ارائه یک شبکه عصبی جدید با ساختاری سازنده[1] و ترکیبی برای حل مسأله کوتاهترین مسیر[2] متقارن با تعداد شهر مشخص

مهدی سعادتمند طرزجان
دانشکده برق، دانشگاه صنعتی خواجه نصیر الدین طوسی، تهران، ایران
m_saadatmand_tarzjan@yahoo.com

محمد اکبرزاده توتونچی
دانشکده مهندسی، دانشگاه فردوسی، مشهد، ایران
akbarzar@eece.unm.edu

چکیده در این مقاله با تعمیم شبکه عصبی CNN-TSP، شبکه عصبی سازنده جدیدی برای حل مسأله کوتاهترین مسیر m شهری ارائه شده است. این شبکه با ساختار فیدبکی-رقابتی خود که ایده اصلیش از مفاهیم شبکه‌های عصبی هاپفیلد و کوهونن گرفته شده، قادر است ظرف مدت کوتاهی پاسخهایی مناسب به SP ارائه دهد. ویژگیهای مذکور، این شبکه را به ابزاری مناسب برای کاربردهای بلادرنگ (real-time) مبدل کرده است.

واژه‌های کلیدی *مسأله فروشنده دوره‌گرد (Traveling Salesman Problem - TSP)، مسأله کوتاهترین مسیر (SP)، بهینه‌سازی، شبکه عصبی کوهونن، شبکه عصبی هاپفیلد، شبکه عصبی CNN-TSP*

۱- مقدمه

مسایل بهینه‌سازی ترکیبی، از قبیل مسأله فروشنده دوره‌گرد و مسأله کوتاهترین مسیر از خانواده مسایل NP-Complete هستند [1]. با پیشرفت تکنولوژی، نیاز به حل سریعتر و مناسبتر این مسایل افزایش یافته است. تعیین مسیر حرکت مته برای سوراخ کردن صفحه‌های PCB [2][3]، تعیین مسیر بهینه انتقال داده در شبکه‌های کامپیوتری [4]، بهینه‌سازی شبکه‌های توزیع برق [5]، پردازش تصویر و تشخیص الگو [6]، هدایت روبات [7]، تفکیک و دسته‌بندی داده‌ها (data partitioning) [8] از جمله زمینه‌هایی هستند که حل TSP برایشان بسیار راه گشاست. TSP، همانطور که از نامش بر می‌آید، عبارت است از یافتن کوتاهترین مسیر بسته ممکن بین N شهر.

روشهایی که تاکنون برای حل TSP ارائه شده‌اند، معمولا بر پایه جستجوگرها و آشکارسازهای ایستا (deterministic) یا احتمالاتی (probabilistic) هستند که از آنجمله می‌توان به الگوریتمهای کلاسیک جستجوی محلی [9]، بازپخت تطبیقی (Simulated Annealing - SA) [10]، شبکه‌های عصبی مصنوعی، الگوریتمهای ژنتیکی (Genetic Algorithms - GA) [11][12]، برنامه‌نویسی تکاملی (evolutionary programming) [13]، سیستم کولونی مورچه‌ها (Ant Colony System - ACO) [14][15]، روشهای آموزش افزایشی مبتنی بر جمعیت (population-based incremental learning) [16]، روشهای fine-tuned learning [17] و ... اشاره کرد. هر یک از این روشها دارای نقاط ضعف و قوت خاص خود است. از روشهای تکاملی مانند GA معمولا به دلیل حجم زیاد محاسبات بیشتر در کاربردهای خارج خط (offline) استفاده می‌شود. با اینکه SA و ACO نسبت به الگوریتمهای تکاملی سریعتر هستند اما باز هم در مقایسه با شبکه‌های عصبی مصنوعی کند محسوب می‌شوند. می‌توان گفت که در میان الگوریتمهای مذکور تقریبا شبکه‌های عصبی مصنوعی از بقیه سریعتر هستند اما در عوض، معمولا دقت آنها کمتر است. با این حال، در مجموع می‌توان گفت که برای کاربردهای بلادرنگ (real time) شبکه‌های عصبی جزء بهترین انتخابها هستند.

---

[1] constructive
[2] Shortest Path - SP

شبکه‌های کوهونن یا نگاشتهای خود سازمانده (Self Organizing Maps - SOM) [19]، هاپفیلد [20][21]، بولین (boolean) [22] و آشوبی (chaotic) [23] از جمله شبکه‌های عصبی هستند که تاکنون برای حل TSP بکار گرفته شده‌اند. ما قبلا در [18] شبکه عصبی جدیدی را برای حل مسأله فروشنده دوره‌گرد ارائه نمودیم. در این سبکه سعی شده است که با تلفیق مفاهیم اساسی شبکه‌های عصبی کوهونن و هاپفیلد، از قابلیتهای این دو شبکه به صورت توأم استفاده شود. هدف ما در این مقاله توسعه شبکه عصبی مذکور برای حل بلادرنگ SP است. منظور ما از SPی m شهری، عبارت است یافتن کوتاهترین مسیر m شهری از بین N شهر موجود که مبدأ و مقصدی از پیش تعیین شده را به هم وصل کند.

