دانشگاه صنعتی امیر کبیر

(پلی تکنیک تهران)

دانشکده ریاضی و علوم کامپیوتر

پروژه کارشناسی ارشد

ریاضی کاربردی (شبکه و الگوریتم)

عنوان

طراحی بهینه زیرساخت شبکه *UMTS*

نگارش

افسانه کریمی

استاد راهنما

دکتر سید مهدی تشکری هاشمی

استاد مشاور

دکتر حسن طاهری

بهمن ۱۳۸۷

دانشگاه صنعتی امیرکبیر
(پلی‌تکنیک تهران)

بسمه تعالی

فرم اطلاعات پایان‌نامه
کارشناسی- ارشد و دکترا

تاریخ:
شماره:

معاونت پژوهشی
فرم پروژه تحصیلات تکمیلی ۷

مشخصات دانشجو:

نام و نام خانوادگی: افسانه کریمی | دانشجوی آزاد ● | بورسیه ○ | معادل ○

شماره دانشجوئی: ۸۵۱۱۳۰۳۹ | **دانشکده:** ریاضی و علوم کامپیوتر | **رشته تحصیلی:** ریاضی | **گروه:** کاربردی

مشخصات استاد راهنما:

نام و نام خانوادگی: دکتر سید مهدی تشکری هاشمی | **درجه و رتبه:** دانشیار

مشخصات استاد مشاور:

نام و نام خانوادگی: دکتر حسن طاهری | **درجه و رتبه:** استادیار

عنوان پایان‌نامه به فارسی : طراحی بهینه زیرساخت شبکه UMTS

عنوان پایاننامه به انگلیسی : Cost-Optimal Planning of UMTS Network Infrastructure

نوع پروژه: کارشناسی ○ | ارشد ● | دکترا ○ | سال تحصیلی: ۸۷-۸۶
کاربردی ○ | بنیادی ○ | توسعه‌ای ● | نظری ○

تاریخ شروع : بهمن ۸۶ | تاریخ خاتمه : بهمن ۸۷ | تعداد واحد : ۶ | سازمان تأمین کننده اعتبار :

واژه‌های کلیدی به فارسی: سیستم جامع ارتباط تلفنی متحرک، طراحی شبکه، روش کلی، مدل برنامه‌ریزی ریاضی، الگوریتم ابتکاری جستجوی محلی، فراابتکاری

واژه‌های کلیدی به انگلیسی : universal mobile telecommunication system (UNTS), network planning, global approach, mathematical programming model, local search heuristic, meta heuristic

مشخصات ظاهری	تعداد صفحات ۹۰	تصویر ●	جدول ●	نمودار ●	نقشه ○	واژه‌نامه ○	تعداد مراجع ۲۳	تعداد صفحات ضمائم
زبان متن	فارسی ●	انگلیسی ○		چکیده	فارسی ●	انگلیسی ●		
یادداشت								

نظرها و پیشنهادها به منظور بهبود فعالیت‌های پژوهشی دانشگاه

استاد:

دانشجو:

امضاء استاد راهنما: **تاریخ:**

۱ : ارائه به معاونت پژوهشی به همراه یک نسخه الکترونیکی از پایان‌نامه و فرم اطلاعات پایان‌نامه بصورت PDF همراه چاپ چکیده (فارسی انگلیسی) و فرم اطلاعات پایان‌نامه
۲ : ارائه به کتابخانه دانشکده (شامل دو جلد پایان نامه به همراه نسخه الکترونیکی فرم در لوح فشرده طبق نمونه اعلام شده در صفحه خانگی کتابخانه مرکزی)

چکیده

شبکه *UMTS*[1] از نسل سوم سیستم‌های مخابراتی متحرک است و قادر به ارائه پهنای باند بیشتر نسبت به نسل‌های قبلی و عرضه چندین رده از سرویس‌های مخابراتی از جمله صوت و خدمات مربوط به داده است. از جمله دستاوردهای آن می‌توان به ارتباطات تصویری، امکان استفاده از اینترنت پرسرعت (ایجاد ارتباط با حداکثر سرعت ۳۸٤ کیلو بیت بر ثانیه) و ویدئو کنفرانس اشاره کرد.

استفاده از الگوریتم‌ها برای طراحی زیرساخت شبکه‌ها در جهت کاهش هزینه شبکه امری ضروری است، در این پایان‌نامه به مسئله طراحی زیرساخت شبکه *UMTS* با در نظر گرفتن سه زیرمسئله طراحی سلول، طراحی شبکه دسترسی و طراحی شبکه مرکزی بطور همزمان می‌پردازیم.

دو الگوریتم ابتکاری برای حل این مساله معرفی خواهند شد، یک الگوریتم جستجوی محلی که یک جواب شدنی خوب برای مسئله می‌دهد و یک الگوریتم ابتکاری جستجوی ممنوع که با استفاده از جواب الگوریتم قبل به عنوان جواب ابتدائی جواب بسیار نزدیکی به جواب بهینه بدست می‌آورد، بطوریکه جواب حاصل از این الگوریتم به طور متوسط در فاصله ٥٦/. درصدی از جواب بهینه (یعنی اختلاف جواب حاصل با جواب بهینه نسبت به جواب بهینه برابر با ۰/۰۰٥٦ است) و در بدترین حالت در فاصله ٦/۰٥ درصدی از جواب بهینه می‌باشد.

