معرفی روشهای اصلاح شده در بهینه سازی کولونی مورچه ها و پیشنهاد یک local optimization

علی اکبر آقامحمدی
گروه کنترل
دانشکده برق وکامپیوتر دانشگاه تبریز
aghamohammadi_ali@yahoo.com

امیر حسین تمجیدی
گروه کنترل
دانشکده برق و کامپیوتر دانشگاه تبریز
amirhosseintamjidi@yahoo.com

چکیده: دراین مقاله مسئله TSP به عنوان یک مسئله بهینه سازی که با روش های کلاسیک قابل حل نیست معرفی شده و از میان روش های متفاوت بهینه سازی هوشمند، بهینه سازی کولونی مورچه ها که نسبت به سایر روش ها پاسخ های بهتری نیز به دست می دهد، مورد بحث واقع شده است. سیر تحولی و تکاملی این الگوریتم ها از ACS تا MACS و GMACS آورده شده است. روش پیشنهادی مورد مقایسه قرار گرفته و نتیجه گرفته شده است که روش پیشنهادی با کاستن از مقدار مقایسه و کا هش نرخ محاسبات، سرعت الگوریتم را مخصوصا در مسائل با ابعاد بزرگ، بهبود می بخشد.

کلمات کلیدی: *بهینه سازی کولونی مورچه ها , Local Optimization ,TSP,MACS*

۱-مقدمه

در مسائل بهینه سازی با تعداد زیاد پارامتر، روش های قدیمی یا کارایی چندانی ندارند و یا وقت و هزینه زیادی را برای حل مسئله صرف می کنند. در این گونه موارد شایسته است که به جای بررسی تمامی فضای جواب، که تقریبا غیر ممکن است، از روش های دیگری استفاده شود که به صورت هوشمند گستره فضای جستجو را کاهش دهند. در راستای این هدف، لگوبرداری از سیستم ها و فرایند های طبیعی و بیولوژیک، که در انجام چنین فرایند هایی موفق تر عمل می کنند، رونق فراوانی یافته است که ازجمله آنها می توان به الگوریتم های ژنتیکی، شبکه های عصبی و بهینه سازی کولونی مورچه ها (Ant Colony Optimization=ACO) اشاره کرد. ACS اولین بارتوسط M. Dorigo و L.M.Gambardella برای حل مسئله TSP مورد استفاده قرار گرفت[۱]. در بخش(۲)، در مورد مفا هیم مربوط به ACS از جمله local updating و global updating و مقادیر مناسب برای پارامترهای موجود در الگوریتم، بحث شده است. در بخش(۳) الگوریتم های اصلاح شده و روش پیشنهادی در ارتباط با local optimization آورده شده است. بخش(۴) به نتیجه گیری اختصاص دارد.

۱-۱- مسئله TSP

مسئله TSP تشکیل شده است از تعدادی شهر که در نقاط مختلف یک نقشه قرار دارند و یک فروشنده دوره گرد می خواهد با طی کردن کوتاه ترین مسیر ممکن، به تمامی این شهرها سفر کند و ضمنا از یک شهر دوبار عبور نکند. حل این مسئله، مخصوصا وقتی تعداد شهرها زیاد باشد، با روشهای تحلیلی ممکن نیست. مسئلهTSP برای تعداد مختلف شهر ها در مختصات دو و سه بعدی، و نیز برای حالتی که فاصله رفت و برگشت بین دو شهر متفاوت باشد (ATSP) به صورت استاندارد درآمده است و پاسخ بهینه برای هر یک ازاین موارد در دسترس می باشد[۴]. ما در مقاله خود از مسائل Eil۵۱ و Kroa۱۰۰ استفاده نموده ایم. مسئله اول شامل ۵۱ شهر و دومی شامل ۱۰۰ شهر است.