در [18] طی شبیه‌سازیهای انجام شده، نشان دادیم که شبکه عصبی پیشنهادیمان از سرعت و دقت مناسبی برخوردار است. به طوری که طی زمان بسیار کوتاهتری نسبت به شبکه کوهونن همگرا می‌شود و جوابهای آن در حدود ۲،۵٪ پاسخ بهینه (برای مسائلی با ۳۰، ۴۰ و ۵۰ شهر) کوتاهتر از پاسخهای شبکه کوهونن است. انتظار داریم که چنانچه بتوانیم شبکه مذکور را برای حل SPی m شهری توسعه دهیم، شبکه قدرت بهینه‌سازی خود را حفظ کند. علی‌رغم جستجوی فراوانی که جهت یافتن مقالاتی در این زمینه صورت گرفت؛ نتیجه‌ای حاصل نشد. اغلب کارهایی که در زمینه حل مسأله کوتاهترین مسیر انجام شده است همانند کار آقای Ali و همکارانشان در [4] و یا روشی که آقای Häkkinen و همکارانشان در [24] استفاده کرده‌اند تعداد شهرهای مسیر اهمیتی ندارد و هدف تنها یافتن کوتاهترین مسیر برای یک مسأله نامتقارن است (مسأله‌ای نامتقارن است که در آن فاصله شهر iام تا شهر jام برابر با فاصله شهر jام تا شهر iام نباشد).

در بخش بعدی قرادادهایی که در نمایش کمیتها و معادلات بکار، رفته ذکر شده است، در بخش سوم شبکه عصبی CNN-TSP را معرفی کرده‌ایم. بخشهای چهارم و پنجم به شرح الگوریتم آموزش و نحوه تعمیم این شبکه، از یک شبکه بهینه‌ساز به یک شبکه سازنده می‌پردازند. بخشهای ششم، هفتم و هشتم نیز به ترتیب به بیان نتایج شبیه‌سازیها، نتیجه‌گیری و مراجع اختصاص یافته‌اند.

۲- قرارداد

در ادامه مقاله، موارد ذیل در نمایش کمیتها رعایت می‌شود:

- بردارها و ماتریسها با حروف بزرگ و درایه‌های آنها با حروف کوچک نشان داده می‌شوند. علاوه براین بردارها با علامت بردار مشخص می‌شوند، مانند: $\vec{X}$.
- درایهٔ واقع در سطر i وستون jاز ماتریس V را با $v_{i,j}$ نشان می‌دهیم.
- N تعداد شهرها، $d_{i,j}$ فاصلهٔ شهر i تا شهر j و D ماتریس فاصلهٔ شهرها از یکدیگر است. هدف ما در این مقاله حل مسألهٔ فروشندهٔ دوره‌گرد متقارن (Symmetric TSP - STSP) است. در STSP فاصلهٔ شهر i تا شهر j برابر با فاصلهٔ شهر j تا شهر i است. بدیهی است که به‌این‌ترتیب D یک ماتریس متقارن خواهد بود.

$$d_{i,j} = \text{Distance between ith \& jth city} \qquad i=1,2,\cdots,N \;،\; j=1,2,\cdots,N$$
$$D=\left[d_{i,j}\right]_{N\times N} \qquad d_{i,j}=d_{j,i} \qquad (۱)$$

- عبارت ${}^{i}x_{k}^{l}$، درایهٔ k از بردار خروجی نرون i که در لایهٔ l شبکه واقع است را نشان می‌دهد. همچنین نرون صفرم، همان نرون N و نرون $N+1$ همان نرون اول است.
- مسیری که شبکه در مرحله tام معرفی می‌کند را با $Ph(t)$ نشان می‌دهیم و معادله $Ph_p^t = a$ می‌گوید که در مرحله tام، شهر a در محل pام مسیر قرار دارد.
- $e(t)$ انرژی شبکه در مرحله tام را نشان می‌دهد.

۳- معرفی شبکهٔ عصبی CNN-TSP برای حل TSP

شبکه عصبی پیشنهادی شامل چهار لایه و خروجی نرونهای هر لایه به شکل بردار است. شکل۱ این شبکه را برای سه شهر نشان می‌دهد. در شکل، ورودی‌ها، خروجی‌ها و مقادیر آستانهٔ نرونها نیز نشان داده شده‌اند. در این شکل همانند مدل کوهونن، مسیر در لایهٔ اول (path layer) با N نرون بیان می‌شود ولی وزنهای این نرونها به سمت مختصات شهرها میل نمی‌کند، بلکه به شهرها اجازهٔ جابجایی در آنها داده می‌شود. معادله‌های (۲)و (۳) عملکرد نرونهای این لایه را بیان می‌کنند. تابع انرژی شبکه که در حین آموزش سیر نزولی دارد در معادله (۴) آورده شده است.

$${}^{i}\vec{X}^{1}=\left[{}^{i}x_{j}^{l}\right] \qquad \forall i=1,2,\ldots,N \qquad (۲)$$

$$ {}^{i}x_j^1 = \begin{cases} 1 & \text{If ith city is located at jth location} \\ 0 & \text{Otherwise} \end{cases} \qquad \forall j = 1,2,\ldots,N \tag{۳} $$

$$ e = \sum_{k=1}^{N}\sum_{i=1}^{N}\sum_{j=1}^{N} d_{i,j}\ {}^{k}x_i^1\ {}^{k-1}x_j^1 \tag{۴} $$

لایهٔ دوم (link layer) نیز شامل N نرون است و هر نرون یک کمان از مسیر را مشخص می‌کند، بنابراین داریم:

$$ {}^{i}\vec{X}^2 = {}^{i}\vec{X}^1 + {}^{i+1}\vec{X}^1 \tag{۵} $$

در لایهٔ سوم (link competitive layer)، $N+1$ نرون داریم که یک نرون، نـرون آسـتانه اسـت. هـر کـدام از نرونهـای باقیمانده نظیر یک شهر است (نرون اول نظیر شهر اول، نرون دوم نظیـر شـهر دوم و ...). معادلـه (۶)، عملکـرد نـرون آستانه و معادله‌های (۷)، (۸)، (۹) و (۱۰)، عملکرد سایر نرونهای این لایه را شرح می‌دهند. در این لایه در هر نرون، بین کمانهایِ مختلف مسیر، رقابتی شکل می‌گیرد و کمان برنده و وزن آن به عنوان خروجی نرون معرفی می‌شوند.

