تحقیقی پیرامون :

مسئله فروشنده دوره گرد

Traveling Salesman Problem

جمشید ناظمی

اردیبهشت ۸۰

مقدمه :

بـــرنامه ريزي توليد و عمليات بخش مهمي از مسائل مديريت و تحقيق در عمليات را به خود مشغول نموده است. كتابها و مقالات گوناگون و متنوعي در زمينه برنامه ريزي توليد نوشته شده است .

آنچـــه برنامه ريزي توليد را به مدلهاي رياضي مرتبط با آنها متصل ميكند جايگاه و نقطه اتصال برنامه ريزي به عمليات است. در يك نگرش كلي فرايند برنامه ريزي توليد و عمليات از بازاريابي آغاز و به تحويل و خدمات پس از فروش باز ميگردد و اگر برنامه ريزي را به تعريف عام آن تلقي نماييم در تمام مراحل كاري نياز به برنامه ريزي و يافتن پاسخهاي موجه و بهينه وجود دارد.

در قالب روشها و برخورد كلاسيك با مسائل توليد ، اين فرايند را ميتوان در دو گروه كلي برنامه ريزي كلان و ساليانه و گروه برنامه ريزي خرد و عملياتي تقسيم نمود. از ديدگاه تحقيق در عمليات مسائل گروه اول از طريق مدلهاي برنامه ريزي خطي ، پويا و تكنـــيكهاي حل متناسب با آن پاسخ داده ميشود . در گروه دوم مسائل معمولا با پيچيدگي زياد و همچنين نياز به پاسخ دهي سريع مواجه هستيم و عمده اين گروه از مسائل در قالب مدلهاي زمانبندي[1] و يا توالي عمليات پاسخ داده ميشود.

پيچـــيدگي مدلهاي اين گروه از مسائل معمولا به مواردي بازميگردد كه رفتار به صورت گسسته[2] است و لذا با تغيير يك عامل پرش[3] در خروجي و يا متغيرهاي تصميم گيري ديگر صورت ميگيرد. در مدلهاي توليدي اين حالتها معمولا به موارد مرتبط با راه انـــدازي و يـــا تغيير خطوط تعبير ميشود. روشهاي حل اينگونه مسائل نيز در ادبيات برنامه ريزي عمليات و توالي عمليات متنوع بوده و راهكارهاي مختلفي پيشنهاد شده است.

طيف بزرگي از اينگونه مسائل پس از مدلسازي به يك مسئله مشهور در ادبيات رياضيات باز ميگردد كه به مسئله فروشنده دوره گـــرد مشهور است. روشهاي حل اين گروه از مسائل بدليل پيچيدگي آن در ادبيات اين مسئله به وفور ديده ميشود ولي هنوز راه حـــلهاي بهيـــنه و كـــارا عرضه نشده است. اين تحقيق در صدد است با مرور ادبيات اين مسئله قضايا و روشهاي ابتكاري (و يا حســـي) حـــل را در اختيار بگذارد. ديدگاه كلي در تحقيق انست كه با مرور روشهاي مختلف به يك حالت خاص در اينگونه مســـائل كـــه حالـــت غير متقارن در اينگونه مسائل است و كاربرد زيادي در توليد دارد پاسخ دهد. ضمن انكه براي حالتهاي متقارن نيز پاسخ متناسب در اختيار گذاشته شود.

---

[1] Scheduling

[2] discrete

[3] Jump

مرور ادبیات

TSP بوسیله یک ماتریس عدد صحیح و غیر منفی n*n تعریف میشود که در آن هر توالی P + 1 ، عدد صحیح از مجموعـه اعداد (n ...، ۲ ، ۱) گرفته میشود و با این شرط که هر یک ازn عدد صحیح حداقل یکبار مشاهده شود و اولین و آخرین عدد صحیح یکسان باشد. در این صورت یک تور (Tour) تعریف میشود که آن را میتوان به صورت زیر نوشت :

$$t = (i_1, i_2, i_3, ...i_{p-1}, i_p, i_1)$$

و این تور یک جواب موجه TSP خوانده میشود . از میان تورهای بدست آمده، توری بهینه است که

$$z_{(t)} = \min \sum_{(i,j)\in t'} C_{ij}$$

و t' به صورت زیر تعریف میشود.

$$t' = [(i_1, i_2), (i_2, i_3, ..., (i_{p-1}, i_p), (i_p, i_1)]$$

و هر جفت مرتب نمایشگر مسیری از تور است.

درحالت معمول، n تعداد شهرها یا گره ها را نشان میدهد در حالیکه جفت مرتب (i, j) مسافتی است که گره ها را به هم متصل میکند . همچنین C_{ij} مسافتی است که از گره i تا گره j وجود دارد ویا به عبارت دیگر طول فاصله (i , j) خوانده میشود.

تور t' نیز مسیر بسته ایست که هر گره را حداقل یکبار طی کرده باشد و طول تور- که با $Z_{(t)}$ نمایش میدهیم- مجموع طول کلیه فواصلی است که در تور وجود دارد.

معیار مسافت :

یـک زیـر تـور (Subtour) $S = (i_1, i_2, i_3, ..., i_k, i_1)$ مسـیر بسته ایست که از کلیه گره ها (n گره) نمیگذرد $(k \prec n)$.

بعلاوه باید توجه نمود که برای آنکه مسئله با معنی باشد طول هر زیر تور باید غیر منفی باشد و به عبارت دیگر :

$$Z_{(s)} = \sum_{(i,j)\in s'} C_{ij} \geq 0$$

که S' نمایش جفت های مرتب مجموعه S است.

لذا فرض میشود که طول هر زیر تور غیر منفی است و این در صورتی ممکن است که :

$C_{ij} \geq 0$ برای کلیه ij

علـیرغم آنکه ظاهرا هیچ گونه محدودیتی جز غیر منفی بودن برای C_{ij} وجود ندارد ولی محدودیتهایی روی معیار مسافت وجود دارد که منجر به روشهای محاسباتی کارآ شده است.

تئوریهای موجود در TSP

در حـال حاضـر تـئوریهای کافـی برای TSP وجود ندارد. منظور اینست که حل TSP در حالت کلی نمیتواند به صورت کارآ همچون مسائلی چون کوتاهترین مسیر انجام شود. در عین حال تئوریهایی وجود دارد که این امکان را بوجود آورده است تا الگوریتم هایی برای حل مسئله بوجود آید. لذا تئوریهایی که در این زمینه وجود دارد بدون اثبات بیان میکنیم.

قضیه یک :

اگـر ماتـریس فواصـل C رابطه نامساوی مثلث را راضی نماید در آنصورت یک تور بهینه وجود دارد که هر گره فقط یکبار و فقط یکبار در آن وجود دارد. توجه میشود که اگر C_{ij} را با C'_{ij} تعویض نماییم در حالیکه C'_{ij} کوتاهترین مسیر i بـه j باشـد در این صورت ، نامساوی مثلث راضی میشود. بنابراین یک مسئله که در آن هر شهر فقط یکبار ملاقات میشود در صورتی قابل حل خواهد بود که C با C' تعویض شود. حال اگر (i_p, i_q) یک فاصله در تور بهینه با ماتریس C' باشـد و همچنین توالی $(i_p, i_s, i_1, ..., i_q)$ کوتاهترین مسیر از i_p به i_q تحت ماتریس C باشد در آنصورت یک تور بهینه تحت ماتریس C وجود دارد که شامل توالی $(i_p, i_s, i_1, ..., i_x)$ است.

این نتیجه مهم است زیرا کلیه الگوریتم های طرح شده برای TSP بدین منظور هستند که کوتاهترین تور را که از هـر گـره فقـط یکبار گذشته باشد بدست آورند. بنابراین فرض میکنیم که C به نحوی انتخاب شده است که یک تور بهینه وجود داشته باشد که در آن هر گره فقط یکبار عبور شده باشد یا مسئله بدست آوردن تور بهینه تحت این محدودیت باشد. لازم بـه تاکید است که قضیه یک محکمتر خواهد بود وقتی C معیار مسافت اقلیدسی را در فضای دو بعدی راضی نماید. لذا همواره فرض میکنیم که گره ها بوسیله نقاطی در فضای دوبعدی تعریف شوند به نحویکه فاصله بین گرههای i و j برابر C_{ij} باشد.

قضیه دوم :

فرض کنید G مجموعه محدب (Convex) از کلیه نقاط در فضای اقلیدسی و دوبعدی باشد در آنصورت یک تور بهینه وجود دارد که در آن ترتیب نسبی نقاط روی گوشه های G پیش بینی شده است.

قضیه سوم :

یک تور بهینه وجود دارد که خود را قطع نمیکند و آن در صورتی است که معیار مسافت اقلیدسی توسط C ارضا شـده باشد. قضیه دوم و سوم کلیه تورهایی که این خصوصیات را ندارند از بررسی خارج میکنند. الگوریتم های بسیاری را بوسیله استفاده از این دو قضیه میتوان بهبود داد و برای مسائل متقارن که لزوما اقلیدسی نباشند میتوان با قضیه بعد(یا قضیه چهارم) نیمی از تورها را حذف کرد.

