مساله پوشش راسی

خاطره قربانی مقدم

۱۷ تیر ۱۳۹۱

فهرست مطالب

لیست تصاویر

لیست جداول

۱.۰ چکیده

یکی از مسایل مهم بهینه سازی ترکیبیاتی مساله ی پوشش راسی می باشد، که در رده ی مسایل ان-پی سخت [۱] است. مساله پوشش راسی را می توان به صورت زیر تعریف کرد :

فرض کنید که یک گراف مانند $G = (V, E)$ داده شده باشد. $C \subseteq V$ را یک پوشش راسی برای گراف G می گوییم، هر گاه برای هر $(u, v) \in E$ حداقل یکی از رئوس u یا v متعلق به C باشد. هدف از حل مساله پوشش راسی پیدا کردن پوشش راسی با اندازه مینیمم می باشد، هدف از این تحقیق به کار گیری الگوریتم های مختلف (حریصانه، تقریبی، فرابتکاری) برای بدست آوردن پوشش راسی مینیمم می باشد.

[۱] NP-hard

فصل ۱

مقدمه

یکی از مسایل مهم بهینه سازی ترکیبیاتی مساله ی پوشش راسی می باشد، که در رده ی مسایل ان-پی سخت [۱] است. مساله پوشش راسی را می توان به صورت زیر تعریف کرد :

فرض کنید که یک گراف مانند $G = (V, E)$ داده شده باشد. $C \subseteq V$ را یک پوشش راسی برای گراف G می گوییم، هر گاه برای هر $(u, v) \in E$ حداقل یکی از رئوس u یا v متعلق به C باشد. هدف از حل مساله پوشش راسی پیدا کردن پوشش راسی با اندازه مینیمم می باشد، در یک نوع از مسایل پوشش راسی که هر راسی دارای یک وزن صحیح می باشد، ما به دنبال کوچکترین زیر مجموعه از رئوس با کمترین وزن می باشیم. [۱۲]

سیر تکاملی حل مساله ی پوشش راسی در زمان های مختلف به شرح زیر می باشد. این مساله اولین با توسط کوک [۲] در سال ۱۹۶۸ بیان شد و در سال ۱۹۷۱ کوک ثابت کرد که این مساله در رده ان-پی سخت است او برای اثبات از مساله ی 3-Sat استفاده کرد و یک نمونه از مساله ی 3-Sat را در یک زمان چند جمله ای به یک نمونه از مساله ی پوشش راسی تبدیل کرد. بعد از آن این مساله هسته اصلی اثبات بسیاری از مسایل ان-پی سخت گردید. [۲]

همانند همه ی مسایل ان-پی سخت الگوریتم های نمایی فراوانی برای حل این مساله وجود دارد ولی تاکنون الگوریتم چند جمله ای قطعی برای حل این مساله ارایه نشده است که بتواند جواب بهینه را بدست آورد. حل تقریبی مساله ی پوشش راسی شامل روشهای ابتکاری و فرا ابتکاری می باشد. الگوریتم های ابتکاری بسیار زیادی برای حل مساله ی پوشش راسی ارایه شده است. یک روش شناخته شده برای حل مساله ی پوشش راسی روشی به نام حذف یالها [۳] است. این روش در سال ۱۹۷۹ توسط گری و جانسون [۴] در [۷] بیان شد. این روش براساس انتخاب راسی با

---

[۱] NP-hard
[۲] Cook
[۳] Edge deletion
[۴] Garey and Johnson

بزرگترین درجه می باشد، این الگوریتم دو تقریب است. اما یکی دیگر از روش های شناخته شده روش جستجوی در عمق می باشد، که توسط کارلا ساویج [۵] در [۳] در سال ۱۹۸۲ ارایه شده است، این روش نیز دو تقریب است ولی نسبت به سایر روش ها از عملکرد خوبی برخوردار نمی باشد و اغلب از این روش استفاده نمی کنند. در سال ۱۹۸۸ روش بیشترین درجه ی حریصانه [۶] توسط پاپادیمتریو و یاناکاکیس [۷] ارایه شد [۴] . پیچیدگی روش آنها تقریبا $\log \Delta$ می باشد. پس از بیان این الگوریتم، الگوریتم دیگری بر گرفته از روش بیشترین درجه ی حریصانه توسط پاپادیمتریو در [۵] می باشد، که در ابتدا یک راس با ماکزیمم درجه را انتخاب می کنیم، تمامی همسایه های آن را در نظر می گیریم و رئوس مجاور با آن را حذف می کنیم. این روش دارای پیچیدگی $\frac{\Delta}{\sqrt{۲}} - ۲$ می باشد و می توان گفت این الگوریتم در جهت بهبود الگوریتم پاپادیمتریو و یاناکاکیس می باشد. یکی دیگر از روش های حل مساله ی پوشش راسی که بهترین نتایج را در بدترین حالت دارد روش پوشش مستقل حریصانه [۸] می باشد که توسط کاراکاستاس [۹] در سال ۲۰۰۵ در [۹] بیان شد. که یک الگوریتم تقریبی با پیچیدگی $\frac{۱}{\sqrt{\log n}} - ۲$ می باشد. (در واقع کار او در جهت بهبود کار باریهودا و ایون [۱۰] می باشد، آنها اثبات کرده اند که پیچیدگی پوشش راسی $\frac{\log \log n}{\sqrt{\log n}} - ۲$ می باشد) .

توجه ۱. *ازجمله مسایلی که با مساله ی پوشش راس مرتبط می باشند عبارتند از:*

مساله کیلیک بیشینه [۱۱]

مساله مجموعه مستقل [۱۲]

مساله پوشش مجموعه [۱۳]

مساله پوشش راسی از جمله مسایلی است که کاربردهای بسیاری در زندگی روزمره دارد، یکی از معروف ترین این کاربردها مانیتورنگ می باشد. به عنوان مثال فرض کنید که می خواهیم راهرو های دانشکده علوم ریاضی دانشگاه فردوسی مشهد را مانیتورینگ کنیم . سوال این است که دوربین ها را چگونه نصب کنیم که با کمترین تعداد دوربین تمامی راهروها زیر نظر باشند؟ در این صورت می توانیم با کمترین تعداد دوربین تمامی دانشکده را زیر نظر بگیریم (

[۵] Carla Savage
[۶] Maximum degree greedy
[۷] Papadimitriou and Yannakakis
[۸] Greedy independent cover
[۹] Karakostas
[۱۰] Bar-Yehuda and Even
[۱۱] Maximum Cliqu Problem
[۱۲] Independent Set
[۱۳] Set Cover

مساله مانیتورینگ برای بیمارستان ها و فروشگاه ها هم مطرح است). حال فرض کنید که می خواهیم در یک منطقه ایستگاه آتش نشانی بزنیم، حال مساله این است آتش نشانی ها را کجا تاسیس کنیم که با کمترین تعداد ایستگاه آتش نشانی تمام منطقه را پوشش دهد؟ این دسته از مسایل و مسایلی از این قبیل را می توانند کاربرد هایی برای مساله ی پوشش راسی باشند. حتی امروزه پوشش راسی در علوم مختلفی مثل طراحی مدار، ارتباطات راه دور، زیست، اقتصاد و هواشناسی کاربرد دارد (برای دیدن جزییات بیشتر به [۱] مراجعه کنید). همان طور که در بالا به آن اشاره شد روش های زیادی برای حل این مساله وجود دارد، در علوم کامپیوتر الگوریتم های تقریبی، الگوریتم هایی برای پیداکردن راه حل های تقریبی برای مسائل بهینه سازی هستند.این الگوریتم ها اغلب برای حل تقریبی مسائل ان پی-سخت کار می‌روند زیرا بسیاری از مسائل بهینه سازی ان پی-سخت هستند (در واقع بررسی کردن درستی جواب اینگونه مسائل با حل کلی آنها معادل است) طبق تئوری پیچیدگی محاسباتی ازمانیکه $P \neq NP$ ، الگوریتم‌های کارامد با زمان چند جمله ای برای چنین مسائلی پیدا نخواهد شد مگر اینکه P=NP که چنین فرضی هم خیلی بعید است . برخلاف روش‌های ابتکاری [۱۴] که راه حلهایی بهینه، اغلب بدون اثبات و بدون کران برای جواب خود هستند؛ الگوریتم‌های تقریبی راه حلهایی شبه بهینه همراه با ضریبی برای میزان تقریب جواب واقعی ارائه می‌دهند همچنین وجود جواب خود را در بازهٔ خطای اعلام شده تضمین می‌کنند .(مثلا جواب آنها ۲ برابر جواب بهینه است) منتها جواب خود را در زمان چندجمله‌ای تولید می‌کنند، الگوریتم‌های تقریبی برای مسائل P نیز استفاده می‌شوند ولی به ازای ورودی‌های بزرگ خوب عمل نمی‌کنند. یکی از مثالهای معروف برای الگوریتم‌های تقریبی، مساله پوشش راسی [۱۵] در گراف است : پیدا کردن یال پوشش داده نشده و اضافه کردن هر دو رأس آن به مجموعه پوشش رأسی تا زمانی که هیچ یال پوشش نیافته نماند . واضح است که مجموعه جوابهای این الگوریتم دو برابر جوابهای بهینه یعنی مجموعه کمترین رأس‌ها برای پوشش دادن همه یالها در یک گراف است؛ پس ضریب ثابت این الگوریتم ۲ است . مساله پوشش راسی جزء مسایل اساسی در گراف است که در رده ی مسایل ان-پی کامل [۱۶] قرار می گیرد ([۲] , [۸]).

در اینجا ما سعی کرده ایم که برخی از الگوریتم های مهم برای حل مساله پوشش راسی را مرور کنیم. در ابتدا به بیان مقدماتی در مورد این مساله می پردازیم و مساله ی پوشش راسی را از دو دیدگاه تصمیم گیری و بهینه سازی مورد مورد بررسی قرار خواهیم داد. در دیگر فصل ها با به کارگیری الگوریتم های مختلف (حریصانه، تقریبی، فرابتکاری) پوشش راسی مینیمم را به دست می آوریم.