$$ \vec{T}^3 = \left\{t_k^3\right\} \qquad ,t_k^3 = \frac{1}{2}\sum_{i=1}^{N}\sum_{i=1}^{N} d_{i,j}\times {}^{k}x_i^2 \times {}^{k}x_j^2 \qquad ,k = 1,2,\cdots N \tag{۶} $$

$$ X = \left[{}^{1}\vec{X}^2 \quad {}^{2}\vec{X}^2 \quad \cdots \quad {}^{N}\vec{X}^2\right] = \left[x_{i,j}\right] \tag{۷} $$

$$ \vec{D}_k = \left\{d_{k,j}\right\} \qquad ,d_{k,j} \in D \tag{۸} $$

$$ {}^{i}\vec{Z}^3 = \vec{D}_i.X - \vec{T}^3 \qquad ,i = 1,2,\cdots,N \tag{۹} $$

$$ {}^{i}\vec{X}^4 = \begin{bmatrix} {}^{i}v^3 \\ {}^{i}k^3 \end{bmatrix} \qquad ,\begin{cases} {}^{i}v^3 = \min\limits_{j=1}^{N}\left({}^{i}z_j^3\right)\Big/x_{i,j} \neq 1 \\ {}^{i}k^3 = index\left({}^{i}v^3\right)_{\vec{z}} \end{cases} \tag{۱۰} $$

که تابع $index(x)_{\vec{Y}}$ محل درایهٔ x در بردار $\vec{Y}$ را برمی‌گرداند و منظور از "/" ، عبارت "به طوری که" است.

در لایهٔ چهارم (path competitive layer) نیز N نرون قرار دارد و همانند لایهٔ سوم هر نرون نظیـر یـک شـهر اسـت (نرون اول نظیر شهر اول، نرون دوم نظیر شهر دوم و ...). نرونهـا دارای وزنهـایی (${}^{i}w^4$, i = 1,2,...,N) هسـتند کـه در پایان هر مرحله از آموزش با توجه به مسیر جدیدی که شبکه معرفی می‌کند، بهنگام خواهند شد و از این لحاظ شـبیه نرونهای شبکهٔ کوهونن هستند. دراینباره در بخش بعدی بیشتر صحبت خواهیم کـرد. معادلـه‌هـای (۱۱)، (۱۲) و (۱۳)، مبین رفتار نرونهای این لایه هستند.

$$ {}^{i}v^4 = {}^{i}v^3 - {}^{i}w^4 \quad , \quad i = 1,2,\cdots,N \tag{۱۱} $$

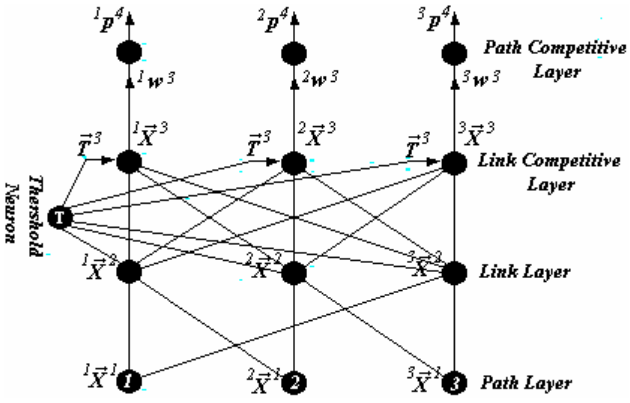
$$ k = \begin{cases} index\left(\min\limits_{i=1}^{N}\left({}^{i}v^4\right)\right) & \min\limits_{i=1}^{N}\left({}^{i}v^4\right) < 0 \quad \text{: The win neuron} \\ 0 & \text{Otherwise} \end{cases} \tag{۱۲} $$

$$ {}^{i}p^4 = \begin{cases} {}^{i}k^3 & i = k \\ 0 & \text{Otherwise} \end{cases} \tag{۱۳} $$

در این لایه نرونی که کمترین وزن را دارد، برنده می‌شود(نرون k) و خروجی‌اش، محلِ جدیـد شـهر هـم‌ارزش را، در مسیر مشخص می‌کند.

۴- الگوریتم آموزش

الگوریتم آموزش شـبکهٔ پیشـنهادی در جـدول۱ آورده شـده اسـت. همـانطور کـه در [18] اثبـات کرده‌ایم، تابع انرژی، طی دوره آموزش سیر نزولـی دارد و به این ترتیب شبکه در نهایـت بـه مسـیری بهتر از مسیر اولیه(و یا همان مسیر، در صورتی کـه در همان مرتبهٔ اول آمـوزش نرونـی برنـده نشـود) همگرا می‌شود. با توجه به اینکـه در هـر مرحلـه از آموزش شبکه، شهری از یک مکـان در مسـیر، بـه مکان دیگری منتقل می‌شود، می‌توان نتیجه گرفت که مسیر نهـایی شـبکه همـان خصوصـیات مسـیر اولیه‌اش را دارد. یعنی اگر مسیر اولیه از تمام نقـاط گذشته باشد، مسیر نهایی نیز از تمام آنهـا خواهـد گذشت و اگر مسـیر اولیـه حلقـه‌ای نداشـته باشـد



