریصانه اغلب جواب های ساده ای هستند. پیچیدگی زمانی در بدترین حالت برای جستجوی حریصانه $O(b^m)$ است که m حداکثر عمق فضای جستجو است. جستجوی حریصانه تمام جواب‌ها یا گره‌ها را در حافظه خود نگاه می‌دارد. بنابراین پیچیدگی فضای آن مشابه پیچیدگی زمانی آن است. میزان کاهش پیچیدگی به مسأله و کیفیت تابع h بستگی دارد و در صورتی که تابع مناسبی انتخاب شود هزینه فوق می تواند بسیار بهبود یابد. توجه کنید که در حالت معمول مسأله پیدا کردن تغییرات بااستفاده از برنامه‌ریزی پویا [۱۵] جواب بهینه را پیدا می‌کند. در بسیاری از موارد مسأله از ما می‌خواهد که یک متغیر را کمینه یا بیشینه کنیم و یا اینکه صورت مسأله، مستقیماً از ما نمی‌خواهد که چیزی را کمینه یا بیشینه کنیم اما می‌توان این مسأله را تبدیل به یک چنین مسأله‌ای کرد. در اکثر مسائل از این دست ما باید تعدادی انتخاب انجام دهیم و مسأله در تعدادی مرحله حل شود. بازه وسیعی از این مسائل توسط الگوریتم‌های حریصانه قابل حل می‌باشند. الگوریتم‌های حریصانه در هر مرحله در نظر می‌گیرند که کدام انتخاب در حال حاضر بهترین وضعیت را برای ما رقم می‌زند و آن را انجام می‌دهند بدون در نظر گرفتن این که انجام این حرکت در آینده ممکن است به ضرر ما تمام شود. بنابراین الگوریتم‌های حریصانه همیشه جواب درست را به ما نمی‌دهند اما خیلی وقت‌ها هم اینطور نیست این دسته از الگوریتم‌ها در علوم رایانه کاربرد وسیعی دارند.

شبه کد الگوریتم حریصانه برای حل مساله پول خرد به صورت زیر می‌باشد:

مسأله : پس دادن بقیه پول مشتری با استفاده از سکه‌های موجود.

ورودی : انواع سکه های موجود (۱ سنتی، ۵ سنتی، ۱۰ سنتی، ۲۰ سنتی، ۲۵ سنتی و نیم دلاری).

خروجی : پس دادن بقیه پول با حداقل تعداد سکه ها

گام ۱) مجموعه $\{۱, ۵, ۱۰, ۲۰, ۲۵, ۵۰\}$ را در C قرار بده؛

گام ۲) s را مقدار پول برگشتی یا پس داده شده به مشتری می نامیم؛

گام ۳) S را مجموعه پول‌هایی که برگشت می‌شود قرار بده $S = \Phi$ ؛

گام ۴) یک انتخاب از مجموعه C را در x بگذار؛

گام ۵) $C - \{x\}$ را محاسبه کن و مقدار آن را در C قرار بده؛

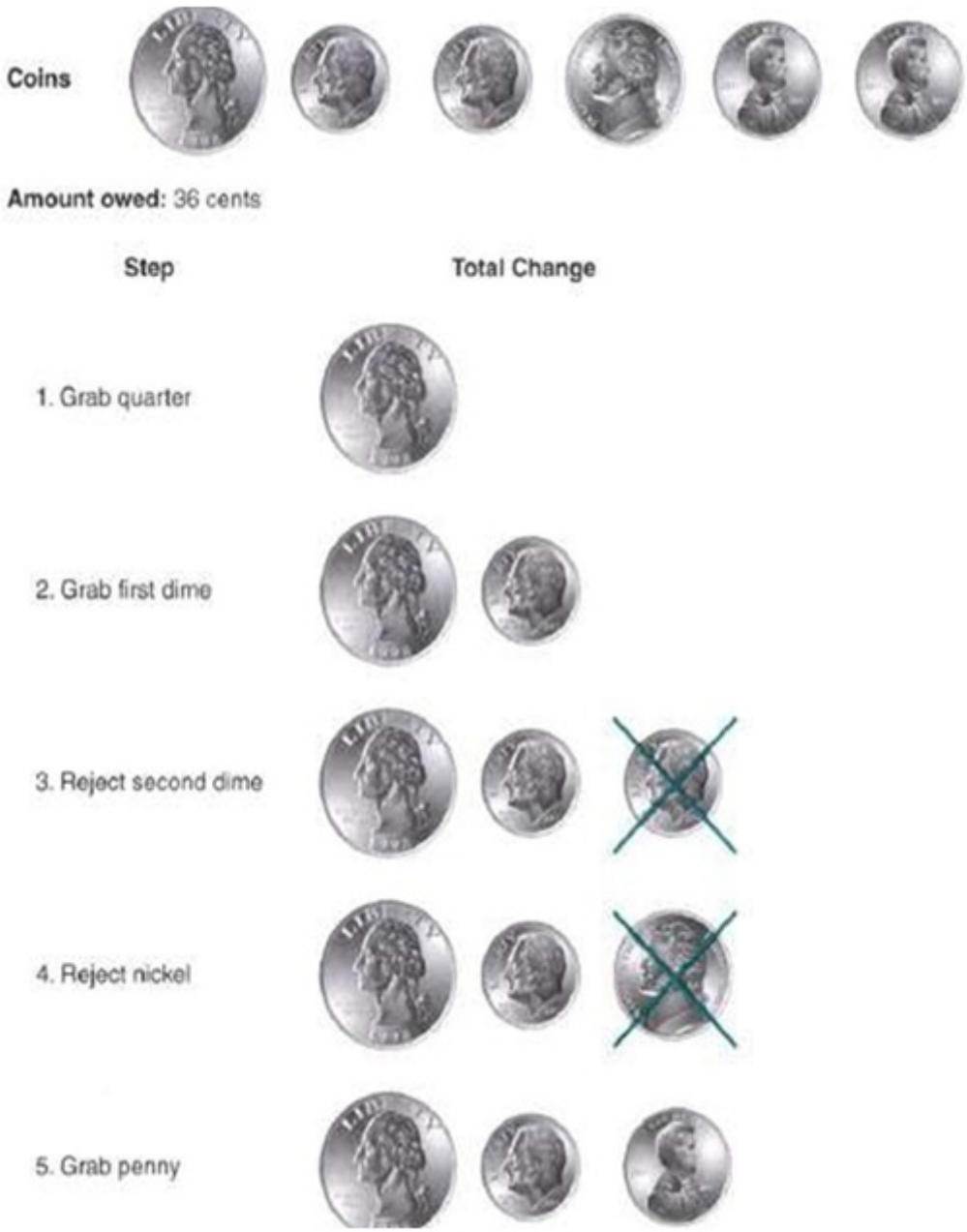
گام ۶) اگر (S, x) شدنی باشد، $S + X$ را محاسبه کن و مقدارش را در S قرار بده؛

گام ۷) اگر به مقدار s رسیدی، S را به عنوان جواب برگردان؛

گام ۱.۷) در غیر اینصورت Φ را برگردان .

همانطور که در شکل (۲.۲) مشاهده می‌شود پاسخ حریصانه شامل ۵ سکه می‌باشد. در صورتی که پاسخ بهینه شامل یک ۱۰ سنتی، یک ۵ سنتی و یک ۱ سنتی می‌باشد.

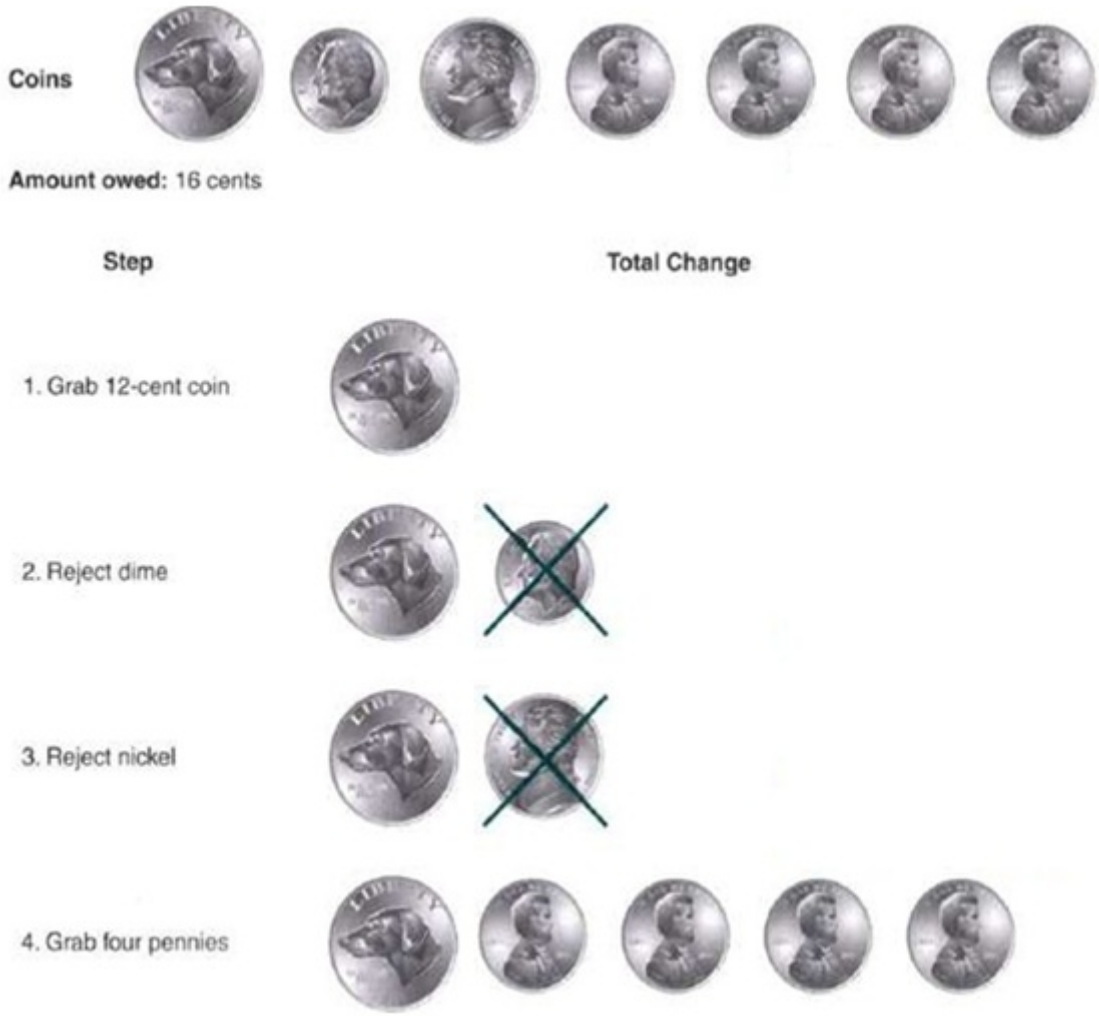
[۱۵] Linear programming Integer unknowns



شکل ۱.۲: بازگرداندن ۳۶ سنت پول باقیمانده به خریدار به روش حریصانه با توجه به سکه‌های موجود. سکه‌های موجود از چپ به راست : ۲۵ سنتی ، ۱۰ سنتی ، ۱۰ سنتی ، ۵ سنتی ، ۱ سنتی ، ۱ سنتی

مثال ۱.۳.۲. خریداری [۱۶] از یک فروشگاه یک جنس ۶۴ تومانی می خرد و ۱۰۰ تومان به فروشنده می دهد و فروشنده باید ۳۶ تومان به او برگرداند اگر فروشنده سکه‌های ۱, ۵, ۱۰, ۲۵، تومانی (از هرکدام حداقل یک نمونه) داشته باشد چگونه می تواند بقیه پول خریدار را برگرداند به نحوی که تعداد سکه‌ها (در کل) کمترین مقدار ممکن باشد؟

[۱۶] Purchaser



شکل ۲.۲: بازگرداندن ۱۶ سنت پول باقیمانده به خریدار به روش حریصانه با توجه به سکه‌های موجود. سکه‌های موجود از چپ به راست : ۱۲ سنتی ، ۱۰ سنتی ، ۵ سنتی ، ۴ سکه ۱ سنتی

یک جواب حریصانه به این صورت است:
در ابتدا هیچ سکه در مجموعه جواب نداریم. از بین سکه‌های موجود بزرگ ترین سکه ممکن یعنی ۲۵ تومانی را انتخاب می کنیم.این مرحله از الگوریتم حریصانه، روال انتخاب [۱۷] است. اگر یک سکه ۲۵ تومانی دیگر را انتخاب کنیم حاصل از ۳۶ تومان بیشتر شده، لذا آن را کنار گذاشته به سراغ سکه ۱۰ تومانی می‌رویم. حال بررسی می‌کنیم اگر این سکه ۱۰ تومانی را به مجموعه انتخابی قبلی اضافه کنیم حاصل از ۳۶ تومان بیشتر می‌شود یا خیر این مرحله، تحقیق عملی بودن [۱۸] می‌باشد. حال اگر این ۱۰ تومان رابه ۲۵ تومان جمع مجموعه انتخاب شده ۳۵ می‌شود که هنوز به ۳۶ نرسیده است. این مرحله از

[۱۷] Selection Procedure
[۱۸] Feasibility Check

الگوریتم، تحقیق حل شدن [۱۹] است. در ادامه، سکه‌های دیگر را به ترتیب مقایسه [۲۰] می‌کنیم و در نهایت با انتخاب سکه یک تومانی درکل با ۳ سکه (۲۵ تومانی و ۱۰ تومانی و یک تومانی) ۳۶ تومان به دست می‌آید و این حداقل تعداد سکه ممکن است. توجه کنید در انتخاب فوق ملاک انتخاب ، برای انتخاب بهترین سکه در هر مرحله (بهینه محلی) سکه است و در انتخاب سکه در هر مرحله به انتخاب‌های قبلی و بعدی کاری نداریم. در این شیوه اجازه فکر کردن درباره یک انتخاب انجام شده را نداریم یعنی هنگامی که سکهای پذیرفته شد به طور دائم جزء حل به حساب می‌آید و هنگامی که سکه‌ای رد می‌شود به طور دائم از حل کنار گذاشته می‌شود. همان طور که مشاهده کردید این روش بسیار ساده است، ولی اصلی ترین نکته این است که آیا این روش الزاماً به یک حل بهینه می‌رسد؟ در رابطه با مسأله خاص، می‌توان اثبات کرد که جواب بهینه است.

۴.۲ کاربردهای روش حریصانه

از جمله کاربردهای مشهور روش حریصانه مسائلی هستند که در ادامه معرفی می‌شوند.

۱.۴.۲ مساله کوله پشتی

مثالی از مسأله کوله پشتی [۲۱] مربوط به دزدی است که با کوله پشتی وارد یک جواهر فروشی می‌شود. اگر وزن قطعات از یک حد بیشینه W فراتر رود کوله پشتی پاره خواهد شد. هر قطعه دارای ارزش و وزن معینی خواهد بود. مسأله‌ای که دزد با آن مواجه است، تعیین حداکثر ارزش قطعات است در حالی که وزن کل آن‌ها از حد معین W فراتر نرود. این مسأله را کوله پشتی صفر ویک می‌گویند .می‌توان مسأله را بصورت زیر بیان کرد. فرض کنید n قطعه وجود داشته باشد، به طوری که وزن قطعه iام و ارزش قطعه iام میباشد و W حداکثر وزنی که کوله پشتی قادر به تحمل آن است. W_i, P_i, W همگی اعداد صحیحی هستند. با فرض این که

$$X_i = \begin{cases} ۱ & \text{اگر قطعه } i\text{ام برداشته شود} \\ ۰ & \text{در غیراینصورت} \end{cases}$$

مدل صفر و یک مساله کوله پشتی به صورت زیر است:

$$\begin{aligned} &minimize \quad \sum_{i=۱}^{n} P_i X_i \\ &subject\ to \\ &\sum_{i=۱}^{n} W_i X_i \\ &X_i \in \{۰, ۱\} \end{aligned} \tag{۲.۲}$$

[۱۹] Solution Check

[۲۰] Comparison

[۲۱] knapsack

جواب جستجوی جامع این است که همه‌ی زیر مجموعه‌های این n قطعه را در نظر بگیریم ، زیرمجموعههایی را که وزن کل آن‌ها از W فراتر نرود ،کنار می‌گذاریم واز میان آن‌هایی که باقی می‌ماند،آن را که بیشترین ارزش را دارد،انتخاب می‌کنیم. در این مسأله هدف پر کردن یک کوله‌پشتی از وسایل پر ارزشی است که وزن‌های مختلفی دارند. این کوله‌پشتی باید به نحوی پر شود که وزن بار آن از حد مجاز بیشتر نشده و ارزش وسایل داخل آن بیشینه باشد. در مسأله‌ی کوله پشتی کسری بر خلاف کوله‌پشتی صفر و یک این امکان وجود دارد که بتوان کسری از یک وسیله - مثل پارچه - را جدا کرده و به وسایل داخل کوله‌پشتی اضافه کرد.

۲.۴.۲ تولید درخت پوشای کمینه

فرض کنید که یک طراح شهری می‌خواهد چند شهر را با جاده‌هایی به هم متصل کند، بطوری که امکان حرکت از شهری به شهر دیگر وجود داشته باشد. اگر برای طراحی این جاده‌ها بودجه ای محدود پیش بینی شده باشد، او احتمالا در طراحی خود، حداقل طول جاده‌ها را در نظر می‌گیرد.
مسأله کوچکترین درخت پوشا روی گراف‌هایی با وزن یال‌های غیرمنفی و پیوسته مطرح می‌شود. گرافی را پیوسته می‌نامیم که هرگاه بین هردو گره دلخواه، یک مسیر مشخص وجود داشته باشد. در یک گراف، یک مسیر از یک گره به خودش را چرخه یا دور می‌نامند. یک درخت، گراف بدون جهت، پیوسته و بدون چرخه است. به عبارتی دیگر درخت، گراف بدون جهت است که در آن دقیقا یک مسیر بین هر زوج از گره‌ها وجود دارد. درخت پوشا یک زیر گراف پیوسته است که شامل همه گره‌ها بوده و یک درخت باشد. یک زیر گراف پیوسته با حداقل وزن باید یک درخت پوشا باشد اما هر درخت پوشایی دارای حداقل وزن نیست. چنین درختی کوچکترین درخت پوشا نامیده می‌شود که لازم به ذکر است که منحصربفرد نیست.