۲-مفاهیم مربوط به ACS

مشاهده شده است که مورچه ها معمولا بعد از گذشت مدت زمانی، کوتاه ترین مسیر را برای دستیابی به غذا می یابند و به صورت دسته جمعی از این مسیر استفاده می کنند. مکانیزم حاکم بر رفتار آنها به این صورت است که هر مورچه به سمت هدف مورد نظر (غذا) حرکت می کند و در مسیر حرکت خود ماده ای به نام فرمون(pheromone) بر جای می گذارد. ضمنا فرمون به جای مانده در مسیر با نرخ ثابتی تبخیر می شود و مورچه های دیگر را به سمت خود جذب می کند. به طور همزمان تعداد زیادی مورچه به این کار پرداخته و مسیر های مختلف را آزمایش می کنند. بنابراین مورچه ها به مسیری همگرا خواهند شد که فرمون در آن از غلظت بیشتری برخوردار است. در ابتدا که فرمونی وجود ندارد مورچه ها در دو راهی ها هیچ رجحانی برای انتخاب یک مسیر خاص ندارند و با توجه به احتمالات، به طور متوسط تعداد مورچه هایی که در هر یک از دو جهت به راه خود ادامه می دهند، مساوی است. اما به دلیل تبخیر، باگذشت زمان مسیرهای کوتاه تر، حاوی فرمون بیشتری خواهند بود و بیشتر مورچه ها به سمت مسیر های کوتاه تر جذب می شوند.

در واقع مورچه ها Agent های ساده ای هستند که با ارتباط فرمونی خود یک حافظه گسترده (distributed) ایجاد می کنند و با بهره گیری از فرمون و این حافظه، جواب مسئله را به صورت شراکتی به دست می آورند.

در این مقاله، ما مسئله های TSP متقارن در مختصات دو بعدی را مورد بررسی قرار داده ایم. ابتدا تعداد شهرها را مشخص می کنیم، هرشهر با یک جفت مرتب (x_i , y_i) نشان داده می شود. d(r,s) فاصله اقلیدسی بین دو شهر r و s می باشد. حال تعداد مشخصی مورچه را به طور تصادفی در شهر های موجود قرار می دهیم و مورچه ها طبق قانون حرکتی که در زیر توضیح داده می شود، شهر بعدی را انتخاب می کنند. در این انتخاب دو معیار به طور همزمان مد نظر قرار می گیرند:

۱. فاصله تا شهر بعدی

۲. مقدار فرمون در مسیر منتهی به شهر بعدی

در ابتدای الگوریتم فرمون موجود در تمامی مسیرها برابر فرض شده ومقداری در بازه [۰،۱] به آن اختصاص داده می شود. سپس شهر مقصد یا شهرs با توجه به معیارهای بالا، از فرمول زیر محاسبه می شود:

$$s = \begin{cases} s_1 = \underset{u \in \text{NotVisited(k)}}{\arg} \left(\max \left\{ [pheromone(r,u)]^{\partial} \cdot [\eta(r,u)]^{\beta} \right\}\right) & \text{if } q \le q0 \\ s_2 & \text{otherwise} \end{cases} \qquad (۱)$$

اگر مسیر بین شهر های r و s را، edge(r,s) بنامیم، pheromone(r,s) مقدار فرمون موجود در edge(r,s) و $\eta(r,s)$ عکس طول edge(r,s) خواهد بود. NotVisited(k) مجموعه شهرهایی است که مورچه k ام از آنها عبور نکرده است. پارامترβ اهمیت نسبی فاصله نسبت به فرمون، در انتخاب شهر بعدی را تعیین می کند. ثابت q.