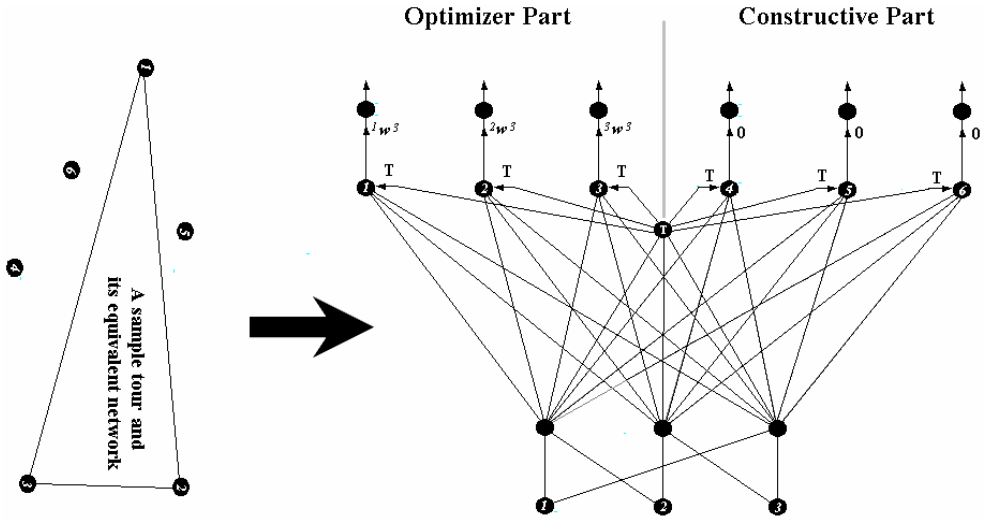
شکل۱: ساختار شبکه عصبی CNN-TSP.

مسیر نهایی نیز حلقه‌ای نخواهد داشت و به عبارت ساده‌تر، اگر مسیر اولیه معتبر باشد، مسیر نهایی نیز معتبر خواهد بود.

جدول ۱: الگوریتم آموزش شبکه عصبی پیشنهادی.
۱. مسیر معتبری را انتخاب می‌کنیم(مسیری معتبر است که از هر شهر تنها یکبار عبور کرده باشد).
۲. خروجی نرونهای لایهٔ اول را با توجه به مسیر اولیه، تعیین می‌کنیم.
۳. وزنهای نرونهای لایهٔ چهارم را طبق معادلهٔ (۱۴) تعیین می‌کنیم. $${}^{t}w^{4}=\sum_{q=1}^{N}\sum_{i=1}^{N}\sum_{j=1}^{N}\left(d_{t,i}+d_{t,j}-d_{i,j}\right)\times^{q}x_{i}^{1}\times^{(q+1)}x_{i}^{1}\times^{(q-1)}x_{j}^{1}$$ (۱۴)
۴. اجازه می‌دهیم شبکه یک مرحله آموزش یابد.
۵. شهر نظیر نرون برنده در لایهٔ چهارم را از محل فعلی‌اش به محل جدیدی که شبکه در خروجی این نرون بیان می‌کند منتقل می‌کنیم. اگر خروجی نرون برنده p باشد، برنده باید در مسیر، بین دو شهری که کمان p ام مسیر جاری را تشکیل می‌دهند قرار گیرد.
۶. اگر در لایهٔ آخر نرونی برنده نشده باشد، شبکه پایدار شده است و آموزش شبکه پایان می‌یابد.
۷. مراحل ۳، ۴، ۵ و ۶ را تا آنجا تکرار می‌کنیم که شبکه پایدار شود.

۵- تعمیم شبکه پیشنهادی به یک شبکه سازنده (Constructive Neural Network)

همانطور که بیان شد پاسخهای این شبکه به مسیر اولیه‌اش وابسته است. اصولا شبکه‌های مبتنی بر تابع انرژی مستعد به بدام افتادن در مینیممهای محلیِ تابعِ انرژیشان هستند. در [18] چندین الگوریتم برای تولید مسیر اولیهٔ مناسب برای این شبکه پیشنهاد کرده و آنها را با هم مقایسه نیز نموده‌ایم. بهترین پاسخها مربوط به روشی بود که شبکه را با استفاده از یک الگوریتم سازنده، به سمت پاسخ شبه بهینه هدایت می‌نمود. اندیشهٔ اصلی این روش این است که در هر مرتبه بر اساس ضابطه‌ای از پیش تعیین شده، از بین شهرهایی که در مسیر قرار ندارند یکی انتخاب شده و در مسیر قرار می‌گیرد و سپس شبکه مسیر جدید را بهینه می‌کند. به عنوان مثال اگر ضابطهٔ مذکور کمترین فاصلهٔ بین شهر و کمانی از مسیر باشد(به خطی که دو شهر پشت سرِهمِ مسیر را، به هم وصل می‌کند یک کمان گوییم)، آنگاه در هر مرحله شهری که فاصله‌اش تا نزدیکترین کمان از بقیه کمتر است، انتخاب شده و در مسیر، در بین دو شهری که نزدیکترین کمان را تشکیل داده‌اند، قرار می‌گیرد. با مطالعه بهترین ضابطه‌ای که در [18]، بدست آمد، دریافتیم که الگوریتمی که بهترین پاسخها را ارائه می‌داد، در حقیقت عملکردی مشابه با خود شبکه داشت و می‌توان با تعمیم شبکه مستقیما آن را پیاده‌سازی نمود. فرض کنید از بین N شهر تنها m شهر ($m<N$) بر روی مسیر قرار گرفته است. بنابراین در لایه‌های اول و دوم تنها m نرون داریم. چنانچه بخواهیم تنها مسیر m شهری موجود را بهینه کنیم کافی است نظیر هر یک از شهرهای موجود بر روی مسیر نرونی در لایه‌های سوم و چهارم در نظر بگیریم. به این ترتیب در هر مرحله از آموزش یکی از نرونهای مسیر از مکانی که در آن قرار دارد به مکان بهتری منتقل می‌شود. اما



شکل۲: یک مسیر سه شهری نمونه و شبکه عصبی نظیر آن.

