یک شبکه عصبی جدید با ساختاری سازنده-ترکیبی برای حل مساله‌های فروشنده دوره‌گرد و کوتاهترین مسیر با تعداد شهر مشخص

مهدی سعادتمند طرزجان
دانشجوی دکتری مهندسی پزشکی- بیوالکتریک – دانشکده فنی - دانشگاه تربیت مدرس
saadatmand@kiaeee.org
محمد رضا اکبرزاده توتونچی
استادیار گروه برق - دانشکده مهندسی - دانشگاه فردوسی مشهد و دانشگاه نیومکزیکو
akbazar@eece.unm.edu
مرتضی خادمی
استادیار گروه برق - دانشکده مهندسی - دانشگاه فردوسی مشهد
khademi@um.ac.ir
(تاریخ دریافت ۸۲/۱۰/۲۰ ، تاریخ دریافت روایت اصلاح شده ۸۴/۲/۲۷ ، تاریخ تصویب ۸۴/۸/۷)

چکیده

در این مقاله یک شبکه عصبی سازنده جدید برای حل مسأله فروشنده دوره‌گرد (TSP) ارائه شده است. ساختار فیدبکی-رقابتی این شبکه از مفاهیم شبکه‌های عصبی هاپفیلد و کوهونن الهام گرفته شده است. شبکه کوهونن با شیوه یادگیری رقابتی‌اش پاسخ‌های قابل قبولی به TSP ارائه می‌دهد اما سرعت همگرایی آن بسیار کم است. در مقابل، شبکه عصبی هاپفیلد با ساختار فیدبکی خود، دارای سرعت همگرایی مناسبی است اما پاسخ‌های آن از دقت کمی برخوردار است. در شبکه عصبی پیشنهادی برای دستیابی به مزایای شبکه‌های هاپفیلد و کوهونن یعنی سرعت همگرایی مناسب و دقت قابل قبول، شیوه یادگیری رقابتی کوهونن و ساختار فیدبکی هاپفیلد ترکیب شده‌اند. نتایج تجربی نشان می‌دهد که شبکه پیشنهادی قادر است ظرف مدت کوتاهی، پاسخ‌هایی مناسب به TSP ارائه دهد؛ بطوری که بر اساس شبیه‌سازی‌های انجام شده، سرعت همگرایی شبکه تقریبا ۲۰ برابر سرعت همگرایی شبکه کوهونن و تفاوت متوسط طول مسیر آن برای ۲۹ مسأله استاندارد از کتابخانه TSPLIB (نسبت به پاسخ‌های بهینه‌ای که در همین کتابخانه ارائه شده)، ۳/۸۱٪ است. همچنین، شبکه پیشنهادی در مقایسه با روش‌های محک متداول شامل آبکاری شبیه‌سازی شده و نگاشت خود سازمانده Budinich's SOM، عملکرد قابل قبولی از خود نشان داده است. بعلاوه، شبکه پیشنهادی بسیار انعطاف‌پذیر می‌باشد. تا آنجا که می‌توان با کمی تنظیم ساختار، از آن برای حل سایر مسائل بهینه‌سازی استفاده نمود. به عنوان مثال، در این مقاله با توسعه ساختار شبکه پیشنهادی، از آن برای حل مسأله «کوتاهترین مسیر با تعداد شهر مشخص» نیز استفاده شده است.

واژه‌های کلیدی: مسأله فروشنده دوره‌گرد (TSP)، مسأله کوتاهترین مسیر با تعداد شهر مشخص (SPSN)، نگاشت خودسازمانده کوهونن، شبکه عصبی هاپفیلد

مقدمه

مسایل بهینه‌سازی ترکیبی[1]، از قبیل مسأله فروشنده دوره‌گرد[2] (TSP) و کوتاهترین مسیر[3] (SP)، از خانواده مسایل NP-Complete هستند [۱]. مدت زمان لازم برای حل این نوع مسائل با افزایش تعداد پارامترها بصورت نمایی افزایش می‌یابد و این درحالی است که با پیشرفت تکنولوژی، نیاز به حل سریعتر و مناسبتر این مسایل بسرعت رو به افزایش است. تعیین مسیر بهینه حرکت مته برای سوراخ کردن صفحه‌های PCB [۲][۳]، تعیین مسیر بهینه انتقال داده در شبکه‌های کامپیوتری [۴]، بهینه‌سازی شبکه‌های توزیع نیرو [۵]، پردازش تصویر و تشخیص الگو [۶]، هدایت روبات [۷]، تفکیک و دسته‌بندی داده‌ها[4] [۸] از جمله زمینه‌هایی هستند که حل TSP برایشان بسیار پراهمیت است.

TSP، همانطور که از نامش بر می‌آید، عبارت است از یافتن کوتاهترین مسیر بسته ممکن بین N شهر. روش‌هایی که تاکنون برای حل این مسأله ارائه شده‌اند،

معمولا بر پایه جستجوگرها و آشکارسازهای معین[۵] یا احتمالی[۶] هستند که از آنجمله می‌توان به الگوریتم‌های کلاسیک جستجوی محلی [۹]؛ آبکاری شبیه‌سازی شده (SA)[۷] [۱۰]؛ شبکه‌های عصبی مصنوعی مانند کوهونن یا نگاشت‌های خود سازمانده[۸] [۱۱]، هاپفیلد [۱۲][۱۳]، بولین[۹] [۱۴] و آشوبی[۱۰] [۱۵]؛ الگوریتم‌های ژنتیکی[۱۱] [۱۶][۱۷]؛ برنامه‌نویسی تکاملی[۱۲] [۱۸]؛ سیستم کولونی مورچه‌ها (ACO)[۱۳] [۱۹][۲۰]؛ روش‌های یادگیری افزایشی مبتنی بر جمعیت[۱۴] [۲۱] و روش‌های آموزش با تنظیم ظریف[۱۵] [۲۲] اشاره نمود. هر یک از این روش‌ها دارای نقاط ضعف و قوت خاص خود است. به عنوان مثال، از روش‌های تکاملی مانند GA علیرغم اینکه دقت آنها از دیگر الگوریتم‌ها چون SA و ACO بیشتر است [۱۹][۲۳]، معمولا به دلیل حجم زیاد محاسبات، بیشتر در کاربردهای برون خط[۱۶] استفاده می‌شود. همچنین با اینکه SA و ACO نسبت به الگوریتم‌های تکاملی سریعتر هستند اما باز هم در مقایسه با شبکه‌های عصبی مصنوعی کند محسوب می‌شوند. بنابراین، می‌توان گفت که در میان الگوریتم‌های مذکور معمولا شبکه‌های عصبی مصنوعی از بقیه سریعتر بوده و لذا برای کاربردهای بلادرنگ[۱۷] در زمره بهترین انتخاب‌ها هستند، هرچند که دقت آنها کمتر است.