قضیه چهارم :

اگر C متقارن باشد و t_1 و t_2 دو تور باشند که در آن گره ها با ترتیب عکس هم ملاقات شده اند یعنی :

$$t_1 = (i_1, i_2, i_3, ...i_n, i_1) \qquad t_2 = (i_1, i_n, ...i_3, i_2, i_1)$$

آنگاه مسافت دو تور با هم برابر خواهد بود یا : $z(t_1) = z(t_2)$.

لـذا اگرچه در حالت کلی (n-1)! تور وجود دارد در مسائل متقارن فقط کافی است که $\frac{(n-1)!}{2}$ تور مورد بررسی قرار گیرد.

قضـیه هـای دیگـری نـیز وجـود دارنـد کـه برای پیش بینی حدود روی طول تور بهینه مفید هستند از جمله مسئله تخصیص (Assignment Problem) که نسبتا حل آن آسان است و به صورت زیر تعریف میشود :

$$\min w = \sum_{i=1}^{n}\sum_{i=1}^{n} C_{ij} X_{ij}$$

$$\sum_i X_{ij} = 1 \qquad \forall j$$

$$\sum_j X_{ij} = 1 \qquad \forall i$$

$$(X_{ij} = 0,1 \forall i, j)$$

قضیه پنجم :

فرض کنید t هر توری باشد که گره ها دقیقا یکبار ملاقات شده باشد و ($X_{ij} = 0$ در صورتیکه (i,j) در تور نباشـد) و $X_{ij} = 1$ اگـر (i,j) در تـور باشـد آنگـاه X_{ij} یـک جـواب موجـه بـه مسـئله تخصـیص بـا $z(t) = \sum_i \sum_j C_{ij} X_{ij} = w$ است.

متاسفانه عکس این مطلب صحیح نیست. جواب موجه به مسئله تخصیص ممکن است تور نباشد. برای مثال به ازای n=4 یـک جـواب موجه مسئله تخصیص $X_{12} = X_{21} = X_{34} = X_{43} = 1$ خواهد بود اما تفسیر آن در ارتباط با TSP بدین معنی است که زیرتورهای (۱و۲و۱) و (۳و٤و۳) انتخاب شوند.

بنابر این مطابق قضیه پنجم اثبات میشود که به ازای کلیه تورها رابطه $\min w \le z(t)$ صادق میباشد.

قضیه ششم :

اگر K_p و K_q اعداد حقیقی و مبین یک جفت گره p و q باشد به نحویکه :

اگر j = 1 , ... , n j ≠ (q $C'_{pj} = C_{pj} - K_p$

و اگر i = 1 , ... , n i ≠ (p $C'_{iq} = C_{iq} - K_q$

و $C'_{pq} = C_{pq} - K_p - K_q$

و در غیر اینصورت $C'_{ij} = C_{ij}$

و $Z'(t)$ طول تور t تحت C' باشد آنگاه $Z(t) - K_p - K_q = Z'(t)$

قضیه ٦ این امکان را میدهد که با ماتریس کاهش یافته با عناصر صفر کار کرد و کاهش مقادیر در تابع هدف تاثیر گذاشته ودر جواب تفاوتی ایجاد نمیکند.

قضیه هفتم :

اگر تور t شامل فاصله (p , q) نباشد آنگاه $Z(t) \geq h_p + h_q$ که در آن

$$h_p = \min_{p\,,\,q \neq j} C_{pj}$$

$$h_q = \min_{p\,,\,q \neq i} C_{iq}$$

قضیه هشتم :

فرض کنید p_1 و p_2 دو ترکیب از اعداد صحیح (۱ + K و .... و۳ و۲و۱) باشد.

$$p_1 = (i_1, i_2, \ldots\ldots, i_k) \qquad p_2 = (j_1, j_2, \ldots\ldots\ldots j_k)$$

و فرض کنید

$$Z(p_m) = \sum_{(i,j) \in p'_m} C_{ij} \qquad m = 1, 2$$

که در آن P'_m نمایش جفت مرتب مربوط به P_m است.

آنگاه اگر

$$C_{1i_1} + Z(p_1) + Ci_k s \langle C_{1j_1} + Z(p_2) + Cj_k s$$

در اینصورت $p_2(1\,,\ ,\ s\ )$ نمیتواند بخشی از یک تور باشند.

با اعمال قضیه ۸ به صورت رابطه برگشت میتوان معادلات برای تعیین تور بهینه بدست آورد.

قبلا خاطر نشان ساختیم که کلیه تورها جواب موجه به مسئله تخصیص هستند اما علاوه بر تورها، زیرتورها نیز جواب موجه مسئله تخصیص هستند. از آنجا که مسئله تخصیص برنامه ریزی خطی از نوع صفر ـ یک است اگر محدودیتهایی را بتوان اعمال نمود که زیرتورها را حذف نماید آنگاه TSP را میتوان به صورت برنامه ریزی خطی صفر ـ یک نوشت. قضیه نهم و دهم این کار را انجام میدهند.

قضیه نهم :

فرض کنید S و S' بخشی از اعداد صحیح i = 1 , ... , n باشد و $S \cap S' = \phi$.در صورتیکه فواصل متقارن باشد ، فرض کنید X_{ij} به صورت زیر تعریف شود :

اگر فاصله i , j (i , j) در تور نباشد	0
اگر فاصله i , j (i , j) در تور باشد	1

در اینصورت تور بهینه را میتوان با حل برنامه ریزی عدد صحیح زیر بدست آورد :

$$\min Z = \sum_{j=2}^{n} \sum_{i=1}^{j-1} C_{ij} X_{ij}$$

S.T $\quad X_{ij} = 0,1 \quad \forall(i = 1,....,j-1)$

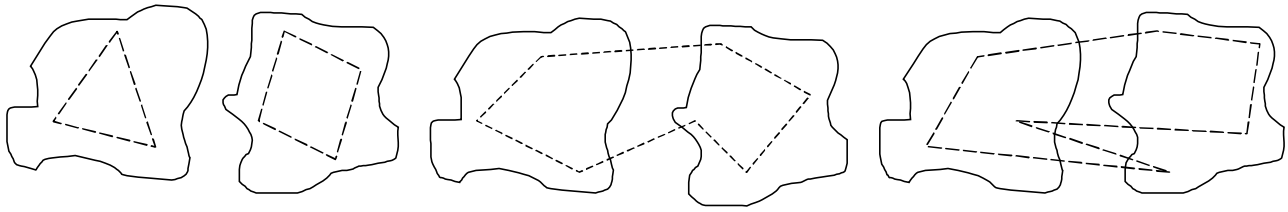
$(j = 2,\ldots,n)$

ودارای محدودیت حلقه زیر باشد:

$$\sum_{i\in S}\sum_{j\in \bar{S}} X_{ij} \geq 2$$

در مدل فوق این مجموعه روابط برای کلیه حالتهایی است که $(S,\bar{S})$ غیرتهی باشند و اگرمجموعه $(S,\bar{S})$ مورد بررسی قرار گیرد انگاه بایستی $(S,\bar{S})$ در نظر گرفته نشود.

برای مشخص شدن رابطه محدودیت حلقه، میتوان از شکلهای زیر وجود بیش از دو رابطه را بین دو زیر مجموعه جدا از هم مشاهده نمود.



الف ب ج

الف) شامل یک زیر تور است ب و ج) زیر تور ندارند

همانطور که ملاحظه میشود تعداد $(2^{n-1}-1)$ از این محدودیت ها برای مسئله n شهر وجود دارد.ودر مسائل غیرمتقارن به تعدادی دو برابر متغییر نیاز وجود دارد.

قضیه دهم :

فرض کنید $X_{ij} = 1,(0)$ در صورتیکه فاصله (i,j) در تور باشد (ونباشد).

تور بهینه با حل برنامه ریزی عدد صحیح زیر بدست می آید :

$$\min Z = \sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij} X_{ij}$$

$$\sum_{i} X_{ij} = 1 \forall i$$

$$\sum_{j} X_{ij} = 1 \forall j$$

$$u_i - u_j + n x_{ij} \leq n-1 \qquad \forall i,j \qquad i \neq j$$

این فرموله کردن تقریبا n^1 محدودیت دارد که از محدودیت های قضیه ۹ کمتر است. اما اینگونه فرموله کردن بدین معنی نیست که این مدل راحتتر حل میشود.