اگر بخواهیم تعداد شهرهای مسیر را افزایش دهیم چه باید کرد. اکنون فرض کنید که در لایه سوم و چهارم علاوه بر نرونهای قبلی، برای شهرهایی که بر روی مسیر نیستند نرونهایی در نظر گرفته و وزن نرونهای اضافه شده به لایه چهارم را نیز برابر صفر قرار دهیم. همانطور که در بخش پنجم ذکر شد در هر یک از نرونهای لایه سوم، بین کمانهای مختلف مسیر رقابتی شکل می‌گیرد و یکی برنده می‌شود به طوری که اگر آن شهر در مسیر، بین دو شهر تشکیل دهنده کمان برنده قرار گیرد مسیری تشکیل می‌شود که :

۱. مسیر جدید علاوه بر شهرهای قبلی شامل شهر مورد نظر نیز است.

۲. این شهر اگر در هر جای دیگر مسیر قبلی قرار گیرد مسیری طولانی‌تر ایجاد خواهد شد(این مطلب نتیجهٔ مستقیم معادلهٔ(۱۰) است).

اکنون اگر در لایه چهارم تنها بین نرونهای خارج مسیر، رقابتی شکل گیرد بدیهی است که خروجی نرون برنده، بیانگر موقعیت شهر نظیرش در مسیر است؛ به طوری که با قرار گرفتن آن در مسیر، کوتاهترین مسیر ممکن که اولا طولی برابر با $m+1$ دارد، ثانیا همه شهرهای مسیر قبلی را داراست، شکل می‌گیرد. با توجه به مطالب ذکر شده می‌توان شبکه پیشنهادی را مطابق جدول۲ و شکل۲ به یک شبکه عصبی سازنده تبدیل نمود. توجه کنید که در این الگوریتم باید از شکل تصحیح شده معادلات (۱۲) و (۱۳)، که در ذیل آورده شده استفاده کنید. شبکه پیشنهادی را CNN-TSP[۳] می‌نامیم.

$$k_1 = \begin{cases} index\left(\min_i \left({}^i v^4\right)\right) & \min_i \left({}^i v^4\right) < 0 \\ 0 & \text{Otherwise} \end{cases} \quad ,i \in \{\text{cities that is located over the path}\} \qquad (۱۵)$$

$${}^i p^4 = \begin{cases} {}^i k^3 & i = k_1 \\ 0 & \text{Otherwise} \end{cases} \quad ,i \in \{\text{cities that is located over the path}\} \qquad (۱۶)$$

$$k_2 = index\left(\min_i \left({}^i v^4\right)\right) \quad ,i \in \{\text{cities that is not located over the path}\} \qquad (۱۷)$$

$${}^i p^4 = \begin{cases} {}^i k^3 & i = k_2 \\ 0 & \text{Otherwise} \end{cases} \quad ,i \in \{\text{cities that is not located over the path}\} \qquad (۱۸)$$

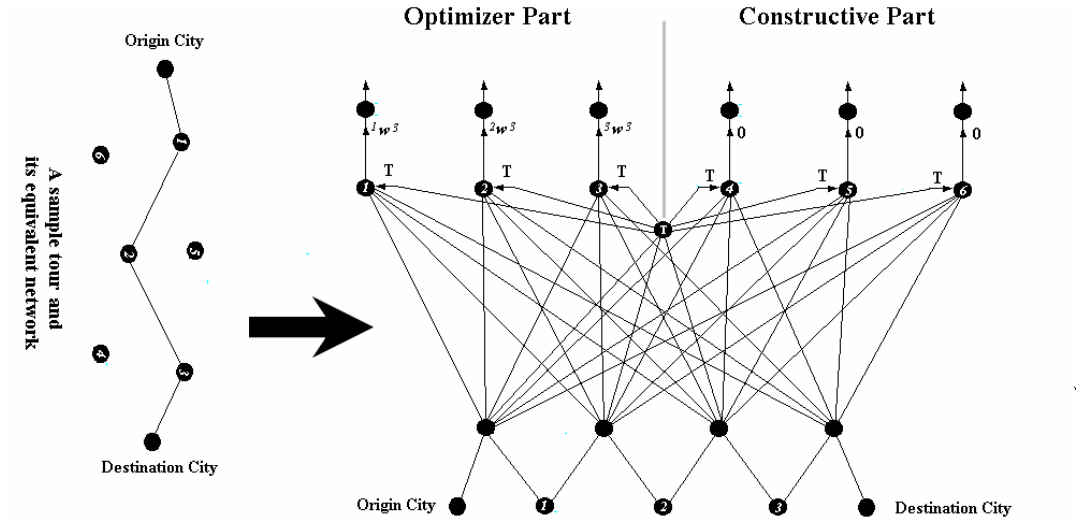
جدول۲: الگوریتم آموزش شکل تعمیم یافته شبکه پیشنهادی.
۱. با چهار شهر خارجی(خارجی‌ترین شهرها) یک مسیر اولیهٔ چهار شهری بسته ایجاد کرده، شبکه را بر اساس آن مقداردهی اولیه کنید. در این حالت در لایه‌های اول و دوم چهار نرون و در لایه‌های سوم و چهارم N نرون داریم.
۲. نرونهای لایه‌های سوم و چهارم را به دو دسته تقسیم نمایید: یک دسته نرونهایی که بر روی مسیر قرار دارند و دسته دیگر نرونهای خارج از مسیر.
۳. در لایه چهارم وزن نرونهای دسته اول را طبق معادله (۱۴) تعیین کنید و وزن نرونهای دسته دوم را برابر صفر قرار دهید. رقابت در لایه چهارم در هر دسته به طور جداگانه انجام خواهد شد. بنابراین الگوریتم آموزش شبکه دارای دو فاز خواهد بود: فاز اول که طی آن مسیر موجود بهبود می‌یابد و مربوط به نرونهای دسته اول است و فاز دوم که طی آن از بین نرونهای دسته دوم، بهترین نرون برنده می‌شود و در مسیر قرار می‌گیرد.
۴. آموزش شبکه را در فاز اول آنقدر ادامه می‌دهیم که شبکه پایدار گردد.
۵. اگر تمام شهرها در مسیر قرار گرفته‌اند، آموزش خاتمه یافته است.
۶. شبکه را در فاز دوم یک مرحله آموزش می‌دهیم. نرون برنده را در مسیر قرار می‌دهیم. به لایه‌های اول و دوم یک نرون نظیر با شهر برنده، اضافه کرده و در لایه‌های سوم و چهارم نرونهای نظیر شهر برنده را از دسته دوم به دسته اول منتقل می‌کنیم.
۷. چهار مرتبه دیگر مرحله ۶ را تکرار کرده، سپس به مرحله ۳ بروید.