۱.۲.۴.۲ الگوریتم کراسکال

در نظریه گراف‌ها، الگوریتم کراسکال [۲۲] الگوریتمی برای یافتن یک زیرگراف فراگیر همبند با کمترین وزن در یک گراف وزن‌دار است (در یک گراف وزن دار، به هر یال وزنی نسبت داده شده‌است). همچنین این الگوریتم برای یافتن کوچکترین درخت فراگیر در یک گراف وزن دار استفاده می‌شود. به عنوان مثال فرض کنید یک شبکه راه آهن که تعدادی شهر را به یکدیگر متصل می‌کند در دست احداث است، می‌خواهیم با داشتن هزینه $C_{i,j}$ مربوط به احداث خط مستقیم بین شهرهای V_j و V_i شبکه را طوری طراحی کنیم که مجموع هزینه‌های ساخت به کمترین مقدار خود برسد. با در نظر گرفتن هر شهر به عنوان یک رأس از گراف وزن دار با وزن‌های $W(V_i, V_j) = C_{ij}$ مسأله به یافتن یک زیر گراف فراگیر همبند با کمترین وزن در یک گراف منجر می‌شود. حال الگوریتم کراسکال زیر را برای حل این مسأله ارائه می‌دهیم :
فرض کنید $G = (V, E)$ گرافی باشد که در آن $V = \{v_1, v_2, \ldots, v_n\}$ مجموعه رئوس و $E = \{e_1, e_2, \ldots, e_n\}$ مجموعه یال های آن می باشد. الگوریتم کراسکال بصورت زیر عمل می کند که :

[۲۲] Kruskal

۱. ابتدا یال e_1 را طوری انتخاب می کنیم که وزن آن کوچکترین مقدار باشد.

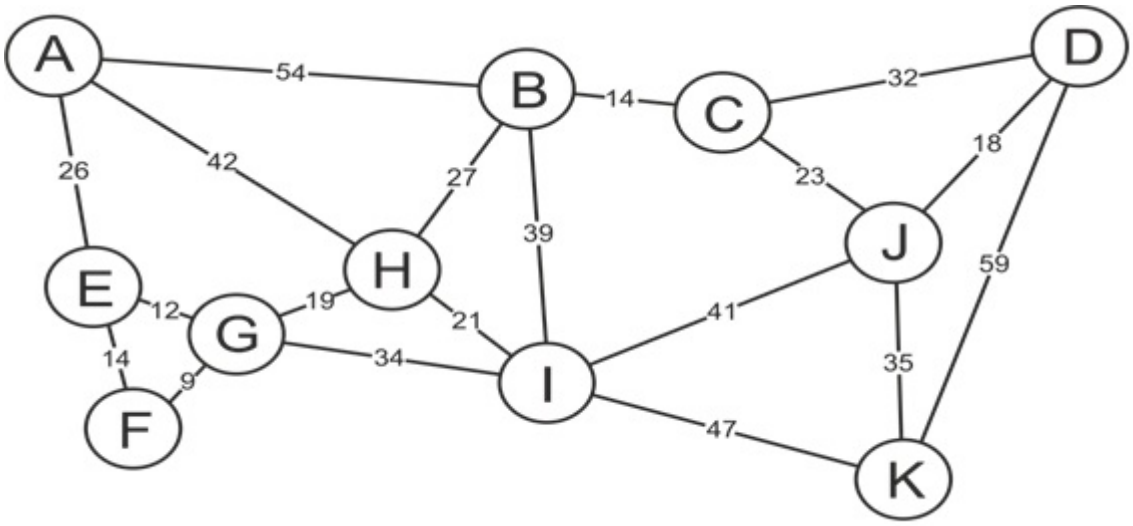
۲. اگر یال‌های $\{e_1, e_2, \ldots, e_{i+1}\}$ انتخاب شده باشند، آنگاه یال e_{i+1} را از میان $E-\{e_1, e_2, \ldots, e_i\}$ به گونه ای انتخاب می کنیم که :

الف) زیر گراف با یال های $\{e_1, e_2, \ldots, e_{i+1}\}$ بدون دور باشد.

ب) از میان یال‌های مشمول شرط (الف) وزن e_{i+1} دارای کمترین وزن ممکن باشد.

۳. در صورتی که مرحله ۲ قابل اجرا نباشد روند الگوریتم را متوقف خواهیم کرد.

درخت پوشای بهینه گراف شکل (۳.۲) که مربوط به احداث خط راه آهن بین چند شهر می باشد را با استفاده از الگوریتم کراسکال می خواهیم بیابیم.
راه حل، با استفاده از الگوریتم کراسکال:



شکل ۳.۲: گراف مربوط به احداث خط راه آهن شهری

۱. یال ها را به ترتیب وزن شماره گذاری میکنیم.

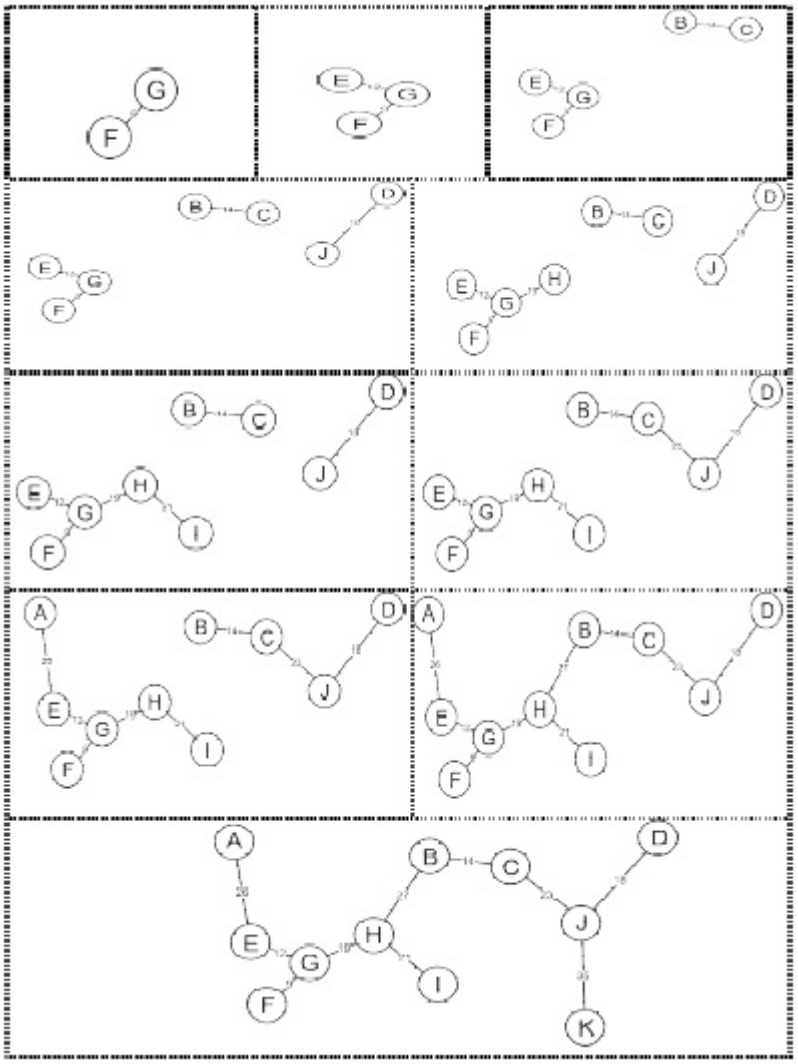
۲. از یال با حداقل هزینه شروع کرده و رئوس دو طرف آن را به وسیله یال بهم وصل می‌کنیم.

۳. کوچکترین یال بعدی را پیدا کرده و مرحله ۲ را تکرار می‌کنیم.

۴. مرحله ۳ را تا جایی ادامه می‌دهیم که تمام رئوس ملاقات گردند.

اولویت [۲۳] مراحل در شکل (۴.۲) از چپ به راست است.

[۲۳] Preference



شکل ۴.۲: گراف‌های مربوط به استفاده الگوریتم کراسکال برای تشکیل درخت پوشای کمینه

ملاحظه ۱.۴.۲. *در طی مراحل، نباید نیم دور یا دور تشکیل شود.*

روش‌های پریم [۲۴] و کراسکال دوروش مشهور تولید درخت پوشای کمینه از یک گراف وزن‌دار هستند که از روش حریصانه بهره می‌برند. منظور از درخت پوشای کمینه، درخت پوشایی از گراف است که مجموع وزن یال‌های آن کمتر یا مساوی مجموع وزن یال‌های سایر درخت‌های پوشای آن گراف است.

قضیه ۲.۴.۲. *[۳۷] هر درخت پوشای U با یال های $\{e_1, e_2, \ldots, e_{v-1}\}$ که توسط الگوریتم کراسکال ساخته می شود یک درخت بهینه است.*

برهان. از طریق تناقض خواهیم داشت که، به ازای هر درخت پوشای T از G به غیر از U کوچکترین مقدار i را به طوری که e_i در T نباشد با $f(t)$ نمایش می دهیم. اکنون فرض کنید که U یک درخت بهینه

[۲۴] Method Prim

نباشد و T را به عنوان درخت بهینه در نظر بگیرید که در آن $f(t)$ دارای بزرگترین مقدار ممکن باشد. فرض کنید $K = f(t)$ این بدان معنی است که $\{e_۱, e_۲, \ldots, e_{k-۱}\}$ هم در T و هم در U هستند. ولی e_k در T نیست پس شامل یک دور یکتای C می‌باشد. فرض کنید $e_{k۱}$ یالی از C باشد که در T هست ولی در U نیست. پس $e_{k۱}$ یال برشی از $e_k + t$ نمیتواند باشد. بنابراین ، $T_۱ = (T + e_k - e_{k۱})$ یک گراف همبند با $v - ۱$ یال بوده و در نتیجه درخت پوشای دیگری برای G خواهد بود. روشن است که :

$$w(T_۱) = w(T) + w(e_k) - w(e_{k۱})$$

ولی در الگوریتم کراسکال e_k به عنوان یالی با کمترین وزن طوری انتخاب شده است که زیر گراف G با یال‌های $\{e_۱, e_۲, \ldots, e_k\}$ بدون دور باشد. چون زیر گراف G با یال‌های $\{e_۱, e_۲, \ldots, e_{k۱}\}$ زیر گرافی [۲۵] از T است. بنابراین آن هم بدون دور است و نتیجه می گیریم که: $w(e_k) \leq w(e_{k۱})$ پس $w(T_۱) \leq w(T)$ پس $T_۱$ هم یک درخت بهینه خواد بوددر صورتی که داریم $f(T_۱) > K = f(T)$ که این با T انتخاب در تناقض است . بنابراین $T = U$ و در نتیجه U یک درخت بهینه است □

۳.۴.۲ محاسبه‌ی کوتاه ترین مسیر

زمانی که قصد داریم کوتاه ترین مسیر [۲۶] از یک مبدأ مشخص به تمامی رئوس دیگر گراف را محاسبه کنیم، الگوریتمی مانند دایجکسترا [۲۷] با استفاده از روش حریصانه به کمک ما می‌آید. شبکه G با M گره و N یال، و هزینه C_{ij} مربوط به هر یال (i, j) در G مفروض است. مسأله کوتاه ترین مسیر عبارت است از : پیدا کردن کوتاه ترین (کم هزینه ترین) مسیر از گره ۱ به M در G. هزینه مسیر، مجموع هزینه‌های یال‌های تشکیل دهنده آن مسیر است. اگر با برقراری شبکه ای بخواهیم یک واحد جریان را از گره ۱ به m، با هزینه مینیمال بفرستیم، مسأله کوتاه ترین مسیر را در قالب جریان شبکه ای می توان در نظر گرفت. از این رو، $\{b_۱ = ۱,\ b_i = ۰,\ b_m = -۱\}$ به ازای m یا $i \neq ۱$
مدل ریاضی مسأله کوتاه ترین مسیر بصورت زیر می باشد :

$$minimize \quad \sum_{i=۱}^{m} \sum_{j=۱}^{m} C_{ij} X_{ij},$$

$$sobject\ to$$

$$\sum_{j=۱}^{m} X_{ij} - \sum_{k=۱}^{m} X_{ki} = \begin{cases} ۱ & i = ۱ \text{ اگر} \\ ۰ & i \neq ۱ \text{ یا } m \text{ اگر} \\ -۱ & i = m \text{ اگر} \end{cases}$$

$$X_{ij} = ۰ \text{ یا } ۱, \quad i, j = ۱, ۲, \ldots, m \qquad (۳.۲)$$

که در آن مجموعه‌ها و محدودیت‌های ۱ − ۰ روی تمام یال‌های موجود در G گرفته شده‌اند. محدودیت‌های ۱ یا ۰ $= X_{ij}$ نشان دهنده آن است که هر یال در مسیر هست یا خیر. در صورت جایگزینی

[۲۵] Sub Graph
[۲۶] Shortest Path
[۲۷] Dijekstra

۱ یا ۰ $X_{ij} = ۰$ با $X_{ij} \geq ۰$ و موجود بودن یک جواب بهینه، آنگاه از روش سیمپلکس باز هم یک جواب شدنی پایه صحیح به دست می‌آید که در آن مقادیر هر متغیر صفر یا یک هستند. از این رو برنامه صحیح را به صورت برنامه خطی زیر حل می‌کنیم :

$$
\begin{aligned}
&minimize \quad \sum_{i=۱}^{m}\sum_{j=۱}^{m} C_{ij}X_{ij},\\
&sobject\ to\\
&\sum_{j=۱}^{m} X_{ij} - \sum_{k=۱}^{m} X_{ki} = \begin{cases} ۱ & i = ۱ \text{ اگر} \\ ۰ & i \neq ۱ \text{ یا } m \text{ اگر} \\ -۱ & i = m \text{ اگر} \end{cases}\\
&X_{ij} \geq ۰, \quad i,j = ۱,۲,\ldots,m
\end{aligned}
\tag{۴.۲}
$$

دوگان [۲۸] مسأله کوتاه ترین مسیر را در نظر بگیرید.

$$
\begin{aligned}
&maximize \quad w_۱ - w_۲\\
&sobject\ to\\
&w_۱ - w_۲ \leq C_{ij}, \quad i,j = ۱,۲,\ldots,m\\
&w_i \text{ نامقید}
\end{aligned}
\tag{۵.۲}
$$

راحت تر است که جایگزینی $w'_i = -w_i$ را انجام دهیم. مسأله را در حالتی که تمام $C_{ij} \geq ۰$ در نظربگیرید.یک روش کارا و قوی مشهور به الگوریتم دایجکسترا برای پیدا کردن کوتاه ترین مسیر از گره ۱ به گره m وجود دارد. با استفاده از این روش به طور خودکار کوتاه ترین مسیرها از گره ۱ به سایر گره‌ها نیز به دست می‌آید.

مرحله‌ی آغازین:

قرار دهید $w'_۱ = ۰$ فرض کنید $X = \{۱\}$

مرحله‌ی اصلی:

فرض کنید $X' = N - X$ و یال‌ها در مجموعه $(X, X') = \{(i,j) : i \in X, j \in X'\}$ در نظر بگیرید. فرض کنید $w'_p + c_{pq} = Min\{w'_i + c_{ij}\}; (i,j) \in (X, X')$ و $w'_q = w'_p + c_{pq}$ و گره q را در X قرار دهید. مرحله‌ی اصلی را دقیقا $m - ۱$ بار تکرار کنید و سپس توقف کنید، یک جواب بهینه برقرار است.

قضیه ۳.۴.۲. *الگوریتم حریصانه دایجکسترا یک یک جواب بهینه تولید می‌کند.*

برهان. فرض کنید، به طور استقرایی، که هر w'_i برای $i \in X$ هزینه کوتاه ترین مسیر از گره ۱ به i است. این مطلب یقینا برای $i = ۱$ درست است. الگوریتم را در لحظه ای بررسی می کنیم که گره جدید q در

[۲۸] Dual

شرف پیوستن به X است. فرض کنید :

$$w'_p + c_{pq} = Min\{w'_i + c_{ij}\}; (i,j) \in (X, X') \quad (۶.۲)$$

اکنون نشان می دهیم که کوتاه ترین مسیر از گره ۱ به گره q، $w'_q = w'_p + c_{pq}$ طول دارد و بطور تکراری میتوان آن را از کوتاه ترین مسیر موجود از گره ۱ به pبعلاوه یال (p,q) ساخت. فرض کنید p مسیری دلخواه از گره ۱ به گره q باشد. کافی است نشان دهیم که طول p بزرگتر یا مساوی w'_q است. از آنجا که گره ۱ در X و گره q فعلا در X است پس p باید شامل یک یال (i,j) شود که در آن $i \in X, j \in X'$ و j می توانند به ترتیب p و q باشند) . طول مسیر p بنابراین مساوی مجموع زیر است :

۱. طول از گره ۱ به i؛

۲. طول یال (i,j) یعنی C_{ij}؛

۳. طول از j به q.

با فرض استقرا، طول از گره ۱ به i بزرگتر یا مساوی w'_i است. به علت این که هزینههای تمام یال‌ها مطابق فرض نامنفی است، پس طول در قسمت ۳ بزرگتر یا مساوی صفر است. بنابراین طول p بزرگتر یا مساوی $w'_i + C_{ij}$ خواهد بود. با توجه به معادله ذکر شده (۶.۲) و از آن جا که $w'_q = w'_p + c_{pq}$ روشن است که طول p بزرگتر یا مساوی w'_q است. بنابراین اثبات کامل و اعتبار الگوریتم تأیید می شود □

۴.۴.۲ کدگذاری و فشرده سازی اطلاعات

کد هافمن [۲۹] یکی از روش‌های فشرده‌سازی اطلاعات است با کدگذاری مجدد کاراکترهای [۳۰] موجود در اطلاعات بر اساس میزان استفاده‌ی آنها، سعی در کم کردن حجم [۳۱] فایل می‌کند. بر اساس این روش، کاراکتری با استفاده‌ی بالا با کد کوتاه تر و کاراکتری با استفاده‌ی کم با کد طولانی‌تر جایگزین می‌شود.