با توجه به اینکه شبکه پیشنهادی از دو شبکه عصبی هاپفیلد و کوهونن الهام گرفته شده، در این مقاله، از میان شبکه‌های فوق تنها این دو شبکه، مورد بررسی قرار گرفته‌اند. هریک از شبکه‌های کوهونن و هاپفیلد دارای مزایا و معایبی هستند. مدل هاپفیلد دارای دو مشخصه اساسی است. اول اینکه دارای تابع انرژی است که مشخصات فیزیکی شبکه و رفتار آن را بیان می‌کند و دوم اینکه به صورت یک فیدبک منفی عمل کرده و در طول آموزش، تابع انرژی آن سیر نزولی دارد. در حالی که مشخصه اساسی مدل کوهونن شیوه یادگیری رقابتی آن است. با اینکه سرعت همگرایی و دقت کوهونن به مقدار پارامترهایش بستگی دارد ولی در مجموع می‌توان گفت که سرعتش کم است و به همین دلیل نمی‌توان از آن در کاربردهای بلادرنگ استفاده نمود. در مقابل، مدل هاپفیلد از سرعت قابل قبولی برخوردار است و از این جهت برای استفاده بلادرنگ مناسب است ولی اشکال اساسی آن وجود مینیمم‌های محلی در تابع انرژی است. زیرا همانطور که در بخش بیان TSP به صورت یک تابع خطی توضیح داده خواهد شد، با اینکه TSP ذاتا مسأله‌ای خطی است، ولی تابع انرژی این شبکه‌ها عموما مجموعی از چندجمله‌ای‌های درجه دو است (به معادله ۱۵ مراجعه کنید). پس یک مسأله خطی توسط یک تابع درجه دو مدل گردیده است [۲۶]. در نتیجه، تعداد مینیمم‌های محلی تابع انرژی بشدت افزایش می‌یابد و به همین دلیل پاسخ‌های شبکه هاپفیلد چندان معتبر نمی‌باشند. همچنین در هر بخش از تابع انرژی، ثابتی ضرب می‌شود که بیانگر اهمیت آن جمله نسبت به دیگر جملات است (ثابت‌های A، B، C و D در معادله ۱۴). این ایده به ظاهر ساده، خود منشأ اشکال اساسی دیگری است و آن حساسیت زیاد شبکه به پارامترهای تابع انرژی است که با افزایش تعداد شهرهای مسیر بیشتر هم می‌شود [۱۲][۲۷].

تاکنون تلاش‌های زیادی جهت فرار از مینیمم‌های محلی و یا حذف آنها از تابع انرژی در شبکه‌های مبتنی بر هاپفیلد انجام شده است. مثلا در [۲۸] سعی شده است که مینیمم‌های محلی تابع انرژی، با اضافه کردن یک ورودی تصحیح کننده و قابل تطبیق به نرون‌ها، حذف شوند. ولی علیرغم بهبودهای قابل توجه در این زمینه، شبکه‌ها هنوز هم به شدت به پارامترهای تابع انرژی حساس هستند. در حقیقت اکثر تلاش‌هایی که تاکنون در این زمینه انجام شده به پیچیده‌تر شدن تابع انرژی و شبکه منجر گردیده است. در حالی که یک راه مناسب برای کم کردن اثر مینیمم‌های محلی، ساده‌تر کردن تابع انرژی شبکه عصبی است.

در این مقاله سعی شده است با تلفیق این دو ایده و ارایه یک شبکه عصبی جدید، از قابلیت‌های این دو شبکه به صورت توأم استفاده شود. شبکه عصبی پیشنهادی از توسعه شبکه ارائه شده در [۲۴] و [۲۵] به یک شبکه سازنده حاصل شده است. شبکه سازنده پیشنهادی بسیار انعطاف‌پذیر است به طوری که با توسعه ساختار، می‌توان از آن در حل دیگر مسائل بهینه‌سازی ترکیبی سود جست. ما از این قابلیت شبکه برای حل مسأله کوتاهترین مسیر با تعداد شهر مشخص[۱۸] (SPSN) استفاده کرده‌ایم. در مقایسه با TSP، محققین کمتر به حل مسأله SPSN پرداخته‌اند. زیرا SPSN با داشتن محدودیت‌هایی از قبیل شهر مبدا، شهر مقصد و تعداد شهر مشخص به مراتب از

TSP پیچیده‌تر است.

شبیه‌سازی‌ها به مقایسه عملکرد شبکه عصبی پیشنهادی و شبکه کوهونن می‌پردازند. همچنین پاسخ‌های شبکه به ۲۹ مسأله استاندارد از کتابخانه TSPLIB بدست آمده و با پاسخ‌های بهینه‌ای که در این کتابخانه ذکر گردیده، مقایسه شده است. خطای نسبی متوسط شبکه طی این شبیه‌سازی‌ها معادل ۳/۸۱٪ تخمین زده می‌شود. بعلاوه، عملکرد شبکه پیشنهادی در حل SPSN نیز در مقایسه با روش‌های کلاسیک از قبیل نزدیکترین همسایه و نزدیکترین کمان قابل قبول می‌باشد.

این مقاله، در ادامه، به شرح ذیل تنظیم شده است. در بخش دوم روش حل TSP به کمک شبکه عصبی کوهونن ذکر شده است. در بخش سوم نمادهایی که در نمایش کمیت‌ها و معادلات بکار رفته، بیان گردیده است. در بخش چهارم TSP را بصورت یک مسأله خطی نشان داده‌ایم. بخش پنجم به شرح شبکه عصبی پیشنهادی اولیه، الگوریتم آموزش آن، نحوه توسعه آن به یک شبکه سازنده و کاربرد آن در حل مساله SPSN اختصاص یافته است. بخش ششم به شرح نتایج تجربی می‌پردازد و در نهایت، در بخش هفتم نیز نتیجه‌گیری ذکر گردیده است. در انتها نیز، ضمیمه الف شامل اثبات همگرایی شبکه پیشنهادی می‌باشد.

جدول ۱: الگوریتم آموزش شبکه کوهونن.

۱. مختصات شهرها نرمال شود به طوری که تمام مؤلفه‌های آنها در بازه $(0,1)$ قرار گیرند.

۲. وزن‌های اولیه نرون‌ها به صورت تصادفی چنان انتخاب شوند که تمام سطح مربع واحد را بپوشانند.

۳. مقادیر مناسبی برای Tr(شرط خاتمه آموزش)، η_0 (سرعت همگرایی شبکه)، G_0 (همسایگی اولیه) و α (سرعت کاهش همسایگی) انتخاب شود. بهترین مقادیر برای این پارامترها را باید برای تعداد شهرهای مختلف با آزمایش و خطا بدست آورد.