[1] *Universal Mobile Telecommunications System*

مقدمه

در سال‌های اخیر اهمیت *UMTS* به عنوان یک سیستم مخابراتی موبایل به میزان قابل توجهی افزایش یافته است. از مزیت‌های مهم و شاخص این سیستم نسبت به سیستم‌های قبلی می‌توان به انتقال داده‌ها به همراه گستره وسیعی از سرویس‌ها از جمله ارتباطات تصویری، اینترنت پرسرعت و . . . اشاره کرد. از طرفی توجه به نیازهای اطلاعاتی و مخابراتی جدید لزوم ایجاد تغییرات در قسمت‌های مختلف شبکه مخابرات را ایجاب می‌کند، بطوریکه مجریان طراحی و ساخت شبکه بخش عظیمی از بودجه خود را روی طراحی زیرساخت شبکه‌ها سرمایه‌گذاری می‌کنند.

در این پایان‌نامه به الگوریتم‌های طراحی زیرساخت شبکه *UMTS* می‌پردازیم که وظیفه این الگوریتم‌ها تعیین تعداد و مکان بهینه راس‌های این شبکه یعنی *RBS*[۲] ها، *RNC*[۳] ها، *MSC*[۴] ها و *SGSN*[۵] ها و همینطور تعیین نحوه اتصال این راس‌ها با توجه به محدودیت‌های موجود در طراحی شبکه، با کمترین هزینه ممکن است. در غالب مقاله‌های ارائه شده برای حل مسئله طراحی زیرساخت شبکه *UMTS* به یکی از زیرمسائل این مسئله پرداخته شده است و در این پایان‌نامه کل سه زیرمسئله به طور همزمان بررسی خواهند شد.

فصل اول به ارائه مفاهیم و تعاریف مقدماتی اختصاص داده شده است. در فصل دوم ساختار شبکه *UMTS* و مسئله طراحی زیرساخت آن و همینطور زیرمسائل مرتبط به این مسئله، مدل ریاضی این زیرمسائل و مقاله‌های ارائه شده برای حل این زیرمسائل بیان می‌شوند. در فصل سوم مدل ریاضی کل مسئله بیان می‌شود که آن را *مدل کلی* مسئله می‌گویند و با کمک الگوریتم دقیق شاخه‌وکران در یک مثال دو مدل کلی و *مدل پی‌درپی* که درآن مسئله به زیرمسائل تقسیم می‌شود و زیرمسائل در ترتیبی خاص حل می‌شوند تا جواب بهینه بدست آید، مقایسه می‌شوند.

---

[۲] *Radio Base Station*

[۳] *Radio Network Controller*

[۴] *Mobile Switching Center*

[۵] *Serving GPRS Support Node*

فصل چهارم به بررسی دو الگوریتم ابتکاری برای حل مدل کلی مسئله طراحی بهینه زیرساخت شبکه *UMTS* می‌پردازد، ابتدا الگوریتم ابتکاری جستجوی محلی معرفی می‌شود که جواب آن به عنوان جواب ابتدائی الگوریتم ابتکاری بعدی که یک الگوریتم جستجوی ممنوع است، استفاده می‌شود. کارایی هر دو الگوریتم ابتکاری با کمک الگوریتم دقیق شاخه‌و کران ارزیابی خواهد شد.

فصل ۱

تعاریف و مقدمات

۱-۱ مقدمه

این فصل به بیان مفاهیم و تعاریف از نظریه گراف و علم بهینه‌سازی اختصاص یافته‌است و به خواننده این امکان را می‌دهد که بحث‌های موجود در فصل‌های آینده را بدون ابهام دنبال کند. اگرچه مفاهیم این فصل را می‌توان در اغلب کتب مقدماتی بهینه‌سازی یافت، اما این فصل سعی در مرتفع ساختن چنین نیازی دارد.

۱-۲ نظریه گراف و شبکه

در این بخش چندین تعریف و نماد پایه‌ای از نظریه گراف و سپس مسائلی از نظریه گراف که در فصل‌های بعد استفاده می‌شوند، بیان می‌شود. تعاریف ارائه شده برگرفته از مراجع [۳, ٤, ۷] می‌باشند، مگر آنکه مشخصا مرجع دیگری ذکر شود.

تعریف: *گراف* G یک سه‌تایی مرتب $(V(G), E(G), \psi_G)$ متشکل از مجموعه ناتهی $V(G)$ شامل راسها، $E(G)$ شامل یالهای مجزا از $V(G)$ و تابع وقوع ψ_G که با هر یال G یک جفت نامرتب و نه لزوما مجزا، از راسهای G را همراه می‌کند، است. اگر e یک یال و u و v راسهایی باشند به قسمی که $\psi_G(e) = uv$ آنگاه می‌گوئیم e، u را به v وصل می‌کند.

اگر در گراف G به هر یال یک زوج مرتب از راس‌های آن نسبت داده شود به آن *گراف جهتدار* می‌گویند.