قضیه یازدهم :

با فرض آنکه گره ها را $(i,1,.....n)$ و فواصل را (i,j) تعریف نماییم و ماتریس فاصله c در شبکه به معنای فاصله گره از مبدا شبکه باشد و همچنین اگر یک گره که به α نمایش داده میشود وجود داشته باشد و فاصله (j,α) را برای هر j که (j,1) یک فاصله در شبکه اولیه باشد در نظر بگیریم آنگاه فاصله d_{ij} در شبکه جدید به صورت زیر خواهد بود.

$$d_{ii} = 0 \qquad \forall_i$$

$$d_{j1} = -\infty \qquad \forall j, j \neq 1$$

$$d_{j\alpha} = k - c_{j1} \qquad \forall j, j \neq \alpha$$

$$d_{ij} = k - c_{ij} \qquad \text{ودر غیر اینصورت}$$

و با این فرض که اگر kهرعدد محدودی باشد و مقدار آن از شرط زیر پیروی نماید:

$$k \succ c_{ij} \text{ بزرگترین } \text{مجموع n}$$

در این صورت طولانی ترین مسیر از ۱ به α در شبکه جدید شامل گره های میانی (2,…..,n) خواهد بود و اگر $(1, i_1, \ldots\ldots, i_{n-1}, \alpha)$ این مسیر طولانی باشد در آنصورت $(1, i_1, \ldots\ldots, i_{n-1}, 1)$ تور بهینه خواهد بود.

روشهای حل ***TSP***

روشهای حل TSP کلا به سه بخش عمومی تقسیم میشوند :

الف) نقطه شروع

ب) روش تولید جواب

ج) قاعده ختم

اگـر قاعده ختم کار بدینصورت باشد که توقف محاسبات در صورتی است که تور بهینه بدست آمده باشد روش حل را دقیق (Exact) می خوانیم. اگر قاعده ختم به نحوی است که فقط با بدست آمدن تور بهینه متوقف نمی شود روش تخمینـی خواهـد بـود. در روش هـای تخمینـی تور بدست آمده در خاتمه حل مسئل به نقطه شروع بستگی دارد. لذا معمولا تعدادی از این تورها تولید میشود و سپس بهترین انتخاب میشود.

تعاریف فوق را میتوان با در نظر گرفتن دو قاعده زیر توضیح داد.

۱) اگر یک تور بدست آید که $z(t^{\circ}) = L$ و در آن L حد پایین کلیه تورها باشد متوقف شوید.

۲) اگـر یک تور t^* پیدا شود که برای کلیه تورهایی که بوسیله تعویض دو عنصر t^* ایجاد میشوند $z(t^*) \le z(t)$ باشد ، آنگاه متوقف شوید.

حالـت ۱ روش دقـیق اسـت یعنـی t° جـواب بهینه است در صورتیکه t^* جواب بهینه محلی است. از آنجا که بیشـتر نقاط شروع و قواعد ختم محاسبه بستگی به روش تولید و جواب دارد برحسب روش تولید جواب آنها را طبقه بندی میکنیم.

- بهبود تور به تور

نقطـه شـروع یـک تور دلخواه است مثلا $t_{\circ} : (1,2,3,.....,n,1)$. قاعده تولید جواب بر مبنای بدست آوردن تور بهتر که در همسایگی تور کنونی است تعریف میشود. در این روش میتوان از یک $t_{\circ}$ جدید شروع کرد و روش قبل را دنبال نمود یا روشهای پیچیده تری را اعمال نمود.

- ساختن تور

نقطه شروع یک گره دلخواه مثلا i_1 است از i_1 یک توالی $(i_1, i_2, ...i_k)$ بوسیله اضافه کردن متوالی گره های دیگـر به گره اولیه بدست می آید. روش هنگامیکه به یک تور برسد ختم می یابد. روش ساختن تور هم به صورت دقیق و هم به صورت تخمینی وجود دارد.

- حذف زیر تور

نقطـه شـروع در ایـن حالـت ، مسئله تخصیص است اگر جواب مسئله تخصیص خود یک تور باشد جواب TSP بدسـت آمـده اسـت. در صـورتیکه مسئله تخصـیص تـور نباشـد، روش حذف زیر تورها انتخاب میشود. روش دقیق حذف زیرتورها، برنامه ریزی خطی عدد صحیح ، روش شاخه و تجدید است.

روش دیگری که میتوان الگوریتمهای حل TSP را تقسیم نمود براساس دقیق یا ابتکاری (تخمینی) بودن الگوریتم حل است که در واقع تقسیم الگوریتها برحسب نزدیکی به تور بهینه است.

الگوریتهای دقیق برای مسئله فروشنده دوره گرد **TSP**

- برنامه ریزی دینامیک

اصل بهینه بودن که در این مورد بکار می رود بر مبنای قضیه ۸ است بطور مشخص اگر $f_{k-1}(i_m|i_1,.....,i_{m-1},i_{m+1},.....i_{k-1})$ طول کوتاهترین مسیر باشد که از گره یک شروع میشود و از $(i_1,.....,i_{m-1},.....i_{k-1})$ میگذرد و در گره i_m ختم می یابد. از قضیه ۸ نتیجه میگیریم که کوتاهترین تور جزیی از گره ۱ به گره j است و از $i_1,....,i_{k-1}$ میگذرد و از تابع زیر معین میگردد.

$$f_2(j|i_1,.....,i_{k-1}) = \min_{m=1,.....,k-1}[\, f_{k-1}(i_m|i_1,.....,i_{m-1},i_{m+1},.....,i_{k-1}) + c_{imj}]$$

با بکارگیری معاملات فوق به صورت رابطه برگشت، به معادله زیر ختم میشود.

$$f_2(j|i_1) = c_1 j_1 + c_{i_1} j \qquad \forall i_1, j \neq 1, i_1 \neq j$$

و ختم تور بهینه با حل روابط زیر بدست می آید.

$$f_n(1|i_1,.....,i_{n-1}) = \min_{m=1,....,n-1}[\, f_{n-1}(i_m|i_1,.....,i_{m-1},i_{m+1},.....i_{n-1}) + c_{i_{m_i}}]$$

پیچیدگی حل این مسئله بر روی کامپیوتر، حجم حافظه مورد نیازاین روش است. برای محاسبه f_{k+1} ، باید کلیه مقادیر f_k روی حافظه موجود باشد آنگاه که f_{k+1} محاسبه شود f_k را میتوان حذف و پاک نمود. بعلاوه با افزایش زمان محاسبه میتوان عناصر f_{k+1} را در حافظه کمکی قرار داد و در محاسبه f_{k+1} به حافظه اصلی عودت داد. مقدار حافظه مورد نیاز برای f_k برابر است با :

g(n,k)=(n-1)!/(k-1)!(n-k-1)!

فرض کنید p مرحله گلوگاه باشد با محاسبه تابع f_p به صورت مستقل برای زیر مجموعه کلیه ترکیبات مقادیر تابع ، حجم حافظه مورد نیاز کاهش می یابد. این روش به منظور محاسبه فقط آن مقادیراز f_k بکار می رود (k=1 , ... , p-1) که در تعیین f_p برای مقادیر ثابت ($i_2,.....i_p$) و کلیه n-p مقادیر j لازم است. بدین معنی که هنگامیکه n-p مقدار f_p معین باشد بار دیگر با f_2 شروع شده و n-p مقدار دیگر محاسبه میشود و لذا هنگامیکه مقادیر f_p موجود باشد (n-p-1) مقدار برای f_{p+1} محاسبه میشود.

برای مسائل متقارن، با n زوج، صرفا لازم است که $f_{\frac{n}{2}}$ محاسبه شود یعنی :

$$f_{\frac{n}{2}}(j|i_1,.....,i_{\frac{n}{2}-1}) + f_{\frac{n}{2}}(i_{\frac{n}{2}+1},.....,i_{n-1})$$

این روابط طول یک تور را مشخص میکند که در بین تمام تورهایی که از گره ۱ و از طریق $(i_1,.....,i_{\frac{n}{2}-1})$ میگذرد و سپس از گره j به گره ۱ از طریق $(i_{\frac{n}{2}+1},.....,i_{n-1})$ میگذرد کوتاهترین مسیر میباشد. بدین ترتیب با جمع کردن مناسب $f_{\frac{n}{2}}$ به مقدار مجموع دوبار تور بهینه بدست می آید.

- برنامه ریزی عدد صحیح

مشکل پیدا کردن تور بهینه در حل برنامه ریزی عدد صحیح قضیه ۹ به دلیل تعداد حلقه های بیشمار $(2^{n-1}-1)$ و این مطلب که $(n^2-n)/2$ متغیرهای x_{ij} در مسائل متقارن باید صفر ـ یک شوند، وجود دارد.

حل یک برنامه ریزی خطی با محدودیت های حلقه و $0 \le x_{ij} \le 1$ لزوما شرط ۱ یا $x_{ij}=0$ را ارضاء نمیکند.