---

[۱۴] Heuristics
[۱۵] vertex cover
[۱۶] NP-Complete

فصل ۲

مفاهیم مقدماتی در مورد مساله پوشش راسی

فرض کنید که G=(V, E) یک گراف ساده و بدون جهت باشد.

۱.۲ تعریف پوشش راسی

پوشش C از گراف G یک زیر مجموعه ای از رئوس است به طوریکه به ازای هر یالی مانند $e = uv \in E$ که انتخاب می شود، حداقل یک راس آن در C باشد یعنی $\forall e = uv \in E, u \in C \vee v \in C$. که هدف از پوشش راسی بدون وزن [۱] پیدا کردن یک پوشش راسی با حداقل تعداد راس می باشد و هدف از پوشش راسی وزن دار [۲] پیدا کردن یک پوشش راسی با حداقل وزن کلی می باشد [۵].

مثال ۱. *مجموعه ی تمام راس ها یک پوشش راس است (مثال بدیهی).*

مثال ۲. *درگراف دو افزاره کامل $k(m,n)$ مینیمم پوشش راس $\min(m,n)$ است.*

قضیه ۱.۱.۲. *مساله ی پوشش راسی ان-پی سخت است.*

برهان. این قضیه در سال ۱۹۷۱ توسط کوک [۳] اثبات شد [۲] ، می دانیم که مساله پوشش راسی NP می باشد (زیرا یک الگوریتم غیر قطعی برای این مساله می توانیم به صورت زیر طراحی کنیم:

ابتدا یک تعدادی راس به تصادف انتخاب کنید (مرحله حدس زدن) سپس چک کنید که آیا این پوششی که انتخاب کرده اید در تعریف پوشش راسی صدق می کند یا خیر(مرحله تصدیق کردن)، اگر صدق کرد جواب Yes تولید می شود

[۱] Minimum vertex cover
[۲] Minimum weighted vertex cover
[۳] cook

در غیر این صورت جواب No تولید می شود.)، حال باید ثابت کنیم که مساله ی 3-Sat کاهش پذیر به پوشش راسی می باشد، یعنی یک نمونه از مساله ی 3-Sat را می توان در یک زمان چند جمله ای به یک نمونه از مساله ی پوشش راسی تبدیل کرد، فرض کنید که B یک عبارت منطقی 3-Sat باشد، که آن را به صورت $B = (x_i, x_j, x_k)$ تعریف می کنیم، حال بایستی یک مثلث سه راسی که در آن هر راس به دو راس دیگر متصل است متناظر با یک لیترال در نظر می گیریم، در نهایت باید k مناسبی را برای پوشش راسی انتخاب کنیم. که در نهایت این k را $l + ۲m$ در نظر می گیریم که در آن m تعداد لیترال ها وl تعداد عبارت های منطقی می باشد.

$\Leftarrow$ فرض کنید که مساله ی 3-Sat حل پذیر باشد، آنگاه می توانیم لیترال های درست از پوشش راسی گراف G بسازیم (یعنی l گره از پوشش راسی). حال یک گره برای هر لیترال انتخاب می کنیم و یال بین آنها را پوشش می دهیم، چون فرض بر این است که 3-Sat حل پذیر است، از هر عبارت حداقل یک لیترال درست را انتخاب می کنیم. بنابراین هر عبارت منطقی مثلثی دارای حداقل یک گره با یال خارجی می باشد. اکنون ما حداکثر دو راس از هر مثلث را برای پوشش یالها نیاز داریم و این نتیجه می دهد که یک پوشش راسی با حداکثر اندازه ی $l + ۲m$ وجود دارد.

$\Rightarrow$ حال فرض کنید که یک پوشش راسی با حداکثر $l + ۲m$ راس وجود داشته باشد. می دانیم که حداقل l راس هر زوج لیترال را پوشش می دهد، زیرا هر یال بین آنها به طریق دیگری پوشش داده نمی شود. از طرفی حداکثر $۲m$ راس دیگر باید پوششی برای عبارت مثلثی باشند به این دلیل که هر مثلث نیاز به دو راس برای پوشش دارد. بیان کردیم که گره های لیترال در پوشش راسی متناظر با یک عبارت منطقی هستند با توجه به ابزارهای ساختن عبارت منطقی هر عبارت حل پذیر دارای ارزش درست می باشد، بنابراین این تبدیل m عبارت ، l لیترال را به صورت ورودی در یافت کرد و یک گراف با دقیقا $۲l + ۳m$ گره می سازد که این تبدیل چند جمله ای است. □

قضیه ۲.۱.۲. *S یک پوشش راسی است اگر و فقط اگر $V - S$ (مکمل مجموعه ی S) مجموعه ی مستقل باشد [۴].*

برهان. فرض کنید که $V - S$ یک مجموعه ی مستقل باشد. یال (u, v) در نظر بگیرید. چون $V - S$ یک مجموعه ی مستقل است بنابراین u و v هر دو نمی توانند عناصری از $V - S$ باشد. بنابراین حداقل یکی از رئوس u و v باید متعلق به S باشد. بنابر تعریف پوشش راسی نتیجه می شود که S پوشش راسی می باشد. □

قضیه ۳.۱.۲. *در هر گراف G عبارات زیر معادلند:*

G دارای یک مجموعه ی مستقل با حداقل k راس است.

متمم G دارای یک کلیک با حداقل k راس است.

G دارای یک پوشش راس با حداکثر $n-k$ راس است [۴].

قضیه ۴.۱.۲. *در گراف $G=(V, E)$ اجتماع ماکزیمم مجموعه مستقل و مینیمم پوشش راسی، مجموعه رئوس گراف V است [۱۲].*

۲.۲ مدل بهینه سازی و مدل تصمیم گیری پوشش راسی

در این قسمت می خواهیم مدل بهینه سازی و مدل تصمیم گیری مساله پوشش راسی را مورد بررسی قرار دهیم.

مدل تصمیم گیری پوشش راسی

ورودی: گراف بدون جهت G و عدد صحیح مثبت k

آیا در گراف داده شده G یک پوشش با حداکثر اندازه k وجود دارد؟ (جواب Yes و No است.) [۴]

مدل بهینه سازی پوشش راسی

ورودی: گراف بدون جهت G

خروجی: یک پوشش راس با اندازه ی مینیمم

در خیلی از موارد مساله ی پوشش راسی در حالت بهینه سازی و تصمیم گیری با هم رابطه دارند. که به عنوان نمونه یک قضیه در این رابطه بیان می کنیم. [۴]

قضیه ۱.۲.۲. *مدل بهینه سازی مساله ی پوشش راسی در زمان چند جمله ای حل می شود اگر وتنها اگر مدل تصمیم گیری آن در زمان چند جمله ای حل شود. (جزییات اثبات را می توان در [۱] مشاهده کنید.)*

۳.۲ مدل ILP

فرض کنید که هر راس دارای هزینه $c(v) \geq ۰$ باشد (در مدل بدون وزن، وزن هر راس را برابر یک می گیریم) مساله ی پوشش راسی وزن دار را می توان به عنوان برنامه ریزی عدد صحیح به صورت زیر فرمول بندی کرد [۴].

$$\begin{cases} \min \ \tilde{z} \approx \sum_{v \in V} c(v) x_v \\ \\ x_u + x_v \geq ۱ \forall u, v \in E \\ x_v \in \{۰, ۱\} \quad \forall v \in V \end{cases}$$

که در مدل های بدون وزن $c(v) = ۱$. برای حل مساله ی پوشش راسی با مدل ILP از روش شاخه و کران استفاده می شود که در قسمت های بعدی توضیح داده می شود.

۴.۲ مدل LP

مدل برنامه ریزی خطی پوشش راسی با در نظر گرفتن فرضیات مساله ی ILP به صورت زیر می باشد [۴]:

$$\begin{cases} \min \ \tilde{z} \approx \sum_{v \in V} c(v) x_v \\ \\ x_u + x_v \geq ۱ \forall u, v \in E \\ x_v \geq ۰ \quad \forall v \in V \end{cases}$$

۵.۲ مقایسه ی مدل ILP و مدل LP

LP مساله پوشش راس را بهتر از ILP حل می کند به دلیل اینکه ناحیه جواب بزرگتر می شود و هر جواب مساله ی ILP یک جواب برای مساله ی LP است و در مساله ی مینیمم سازی $\text{OPT(LP)} \leq \text{OPT(ILP)}$.

فصل ۳

بررسی الگوریتم هایی برای محاسبه ی پوشش راسی مینیمم

۱.۳ الگوریتم حریصانه برای مسله ی پوشش راسی

الگوریتم های حریصانه الگوریتم های ساختنی هستند یعنی سعی در ساختن جواب بهینه دارند. در بیشتر مسایل COP الگوریتم های حریصانه قادر به تولید جواب بهینه سراسری نیستند. ولی می توانند جواب های مناسبی برای نقطه ی شروع تولید کنند.

یک الگوریتم حریصانه برای پوشش راسی به صورت زیر می باشد [۱۲]،

ورودی: گراف ساده و بدون جهت $G = (V, E)$.

گام صفر: $C = \emptyset$.
گام یک: از بین راسها باقیمانده، راسی را انتخاب کن (v) که دارای بیشترین درجه باشد، $C \leftarrow C \cup \{v\}$.
گام دوم: v و یالهای (راس های منفرد) متصل به آن را حذف کن اگر راسی باقیمانده است به یک برو.

این الگوریتم جواب بهینه را نمی دهد، زیراپوشش راسی یک مساله ی ان-پی سخت است.

پیچیدگی این الگوریتم متعلق به $O(n^۲)$ است. و یک الگوریتم $\ln(n)$ تقریب است.