۶- کاربرد CNN-TSP در حل مسأله کوتاهترین مسیر

ساختار رقابتی CNN-TSP به آن قابلیت انعطاف زیادی می‌بخشد؛ بطوری که می‌توان با کمی تغییرات، از آن برای حل دیگر مسائل NP-Complete استفاده کرد. در اینجا نحوه حل مسأله کوتاهترین مسیر m شهری را با استفاده از شبکه عصبی پیشنهادی بیان می‌کنیم. برای حل SP، ساختار CNN-TSP را مطابق شکل3 تغییر می‌دهیم. نرونهای اول و آخر به ترتیب نظیر با شهرهای مبدا و مقصد هستند. از آنجا که این دو شهر همواره در ابتدا و انتهای مسیر قرار دارند در شبکه به صورت ثابت قرار می‌گیرند. بنابراین تعداد کمانها یکی کمتر از تعداد شهرها در لایه اول خواهد بود. سایر

[3] Constructive Neural Net to solve TSP (CNN-TSP)

لایه‌های شبکه (لایه‌های سوم و چهارم) بدون تغییر باقی می‌مانند. باید دقت داشت که برای شهرهای مبداء و مقصد در این دو لایه نرونی در نظر گرفته نمی‌شود. الگوریتم آموزش این شبکه همانند CNN-TSP است. ما این شبکه را CNN-SP می‌نامیم. نکته جالب در این است که چنانچه از بین N شهر موجود یکی را هم به عنوان مبداء و هم به عنوان مقصد در نظر بگیریم و از شبکه برای یافتن کوتاهترین مسیر N-1 شهری بین مبداء و مقصد مذکور استفاده کنیم در حقیقت شبکه به ما پاسخی برای مسأله‌ی TSPای نظیر نقاط N نقطه مفروض داده است. به عبارت دیگر توانسته‌ایم از CNN-SP برای حل مسأله فروشنده دوره‌گرد نیز استفاده نماییم. می‌توان از این خاصیت برای بررسی دقت CNN-SP استفاده نمود. بدیهی است که زمانی که از CNN-SP برای حل TSP استفاده می‌کنیم، شبکه بیشترین خطای ممکن را از خود نشان می‌دهد؛ زیرا مسیر در این حالت دارای بیشترین تعداد شهر است.



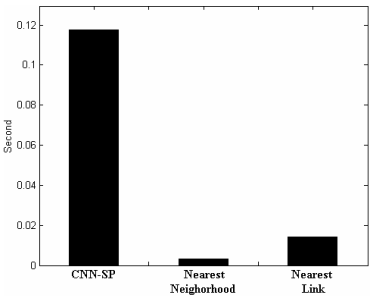
شکل۳: ساختار تصحیح شده شبکه عصبی پیشنهادی برای حل SP.

۷- شبیه‌سازی

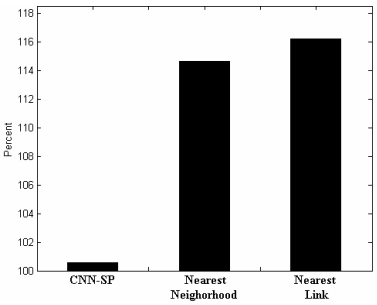
شبیه‌سازیهای انجام شده شامل دو بخش است. در بخش اول عملکرد CNN-SP با دو الگوریتم کلاسیک مقایسه گردیده است. سپس در بخش دوم دقت CNN-SP با دقت CNN-TSP مقایسه می‌شود. شبیه‌سازیها با استفاده از کامپیوتر شخصی AMD Athlon 1600MHz، تحت نرم‌افزار Matlab انجام شده است.

در ابتدا پاسخهای الگوریتم CNN-SP را برای یافتن بهترین مسیر ۵۰ شهری از بین ۱۰۰ شهر، با پاسخهای الگوریتم نزدیکترین همسایه و نزدیکترین کمان مقایسه نموده‌ایم. در الگوریتم نزدیکترین همسایه، در هر مرحله از محلی که در آن قرار داریم به نزدیکترین شهر می‌رویم؛ در حالی که در الگوریتم نزدیکترین کمان در هر مرحله فاصله تمام شهرهای خارج مسیر تا مراکز کمانهای مسیر محاسبه می‌شود و شهری که کمترین فاصله را دارد در مسیر قرار داده می‌شود. برای ۲۰۰ توزیع ۱۰۰ شهری، پاسخها و زمان همگرایی هر سه الگوریتم محاسبه گردید. از آنجا که طول مسیرها، وابسته به توزیع شهرهاست، لازم است که قبل از میانگین‌گیری، آنها را نرمال کرد. در هر مرحله برای هر توزیع سه مسیر بدست می‌آید، یکی توسط شبکه عصبی و دو مسیر دیگر توسط الگوریتمهای نزدیکترین همسایه و نزدیکترین کمان که یک مجموعه سه‌تایی را تشکیل می‌دهند. برای نرمال کردن طول این مسیرها، در هر مجموعه از بین سه مسیر، کوتاهترین مسیر انتخاب و طول هر سه مسیر بر طول آن تقسیم می‌شود. در پایان از طول نرمال شده همه مسیرهای تولید شده توسط هر الگوریتمها میانگین می‌گیریم. بدیهی است که اعداد حاصل معیاری هستند از طول مسیرهای تولید شده توسط هر یک از الگوریتمها نسبت به دیگری. نتایج در شکلهای ۴ و ۵ نشان داده شده است. همانطور که در شکل۴ می‌بینید، پاسخهای شبکه به مراتب از دو الگوریتم دیگر بهتر است و تقریبا عدد ۱۰۰٪ را نشان می‌دهد؛ در حالی که نزدیکترین ستون که مربوط به الگوریتم نزدیکترین همسایه است بیش از ۱۱۴٪ را نشان می‌دهد. البته در ازای این بهبود ما هزینه‌ای هم پرداخته‌ایم و آن افزایش زمان محاسباتی است که در شکل۵ نشان داده شده است هرچند که هنوز هم زمان محاسباتی ما کمتر از ۰،۱۲ ثانیه است که عددی قابل قبول است.