در سال ۱۹٥٤ ، Dantzig , Fulkerson , Johnson یک جواب بهینه برای مسئله ٤۲ شهر با استفاده از این فرمول پیدا کردند. آنها با مسئله تعداد زیاد محدودیتها بدین صورت برخورد نمودند که با تعداد محدودی شروع کردند و سپس تعداد دیگر را زمانیکه به آن برای بلوکه کردن زیر تورها نیاز بوده است اضافه نمودند سرانجام ادعا شد که برای مسائل موجود، یک برنامه ریزی خطی معمولی میتواند مقادیر عدد صحیح به x_{ij} بدهد. این محدودیت توسط صفحه های برشی (Cutting Plane) اعمال شد (Gomory) .

بعد از Gomory ، افراد دیگری چون Zelnan , Tucker , Miller در سال ۱۹٦۰ با استفاده از صفحه های برشی و فرمول قضیه ۱۰ کار کردند ولی به نتایج خوبی نرسیدند.

این مورد تا سال ۱۹٦٦ ادامه داشت که Martin گزارش حل مسئله ٤۲ شهر Dantzig را بوسیله برنامه ریزی خطی عدد صحیح در کمتر از ٥ دقیقه روی IBM گزارش داد.

نتیجه بدست آمده مارتین برعکس میلر بدین دلیل بوده است که از محدودیت های مسئله ۹ استفاده نموده است زیرا محدودیتهای مسئله ۹ برای خارج کردن جوابهای کسری مناسبتر هستند. برای مثال فرض کنید زیرتورهای (۱ و ۳ و ۲ و ۱) و (٤و n و.....و٥و٤) جواب باشند این تورها را میتوان با محدودیت قضیه ۹ خارج نمود(بلوکه نمودن زیر تور) بدینصورت که S={1,2,3}

$$x_{12}+x_{23}+x_{31} \le 2$$

در حالیکه برای قضیه ۱۰ این محدودیت ها لازم است :

$$u_1-u_2+n\,x_{12} \le n-1$$
$$u_2-u_3+n\,x_{23} \le n-1$$
$$u_3-u_4+n\,x_{31} \prec n-1$$

که با جمع کردن آنها میتوان رابطه زیر را بدست آورد :

$$x_{12} + x_{23} + x_{31} \le 3 - \frac{3}{n}$$

اگـرچه این محدودیت کافی است تا زیر تور را بلوکه نماید ولی ضعیفتر از محدودیت مسئله ۹ است و علیرغم این مطلـب کـه فرمولـه کـردن قضـیه ۹ بـه محدودیـت های بیشتری احتیاج دارد معمولا تعداد کمی از آنها مورد استفاده قرار میگیرد.

- **شاخه و تحدید**

الگوریـتم شـاخه و تحدیـد بوسـیله Shapiro , Little , Eastman توسـعه یافته است. روش Little الگوریتم سازنده است در حالیکه روشهای Shapiro , Eastman الگوریتم حذف زیر تور است.

الگوریـتم توسـعه یافته توسط Eastman و بهبود یافته توسط Shapiro یک روش جستجو است که یک تور را به زیر تور تقسیم میکند و حد پایین روی کلیه زیرتورها را بدست مي آورد. حد اولیه توسط روش تخصیص بدست می آید. این حد مقدار بدست آورده از حل مسئله تخصیص است.

اگـر جواب بدست آمده موجه باشد در آنصورت یک زیر تور که تعدادk فاصله دارد به k شاخه تقسیم میشود و اگـر یـک زیر تور به صورت $(i_1, i_2, \ldots\ldots, i_k, i_1)$ باشد آنگاه در شاخه ۱ ، $c_{i_1 i_2} = \infty$ و برای شاخه ۲ $c_{i_1 i_2} = \infty$ و .....و بدین ترتیب در این حالت فاصله های (i_1, i_2) ، ..... را حذف میکند. Shapiro شاخه ای را انتخاب میکند کـه کوچکتریـن k را داشـته باشـد ایـن روش ظاهـرا خوب است ولی لزوما بهترین انتخاب نیست. در این حالت k برنامه تخصیص جدید حل میشود و حد پایین کلیه تورها را در زیرمجموعه های خود بدست میدهد. اگر هر یک از این k راه حل تورهایی باشد و اگر هزینه یکی از این تورها کوچکتر یا مساوی حد پایین در دیگر زیر مجموعه ها باشد آنگاه آن تور بهینه اسـت. در غـیر اینصـورت زیـر مجموعـه ای که حد پایین تری دارد انتخاب و شاخه ای در آن ایجاد میشود و به هر حال میتوان مطمئن بود که تور بهینه بدست آید.

Shapiro در مـورد مسـائل مـتقارن وجود مشکل را گزارش میدهد. به طور مشخص، تعداد زیرتورهایی با طول ۲ زیاد است. برای این منظور او روش دیگری را انتخاب میکند. وی از فرمول قضیه ۹ برای این منظور استفاده میکند و برای مقدار اولیه از n محدودیت استفاده میکند تا در آن n حالتیکه مجموعه s شامل یک گره میباشد را حذف نماید . این عمـل سـبب میشـود که جواب زیر تورها شامل طول ۲ نباشد، دنباله روش مانند روش قبل خود وی دنبال میشود و در ادامه محدودیت های قضیه ۹ اعمال نمی شود.

الگوریـتم توسـعه یافـته توسـط Little تاکتیک دیگری را برای شاخه و تحدید استفاده میکند. محاسبه حدود بر مبـنای قضـیه ٦ و ۷ است و تحت عنوان کاهش ماتریس مشهور است. ماتریس کاهش یافته برای شاخه زدن و تبدیل تور به دو زیـر مجموعه مورد استفاده قرار میگیرد. این روش با قبول یک فاصله در تور و عدم قبول آن در زیر مجموعه دیگر انجام می پذیرد.

روش حـل دقیق TSP را به همین جا ختم میکنیم اما خاطر نشان میکنیم که میتوان مسائل بزرگ را با اعمال روش تقسیم کردن کوچک نمود.

تصور کنید که از گره های $(i_1,.....,i_{n-k}) \in s$ باید با یک ترتیب دلخواه عبور نمود. آنگاه n گره اولیه مسئله را میتوان به مسئله k+1 گره تبدیل نمود به عبارت دیگر کلیه گره های اولیه در $\bar{s}$ اضافه یک گره اضافه θ ،گره های مسئله را مشخص میکند. در این حالت c' به صورت زیر تعریف میشود :

$c'_{ij} = c_{ij} - - - - - - i_{ij} \in \bar{s}$ $\qquad$ $c'_{\theta j} = c_{i_{n-k}} - - - - - - j \in \bar{s}$ $\qquad$ $c'_{i\theta} = c_{ii_1} - - - - - - i \in \bar{s}$

اگر یک تور بهینه تحت $(i_\theta, i_{n-k+1},.....,i_n)c'$ باشد مسئله اولیه تور بهینه $(i_1,.....,i_{n-k}, i_{n-k+1},.....,i_n)$ را خواهد داشت.

الگوریتمهای ابتکاری (حسی) برای مسئله فروشنده دوره گرد TSP

روشهای زیادی برای حل TSP عرضه شده است که برحسب مورد هر یک قابلیت بیشتری نسبت به دیگر روشها دارد. لـذا در صورتیکه یک جواب تخمینی (و نه بهینه) مورد انتظار باشد میتوان یکی از روشهای سریع ساخت تور را مورد استفاده قـرار داد و بـرای نـتایج دقیقتر روشهای ترکیبی مناسبتر خواهد بود. به عنوان یک بررسی اجمالی این روشها را به صورت دسته بندی شده ارایه میکنیم.

روشهای ساخت تور Tour Construction Procedures

- الف) نزدیکترین همسایگی

روش کار :

گام یک : با هر گره دلخواه شروع کیند

گام دوم : نزدیکترین گره به آخرین گره مسیر انتخاب شده را پیدا کنید این گره را به مسیر اضافه کنید.

گام سوم : گام دوم را آنقدر تکرار کنید که کلیه گره ها در مسیر واقع شوند، آنگاه گره اول و آخر را به هم متصل کنید.

رفتار در بدترین حالت :

$\frac{1}{2}[\log_2(n)] \leq$ طول تور بدست آمده / طول تور بهینه

تعداد محاسبات :

این روش به میزان n^2 محاسبه احتیاج دارد.

نتیجه :

در تجزیه و تحلیل محاسباتی اگر این روش را n بار با انتخاب هر یک از گره ها به عنوان اولین گره تکرار کنید در اینصورت تعداد محاسبات n^3 خواهد بود.

- ب) صرفه جویی کلارک و رایت **Clarke and Wright Saving**

روش کار :

گام یک : هر گره دلخواه را به عنوان گره مرکزی شماره یک انتخاب کنید.

گام دوم : صرفه جویی را محاسبه کنید $s_{ij} = c_{i1} + c_{1j} - c_{ij}$ i,j = 2,3,4,……,n

گام سوم : صرفه جویی را از بزرگ به کوچک ردیف کنید

گام چهارم : با اولین گره در لیست صرفه جویی شروع کنید و زیر تورهای بزرگتر را با چسباندن گره های i و j به آن ایجاد کنید تا تور کامل شود.