حال اگر ادعا کنیم که الگوریتم یک پوشش بهینه برای راس ارائه می دهد، براحتی می توانیم این ادعا را رد کنیم، زیرا اگر قرار می دهیم M را مجموعه ماکزیمال تطابق. یالهای انتخاب شده که بوضوح پوشیده شده اند. برای بقیه یالها که پوشیده نشده اند (فرض کنید که یال e متعلق به M نباشد)، $M \cup e$ یک تطابق است که ماکزیمال بودن M را بر هم می زند.

قضیه ۱.۱.۳. *پوشش ساخته شده توسط الگوریتم حریصانه $\ln(n)$ -تقریب می باشد [۴].*

متاسفانه می توان مثالی ساخت (برای دیدن مثال می توان به [۴] مراجعه کرد.) که در الگوریتم های حریصانه رشد خطا متناسب با $\ln(n)$ است، یعنی می تون مثالی ساخت که خطا را هرچه قدر خواسته باشیم زیاد کنیم.

۲.۳ الگوریتم تقریبی برای مسله ی پوشش راسی

ورودی: گراف ساده و بدون جهت G [۴].

گام صفر: $C = \emptyset$.
گام یک: یال (u, v) را به تصادف انتخاب کن و $C \leftarrow C \cup \{u, v\}$.
گام دوم: تمام یالهای مجاور با آن را حذف کن (یعنی تمام یالهای که به رئوس u و v متصل هستند.)

اکنون ادعا می کنیم که اگر C پوشش به دست آمده از الگوریتم باشد و C^* پوشش بهینه باشددر این صورت حداکثر خطای نسبی یک می باشد.

برهان. ابتدا می دانیم که C یک پوشش است و هر یال انتخاب شده توسط الگوریتم راس مشترکی ندارد زیرا وقتی یک مجموعه یال انتخاب می کنیم و یک مجموعه یال حذف می شود، چون هر یالی را که انتخاب می کندتمام یالهای متصل به آن حذف می شود، پس از این به بعد هر یالی که انتخاب می شود هیچ یک از آن دو سر وجود ندارند. از طرفی چون C^* یک پوشش است بنابراین از هر یال باید یک سر آن انتخاب شده باشد، فرض کنید که E' مجموعه ی کلیه یالهایی باشدکه توسط الگوریتم انتخاب شده اند و $e' = (a, a') \in E'$. C^* از هر یال باید حداقل یک راس بردارد، در C هر دو راس a, a' وجود دارد. چون C^* از هر یال باید یکی را بردارد پس C^* از نصف کمتر نمی باشد. پس

$$\frac{|C|-|C^*|}{|C^*|} \leq ۱.$$

□

توجه ۱. *در الگوریتم تقریبی کران خطا نسبی به اندازه ی ورودی ربطی ندارد.*

توجه ۲. *در الگوریتم تقریبی کران خطا به اندازه ی ورودی ربطی ندارد ولی در الگوریتم حریصانه هر چقدر که بخواهیم می توانیم کران خطا را زیاد کنیم [۴].*

۳.۳ استفاده از برنامه ریزی خطی در طراحی الگوریتم تقریبی

می دانیم که مسایل COP اکثرا به مدل ILP قابل بیان هستند، یکی از اصول طراحی الگوریتم های تقریبی استفاده از مدل LP مساله است یعنی ابتدا قیدهای صحیح بودن را آزاد می کنیم و پس از حل مساله LP جواب حاصل را با استفاده از الگوریتمی به یک جواب شدنی مساله ILP تبدیل می کنیم (جواب شدنی COP) اگر روی فاصله این جواب از جواب بهینه بتوانیم کرانی پیدا کنیم یک الگوریتم تقریبی در دست است. حال می خواهیم با استفاده از مدل LP مساله ی پوشش راسی وزن دار یک الگوریتم تقریبی برای آن بدست آوریم [۴].

۴.۳ مساله ی پوشش راسی وزن دار

ورودی گراف ساده ی $G = (V, E)$ که درآن $i = ۱, ..., |v|, w_i = w(v_i)$ وزن هر راس است هدف پیدا کردن یک $C \subseteq V$ به طوریکه C یک پوشش راسی باشد و $\sum_{v \in C} w(v)$ کمترین باشد.

مدلILP

$$x_i = \begin{cases} ۱ & v_i \in C \\ ۰ & v_i \notin C \end{cases}$$

$$\min \sum_{i=۱}^{|v|} w_i x_i$$
$$\begin{cases} x_i + x_j \geq ۱, & (v_i, v_j) \in E \\ x_i \in \{۰ \ , \ ۱\} \end{cases}$$

۵.۳ الگوریتم تقریبی پوشش راسی وزن دار بر اساس مدل LP

گام اول: مساله ی زیر را حل کنید در این مرحله قید صحیح بودن را آزاد می کنیم.

$$\min \sum_{i=۱}^{|v|} w_i x_i$$
$$\begin{cases} x_i + x_j \geq ۱ & (v_i, v_j) \in E \\ x_i \leq ۱ \\ x_i \geq ۰. \end{cases}$$

گام دوم: اگر فرض کنید که $x^{*\,LP}$ جواب بهینه مساله ی LP باشد برای $j = ۱, ..., |V|$ قرار می دهیم $C = C \cup \{v_j\}$ اگر $x_j^{*\,LP} \geq \frac{۱}{۲}$.

این الگوریتم دارای مرتبه ی زمانی چند جمله ای می باشد. و یک الگوریتم یک تقریب است.

یک الگوریتم تقریبی دیگر به صورت زیر می باشد.

فرض کنید که گراف $G = (V, E)$ داده شده باشد، در این بخش می خواهیم بررسی کنیم که چگونه یک پوشش راسی مینیمم را محاسبه کنیم. برای محاسبه ی پوشش راسی بهینه از یک الگوریتم شاخه ای)(branching که بر روی (G, S) تعریف می شود که در آن G یک گراف می باشد و $S \in V$ و دارای مقدار اولیه صفر می باشد. می خواهیم گراف G را به زیر گراف هایی بشکنیم و رئوسی از یک پوشش راسی را در S محاسبه کنیم. در واقع الگوریتم branching یک الگوریتم تکراری است، که در هر مرحله با یک درخت از زیر مساله سرو کار داریم.

الگوریتم branching در الگوریتم شاخه ای قاعده ی زیر بارها و بارها تکرار می شود:

فرض کنید که (H, S) که در آن H مخالف تهی است داده شده باشد، یال (u, v) را از گراف H بردار بنابراین مساله را به دو زیر مساله تقسیم می شود: ۱. $H(u), Suu$ با $H(U)$ ای که با H ای که تمام یالهایی که یک سر آنها راس u است برداشته شود و همچنین راسهای منفرد کنار گذاشته شود. ۲. $H(v), Suv$ با $H(v)$ ای که با H ای که تمام یالهایی که یک سر آنها راس u است برداشته شود و همچنین راسهای منفرد کنار گذاشته شود.

سپس درخت شاخه ای را با استفاده از روش جستجو در سطح گسترش دهید

قضیه ۱.۵.۳. *اولین سطح از درخت شاخه ای که شامل یک را س با برچسب $(۰, S)$ می باشد، اندازه ی پوشش راسی مینیمم را می دهد و S هم یک پوشش راسی مینیمم است. مرتبه ی زمانی پوشش راسی مینیمم متعلق به $۲OPT.|G|$.*

برهان. الگوریتم شاخه ای را می توانیم تا رسیدن به جواب بهینه ادامه دهیم.، این سطح شامل شاخه ای است با $۲OPT - ۱ = ۲OPT - ۱ + ... + ۲ + ۱$ راس و مرتبه زمانی هر دفعه فرخوانی الگوریتم شاخه ای متعلق به $O|G|$ می باشد. مسایل پارامتر ثابت (عنوان) یک مساله با پارامتر ثابت k را یک مساله با پارامتر ثابت می گوییم اگر آن مساله حل شدنی باشد و زمان الگوریتمی که آن مساله را حل می کنند متعلق به $O(f(k).poly(n))$ برای تابع معلوم f و عدد ثابت k (یعنی اینکه الگوریتم دارای زمان اجرای چند جمله ای باشد). خانواده ای از مسایل با پارامتر ثابت را با FTP نمایش می دهیم (برای مشاهده ی جزییات بیشتر به [۶] مراجعه کنید). □

قضیه ۲.۵.۳. *مساله ی پوشش راسی متعلق به FTP*

قضیه ۳.۵.۳. *الگوریتم شاخه ای را برای k مرحله انجام دهید، اگر گره ای با بر چسب $(۰, S)$ وجود داشته باشد، جواب yes است در غیر این صورت جواب No است. در این صورت زمان اجرای این الگوریتم متعلق به $O(۲^k.|G|) = O(۲^k.(n + m)$*

۶.۳ الگوریتم جستجوی محلی بر روی مساله ی پوشش راسی

الگوریتم های مختلفی برای محاسبه ی مینیمم پوشش راسی وجود دارد اما بهترین نتایج به شرح زیر می باشد: قدیمی ترین تلاش برای حل مسایل COP الگوریتم پوشش راسی می باشد، البته در بیشتر مواقع این الگوریتم جواب بهینه ی را با تقریبی با احتمال صفر پیدا می کند. این الگوریتم بسیار خوب است و جواب را در هر گام بهتر می کند به همین دلیل این الگوریتم به بهبود بخشنده [۱] معروف است. در مورد الگوریتم جستجوی محلی سه نکته از اهمیت بسیار زیادی برخوردار است که عبارتند از: نقطه ی شروع، ناحیه همسایگی و تغییرات. به طور کلی می توان گفت الگوریتم جستجوی محلی ساختارش بر روی همسایگی بنا شده است. همسایگی برای مساله ی پوشش راسی را می توان اینگونه تعریف کرد که اگر s به عنوان مرکز همسایگی باشد در اطراف این نقطه به دنبال پوشش های راسی دیگری می گردیم که تعداد اعضای آن ها کمتر از مرکز باشد و یا اینکه تعداد اعضای آن ها با هم برابر باشد و تنها در یک عضو اختلاف داشته باشند. پیدا کردن نقطه شروع برای جستجوی محلی در پوشش راسی کار سختی نمی باشد کافیست یک پوشش تصادفی را به عنوان نقطه ی شروع بگیریم. در این صورت الگوریتم به صورت زیر می باشد،

گام صفر: نوع همسایگی و نقطه ی شروع و شرایط توقف را وارد کنید.
گام اول: $S^c \leftarrow S^\circ$
گام دوم: اگر شرایط توقف حاصل شده است S^c را به عنوان جواب برگردان.
گام سوم: جوابی در همسایگی S^c یعنی $y \in S^c$ را انتخاب کنید. منظور از انتخاب این است که اگر $C(y) < C(S^c) \Rightarrow S^c \leftarrow y$.
گام چهارم: به گام دو برگرد.