۴. k برابر یک قرار داده شود.

۵. مجموعه‌های C و W به ترتیب شامل همه شهرها و همه نرون‌ها می‌باشند. بعلاوه، E_k برابر صفر قرار داده شود.

۶. شهر i با مختصات $\left(x_1^i, x_2^i\right)$ به صورت تصادفی از مجموعه C انتخاب و به شبکه داده شود. در این حالت، بین نرون‌ها رقابتی شکل می‌گیرد و نرونی که کمترین فاصله را تا این شهر دارد، برنده می‌گردد (فرض کنید نرون w برنده شده است).

$$v_j^i = \sqrt{\left(x_1^i - w_1^j\right)^2 + \left(x_2^i - w_2^j\right)^2} \qquad , j = 1,2,\cdots,N \qquad (۱)$$

$$v_w^i = \min_{j=1}^{N}\left(v_j^i\right) \qquad (۲)$$

۷. وزن‌های نرون w و نرون‌های همسایه‌اش به سمت مختصات شهر i در صفحه طبق معادلات ذیل انتقال داده شود.

$$d_{j,w} = \min\left(|w-j|, |w+(N-j)|, |j+(N-w)|\right) \qquad , j = 1,2,\cdots,N \qquad (۳)$$

$$w_p^j = w_p^j + \eta_0 \times \exp\left(-d_{j,w}^{\,2} \Big/ G^j\right) \times \left(x_k^i - w_k^j\right) \qquad (۴)$$

$$G^j = G_k \times \left(1 - v_j^i / \sqrt{2}\right) \qquad (۵)$$

$$G_k = G_{k-1} \times (1-\alpha) \qquad , 0 < \alpha < 1 \qquad (۶)$$

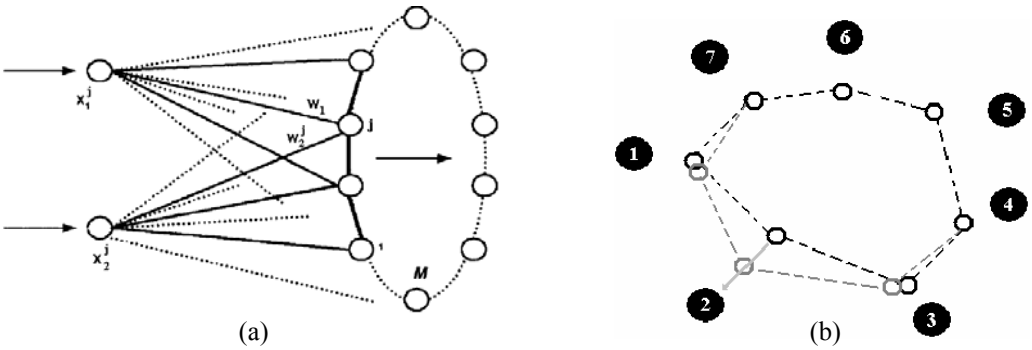
که $d_{j,w}$ فاصله نرون i تا نرون w در شبکه عصبی و G_k پارامتر همسایگی است که در حین آموزش کاهش می‌یابد. با کاهش G_k شبکه همگرا شده و به سمت پاسخ نهایی‌اش میل می‌کند.

۸. نرون w از مجموعه W و شهر i از مجموعه C حذف و پارامتر E_k مطابق معادله ذیل بهنگام گردد.

$$E_k = E_k + v_w^i \qquad (۷)$$

۹. اگر مجموعه‌های W و C تهی نیستند به مرحله ۶ رجوع شود.

۱۰. اگر $E_k < Tr$، آموزش پایان یافته است و در غیر این صورت $k = k+1$ و به مرحله ۵ رجوع شود.



شکل ۱: (a) ساختار شبکه کوهونن برای حل TSP: این شبکه دارای دولایه است (لایه‌های ورودی و خروجی). در لایه خروجی، هر نرون دارای دو وزن است که مختصات آن را در صفحه بیان می‌کند. (b) نحوه آموزش شبکه: در هر مرحله، مختصات یک شهر به عنوان ورودی به شبکه داده می‌شود و وزن‌های نرون برنده و همسایه‌هایش، به سمت آن مختصات شهر میل داده می‌شود. در این شکل شهرها با دوایر توپر و نرونها با دوایر توخالی نشان داده شده‌اند.

حل TSP بوسیله شبکه عصبی کوهونن

شبکه عصبی که در این بخش به معرفی آن می‌پردازیم، از نظر ساختار و نحوه آموزش مشابه با شبکه‌هایی است که در [۲۹] و [۳۰] ارائه شده است. این شبکه همانطور که در شکل (۱) نشان داده شده است، دارای لایه‌های ورودی و خروجی است. در لایه خروجی به ازای هر شهر یک نرون در نظر گرفته می‌شود و هر نرون دارای دو وزن است که مختصات آن نرون را در صفحه بیان می‌کند. مطابق جدول (۱)، در حین آموزش، وزن‌های مذکور به سمت مختصات شهرها میل کرده، در نهایت هر نرون منطبق بر یک شهر خواهد شد. آموزش این شبکه بر پایه این اندیشه استوار است که در هر مرحله مختصات شهری به عنوان ورودی به شبکه داده می‌شود و وزن‌های نرون برنده و همسایه‌هایش به سمت مختصات شهر مذکور میل می‌کنند. سپس نرون برنده از مجموعه نرون‌هایی که برای شهر بعدی در رقابت شرکت می‌کنند حذف می‌شود. در شکل (۱) و جدول (۱) به ترتیب ساختار شبکه نگاشت خودسازمانده کوهونن و الگوریتم آموزش آن نشان داده شده است.

قرارداد

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

- N بیانگر تعداد کل شهرها است.
- M تعداد شهرهای روی مسیر را نشان می‌دهد.
- به خطی که دو شهر متوالی مسیر را به هم وصل می‌کند، یک کمان گوییم.
- بردارها و ماتریس‌ها با حروف بزرگ و درایه‌های آنها با حروف کوچک نشان داده می‌شوند. بعلاوه، بردارها با علامت بردار مشخص می‌شوند، مانند: $\vec{X}$.
- درایه واقع در سطر i وستون j از ماتریس D را با $d_{i,j}$ نشان می‌دهیم.
- همچنین، درایه j از بردار $\vec{D}_i$ نیز با $d_{i,j}$ نشان داده می‌شود. بنابراین، با توجه به قرارداد فوق، روابط ذیل صحیح هستند:

$$D=\left[\vec{D}1,\vec{D}2,..,\vec{D}N\right]=\left[d_{i,j}\right]_{N\times N}$$
$$i=1,2,\cdots,N\ ,\ j=1,2,\cdots,N \qquad (۸)$$