۵.۲ روش‌های حریصانه برای حل مساله p-میانه

در این بحث به بیان روش‌های حریصانه برای حل مسأله p- میانه میپردازیم. فرض کنید که $N = (V(N), E(N))$ یک گراف بدون جهت با وزن رأس‌های مثبت [۳۲] و طول یال‌های مثبت باشد. حال باید p سرویس دهنده را روی یالها یا رأس‌های N مکان یابی کنیم، به گونه‌ای که مجموع فاصله وزنی رئوس تا نزدیکترین سرویس دهنده حداقل شود. مسأله p- میانه کاربردهای فراوانی دارد که از آن جمله می توان به مکان یابی مراکز شبکه [۳۳] ارتباطی کامپیوتری [۲۰] ، مراکز توزیع [۳۴] کالا، مراکز اداری،

[۲۹] Huffman
[۳۰] Character
[۳۱] Volume
[۳۲] Positive
[۳۳] Net
[۳۴] Distribution

مراکز نظامی [۳۵]، ایستگاه های اتوبوس و مراکز پستی اشاره کرد.
در مسأله p - میانه، میانه‌ها می‌توانند هر نقطه ای از شبکه انتخاب شوند، که در این حالت مسأله را مسأله p - میانه محض مینامند. اما اگر هدف پیدا کردن میانه، تنها روی رئوس باشد، مسأله را مسأله p- میانه رأسی نامند. در ادامه هر جا از اصطلاح مسأله p- میانه استفاده شد منظور مسأله p- میانه محض است. فرض کنید F یک مجموعه m عضوی از سرویس دهنده‌ها و n تقاضا یا مشتری باشد. هدف مسأله p- میانه یافتن زیر مجموعه‌ای با p سرویس دهنده است به طوری که مقدار هزینه و خدمت رسانی برای همه تقاضاها و مشتری‌ها حداقل شود. در مسأله p - میانه روی شبکه‌ها هدف ما پیدا کردن مجموعه‌ی $X + \{X_۱, X_۲, \ldots, X_p\}$، شامل مکان p سرویس دهنده روی شبکه $N = V(N), E(N)$ است، به طوری که اگر هر رأس v_i دارای وزن w_i باشند، مجموع فاصله‌های وزنی این مجموعه تا تمام رئوس روی شبکه N حداقل شود. به عبارتی در این حالت، هدف، کمینه کردن مجموع وزنی فاصله‌ها خواهد بود. یعنی:

$$Minf(x) = \sum_{v_i \in N} w_i d(X, v_i) \qquad (۷.۲)$$

فاصله هر رأس $v \in V$ از مجموعه X به صورت فاصله v تا نزدیکترین سرویس دهنده در X تعریف می‌شود. یعنی :
$d(X, v) = Min\ d(x_i, v); x_i \in X$ فاصله کوتاه ترین مسیر بین دو راس v_i و v_j روی $d(v_i, v_j$ شبکه می‌باشد.
یک مدل برنامه ریزی خطی صفر و یک برای مساله p-میانه به صورت زیر می‌باشد:

$$minimize\ \ z = \sum_{i=۱}^{n} \sum_{j=۱}^{n} w_i d_{ij} x_{ij}$$

$$subject\ to$$

$$\sum_{j=۱}^{n} = ۱, \qquad i = ۱, \ldots, n \qquad (آ۸.۲)$$

$$X_{ij} \leq y_j, \qquad i = ۱, \ldots, n, \ \ j = ۱, \ldots, n \qquad (ب۸.۲)$$

$$\sum_{j=۱}^{n} = p \qquad (ج۸.۲)$$

$$X_{ij} = ۰\ \ ۱, \quad i = ۱, \ldots, n, \ \ j = ۱, \ldots, n$$

$$y_j = ۰\ \ ۱ \qquad i = ۱, \ldots, n, \ \ j = ۱, \ldots, n$$

در فرمول فوق متغیر های X_{ij} و y_j به صورت زیر تعریف میشوند:

$$X_{ij} = \begin{cases} ۱ & \text{اگر راس } i \text{ به سرویس دهنده } j \text{ اختصاص داده شود} \\ ۰ & \text{در غیر اینطورت} \end{cases}$$

[۳۵] Military

$$y_j = \begin{cases} ۱ & \text{اگر راس } j \text{ به عنوان مکان سرویس دهنده اختصاص داده شود} \\ ۰ & \text{در غیر اینطورت} \end{cases}$$

در مدل ارائه داده شده محدودیت (۸.۲آ) بیانگر این مسأله است که هر رأس باید تنها به یک سرویس دهنده اختصاص یابد. محدودیت (۸.۲ب) نشان میدهد که رأس ها را تنها می توان به رئوسی اختصاص داد که بعنوان مکان یک سرویس دهنده در نظر گرفته شده باشد. یعنی به ازای آن رأس y_j باشد. محدودیت (۸.۲ج) نشان دهنده این مطلب است که p سرویس دهنده مورد نیاز است.
الگوریتم‌های بناکننده [۳۶] و سازنده حریصانه‌ای برای حل این مسأله بیان شده است که در زیر به بررسی آن‌ها می‌پردازیم.

۱.۵.۲ الگوریتم تصادفی

در الگوریتم تصادفی [۳۷] p سرویس دهنده بصورت تصادفی و یکنواخت انتخاب میشوند. این انتخاب خود دارای پیچیدگی زمانی $O(m)$ است و هزینه خدمت رسانی برای هر مشتری دارای پیچیدگی زمانی $O(pn)$ است.لذا پیچیدگی زمانی برای کل الگوریتم تصادفی $O(m+pn)$ میباشد.

۲.۵.۲ الگوریتم Rpg

در الگوریتم Rpg [۳۸] یک بخش α از p سرویس دهنده که α یک پارامتر ورودی است به صورت تصادفی انتخاب می‌شود. آنگاه جواب به روش حریصانه کامل می‌شود که بدترین حالت پیچیدگی زمانی الگوریتم $O((i-\alpha)(pmn)+\alpha(m+pn))$ است. مقدار بصورت تصادفی ویکنواخت [۳۹] از بازه [۰،۱] انتخاب می‌شود.

۳.۵.۲ الگوریتم $Rgreedy$

این نوع الگوریتم همانند الگوریتم حریصانه است اما در هر گام i بجای انتخاب بهترین از میان $m-i+۱$ انتخاب، بصورت تصادفی از $\alpha(m-i+۱)$ انتخاب خواهد شد که α از بازه [۰،۱) انتخاب و یک پارامتر ورودی است.
توجه کنید که اگر $\alpha \to ۰$ باشد آنگاه هم ارز روش حریصانه است و در صورتی که $\alpha \to ۱$ هم ارز روش تصادفی خواهد شد. این الگوریتم دارای پیچیدگی زمانی $O(pmn)$ است.

---

[۳۶] Constructive
[۳۷] Random Algorithm
[۳۸] Random Plus Greedy
[۳۹] Uniform

۴.۵.۲ الگوریتم $Pworst$

در این الگوریتم اولین سرویس دهنده بصورت تصادفی و یکنواخت انتخاب می‌شود. سپس سرویس دهنده‌های دیگر به آن اضافه خواهد شد. برای هر مشتری تفاضل بین این که چقدر هزینه تخصیص یافته است و چقدر بهترین تخصیص هزینه خواهد داشت را محاسبه می‌کند. سپس یک مشتری را به صورت تصادفی انتخاب می‌کند، با احتمال مناسب برای این مقدار، نزدیکترین سرویس دهنده را باز می‌کند. مشتریانی که برای آنها جواب کنونی بدتر است شانس بیشتری در انتخاب شدن را دارد. پیچیدگی زمانی الگوریتم $O(mn)$ است.

۵.۵.۲ الگوریتم $Sample$

این روش مانند روش حریصانه است اما بجای انتخاب بهترین همه انتخاب های ممکن تنها $q < m$ انتخاب آن هم بصورت تصادفی و یکنواخت مد نظر قرار میدهد. از بین آنها سودمندترین انتخاب خواهد شد. پیچیدگی زمانی الگوریتم $O(m + pqn)$ است. به جهت این که زمان اجرای الگوریتم به طور قابل توجهی کاهش یابد، مقدار q را به اندازه کافی کوچک و برابر $q = \log ۲(m/k)$ در نظر میگیریم.

۶.۲ مقایسه‌ی الگوریتم‌های سازنده روی مساله p-میانه

در ابتدا مشخص نبود که کدام الگوریتم کارایی لازم را دارا است لذا براي آنالیز بهتر این الگوریتم‌های سازنده، آزمایشی را روی مسأله p - میانه با اجرای کامل این الگوریتم ها و مقایسه با جواب های حاصل از روش $GRASP$ که در فصل چهارم شرح داده میشود ترتیب می‌دهیم که نتایج آن در جدول (۱.۲) نمایش داده می‌شود. برای توضیح این نتایج نیاز به تعاریفی داریم.

تعریف ۱.۶.۲. مقدار میانگین [۴۰] جواب را که با $Avg(.)$ نشان می‌دهیم از طریق روش $GRASP$ بدست می‌آید.

تعریف ۲.۶.۲. نشان دادن این که چقدر مقدار میانگین جواب بدست آمده بوسیله هر روش بالا یا پایین است را انحراف [۴۱] نسبی می‌نامیم که بر حسب درصد می‌باشد و با $DEV\%$ نمایش می‌دهیم.

جدول(۱.۲) دارای پنج ستون می‌باشد که در مورد آن ستون‌ها در زیر توضیحاتی داده می‌شود. ستون اول، به نام گذاری الگوریتم‌ها اختصاص داده شده است. میانگین انحرافات نسبی در هر روش را در جدول، در ستون دوم و ستون چهارم جدول مربوط به زمان صرف شده برای میانگین انحرافات نسبی می‌باشد. ستون سوم جدول بصورت زیر محاسبه می‌شود: الگوریتم‌ها بر اساس انحراف نسبی دسته بندی و مرتب می‌شوند به گونه‌ای که بهترین آنها یک امتیاز[۴۲]، دومین الگوریتم بر اساس انحراف نسبی

[۴۰] Average
[۴۱] Deviation
[۴۲] Point

جدول ۱.۲: نتایج بکارگیری الگوریتم‌های سازنده و بناکننده حریصانه روی مساله p-میانه

$METHOD$	$QUALITY$		$TIME$	
	$DEV\%$	$NARANK$	$DEV\%$	$NARANK$
$Pworst$	-۰.۰۰۶	-۰.۴۰۰	-۱۸.۰۷	-۰.۴۸۰
$Rgreedy$	۰.۰۲۰	-۰.۴۰۰	۳۵.۸	۰.۴۰۰
$Random$	۰.۰۱۵	۰.۰۰۰	-۲۴.۹	-۰.۸۴۰
Rpg	۰.۰۰۹	۰.۶۲۰	-۱۲.۳	-۰.۳۰۰
$Sample$	-۰.۰۲۹	-۰.۲۲۰	-۱۹.۵	-۰.۶۰۰

دو امتیاز و به ترتیب تا پنجمین الگوریتم پنج امتیاز را بخود تخصیص می‌دهند و در هنگامی که یک گره‌ای وجود داشته باشد امتیازات بصورت برابر میان الگوریتم‌های موجود تقسیم می‌شوند. اگر انحرافات نسبی برای این پنج الگوریتم بترتیب ۰/۰۳, ۰/۰۱, −۰/۰۲, −۰/۰۲, −۰/۰۳ باشد بنابراین الگوریتم ها بترتیب ۵, ۴, ۲/۵, ۲/۵, ۱ امتیاز خواهند گرفت. امتیاز گرفته شده توسط الگوریتم را رتبه نرمال [۴۳] الگوریتم نامند و با $NRANK$ نمایش می‌دهند. رتبه نرمال هر الگوریتم در بازه [۱،۱-] قرار دارد که اگر الگوریتمی بهتر از سایر الگوریتمها باشد رتبه نرمال آن به ۱- نزدیک تر و اگر از سایر الگوریتم‌ها بدتر عمل کند رتبه نرمال آن به ۱ نزدیکتر خواهد بود. ستون پنجم جدول به زمان صرف شده اجرای رتبه نرمال هر الگوریتم را نشان می‌دهد. در حالت کلی ستون سوم مربوط به کیفیت جواب و ستون پنجم مربوط به زمان اجرای الگوریتم می‌باشد. جدول نشان می‌دهد که الگوریتم‌ها بر حسب زمان سپری شده بهتر قابل تشخیص هستند نسبت به کیفیت جواب هایی که فراهم می‌کنند. تحلیل بدترین مورد پیچیدگی زمانی پیشنهاد می کند که $Rgreedy$ از سایر الگوریتم‌های دیگر کندتر عمل می‌کند. در واقع این الگوریتم نسبت به آنچه که جدول نشان می‌دهد کندتر عمل می کند. با توجه به ستون پنجم الگوریتم Rpg نسبت به سه الگوریتم دیگر کندتر عمل می‌کند و البته ما تمایل به این داریم که از روشی استفاده کنیم تا سریعتر به جواب با کیفیت بالا برسد. از میان این الگوریتم‌ها بنظر می‌رسد که $Random, Sample$ و $Pworst$ سریعتر و جواب‌هایی با کیفیت بهتری را یافت کنند. مشاهده می کنیم که روش $Sample$ با پیچیدگی زمانی کمتر به کیفیت جواب بالاتر دست میابد.

[۴۳] Normal Rank

فصل ۳

جستجوی محلی

۱.۳ مقدمه

در علم کامپیوتر، جستجوی محلی [۳۳] یک روش فرا ابتکاری [۳۷، ۴۹] برای حل مسائل بهینه سازی سخت، بصورت محاسباتی می‌باشد. جستجوی محلی می‌تواند در مسائلی مورد استفاده قرار گیرد که می‌تواند به عنوان یافتن جوابی برای حداکثر رساندن یک معیار در میان تعدادی از جواب‌های پیش رو، مطرح شود. الگوریتم‌های جستجوی محلی از یک جواب به جواب دیگر در فضایی از جواب‌های پیش رو (فضای جستجو) با استفاده از تغییرات محدود حرکت می‌کنند تا یک جواب به نظر مطلوب یافت شود یا یک زمانی سپری شود. الگوریتم‌های جستجوی محلی در تعداد زیادی از مسائل سخت محاسباتی، از جمله مسائلی از علم کامپیوتر (مخصوصا هوش مصنوعی)، ریاضیات [۱] ، تحقیق در عملیات [۲] [۱۳، ۱۸] ، مهندسی [۳] ، بیو انفورماتیک به طور گسترده به کار می‌رود. نمونه‌های از الگوریتم‌های جستجوی محلی، الگوریتم‌های $WalkSAT$ و الگوریتم ۲- opt برای مسأله فروشنده دوره گرد می‌باشد.

۲.۳ جستجوی محلی

جستجوي محلي پايه، اغلب بهبود تكراري ناميده ميشود، زيرا هر حركت در صورتي انجام مي‌شود كه جواب نتيجه، بهتر از جواب فعلي باشد. به محض يافتن كمينه محلي، الگوريتم پايان مي‌يابد.
تابع بهبود[۴] f [۲۷] ، در حقيقت مي‌تواند اولين بهبود يا بهترين بهبود يا هر گزينه بين اين دو باشد. اولين نوع تابع بهبود، همسايگي $N(s)$ را بررسي كرده و اولين جوابي را كه بهتر از s است انتخاب مي‌كند، دومين نوع تابع بهبود، كاملا همسايگي را كاوش كرده و جواب با كمترين مقدار تابع هدف را برمي‌گرداند.

[۱] Mathematics
[۲] Operation Research
[۳] Engineering
[۴] Improvement Function

هر دو روش در کمینه‌هاي محلي متوقف مي‌شوند. پس کارايي آنها به شدت وابسته به تعريف f, s و N ميباشد. کارايي رويه‌هاي بهبود تکراري در مسايل بهينه سازي معمولا اصلا رضايت بخش نيست. پس تکنيک‌هاي زيادي براي ممانعت از گرفتار شدن الگوريتم‌ها در کمينه‌هاي محلي توسعه يافتند، که اين کار با اضافه کردن مکانيزم‌هايي انجام مي‌شود که به آنها اجازه گريز از کمينه‌هاي محلي را مي‌دهند. اين مسأله همچنين باعث مي‌شود شرايط خاتمه الگوريتم‌های فراابتکاري، پيچيده‌تر از رسيدن ساده به کمينه محلي باشد. درواقع، شرايط خاتمه می‌تواند شامل: حداکثر زمان CPU ، حداکثر تعداد تکرار، پيدا شدن جواب s با $f(s)$ کمتر از مقدار آستانه تعريف شده يا حداکثر تعداد تکرار بدون بهبود باشد.

۳.۳ ساختار الگوريتم جستجوی محلی

بسياری از مسائل را می‌توان از لحاظ فضای جستجو و هدف به چندين روش مختلف بيان کرد. برای مثال، در مسأله فروشنده دوره گرد، يک جواب می‌تواند يک دور [۵](چرخه) و ميزان به حداکثر رساندن ترکيبی از تعداد گره‌ها و طول چرخه باشد. همچنين يک جواب ديگر می‌تواند يک مسير باشد و وجود يک چرخه، بخشی از هدف باشد. در اين روش نياز به يک جواب می‌باشد که اين جواب اوليه به عنوان مثال می‌تواند از روش حريصانه بدست آيد. الگوريتم جستجوی محلی از يک جواب پيش رو آغاز می‌شود و سپس به صورت مکرر به جواب مجاور [۶] حرکت می‌کند. اين تنها زمانی امکان پذير است که روابط همسايه‌ای و مجاورت در فضای جستجوی مسأله تعريف شده باشد. به عنوان مثال، در مسأله پوشش رأسی، مجاورت يک پوشش رأسی، ساير پوشش‌های رأسی مختلف توسط يک گره می‌باشد.