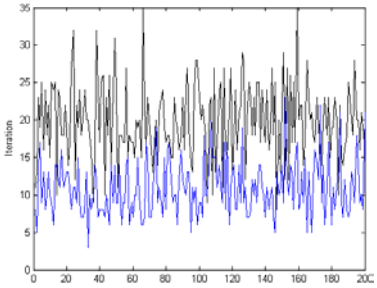
اکنون از همان ۲۰۰ توزیع ۱۰۰ شهری قبلی، برای مقایسه عملکرد CNN-TSP و CNN-SP استفاده می‌کنیم. اینبار هدف یافتن بهترین پاسخ برای TSPای نظیر با هر توزیع است. نتایج در شکلهای ۶، ۷ و ۸ نشان داده شده است.



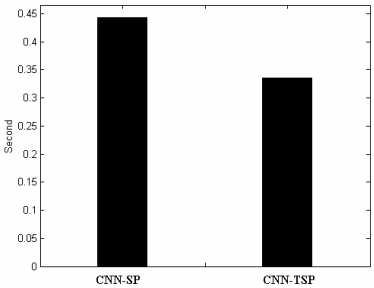
شکل۵: زمان همگرایی الگوریتمهای CNN-SP (میله اول)، نزدیکترین همسایه (میله دوم) و نزدیکترین کمان (میله سوم) بر حسب ثانیه.



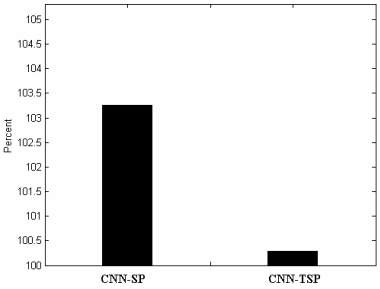
شکل۴ :طول متوسط مسیرهای ۵۰ شهری پیشنهاد شده توسط الگوریتمهای CNN-SP (میله اول)، نزدیکترین همسایه (میله دوم) و نزدیکترین کمان (میله سوم) نسبت به طول مسیر کوتاهتر برای ۲۰۰ توزیع ۱۰۰ شهری.



شکل۸: تعداد تکرارهای CNN-SP (منحنی تیره بالا) و CNN-TSP (منحنی روشنتر پایین) در فاز بهینه‌سازی برای هر توزیع.



شکل۷: زمان همگرایی شبکه‌های CNN-SP (میله اول) و CNN-TSP (میله دوم) بر حسب ثانیه.



شکل۶ :طول متوسط مسیرهای ۱۰۰ شهری پیشنهاد شده توسط الگوریتمهای CNN-SP (میله اول) و CNN-TSP(میله دوم) نسبت به طول مسیر کوتاهتر برای ۲۰۰ توزیع ۱۰۰ شهری.

همانطور که در شکل۶ مشاهده می‌کنید، طول مسیرهای معرفی شده توسط CNN-SP تقریبا به اندازه ۳٪ از طول مسیرهای CNN-TSP بلندتر است. همچنین با مراجعه به شکل۷ در می‌یابیم که زمان محاسباتی CNN-SP، تقریبا ۰،۱ ثانیه طولانی‌تر بوده است. با توجه به اینکه ساختار هر دو شبکه یکسان است، عاملی سبب چنین اختلافی شده است، مسیر اولیه چهار شهری است که CNN-TSP را با آن مقدار دهی اولیه می‌کنیم. این مسیر اولیه که شامل خارجی‌ترین نقاط توزیع است، شبکه را به سمت پاشخهای بهینه‌تری هدایت می‌کند. در حقیقت CNN-TSP برای حل TSP بهینه گردیده و دقت بیشتر آن نسبت به CNN-SP، امری قابل پیشبینی بود. طولانیتر بودن زمان همگرایی CNN-SP نیز به دلیل بیشتر بودن تعداد تکرارهای شبکه در فاز بهینه‌سازی، نسبت به CNN-TSP است. این مطلب در شکل۸ نشان داده شده است. طبق شبیه‌سازیهایی که قبلا بر روی مسائل کتابخانه TSPLIB [25] انجام شده است، دقت CNN-TSP در حدود ۳،۶۰٪ تخمین زده شد؛ بنابراین دقت CNN-SP در بدترین حالت در حدود ۶،۶٪ خواهد بود که دقت قابل قبولی است.

۸- نتیجه‌گیری

در این مقاله یک شبکه عصبی CNN-TSP برای حل مسأله SP تعمیم داده شده است. در طراحی CNN-TSP سعی گردیده با بهره‌گیری از ویژگیهای اساسی شبکه‌های عصبی هاپفیلد و کوهونن، مزایای هریک حفظ و از معایب آنها پرهیز شود. شبیه‌سازیها نشان داد که شبکه تعمیم‌یافته از دقت و سرعت قابل قبولی برخوردار است و از این رو می‌توان از آن در کاربردهای بلادرنگ سود جست.