رفتار در بدترین حالت :

تعداد دقیق آن مشخص نیست ولی بطور کلی میتوان گفت نسبت بدترین حالت تابعی خطی از Log(n) است.

تعداد محاسبات :

محاسبه ماتریس B در گام دوم به میزان cn^2 عملیات احتیاج دارد که مقدار C را برابر Log(n) برآورد کرده اند.

نتیجه :

اگر این روش با انتخاب هریک ازگره ها به عنوان گره اولیه تکرار شود Log(n) n^3 محاسبه انجام خواهد شد.

- ج) روش های جایگذاری **Insertion Procedures**

در روش جایگذاری یک زیر تور با k گره در مرحله k ام محاسبات در نظر گرفته میشود و سعی میشود گره ای که در زیر تور موجود نیست به آن متصل شود و در این رابطه محلی که باید این گره جایگذاری شود تعیین میگردد. بنابراین در این روش دو گام انتخاب و جایگذاری را در روشهای مختلف زیر تشریح میکنیم و بدترین رفتار روش را بیان میکنیم.

- ۱- ج) نزدیکترین جایگذاری **Nearest Insertion**

روش کار :

گام یک : با یک زیر گراف که فقط شامل گره i باشد شروع کنید

گام دوم : گره k را به نحوی پیدا کنید که c_{ik} مینیمم باشد و زیر تور i - k - i را تشکیل دهید

گام سوم : گام انتخاب ـ با زیر تور موجود شروع کنید و گره k را پیدا کنید که در زیر تور نباشد و نزدیکترین گره نسبت به هر یک از گره های زیر تور باشد .

گام چهارم : گام جایگذاری ـ فاصله (i,j) را در زیر تور پیدا کنید که مقدار $c_{ik} + c_{kj} - c_{ij}$ را مینیمم کند. K را بین دو گره i و j قرار دهید.

گام پنجم : به گام سوم بروید مگر آنکه یک تور کامل(یا همیلتن) یافته باشید.

بدترین رفتار :

≤ 2 طول تور بدست آمده / طول تور بهینه

تعداد محاسبات :

این الگوریتم به n^2 محاسبه احتیاج دارد

- ۲- ج) کم هزینه ترین جایگذاری **Cheapest Insertion**

روش کار :

مانند روش نزدیکترین جایگذاری عمل میشود و فقط گام سوم و چهارم آن با گام زیر تعویض میگردد

گام سه : با هر(تمام موارد) k که در زیر تور نباشد شروع کنید و (i,j) را به نحوی پیدا کنید که $C_{ik}+C_{kj}-C_{ij}$ حداقل شود و آنگاه k را بین دو گره i و j قرار دهید.

بدترین رفتار :

مانند روش نزدیکترین جایگذاری

تعداد محاسبات :

این الگوریتم به n^2 Log(n) محاسبه احتیاج دارد

- ۳- ج) جایگذاری دلخواه **Arbitrary Insertion**

روش کار :

مانند روش نزدیکترین جایگذاری عمل میشود با این تفاوت که در گام سوم هر گره k که در زیر تور نباشد بطور دلخواه انتخاب میشود تا وارد تور شود.

بدترین رفتار :

طول تور بدست آمده / طول تور بهینه $\leq 2Ln(n)+0.16$

تعداد محاسبات :

این الگوریتم به n^2 محاسبه احتیاج دارد

- ٤- ج) دورترین جایگذاری **Farthest Insertion**

روش کار :

مانند نزدیکترین جایگذاری عمل میشود با این تفاوت که در گام سوم، "نزدیکترین به" با "دورترین از" تعویض میشود و در گام دوم "حداقل" با "حداکثر" جایگزین میشود.

بدترین رفتار :

مانند روش نزدیکترین جایگذاری

تعداد محاسبات :

مانند روش نزدیکترین جایگذاری

- ٥- ج) پوسته محدب **Convex Hull**

نشان داده شده است که اگر هزینه های C_{ij} از روابط اقلیدسی پیروی کنند و H پوسته محدب گره ها در فضای دوبعدی باشد آنگاه توالی گره هایی که در مرز H ظاهر میشوند در تور بهینه نیز از ترتیبی که در H ظاهر شده اند پیروی میکنند. این روش سرعت و دقت قابل ملاحظه را نشان داده است.

روش کار :

گام یک : پوسته محدب را از یک مجموعه گره ها تشکیل دهید. این پوسته یک زیر تور اولیه را ارایه میکند.

گام دوم : برای هر k که هنوز در زیر تور وجود ندارد مشخص کنید که بین کدام دو گره i و j قرار گیرد. بدین معنی که برای هر k ، (i,j) را پیدا کنید که $C_{ik}+C_{kj}-C_{ij}$ حداقل شود.

گـام سـوم : کلـیه (i,k,j) هـای بدسـت آمده در گام دوم را تشکیل دهید و تعیین کنید برای کدام (i^*, k^*, j^*) مقدار $\frac{c_{i^*k^*}+c_{k^*j^*}}{c_{i^*j^*}}$ حداقل میشود.

گام چهارم : گره k^* را بین i^* و j^* قرار دهید.

گام پنجم : به گام دوم بروید و این عمل را آنقدر تکرار کنید تا یک تور همیلتن پیدا شود.

تعداد محاسبات :

این روش مانند روش کم هزینه ترین جایگذاری به n^2 Log(n) محاسبه احتیاج دارد.

- د) روش حسی کریستو فاید **Cristofide's Heuristic**

این روش برای حل TSP هایی بکار میرود که ماتریس هزینه از نامساوی مثلث پیروی می نماید.

روش کار :

گام یک : درخت پوششی T (Spanning Tree) که در مجموعه G حداقل باشد را پیدا کنید.

گام دوم : کلیه گره های درجه فرد در T را مشخص کنید.

حـال یـک مسـئله (Minimum Cost Perfect Matching) را روی گـره هـای درجـه فرد با استفاده از ماتریس هزینه ابتدایی حل کنید. شاخه های بدست آمده از حل مسئله را به شاخه های موجود در T بیفزایید تا یک حلقه اولر بدست آید. در این زیر گراف هر گره دارای درجات زوج میباشد و بعضی از گره ها درجاتی بیش از ۲ خواهند داشت.

گام سوم : چند ضلعی هایی که با گره های درجات بیشتر از ۲ بوجود می آیند بردارید و حلقه اولر را به حلقه همیلتن تبدیل نمایید.

بدترین رفتار :

$\leq 1/5$ طول تور بدست آمده / طول تور بهینه

تعداد محاسبات :

چـون در این روش گلوگاه حل مسئله Matching است بنابراین به $o(n^3)$ عملیات احتیاج دارد و در اغلب اوقات تعداد گره هایی با درجات فرد خیلی کمتر از n است.

روشهای بهبود تور **Tour Improrement Procedures**

یقیـنا بهترین روشهای حسی شناخته شده برای TSP ، روشهای تعویض شاخه است. روشهای حسی 2 - opt و 3 - opt که به وسیله Lin معرفی شده اند بر این مبنا استوار هستند.

روش کار :

گام یک : یک تور ابتدایی پیدا کنید. معمولا این تور به صورت تصادفی از مجموعه کلیه تورها انتخاب میشود.

گام دوم : تور را با استفاده از یکی از روشهای تعویض شاخه بهبود دهید.

گام سوم : گام دوم را آنقدر تکرار کنید که بهبود دیگری حاصل نشود.

بدترین رفتار :

با توجه به آنکه کدام روش انتخاب شود متفاوت است ولی برای opt - 2 نشان داده شده است که :

$2 \prec$ طول تور بدست آمده / طول تور بهینه

روشهای ترکیبی **Composite Procedures**

روش کار :

گام یک : یک تور ابتدایی پیدا کنید.

گام دوم : روش opt - 2 را برای تور بدست آمده در گام یک اعمال کنید.

گام سوم : روش opt - 3 را برای تور بدست آمده در گام دوم تکرار کنید.

این روش نسبتا سریع بوده و نتایج خیلی خوبی بدست میدهد.

الگوریتم **Lin and Kernighan**

از آنجا که قلب دو روش حسی بهبود تور و روشهای ترکیبی بر مبنای الگوریتم Lin استوار است و چون با روشهای ابتکاری موجود، کارآیی و زمان اجرا با مقدار n سریعا افزایش می یابد لذا این روش را که برای مسائل بزرگتر از ٦٠ شهر نیز بکار رفته است ارایه می نماییم.

این الگوریتم روش حسی را معرفی میکند که جوابهای بهینه را با دفعات بیشتر و در زمان اجرایی که با n^2 افزایش می یابد در اختیار میگذارد. روش مبتنی بر یک روند عمومی است که کاربرد زیادی دارد این روش با موفقیت زیاد برای تقسیم گراف بکار برده شده است.

اساس این روش بر گامهای زیر استوار است :

گام یک : یک جواب موجه تصادفی از گره های T که معیار C را راضی کند.