۷.۳ الگوریتم bar-Yehuda

برای مدل LP این الگوریتم به صورت زیر می باشد [۷].

گام اول: در ابتدا هیچ راسی انتخاب نشده است.
گام دوم: تا زمانی که یک یال پوشش داده نشده است دستورات زیر را تکرار کن،
یک یال دلخواه را انتخاب کن.
یکی از دو سر یال انتخابی را در نظر بگیر و وزن آن را y_i را به یال انتخاب شده بده و با وزن یالهایی که از قبل انتخاب شده بود جمع کن $w = w + y_i$ و سپس تمام یالهای منتهی به این وزن صفر کن و تا زمانی انجام بده که y_i ای که در شرط $w \leq y_i$ وجود داشته باشد.
گام سوم: مجموعه راس های انتخاب شده یک پوشش برای گراف است.

---

[۱] improving search

قضیه ۱.۷.۳. *الگوریتم (Bar-Yehuda) برای گراف کامل با n*

$$OPT_{LP} = \frac{n}{۲}$$
$$OPT_{IP} = n - ۱$$
$$\lim_{n\longrightarrow\infty} \frac{OPT_{IP}}{OPT_{LP}} = \lim_{n\longrightarrow\infty} \frac{n-۱}{\frac{n}{۲}} = ۲$$

۸.۳ الگوریتم شاخه و کران

الگورتم شاخه و کران کامل است به این معنی که جواب صحیح را تضمین می کند [۱۲] حتی با وجود اینکه پیچیدگی زمانی آن می تواند با سایز گراف نمایی باشد. این الگوریتم اغلب بر روی روش های اتفاقی اجرا می شود که در بیشتر اوقات یک جواب بهینه ی محلی را بعد از جستجوی قسمت کوچکی از فضای جستجو می دهد.

شاخه و کران یک الگوریتم بازگشتی است که با تصمیم گیری در مورد حضور و یا عدم حضور یک نقطه در پوشش در هر مرحله با بازگشت به نقطه های فضای وضعیت را کشف می کند. کل فضای وضعیت در هر مرحله به وسیله یک درخت نشان داده می شود که هر سطح آن برای حضور یا عدم حضور یک نقطه تصمیم گیری می شود و هر نقطه دو امکان دارد یکی مرتبط با انتخاب آن نقطه در پوشش است و امکان دیگر در واقع نادیده گرفتن آن نقطه است. یک نقطه و همه ی یال های مجاور با آن ها برداشته می شوند تا اینکه یک نقطه نادیده گرفته شده باشد باقی می ماند اما این نقطه برای سطح های پایین تر امکان انتخاب ندارد پس بازگشت به صورت می گیرد. به طور کلی بازگشت هنگامی صورت می گیرد که وقتی هیچ یال بیشتری برای پوشش وجود ندارد یا وقتی شرایط کران ملاقات شده است. زیر مجموعه ای از نقاط که یک پوشش معتبر را به دست می دهد شناخته م یشود و کمترین آن به عنوان پوشش راس انتخاب می شود. کران قادر می سازد که شاخه ها یی را که نتیجه ی مطلوب نبوده حذف شوند. شرایط کران برای راس جاری i درجه راس $d(i)$ می باشد که با تعدادی همسایگی برابر است. اگر uncov تعداد یال هایی ابشد که هنوز پوشیده نشده است و ما هنوز k نقطه باقیمانده برای اینکه پوشیده شود داشته باشیم کران پایین تعداد یال های پوشیده نشده به صورت زیر است.

$$\max[۰, uncov - \max d(j۱) + d(j۲) + ... + d(jk)]$$

هر شاخه ای که این کران را نقض کند شکسته می شود. الگوریتم که در ادامه بیان می شود اندازه ی کوچکترین پوشش در best و تعداد پوشش داده شده در X ذخیره می شود که می خواهیم مساله ی پوشش راس را بر روی آن اجرا کنیم.

begin
if all edges all covered then
begin
if X< best then
best=X
return
end
F=best-X
$D = d_۱ + ... + d_t$
if D<number of uncovered edges then.
return. comment bound
take one free vertex i with largest current degree di
mark i as covererd; comment left subtree
$X = X - ۱$
remove all incident edges i, j from E
branch and bound (G, best, X)
insert all edges i, j which have been removed
$X = X - ۱$
if X> number of current neighbors then
begin comment right subtree
mark i as uncovered
for all neighbors j of i do
begin
mark j as covered
$X = X + ۱$
remove all incident edges j, k from E
end
branch and bound (G, best, X)
for all neighbors j of i do
begin
mark j as free
$X = X - ۱$
end
insert all edges j, k whitch have been removed
end
mark i as free
return
end

در هر بار به روز رسانی کران F بزرگترین مجموعه ی همبندی راس های جاری به روز می شود. بنابرای بهتر است جدول از راس ها به عنوان دو آرایه V_1 و V_2 پیاده سازی کرد این آرایه فهرستی از در جه راس ها هستند. مجموعه ی V_1 شامل F راس آزاد از بزرگترین درجات جاری هستند و آرایه دیگرر شامل بقیه راس های آزاد است. هنگامی که درجه ی راس تغییر می کند این راس به مجموعه ی دیگری منتقل می شود و سرانجام آرایه تغییر می کند.

فصل ۴

الگوریتم های فرا ابتکاری

روش‌ها و الگوریتم‌های بهینه‌سازی به دو دسته الگوریتم‌های دقیق [۱] و الگوریتم‌های تقریبی [۲] 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. الگوریتم‌های تقریبی قادر به یافتن جواب‌های خوب (نزدیک به بهینه) در زمان حل کوتاه برای مسائل بهینه‌سازی سخت هستند. الگوریتم‌های تقریبی نیز به دو دسته الگوریتم‌های ابتکاری [۳] و فراابتکاری [۴] تقسیم‌بندی می شوند. دو مشکل اصلی الگوریتم‌های ابتکاری، قرار گرفتن آنها در بهینه‌های محلی، و عدم قابلیت آنها برای کاربرد در مسائل مختلف است. الگوریتم‌های فراابتکاری یا متاهیوریستیک‌ها برای حل این مشکلات الگوریتم‌های ابتکاری ارائه شده‌اند. در واقع الگوریتم‌های فراابتکاری، یکی از انواع الگوریتم‌های بهینه‌سازی تقریبی هستند که دارای مکانیزم‌های خروج از بهینه محلی می‌باشند و قابل کاربرد در طیف وسیعی از مسائل هستند. که در این بخش به بیان دو الگوریتم فرا ابتکاری به نام های مورچگان و ژنتیک می پردازیم.

۱.۴ الگوریتم مورچگان

استفاده از الگوریتم‌های فرا ابتکاری در حل مسئله بهینه‌سازی امری ضروری و اجتناب‌ناپذیر است. این روش [۱۳] از توانایی مورچه‌ها در پیدا کردن کوتاه‌ترین مسیر بین لانه و یک منبع غذایی الهام گرفته است. وقتی مورچه‌ها در محیط اطراف حرکت می‌نمایند، اثری شیمیایی به نام فرومون از خود بجای می‌گذارند. وقتی جمعیتی از مورچه‌ها از چند

[۱] Exact
[۲] Approximate algortithms
[۳] Heuristic
[۴] Meta-heuristic

مسیر بین لانه و یک منبع غذایی حرکت می‌کنند، پس از مدت زمان معینی مشاهده می‌شود که در مسیرهای متفاوت، فرومونهای برجای گذاشته شده متفاوت می‌باشد. این امر ناشی از این واقعیت است که مورچه‌هایی که در مسیر کوتاه حرکت می‌کنند، به علت کوتاه‌تر بودن مسیر در یک مدت زمان معین‌تردد بیشتری داشته‌اند چون مورچه‌ها، مسیر کوتاه‌تر را انتخاب کرده‌اند. با استفاده از روش مورچه‌ها، روش جستجوئی پیاده‌سازی می‌شود که در هر مرحله‌ای از اطلاعات مراحل قبلی برای رسیدن به هدف استفاده میگردد.

۲.۴ تاریخچه الگوریتم مورچگان

الگوریتم بهینه سازی کلونی مورچه ها [۵] در سال ۱۹۹۲ توسط مارکو دوریگو [۶] و در رساله دکتری وی مطرح شد، یکی از بارزترین نمونه ها برای روش های هوش جمعی است. این الگوریتم از روی رفتار جمعی مورچه ها الهام گرفته شده است. مورچه ها با همکاری یکدیگر، کوتاه ترین مسیر را میان لانه و منابع غذایی پیدا می کنند تا بتوانند در کمترین زمان مواد غذایی را به لانه منتقل کنند. هر کدام از مورچه ها، به تنهایی قادر به انجام چنین کاری نیستند. اما با همکاری و پیروی از چند اصل ساده، بهترین راه را پیدا می کنند. به عنوان مثال، عملکرد مورچه های آرژانتینی در یافتن کوتاه ترین مسیر بین لانه و منبع غذایی، بسیار عجیب و حیرت انگیز است. مورچه آرژانتینی عملا کور است و طبعا کوتاه ترین مسیر برای او مفهومی ندارد و توسط او قابل شناخت نمی باشد. اما با وجود چنین کمبودی، توده ای از این مورچه ها می توانند با همکاری یکدیگر، کوتاه ترین مسیر موجود بین لانه و محل مواد غذایی را پیدا کنند. این الگوریتم برای حل مسائلی که به صورت پیدا کردن کوتاه ترین مسیر در یک گراف قابل بیان هستند، طراحی شده است.