بهينه سازی محلی با مجاورهايش ، که مستلزم تغيير k مؤلفه از جواب‌ها می‌باشند اغلب به عنوان $k-opt$ شناخته ميشوند. بطور معمول، هر جواب پيش رو، بيشتر از يک جواب مجاور دارد. انتخاب هر کدام از جواب‌ها برای حرکت، تنها با استفاده از اطلاعاتی در باره جواب‌های مجاور با جواب فعلی صورت می‌گيرد و به همين جهت جستجوی محلی ناميده می‌شود. زمانی که جواب مجاور به گونه‌ای انتخاب شود که به صورت محلی معيار را به حداکثر برساند روش فرا ابتکاری فوق را الگوريتم تپه نوردی [۷] می‌نامند. وقتی هيچ تنظيماتی برای بهبود در مجاورت و همسايگی صورت نگيرد، جستجوی محلی، در يک نقطه بهينه محلی گير کرده‌است.

اين مشکل بهينه سازی محلی، را می‌توان با استفاده از راه اندازی مجدد (تکرار الگوريتم جستجوی محلی با شرايط اوليه متفاوت) يا طرح‌های پيچيده‌تری بر اساس تکرار، مانند جستجوی محلی تکرار شده در حافظه، جستجوی بهينه سازی فعال [۸] در حافظه با تغييرات تصادفی کمتر مانند شبيه سازی تبريدی [۹] ، برطرف کرد. پايان الگوريتم جستجوی محلی [۵۶] می‌تواند بر مبنای يک محدوده زمانی باشد، انتخاب معمول ديگر برای پايان دادن به الگوريتم، هنگامی است که بهترين جواب پيدا شده توسط الگوريتم، در

[۵] Round

[۶] Adjecent

[۷] hill climbing

[۸] Active

[۹] Annealing Simulated

گام‌های معینی بهبود نیافته باشد.
الگوریتم جستجوی محلی می‌تواند یک جواب معتبر را ارائه دهد، حتی اگر در هر زمانی قبل از اینکه پایان یابد دچار وقفه شود. به طور معمول الگوریتم‌های جستجوی محلی، الگوریتم‌هایی تقریبی یا ناکامل هستند، از این جهت که، حتی اگر بهترین جواب پیدا شده توسط الگوریتم بهینه نباشد جستجو ممکن است متوقف شود. این، حتی در صورتی که خاتمه الگوریتم، به علت عدم امکان بهبود جواب باشد، می‌تواند اتفاق بیفتد، بدین صورت جواب بهینه می‌تواند دور از مجاورت و همسایگی جواب‌هایی باشد که الگوریتم از آن‌ها عبور کرده‌است.
برای مسائل خاصی امکان دارد همسایه‌هایی را در نظر بگیریم که بسیار بزرگ هستند . اگر بهترین جواب در میان همسایگان، بتواند بطور کارآمد و مؤثر یافت شود، چنین الگوریتم‌هایی به عنوان الگوریتم‌های جستجوی مجاور با مقیاس بسیاربزرگ، نامگذاری می‌شوند.

۴.۳ زمینه های الگوریتم جستجوی محلی

جستجوی محلی، جزئی از زمینه‌های زیر است:

- الگوریتم‌های فرا ابتکاری
- بهینه سازی تصادفی [۱۰]
- بهینه سازی

زمینه‌های درون جستجوی محلی شامل موارد زیر است:

- الگوریتم تپه نوردی
- الگوریتم شبیه سازی تبریدی
- الگوریتم جستجوی ممنوعه [۱۱] [۴۸]

۵.۳ فضای واقعی جستجو

چند روش برای انجام جستجوی محلی از فضاهای واقعی جستجو [۱۲] وجود دارد:

- لوس - جاکولا [۱۳] : جستجو به صورت محلی با استفاده از یک توزیع یکنواخت.
- بهینه سازی تصادفی : جستجوی محلی با استفاده از یک توزیع نرمال.

[۱۰] Random Optimization
[۱۱] Tabu Search
[۱۲] Real Spaces Search
[۱۳] Luus–Jaakoa

- جستجوی تصادفی [۱۴] : جستجوی محلی توسط نمونه برداری کره چند بعدی پیرامون موقعیت جاری.
- جستجوی الگو [۱۵] : برداشتن گام در امتداد محوری از فضای جستجو با استفاده از کاهش نمایی اندازه گام

۶.۳ کاربرد الگوریتم‌های جستجوی محلی

برخی از مسائلی که در آنها الگوریتم جستجوی محلی [۳۳، ۵۶] به کار گرفته شده‌اند عبارتند از:

۱.۶.۳ مسأله پوشش رأسی

که در آن جواب پوشش راسی از یک گراف است و هدف یافتن جوابی با حداقل تعداد گره‌ها می‌باشد.

۲.۶.۳ مسأله فروشنده دوره گرد

در مساله فروشنده دوره‌گرد [۱۴، ۱۹] که جواب، چرخه‌ای شامل همه گره‌های گراف است و هدف به حداقل رساندن طول کل چرخه می‌باشد.

۳.۶.۳ مسأله رضایت بولی

یک جواب پیش رو، انتساب‌های درست است و هدف به حداکثر رساندن تعداد اجزای رضایت بخش از طریق انتساب‌های درست است ؛در این مورد، جواب نهایی این است که تنها زمانی که کل عبارت درست باشد مورد استفاده قرار می‌گیرد.

۴.۶.۳ مسأله زمان بندی پرستار

که در آن یک جواب، انتساب پرستاران به شیفت‌های کاری است به گونه‌ای که تمام محدودیت‌های ایجاد شده را ارضاء نماید.

۵.۶.۳ مسأله مکان یابی

مسأله مکان یابی [۱۶] سرویس دهنده‌ها [۱۷] و دیگر مسائل تسهیل موقعیت مرتبط، که الگوریتم جستجوی محلی، بهترین نسبت تقریبی شناخته شده از بدترین دیدگاه، را ارائه می‌دهد.

---

[۱۴] Random Search
[۱۵] Search Pattern
[۱۶] Location
[۱۷] Facility

۱.۵.۶.۳ مثالی از مساله مکان یابی سرویس دهنده‌ها

مسأله مکان یابی سرویس دهنده‌های بدون ظرفیت زیر را در نظر می‌گیریم. مدل این مسأله مکان یابی همان مدل معرفی شده در معادله (۷.۲) می‌باشد. فرض می‌کنیم، تعداد تقاضا و یا تعداد مشتری ها برای مسأله $m = ۶$ و تعداد سرویس دهنده‌های این مسأله $n = ۴$ باشد. ماتریس هزینه‌های سفر در جدول (۱.۳) نمایش داده شده‌اند. فرض می‌کنیم $N = \{۱, ۲, ۳, ۴\}$ مجموعه سرویس دهنده‌ها و

جدول ۱.۳: ماترس هزینه‌های سفر مساله مکان یابی سرویس دهنده‌ها

سرویس دهنده‌ها	۱	۲	۳	۴
مشتری ۱	۶	۲	۳	۴
مشتری ۲	۱	۹	۴	۱۱
مشتری ۳	۱۵	۲	۶	۳
مشتری ۴	۹	۱۱	۴	۸
مشتری ۵	۷	۲۳	۲	۹
مشتری ۶	۴	۳	۱	۵

$f = \{۲۱, ۱۶, ۱۱, ۲۴\}$ هزینه تأسیس سرویس دهنده‌ها باشد. مجموعه s را به عنوان یک جواب مسأله با این شرایط که $s \subset N$ و $s \neq \emptyset$ در نظر می‌گیریم. همسایگی برای s را به صورت زیر تعریف می‌کنیم

$$Q(s) = \{T \subset N | T = s \cup \{j\} \text{ for } j \notin s \text{ or } T = s - \{i\} \text{ for} i \in s\} \qquad (۱.۳)$$

یعنی مجموعه‌هایی که یا یک رأس به s اضافه شده است یا یک رأس از آن کم شده است. برای هر جواب s، هزینه متناظر با این جواب به صورت زیر محاسبه و نمایش داده می‌شود :

$$C(s) = \sum_{i=۱}^{۶} \sum_{j \in s} Min(C_{ij} + \sum_{j \in s} f_j \qquad (۲.۳)$$

معادله (۱.۳) نشان دهنده تخصیص مشتری i به سرویس دهنده j می‌باشد که نزدیکترین سرویس دهنده را به مشتری i می‌رساند. به عنوان مثال فرض کنید که از جواب $s(۰) = \{۱, ۲\}$ آغاز کنیم هزینه متناظر با توجه به معادله (۲.۳) به صورت زیر می‌باشد :

$$c(s(۰)) = (۲ + ۱ + ۲ + ۹ + ۷ + ۳) + ۲۱ + ۱۶ = ۶۱$$

همسایگی برای $s(۰)$ با توجه به معادله (۱.۳) به صورت زیر می‌باشد که:

$$Q(s(۰)) = \{۱\}, \{۲\}, \{۱, ۲, ۳\}, \{۱, ۲, ۳, ۴\}$$

اگر برای هر همسایگی هزینه مربوط به آن را محاسبه کنیم خواهیم داشت :

$$c(\{۱\}) = ۶۳, \quad c(\{۲\}) = ۶۶, \quad c(\{۱, ۲, ۳\}) = ۶۰, \quad c(\{۱, ۲, ۳, ۴\}) = ۸۴ \qquad (۳.۳)$$

بنابراین $s(۱) = \{۱, ۲, ۳\}$ با $c(s(۱)) = ۶۰$ (کمترین هزینه) انتخاب می‌شود. با ادامه این روند و تکرار خواهیم داشت : $Q(s(۱)) = \{۱, ۲\}, \{۱, ۳\}, \{۲, ۳\}, \{۱, ۲, ۳, ۴\}$، بهترین همسایه $s(۱)$ مجموعه $s(۲) = \{۲, ۳\}$ با مقدار $c(s(۲)) = ۴۲$ می‌باشد که همسایگی آن به صورت زیر است :

$Q(s(۲)) = \{۲\}, \{۳\}, \{۱, ۲, ۳\}, \{۲, ۳, ۴\}$ ، بهترین همسایه $s(۲)$ ، مجموعه $s(۳) = \{۳\}$ با مقدار هزینه $c(s(۳)) = ۳۱$ می‌باشد. هیچ کدام از همسایگی‌های $s(۳)$ از $s(۳)$ بهتر نیستند پس توقف می‌کنیم چرا که $s(۳) = \{۳\}$ یک جواب بهینه موضعی نسبت به همسایگی تعریف شده می‌باشد .

فصل ۴

روش جستجوي تطابقی تصادفی حریصانه

۱.۴ مقدمه

روش گرسپ [۱][۷، ۸، ۹، ۱۵، ۳۰]، جستجوی تصادفی تطابقی حریصانه است که در حوضه حل مسائل بهینه سازی همچون علم مکان یابی و.... قرار دارد و دارای کاربردهای فراوانی در این علم می‌باشد. به عنوان مثال می‌توانیم حالت خاصی از مسائل بهینه سازی را مانند مسأله p- میانه و یا p- مرکز در نظر بگیریم که با استفاده از روش $GRASP$ براحتی قابل حل می باشد. از مزیت‌های این روش نسبت به سایر روش‌های حریصانه، دیگر پیچیدگی زمانی کمتر و کیفیت بالای جواب برای مسأله را می‌توان ذکر کرد. در تمامی روش‌های ارائه شده برای حل و یافتن جواب بهینه در مسائل بهینه سازی تلاش شده است که زمان صرف شده کاهش و کیفیت و مطلوبیت جواب یا جواب‌ها افزایش یابد که در این خصوص روش $RASPG$ روشی با الگوریتمی بسیار قوی است که از لحاظ زمان اجرا و کیفیت جواب بدست آمده نسبت به روش های دیگر بهتر و مفیدتر عمل می کند.

۲.۴ تاریخچه روش $GRASP$

روش $GRASP$ در اواخر سال ۱۹۸۷ میلادی نخستین بار توسط شوگان [۲] و هارت [۳] [۵۵] در انگلستان پیشنهاد و معرفی گردید و سپس در همان سال ۱۹۸۷ میلادی پژوهشگرانی چون فئو [۴] و رزنده [۵] از کشور برزیل این روش را برای مسأله برنامه ریزش پوششی [۶] مورد مطالعه و استفاده کردند. بعدها افراد

---

[۱] (GRASP)

[۲] A. W. Shogan

[۳] J. P. Hart

[۴] T. A. Feo

[۵] M. G. C. Resende

[۶] Covering Problem

دیگری چون ریبیرو [۷] [۷] ، فرناندز [۸] [۴۳] ، گلاوور [۹] [۴۹] و... روی روش $GRASP$ متمرکز شدند و به نتایج شگرفی در به کار گیری این روش برای حل مسائل بهینه سازی دست یافتند. کاربرد روش $GRASP$ برای مسائل بهینه سازی جهت یافتن جواب یا جواب‌هایی با کیفیت بالا همراه با استفاده از مسیر پیوستگی [۱۰] در طول بهینه سازی می‌باشد. روش مسیر پیوستگی [۶۲] با عملکردی مناسب از طریق مسیر یابی کمک شایانی را در بالا بردن کیفیت و بهبود جواب‌های یافت شده در روش $GRASP$ می‌کند. روش مسیر پیوستگی ابتدا توسط گلاور طرح و معرفی گردید، که بعد ها از این روش در روش جستجو هایی چون جستجوی تابو، جستجوی محلی و جستجوی پراکنده [۱۱] [۵۱] بکار گیری شد.

۳.۴ ساختار روش $GRASP$

روش $GRASP$ را روش مرحله ای [۶] نیز می‌نامند چرا که در الگوریتم این روش، از دو فاز اساسی و پایه استفاده می شود که فاز اول آن را فاز ساخت [۱۲] و فاز دوم آن را فاز جستجوی محلی می‌نامند. در الگوریتم های حریصانه، از یک جواب تهی شروع کرده و با تخصیص مقادیر به متغیر های تصمیم در هر مرحله، سعی در به دست آوردن یک جواب کامل می شود. تخصیص مقدار برای هر متغیر در الگوریتم های حریصانه با تعریف یک مجموعه عناصر برای ایجاد جواب آغاز می‌شود. یک جواب از مسئله با انتخاب عناصر این مجموعه برای متغیرها به دست می‌آید. در هر گام از الگوریتم‌های حریصانه، یکی از عناصر با کمترین هزینه از بین مجموعه عناصر تخصیص داده نشده برای اضافه شدن به جواب جزئی، انتخاب می‌شود . جستجوی حریصانه یا نزدیک بینانه، در هر مرحله بهترین انتخاب از عناصر جواب را بدون در نظر گرفتن این که این انتخاب در آینده چه تبعاتی دارد انجام می‌دهند. به عبارتی، در جستجوی حریصانه همیشه حرکتی که به جواب بهینه محلی که به کمترین مقدار افزایش در تابع ارزیابی جواب منجر می شود، انجام می‌گردد . تنها در مسائل خاصی می‌توان الگوریتم حریصانه ارائه کرد که همواره جواب بهینه تولید کند. روش جستجوی تصادفی تطابقی حریصانه، یک ساختار کلی برای تولید جواب ارائه می‌کند و به دلیل ساختار منعطف آن، با سایر روش‌های فرا ابتکاری قابل ترکیب است. هر گام از الگوریتم جستجوی تصادفی تطابقی حریصانه، شامل گام‌های ایجاد جواب و جستجوی محلی است. در صورتی که جواب به دست آمده در فاز ساخت یک جواب موجه نباشد، با تغییراتی تبدیل به یک جواب موجه می‌شود. سپس این جواب به دست آمده وارد فاز جستجوی محلی شده تا جواب بهینه محلی به دست آید. در صورتی که جواب به دست آمده از فاز جستجوی محلی، از بهترین جواب فعلی بهتر باشد، این جواب جایگزین بهترین جواب فعلی می‌شود. این مراحل تا برآورده شدن معیار توقف الگوریتم، ادامه می‌یابد. استراتژی الگوریتم جستجوی تصادفی تطابقی حریصانه، برای فرار از دام

---

[۷] C. G. Ribeiro
[۸] E. Fernandes
[۹] F. Glover
[۱۰] Path Relinking
[۱۱] Scatter Search
[۱۲] Construction Phase

بهینه محلی، شروع کردن از جواب‌های اولیه مختلف در فاز جستجوی محلی است. از آن جا که روش $GRASP$ یک روش تکراری می‌باشد لذا در هر تکرار آن از هر دو فاز اساسی استفاده و به کارگیری می‌شود. $GRASP$ ، فراابتکاری ساده است که ابتکارات ساختاري و جستجوي محلي را ترکیب مي کند. در زمان اتمام رویهء جستجو، بهترین جواب یافته شده برگردانده مي‌شود. مکانیزم ساخت جواب با دو جزء اصلي مشخص مي گردد. تابع ابتکاري سازنده [۱۳] پویا و تصادفي کردن. فرض کنیم جواب s شامل زیرمجموعه اي از مجموعه عناصر(اجزاي جواب) است، جواب با اضافه کردن مرحله به مرحلهء یک عنصر جدید در هر زمان، ساخته مي‌شود. انتخاب عنصر بعدي با برداشتن تصادفي عنصر به صورت یکنواخت از لیست کاندیداها انجام مي‌شود. عناصر براساس مقیاس ابتکاري، رتبه بندي شده‌اند که به آن‌ها امتیازي مي‌دهد که تابعي از مزیت درج این عنصر در جواب جزئي فعلي است. مقادیر ابتکاري در هر مرحله بروز رساني مي‌شوند پس امتیاز عناصر برحسب انتخاب‌هاي ممکن، در طي فاز ساخت، تغییر مي‌کند. این تابع ابتکاري ساختاري، پویاست برخلاف انواع ایستا که فقط یکبار در زمان شروع ساخت، به عناصر امتیازات را نسبت مي‌دهد. به عنوان مثالي از تابع ابتکاري ایستا، در مسأله فروشنده دوره گرد [۱۴، ۱۹] ، هر یالي که هزینه کمتري دارد امتیاز بیشتري دارد. در مورد حالت پویا، تابع ابتکاري درج ارزان ترین عنصر، مي‌تواند به این صورت باشد که امتیاز هر عنصر بر اساس جواب جزئي فعلي ارزیابي گردد. طول α تاثیر زیادي بر قدرت تابع ابتکاري دارد. اندازه آن مي‌تواند ثابت باشد یا با هر تکرار تغییر یابد. فاز دوم الگوریتم، فرایند جستجوي محلي است که مي‌تواند یک جستجوي محلي پایه یا تکنیک پیشرفته‌اي مثل الگوریتم جستجو یا جستجو ممنوع باشد. $GRASP$ در صورت برقراري دو شرط زیر، مي‌تواند موثر واقع شود:

۱. مکانیزم [۱۴] ساخت جواب از امیدوارکننده ترین مناطق فضاي جستجو نمونه برداري کند.