تعریف: گراف جهتداری که به یالهایش (راسهایش) مقادیر عددی نسبت داده شده است و این مقادیر می‌تواند مربوط به هزینه و ظرفیت (عرضه و تقاضا) آنها باشد را *شبکه* می‌نامند.

تعریف: برای یک راس v از گراف G *همسایگی* آن به صورت زیر تعریف می‌شود:

$$N(v) = \{ u \in V(G) \mid vu \in E(G) \}$$

تعریف: دو راس u و v را *مجاور* گویند اگر e یالی از گراف G موجود باشد بطوریکه

$$\psi_G(e) = uv$$

تعریف: درجه راس v برابر است با تعداد راسهای که در مجاورت v هستند یعنی:

$$\deg(v) = | N(v) |$$

تعریف: گراف G' را یک *زیرگراف* از گراف G نامند اگر

$$V(G') \subseteq V(G) \quad , \quad E(G') \subseteq E(G)$$

گراف G' را یک *زیرگراف فراگیر* از گراف G نامند اگر

$$V(G') = V(G) \quad , \quad E(G') \subseteq E(G)$$

تعریف: گراف G را در نظر بگیرید، یک گشت در G شامل دنباله‌ای از راسها و یالهای آن به صورت $i_1 - a_1 - i_2 - \ldots - i_{r-1} - a_{r-1} - i_r$ است که برای $K = 1, 2, 3, \ldots, r-1$ داریم

$$a_k = (i_k,\ i_{k+1}) \in E(G) \quad \text{یا} \quad a_k = (i_{k+1},\ i_k) \in E(G)$$

گشت جهتدار همانند یک گشت تعریف می‌شود با این تفاوت که برای هر دو راس متوالی i_k و i_{k+1} از آن گشت $a_k = (i_k,\ i_{k+1})$ باشد.

تعریف: یک گشت (گشت جهتدار) که تمامی راسهای آن مجزا باشد یک مسیر (*مسیر جهتدار*) نامیده می‌شود.

تعریف: یک مسیر $i_1 - a_1 - i_2 - \ldots - i_{r-1} - a_{r-1} - i_r$ به همراه یال $(i_1,\ i_r)$ و یا یال $(i_r,\ i_1)$ را یک *دور* نامند.

یک مسیر جهتدار $i_1 - a_1 - i_2 - \ldots - i_{r-1} - a_{r-1} - i_r$ به همراه یال $(i_r,\ i_1)$ را یک *دور جهتدار* نامند.

تعریف: دو راس u و v در گراف G *مرتبط*اند اگر لااقل یک مسیر از راس u به راس v در G موجود باشد.

گراف G را *همبند* نامند اگر هر دو راس $u, v \in V(G)$ مرتبط باشند.

تعریف: یک گراف همبند را که شامل هیچ دوری نباشد *درخت* نامند.

یک گراف را که شامل هیچ دوری نباشد، *جنگل* گویند.

تعریف: درخت T را یک *درخت فراگیر از گراف* G نامند اگر یک زیرگراف فراگیر از گراف G باشد.

تعریف: یک درخت با یک راس مشخص با عنوان ریشه را *درخت ریشه‌دار* می‌گوئیم و می‌توان آن را به صورتی در نظر گرفت مثل اینکه از ریشه‌اش آویزان شده است.

تعریف: گراف G را *دوبخشی* نامند اگر بتوان $V(G)$ مجموعه راسها را به زیرمجموعه‌های غیر تهی N_1 و N_2 افراز نمود بطوریکه هر یال از گراف یک راس از N_1 را به یک راس از N_2 وصل کند.

توجه: گاهی G را به طور مختصر و به صورت $G = (V, E)$ نشان می‌دهند.

اکنون می‌توان مسائلی از نظریه گراف را بیان کرد:

- **مسئله کوتاهترین مسیر:** فرض کنید هر یال $(i\ ,\ j)$ در شبکه G هزینه‌ای برابر با c_{ij} داشته باشد که این می‌تواند معرف هزینه، میزان مسافت و یا زمان طی بین دو راس i و j باشد، هدف مسئله کوتاهترین مسیر یافتن یک مسیر از راس مشخص شده منبع s تا راس تعیین شده مقصد t است به نحوی که مجموع هزینه یالهای روی این مسیر کمترین هزینه باشد.

- **مسئله جریان بیشینه:** فرض کنید جریان قابل عبور از هر یال $(i\ ,\ j)$ محدود به مقدار u_{ij} باشد هدف مسئله ماکزیمم جریان، ارسال حداکثر مقدار جریان از راس مشخص شده منبع به سمت راس تعیین شده مقصد با توجه به محدودیت یالها می‌باشد.

- **مسئله درخت فراگیر با کمترین هزینه:** با فرض هزینه c_{ij} برای هر یال (i, j) در شبکه G هدف مسئله درخت فراگیر با کمترین هزینه یافتن یک درخت فراگیر از شبکه است به نحوی که مجموع هزینه یالهای درخت فراگیر کمترین مقدار باشد.

- **مسئله تطبیق**: $M \subseteq E(G)$ یک *تطبیق* در گراف G نامیده می‌شود اگر هیچ دو یالی در M مجاور نباشند. مسئله تطبیق به دنبال یافتن یک تطبیق است بطوریکه معیارهای معینی بهینه گردد. به عنوان نمونه می‌توان از حداکثر نمودن تعداد یالها، بیشینه یا کمینه کردن مجموع وزن یالها نام برد.