۹- مراجع

[1] P. Crescenzi, V. Kann, “A compendium of NP optimization problems”, Oct. 1995. *http:// www.zvne.fer.hr/ ~zmija/ resources/ science_resources/nn_for_optimization/index.html*

[2] K. Fujimura, K. Obu-Cann, H. Tokutaku, “Optimization of surface component mounting on the printed circuit board using SOM-TSP method”, 9th International Conference on Artificial Neural Networks, Vol. 2, No. 470, 1999.
[3] K. Fujimura, S. Fujiwaki, O.-C. Kwaw, H. Tokutaka, “Optimization of electronic chip-mounting machine using SOM-TSP method with 5 dimensional data”, International Conferences on Info-tech and Info-net (ICII), 4: 26-31, 2001.
[4] M. K. Mehmet Ali, F. Kamoun, “Neural networks for shortest path computation and routing in computer networks”, IEEE Trans. Neural Networks, 4(6), Nov. 1993.
[5] T. Onoyama, T. Maekawa, S. Kubota, Y. Taniguchi, S. Tsuruta, “Intelligent evolutional algorithm for distribution network optimization”, Proceedings of the International Conference on Control Applications, 2: 802 -807, 2002.
[6] D. Banaszak, G.A. Dale, A.N. Watkins, J.D. Jordan, “An optical technique for detecting fatigue cracks in aerospace structures”, 18th International Congress on Instrumentation in Aerospace Simulation Facilities (ICIASF), pp. 27/1-27/7, 1999.
[7] D. Barrel, J.-P. Perrin, E. Dombre, A. Liengeois, “An evolutionary simulated annealing algorithm for optimizing robotic task ordering”, Proceedings of the IEEE International Symposium on Assembly and Task Planning (ISATP), pp. 157 -162, Jul.1999.
[8] C.-H. Cheng, W.-K. Lee, K.-F. Wong, “A genetic algorithm-based clustering approach for database partitioning”, IEEE Trans. on Systems, Man, and Cybernetics-Part C: Applications and Reviews, Vol. 32, No. 3, Aug. 2002.
[9] J. Gu, X. Huang, “Efficient local search with search space smoothing: a case study of the traveling salesman problem (TSP)”, IEEE Trans. on Systems, Man, and Cybernetics, Vol. 24, No. 5, May 1994.
[10] F. Tian, L. Wang, “Chaotic simulated annealing with augmented Lagrange for solving combinatorial optimization problems”, 26th Annual Conference of the IEEE Industrial Electronics Society (IECON), 4: 2722 -2725, 2000.
[11] L. Jiao, L. Wang, “A novel genetic algorithm based on immunity”, IEEE Trans. on Systems, Man, and Cybernetics-Part A: Systems and Humans, Vol. 30, No. 5, Sep. 2000.
[12] R. Baraglia, J. I. Hidalgo, R. Perego, “A hybrid heuristic for the traveling salesman problem”, IEEE Trans. on Evolutionary Computation, Vol. 5, No. 6, Dec. 2001.
[13] D. B. Fogel, “Applying evolutionary programming to selected traveling salesman problems”, Cybernetics and Systems, Vol. 24, pp. 27-36, 1993.
[14] T. Stützle, M. Dorigo, “ACO algorithms for the traveling salesman problem”, In K. Miettinen, M. Makela, P. Neittaanmaki, J. Periaux, editors, Evolutionary Algorithms in Engineering and Computer Science, Wiley, 1999.
[15] M. Dorigo, L. M. Gambardella, “Ant colonies for the traveling salesman problem”, BioSystems, Vol. 43, pp. 73-81, 1997.
[16] Z. He, C. Wei, B. Jin, W. Pei, L. Yang, “A new population-based incremental learning method for the traveling salesman problem”, Proceedings of the 1999 Congress on Evolutionary Computation, 1999.
[17] S. P. Coy, B. L. Golden, G. C. Runger, E. A. Wasil, “See the forest before the trees: fine-tuned learning and its application to the traveling salesman problem”, IEEE Trans. on Systems, Man, and Cybernetics-Part A: Systems and Humans, Vol. 28, No. 4, July 1998.
[18] M. Saadatmand-T., M.-R. Akbarzadeh-T., M. Khademi, “A novel hybrid neural network for the traveling salesman problem (TSP)”, 9th international Iranian Conference on Electrical Engineering (ICEE), May 2001.
[19] H.-D. Jin, K.-S. Leung, M.-L. Wong, Z.-B. Xu, “An efficient self-organizing map designed by genetic algorithms for the traveling salesman problem”, IEEE Trans. on Systems, Man, and Cybernetics-Part B: Cybernetics, Accepted for future publication, 2003.
[20] Y. Takahashi, “Mathematical improvement of the Hopfield model for feasible solution to the traveling salesman problem by a synapse dynamical system”, IEEE Trans. on Systems, Man, Cybernetics-Part B: Cybernetics, Vol. 28, No.6, Dec. 1998.
[21] S. Abe, A. H. Gee, “Global convergence of the Hopfield neural network with nonzero diagonal elements”, IEEE Trans. on Circuits and Systems II: Analog and Digital Signal Processing, Vol. 42, No. 1, Jan. 1995.
[22] S. Bhide, N. John, M. R. Kabuka, “A real-time solution for the traveling salesman problem using a boolean neural network”, ICNN, Mar. 1993.
[23] Y. He, L. Wang, “Chaotic neural networks and their applications”, Proceedings of the 3rd World Congress on Intelligent Control and Automation, 2: 826 -830, 2000.
[24] J. Häkkinen, M. Lagerholm, C. Peterson, B. Söderberg, “Local routing algorithms based on Potts neural networks”, IEEE Trans. on Neural Networks, Vol. 11, No. 4, July 2000.
[25] B. Bixby, G. Reinelt, “TSPLIB-A library of traveling salesman and related problem instances”, Feb. 1995. *ftp://softlib.rice.edu/pub/tsplib/tsp/*