در بازه [۰،۱]، و مقداری نزدیک به یک است. در هر مرحله الگوریتم عـدد q را بـه صـورت تـصادفی در بـازه [۰،۱] تولید می کند. اگر عدد تولید شده از .q کوچکتر بود، شهر مقصد s_1 خواهد بود و در غیر این صورت از s_2 استفاده خواهد شد که برمبنای قانون احتمال معرفی شده در زیر بدست می آید و تا حدی به الگوریتم حالت تصادفی تزریق می کند و بدون وجود s_2 احتمال همگرایی الگوریتم به مینیمم موضعی بالا خواهد بود چرا که اسـتفاده از s_2 فـضای جستجو را گسترش می دهد و به الگوریتم کمک می کند که تا حدی از دام مینیمم موضعی برهد. بنابراین با فرض اینکه در لحظه فعلی مورچه k ام در شهر r قرار دارد و s یکی از شهر هایی است که مورچه k ام از آن عبور نکرده است، احتمال انتخاب شهر s به عنوان شهر بعدی از رابطه زیر محاسبه می شود:

$$p_k(r,s)=\begin{cases}\dfrac{[pheromone(r,s)]^{\partial}\cdot[\eta(r,s)]^{\beta}}{\sum_{u\in NotVisited(k)}[pheromone(r,u)]^{\partial}\cdot[\eta(r,u)]^{\beta}} & \text{if } s\in \text{NotVisited(k)}\\ 0 & \text{otherwise}\end{cases} \qquad (۲)$$

برای بدست آوردن جواب بهینه، مقادیر پارامتر ها باید به نحو مناسبی انتخاب شوند. مقادیر مناسب تاحـد زیـادی وابسته به مسئله خواهند بود اما طبق نتایج شبیه سازی های انجام شده مقادیر $q_0 = ۰,۹$ و $\beta = ۶$ و $\partial = 1$ منجر به نتایج مناسبتری شدند.

با حرکت مورچه ها مقدار فرمون در مسیر های پیموده شده، در دو مرحله تغییر می کند:

۱- Global updating

هر بارکه همه مورچه ها به شهری که سفر خود را ازآنجا آغاز کرده بودند، بازگـشتند (تمـام tour هـا بـه پایـان رسیدند) طول مسیر ها را بدست می آوریم و مسیر هر کدام از مورچـه هـا را کـه کوتـاه تـر از بقیـه بـود، انتخـاب می کنیم. به منظور انجام global updating ابتدا فرمون تمامی مسیر ها را به یک نسبت کم می کنیم (در بحث حاضر در این مرحله مقدار فرمون تمامی مسیر ها در ۰،۹ ضرب شده است.) و مقداری فرمون به تمـام edge هـایی که عضو بهترین مسیر هستند اضافه می کنیم. مقدار فرمونی که بایـد بـه edge هـای مـسیر بهینـه افـزوده شـود Δpheromone(r,s)، با عکس طول کوتاه ترین مسیر برابرخواهد بود.

$$pheromone(r,s)=(1-\alpha)\cdot pheromone(r,s)+\alpha\cdot\Delta pheromone(r,s) \qquad (۳)$$

$$\Delta pheromone(r,s)=\begin{cases}(L_{gb})^{-1} & \text{if (r,s)} \in \text{global best tour}\\ 0 & \text{otherwise}\end{cases}$$

که α پارامتر تضعیف فرمون (تبخیر) در مرحله global updating می باشد و L_{gb} طول کوتاه ترین مسیر طی شده تا این مرحله از الگوریتم می باشد.

۲- Local updating

برای اینکه علاوه بر بهترین مورچه به دیگر مورچه ها هم اهمیت قایل شویم و بتوانیم از اطلاعات با ارزش مـسیر آنها استفاده کنیم local updating را طراحی می کنیم یعنی هرگاه یک مورچه از شهری به شهر دیگر مـی رود، باید در مسیر پیموده شده مقداری فرمون طبق فرمول زیر تزریق شود و در عین حال بـه طـور همزمـان بایـد عمـل تبخیر نیز اعمال شود.

$$pheromone(r,s)=(1-\rho)\cdot pheromone(r,s)+\rho.\Delta pheromone(r,s) \qquad (٤)$$