- **مسئله تطبیق دوبخشی**: مسئله تطبیق بر روی گراف‌های دوبخشی، مسئله تطبیق دوبخشی نامیده می شود.

- **مسئله فروشنده دوره‌گرد:** یک شبکه را در نظر بگیرید، مسئله تعیین گشتی است که مجموع وزن یالهای آن کمینه شود.

۱-۳ مسائل بهینه‌سازی ترکیباتی

یک مسئله بهینه‌سازی ترکیباتی مثل (S, f) که در آن S مجموعه متناهی از تمامی جوابهای شدنی مسئله و $f : S \to \mathbf{R}$ که به هر $i \in S$ یک هزینه مشخص را نسبت می‌دهد، در نظر بگیرید در حالت کمینه‌سازی، مسئله به دنبال یافتن $i_{opt} \in S$ می‌باشد بطوریکه:

$$f(i_{opt}) \le f(i) \qquad \forall i \in S$$

برای حل اینگونه مسائل لازم است الگوریتم‌هایی طراحی شوند که با سرعتی بیش از تکنیک‌های شمارشی ساده عمل کنند.

۱-۳-۱ دیدگاه‌های کلی در حل مسائل بهینه‌سازی ترکیباتی

دیدگاه‌های کلی در حل مسائل بهینه‌سازی ترکیباتی را می‌توان به سه دسته کلی تقسیم نمود:

۱. الگوریتم‌های دقیق
۲. الگوریتم‌های تقریبی
۳. الگوریتم‌های ابتکاری

الگوریتم‌های دقیق یک جستجوی ساخت یافته برای رسیدن به جواب بهینه انجام می‌دهند یعنی در این روشها حل دقیق مسئله و یافتن جواب بهینه مدنظر می‌باشد. در مقابل الگوریتم‌های تقریبی لزوما جهت یافتن جواب بهینه تلاش نمی‌کنند بلکه در این روشها هدف دست یافتن به جوابهای نزدیک به جواب بهینه می‌باشد، در اغلب موارد این روش قابل قبول می‌باشد زیرا می‌توانیم با یک الگوریتم نسبتا سریع یک مساله را بطور تقریبی حل کنیم و همینطور می‌توان ثابت کرد جواب بدست آمده تقریبا نزدیک به جواب بهینه می‌باشد. و بالاخره الگوریتم‌های ابتکاری در بیشتر موارد درست کار می‌کنند ولی هیچ تضمین خاصی برای رسیدن به جواب بهینه و یا جوابهای نزدیک به آن و با سرعتی قابل قبول نداریم.

۱-٤ الگوریتم دقیق شاخه و کران

یکی از الگوریتم‌های دقیق الگوریتم شاخه و کران است کارایی یک الگوریتم شاخه و کران در عمل، کاملا وابسته به الگوریتمی است که جهت محاسبه کران پایین مساله مورد استفاده قرار می‌گیرد. دو شرط سریع بودن الگوریتم و نزدیک بودن کران پایین به جواب بهینه تأثیر زیادی در زمان اجرای الگوریتم شاخه و کران دارند.

یکی از روشهای معمول برای این منظور، خفیف‌سازی مساله است. در این روش ابتدا مساله بصورت یک مساله خطی اعداد صحیح مدل می‌شود و سپس با حذف بعضی از قیود سعی در یافتن الگوریتمی کارا برای مساله جدید دارد. یکی از خفیف‌سازی‌های ممکن حذف قید صحیح بودن متغیرهای مساله خطی اعداد صحیح و تبدیل کردن آن به یک مساله خطی می‌باشد. در اینصورت به راحتی می‌توان با استفاده از الگوریتم‌هایی همچون الگوریتم سیمپلکس مساله

خطی را حل نمود. واضح است که جواب بهینه مساله خطی، یک کران پایین برای جواب بهینه مساله اصلی می‌باشد.

در این بخش بطور مختصر به بررسی چگونگی یافتن جواب بهینه توسط روش شاخه و کران به کمک مساله خفیف سازی می‌پردازیم. مدل برنامه‌ریزی خطی اعداد صحیح زیر را در نظر بگیرید:

$$Minimize\ cx$$
$$s.t.$$
$$x \in F$$

که در آن $x=(x_1, x_2, \ldots, x_n)$، F مجموعه تمامی جوابهای شدنی مساله و در واقع جوابهایی هستند که در سیستم زیر به عنوان قیدهای مساله صدق می‌کنند:

$$Ax = b$$
$$x_j = \{0,1\} \quad \forall \ \ j = 1, 2, 3, \ldots, n$$

واضح است که هر n تایی مرتب از عناصر صفر و یک که در معادله $Ax=b$ صدق کند به عنوان یک جواب شدنی برای مساله می‌باشد. یک راه حل بدیهی برای حل این مساله بررسی تمامی جوابهای شدنی مساله و یافتن جواب بهینه، با کمترین مقدار cx است. در اینصورت حل مساله نمایی می‌شود برای مثال در یک مساله با ۱۰۰ متغیر تصمیم گیری، 2^{100} جواب وجود دارد که با این فرض که بررسی هر جواب 10^{-9} ثانیه طول بکشد، زمان اجرای کل برنامه یک میلیون میلیون سال طول می‌کشد.