الگوریتم بهینه سازی کلونی مورچه ها یا ACO از رفتار مورچه های طبیعی که در مجموعه ها بزرگ در کنار هم زندگی می کنند الهام گرفته شده است. الگوریتم های دیگری نیز بر اساس الگوریتم مورچه ها ساخته شده اند که همگی سیستم های چند عاملی هستند و عامل ها مورچه های مصنوعی یا به اختصار مورچه هایی هستند که مشابه با مورچه های واقعی رفتار می کنند. الگوریتم مورچه ها، یک مثال بارز از هوش جمعی هستند که در آن عامل هایی که قابلیت چندان بالایی ندارند، در کنار هم و با همکاری یکدیگر می توانند نتایج بسیار خوبی به دست بیاورند. این الگوریتم برای حل

[۵] Ant Colony
[۶] Marco Dorigo

و بررسی محدوده وسیعی از مسائل بهینه سازی به کار برده شده است.

۱) اجتماعی بودن:

مطالعات نشان داده است که مورچه‌ها حشراتی اجتماعی هستند که در کلونی‌ها زندگی می‌کنند و رفتار آنها بیشتر در جهت بقاء کلونی است تا در جهت بقاء یک جزء از آن .

در دنیای واقعی مورچه‌ها ابتدا به طور تصادفی به این سو و آن سو می‌روند تا غذا بیابند. سپس به لانه بر می‌گردند و ردّی از فرومون [۷] به جا می گذارند. چنین ردهایی پس از باران به رنگ سفید در می‌آیند و قابل رویت اند. مورچه‌های دیگر وقتی این مسیر را می‌یابند، گاه پرسه زدن را رها کرده و آن را دنبال می‌کنند. سپس اگر به غذا برسند به خانه بر می‌گردند و رد دیگری از خود در کنار رد قبل می گذارند؛ و به عبارتی مسیر قبل را تقویت می‌کنند. فرومون به مرور تبخیر می‌شود که از سه جهت مفید است. باعث می‌شود مسیر جذابیت کمتری برای مورچه‌های بعدی داشته باشد. از آنجا که یک مورچه در زمان دراز راه‌های کوتاه‌تر را بیش تر می‌پیماید و تقویت می‌کند هر راهی بین خانه و غذا که کوتاه‌تر(بهتر) باشد بیشتر تقویت می‌شود و آنکه دورتر است کمتر.

اگر فرومون اصلاً تبخیر نمی‌شد، مسیرهایی که چند بار طی می‌شدند، چنان بیش از حد جذّاب می‌شدند که جستجوی تصادفی برای غذا را بسیار محدود می‌کردند.

وقتی غذای انتهای یک مسیر جذاب تمام می‌شد رد باقی می ماند.لذا وقتی یک مورچه مسیر کوتاهی (خوبی) را از خانه تا غذا بیابد بقیهٔ مورچه‌ها به احتمال زیادی همان مسیر را دنبال می‌کنند و با تقویت مداوم آن مسیر و تبخیر ردهای دیگر، به مرور همهٔ مورچه‌ها هم مسیر می‌شوند. هدف الگوریتم مورچه‌ها تقلید این رفتار توسط مورچه‌هایی مصنوعی ست که روی نمودار در حال حرکت اند. مساله یافتن کوتاه‌ترین مسیر است و حلالش این مورچه‌های مصنوعی اند.

۲) هوش جمعی:

مورچه‌ها با وجود کور و کم‌هوش بودن کوتاهترین مسیر رفت و برگشت از خانه تا غذا را پیدا می‌کنند. این یکی از مهمترین و جالبترین رفتار مورچه‌ها می‌باشد که این نوع رفتار مورچه‌ها دارای نوعی هوشمندی توده‌ای است که عناصر رفتاری تصادفی(احتمال) دارند و بین آنها (همدیگر) هیچ نوع ارتباط مستقیمی وجود ندارد و آنها تنها بصورت غیرمستقیم و با استفاده از نشانه‌ها با یکدیگر در تماس هستند.

مورچه‌ها چگونه کوتاهترین مسیر را انتخاب می‌کنند؟

[۷] Pheromone

مورچه‌ها هنگام راه رفتن از خود ردی از ماده شیمیایی فرومون می‌گذارند که البته این ماده بزودی تبخیر می‌شود ولی در کوتاه مدت بعنوان رد مورچه بر سطح زمین باقی می‌ماند. یک رفتار پایه‌ای ساده در مورچه‌ها وجود دارد.آنها هنگام انتخاب بین دو مسیر بصورت احتمالاتی [۸] مسیری را انتخاب می‌کنند که فرومون بیشتری داشته باشد یا بعبارت دیگر مورچه‌های بیشتری قبلاً از آن جا عبور کرده باشند.

مسیریابی با الهام از کلونی مورچه ها:

ترشح اسید فرمیک در مسیر حرکت * دنبال کردن مسیرهای با اسید فرمیک بیشتر * تبخیر

تعاملات محلی ، محدود و ساده اعضای یک دسته و جمعیت با محیط ، منتهی به یک رفتار جمعی هوشمندانه میشوداین تعاملات غالبا غریزی بوده وبدون نظارت انجام می گیرندنتیجه آن غالبا یک رفتار پیچیده و هوشمندانه جمعی و بطورخاص انجام بعضی بهینه سازی های پیچیده است. این نوع هوشمندی هیچ نیازی به کنترل مرکزی و دید کلی نسبت به سیستم ندارد. الگوریتم های مبتنی برآمیزی در حل مسائل بهینه سازی ترکیبی داشته اند، در بر گیرندهٔ تعداد زیادی مورچه است که این مورچه ها در طول گراف حرکت می کنند وکوتاهترین مسیر را پیدا می کنند . مورچه ها ، نوعادانشی درباره اینکه کدام مسیر کوتاه تر است را ندارند بنابراین آنها به تنهایی ، فقط یک مسیر با کیفیت پایین را می توانند پیدا کنند ، ولی هماهنگی سراسری در میان مورچه های یک کولونی باعث می شود که مسیر های بهینه و کوتاه پیدا شوند . رفتار این مورچه ها(مورچه های مصنوعی) ، از مورچه های واقعی مدل می شود . درجهان واقعی مورچه ها در حین حرکت درطول مسیرشان یک مقدار فرومون را در مسیر از خودبه جای می گذارند . تصمیم گیری حرکت مورچه دریک مسیر بر اساس غلظت فرومون آن مسیر انجام میشود . مورچه ها ترجیح می دهند از مسیری حرکت کنند که مقدار فرومون آن زیاد است . از طرفی مورچه های مصنوعی یکسری خصوصیات گسترده تری دارند

که در مورچه های طبیعی یافت نمی شود . به اینصورت که حرکت آنها معمولا سازگار با عملیات قبلی شان است که در یک ساختار داده ویژه ای ذخیره شده است . این مورچه ها در طول حرکت از یک گره به گره دیگر مقداری فرومون را متناسب با مسیر ازخود به جای می گذارند ، به این صورت که اگر مورچه در حرکت خود از گره مبدأ به گره مقصد مسیر کوتاه ومناسبی را انتخاب کرده باشد ، میانگین توزیع فرومون در آن مسیر زیاد خواهد بود و از طرف دیگر اگر مسیرضعیفی را طی کرده باشد ، مقدار فرومون در طول مسیر کم خواهد بود. در واقع مقدار فرومون باقی مانده در مسیر متناسب با کیفیت مسیر است . به این ترتیب مورچه ها در یک گراف ، کوتاهترین مسیر ها را پیدا می کنند.

[۸] statistical

۳.۴ ویژگیهای الگوریتم مورچگان

این الگوریتم مورچگان:

. چندمنظوره می باشد

. می‌تواند برای انواع مشابه یک مسأله به کار رود.

. قوی می باشد

یعنی با کمترین تغییرات برای دیگر مسائل بهینه‌سازی ترکیبی به کار برده می‌شود

۳.یک روش مبتنی بر جمعیت می‌باشد. مزیتهای ACO : ایجاد انعطاف در حل هرگونه مسئله بهینه‌سازی پسخورد مثبت (پسخورد مثبت، منجر به کشف سریع جوابهاب خوب می‌شود) محاسبات توزیع شده (محاسبات توزیع شده از همگرایی زودرس و بی‌موقع جلوگیری می‌کند) هیوریستیک آزمند سازنده (به کشف جوابهای قابل قبول در مراحل اولیه جستجو کمک می‌کند).

کاربردهای الگوریتم مورچگان : از کاربردهای الگوریتم ACO می‌توان به بهینه کردن هر مسئله‌ای که نیاز به یافتن کوتاهترین مسیر دارد استفاده می شود:

·مسیریابی داخل شهری و بین شهری

·مسیریابی بین پست‌های شبکه‌های توزیع برق ولتاژ بالا

·مسیریابی شبکه‌های کامپیوتری

· مسیر یابی تامین مواد اولیه جهت تولید به هنگام

·برنامه ریزی دروس دانشگاهی

·توازن بار ترافیک شبکه ومسیریابی مبتنی برمهندسی ترافیک

·کاوش استفاده از وب با استفاده از کلونی مورچه ها

·مسئله زمان بندی حرکت قطار ها

·الگوریتم ی مورچه و کاربرد آن در برنامه ریزی پرواز

·بهینه سازی سکوهای دریا

·استفاده ازالگوریتمهای الهام گرفته از کلونی مورچه ها در مسیریابی شبکه های کامپیوتری

·مسأله راهیابی در شبکه های مخابرات راه دور.