که در آن $(1-\rho)$ برای شبیه سازی تبخیر مورد استفاده قرار می گیرد. در صورتیکه مورچه ای از edge ای عبور کند، فرمون آن edge توسط $\Delta pheromone$ افزایش می یابد. مقدار $\Delta pheromone$، با روشهای مختلفی مقدار دهی می شود. در مرجع [۱] از روش Q-learning استفاده شده است که مخصوص حالتی است که الگوریتم میزان $\Delta pheromone$ تزریقی را خودش یاد می گیرد .اطلاعات بیشتر در مرجع[۲] آمده است. در این روش مقدار فرمون تزریقی برابر خواهد بود با:

$$\Delta pheromone(r,s) = \gamma \cdot \max_{z \in NotVisited(k)} pheromone(s,z) \qquad (۵)$$

که ضریب γ در آن مقداری ثابت و در بازه [۰،۱] است. یک روش دیگر آن است که $\Delta pheremone(r,s)$ را مساوی $\phi 0$ (مقدار ثابت) قرار دهیم و روش سوم مساوی قرار دادن آن با صفر می باشد(حذف local updating). که طبق شبیه سازی های انجام شده روشهای اول و دوم هر دو نتایج خیلی بهتری نسبت به روش سوم بدست می دهند چرا که حذف کردن local updating باعث می شود الگوریتم خیلی سریع به مینیمم موضعی همگرا شود.

۳- الگوریتم های اصلاح شده :

۳-۱- MACS (Multiple Ant Colony System)

در این روش از ایده کولونی های موازی استفاده شده است و سعی در جلوگیری از همگرایی به مینیمم های موضعی دارد. روش ACO از فیدبک مثبت استفاده می کند اما در MACS می توان با تغییر پارامتر ها، فیدبک منفی را هم وارد الگوریتم کرد. در این روش M کولونی مورچه داریم که در هرکدام m مورچه موجود می باشد و (h,k) مورچه k ام است که به کولونی h ام تعلق دارد. در زمان t، $m \times M$ مورچه بین شهر ها در حال حرکت می باشند. هر مورچه در بازه زمانی [t,t+۱]، از شهر i به شهر j می رود که نحوه انتخاب شهر j در زیر توضیح داده می شود:

اگر $pheromone_{ij}^{h}(t)$ را میزان فرمون edge(i,j) در کولونی h ام در زمان t بگیریم و ρ ضریب تبخیر و L^{hk} طول tour طی شده توسط مورچه ی (h,k) باشد، بعد از n بازه ی زمانی میزان فرمون بین شهرهای i و j در کولونی h ام برابر است با :

$$pheromone_{ij}^{h}(t+n) = \rho.pheromone_{ij}^{h}(t) + \sum_{k=1}^{m} \Delta pheromone_{ij}^{hk} \qquad (۶)$$

$$\Delta pheromone_{ij}^{hk} = \begin{cases} Q / L^{hk} & \text{if ant (h,k) uses edge (i,j) between t and t+n} \\ 0 & \text{otherwise} \end{cases} \qquad (۷)$$

جمله اول در رابطه (۶) تبخیر را شبیه سازی می کند وجمله دوم بیان می کند که در اضافه شدن فرمون به edge(i,j)، الگوریتم تمامی کولونی ها را بررسی می کند و هرکولونی که edge(i,j) در آن طی شده است، در

سیگما اثرخود را نشان می دهد. از طرفی با توجه به تعریف L_{hk} مشاهده می شود اثری که هر مورچه بر جای می گذارد، با طول مسیری که طی کرده است نسبت عکس دارد یعنی مورچه هایی که مسیرکوتاه تری را طی کرده اند، اثر گذاری بیشتری روی افزایش فرمون دارند. پارامتر Q هم میزان اضافه شدن فرمون در local updating را مشخص می کند.

در واقع در این الگوریتم local updating نقش موثرتری را ایفا می کند چرا که به دلیل تعدد کولونی ها، تعداد ant ها بالا رفته و فضای جستجو وسیع تر شده است و در واقع احتمال از دست دادن مسیر های مناسب در اثر به دام افتادن در مینیمم های موضعی، بسیار پایین می آید.