- $D=[d_{i,j}]$، ماتریس فاصله شهرها از یکدیگر می‌باشد که در آن $d_{i,j}$ معرف فاصله شهر i تا شهر j است. هدف ما در این مقاله حل مسأله فروشنده دوره‌گرد متقارن[۱۹] (STSP) است. در STSP فاصله شهر i تا شهر j برابر با فاصله شهر j تا شهر i است. بدیهی است که به این ترتیب D یک ماتریس متقارن خواهد بود:

$$d_{i,j}=d_{j,i}$$

- عبارت ${}^{i}x_k^l$، درایه k از بردار خروجی نرون i که در لایه l شبکه واقع شده است را نشان می‌دهد.
- نرون صفرم، همان نرون M و نرون $M+1$ همان نرون اول است.
- مسیری که شبکه در مرحله t معرفی می‌کند را با $\vec{\varphi}^t$ نشان می‌دهیم. بنابراین، φ_p^t بیانگر شهری است که در مرحله t، در محل p مسیر قرار گرفته است.
- e^t انرژی شبکه در مرحله t را نشان می‌دهد.

بیان TSP به صورت یک تابع خطی

TSP را می‌توان مطابق معادله‌های زیر به صورت یک تابع خطی مدل کرد [۲۶].

$$v_{i,j}=\begin{cases}1 & \text{If } i\text{-th city is located after } j\text{-th city}\\ 0 & \text{Otherwise}\end{cases} \quad (۹)$$

$$L=\sum_{i=1}^{N}\sum_{j=1}^{N}d_{i,j}v_{i,j} \quad (۱۰)$$

که L طول مسیر است. برای اینکه مسیر از هر شهر فقط یکبار بگذرد، باید شرایط ذیل نیز برقرار باشد.

$$\sum_{i=1}^{N}\sum_{j=1}^{N}v_{i,j}=N \quad (۱۱)$$

$$\sum_{i=1}^{N}v_{i,j}=1 \qquad \forall j=1,2,\ldots,N \quad (۱۲)$$

$$\sum_{j=1}^{N}v_{i,j}=1 \qquad \forall i=1,2,\ldots,N \quad (۱۳)$$

طبق روش رایج در حل TSP با استفاده از مدل هاپفیلد، برای هر شهر، N نرون در نظر گرفته می‌شود و خروجی نرون j از نرون‌های وابسته به شهر i زمانی برابر با یک است که شهر i در محل j مسیر قرار گرفته باشد [۴][۲۷]. تابع انرژی شبکه نیز برابر با حاصل جمع چندین چندجمله‌ای درجه دو است که هر کدام از آنها وظیفه شبیه‌سازی یکی از معادله‌های (۱۰) تا (۱۳) را بر عهده دارد. معادله زیر، تابع انرژی را که Hopfield و Tank برای حل TSP پیشنهاد کرده‌اند، نشان می‌دهد [۲۲].

$$\begin{aligned}E=&\frac{A}{2}\sum_{x=1}^{N}\sum_{i=1}^{N}\sum_{\substack{j=1\\ j\neq i}}^{N}v_{x,i}v_{x,j}+\\ &\frac{B}{2}\sum_{i=1}^{N}\sum_{x=1}^{N}\sum_{\substack{y=1\\ y\neq x}}^{N}v_{x,i}v_{y,i}+\\ &\frac{C}{2}\left[\sum_{x=1}^{N}\sum_{i=1}^{N}v_{x,i}-N\right]^2+\\ &\frac{D}{2}\sum_{x=1}^{N}\sum_{\substack{y=1\\ y\neq x}}^{N}\sum_{i=1}^{N}d_{x,y}v_{x,i}(v_{y,i+1}+v_{y,i-1})\end{aligned} \quad (۱۴)$$

در معادله (۱۴)، دو جمله اول سبب می‌شوند تا مسیر از هر شهر فقط یک مرتبه عبور کند (شرایط معادله‌های ۱۲ و ۱۳)، جمله سوم باعث می‌شود تا مسیر از تمام شهرها بگذرد (شرط معادله ۱۱) و در نهایت جمله آخر باعث می‌نیمم شدن طول مسیر می‌شود (شرط معادله ۱۰). اما اولین مشکلی که در اینجا خود نمایی می‌کند آن است که کمینه کردن تابع انرژی معادله (۱۴)، علیرغم بخش‌های مختلف آن، نمی‌تواند تحقق شرایط معادله‌های (۱۱)-(۱۳) را تضمین کند. برای حل این مشکل معمولا ضرایب A، B و C را در مقایسه با D به اندازه کافی بزرگ انتخاب می‌کنند. اما در این حالت نیز، امکان نقض شدن شرایط مذکور وجود دارد. لذا معمولا دستیابی به یک پاسخ معتبر و مناسب در شبکه هاپفیلد، نیازمند تکرار الگوریتم یادگیری، با ضرایب A، B، C و D مختلف است.

بعلاوه، پیاده‌سازی معادله‌های (۱۱)، (۱۲) و (۱۳) در تابع انرژی به پیچیده شدن آن و افزایش شدید می‌نیمم‌های محلی، می‌انجامد. همچنین این کار باعث ورود پارامترهای A، B، C و D به تابع انرژی می‌شود که نیاز به مقداردهی اولیه دارند. ما می‌خواهیم تابع انرژی شبکه‌مان تنها شامل معادله (۱۰) باشد و سایر شرایط مسأله در ساختار الگوریتم آموزشی رعایت شود.

شبکه عصبی پیشنهادی

در ادامه مقاله، در ابتدا شبکه عصبی پیشنهادی اولیه شرح داده می‌شود. سپس، این شبکه به یک شبکه عصبی سازنده توسعه می‌یابد و در نهایت، نحوه توسعه شبکه پیشنهادی برای حل مساله SPSN شرح داده شده است.