گام دوم : تلاش در جهت یافتن جواب موجه T' با تبدیل T

گام سوم : اگر جواب بهبود یافته باشد یعنی $f(T') \prec f(T)$ ، آنگاه T را با T' تعویض کنید و از گام دوم تکرار کند.

گام چهارم : اگر جواب بهبود یافته پیدا نشود T یک جواب بهینه محلی (Local Optimum) است. از گام یک با یک جواب دیگر شروع کنید.

قلب روش ، گام دوم است و یکی از کارهایی که میشود تعویض k عنصر از T با k عنصر ازS_T است. به نحویکه جواب بدست آمده موجه و بهبود یابد و این عمل تا آنگاه ادامه می یابد که با عمل تعویض بهبود حاصل نشود ودر اینصورت جواب بهینه محلی بدست آمده است.

مشخص کردن k بسیار مشکل است چرا که محاسبات با افزایش k سریعا افزایش می یابد و بسیار مشکل است که از قبل مقدار k را که بهترین جواب را در ارتباط با زمان محاسبه و کیفیت جواب تعیین میکند ،مشخص نمود.

این روش حسی مبتنی بر عمومیت دادن تبدیل تعویضی است، فرض کنید که T جواب غیر بهینه ولی موجه باشد. آنگاه میتوان گفت T بهینه نیست زیرا k عنصر $x_1,.....x_k$ در T وجود دارد که خارج از محل خود هستند و برای آنکه T بهینه شود باید k عنصر $y_1,.....y_k$ از S_T جایگزین آنها شود. بنابراین مسئله تعریف x , k و y ها است.

از آنجا که نمیدانیم k چه باید باشد به نظر مصنوعی است که k را ثابت نگاهداشته و سپس کلیه زیرمجموعه های k عضوی T را در نظر بگیریم در مقابل سعی می کنیم k ، $x_1,.....x_k$ و $y_1,.....y_k$ را تا حد امکان عنصر به عنصر پیدا کنیم. در نتیجه سعی میکنیم ابتدا x_1 و y_1 که بدترین زوج خارج از محل هستند پیدا کنیم سپس با کنار گذاشتن x_1 و y_1 پیدا شده، x_2 و y_2 را که بدترین زوج باقیمانده هستند انتخاب میکنیم و به همین ترتیب، بنابراین به صورت الگوریتم وار :

۱ـ یک جواب اولیه تصادفی T تولید کنید

۲ـ الف : i = 1

ب : x_i و y_i را از بالاترین جفت در مرحله i ام انتخاب کنید. این بدین معناست که x_i و y_i به نحوی انتخاب شوند که بهبود ماکزیمم شود.

وقتی $x_1,.....x_i$ با $y_1,.....y_i$ تعویض شود x_i از مجموعه $T-\{x_1,.....x_{i-1}\}$ و y_i از مجموعه $S_T-\{y_1,.....y_{i-1}\}$ انتخاب میشود.

ج : اگر مشخص شود که بهبود دیگری نمیتوان بدست آورد به مرحله سوم بروید در غیر اینصورت i = i +1 و به مرحله دوم ب بروید.

۳ـ اگر بهترین بهبود پیدا شده برای i = k باشد. $x_1,.....x_k$ را با $y_1,.....y_k$ تعویض کنید تا یک T جدید بدست آید و به مرحله دوم بروید. اگر بهبودی یافت نشد به مرحله چهارم بروید.

٤ـ در صورت لزوم مجددا از مرحله یک با جواب اولیه دیگر شروع کنید.

برای این کار به موارد زیر نیاز داریم :

۱ـ روش انتخاب که به صورت سریع و کارآ جفت عنصری که بیشتر از همه خارج از محل هستند مشخص نماید.

۲ـ تابع ساده ای که جمع سود ناشی از یک مجموعه تعویض پیشنهادی را نشان دهد. از نظر عملکرد بهتر است این تابع بدینصورت عمل کند که با تعویض های ممتد به صورت جمعی، سود را نشان دهد.

این تابع با مشخصه فوق، نیازدارد تا بروز g_i[٤] منفی را به محض ظهور را برطرف کند در واقع هر گاه که $\sum_{i=1}^{k} g_i \le 0$ برای کلیه k باشد در آنصورت متوقف می شویم.

۳ـ در صورت تعویض $x_1,.....x_k$ با $y_1,.....y_k$ باید اطمینان از موجه بودن وجود داشته باشد تا از کار زیاد برای تعویض مواردی که موجه نخواهد بود جلوگیری شود.

٤ـ قاعده توقف که مشخص نماید بهبود دیگری در جستجوی عناصر دیگر برای تعویض وجود نخواهد داشت یا حداقل به نقطه ای رسیده ایم که برگشت صورت میگیرد.

٥ـ باید $x_1,.....x_k$ و $y_1,.....y_k$ جدا از هم باشند هر گاه از نقطه ای به یک سمت حرکت داده شد در همان مرحله بازگشت وجود نداشته باشد.

---

[٤] سود ناشی از تغییر

الگوریتم پایه

فرض کنید S مجموعه کلیه فواصل بین n شهر باشد و فرض کنید T یک زیر مجموعه n تایی از S است که یک تـور را تشـکیل میدهد. فرض کنید یک تور دلخواه T با طول f(T) وجود داشته باشد و تور دیگر T' با طول $f(T')$ که $f(T') \prec f(T)$. فرض کنید T و T' با یک مجموعه k فاصله از هم متمایز میشوند. الگوریتم تلاش میکند که T را به T' تبدیل نماید. بدین معنی که دو مجموعه فاصله های $X = \{x_1,.....x_k\}$ و $y = \{y_1,.....y_k\}$ به نحوی باشد که اگر این فاصله ها در X حذف شود وبا فاصله های موجود در y تعویض شود نتیجه توری با هزینه کمتر شود.

فرض کنید طول x_i و y_i به ترتیب $|x_i|$ و $|y_i|$ در آنصورت :

از آنجـا کـه بدنبال توالی از g_i ها میگردیم که جمع آنها مثبت باشد لذا تنها لازم است توالیهایی را دنبال کنیم کـه کلیه جزءهای آن نیز مثبت باشد این معیار ما را قادر میسازد که تعداد توالیهایی را که باید آزمایش کنیم کاهش دهیم واین معیار قلب قاعده ختم است.

الگوریتم :

۱ـ یک تور تصادفی اولیه تولید کنید

۲ـ $G'' = 0$ (G'' بهتریـن بهـبودی اسـت کـه تاکنون اتفاق افتاده است). هر گره فرضی مثلا t_1 را انتخاب کنید و فرض کنید x_1 یکی از فواصل در مجاورت t_1 باشد. i = 1

۳ـ از نقطه انتهایی دیگر t_2 از x_1 ، y_1 به t_3 با $g_1 \succ 0$ را انتخاب کنید اگر چنین y_1 وجود نداشته باشد به گام (٦ ـ د) بروید.

٤ـ i = i +1 . x_i و y_i را به صورت زیر انتخاب کنید.

الف) x_i به صورتی انتخاب میشود اگر t_{2i} به t_1 منجر به بوجود آوردن یک تور شود.

ب) y_i یـک فاصـله در نقطـه انتهایی t_{2i} است که با x_i شریک بوده و تحت محدودیتهای ج ، د ، ه است. اگر y_i وجود نداشته باشد به گام پنجم بروید.

ج) بـرای تضـمین آنکـه x هـا و y هـا از هـم جدا باشند، x_i نمیتواند فاصله ای باشد که قبلا به هم متصل شده باشد و همینطور y_i نمیتواند فاصله ای باشد که قبلا شکسته باشد.

د) $G_i = \sum_1^i g_i \prec 0$

ه) به منظور اطمینان از موجه بودن "الف" میتواند در i = 1 راضی شود و y_i انتخاب شده باید امکان شکستن $x_i + 1$ را بدهد.

ی) قـبل از آنکه y_i ایجاد شود امتحان میکنیم اگر اتصال t_{2i} به t_i منجر به نتیجه بهتری از آنچه قبلا بدست آمده است را میدهد (چون معیار موجه بودن برای $i \geq 2$ راضی شده است میدانیم که منجر به یک تور میشود)

فـرض کنید y_i یک فاصله باشد که t_{2i} را به t_i وصل میکند و $g_i = |y_i^*| - |x_i^*|$ ، اگر $G_{i-1} + g_i^* \succ G^*$ باشد آنگاه $G^* = G_{i-1} + g_i^*$ و k = i

۵ـدر صورتیکه یا هیچ فاصله ای بین x_i و y_i نباشد (٤ ـ د) و (٤ ـ ه) که را راضی نماید و یا وقتیکه $G_i \prec G^*$ ، ساخت x_i و y_i در گام دوم تا چهارم را متوقف کنید (قاعده ختم).

اگـر $G^* \succ 0$ در آنصـورت تـور T' را انتخاب کنید با $f(T') \prec f(T) - G$ و کلیه فرآیند را از گام دوم با T' دنبال کنید.