روش شاخه و کران سعی در کاهش مقدار جستجو در فضای جوابها به یک مقدار منطقی دارد که برای این منظور به صورت زیر عمل می‌نماید:

فرض کنید F_1 با اضافه کردن محدودیت $x_1 = 0$ و F_2 با اضافه کردن محدودیت $x_1 = 1$ به مساله از F بدست آیند. واضح است که $F = F_1 \cup F_2$ و همچنین یک جواب بهینه بر روی مجموعه F، یک جواب بهینه بر روی هر یک مجموعه‌های F_1 و F_2 نیز می‌باشد هرچند ممکن است شدنی نباشد. فرض کنید که جواب بهینه $\bar{x}$ بر روی مجموعه F_2 با مقدار $Z(\bar{x}) = 100$ معین باشد. به منظور جلوگیری کردن از هزینه سرسام آور حل مساله بر روی مجموعه F_1 به اینصورت عمل می‌کنیم که بجای حل مساله اصلی برروی مجموعه F_1، یک صورت ساده شده مساله را با حذف بعضی از محدودیتهای مساله بر روی این مجموعه حل می‌کنیم. حال فرض

کنید که x' مقدار بهینه مساله خفیف سازی شده با مقدار $Z(x')$ بر روی مجموعه F_1 باشد (می‌دانیم که $Z(\overline{x})$ یک کران پایین برای جواب بهینه مساله اصلی برروی مجموعه F_1 می‌باشد)، در اینصورت چهار حالت ممکن است رخ دهد:

۱. جواب x' موجود نباشد به این معنا که مساله در حالت خفیف شده دارای جواب شدنی نباشد

۲. جواب بهینه x' همچنان یکی از اعضای مجموعه F_1 باشد (حتی با توجه به حذف بعضی از قیدهای مساله)

۳. جواب بهینه x' در مجموعه F_1 نباشد و مقدار تابع هدف آن در رابطه زیر صدق کند

$$Z(x') \geq Z(\overline{x}) = 100$$

٤. جواب بهینه x' در مجموعه F_1 نباشد و مقدار تابع هدف آن در رابطه زیر صدق کند

$$Z(x') < Z(\overline{x}) = 100$$

در هر یک از این چهار حالت به طریقه زیر عمل می نماییم.

در حالت (۱) از آنجائیکه مساله خفیف شده دارای جواب شدنی نمی‌باشد، واضح است که در حالت کلی مساله نیز مجموعه F_1 خالی می‌باشد، در اینصورت جواب x' جواب کلی مساله بر روی مجموعه F می‌باشد. در حالت (۲) از آنجائیکه جواب بهینه مساله خفیف شده یکی از اعضای مجموعه F_1 می‌باشد پس این جواب همچنان برای مساله اصلی شدنی می‌باشد، واضح است که این جواب، یک جواب بهینه برای مساله اصلی بر روی مجموعه F_1 نیز هست. حال هرکدام از جوابهای $\overline{x}$ و x' که دارای مقدار تابع هدف کمتری باشند، جواب بهینه مساله اصلی بر روی مجموعه F می‌باشد. در حالت (۳) از آنجائیکه مقدار تابع هدف $Z(x')$ یک کران پایین برای مقدار بهین مساله اصلی برروی F_1 می‌باشد و با توجه به اینکه $Z(x') \geq Z(\overline{x})$ واضح است که $\overline{x}$ یک مقدار بهین برای مساله اصلی برروی مجموعه F می‌باشد.

اما در مورد حالت (٤)، در این حالت هنوز جواب بهینه مساله بر روی مجموعه F بدست نیامده است. برای این منظور مجموعه F_1 به دو مجموعه F_3 و F_4 افراز می‌گردد به اینصورت که مجموعه F_3 با اضافه کردن محدودیتهای $x_1 = 0$ و $x_2 = 0$ و F_4 با اضافه کردن

محدودیتهای $x_1 = 0$ و $x_2 = 1$ به مساله اصلی از F بدست می‌آیند. حال می‌توان با خفیف‌سازی مساله روی دو مجموعه F_3 و F_4 با بکارگیری ایده بالا سعی در یافتن جواب بهینه را ادامه داد.

در حالت کلی مساله شاخه و کران بصورت سیستماتیک ناحیه جوابهای شدنی F را به زیرناحیه‌های F_1 و F_2 و F_3 و . . . و F_k افراز می‌نماید و با استفاده از بهترین جواب شدنی $\overline{x}$ که تا این مرحله بدست آمده سعی در یافتن جواب بهین مساله (با توجه به حالتهای گفته شده) دارد. اگر رابطه $c\overline{x} > cx^k$ برای جواب بهین یافته شده برای مساله خفیف شده برروی ناحیه F_k برقرار باشد، آنگاه این ناحیه به چند زیرناحیه دیگر تقسیم می‌شود و این کار تا جایی ادامه می‌یابد که یکی از سه حالت اول تا سوم برای تمامی زیرناحیه‌های ایجاد شده برقرار شود که در اینصورت جواب بهینه مساله اصلی برروی ناحیه جوابهای شدنی F بدست آمده است.