۴.۴ الگوریتم مورچگان برای مساله ی پوشش راسی

الگوریتم ACO برای یافتن پوشش راسی مینیمم به صورت زیر می باشد:

Initialize
Represent the underlying problem by a connected graph.
Set initial pheromone to a small constat for each edge.
Reapeat
For each ant do
Repeat
Move to the next node according to the node transition rule.
Until a trial solution is constructed.
For each edge do
Update the pheromone updating rule.
Until the stopping criterion is satisfied.
Output the global best solution.

مجموعه ی $V' \subseteq V$ را در نظر می گیریم، مورچه باید دقیقا بر روی پوششی از راسها ی V' حرکت کند. اما ممکن است یک مسیر بین این راسها وجود نداشته باشد که برای غلبه بر مشکل گراف کامل $Gc = (V, Ec)$ را طوری در نظر می گیریم که شامل راسهای V باشد و هر جفت از راسهای آن به یک یال از Ec متصل باشد.

برای هر یال (i, j) ما یک تابع همبند به صورت $C : E_۰ \to ۰, ۱$ که $C(i,j) = \begin{cases} ۱, & (i,j) \in E \\ ۰ & (i,j) \in E_۰ - E \end{cases}$

۵.۴ به روز کردن شرایط در مساله ی پوشش راسی

ارزش همبندی یک یال در E وقتی یالهای آن راس توسط مورچه ملاقات شد به روز می شود [۱۳] برای مورچه ی k -ام داریم

$$C(i,\ j) = \frac{۱}{n}$$

برای $C(i,\ j) \in E$ برای راس i یا j توسط مورچه k -ام ملاقات شود. که n تعداد راس های گراف است.

این به روز رسانی دارای دو سود است:

درجه ی تمایل برای هر راس که به طور دقیق برای راس بعدی که انتخاب خواهد شد از رابطه زیر تخمین زده می شود.

$$D_j^k = \sum_{(r,\ j)} C(r,\ \ j)$$

برای هر راس که عدد مربوط به آن بزرگتر باشد انتخاب می شود، اگر کوچکتر از یک باشد مورچه متوقف می شود.

نکته: قبل از شروع سیکل بعدی ارزش همبندی باید $C(i,\ j)$

۶.۴ قوانین تغییر حالت

یکی از قسمت های اصلی حل ACO قوانین تغییر حالت است [۱۳]. که نتایج در دو بخش بهینگی و بهره وری بهبود یابد. این مرحله را تغییر حالت می نامند. لذا یک بهره وری را برای آن تغییر می کنیم و عدد مربوط به برتری را روی راس ها قرار می دهیم. قوانین مرحله تغییر حالت احتمال انتخاب راس بعدی برای مورچه k را توصیف می کند،

$$P_j^k = \frac{\tau^\alpha j \eta_{jk}^\beta}{\sum\limits_{r \in A_k} \tau^\alpha r \eta_{rk}^\beta}$$

که A_k مجموعه ی راس های دست یافتنی برای مورچه k ام است.

همچنین $\tau^\alpha j$ شدت فرمون ها برای راس j و η_{rk}^β میزان وضوح فرمون هاست.

مورچه k-ام آنقدر نیرو ندارد که همه ی راس ها را برای یافتن بهترین راس طی کند. بنابراین ما v_k را مجموعه راس ها یی که مورچه k-ام می بیند تعریف می کنیم.

$$V_k = \{j \mid \sum_{r \in A_k} C(r,\ j)\}.$$

۷.۴ قوانین به روز کردن فرمون در مساله پوشش راسی

الگوریتم ACO به همکاری بین مورچه های نماینده تکیه می کند و قوانین به روزکردن فرمون ها را به صورت سراسری و محلی اجرا می کنیم [۱۳].

۱) در انتهای هر سیکل فرمون ها بر روی راس هایی که جواب بهتر جاری را دارند قرار می گیرند.

فرض کنیم V_c' بهترین جواب جاری باشد برای هر $i \in V_c'$ فرمون ها از قوانین سراسری به روز می شوند

$$\iota_i = (۱ - \rho)\tau_i + \rho \Delta\tau_i$$

که

$$\Delta\tau_i = \frac{۱}{|V_c'|}$$

و $\rho(۰, ۱)$ یک پارامتر بدست آمده از شبیه سازی کردن میزان تبخیر چگالی فرمون هاست و موجب م یشود مورچه تصمیم بد قبلی را فراموش کند.

۲) قوانین جستجوی محلی را به این علت به کار می بریم که به مورچه اجازه دهیم تا مورچه جدید فضای جواب را در راس های دورتر برای یافتن جواب مناسب با احتمال بالاتری جستجو کند که این کار با استفاده از رابطه ی زیر انجام می شود:

$$\tau_i = (۱ - \rho')\tau_i + \rho'\tau_۰$$

که $\rho'(۰, ۱)$ پارامتر سازگاری فرمون های قبلی بر روی راس i است و τ_n مقدار اولیه فرمون ها بر روی راس قبل از شروع الکوریتم است.

۸.۴ ضابطه یا معیار توقف الگوریتم مورچگان برای مساله ی پوشش راسی

می تواند ماکزیمم میزان تکرار، بر روی یک CPU ثابت یک زمان محدود، یا هر معیار توقف دیگری که نتیجه الگوریتم را بهتر کند [۱۳].

۹.۴ الگوریتم ژنتیک

الگوریتم ژنتیک چیست؟

الگوریتم ژنتیک [۹] تکنیک جستجویی در علم رایانه برای یافتن راه‌حل تقریبی برای بهینه‌سازی و مسائل جستجو است [۱۱]. الگوریتم ژنتیک نوع خاصی از الگوریتمهای تکامل است که از تکنیکهای زیست‌شناسی فرگشتی مانند وراثت و جهش استفاده می‌کند.

در واقع الگوریتم‌های ژنتیک از اصول انتخاب طبیعی داروین برای یافتن فرمول بهینه جهت پیش‌بینی یا تطبیق الگو استفاده می‌کنند.الگوریتم‌های ژنتیک اغلب گزینه خوبی برای تکنیک‌های پیش‌بینی بر مبنای رگرسیون هستند. مختصراً گفته می‌شود که الگوریتم ژنتیک (GA) یک تکنیک برنامه‌نویسی است که از تکامل ژنتیکی به عنوان یک الگوی حل مسئله استفاده می‌کند.مسئله‌ای که باید حل شود ورودی است و راه‌حلها طبق یک الگو کد گذاری می‌شوند که تابع fitness نام دارد هر راه حل کاندید را ارزیابی می‌کند که اکثر آنها به صورت تصادفی انتخاب می‌شوند.

[۹] Genetic Algorithm - GA

الگوریتم‌های ژنتیک از اصول انتخاب طبیعی داروین برای یافتن فرمول بهینه جهت پیش‌بینی یا تطبیق الگو استفاده می‌کنند. الگوریتم‌های ژنتیک اغلب گزینه خوبی برای تکنیک‌های پیش‌بینی بر مبنای رگرسیون هستند. رای مثال اگر بخواهیم نوسانات قیمت نفت را با استفاده از عوامل خارجی و ارزش رگرسیون خطی ساده مدل کنیم، این فرمول را تولید خواهیم کرد : قیمت نفت در زمان t = ضریب ۱ نرخ بهره در زمان t + ضریب ۲ نرخ بیکاری در زمان t + ثابت ۱ . سپس از یک معیار برای پیدا کردن بهترین مجموعه ضرایب و ثابت‌ها جهت مدل کردن قیمت نفت استفاده خواهیم کرد. در این روش ۲ نکته اساسی وجود دارد. اول این که روش خطی است و مسئله دوم این است که ما به جای اینکه در میان ''فضای پارامترها'' جستجو کنیم، پارامترهای مورد استفاده را مشخص کرده‌ایم.

با استفاده از الگوریتم‌های ژنتیک ما یک ابر فرمول یا طرح، تنظیم می‌کنیم که چیزی شبیه ''قیمت نفت در زمان t تابعی از حداکثر ۴ متغیر است'' را بیان می‌کند. سپس داده‌هایی برای گروهی از متغیرهای مختلف، شاید در حدود ۲۰ متغیر فراهم خواهیم کرد. سپس الگوریتم ژنتیک اجرا خواهد شد که بهترین تابع و متغیرها را مورد جستجو قرار می‌دهد. روش کار الگوریتم ژنتیک به طور فریبنده‌ای ساده، خیلی قابل درک و به طور قابل ملاحظه‌ای روشی است که ما معتقدیم حیوانات آنگونه تکامل یافته‌اند. هر فرمولی که از طرح داده شده بالا تبعیت کند فردی از جمعیت فرمول‌های ممکن تلقی می‌شود.

متغیرهایی که هر فرمول داده‌شده را مشخص می‌کنند به عنوان یکسری از اعداد نشان داده‌شده‌اند که معادل (DNA) آن فرد را تشکیل می‌دهند.

موتور الگوریتم ژنتیک یک جمعیت اولیه از فرمول ایجاد می‌کند. هر فرد در برابر مجموعه‌ای از داده‌های مورد آزمایش قرار می‌گیرند و مناسبترین آنها (شاید ۱۰ درصد از مناسبترین‌ها) باقی می‌مانند؛ بقیه کنار گذاشته می‌شوند. مناسبترین افراد با هم جفتگیری (جابجایی عناصر دی ان ای) و تغییر (تغییر تصادفی عناصر دی ان ای) کرده‌اند. مشاهده می‌شود که با گذشت از میان تعداد زیادی از نسلها، الگوریتم ژنتیک به سمت ایجاد فرمول‌هایی که دقیقتر هستند، میل می‌کنند. در حالی که شبکه‌های عصبی هم غیرخطی و غیرپارامتریک هستند، جذابیت زیاد الگوریتم‌های ژنتیک این است نتایج نهایی قابل ملاحظه‌ترند. فرمول نهایی برای کاربر انسانی قابل مشاهده خواهد بود، و برای ارائه سطح اطمینان نتایج می‌توان تکنیک‌های آماری متعارف را بر روی این فرمول‌ها اعمال کرد. فناوری الگوریتم‌های ژنتیک همواره در حال بهبود است و برای مثال با مطرح کردن معادله ویروس‌ها که در کنار فرمول‌ها و برای نقض کردن فرمول‌های ضعیف تولید می‌شوند و در نتیجه جمعیت را کلاً قویتر می‌سازند.