٦ـ اگر $G^* = 0$ روش برگشت (Back Tracking) محدود دنبال میشود.

الف) با انتخاب y_2 به منظور افزایش طول، مراحل ٤ و ٥ را دنبال کنید تا آنگاه که معیار $g_1 + g_2 \succ 0$ راضی شود. به هر حال با بدست آمدن هر بهبود به گام دوم بروید.

ب) اگـر کلـیه انتخابهای y_2 در گام (٤ ـ ب) بدون فایده مصرف شود به گام (٤ ـ الف) بروید و سعی کنید که x_2 به صورت متناوب انتخاب شود.

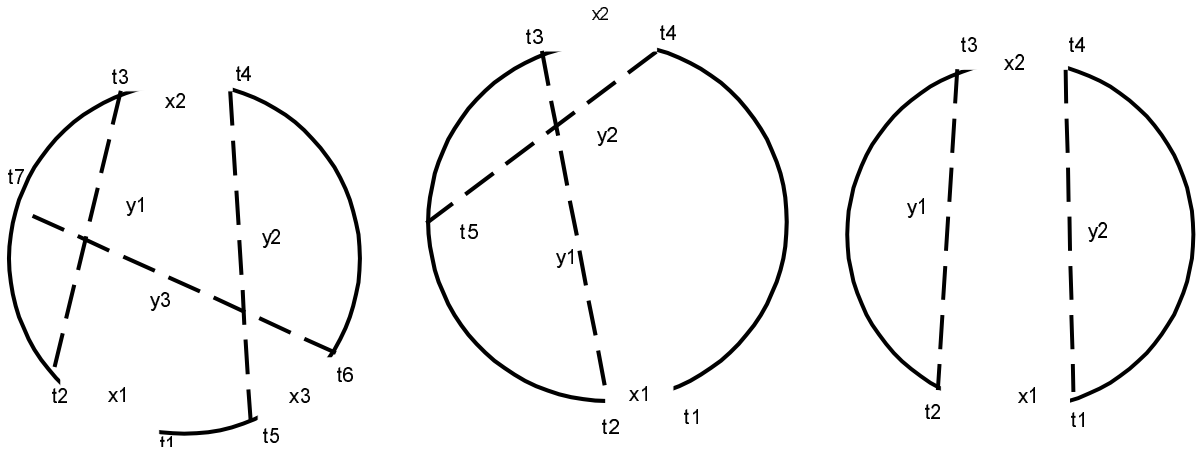
ج) اگـر ایـن کار نیز قادر به بهبود نباشد وقتی y_1 ها به صورت افزایشی طول مسیر منظم شده باشد برگشت به گام (٣) انجام میشود.

د) اگر y_1 ها بدون فایده مصرف شود سعی شود x_1 به صورت متناوب در گام دوم بکار رود.

* توجه : روش برگشت فقط وقتی انجام میشود که بهبود بدست نیاید و فقط در سطوح ١ و ٢ (٢ یا i = 1) انجام میشود.

٧ـ الگوریتم هنگامی خاتمه می یابد که کلیه n مقدار t_1 بدون بهبود مورد استفاده قرار گیرد، در این زمان میتوان تورهای تصادفی دیگر را از مرحله (١) شروع کرد .

حالـت خـاص : تنها موردی که باید بحث شود حالتی است که x_2 بین t_1 و t_3 باشد ما این امکان را علیرغم پیچیده شدن بوجود می آوریم چرا که کارآیی را افزایش میدهد.انتخاب یک جایگزین غیرموجه فقط برای i = 2 مجاز است.



شکل – (a - ١) شکل – (b - ١) شکل – (c - ١)

حلقه : $t_1 - t_4 - t_5 - t_2 - t_3 - t_6 - t_1$ حلقه : $t_1 - t_5 - t_4 - t_6 - t_7 - t_3 - t_2 - t_8 - t_1$

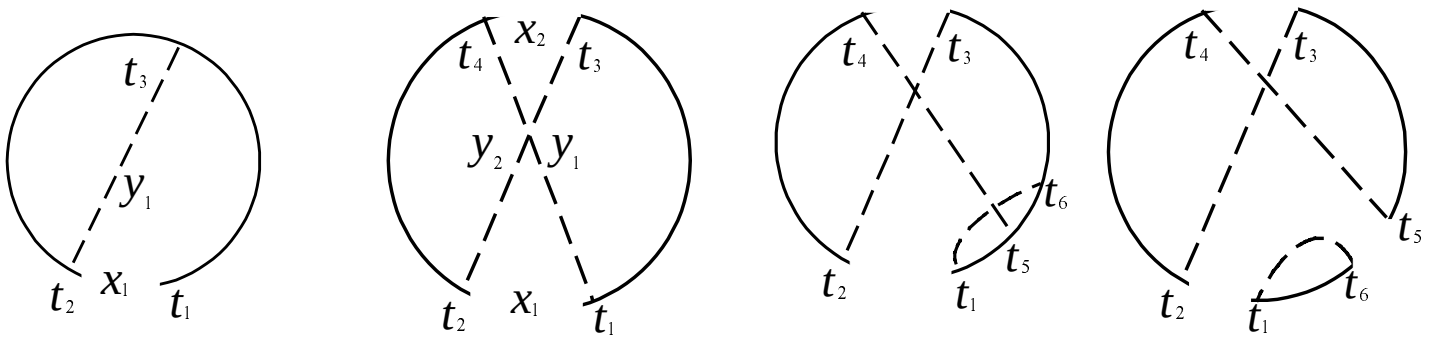
یا $t_1 - t_4 - t_5 - t_2 - t_3 - t_6 - t_1$ یا $t_1 - t_5 - t_4 - t_6 - t_7 - t_3 - t_2 - t_8 - t_1$

حال اگر t_5 بین t_2 و t_3 باشد (شکل ٦-١) t_6 میتواند در هر طرف t_5 باشد.

اگر t_5 بین t_1 و t_4 باشد (شکل c -١) صرفا یک انتخاب برای t_6 وجود دارد و t_6 باید بین t_4 و t_5 قرار گیرد تا شرط موجه بودن رعایت شود بعلاوه t_7 باید بین t_2 و t_5 باشد در آنصورت t_8 میتواند در هر سمت t_7 قرار گیرد و در هر صورت موردی انتخاب میشود که $|x_4|$ حداکثر باشد.

بعد از این حالت خاص به روش نرمال الگوریتم در گام (٤) باز میگردیم.

مثال :



(الف) (ب) (ج) (د)

با یک تور شروع کنید که در شکل (٢ الف) نمایش داده شده است و دو گره مجاور t_1 و t_2 و فاصله بین آن دو را در نظر بگیرید. فرض کنید t_3 گره ای باشد که به t_2 نزدیکتر است و y_1 فاصله (t_2,t_3) باشد. چون x ها و y ها جدا از هم هستند y_1 مجاز نیست جزء فواصلی باشد که به t_2 متصل هستند. $g_1=|x_1|-|y_1|$ اگر مثبت نباشد به مرحله (٦ ـ د) می رویم و t_2 را همسایه دیگر t_1 در نظر می گیریم I = 1 و t_4 را در همسایگی t_3 در نظر می گیریم (شکل ١ـ ب) همانطور که در شکل نشان داده شده است x_2 فاصله t_3 و t_4 است.

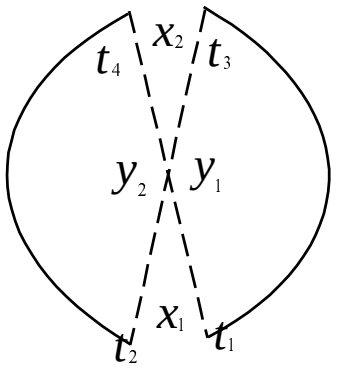
حال اگر y_2 انتخاب شودتا t_4 را به t_1 متصل کند تور جدیدی بوجود می آید و اگر $g_1+g_2 \succ 0$ در آنصورت میتوان T را با تعویض x_1 و x_2 با y_1 و y_2 بهبود داد. این بهبود را به خاطر بسپارید یعنی G^* به ازای k = 2 در مرحله (٤ ـ ی).

t_5 را بعنوان نزدیکترین همسایه t_4 انتخاب کنید و فاصله (t_4,t_5) را در نظر بگیرید دوباره t_5 نمیتواند هیچ یک از گره هایی که به t_4 متصل شده اند باشد. همانطور که در شکل (٢ ـ ج) دیده میشود فقط یک انتخاب برای t_6 و x_3 وجود دارد و اگر x_3 فاصله دیگری باشد که به t_5 متصل است در آنصورت تور به دو قسمت مجزا تقسیم میشود (٢ ـ د).