در عمل برای بکارگیری مساله شاخه و کران ما نیاز به یافتن روندی مناسب جهت تقسیم ناحیه جواب در هر مرحله جهت پیدا کردن جواب شدنی خوب و در نهایت روشی مناسب جهت خفیف‌سازی مساله داریم. بکارگیری روشی مناسب جهت خفیف‌سازی مساله به منظور یافتن کران پایین برای مساله اصلی نقش بسیار مهمی در کاهش زمان اجرای الگوریتم شاخه وکران دارد. عملاً یک خفیف‌سازی ضعیف باعث می‌شود که حالتهای ۲ و ۳ در زمان اجرای الگوریتم شاخه و کران بندرت بوقوع پیوندند که این باعث گسترده شدن فضای مورد جستجو در الگوریتم شاخه و کران می‌شود. به عبارت دیگر یک خفیف‌سازی مناسب باعث کاهش فضای مورد جستجو برای یافتن جواب بهین گردد.

۱-۵ الگوریتم جستجوی محلی

با این فرض که (S, f) یک مسئله بهینه‌سازی ترکیباتی باشد ، $N : S \to 2^S$ را یک ساختار همسایگی نامند اگر این نگاشت برای هر جواب شدنی $i \in S$ یک مجموعه $S_i \subset S$ از جوابهای شدنی را تحت عنوان مجموعه همسایگی جواب شدنی i معرفی کند. هر عضو $j \in S_i$ همسایه i، نامیده می‌شود و داریم:

$$j \in S_i \quad \Leftrightarrow \quad i \in S_j$$

با این فرض که (S, f) یک مسئله بهینه‌سازی ترکیباتی و N یک ساختار همسایگی باشد، روش انتخاب یک همسایه j از مجموعه همسایگی i را مکانیزم تولید نامند. مکانیزم تولید می‌تواند کاملا حریصانه و یا کاملا تصادفی و یا ترکیبی از این دو در نظر گرفته شود.

فرض کنید در یک مسئله بهینه‌سازی مکانیزم تولید کاملا حریصانه باشد، الگوریتم جستجوی محلی با شروع از یک جواب که معمولا بصورت تصادفی انتخاب می‌شود و با جستجو در مجموعه همسایگی جواب جاری در هر مرحله سعی دارد جوابی بهتر با توجه به هزینه جوابها بیابد و آن را با جواب جاری جایگزین سازد. الگوریتم زمانی به پایان میرسد که قادر به یافتن هیچگونه جواب بهبوددهنده‌ای با توجه به جواب جاری و نیز مجموعه همسایگی آن نباشد.

۱-٦ معرفی تعدادی از الگوریتم‌های ابتکاری

الگوریتم سردشدن شبیه‌سازی شده: این روش در دهه ۱۹۸۰ میلادی توسط کِرک‌پاتریک[٦] و همکارانش از یک سو و کرنی[۷] از سوی دیگر در جهت بهینه‌سازی ترکیباتی بنا شد. آثار مولفین را به ترتیب می‌توان در مراجع [۱۷]، [۱۸] و [۵] یافت. این روش که بر اساس شبیه‌سازی فرآیند سردشدن تدریجی یک ماده فلز در جهت نیل به حالت پایدار، برای حل مسائل بهینه-سازی ترکیباتی بنا شده بود بعدها در زمره‌ای بهترین روش‌های جستجوی محلی قرار گرفت.

الگوریتم کلنی مورچه: در اوایل دهه ۱۹۹۰ دوریگو[۸] توانست با الهام گرفتن از رفتار مورچه‌های واقعی در هنگام جستجوی غذا، این الگوریتم را برای حل مساله فروشنده دوره‌گرد ارائه نماید [۸]. رفتار مورچه‌های واقعی نشان می‌دهد که با وجود اینکه آنها حشرات تقریبا کوری هستند، اگر یک مانع بین مسیر لانه و منبع غذایی مورچه‌ها قرار گیرد بطوریکه طول مانع از طرفین مساوی نباشد، آنها بعد از مدتی کوتاهترین مسیر را پیموده و در یک مسیر قرار می‌گیرند. علت این امر ماده شیمیایی تبخیرشونده‌ای بنام فرمون است که مورچه‌ها هنگام حرکت مقداری از آن را در مسیر می‌ریزند.