مختصراً گفته می‌شود که الگوریتم ژنتیک یک تکنیک برنامه‌نویسی است که از تکامل ژنتیکی به عنوان یک الگوی حل مسئله استفاده می‌کند. مسئله‌ای که باید حل شود ورودی است و راه حلها طبق یک الگو کدگذاری می‌شوند که تابع fitness نام دارد و هر راه حل کاندید را ارزیابی می‌کند که اکثر آنها به صورت تصادفی انتخاب می‌شوند.

الگوریتم ژنتیک (GA) یک تکنیک جستجو در علم رایانه برای یافتن راه حل بهینه و مسائل جستجو است. الگوریتم‌های ژنتیک یکی از انواع الگوریتم‌های تکاملی‌اند که از علم زیست‌شناسی مثل وراثت، جهش، [انتخاب ناگهانی(زیست‌شناسی)]انتخاب ناگهانی]، انتخاب طبیعی و ترکیب الهام گرفته شده.

عموماً راه‌حلها به صورت ۲ تایی ۰ و ۱ نشان داده می‌شوند، ولی روشهای نمایش دیگری هم وجود دارد. تکامل از یک مجموعه کاملاً تصادفی از موجودیت‌ها شروع می‌شود و در نسلهای بعدی تکرار می‌شود. در هر نسل، مناسبترین‌ها انتخاب می‌شوند نه بهترین‌ها.

یک راه‌حل برای مسئله مورد نظر، با یک لیست از پارامترها نشان داده می‌شود که به آنها کروموزوم یا ژنوم می‌گویند. کروموزوم‌ها عموماً به صورت یک رشته ساده از داده‌ها نمایش داده می‌شوند، البته انواع ساختمان داده‌های دیگر هم می‌توانند مورد استفاده قرار گیرند. در ابتدا چندین مشخصه به صورت تصادفی برای ایجاد نسل اول تولید می‌شوند. در طول هر نسل، هر مشخصه ارزیابی می‌شود وارزش تناسب [۱۰] توسط تابع تناسب اندازه‌گیری می‌شود. گام بعدی ایجاد دومین نسل از جامعه است که بر پایه فرآیندهای انتخاب، تولید از روی مشخصه‌های انتخاب شده با عملگرهای ژنتیکی است: اتصال کروموزوم‌ها به سر یکدیگر و تغییر.

برای هر فرد، یک جفت والد انتخاب می‌شود. انتخاب‌ها به گونه‌ای‌اند که مناسبترین عناصر انتخاب شوند تا حتی ضعیفترین عناصر هم شانس انتخاب داشته باشند تا از نزدیک شدن به جواب محلی جلوگیری شود. چندین الگوی انتخاب وجود دارد: چرخ منگنه‌دار(رولت)، انتخاب مسابقه‌ای [۱۱] و....

معمولاً الگوریتم‌های ژنتیک یک عدد احتمال اتصال دارد که بین ۰.۶ و ۱ است که احتمال به وجود آمدن فرزند را نشان می‌دهد. ارگانیسم‌ها با این احتمال دوباره با هم ترکیب می‌شوند. اتصال ۲ کروموزوم فرزند ایجاد می‌کند، که به نسل بعدی اضافه می‌شوند. این کارها انجام می‌شوند تا این که کاندیدهای مناسبی برای جواب، در نسل بعدی پیدا شوند. مرحله بعدی تغییر دادن فرزندان جدید است. الگوریتم‌های ژنتیک یک احتمال تغییر کوچک و ثابت دارند که معمولاً درجه‌ای در حدود ۰.۰۱ یا کمتر دارد. بر اساس این احتمال، کروموزوم‌های فرزند به طور تصادفی تغییر می‌کنند

[۱۰] fitness
[۱۱] Tournament

یا جهش می‌یابند، مخصوصاً با جهش بیت‌ها در کروموزوم ساختمان داده‌مان.

این فرآیند باعث به وجود آمدن نسل جدیدی از کروموزوم‌هایی می‌شود، که با نسل قبلی متفاوت است. کل فرآیند برای نسل بعدی هم تکرار می‌شود، جفت‌ها برای ترکیب انتخاب می‌شوند، جمعیت نسل سوم به وجود می‌آیند و .... این فرآیند تکرار می‌شود تا این که به آخرین مرحله برسیم.

شرایط خاتمه الگوریتم‌های ژنتیک عبارتند از:

* به تعداد ثابتی از نسل‌ها برسیم.
* بودجه اختصاص داده‌شده تمام شود(زمان محاسبه/پول).
* یک فرد(فرزند تولید شده) پیدا شود که مینیمم (کمترین) ملاک را برآورده کند.
* بیشترین درجه برازش فرزندان حاصل شود یا دیگر نتایج بهتری حاصل نشود.
* بازرسی دستی.
* ترکیبهای بالا.

۱۰.۴ روش های نمایش

قبل از این که یک الگوریتم ژنتیک برای یک مسئله اجرا شود، یک روش برای کد کردن ژنوم‌ها به زبان کامپیوتر باید به کار رود. یکی از روش‌های معمول کد کردن به صورت رشته‌های باینری است: رشته‌های ۰و۱. یک راه حل مشابه دیگر کدکردن راه حل‌ها در آرایه‌ای از اعداد صحیح یا اعشاری است، که دوباره هر جایگاه یک جنبه از ویژگی‌ها را نشان می‌دهد. این راه حل در مقایسه با قبلی پیچیده‌تر و مشکل‌تر است. مثلاً این روش توسط استفان کرمر، برای حدس ساختار ۳ بعدی یک پروتئین موجود در آمینو اسیدها استفاده شد. الگوریتم‌های ژنتیکی که برای آموزش شبکه‌های عصبی استفاده می‌شوند، از این روش بهره می‌گیرند [۱۱].

سومین روش برای نمایش صفات در یک GA یک رشته از حروف است، که هر حرف دوباره نمایش دهنده یک خصوصیت از راه حل است.

۱۱.۴ عملگرهای یک الگوریتم ژنتیک

در هر مسئله قبل از آنکه بتوان الگوریتم ژنتیک را برای یافتن یک پاسخ به کار برد به دو عنصر نیاز است:در ابتدا روشی برای ارائه یک جواب به شکلی که الگوریتم ژنتیک بتواند روی آن عمل کند لازم است. در روش سنتی یک جواب به صورت یک رشته از بیتها، اعداد یا نویسه ها نمایش داده می‌شود.دومین جزء اساسی الگوریتم ژنتیک روشی است که بتواند کیفیت هر جواب پیشنهاد شده را با استفاده از توابع تناسب محاسبه نماید. مثلاً اگر مسئله هر مقدار وزن ممکن را برای یک کوله پشتی مناسب بداند بدون اینکه کوله پشتی پاره شود، (مسئله کوله پشتی را ببینید) یک روش برای ارائه پاسخ می‌تواند به شکل رشته ای از بیتهای ۰ و۱ در نظر گرفته شود، که ۱ یا ۰ بودن نشانه اضافه شدن یا نشدن وزن به کوله پشتی است.تناسب پاسخ، با تعیین وزن کل برای جواب پیشنهاد شده اندازه گیری می‌شود [۱۱].

۱۲.۴ ایده ی اصلی

ر دهه هفتاد میلادی دانشمندی از دانشگاه میشیگان به نام جان هلند ایده استفاده از الگوریتم ژنتیک را در بهینه‌سازی‌های مهندسی مطرح کرد. ایده اساسی این الگوریتم انتقال خصوصیات موروثی توسط ژن‌هاست. فرض کنید مجموعه خصوصیات انسان توسط کروموزوم‌های او به نسل بعدی منتقل می‌شوند. هر ژن در این کروموزوم‌ها نماینده یک خصوصیت است. بعنوان مثال ژن ۱ می‌تواند رنگ چشم باشد، ژن ۲ طول قد، ژن ۳ رنگ مو و الی آخر. حال اگر این کروموزوم به تمامی، به نسل بعد انتقال یابد، تمامی خصوصیات نسل بعدی شبیه به خصوصیات نسل قبل خواهد بود. بدیهیست که در عمل چنین اتفاقی رخ نمی‌دهد. در واقع بصورت همزمان دو اتفاق برای کروموزوم‌ها می‌افتد. اتفاق اول جهش [۱۲] است. ”جهش“ به این صورت است که بعضی ژن‌ها بصورت کاملاً تصادفی تغییر می‌کنند. البته تعداد این گونه ژن‌ها بسیار کم می‌باشد اما در هر حال این تغییر تصادفی همانگونه که پیشتر دیدیم بسیار مهم است. مثلاً ژن رنگ چشم می‌تواند بصورت تصادفی باعث شود تا در نسل بعدی یک نفر دارای چشمان سبز باشد. در حالی که تمامی نسل قبل دارای چشم قهوه‌ای بوده‌اند. علاوه بر ”جهش“ اتفاق دیگری که می‌افتد و البته این اتفاق به تعداد بسیار بیشتری نسبت به ”جهش“ رخ می‌دهد چسبیدن دو کروموزوم از طول به یکدیگر و تبادل برخی قطعات بین دو کروموزوم است. این مسأله با نام [۱۳] شناخته می‌شود. این همان چیزیست که باعث می‌شود تا فرزندان ترکیب ژنهای متفاوتی را (نسبت به والدین خود) به فرزندان خود انتقال دهند [۱۱].

[۱۲] Mutation
[۱۳] Crossover

۱۳.۴ الگوریتم ژنتیک ترکیبی بر روی مساله پوشش راسی

یک الگوریتم ژنتیک ترکیبی از ترکیب یک الگوریتم جستجوی محلی [۱۴] با الگوریتم ژنتیک معمولی بدست آمده است [۱۱]. مطالعات نشان می دهد که الگوریتم ژنتیک ترکیبی در اکثر مواقع از الگوریتم ژنتیک معمولی بهتر عمل می کند ([۱۰] ، [۵]).