شبکه عصبی پیشنهادی اولیه

در شبکه عصبی پیشنهادی اولیه، فرض می‌شود کـه تمام شهرها بر روی مسیر قرار دارند لذا، $M=N$. شبکه عصبی اولیه شامل چهار لایه می‌باشد و خروجی نرون‌های هر لایه به شکل بردار است. شکل (۲) این شبکه را برای سه شهر نشان می‌دهد. در شکل، ورودی‌ها، خروجی‌ها و مقادیر آستانه نرون‌ها نیز نشان داده شده‌اند. در این شکل همانند مدل کوهونن، مسیر در لایه اول (path layer) با M نرون بیان می‌شود ولی وزن‌های این نرون‌ها به سمت مختصات شهرها میل نمی‌کند، بلکه به شهرها اجازه جابجایی در آنها داده می‌شود. خروجی نرون i در لایه اول شبکه پیشنهادی چنین تعریف می‌گردد:

$$ {}^{i}\vec{X}^{1} = \left[{}^{i}x_{j}^{1} \right],\ i = 1,2,\ldots,M\ ,\ j = 1,2,\ldots,N $$

به طوری که

$$ {}^{i}x_{j}^{1} = \begin{cases} 1 & \text{If } j\text{-th city is located at } i\text{-th location} \\ 0 & \text{Otherwise} \end{cases} $$

(۱۵)

بنابراین، مسیر فعلی پیشنهادی توسط شبکه عبارت است از:

$$ \varphi_i^t = index\left(\max_{j=1}^{N} ({}^{i}x_{j}^{1}) \right),\quad i = 1,2,\ldots,M $$

(۱۶)

همچنین داریم:

$$ {}^{i}x_{j}^{1} = \begin{cases} 1 & j = \varphi_i^t \\ 0 & \text{Otherwise} \end{cases} $$

(۱۷)

به این‌ترتیب تابع انرژی شبکه مطابق معادله ذیل تعریف می‌شود که برابر با طول مسیر است.

$$ e^t = \sum_{k=1}^{M}\sum_{i=1}^{N}\sum_{j=1}^{N} d_{i,j}\ {}^{k}x_{i}^{1}\,{}^{k-1}x_{j}^{1} $$

(۱۸)

اندیشه اصلی الگوریتم یادگیری چنین است که ابتدا یک مسیر اولیه به شبکه معرفی گردد و شبکه بر اساس آن به مسیر بهینه‌ای همگرا شود، به‌طوری‌که در هر مرحله از آموزش، حتی قبل از همگرا شدن شبکه، مسیری که شبکه به ما معرفی می‌کند، معتبر باشد. در حقیقت، شبکه پیشنهادی همانند یک فیدبک منفی عمل کرده و در نهایت به یکی از مینیمم‌های محلی تابع انرژی همگرا می‌شود.

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

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

(۱۹)

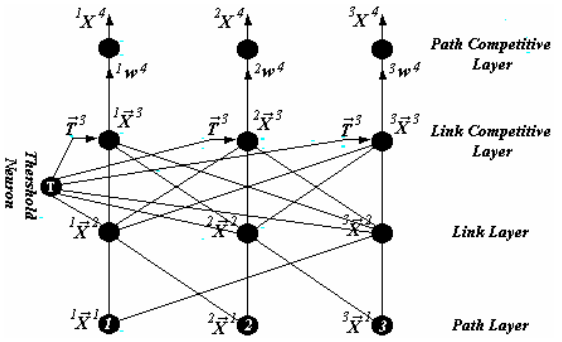
در لایه سوم (link competitive layer)، $N+1$ نرون داریم که یکی از آنها، نرون آستانه است. معادله (۲۰)، عملکرد نرون آستانه را شرح می‌دهد:

$$ \vec{S}^{3} = \left\{ s_k^3 \right\},\ k = 1,2,\cdots M $$

به طوری که،

$$ s_k^3 = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} d_{i,j} \times {}^{k}x_{i}^{2} \times {}^{k}x_{j}^{2}\ . $$

(۲۰)



شکل ۲: شبکه عصبی پیشنهادی اولیه.

هر یک از نرون‌های باقیمانده متناظر با یک شهر است (نرون اول متناظر با شهر اول، نرون دوم متناظر با شهر دوم و ...). وزن هر یک از این نرون‌ها، مطابق معادله (۲۱) تعریف می‌شود:

$$ {}^{i}w_{j}^{3} = \vec{D}_i^T \vec{X}_j^2 - s_j^3 $$
$$ i = 1,2,\cdots,N\ ,\ j = 1,2,\cdots,M $$

(۲۱)

همچنین، خروجی نرون‌های این لایـه نیـز مطـابق معادلـه ذیل محاسبه می‌شود:

$$ {}^{i}\vec{X}^{3} = \begin{bmatrix} {}^{i}k^{3} \\ {}^{i}v^{3} \end{bmatrix},\quad i = 1,2,\ldots,N $$
$$ \begin{cases} {}^{i}k^{3} = index\left(\min_{j\in P_i}\left({}^{i}w_{j}^{3} \right) \right) \\ {}^{i}v^{3} = {}^{i}w_{i_{k^3}}^{3} = \min_{j\in P_i}\left({}^{i}w_{j}^{3} \right) \end{cases}, $$
$$ P_i = \left\{ j \middle| j = 1,2,\ldots,M; {}^{j}x_{i}^{2} \neq 1 \right\} $$

(۲۲)

که تابع $index(x)$ محل درایه x در بردار $\vec{X}$ را برمی‌گرداند. در هر نرون از این لایه، بین کمان‌های مختلف مسیر، رقابتی شکل می‌گیرد و کمان برنده و وزن آن به عنـــوان

خروجی نرون معرفی می‌شوند.

لایه چهارم (path competitive layer) نیز شامل N نرون بوده و همانند لایه سوم هر نرون متناظر با یک شهر است.

وزن‌های نرون‌ها (${}^{i}w^{4}$, $\mathrm{i}=1,2,...,\mathrm{N}$) نیز در پایان هر مرحله از آموزش با توجه به مسیر جدیدی که شبکه معرفی می‌کند، بهنگام می‌شود. معادله‌های (۲۳)-(۲۵) مبین رفتار نرون‌های این لایه هستند.

$$ {}^{i}v^{4} = {}^{i}v^{3} - {}^{i}w^{4}, \quad i = 1, 2, \cdots, N \tag{۲۳}$$

$$ \omega = index\left(\min_{i=1}^{N} \left({}^{i}v^{4} \right) \right): \ \text{The winner} \tag{۲۴}$$

$$ {}^{i}x^{4} = \begin{cases} {}^{i}k^{3} & i = \omega \text{ and } {}^{\omega}v^{4} < 0 \\ 0 & \text{Otherwise} \end{cases} \tag{۲۵}$$

در این لایه نرونی که کمترین وزن را دارد، برنده می‌شود (نرون k) و خروجی‌اش، محل جدید شهر متناظر با آن را در مسیر مشخص می‌کند.

الگوریتم آموزش شبکه پیشنهادی اولیه

الگوریتم آموزش شبکه پیشنهادی اولیه در جدول (۲) آورده شده است. همانطور که در ضمیمه الف اثبات شـــده است، تابع انرژی طی دوره آموزش سیر نزولی دارد و به این ترتیب شبکه در نهایت به مسیری بهتر از مسیر اولیه (و یا همان مسیر، در صورتی که در همان مرتبه اول آموزش نرونی برنده نشود) همگرا می‌شود. با توجه به اینکه در هر مرحله از آموزش شبکه، شهری از یک مکان در مسیر، به مکان دیگری منتقل می‌شود، می‌توان نتیجه گرفت که مسیر نهایی شبکه همان خصوصیات مسیر اولیه‌اش را دارد. یعنی اگر مسیر اولیه از تمام شهرها گذشته باشد، مسیر نهایی نیز از تمام آنها خواهد گذشت و اگر مسیر اولیه از هر شهر تنها یکبار عبور کرده باشد، مسیر نهایی نیز چنین خواهد بود و به عبارت ساده‌تر، اگر مسیر اولیه معتبر باشد، مسیر نهایی نیز معتبر خواهد بود.