---

[٦] *Kirkpatrick*

[۷] *Cerny*

[۸] *Dorigo*

الگوریتم جستجوی ممنوع: ریشه این روش در اواخر دهه ۱۹۶۰ و اوایل دهه ۱۹۷۰ بوجود آمده است. اولین بار توسط گلاور[۹] در سال ۱۹۸٦ معرفی شد. جستجوی ممنوع یک روش اصلاحی است که از یک جواب ابتدایی شدنی s آغاز می‌شود و سپس در زیرمجموعه همسایگی‌های s که مجموعه همسایگی‌ها منهای مجموعه ممنوع است، حرکت کرده و بهترین جواب s' را در آن جستجو می‌کند تا اینکه فرآیند با رسیدن به یک معیار پایانی متوقف شود. مجموعه ممنوع به منظور ایجاد یک مکانیزیم پادحلقه ایجاد و بکار گرفته می‌شود زیرا اگر $f(s)$ نشان دهنده هزینه s باشد آنگاه $f(s')$ ممکن است بزرگتر از $f(s)$ شود. بطور اخص، اگر $B(s)$ مجموعه ویژگی‌های جواب s و $E(s)$ نشان‌دهنده همه جواب‌هایی باشد که دارای یک ویژگی (یک مولفه) از جواب s باشد آنگاه برای θ تکرار، هر جواب s' که دارای ویژگی در $B(s)$ باشد، ممنوع می‌شود مگر اینکه $f(s')$ کمتر از هزینه بهترین جواب شناخته‌شده‌ای باشد که در $E(s)$ با آن برخورد کرده‌ایم. این قانون آخر به عنوان یک معیار رضایتمندی شناخته می‌شود.

۱-۷ پیچیدگی محاسباتی

وقتی که از الگوریتم‌های مختلف برای حل یک مسئله استفاده می‌کنیم باید معیاری وجود داشته باشد که براساس آن:

۱. به بررسی دشواری حل مسائل به وسیله کامپیوتر و به صورت الگوریتمی بپردازیم
۲. نتیجه بگیریم کدام الگوریتم از الگوریتم دیگر بهتر است

این معیار، نظریه پیچیدگی محاسباتی نام دارد. این نظریه بخشی از نظریه محاسباتی است که با منابع مورد نیاز برای حل یک مسئله سروکار دارد. عمومی‌ترین منابع عبارت از زمان لازم برای حل کردن مسئله، فضای مصرفی و حافظه موردنیاز می‌باشند، یعنی الگوریتمی بهتر است که فضا و زمان کمتری را مصرف کند. با وجود اینکه معمولا بازدهی الگوریتم‌ها بر حسب زمان تحلیل می‌شوند، اما سایر منابع مانند تعداد پردازنده‌های موازی البته در حالت پردازش موازی نیز باید در نظر گرفته شوند.

---

[۹] *Glover*

باید به این نکته توجه داشت که نظریه پیچیدگی با نظریه قابل حل بودن متفاوت می‌باشد. این نظریه در مورد قابل حل بودن یک مسئله بدون توجه به منابع موردنیاز آن بحث می‌کند. همچنین این نظریه بیان می‌کند که کدام مسائل قابل‌حل و کدام مسائل غیرقابل‌حل می‌باشند.

تعریف: *عمل اصلی* در یک الگوریتم عبارت از دستور یا گروهی از دستورها است بطوریکه کل کار انجام شده توسط الگوریتم متناسب با تعداد دفعاتی باشد که توسط این دستور یا گروهی از دستورها انجام می‌شود. در واقع هیچ قاعده صریحی برای انتخاب عمل اصلی وجود ندارد و این کار با تجربه و داوری درست صورت می‌پذیرد بنابراین اگر اندازه مسئله افزایش یابد و زمان اجرای الگوریتمی کندتر از الگوریتمی دیگر رشد کند آنگاه کاراتر از آن نیز می‌باشد.

در این تحلیل نمی‌خواهیم هر یک از دستورات اجرا شده را شمارش کنیم زیرا تعداد دستورها به نوع زبان برنامه‌نویسی و نحوه نوشتن برنامه بستگی دارد. در عوض به معیاری نیاز داریم که مستقل از کامپیوتر و زبان برنامه‌نویسی باشد. بطور کلی زمان اجرای یک الگوریتم با افزایش اندازه مسئله (تعداد ورودی‌های مسئله) زیاد می‌شود و زمان اجرا با تعداد دفعاتی که عملیات اصلی انجام می‌شود تناسب دارد. پس تحلیل پیچیدگی زمانی یک الگوریتم عبارت است از تعداد دفعاتی که عمل اصلی به ازای هر مقدار از اندازه مسئله انجام می‌شود.

فرض کنید $f: \mathbf{Z}^+ \to \mathbf{R}^+$ و $g = \mathbf{Z}^+ \to \mathbf{R}^+$ توابعی تعریف شده روی مجموعه اعداد صحیح مثبت باشند:

- اگر ثابت حقیقی و مثبت c و عدد صحیح نامنفی n_0 وجود داشته باشد بطوریکه:

$$\forall n > n_0 \quad f(n) \le cg(n)$$

می‌نویسیم $f(n) = O(g(n))$ یا می‌گوییم $f(n)$ ، $O(g(n))$ است، یعنی f سریعتر از g رشد نمی‌کند.

- اگر برای هر ثابت حقیقی و مثبت c، عدد صحیح نامنفی n_0 وجود داشته باشد بطوریکه:

$$\forall n > n_0 \quad f(n) \le cg(n)$$

می‌نویسیم $f(n) = o(g(n))$.

- اگر ثابت حقیقی و مثبت c و عدد صحیح نامنفی n_0 وجود داشته باشد بطوریکه:

$$\forall n > n_0 \quad f(n) \geq cg(n)$$

می‌نویسیم $f(n) = \Omega(g(n))$.