برای حل یک مساله با الگوریتم ژنتیک لازم است پا رامترهای زیر مشخص شود.

نمایش ژنتیک جواب های شدنی که فرد نامیده می شود، یک جمعیت اولیه، یک هدف یا برازندگی و عملگرهای ژنتیک (انتخاب، تقاطع و جهش).

در الگوریتم ژنتیک در طول تقاطع کروموزم های دو والد بر اساس تابع برازندگی انتخاب می شود. بنابراین مهم ترین قسمت در الگوریتم های ژنتیک عمل تقاطع می باشد. هدف این است که با استفاده ا ز عملگرهای انتخاب ترکیب متفاوتی ساخته شود و بهترین ترکیب انتخاب شود. در اینجا تقاطع احتمال را با احتمال $P_c = ۸۰$ در صد در نظر می گیریم برای هر انتخاب عملگر تقاطع می توانیم از روش انتخاب مسابقه استفاده کنیم. مرحله جهش تنها می توانند برای هر زوج رقم صفر را به یک و رقم یک را به صفر تبدیل کند. و این کار را با احتمال $P_m = ۵$ در صد انجام می دهد. برای ارزیابی تابع برازندگی، مینیمم ارزش تابع را برای هر فرد محاسبه می کنیم. زمانی تابع متوقف می شود که برای ۵۰۰ تولید تغییری حاصل نشود.

۱۴.۴ تقاطع راسی در مساله ی پوشش راسی

فرض کنید که $G = (V, E)$ داده شده باشد که در آن G یک گراف بدون جهت است و V هم مجموعه ای از رئوس می باشد و E هم مجموعه ای از یال ها است. VT و ET به ترتیب جدولی از رئوس و یال ها می باشد. در مساله ی پوشش راسی V یعنی راس ها ژن هایی از کروموزوم ها هستند و P_1 و P_2 دو والد می باشند که برای عمل تقاطع انتخاب می شوند و V' هم جوابی برای پوشش راسی می باشد. در این الگوریتم از الگوریتم HVX

برای عمل تقاطع استفاده می کنیم که این الگوریتم به صورت زیر می باشد [۱۱].

شکل زیر چگونگی کار الگوریتم را نشان می دهد. مقدار دهی کروموزوم ها از طریق انتخاب راس ها به به صورت

[۱۴] local search

begin
$V' =$
creat tables VT and ET
$VT = (F(v), N(v))$, where $F(v)$ is frequency of the vertex v in $P_۱$ and $P_۲$, $N(v)$ is the degree of vertex v in G, for $\forall v \in P_۱$ and $\forall v \in P_۲$
ET=$E(x.y)$ for $\forall E \in G$
while ET<> do
select $v_۱ \in VT$ such that $N(v_۱) > N(v)$ for $\forall v \in VT$. If more than one vertex has same number of degree then select that vertex, whose frequency $F(v_۱)$ is high. If still more than one vertex is candidate for selection then select any vertex randomly . say $v_۱$
$ET = ET - E(x, y) \mid x = v_۱ or y = V_۱$
$V' = V' \cup v_۱$
end while
return V'
end

تصادفی می باشد. و در نهایت جواب نهایی کروموزومی است که مقدار تابع برازندگی بهتری داشته باشد.

۱.۱۴.۴ مثال کاربردی از الگوریتم ژنتیک برای مساله ی پوشش راسی

دو والد $P_۱ = \{۱, ۳, ۴, ۶\}$ و $P_۲ = \{۱, ۲, ۳, ۴\}$ را در نظر بگیرید [۱۱]، اکنون می خواهیم از الگوریتم HVX برای تولید فرزند استفاده کنیم. بنابراین ابتدا بایستی جداول VT و ET را ایجاد کنیم، سپس بر اساس آنها راس v را انتخاب کنیم، سپس تمام یال هایی را که با راس v مجاورند را حذف کنیم، و آن راس را به V' بیافزاییم و این روند را آنقدر ادامه می دهیم که ET تهی شود. این روند گام به گام در جدول های بالا می بینید.

در این بخش می خواهیم تبدیل کروموزوم ها به صفر و یک را از طریق مثال توضیح دهیم. فرض کنید که دو والد به صورت $P_۱ = ۱, ۳, ۴, ۶$ و $P_۲ = ۱, ۲, ۳, ۴$ در اینجا $P_۱$ بدین معنا است که رئوس $\{۱, ۳, ۴, ۶\}$ قسمتی از جواب هستند و رئوس $\{۲, ۵\}$ قسمتی از جواب نیستند، در اینجا چون ما شش راس داشتیم پس طول کروموزوم ها هم شش است (یعنی دارای شش بیت می باشند). عدد موجود رد هر بیت به این بستگی دارد که آیا آن راس در والد وجود دارد یا خیر اگر وجود داشته باشد یک می گذاریم در غیر این صورت صفر قرار می دهیم [۱۱] در این صورت داریم:

$P_۱ = ۱ ۰ ۰ ۱ ۰ ۱$

$P_۲ = ۱ ۱ ۱ ۱ ۰ ۰$

اکنون فرزندی که توسط $P_۱$ و $P_۲$ توسط الگوریتم HVX تولید می شود به صورت $\{۱, ۲, ۳\}$ زیر می باشد. که کد

آن به صورت زیر می باشد:

$$V' = ۱\ ۱\ ۱۰۰۰$$

نرخ جهش در الگوریتم ژنتیک ۵ در صد می باشد یعنی با این احتمال ۵ در صد تعدادی از یک ها تبدیل به صفر می شوند و تعدادی از صفرها تبدیل به یک می شوند.

۱۵.۴ تابع برازندگی الگوریتم ژنتیک برای مساله پوشش راسی

تابع برازندگی نقش بسیار مهمی را در الگوریتم ژنتیک اجرا می کند زیرا به کمک آن می توانیم بهترین کروموزوم را پیدا کنیم. تابع برازندگی برای مساله پوشش راسی به صورت زیر می باشد:

$$F = \sum_{i=۱}^{V} V_i \qquad V_i = ۱,\ if\ V_i \in V'\ \ else\ \ ۰$$

در الگوریتم ژنتیک یک فرزند از کروموزوم های والد خود استفاده می کند. بنابراین معمولا در جمعیت بعدی ۵۰ کروموزوم های والد به فرزند می رسد. لازم به یاد آوری است که تمامی کروموزوم ها برای تولید فرزند از الگوریتم HVX در صد استفاده می کنند. ما معتقدیم که کروموروم ها نقش بسیار مهمی را در تولید فرزند دارند، زیرا به کمک آن ها می توانیم جواب بهینه ی سراسری را بیابیم و در این بین برای اینکه در جواب محلی گیر نکنیم از جهش استفاده می کنیم. در مساله ی پوشش راسی از ساده ترین نوع جهش استفده می کنیم در این نوع جهش تمامی صفر ها به یک و یک ها به صفر تبدیل می شوند. فرض کنید که $P(t)$ جمعیت اولیه باشد و $C(t)$ فرزند تولید شده از این جمعیت باشد و فرض کنید که μ تعداد کروموزوم ها در $P(t)$ باشد.

۱۶.۴ الگوریتم کلی ژنتیک برای مساله ی پوشش راسی

بنابراین الگوریتم ژنتیک برای مساله ی پوشش راسی به صورت زیر می باشد. در صفحه ی بعدی می توانید مقایسه این الگوریتم با سایر الگوریتم ها را ببینید.

Algorithm HGA
begin
t = 0
initialize P(t) randomly
while (t < number of generation) do
evaluate P(t) using fitness function
elitism(best found so far)
select best 50 % chromosomes from P(t) and put them in P(t+1)
m = 0
while (m < μ) do
select m^{th} and $(m+1)^{th}$ chromosome from P(t)
n' = HVX(m, m+1)
n= mutation(n')
place n in C(t)
m = m + 2
end while
apply LOT on C(t) and place them in P(t+1)
t = t + 1
end while
return best found so far from P(no of generation)
end

کتاب‌نامه

[1] E. Angel and R. Campigotto and Christian Laforest, Algorithms for the Vertex Cover Problem on Large Graphs, Springer 12 (2010)233–235.

[2] S. Cook, The Complexity of Theorem-Proving Procedures, Third ACM Symposium on Theory of Computing, ACM, New York. Res. 8 (1971) 151-158.

[3] C. Savage, Depth-first search and the vertex cover problem, Information Processing Letters Res. 14(5) (1982) 233–235.

[4] C. Papadimitriou and M. Yannakakis, Optimization, approximation, and complexity classes, In STOC 88: Proceedings of the twentieth annual ACM symposium on Theory of computing Res. 82 (1988)229–234.

[5] C. H. Papadimitriou and K. Steiglitz, Combinatorial Optimization, Springer, 1982.

[6] R. G. Downey, M.R. Fellows, Parameterized complexity, Springer-Verlag, New York (1999).

[7] M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W. H. Freeman and Co., New Yorkو Springer.7 (1979).

[8] R. Karp, Reducibility Among Combinatorial Problems, Information Sciences, ed. R. E. Miller and J. W. Thatcher. Res. 83 (1972) 85-103.

[9] G. Karakostas, A better approximation ratio for the vertex cover problem, Management Science. 3 (2005) 1043–1050.

[10] S. Khuri and Th. Bäck, An Evolutionary Heuristic for the Minimum Vertex Cover Problem, Information Sciences. 8 (1994) 86-90.

[11] K. Kotecha and N. Gambhava, A Hybrid Genetic Algorithm for Minimum Vertex Cover Problem , Springer 120 (1982) 25-30.

[12] B. Kort and J. Vygen, Combinatiotorial optimization theory and algorithm, springer, 2000.

[13] M. Tuba and R. Jovanovic, An Analysis of Different Variations of Ant Colony Optimization to the Minimum Weight Vertex Cover Problem , Information Sciences 6 (2009) 936-945.