۲. جواب ساخته شده با تابع ابتکاري سازنده، به جواب‌هاي کمینه محلي متفاوت تعلق داشته باشند.

شرط اول از طریق انتخاب توابع ابتکاري ساختاري موثر و طول لیست کاندید مناسب برقرار مي‌گردد ولي دومي با انتخاب تابع ابتکاري ساختاري و جستجوي محلي به شکلي که با هم مناسبت داشته باشند، برقرار مي‌شود. این الگوریتم به دلیل سادگي، عمدتا بسیار سریع است و در زمان کوتاه جواب هاي نسبتا خوبي را ارائه مي‌کند. به علاوه مي تواند با موفقیت در سایر تکنیک هاي جستجو، ادغام گردد. این روش بر خلاف روش هایی نظیر الگوریتم ژنتیک و الگوریتم اجتماع مورچگان، از طبیعت الگو نگرفته است. در این روش برای فرار از به دام افتادن جواب بهینه محلی، از نقاط شروع مختلفی برای گام جستجوی محلی استفاده می‌شود. در این روش، پارامتر های کمی (پارامتر تعیین لیست محدود شده، معیار توقف) نیاز به تعیین شدن دارند. همچنین این الگوریتم می تواند به راحتی به صورت موازی روی چندین پردازنده اجرا شود. در عین حال در حالت پایه این روش، از ساختار حافظه در طول جستجو استفاده نمی‌شود.

[۱۳] Constructive Heuristic

[۱۴] Mechanism

۴.۴ کاربرد های روش $GRASP$

روش $GRASP$ کاربردهای فراوانی در مسائل بهینه سازی و مکان یابی دارد. نخستین کاربرد روش $GRASP$ را فئو و رزنده [۴، ۱۴] برای مسأله برنامه ریزی پوششی به کار گرفتند. پژوهشگران برای مسائل مختلفی از این روش جهت دست یابی به جواب یا جواب هایی با کیفیت بالا استفاده کردند. مسائلی چون مسیر یابی [۱۵] [۲۴، ۵۹] ، منطق [۱۶] ، پوشش [۵۳] ، طبقه بندی [۱۷] ، مکان یابی، بهینه در گراف ها [۱۸] [۲] ، مجموع برش ها [۱۹] ،مسأله $p-$ میانه [۲۰] ، ارتباطات مخابراتی و ....از جمله مسائلی بودند که از روش $GRASP$ استفاده نمودند. سابقه این الگوریتم به روش جستجوی محلی تصادفی تکرار شونده و الگوریتم های ابتکاری نیمه حریصانه بر میگردد. به طور کلی کاربردهای روش جستجوی تصادفی تطابقی حریصانه در ادبیات موضوع در حوزه های زمان بندی، مسیر یابی، مکان یابی، نظریه گراف و مسئله تخصیص دیده میشود. روش جستجوی تصادف تطابقی حریصانه در مسائل زمان بندی متفاوتی مانند زمان بندی پروازها، زمان بندی ماشین‌های موازی، زمان بندی مسائل تک ماشینه با زمان آماده سازی وابسته به توالی، زمان بندی کار کارگاهی و زمان بندی جریان کارگاهی با هزینه های آماده سازی به کار گرفته شده است. پایه مهم الگوریتم جستجوی تصادفی تطابقی حریصانه، الگوریتم های حریصانه می باشد. روش $GRASP$ یک روش فراابتکاری و چند مرحله ای برای حل مسائل ترکیبی بهینه سازی به فرم ساده مسأله (۱.۴) ذکر شده در زیر بکار میرود.

$$\begin{aligned} &Minimize \quad f(x) \\ &Subject\ to \\ &x \in X \end{aligned} \tag{۱.۴}$$

که در مدل مسأله (۱.۴) تابع $f(x)$ یک تابع هدف [۲۱] برای مسأله و مجموعه X جواب‌های شدنی [۲۲] مسأله می‌باشد.

شبه کد [۲۳] روش فراابتکاری $GRASP$ [۲۴] به صورت زیر است :

۱) ورودی را بخوان؛

۲) برای حداکثر تکرارها ، $K = ۱, ۲, \ldots$ مراحل زیر را انجام بده ؛

---

[۱۵] Routing
[۱۶] Logic
[۱۷] Partition
[۱۸] Optimization In Graph
[۱۹] Sum Cut
[۲۰] P-Median Problem
[۲۱] Objective Function
[۲۲] Feasible Solution
[۲۳] Pseudo – Code
[۲۴] Metaheuristic

۱-۲) جواب فاز ساخت را در $Solution$ قرار بده؛

۲-۲) جواب بهبود شونده (جستجوی محلی) را در $Solution$ قرار بده؛

۳-۲) جواب را بروز رسانی کن؛

۳) بهترین جواب را برگردان؛

حالت خاص برای مسائل حداقل سازی می‌توان شبه کد زیر را بیان کرد :
ورودی : تعداد تکرار ها $imax$
خروجی : جواب بهینه $x^* \in X$

گام ۱) ∞ را در f^* قرار بده؛

گام ۲) برای $i = ۱, \ldots, imax$ مراحل زیر را انجام بده ؛

گام ۱-۲) فاز ساخت تصادفی حریصانه را روی x انجام بده و مقدارش را در x بگذار؛

گام ۲-۲) جستجوی محلی را روی x انجام و مقدارش را در x قرار بده؛

گام ۳-۲) اگر $f(x) < f^*$ باشد آنگاه $f(x)$ را در f^* قرار بده؛

گام۳) x را در x^* قرار بده؛

معیار توقف در شبه کد روش فراابتکاری بالا به وسیله تعداد تکرار ها یعنی $MaxInteration$ معین می‌شود. هنگامی که زمان محاسبه شده از یک تکرار به تکرار دیگر خیلی زیاد نباشد زمان محاسبه مجموع پیش بینی می‌شود که با افزایش تکرارها زمان صرف شده به صورت خطی افزایش می‌یابد. در نتیجه، تعداد تکرارهای بیشتر زمان محاسبه افزایش و جواب بهتری یافت خواهد شد.

۵.۴ فاز ساخت روش فرا ابتکاری $GRASP$

فاز ساخت توسط اعضای داوطلب که توانایی ترکیب و پیوند با جواب‌های جزئی زیر ساخت را دارند شکل می‌گیرد. انتخاب عضو بعدی برای ترکیب به وسیله تابع ارزیابی که تابع ارزیاب حریصانه نامیده می‌شود،تعیین می‌گردد. تابع ارزیاب حریصانه معمولا مقدار سود در مقدار تابع هدف را نشان می‌دهد. ارزیابی اعضاء به وسیله این تابع منجر به ایجاد مجموعه داوطلب محدودی می‌شود که با RCL [۲۵] نمایش می‌دهیم که این مجموعه به وسیله بهترین اعضاء شکل گرفته است، یعنی آن دسته از جواب‌هایی که در جریان ترکیب با جواب جزئی کوچکترین مقادیر سود را دارا هستند این خاصیت نمونه ای از حریصانه بودن الگوریتم می‌باشد. یک عضو از RCL به صورت تصادفی برای ترکیب انتخاب می شود. جواب‌های تولید شده به وسیله ساخت تصادفی حریصانه لزومی ندارد که بهینه باشد. شبه کد فاز ساخت به صورت زیر است :

[۲۵] Restricted Candidate List

گام ۱) $\emptyset$ را در $Solution$ قرار بده؛

گام ۲) مقادیر سود و رشد اعضای داوطلب را ارزیابی کن؛

گام ۳) تا زمانی که $Solution$ کامل نیست مراحل زیر را انجام بده؛

گام ۳-۱) لیست داوطلب محدود (RCL) را بساز؛

گام ۳-۲) s را از RCL به صورت تصادفی انتخاب کن؛

گام ۳-۳) $Solutions$ را در $Solution$ قرار بده ؛

گام ۳-۴) مجددا مقادیر سود و رشد را ارزیابی کن؛

گام ۴) $Solution$ را برگردان؛

۶.۴ ساخت لیست اعضای داوطلب محدود RCL

یک خصوصیت جذاب روش فرا ابتکاری $GRASP$ اجرای [۲۶] آسان این روش است، چرا که پارامتر های کمتری را نسبت به سایر روش ها نیاز دارد. RCL یکی از اجزای بسیار مهم در فاز نخست روش فرابتکاری $GRASP$ می‌باشد. برای ساخت RCL مسأله حداقل سازی (۱.۴) را در نظر می‌گیریم. مجموعه E شامل تمام عناصر کاندید تخصیص داده نشده، می‌باشد. $C(e)$ را مقدار سود مربوط به هر عضو $e \in E$ تعریف می‌کنیم، در واقع مقدار $C(e)$ ، برابر مقدار هزینه ناشی از اضافه شدن عنصر i ام به جواب است. در هر تکرار $GRASP$ ، کوچکترین و بزرگترین مقادیر سود را برای هر عضو $e \in E$ با C_{Min} و C_{Max} نشان می‌دهیم که :

$$C_{Min} = Min\{C(e) \mid e \in E\} \quad (۲.۴)$$

$$C_{Max} = Max\{C(e) \mid e \in E\} \quad (۳.۴)$$

لیست داوطلب محدود به وسیله اعضای شدنی $e \in E$ با بهترین مقادیر سود یعنی آن دسته از اعضایی که نسبت به سایر اعضاء مقادیر سود کمتری را دارند ساخته می‌شود. به طور کلی دو روش مقداری و تعدادی، برای تشکیل مجموعه عناصر کاندید محدود شده، وجود دارد. در روش تعدادی، مجموعه عناصر کاندید محدود شده با انتخاب p عنصر از بین عناصر کاندید با کمترین مقدار هزینه، تشکیل می شود که p یک پارامتر است. در روش مقداری، درصدی از عناصر کاندید با کمترین هزینه، در مجموعه عناصر کاندید محدود شده، قرار می‌گیرند. در این جا، عناصری که مقدار هزینه آن ها از یک حد مشخص کمتر باشد، در مجموعه عناصر کاندید محمدود شده، قرار می‌گیرند. RCL به پارامتر α که $\alpha \in [۰, ۱]$ بستگی دارد. مقدار مرزی و حدی برای هر عضو $e \in E$ که در RCL قرار می‌گیرد به صورت زیر می باشد :

$$C(e) \in [C_{Min}, C_{Min} + \alpha(C_{Max} - C_{Min})] \quad (۴.۴)$$

[۲۶] Performance

هر تکرار روش $GRASP$ یک جواب نمونه از توزیع نامعلوم که میانگین و واریانس توابعی طبیعی RCL است را تولید می‌کند. اگر فرض کنیم RCL محدود به یک عضو باشد آنگاه جواب برگزیده در همه تکرار ها تولید خواهد شد، که واریانس توزیع صفر و میانگین برابر مقدار جواب حریصانه است. اگر RCL اعضای بیشتری را بپذیرد آنگاه جواب های مختلفی تولید خواهد شد که در نهایت واریانس بزرگتر می شود لذا حریصانه نقش کمتری را ایفا می‌کند و میانگین مقدار جواب بدتر خواهد شد.

گام ۱) $\emptyset$ را در $Solution$ قرار بده؛

گام ۲) مجموعه داوطلب را مقدار دهی اولیه کن و E را در C قرار بده ؛

گام ۳) برای هر عضو $e \in E$ مقدار سود و رشد $C(e)$ را ارزیابی کن؛

گام ۴) تا زمانی که $C \neq \emptyset$ که مراحل زیر را انجام بده؛

گام ۴-۱) $Min\{C(e) \mid e \in E\}$ را در C_{Min} قرار بده؛

گام ۴-۲) $Max\{C(e) \mid e \in E\}$ را در C_{Max} قرار بده؛

گام ۴-۳) $\{e \in E | C(e) \leq [C_{Min}, C_{Min} + \alpha(C_{Max} - C_{Min})]\}$ را در RCL قرار بده؛

گام ۴-۴) s را از RCL به صورت تصادفی انتخاب کن؛

گام ۴-۵) $Solution \cup \{s\}$ را در $Solution$ قرار بده؛

گام ۴-۶) مجموعه C را بروز رسانی کن؛

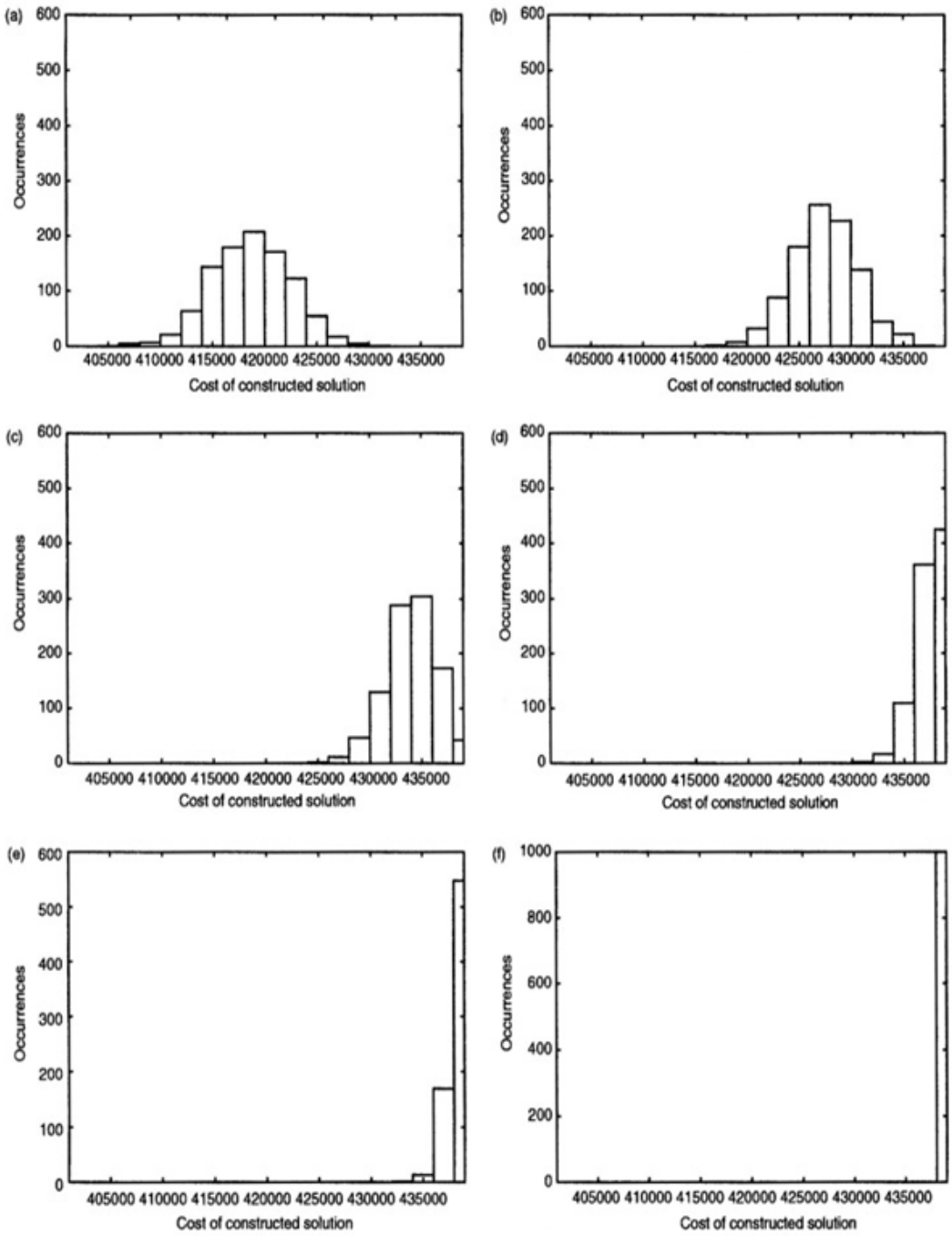
گام ۴-۷) برای هر عضو $e \in C$ مقدار سود و رشد $C(e)$ را ارزیابی کن؛

گام ۵) $Solution$ را برگردان.