- اگر ثابت‌های حقیقی و مثبت c و c' و عدد صحیح نامنفی n_0 وجود داشته باشد بطوریکه:

$$\forall n > n_0 \quad cg(n) \leq f(n) \leq c'g(n)$$

می‌نویسیم $f(n) = \Theta(g(n))$.

تعریف‌های ذکر شده در بالا نشان می‌دهند در صورتی *f(n)*، $\Theta(g(n))$ است که *o(g(n))* و *O(g(n))* باشد.

۱-۷-۱ رده‌های پیچیدگی

در طبقه‌بندی مسائل، می‌توان آنها را به رده‌هایی تقسیم کرد به طوری که مسائل در یک رده از حیث زمان یا فضای موردنیاز با هم مشابهت دارند. این رده‌ها در اصطلاح رده‌های پیچیدگی خوانده می‌شوند. معروفترین کلاس‌های پیچیدگی چندجمله‌ای‌ها و غیر چندجمله‌ای‌ها هستند که مسئله‌ها را از نظر زمان موردنیاز تقسیم‌بندی می‌کنند.

تعریف: مسئله‌هایی با *پیچیدگی‌های چندجمله‌ای*، مسئله‌هایی هستند که الگوریتم‌های سریع برای پیدا کردن جواب آنها وجود دارند.

تعریف: مسئله‌هایی با *پیچیدگی‌های غیرچندجمله‌ای*، مسئله‌هایی هستند که گرچه ممکن است پیدا کردن جواب برای آنها نیاز به زمان زیادی باشد اما بررسی کردن درستی جواب به وسیله الگوریتم سریع ممکن است.

در ادامه به تعریف‌های مقدماتی موردنیاز برای معرفی رده‌های پیچیدگی می‌پردازیم:

تعریف: یک الگوریتم *کارا* الگوریتمی است که پیچیدگی محاسباتی آن به یک تابع چندجمله‌ای از اندازه ورودی‌اش محدود شود، یعنی اگر n اندازه ورودی باشد آنگاه یک چندجمله‌ای *p(n)* وجود دارد بطوریکه پیچیدگی محاسباتی الگوریتم *O(p(n))* باشد.

رده *P*: مجموعه‌ای از مسائل تصمیم‌گیری (مسائلی که پاسخ آنها بله یا خیر است) که می‌توان آنها را توسط الگوریتم‌های کارا حل کرد.

منظور از *نمونه* یک مسئله، یک مجموعه خاص از مقادیر یا کمیت‌هایی است که بطور کامل متغیرهای مساله (ورودی‌های مساله) را توصیف می‌کند.

یک الگوریتم *نامعین* برای مسائل تصمیم‌گیری مثل مساله A دارای دو مرحله است:

۱- مرحله حدس: یک پاسخ ممکن برای یک نمونه I از مسئله A است که آن را با C (I) نمایش می‌دهند

۲- مرحله تصدیق: با استفاده از پاسخ ممکن C (I) و ورودی I بطور موثر بررسی می‌کند که آیا C (I) می‌تواند تعیین کند که I یک نمونه مثبت است (یعنی خروجی مساله بازای آن بلی است) یا خیر

یک الگوریتم *نامعین کارا*، الگوریتمی نامعین است که مرحله تصدیق آن یک الگوریتم کارا باشد.

رده *NP* : مجموعه‌ای از مسائل تصمیم‌گیری است که می‌توانند توسط الگوریتم‌های نامعین کارا حل شوند.

۱-۷-۲ معرفی *NP*-سخت

فرض کنید می‌خواهیم مساله A را حل کنیم و الگوریتمی دتریم که مسئله B را حل می‌کند. همچنین فرض کنید می‌خواهیم الگوریتمی بنویسیم که نمونه y از مسئله B را از نمونه x از مسئله A طوری بسازد که که یک الگوریتم برای مسئله B جوابهای "بله" برای y بدهد اگر و فقط اگر جواب مسئله A برای نمونه x "بله" باشد. چنین الگوریتمی الگوریتم تبدیل نامیده می‌شود. در واقع تابعی است که هر نمونه از مسئله A را به نمونه‌ای از مسئله B تصویر می‌کند.

اگر یک الگوریتم کارا از مساله تصمیم‌گیری A به مساله تصمیم‌گیری B وجود داشته باشد، مساله A *کاهش‌پذیر کارا* به مساله B می‌باشد و این را با نماد $A \alpha B$ نمایش می‌دهند.

اگر مسئله تصمیم‌گیری B در $\boldsymbol{P}$ و $A \alpha B$ در این صورت مساله تصمیم‌گیری A نیز در $\boldsymbol{P}$ خواهد بود.

رده *NP*-سخت: مجموعه‌ای از تمامی مسائل تصمیم‌گیری که هر مساله ***NP*** کاهش‌پذیر کارا به هر یک از این مسائل باشد.

۱-۸ نتیجه‌گیری

فصل اول را به بیان مفاهیم و تعاریف اولیه از نظریه گراف و علم بهینه‌سازی پرداختیم تا در راستای ادامه کار پایان‌نامه و حین استفاده از این مبانی، نیاز خواننده به مطالعه کتب مقدماتی بهینه‌سازی را مرتفع سازیم.