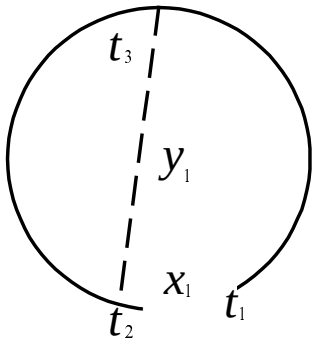
دوباره امتحان میکنیم که شاید با اتصال t_6 به t_1 نتیجه بهتری از آنچه از اتصال t_4 به t_4 داشتیم بوجود آید در آنصورت G^* را به هنگام نموده و k = 3 خواهد بود. اگر $g_1+g_2+g_3 \prec G^*$ قاعده ختم می گوید که تعویض مفید در k_2 صورت می گیرد. اگر بهترین انتخاب y_3 برای (t_6,t_1) باشد آن را انجام داده ایم در غیر اینصورت با انتخاب t_7 این روش را ادامه می دهیم.

روش حسی پیشنهادی

کلیه روشهای حسی پیشنهاد شده عموما مبتنی بر ماتریس هزینه متقارن میباشند و کار روی ماتریس های نامتقارن به دلیل کاربرد محدود صورت نگرفته است. بعلاوه خصوصیت ماتریس نامتقارن سبب پیچیده تر شدن حل TSP میشود زیرا برخلاف حالت تقارن با گسستن فاصله بین دو گره و انتخاب مسیرهای دیگر سبب غیرموجه شدن تور و در بعضی موارد به دلیل عدم توجه به نامساوی بودن مسیر رفت و برگشت منجر به عدم بهبود تور و افزایش هزینه تور گردد. توضیح این پیچیدگی را با یک مسئله نمونه تشریح میکنیم.



(شکل ب)



(شکل الف)

فرض کنید شکل (۳ ـ الف) تور ابتدایی باشد و دو گره t_1 و t_2 را در نظر بگیرید و فرض کنید گره t_3 نزدیکترین گره به t_2 باشد بعلاوه فرض کنید (مطابق هر روش دلخواه) فاصله t_3 و t_4 بهتر است گسسته شود در اینحالت توجه میشود که یک جواب موجه $(t_2-t_3-t_1-t_4-t_2)$ است. پیچیدگی مسئله از آنجا بروز میکند که اگرچه $y_1+y_2 \prec x_1+x_2$ است ولی لزوما تور جدید بهبود نیافته است و حتی امکان دارد طول تور افزایش یافته باشد.

زیرا در جواب اخیر مسیر t_2 <--- t_4 بر خلاف تور اولیه یعنی t_4 <--- t_2 انتخاب شده است و برای این تعویض باید دلیل و معیاری وجود داشته باشد زیرا به دلیل عدم تقارن ماتریس هزینه این دو مسیر با هم برابر نیستند.

به عبارت ساده تر تور ایجاد شده در TSP نامتقارن تور جهت دار است در صورتیکه در TSP متقارن تور جهت دار نیست و لذا دو تور $(t_2-t_3-t_1-t_4-t_2)$ با $(t_2-t_4-t_1-t_3-t_2)$ با هم برابر هستند.

الگوریتم پیشنهادی به دلیل این پیچیدگی صرفا بر مبنای دو فاصله گسسته در تور عمل کرده و به جز دو فاصله گسسته دیگر مسیرها را عینا در تور ایجاد شده مرحله بعد ثابت نگاه میدارد بنابراین مقایسه ها را در حداقل خود نگاه میدارد.

الگوریتم حسی برای مسائل نامتقارن

از آنجا که کلیه الگوریتم های حسی موجود مبتنی بر مسائل متقارن هستند لذا برای مسائل نامتقارن کار نمیکند و باید تعدیلاتی در آنها صورت گیرد. در مواردی نیز اساسا پایه الگوریتم بر متقارن بودن مسئله مبتنی است. لذا روش زیر را که با الهام از کار Lin و تجزیه تور به دو زیر تور و بدست آوردن حل 2 - opt و 3 - opt است ارایه میشود.

اساس الگوریتم بر محاسبه تابع بهبود که به صورت جمعی تعریف میشود استوار است. در صورتیکه تابع بزرگتر از صفر باشد بهبود حاصل شده است و تعویض انجام می گیرد در غیر این صورت علیرغم آنکه در ابتدا ظاهرا بهبود بدست می آید ولی طول مسیر افزایش می یابد و لذا تعویض انجام نمیشود.

قاعده ختم : هنگامی است که کلیه گره ها مطابق الگوریتم سنجیده شوند وبهبود یا تعویض صورت نگیرد.

مراحل الگوریتم :

گام یک : یک تور موجه ابتدایی به صورت تصادفی انتخاب کنید و NC = 0 [۵].

گام دوم : از یک گره دلخواه مثلا i شروع کنید. نزدیکترین گره به آن را پیدا کنید. اگر این گره i =i +1 و یا i =i+n- 1= i -1 باشد آنگاه گره بعدی را انتخاب نمایید

شماره گره برابر است با i = i + 1 و NC = NC + 1 گام دوم را تکرار کنید. در غیر اینصورت به گام سوم بروید.

گام سوم : اگر این گره، گره j ام باشد آنگاه تابع بهبود را به صورت زیر محاسبه کنید.

$$g_i = d(i,i+1) + d(j-1,j) - d(i,j) + g_{hi}$$ [۶]

که در آن d(i , j) فاصله بین گره i و j است و g_{hi} به صورت زیر تعریف میشود :

$$Min\, g_{hi} = \{d(k,l) - d(k,i+1) - d(j-1,l)\} \qquad \forall K,L \in S, S = \{j, j+1, ..., i-1, i\}$$

L و K دو گره متوالی در زیر مجموعه گره های S است که یک زیر تور (i , j , j+1 , ……, i-1 , i) را تشکیل میدهند و تابع g_{hi} را حداکثر میکنند.

حالت خاص اول : اگر بین گره i و j فقط یک گره موجود باشد و آن را k بنامیم ، رابطه g_{hi} به صورت زیر تعریف میشود :

$$Min\, g_{hi} = \{d(x,y) - d(x,k) - d(k,y)\} \qquad \forall x,y \in S, S = \{j, j+1, ..., i-1, i\}$$

حالت خاص دوم : اگر بین گره i , j فقط یک گره موجود باشد و آن را k بنامیم ، رابطه g_{hi} به صورت زیر تعریف میشود :

$$Min\, g_{hi} = \{[(d(j+1,i) - d(j+1,i+1) - d(j-1,i)],[(d(j,j+1) - d(j-1,j+1) - d(j,i+1)]\}$$

با پیدا کردن L و K و تابع g_i به گام چهارم بروید.

گام چهارم : اگر $\forall K,L \in S$ $g_i \prec 0$ آنگاه بهبودی با این تغییر حاصل نمیشود.

i = i + 1 و NC = NC + 1 به گام دوم بروید.

در غیر اینصورت تور را به صورت زیر به هنگام کنید.

$$t_i = (i, j, j+1, ..., K, i+1, i+2, ..., j-1, L, L+1, ..., i-1, i)$$

---

[۵] شمارنده تعداد گره های تحلیل شده در هر بار بررسی الگوریتم است.

[۶] رابطه دو گره I , I+1 گسسته ، رابطه دو گره j,j+1 نیز گسسته شده و رابطه گره I,j برقرار میشود

NC = 0 و i = i + 1 به گام دوم بروید.

تبصره : برای حالت خاص اول ، تور را به صورت زیر به هنگام نمایید:

$$t_i = (i, j, \ldots\ldots\ldots\ldots\ldots\ldots, K, \ldots\ldots\ldots\ldots\ldots, i)$$

و برای حالت خاص دوم ، تور را بر حسب نوع انتخاب به صورت زیر به هنگام نمایید:

برای حداقل ناشی از انتخاب بخش اول رابطه g_{hi}

$$t_i = (i, j, j+1, i+1, i+2, \ldots\ldots\ldots\ldots, j-1, i)$$

برای حداقل ناشی از انتخاب بخش دوم رابطه g_{hi}

$$t_i = (i, j, i+1, \ldots\ldots\ldots\ldots\ldots\ldots, j-1, i-1, i)$$

قاعده ختم : در هر مرحله اگر $NC \succ N$ آنگاه آخرین تور، تور بهینه است.

تعداد محاسبات : حداکثر با توان دوم N تغییر میکند.

فهرست منابع :

1- Optimization Algoritms for Networks and Graghs – Minieka, 1978 (261)
2- Material Handling Systems Design – Apple, 1972 (333)
3- Bellmore, M., and G.L.Nemhauser, 1968 – The Traveling Salesman Problem: A Survey, Orsa, Vol.16, pp. 538-558
4- Lin, S., and B.W. Kernighan, 1973. An Effective Hearistic Algorithm for the Traveling Salesman Problem, Orsa, Vol.21, no.2, pp. 498-516
5- B.Golden, L.Bodin, T.Doyle and W.Stewarts, 1981. Approximate Traveling Salesman Algorithms, Orsa, Vol.29, pp. 694-710