۷.۴ رفتار پارامتر α

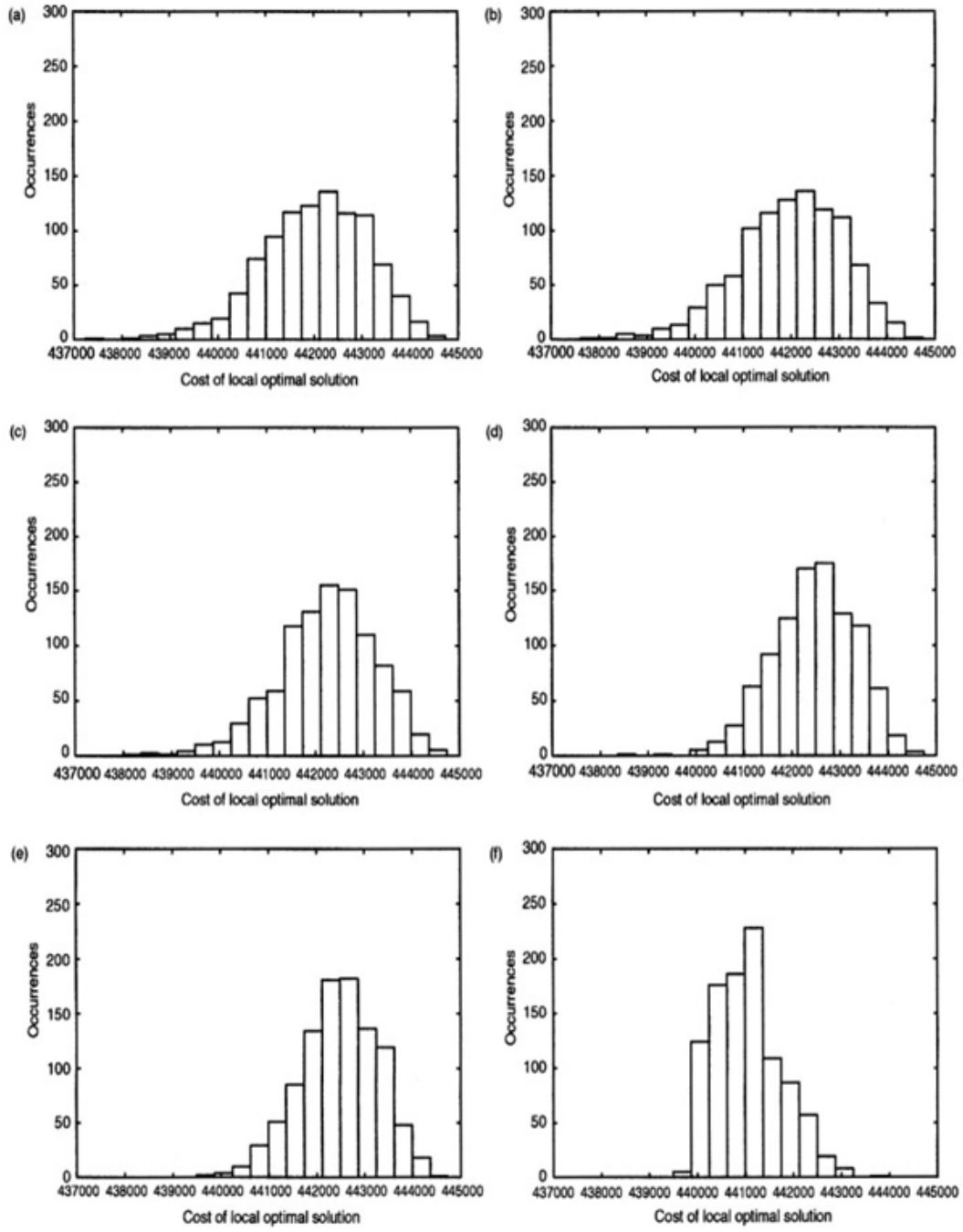
رفتار الگوریتم با تغییر پارامتر α تغییر می‌کند که $\alpha \in [۰, ۱]$ بدین صورت که هنگامی که مسأله برنامه ریزی ما به صورت حداقل سازی باشد اگر $\alpha = ۰$ باشد به الگوریتم حریصانه محض می‌رسیم و هنگامی که $\alpha = ۱$ باشد الگوریتم تصادفی محض را خواهیم داشت و در حالت دیگر اگر مسأله برنامه ریزی به صورت حداکثر سازی باشد خواهیم داشت که اگر $\alpha = ۰$ باشد به الگوریتم تصادفی محض و هنگامی که $\alpha = ۱$ باشد الگوریتم حریصانه محض را داریم. در واقع پارامتر α میزان حریصانه و تصادفی بودن الگوریتم را مشخص و بررسی می‌کند. نمودار ها در شکل (۱.۴) مقادیر مختلف α را روی مثال $MAXSAT$ [۵۲، ۵۸] با ۱۰۰ متغیر و ۸۵۰ محدودیت نشان می دهد. در نمودار (۱.۴) توزیع مقادیر جواب فاز ساخت برای پارامتر های مختلف α را مشاهده می نماییم. هنگامی که مقدار α از صفر به یک میل می‌کند میانگین مقادیر جواب نیز افزایش می‌یابد، بعبارت دیگر هنگامی که از حالت تصادفی

محض به حالت حریصانه محض می‌رویم میانگین مقادیر جواب افزایش می‌یابد در صورتی که واریانس به صفر می‌رسد. در نمودار (۳.۴) توزیع مقادیر جواب فاز جستجوی محلی را برای پارامترهای مختلف α را مشاهده می‌کنیم. نمودار شکل (۲.۴) نتایج را بر اساس کیفیت جواب و زمان سپری شده ارائه می‌دهد.



شکل ۱.۴: توزیع مقادیر جواب فاز ساخت برای ارزش های مختلف پارامتر α از تابع RCL

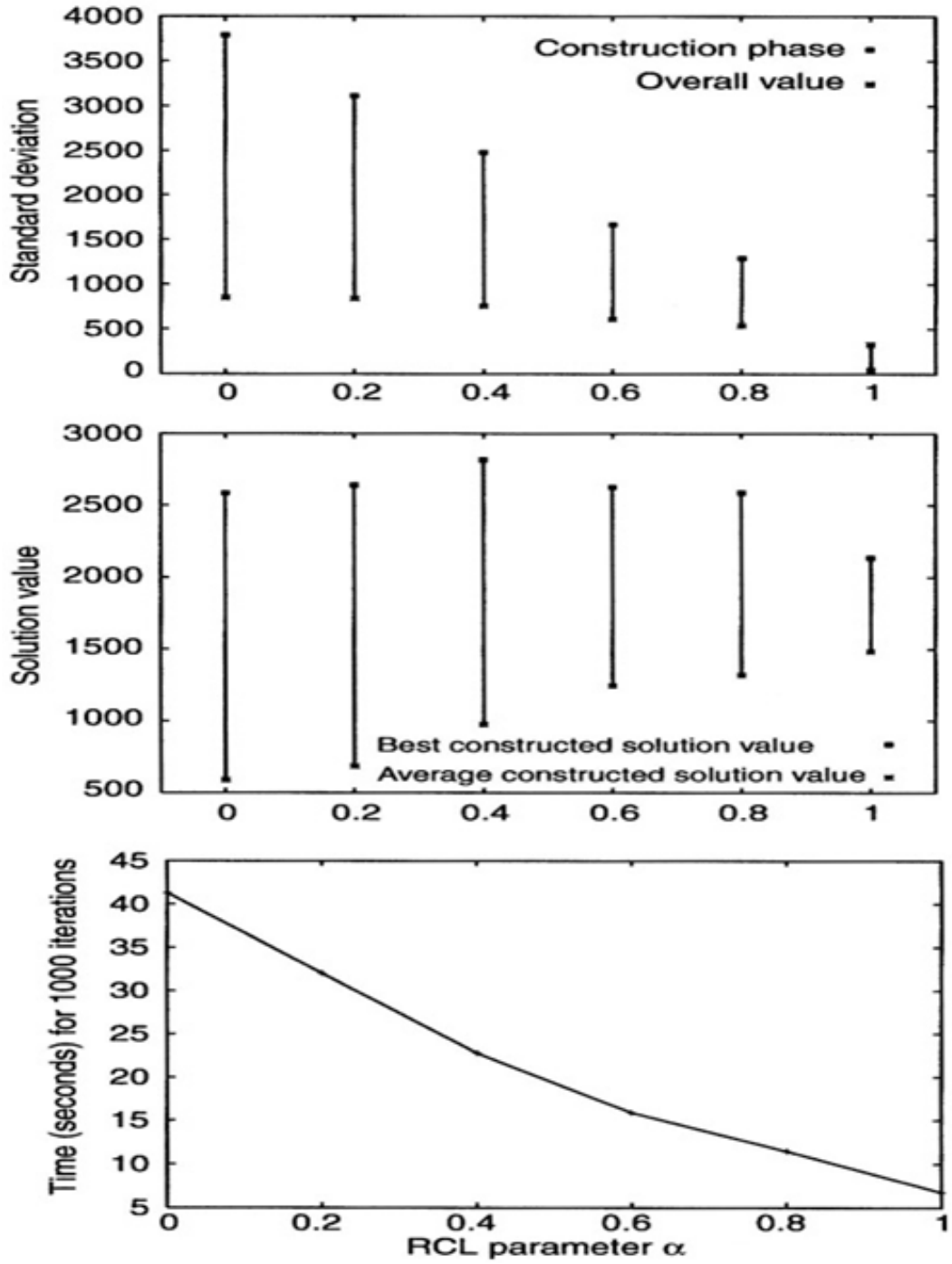
$$(a)\,\alpha = ۰/۰\,(RandomConstruction), \quad (b)\,\alpha = ۰/۲, \quad (c)\,\alpha = ۰/۴$$
$$(d)\,\alpha = ۰/۶, \quad (e)\,\alpha = ۰/۸, \quad (f)\,\alpha = ۱/۰\,(GreedyConstruction)$$



شکل ۲.۴: : توزیع مقادیر جواب فاز جستجوی محلی برای ارزش های مختلف پارامتر α از تابع RCL

$$(a)\,\alpha = ۰/۰(Random), \quad (b)\,\alpha = ۰/۲, \quad (c)\,\alpha = ۰/۴$$

$$(d)\,\alpha = ۰/۶, \quad (e)\,\alpha = ۰/۸, \quad (f)\,\alpha = ۱/۰(Greedy)$$



شکل ۳.۴: انحراف استاندارد مقادیر یافت شده، بهترین و میانگین جواب و مجموع زمان سپری شده

انتخاب مناسب پارامتر α در RCL یک شاخص بسیار پر اهمیت به شمار می‌رود و تعادلی خوب بین زمان محاسبه و کیفیت جواب ایجاد می‌کند. پاریس [۲۷] [۶۶] و ریبیرو [۲۸] [۶۸] برای پارامتر α مقادیر و راهکارهای متفاوتی رادرنظرگرفتند. آن‌ها روش $GRASP$ را معرفی کردند و برای پارامتر α مقادیر متناوب و متفاوتی را که با توجه به کیفیت جواب های اخیر بدست آمده اصلاح می‌شود در نظر گرفتند. استراتژی [۲۹] های مختلف مقادیر پارامتر α می‌تواند به صورت زیر باشد :

E: پارامتر α بصورت تصادفی، از توزیع احتمال گسسته [۳۰] و یکنواخت از بازه $[۰, ۱]$ انتخاب می شود.

R: پارامتر α با توجه به روش $GRASP$ متناوب و خودبخود تغییر می‌کند.

H: پارامتر α بصورت تصادفی از توزیع احتمال گسسته غیریکنواخت انتخاب می‌شود.

F: پارامتر α را به تدریج از مقدار یک به سمت صفر نزدیک به انتخاب حریصانه، ثابت [۳۱] در نظر می‌گیریم.

برای ارزیابی کارایی این استراتژی‌ها، آن‌ها را برای حل چهار نوع مسأله بهینه سازی آزمایش و نتایج آن در جدول (۷.۴) نمایش داده شده است، که این چهار گونه مسأله بهینه سازی عبارتند از: تجزیه ماتریس‌ها [۳۲] $(p-۱)$، مسأله ترافیک و انتقال ماهواره‌های ارتباطاتی [۳۳] $(p-۲)$ [۱۶، ۶۹] ، مجموعه پوششی [۳۴] $(p-۳)$ [۳۸] و گراف‌های دو وجهی [۳۵] $(p-۴)$ [۷۰، ۷۲] . در مورد حالت E، انتخاب پارامتر الگوریتم از مجموعه $A = \{\alpha_۱, \alpha_۲, \alpha_۳, \ldots, \alpha_m\}$ صورت می‌گیرد. در تکرار اول، تمام مقادیر α_i دارای احتمال انتخاب یکسانی برابر $\frac{m}{۱}$ هستند. بعد از چندین تکرار متوالی، احتمال انتخاب مقدار α_i برای پارامتر α بروز رسانی می‌شود. این بروز رسانی بر اساس میانگین مقدار تابع هدف جواب‌هایی است که در تکرارهای قبلی، با انتخاب مقدار α_i حاصل شده است. به این روش، تنظیم خودکار و به آن الگوریتم جستجوی تصادفی تطابقی حریصانه واکنشی گفته می‌شود. بعد از بروزرسانی، احتمال انتخاب هر مقدار α_i برای پارامتر α به کمک روابط (۵.۴) و (۶.۴) مجددا به دست می آید.

$$p_i = \frac{q_i}{\sum_{j=۱}^{m} q_j} \tag{۵.۴}$$

$$q_i = \frac{Z'}{A_i} \tag{۶.۴}$$

---

[۲۷] Prais
[۲۸] Ribeiro
[۲۹] Strategy
[۳۰] Discrete
[۳۱] Fix
[۳۲] Matrix
[۳۳] Traffic Assignment In Communication Satellite
[۳۴] Covering Set
[۳۵] Graph Planarization

در رابطه (۶.۴) مقدار Z' معادل مقدار تابع هدف بهترین جواب فعلی و A_i برابر میانگین مقدار تابع هدف به ازای تمام جواب‌هایی است که با انتخاب مقدار α_i برای پارامتر α به دست آمده است.
در مورد حالت R مجموعه مقادیر متفاوتی از α را در نظر میگیریم. در حالت H مقادیر به صورت زیر می‌باشد:

$$P(=۰/۱)=۰/۵, P(=۰/۲)=۰/۲۵, P(=۰/۳)=۰/۱۲۵, P(=۰/۴)=۰/۰۳$$

$$P(=۰/۵)=۰/۰۳, P(=۰/۶)=۰/۰۳, P(=۰/۷)=۰/۰۱, P(=۰/۸)=۰/۰۱$$

$$P(=۰/۹)=۰/۰۱, P(=۱/۰)=۰/۰۵$$

و سر انجام استراتژی آخر یعنی حالت F، این مکانیزم کاهشی، موجب می‌شود که الگوریتم در تکرارهای اولیه، جستجو را در فضای جواب انجام داده و سپس با کاهش مقدار α، تمایل الگوریتم به جستجو در بین جواب‌های با کیفیت بالاتر پیش می‌رود. مقدار پارامتر که در تکرار i ام از الگوریتم برای تولید جواب استفاده می‌شود، از رابطه (۷.۴) به دست می‌آید. در این رابطه، مقدار پارامتر $maxiter$ برابر حداکثر تعداد گام‌های اجرای الگوریتم $GRASP$ می‌باشد.

$$\alpha_i = ۱ - \frac{i}{maxiter} \qquad (۷.۴)$$

در استراتژی E تمرکز روی معیار متنوع سازی می‌باشد و در مورد F تعادل نسبی بین معیارهای متنوع سازی و پرقدرت سازی برقرار شده است. در استراتژی R تمرکز بیشتر روی معیار پرقدرت سازی و کیفیت جواب می‌باشد.

جدول ۱.۴: ارزیابی و مقایسه استراتژی‌های مختلف مقادیر پارامتر α

		R		*E*		*H*		*F*	
Problem	Total	Hits	Time	Hits	Time	Hits	Time	Hits	Time
P-1	36	34	579.0	35	358.2	32	612.6	24	642.8
P-2	7	7	1346.8	6	1352.0	6	668.2	5	500.7
P-3	44	22	2463.7	23	2492.6	16	1740.9	11	1625.2
P-4	37	28	6363.1	21	7292.9	24	6326.5	19	5972.0
Total	124	91		85		78		59	

فصل ۵

مسیر پیوستگی

۱.۵ مقدمه

مسیر پیوستگی [۱] [۴] یک شاخص مهم در $GRASP$ است که منجر به بهبود قابل توجهی در کیفیت جواب می‌شود. مسیر

پیوستگی در ابتدا توسط گلاوور [۴۹] به عنوان یک استراتژی تقویت برای کشف خط سیرها و مسیرهای اتصال هر جواب بدست آمده با استفاده از جستجوی تابو یا جستجوی پراکنده معرفی گردید و بعدها توسط لاگونا [۲] [۵۰]، [۵۱]،[۶۰] و مارتی [۳] [۴۳]،[۵۱] در برخی مسائل دیگر به کار گرفته شد. روش مسیر پیوستگی استراتژی ساده برای بهبود جواب می‌باشد. جستجوي پخشي (يا اتصال مجدد مسير) مجموعه‌اي از جواب‌ها را از مجموعه جواب‌هاي مرجع انتخاب شده متناظر با جواب‌هاي عملي مسأله، توليد مي كند. اين كار با ساخت تركيباتي از زيرمجموعه‌هاي مجموعه مرجع فعلي انجام مي‌گيرد . جواب هاي حاصل، جواب‌هاي آزمايشي [۴] ناميده مي‌شوند. آنها ممكن است از لحاظ عملي ممكن نباشند و به همين دليل به وسيله رويه‌هاي تعمير فضاي جستجو اصلاح مي‌گردند. سپس يک مكانيزم بهبود جهت بهتر ساختن مجموعه جواب‌های آزمايشي (كه معمولا يک جستجوي محلي است) اعمال مي‌گردد. اين جواب‌هاي بهبود يافته، مجموعه جواب‌هاي پراكنده [۵] [۴] را تشكيل مي‌دهند. مجموعه جديدي از جواب‌هاي مرجع كه در تكرار بعدي استفاده خواهند شد از مجموعه مرجع جاري و جواب‌هاي پراكنده، انتخاب مي‌گردد. مسیر پیوستگی از یک یا تعدادی از مسیرها آغاز می‌شود، مسیر ها در فضای جواب به سمت جواب‌های برتر هدایت می‌شوند تا جواب بهتری را کشف کنند. این مسیر به وسیله انتخاب حرکت‌هایی که جواب‌هادی یا جواب راهنما [۶] نامیده می‌شوند به انجام می‌رسد.