$\pi_{ij}^{h}(t)$ میزان تمایل به انتخاب شهر j است(که در اثر ارتباط بین کولونی ها شکل می گیرد.)

$$\pi_{ij}^{h}(t)=\{\ \prod_{l=1}^{M}[pheromone_{ij}^{l}(t)+C(h)]^{\alpha(h,l)}\ \}\cdot(1/d_{ij})^{\beta(h)} \qquad (۸)$$

$\alpha(h,l)$ میزان تاثیر کولونی l بر h را نشان می دهد، یعنی اگر مثبت باشد، کولونی l بر روی کولونی h تاثیر مثبت خواهد داشت (افزایش فرمون) و اگر منفی باشد، فیدبک منفی ایجاد خواهد شد(کاهش فرمون) و اگر $\alpha(h,l)$ صفر باشد، کولونی l برکولونی h تاثیری نخواهد داشت. $C(h)$ پارامتر بایاس است و همیشه میزا ن ثابتی از تمایل را در کولونی h بایاس می کند. d_{ij} هم طول edge(i,j) است.

پس قانون احتمال برای انتخاب شهر مقصد به صورت زیر در می آید:

$$p_{ij}^{hk}=\begin{cases}\dfrac{\pi_{ij}^{h}(t)}{\sum_{u\in NotVisited^{hk}(t)}\pi_{il}^{h}(t)} & \text{if } j\in NotVisited^{hk}(t)\\ 0 & \text{otherwise}\end{cases} \qquad (۹)$$

۳-۲. GMACS (Genetically Modified ACS):

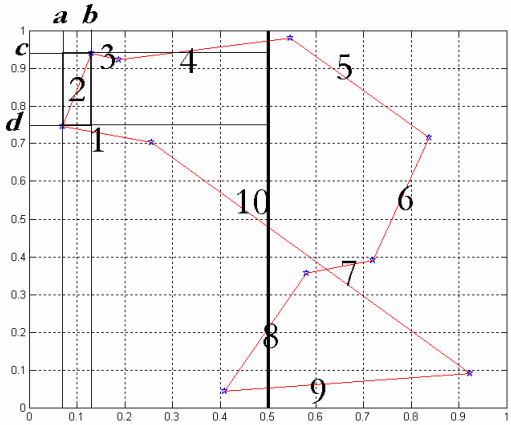
بررسی ها در جهت کشف روابط میان پارامترها با هم و تاثیر آنها بر سرعت همگرایی، منجر به استفاده از GMACS شد. استفاده از الگوریتم ژنتیکی باعث می شود که مقدار پارامتر ها بهبود یابد و مسیر های کوتاه تر توسط مورچه ها ایجاد شود. در این روش ارزش هر مورچه برابر است با تفاضل طول مسیری که آن مورچه در تکرار قبلی طی کرده و بهترین طول بدست آمده توسط مورچه های دیگر. fitness function برای هر مورچه برابر است با ارزش هر مورچه، تقسیم بر متوسط ارزش در میان جمعیت مورچه ها. روش کار بدین صورت است که کولونی با مقادیر معقولی از پارامتر ها [۳]، مقدار دهی اولیه می شود. سپس برای انتخاب والدین و تولید مثل از روش tournament استفاده می شود. برای crossover از اپراتور uniform استفاده می شود. اپراتور mutation هم با احتمال ۰،۱ رخ می دهد و به طور تصادفی مقدار پارامتر را به مقدار پنج درصد کاهش یا افزایش می دهد.