توسعه شبکه اولیه به یک شبکه سازنده[۲۰]

همانطور که بیان شد پاسخ‌های شبکه فوق به مسیر اولیه‌اش وابسته است. اصولا شبکه‌های مبتنی بر تابع انرژی مستعد به بدام افتادن در مینیمم‌های محلی تابع انرژی هستند. در [۲۴] چندین الگوریتم برای تولید مسیر اولیه مناسب برای این شبکه پیشنهاد و مقایسه شده‌اند. بهترین پاسخ‌ها مربوط به روشی می‌باشد که شبکه را با استفاده از یک الگوریتم سازنده، به سمت پاسخی نزدیک به بهینه هدایت می‌نماید.

جدول ۲: الگوریتم آموزش شبکه عصبی پیشنهادی اولیه.

۱. مسیر معتبری انتخاب شود (مسیری معتبر است که از هر شهر فقط یکبار گذشته باشد). همچنین، شمارنده t برابر صفر قرار داده می‌شود.

۲. خروجی نرون‌های لایه اول مطابق معادله (۱۷) تعیین می‌گردد.

۳. وزن‌های نرون‌های لایه چهارم طبق معادله (۲۶) تعیین می‌شود.

$$ {}^{q}w^{4} = \sum_{r=1}^{M} \sum_{i=1}^{N} \sum_{j=1}^{N} \left(d_{q,i} + d_{q,j} - d_{i,j} \right) \times {}^{r}x_{q}^{1} \times {}^{(r+1)}x_{i}^{1} \times {}^{(r-1)}x_{j}^{1} \tag{۲۶}$$

۴. شبکه یک مرحله اجرا می‌شود.

۵. اگر هیچ نرونی در لایه آخر برنده نشده و خروجی تمام نرون‌های این لایه صفر باشد، شبکه همگرا شده و آموزش خاتمه می‌یابد.

۶. در غیر این صورت، مطابق معادله (۲۴)، در لایه چهارم، نرون ω برنده شده است. با فرض اینکه این نرون در محل a از مسیر قرار گرفته است، داریم:

$$ {}^{\omega}x^{4} = {}^{\omega}k^{3} \neq 0 \ , \quad \varphi_{a}^{t} = \omega, $$

در این حالت، مسیر جدید مطابق معادله ذیل تعریف می‌شود:

$$ \vec{\varphi}^{t+1} = [\varphi_{1}^{t}, \varphi_{2}^{t}, \ldots, \varphi_{{}^{\omega}x^{4}}^{t}, \omega, \varphi_{{}^{\omega}x^{4}+1}^{t}, \ldots, \varphi_{a-1}^{t}, \varphi_{a+1}^{t}, \ldots, \varphi_{M}^{t}]^{T} . \tag{۲۷}$$

به عبارت دیگر، شهر برنده از محل فعلی‌اش به محل جدیدی که شبکه در خروجی این نرون بیان می‌کند، منتقل می‌شود.

۷. مراحل ۲-۶ تا آنجا تکرار می‌شود که شبکه همگرا گردد.

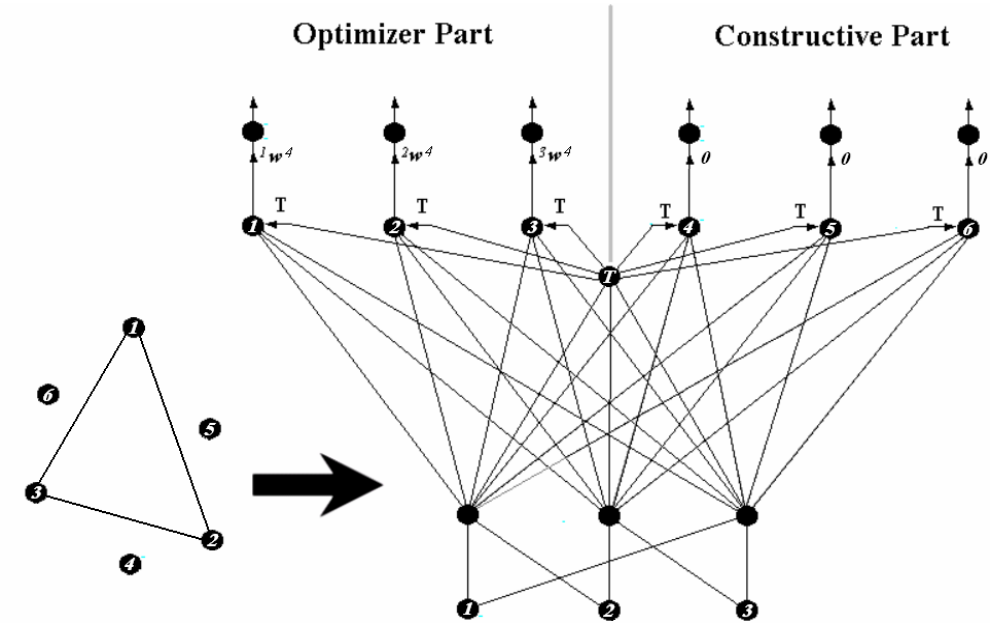
اندیشه اصلی در این روش بدین شرح است که در هر مرحله، بر اساس ضابطه‌ای از پیش تعیین شده، از بین شهرهایی که در مسیر قرار ندارند، یکی انتخاب شده و در مسیر قرار گیرد و سپس شبکه، مسیر جدید را بهینه کند. به عنوان مثال، اگر ضابطه مذکور کمترین فاصله بین شهر و کمانی از مسیر باشد، آنگاه در هر مرحله شهری که فاصله‌اش تا نزدیکترین کمان از بقیه کمتر است، انتخاب می‌شود و در بین دو شهری که نزدیکترین کمان را تشکیل داده‌اند، قرار می‌گیرد.

با مطالعه بر روی بهترین ضابطه‌ای که در [۲۴] ارائه شده است، دریافتیم که الگوریتمی که بهترین پاسخ‌ها را ارائه می‌دهد، در حقیقت عملکردی مشابه با خود شبکه دارد و می‌توان با توسعه 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 متناظر با 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.