---

[۱] Path Relinking

[۲] Lagouna

[۳] Marti

[۴] Trial

[۵] Dispersed

[۶] Guiding

۲.۵ ساختار روش مسیر پیوستگی

استفاده از مسیر پیوستگی در روش $GRASP$ به عنوان یک استراتژی تقویت برای هر جواب بهینه مکانی بکار می‌رود. روش با محاسبه تفاضل متقارن بین دو جواب، جواب آغازی و جواب هدف آغاز می‌شود یعنی مجموعه‌ای از حرکات لازم است که از یا جواب آغازی به جواب هدف برسند. مسیری از جواب‌ها، به وسیله ارتباط بین جواب‌های و تولید می‌شود. مسیر پیوستگی برای زوج مرتب و جفت [۷] های جواب به کار گیری می‌شود که در آن جواب بهینه مکانی است که به وسیله هر تکرار $GRASP$ بعد از جستجوی محلی تولید می‌شود و یکی از معدود جواب‌های مجموعه برتر [۸] می‌باشد که از منبع به صورت تصادفی ساخته شده است. بهترین جواب x^* در این مسیر به وسیله الگوریتم برمی‌گردد. در هر مرحله، روش همه حرکات و جابجایی‌های $m \in \triangle(x, x_t)$، را از جواب جاری x، بررسی می‌کند و یکی از آن هایی که کمترین مقدار جواب را دارد انتخاب می‌کند، یعنی آن جابجایی‌هایی که کمترین $f(x \oplus m)$ را داشته باشد. $x \oplus m$ جواب حاصل از به کار گرفتن m جابجایی و حرکت از جواب x می‌باشد. بهترین جابجایی، m^* است که جواب $x \oplus m^*$ را تولید می کند و روش هنگامی که به جواب برسیم خاتمه می‌یابد یعنی زمانی که تساوی $\triangle(x, x_t) = \emptyset$ برقرار باشد. شبه کد زیر، ساختار و چگونگی کار الگوریتم را نشان می‌دهد:

۳.۵ شبه کد های مربوط به روش مسیر پیوستگی

شبه کد روش مسیر پیوستگی به صورت زیر می‌باشد :
ورودی : جواب آغازین و جواب هدف
خروجی : بهترین جواب x^* در مسیری بین x_s و x_t

گام ۱) تفاضل متقارن بین x_s و x_t یعنی $\triangle(x_s, x_t)$ را محاسبه کن ؛

گام ۲) $min\{f(x_s), f(x_t)\}$ را محاسبه کن و مقدارش را در f^* قرار بده؛

گام ۳) $argmin\{f(x_s), f(x_t)\}$ را محاسبه و مقدار آن را در f^* قرار بده؛

گام ۴) x_s را در x قرار بده؛

گام ۵) تا زمانی که $\triangle(x_s, x_t) \neq \emptyset$ مراحل زیر را انجام بده؛

گام ۵-۱) $argmin\{f(x \oplus m) : m \in \triangle(x_s, x_t)\}$ را محاسبه کن و مقدارش را در m^* قرار بده؛

گام ۵-۲) $\triangle(x_s, x_t) - m^*$ را محاسبه کن و در $(x \oplus m^*, x_t)$ قرار بده؛

گام ۵-۳) $x \oplus m^*$ را در x قرار بده؛

---

[۷] Pair

[۸] Elite Set

گام ۴-۵) اگر $f(x) < f^*$ آن گاه $f(x)$ را در f^* قرار بده؛

گام ۵-۵) x را در x^* قرار بده.

همچنین شبه کد روش $GRASP$ با مسیر پیوستگی برای مسائل حداقل سازی به صورت زیر است :
ورودی : تعداد تکرار ها $imax$
خروجی : جواب بهینه $x^* \in X$

گام ۱) $\emptyset$ را در P قرار بده؛

گام ۲) ∞ را در f^* قرار بده؛

گام ۳) برای $i = ۱, \ldots, imax$ مراحل زیر را انجام بده؛

گام ۱-۳) فاز ساخت تصادفی حریصانه را روی X انجام بده و در X مقدارش را بگذار؛

گام ۲-۳) جستجوی محلی را برای X انجام بده و مقدار آن را در X قرار بده؛

گام ۳-۳) اگر $i \geq ۲$ آنگاه، جواب های $Y \subseteq P$ از منبع را به صورت تصادفی برای پیوستن به x انتخاب کن؛

گام ۴) برای $y \in Y$ مراحل زیر را انجام بده؛

گام ۱-۴) x_s برای جواب آغازین و x_t برای جواب هدف که x یا y است را تعیین کن؛

گام ۲-۴) مسیر پیوستگی بین (x_s, x_t) را محاسبه کن و مقدارش را در x_p قرار بده؛

گام ۵) اگر $f(x_p) < f^*$ آن گاه $f(x_p)$ را در f^* قرار بده؛

گام ۱-۵) مقدار x_p را در x^* قرار بده؛

گام ۶) مقدار $PostOptimizeP$ را در P قرار بده؛

گام ۷) مقدار $argmin\{f(x), x \in P\}$ را در x^* بگذار؛

۴.۵ استراتژی های مختلف مسیر پیوستگی

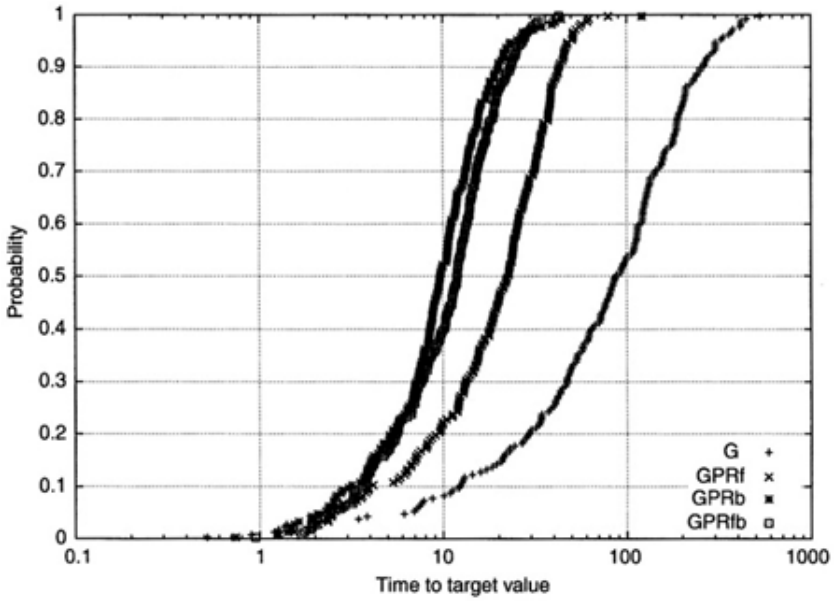
شیوه‌های متفاوتی برای اجرا و بکارگیری روش مسیر پیوستگی در $GRASP$ معرفی و ارائه شده است، در زیر به معرفی این شیوه‌ها می‌پردازیم:
۱) G [۹] : در این استراتژی مسیر پیوستگی را بکارگیری نمی‌کنیم و $GRASP$ به تنهایی اجرا می‌شود.

[۹] Pure Grasp

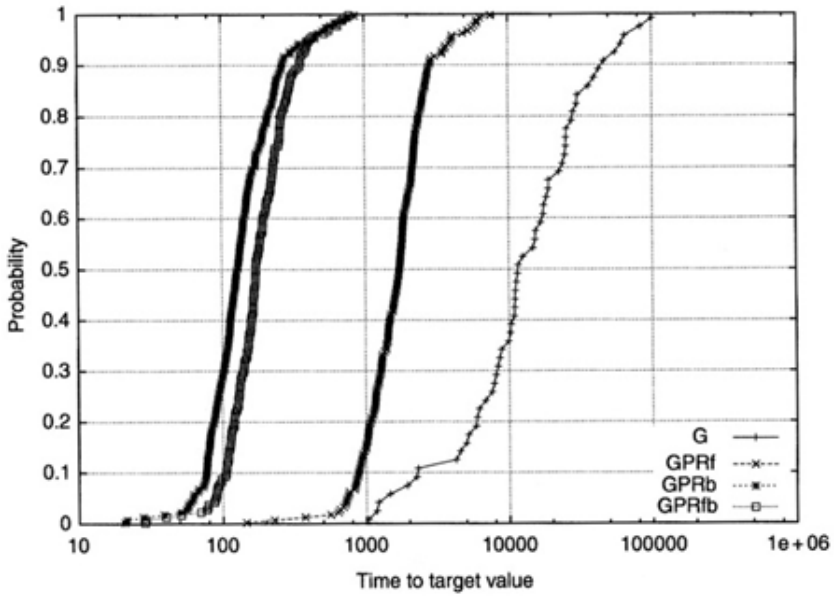
۲) $GPRF$ [۱۰] : این نوع استراتژی علاوه بر روش G مسیر پیوستگی از جواب بهینه مکانی آغاز می‌شود و جواب هادی و راهنما [۱۱] را به صورت تصادفی از مجموعه برتر [۱۲] انتخاب می‌نماید. در واقع در این حالت این روش را روش پیشرو [۱۳] می‌نامند.
۳) $GPRB$ [۱۴] : این نوع علاوه بر G، مسیر پیوستگی که از انتخاب تصادفی جواب از مجموعه جواب برتر آغاز و از جواب بهینه مکانی به عنوان جواب هادی یا راهنما استفاده می‌کند. این روش را روش پسرو [۱۵] می نامند .
۴) GPRFB [۱۶] : این نوع استراتژی ترکیب $GPRF$ و $GPRB$ است که مسیر پیوستگی از دو جهت [۱۷] انجام می‌شود.
این چهار استراتژی را از نظر کیفیت جواب و پیچیدگی زمانی مقایسه و ارزیابی می‌نماییم. برای مطالعه تأثیر پذیری این چهار استراژی در مسیر پیوستگی روی $GRASP$ ، آن ها روی دو مثال Att و $Fr۷۵۰a$[۷۱] آزمایش و بررسی شده‌اند. در هر تکرار $GRASP$ اجرای‌های مستقلی در نظر گرفته شده است. نسبت رفتارهای این چهار گونه را روی دو مثال فوق در شکل (۱.۵) و (۲.۵) نمایش داده شده است. به عنوان مثال در ۱۰۰ ثانیه احتمال برای $GPRFB$، ۹.۲۵٪ می باشد در حالی که این احتمال برای $GPRB$ به ۲۸.۷۵٪ افزایش می‌یابد. در ۲۰۰ ثانیه احتمال برای G به ۸.۳۳٪ می‌رسد در صورتی که این احتمال برای $GPRF$ به ۶۵.۲۵٪ افزایش می‌یابد. حالتی را که در آن احتمال ۵۰٪ باشد را در نظر می‌گیریم بازه های زمانی برای چهار گونه $GPRBG, GPRF, GPRFB$ ،به ترتیب برابر ۱۲۹، ۱۷۲، ۱۷۲۷، ۱۰۹۳۳ می‌باشند. مطابق این نتایج گونه $GPRB$ که مسیر پیوستگی با حالت پسرو یعنی جواب برتر به عنوان جواب آغازین و از جواب بهینه مکانی به عنوان جواب هادی یا راهنما استفاده می‌کند، تأثیر گذاری بیشتری نسبت به سایر گونه‌های دیگری را دارد که در شکل (۳.۵) رفتار استراتژی $GPRB$ را روی مثال Att نمایش می‌دهیم. در جدول (۴.۵) مقادیر جواب با زمان ثابت به ترتیب ۱۰ و ۱۰۰ ثانیه را روی مثال Att برای چهار استراتژی مورد نظر و بحث شده ارائه شده است. در این جدول میانگین و بهترین مقادیر جواب در هر اجرا را به دست آمده است. با مشاهده جدول می‌بینیم که در حالات پسرو به طور سیستماتیک به جواب بهتری می‌رسیم. استراتژی‌های $GPRB$ و $GPRFB$ رفتارهای یکسانی را دارند. استراتژی $GPRB$ نتایج بهتری را در هر دو مورد میانگین و بهترین مقادیر جواب به دست می‌آورد. استراتژی $GPRB$ در محدوده زمانی ۱۰ ثانیه و استراتژی $GPRFB$ در محدوده زمانی ۱۰۰ ثانیه به نتایج بهتری می‌رسد. در مورد $GPRB$ بهترین مقادیر طی ۷ اجرا و $GPRFB$ در تنها دو اجرا یافت خواهد شد. این در حالی است که وقتی زمان به ۱۰۰ ثانیه افزایش می‌یابد $GPRB$ بهترین مقادیر را در طی چهار مرحله اجرا و $GPRFB$ بهترین مقادیر را طی

[۱۰] Grasp With Forward Path Relinking
[۱۱] Guiding
[۱۲] Elite Set
[۱۳] Forward
[۱۴] Grasp With Backward Path Relinking
[۱۵] Backward
[۱۶] Grasp With Forward - Backward Path Relinking
[۱۷] Direction

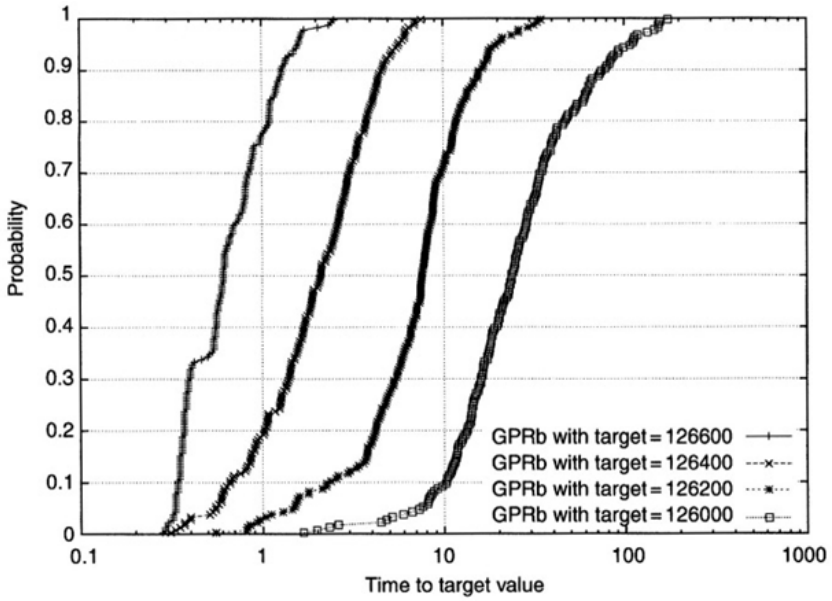
پنج اجرا یافت خواهند کرد.



شکل ۱.۵: توزیع های زمانی مقادیر جواب برای $GRASP, GPRF, GPRB, GPRFB$ روی مثال $FR۷۵۰a$



شکل ۲.۵: توزیع های زمانی مقادیر جواب برای $GRASP, GPRF, GPRB, GPRFB$ روی مثال att



شکل ۳.۵: : توزیع های زمانی مقدار جواب هدف برای $GPRB$ روی مثال att

جدول ۱.۵: مقادیر جواب با زمان ثابت روی مثال att

	10 s		100 s	
Variant	Best	Average	Best	Average
GPR	126602.883	126694.666	126227.678	126558.293
GPRf	126301.118	126578.323	126082.790	126228.798
GPRb	125960.336	126281.156	125665.785	125882.605
GPRfb	125961.118	126306.736	125646.460	125850.396

پیوست آ

نمایش کدهای استفاده شده

آ.۱ شبه کد گرسپ با مسیر پیوستگی برای مسائل حداقل سازی

```
Data : Number of iterations imax
Result: Solution x*   X
P ←\emptyset ;
f ^* ← ϖ ;
for i = 1, . . . , imax do
x ← GreedyRandomizedConstruction(x) ;
x ← LocalSearch(x);
if i   2 then
Choose, at random, pool solutions Y  º P to relink with x ;
for y  ¸ Y do
Determine which (x or y) is initial xs and which is target xt ;
x_p ← PathRelinking(x_s,x_t);
Update the elite set P with x_P;
if f ( x_p) < f * then
f ^* ← f ( x_p);
x ^* ← x_p;
end
end
end
P  = PostOptimize{P };
x^*  = argmin{f (x), x   P };
```

آ.۲ شبه کد روش مسیر پیوستگی

```
Data : Starting solution xs and target solution xt
Result: Best solution x*  in path from xs  to xt
Compute symmetric difference Δ( x_s,x_t );
f^*  ← min{f ( x_s), f (x_t )};
x^*  ← argmin{f (x_s ), f (x_t )};
x ← ; x_s
while Δ(x,x_t) = \emptyset do
m^* ← argmin{f (x \oplus m) : m  Δ(x,x_t)}; Δ(x \oplus m*,x_t ) ← Δ(x, x_t) \ {m^*};
x ← x \oplus m^*;
if f (x) < f ^*  then
f^*  ← f (x);
x^*  ← x;
end
end
```

آ.۳ پالایش و تعمیم شبه کد فاز ساخت

```
P  ← \emptyset
Initialize the candidate set : C ← E ;
Evaluate the incremental cost c(e) for all e \in C ;
While C   \neq  \emptyset Do
C_Min ← Min{C(e)| e\in C}
C_Max ← Max{C(e)| e  \in C}
RCL   ← {e  \in  C|C(e) <= [C_Min, C_Min+a(C_Max - C_Min)]}
 Select an element s from the RCL at random
Solution   ← Solution  { s }
Update the candidate set C ;
Reevaluate the increment costs c(e) for all e \in C ;
End;
Return solution ;
```

آ.۴ شبه کد فاز ساخت روش GRASP

```
Solution  ← \emptyset
Evaluate the incremental cost of the candidate elements ;
While solution is not a complete solution do
Build the restricted candidate list (RCL);
Select an element s from the RCL at random ;
Solution ← Solution  \cup  { s };
Reevaluate the incremental costs ;
end ;
Return solution ;
```

آ.۵ شبه کد GRASP برای مسائل مینیمم سازی

```
Data : Number of iterations imax
Result: Solution x^*  \in   X
f ^* ←  ϖ;
for i = 1, . . . , imax do
x ← GreedyRandomizedConstruction();
x ← LocalSearch(x);
if f ( x_p) < f * then
f ^* ← f (x);
x^* ← x;
end
end
```

آ.۶ شبه کد روش فرا ابتکاری GRASP

```
Read _ input ;
for k = 1 , . . . , Max - Interations do
Solution Greedy_Randomized_Construction(seed) ;
Solution Local_Search(Solution);
Update_Solution(Solution , Best _ Solution);
end ;
return Best _Solution ;
```

مراجع

[1] S. Abdinnour-Helm and S .W. Hadley (2000) Tabu search based heuristics for multi-floor facility layout. International Journal of Production Research, 38, 365-383.