۳-۳-راه حل پیشنهادی برای تصحیح مسیر های متقاطع

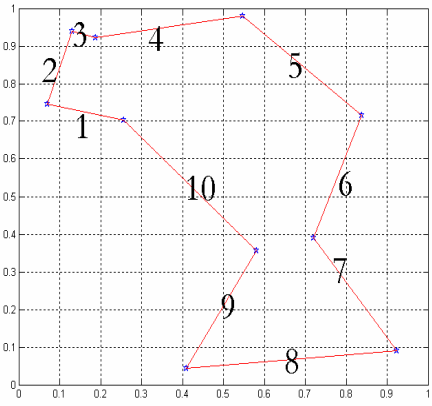
در این بخش در مورد مشکلی که در اجرای الگوریتم اتفاق می افتد یعنی وجود edge های متقاطع در طول مسیر، بحث خواهد شد.

به سادگی اثبات می شود که در جواب بهینه مسئلهTSP، هیچ برخوردی میان edge ها وجود نخواهد داشت.بااعمال روش ACO به مسئله TSP و بررسی iteration های مختلف مشاهده می شود که تقریبا درتمامی مراحل‌بین بعضی edge ها، تقاطع وجود دارد که باعث می شود سرعت همگرایی کاهش یابد و یا در اغلب

موارد در مینیمم موضعی به دام بیفتد. مشکل اصلی در این ارتباط که با احتمال بالا رخ می دهد این است که مسیر انتهایی که بین آخرین شهر و اولین شهر وجود دارد، باعث ایجاد تقاطع می شود. این اشکال به خاطر ماهیت ACO است چرا که مورچه ها بر حسب پارامترهای فاصله وفرمون حرکت می کنند و در گام های ابتدایی که فرمون برای مسیر های متفاوت یکسان است، فاصله، پارامتر غالب در تعیین مسیر می باشد و مورچه ها با احتمال خیلی بالاتر به سمت مسیر هایی با طول کوتاه ترحرکت می کنند و ادامه این روند باعث می شود که در آخرین شهر از نقطه ابتدایی مسیرخیلی دور شده باشند و چون تنها شهر باقی مانده همان شهر اول است، پس اجبارا و در اکثر موارد یک مسیر طولانی و با ایجاد تقاطع های فراوان در بین این دو شهر تشکیل خواهد شد. ما برای از بین بردن این مشکل یک local optimization را برروی بهترین مسیر پیاده می کنیم و سپس مسیر متقاطع را که نقش قطر های یک چهار ضلعی را دارد به مسیر بهینه ای که در واقع یک جفت از ضلع های چهار ضلعی است، تبدیل می کنیم و این فرآیند را تا جایی که دیگر تقاطعی در آن باقی نماند، انجام می دهیم. روش مشابهی که هم اکنون مورد استفاده قرار می گیرد روش $2-opt$ است. تفاوت روش ما با روش $2-opt$ آن است که در $2-opt$ هر edge ای با همه edge های دیگر مورد مقایسه قرار می گیرد. در هر مقایسه دو edge مورد مقایسه حذف می شوند در نتیجه مسیر کلی جواب به دو مسیرک جدا از هم تقسیم می شود. به دوطریق می توان این مسیرک ها را به یکدیگر وصل کرد تا مسیر بسته شود. هر بار با امتحان هر دو شیوه اتصال، آرایشی انتخاب می شود که طول کمتری را بدست دهد. در روش پیشنهادی در این مقاله اولا با اجرای یک الگوریتم خاص فقط مسیر هایی را مورد بررسی قرار می دهیم که با هم تقاطع داشته باشند (چون که در دیگر مسیر ها که تقاطع وجود ندارد آرایش جدید متقاطع بوده و بی تردید به ایجاد مسیر طولانی تر می انجامد). ثانیا با تقسیم صفحه به Set (مجموعه) های مختلف وبررسی تقاطع در هر یک از مجموعه ها، سرعت عملیاتی را افزایش می دهیم. این الگوریتم را با شکل ساده زیر توضیح می دهیم.



شکل (۱)- شهرها به دو set تقسیم می شوند (نیمصفحه سمت چپی، set۱ وسمت راستی set۲ می باشد.)