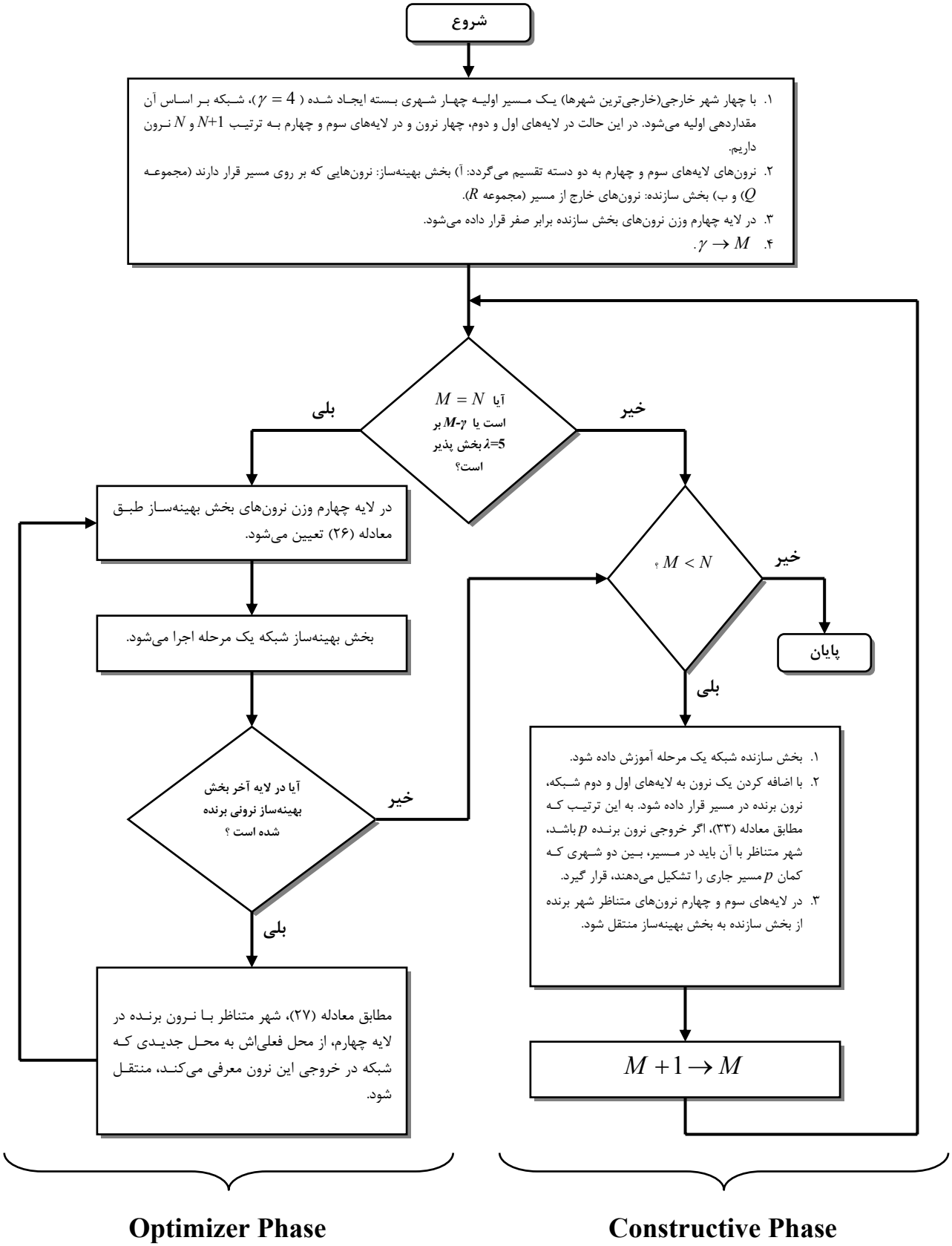
اما اگر بخواهیم تعداد شهرهای مسیر را افزایش دهیم چه باید کرد؟ اکنون فرض کنید که در لایه سوم و چهارم علاوه بر نرون‌های قبلی، متناظر با شهرهایی که بر روی مسیر قرار ندارند، نرون‌هایی در نظر گرفته و وزن نرون‌های اضافه شده به لایه چهارم را نیز برابر صفر قرار دهیم. همانطور که در بخش پنجم ذکر شد در هر یک از 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 می‌باشد و ۲) این شهر اگر در هر جای دیگر مسیر قبلی قرار گیرد مسیری طولانی‌تر ایجاد خواهد شد (این مطلب نتیجه مستقیم معادله ۲۲ است).

اکنون اگر در لایه چهارم تنها بین نرون‌های خارج از مسیر رقابتی شکل گیرد بدیهی است که خروجی نرون برنده، بیانگر موقعیت شهر متناظرش 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 دارا می‌باشد، شکل می‌گیرد.

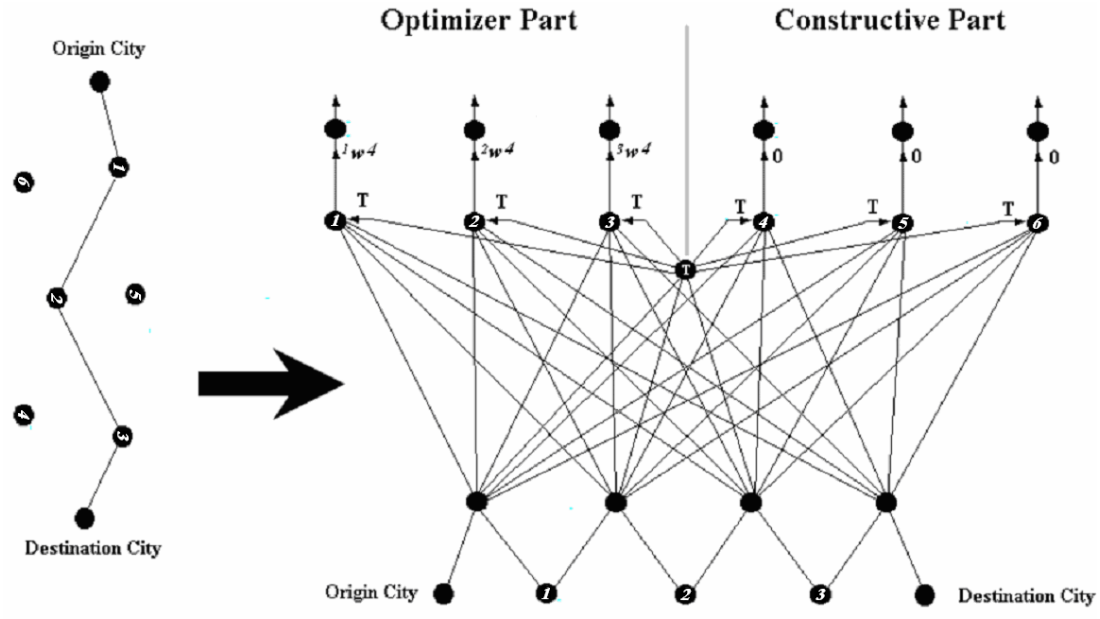
با توجه به مطالب ذکر شده، می‌توان شبکه پیشنهادی را مطابق با ساختار نشان داده شده در شکل (۳) و الگوریتم آموزش شکل (۴) به یک شبکه عصبی سازنده تبدیل نمود. در شبکه سازنده، نرون‌های لایه سوم و چهارم به دو بخش تقسیم شده‌اند: ۱) بخش بهینه‌ساز که شامل نرون‌های روی



شکل ۳: یک مسیر سه شهری نمونه و شبکه توسعه یافته متناظر با آن.