[2] J. Abello, P.M. Pardalos and M.G.C. Resende (1999) On maximum clique problems in very large graphs. In: J. Abello and J. Vitter (eds.), External Memory Algorithms and Visualization, volume 50 of DIMACS Series on Discrete Math-ematics and Theoretical Computer Science. American Mathematical Society, pp. 199-130.

[3] R .K. Ahuja, J.B. Orlin and A. Tiwari (2000) A greedy genetic algorithm for the quadratic assignment problem. Computers and Operations Research, 27, 917-934.

[4] R .M. Aiex, M.G.C. Resende, P.M. Pardalos and G. Toraldo (2000) GRASP with path-relinking for the three-index assignment problem. Technical report, AT & T Labs-Research.

[5] R .M. Aiex, M.G.C. Resende and C.C. Ribeiro (2002) Probability distribution of solution time in GRASP: an experimental investigation. Journal of Heuristics, 8, 343-373.

[6] A .C. Alvim (1998) Parallelization strategies for the metaheuristic GRASP. Master's thesis, Department of Computer Science, Catholic University of Rio de Janeiro, Brazil (in Portuguese).

[7] A .C. Alvim and C.C. Ribeiro (1998) Load balancing for the parallelization of the GRASP metaheuristic. In: Proceedings of the X Brazilian Symposium on Computer Architecture, B zios, pp. 279-282 (in Portuguese).

[8] S. Areibi and A. Vannelli (1997) A GRASP clustering technique for circuit partitioning. In: J. Gu and P.M. Pardalos (eds.), Satisfiability Problems, Volume 35 of DIMACS Series on Discrete Mathematics and Theoretical Computer Science. American Mathematical Society, pp. 711-724.

[9] M .F. Argüello, J.F. Bard and G. Yu (1997) A GRASP for aircraft routing in response to groundings and delays. Journal of Combinatorial Optimization, 1, 211-228.

[10] M .F. Argüello, T.A. Feo and O. Goldschmidt (1996) Randomized methods for the number partitioning problem. Computers and Operations Research, 23, 103-111.

[11] M .Armony, J.C. Klincewicz, H. Luss and M.B. Rosenwein (2000) Design of stacked self-healing rings using a genetic algorithm. Journal of Heuristics, 6, 85-105.

[12] J .B. Atkinson (1998) A greedy randomised search heuristic for time-constrained vehicle scheduling and the incorporation of a learning strategy. Journal of the Operational Research Society, 49, 700-708.

[13] J .F. Bard and T.A. Feo (1989) Operations sequencing in discrete parts manufacturing. Management Science, 35, 249-255.

[14] J .F. Bard and T.A. Feo (1991) An algorithm for the manufacturing equipment selection problem. IIE Transactions, 23, 83-92.

[15] J .F. Bard, T.A. Feo and S. Holland (1996) A GRASP for scheduling printed wiring board assembly. IIE Transactions, 28, 155-165.

[16] J .F. Bard, L. Huang, P. Jaillet and M. Dror (1998) A decomposition approach to the inventory routing problem with satellite facilities. Transportation Science, 32, 189-203.

[17] R .Battiti and G. Tecchiolli (1992) Parallel biased search for combinatorial optimization: Genetic algorithms and tabu. Microprocessors and Microsystems, 16, 351-367.

[18] J .E. Beasley (1990) OR-Library: Distributing test problems by electronic mail. Journal of the Operational Research Society, 41, 1069-1072.

[19] S .Binato, W.J. Hery, D, Loewenstern and M.G.C. Resende (2002) A GRASP for job shop scheduling. In: C.C. Ribeiro and P. Hansen (eds.), Surveys in Metaheuristics. Kluwer

[20] S . Binato and G.C. Oliveira (2002) A reactive GRASP for transmission network expansion planning. In: C.C. Ribeiro and P. Hansen (eds.), Essays and Surveys in Metaheuristics. Kluwer Academic Publishers, pp. 81-100.

[21] J .L. Bresina (1996) Heuristic-biased stochastic sampling. In: Proceedings of the Thirteenth National Conference on Artificial Intelligence. Portland, pp. 271-278.

[22] S .A. Canuto, M.G.C. Resende and C.C. Ribeiro (2001) Local search with perturbations for the prize-collecting Steiner tree problem in graphs. Networks, 38, 50-58.

[23] S .A. Canuto, C.C. Ribeiro and M.G.C. Resende (1999) Local search with perturbations for the prize-collecting Steiner tree problem. In: Extended Abstracts of the Third Metaheuristics International Conference. Angra dos Reis, pp. 115-119.

[24] C. Carreto and B. Baker (2002) A GRASP interactive approach to the vehicle routing problem with backhauls. In: C.C. Ribeiro and P. Hansen (eds.), Essays and Surveys in Metaheuristics. Kluwer Academic Publishers, pp. 185-199.

[25] I. Charon and O. Hudry (1993) The noising method: a new method for I. combinatorial optimization. Operations Research Letters, 14, 133-137.

[26] I. Charon and O .Hudry (2002) The noising methods: a survey. In: C.C. Ribeiro and C.C. Ribeiro (eds.), Essays and Surveys in Metaheuristics. Kluwer Academic Publishers, pp. 245-261.

[27] V .D. Cung, S .L. Martins, C .C. Ribeiro and C. Roucairol (2002) Strategies for the parallel implementation of metaheuristics. In: C.C. Ribeiro and C.C. Ribeiro (eds.), Essays and Surveys in Metaheuristics. Kluwer Academic Publishers, pp. 263-308.

[28] P .De, J .B. Ghosj and C .E. Wells (1994) Solving a generalized model for con due date assignment and sequencing. International Journal of Production Economics, 34, 179-185.

[29] H .Delmaire, J .A. Diaz, E . Fern ndez and M. Ortega (1999) Reactive GRASP and Tabu Search based heuristics for the single source capacitated plant location problem. INFOR, 37, 194-225.

[30] A .S. Deshpande and E . Triantaphyllou (1998) A greedy randomized adaptive search procedure (GRASP) for inferring logical clauses from examples in polynomial time and some extensions. Mathematical Computer Modelling, 27, 75-99.

[31] N .Dodd (1990) Slow annealing versus multiple fast annealing runs: an empirical investigation. Parallel Computing, 16, 269-272.

[32] A .Drexl and F. Salewski (1997) Distribution requirements and compactness constraints in school timetabling. European Journal of Operational Research, 102, 193-214.

[33] H .T. Eikelder, M .Verhoeven, T .Vossen and E .Aarts (1996) A probabilistic analysis of local search. In: I. Osman and J. Kelly (eds.), Metaheuristics: Theory and Applications. Kluwer Academic Publishers, pp. 605-618.

[34] M .G .C. Resende and C .C .Ribeiro (2001) A GRASP with path-relinking for private virtual circuit routing. Technical report, AT & T Labs Research

[35] T .A. Feo and J .F. Bard (1989) The cutting path and tool selection problem in computer-aided process planning. Journal of Manufacturing Systems, 8, 17-26.

[36] T .A. Feo, J .F. Bard and S. Holland (1995) Facility-wide planning and scheduling of printed wiring board assembly. Operations Research, 43, 219-230.

[37] T .A. Feo and J .L. Gonz ález-Velarde (1995) The intermodal trailer assignment problem: models, algorithms, and heuristics. Transportation Science, 29, 330-341.

[38] T .A. Feo and M .G. C. Resende (1989) A probabilistic heuristic for a computationally difficult set covering problem. Operations Research Letters, 8, 67-71.

[39] T .A. Feo and M .G .C. Resende (1995) Greedy randomized adaptive search procedures. Journal of Global Optimization, 6, 109-133.

[40] T .A. Feo, M. .G. C. Resende and S.H. Smith (1994) A greedy randomized adaptive search procedure for maximum independent set. Operations Research, 42, 860-878.

[41] T .A. Feo, K. Sarathy and J .McGahan (1996) A GRASP for single machine scheduling with sequence dependent setup costs and linear delay penalties. Computers and Operations Research, 23, 881-895.

[42] T.A. Feo, K. Venkatraman and J.F. Bard (1991) A GRASP for a difficult single machine scheduling problem. Computers and Operations Research, 18, 635-643.

[43] E. Fernndez and R. Mart(1999) GRASP for seam drawing in mosaicking of aerial photographic maps. Journal of Heuristics, 5, 181-197.

[44] P . Festa and M .G. C. Resende (2002) GRASP: an annotated bibliography. In: C .C. Ribeiro and P .Hansen (eds.), Essays and Surveys in Metaheuristics. Kluwer Academic Publishers, pp. 325-367.

[45] P .Festa, M .G. C. Resende, P. Pardalos and C .C. Ribeiro (2001) GRASP and VNS for Max-Cut. In: Extended Abstracts of the Fourth Metaheuristics International Conference. Porto, pp. 371-376.

[46] C. Fleurent and F. Glover (1999) Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory. INFORMS Journal on Computing, 11, 198-204.

[47] J.B. Ghosh (1996) Computatinal aspects of the maximum diversity problem. Operations Research Letters, 19, 175-181.

[48] F .Glover (1996) Tabu search and adaptive memory programming—advances, applications and challenges. In: R.S. Barr, R.V. Helgason and J.L. Kennington (eds.), Interfaces in Computer Science and Operations Research. Kluwer, pp. 1-75.

[49] F . Glover (2000) Multi-start and strategic oscillation methods—principles to exploit adaptive memory. In: M. Laguna and J.L. Gonz les-Velarde (eds.), Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research. Kluwer, pp. 1-24.

[50] F . Glover and M. . Laguna (1997) Tabu Search. Kluwer.

[51] F . Glover, M. .Laguna and R. Mart (2000) Fundamentals of scatter search and path relinking. Control and Cybernetics, 39, 653-684.

[52] M .G. C. Resende, L .S. Pitsoulis and P.M. Pardalos (1997) Approximate solution of weighted MAX-SAT problems using GRASP. In: J. Gu and P.M. Pardalos (eds.), Satisfiability Problems, volume 35 of DIMACS Series on Discrete Math-ematics and Theoretical Computer Science, American Mathematical Society, pp. 393-405.

[53] P .L. Hammer and D .J. Rader, Jr. (2001) Maximally disjoint solutions of the set covering problem. Journal of Heuristics, 7, 131-144.

[54] P . Hansen (2002) Developments of variable neighborhood search. In: C.C. Ribeiro and P. Hansen (eds.), Essays and Surveys in Metaheuristics. Kluwer Academic Publishers, pp. 415-439.

[55] J .P. Hart and A .W. Shogan (1987) Semi-greedy heuristics: an empirical study. Operations Research Letters, 6, 107-114.

[56] H . Hoos and T . Stützle (1999) Towards a characterisation of the behaviour of stochastic local search algorithms for SAT. Artificial Intelligence, 112, 213-232.

[57] J .G. Klincewicz (1992) Avoiding local optima in the p-hub location problem using tabu search and GRASP. Annals of Operations Research, 40, 283-302.

[58] P .M. Pardalos, L .S. Pitsoulis and M.G.C. Resende (1996) A parallel GRASP for MAX-SAT problems. Lecture Notes in Computer Science, 1184, 575-585.

[59] G . Kontoravdis and J.F. Bard (1995) A GRASP for the vehicle routing problem with time windows. ORSA Journal on Computing, 7, 10-23.

[60] M . Laguna, T.A. Feo and H.C. Elrod (1994) A greedy randomized adaptive search procedure for the two-partition problem. Operations Research, 42, 677-687.

[61] M . Laguna and J.L. Gonzlez-Velarde (1991) A search heuristic for just-intime scheduling in parallel machines. Journal of Intelligent Manufacturing, 2, 253-260.

[62] M . Laguna and R. Mart (1999) GRASP and path relinking for 2-layer straight line crossing minimization. INFORMS Journal on Computing, 11, 44-52.

[63] C .C. Ribeiro and M .C. Souza (2002) Variable neighborhood search for the degree constrained minimum spanning tree problem. Discrete Applied Mathematics, 118, 43-54

[64] P . Hansen (1997) Variable neighborhood search. Computers and Operations Research, 24, 1097-1100.

[65] L . Osborne and B. Gillett (1991) A comparison of two simulated annealing algorithms applied to the directed Steiner problem on networks. ORSA Journal on Computing, 3, 213-225.

[66] M . Prais and C .C. Ribeiro (1999) Parameter variation in GRASP implementations. In: Extended Abstracts of the Third Metaheuristics International Conference. Angra dos Reis, pp. 375-380.

[67] C .C. Ribeiro and M .C. Souza (2002) Variable neighborhood search for the degree constrained minimum spanning tree problem. Discrete Applied Mathematics, 118, 43-54

[68] M . Prais and C .C. Ribeiro (2000) Parameter variation in GRASP procedures. Investigaci n Operativa, 9, 1-20.

[69] M .Prais and C .C. Ribeiro (2000) Reactive GRASP: an application to a matrix decomposition problem in TDMA traffic assignment. INFORMS Journal on Computing, 12, 164-176.

[70] M .G. C. Resende and C .C. Ribeiro (1997) A GRASP for graph planarization. Networks, 29, 173-189.

[71] M .G .C. Resende and C .C. Ribeiro (2001) A GRASP with path-relinking for private virtual circuit routing. Technical report, AT & T Labs Research

[72] C .C. Ribeiro and M .G. C. Resende (1999) Algorithm 797: Fortran subroutines for approximate solution of graph planarization problems using GRASP. ACM Transactions on Mathematical Software, 25, 342-352.

واژه‌نامه فارسی به انگلیسی

آر سی ال ..... RCL
آي دي آ ..... IDA
اس ام آ ..... SMA
الگوریتم ژنتیک ..... Genetic Algorithm
انحراف ..... Deviation
بنا کننده ..... Constructive
بهینه سازي ..... Optimization
پارامتر ..... Parameter
پی – مرکز ..... P - Center
پی – میانه ..... P - Median
تابع بهبود ..... Improvement Function
تابع هدف ..... Target Function
تابع هدف ..... Target Function
ترکیب ..... Combination
تحقیق در عملیات ..... Operation Research
تصادفی ..... Random
جستجوي پرتوي ..... Beam Search
جستجوي تابو ..... Tabu Search
جستجوي حریصانه ..... Greedy Search
جستجوي محلی ..... Local Search
رأس ..... Vertex
رتبه ..... Rank
سرویس دهنده ..... Facility
شبکه ..... Net
شبه کد ..... Psedo - Cod

فرا ابتکاري ..... Metaheuristic
فروشنده دوره گرد ..... Travelling Salesman
کاربرد ..... Application
کاهش ..... Decrease
کوله پشتی ..... Knapsack
گرسپ ..... Grasp
مجموعه پوششی ..... Set Covering
مسیر ..... Path
مسیر یابی ..... Routing
مکان یابی ..... Location
میانگ ..... Average
نمونه ..... Sample
همسایگی ..... Neighbourhood

واژه‌نامه انگلیسی به فارسی

کاربرد ..... Application
میانگین ..... Average
جستجوي پرتوي ..... Beam Search
ترکیب ..... Combination
بنا کننده ..... Constructive
کاهش ..... Decrease
انحراف ..... Deviation
سرویس دهنده ..... Facility
الگوریتم ژنتیک ..... Genetic Algorithm
گرسپ ..... Grasp
جستجوي حریصانه ..... Greedy Search
آي دي آ ..... IDA
تابع بهبود ..... Improvement Function
کوله پشتی ..... Knapsack
جستجوي محلی ..... Local Search
مکان یابی ..... Location
فرا ابتکاري ..... Metaheuristic
همسایگی ..... Neighbourhood
شبکه ..... Net
تحقیق در عملیات ..... Operation Research
بهینه سازي ..... Optimization
پارامتر ..... Parameter
مسیر ..... Path
پی – مرکز ..... P - Center
پی – میانه ..... P - Median

شبه کد ..... Psedo - Cod
تصادفی ..... Random
رتبه ..... Rank
آر سی ال ..... RCL
مسیر یابی ..... Routing
نمونه ..... Sample
مجموعه پوششی ..... Set Covering
اس ام آ ..... SMA
جستجوي تابو ..... Tabu Search
تابع هدف ..... Target Function
فروشنده دوره گرد ..... Travelling Salesman
رأس ..... Vertex

Aabstract

GRASP is a multi-start metaheuristic for combinatorial problems, in which each iteration consists basically of two phases: construction and local search.
Builds a feasible solution, whose neighborhood is investigated until a local minimum is found during the local search phase. The best overall solution is kept as the result. In this thesis, we first describe the basic components of GRASP. Successful implementation techniques and parameter tuning strategies are discussed and illustrated by numerical results obtained for different appli cations. Enhanced or alternative solution construction mechanisms and techniques to speed up the search are also described: Reactive GRASP, cost perturbations, bias functions, memory and learning, local search on partially constructed solutions, hashing, and filtering. We also discus in detail implementation strategies of memory-based intensification and post-optimization tech niques using path-relinking. Hybridizations with other metaheuristics, parallelization strategies, and applications are also reviewed.

keywords: Algorithm Metaheuristic, Greedy Search, Local search, Grasp, Path relinking.

University of Shahrood

Faculty Of Mathematical Sciences

Solving Of Optimization Problem With Grasp Method

Seyed Ehsan Torabi

Supervisor

Dr.Jafar Fathali

July 2015