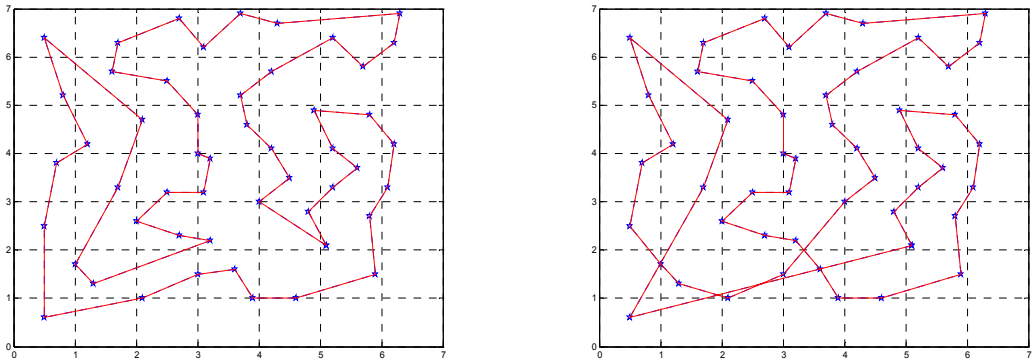


شکل(۲)-مسیر تصحیح شده

فاز۱: در بررسی امکان برخورد edge ها، فقط edge هایی را در الگوریتم بررسی تقاطع وارد می کنیم که اولا در set حاوی edge مورد نظر، عضو باشند و ثانیا امکان ایجاد تقاطع را داشته باشند مثلا برای ۲ edge در شکل (۱)، edge های مورد مقایسه اولا باید در set۱ باشند و ثانیا نباید با ۲ edge سر مشترک داشته باشد همچنین نباید هر دو سر آن در یکی از نواحی چهار گانه زیر باشد: ۱)سمت راست x=b؛ ۲)سمت چپ x=a؛ ۳)بالای y=c؛ ۴)پایین y=d. این کار از افزایش محاسبات به صورت نمایی جلوگیری می کند. لازم به ذکر است نقاط $(x=b, y=c)$ و $(x=a, y=d)$ دو سر ۲ edge هستند.

فاز ۲: پس از تشـخیص مسیر های متقاطع، آنها را تصحیح می کنیم (شکل(۲)). سپس عملیات global updating را انجام می دهیم.

اجرای این الگوریتم برای مسئله Eil۵۱ در شکل (۳) نمایش داده شده است که مربوط به اجـرای تکـرار اول در حل مسئله می باشد و مشاهده می شود که این کار، تکرار اول و تکـرار هـای بعـد از آن را بـه نحـو مـوثری بهینـه می کند و از این طریق طبق شبیه سازی های انجام شده تعداد تکرار ها کاهش می یابد.



شکل (۳)- روش پیشنهادی در تکرار اول، مسیر ایجاد شده را که چندین تقاطع دارد تصحیح می کند .

۴-نتیجه گیری

در این مقاله ما یک روش برای حذف مسیر های متقاطع را پیشنهاد کـردیم کـه در هـر یـک از انـوع معرفـی شـده ACO قابل پیاده سازی است وتوضیح دادیم که این روش با قرار دادن شهرها در چندین مجموعه و جلوگیری از مقایسه های اضافی که در $2-opt$ انجام می گیرد، سرعت بالاتری نسبت به آن دارد.

۵-مراجع:

[۱] Dorigo, M. and Gambarella, L.M.,"Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Person", ۱۹۹۷, IEEE Transactions on Evolutionary Computation, Vol .۱, No .۱

[۲] Gambardella, L. M. and Dorigo, M. ۱۹۹۵ Ant-Q: A Reinforcement Learning Approach to the Traveling Salesman Problem, Proc. ML-۹۵, Twelfth International Conf. on Machine Learning, ۲۵۲-۲۶۰.

[۳] Gaertner, D. and Clark, K.,"On Optimal Parameters for Ant Colony optimization algorithms"

[٤] http://www.iwr.unihelderberg.de/iwr/comopt/sof/TSPLIB۹۵/TSPLIB.html