شکل ۴: نمودار بلوکی الگوریتم آموزش شبکه عصبی پیشنهادی (CNN-TSP).



شکل ۵: ساختار تصحیح شده شبکه عصبی پیشنهادی برای حل SPSN.

مسیر (مجموعه Q) است و ۲) بخش سازنده که شامل نرون‌های خارج از مسیر (مجموعه R) می‌باشد:

$$Q=\left\{ j \,\middle|\, j=index\left(\max_{k=1}^{N}(x^{1}_{i,k})\right),\ i=1,2,\ldots,M \right\} \quad (۲۸)$$

$$R=\{1,2,\ldots,N\}\text{-Q} \quad (۲۹)$$

همچنین در لایه چهارم از شبکه سازنده پیشنهادی، به جای معادلات (۲۴) و (۲۵)، از شکل تصحیح شده آنها استفاده می‌شود:

$$\omega_{opt}=index\left(\min_{i\in Q}({}^{i}v^{4})\right) \quad (۳۰)$$

$$\omega_{cns}=index\left(\min_{i\in R}({}^{i}v^{4})\right) \quad (۳۱)$$

$$x_i^4=\begin{cases} k^3_{\omega_{bpt}} & i=\omega_{opt},\ {}^{\omega_{bpt}}v^4<0 \\ k^3_{\omega_{cns}} & i=\omega_{cns} \\ 0 & \text{Otherwise} \end{cases}$$

$$i=1,2,\ldots,N \quad (۳۲)$$

الگوریتم آموزش شبکه پیشنهادی نیز مطابق ساختار شبکه به دو فاز سازنده و بهینه‌ساز تقسیم می‌شود. در فاز سازنده، نرون‌های بخش سازنده و در فاز بهینه‌ساز نرون‌های بخش بهینه‌ساز در فرآیند آموزش شرکت می‌کنند. همانطور که در شکل (۴) نشان داده شده است، آموزش شبکه سازنده پیشنهادی، با مقداردهی شبکه با یک مسیر چهار شهری ($M=\gamma=4$) آغاز می‌شود. این مسیر اولیه شامل خارجی‌ترین شهرها است. سپس آموزش شبکه در فاز سازنده ادامه می‌یابد. در هر مرحله از فاز سازنده، یک شهر از میان شهرهای خارج مسیر (بخش سازنده شبکه) در لایه چهارم برنده شده و شهر متناظر با آن مطابق معادله ذیل در مسیر قرار می‌گیرد.

$$\bar{\varphi}^{t+1}=[\varphi^t_1,\ldots,\varphi^t_{\omega_{cns}x^4},\omega_{cns},\varphi^t_{\omega_{cns}x^4+1},\ldots,\varphi^t_M]^T \quad (۳۳)$$

آموزش شبکه در فاز سازنده، λ=5 مرحله ادامه می‌یابد. به این ترتیب، در انتهای فاز سازنده، λ شهر به مسیر اضافه شده است. سپس، آموزش شبکه به فاز بهینه‌ساز منتقل می‌شود. در فاز بهینه‌ساز، مسیر فعلی بهینه می‌شود. به این ترتیب که در هر مرحله از فاز بهینه‌ساز، شهر برنده در لایه چهارم (از میان شهرهای روی مسیر که بخش بهینه‌ساز شبکه را تشکیل می‌دهند)، مطابق معادله (۲۷) از محل فعلی به محلی مناسبتر منتقل می‌شود (توجه شود که در اینجا ω_{opt} باید جایگزین ω در معادله ۲۷ گردد).

چنانچه در فاز بهینه‌ساز، در لایه چهارم، نرونی برنده نشده باشد، شبکه در فاز بهینه‌ساز همگرا گردیده و دوباره آموزش به فاز سازنده منتقل می‌شود. این فرآیند تا آنجا ادامه می‌یابد که تمام شهرها وارد مسیر شده ($M=N$) و شبکه در فاز بهینه‌ساز همگرا شود. تعداد شهرهایی که باید

در فاز سازنده به مسیر اضافه شود (λ) به صورت تجربی بدست آمده است. شبکه سازنده پیشنهادی CNN-TSP[۲۱] نامیده می‌شود.

توسعه شبکه پیشنهادی به منظور حل مسأله SPSN

ساختار رقابتی CNN-TSP به آن قابلیت انعطاف زیادی می‌بخشد. بطوری که می‌توان با توسعه ساختار، از آن برای حل دیگر مسائل NP-Complete استفاده نمود. در اینجا نحوه حل مسأله کوتاهترین مسیر با تعداد شهر مشخص (SPSN) را با استفاده از شبکه عصبی پیشنهادی بیان می‌کنیم. در این مسأله هدف یافتن کوتاهترین مسیر ممکن بین مبدا و مقصدی مشخص است؛ به‌طوری که از تعداد معینی شهر بگذرد.

در مقالات به SPSN، علیرغم کاربردهای فراوان آن مانند جمع‌آوری زباله‌های زیان‌آور (از قبیل ضایعات نیروگاه‌های هسته‌ای و سایر مواد شیمیایی خطرناک در کارخانجات) و برنامه‌ریزی پروازهای مسافرتی و باری (جهت بهینه‌سازی مصرف سوخت و کسب جداکثر سود)، کمتر توجه شده است. در مسأله کوتاهترین مسیر[۲۲] (SP) کلاسیک، تعداد شهرها از قبل مشخص نمی‌باشد، در حالی که در اینجا با مشخص کردن تعداد شهرهای مسیر عملا یک محدویت جدید به مسأله اضافه شده است.

برای حل مسأله SPSN، ساختار CNN-TSP را مطابق شکل (۵) توسعه می‌دهیم. نرون‌های اول و آخر به ترتیب متناظر 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 لایه‌های شبکه (لایه‌های سوم و چهارم) بدون تغییر باقی می‌مانند. باید دقت نمود که برای شهرهای مبدا و مقصد در این دو لایه نرونی در نظر گرفته نمی‌شود. همچنین الگوریتم آموزش این شبکه که CNN-SPSN نامیده می‌شود، مشابه CNN-TSP است.

چنانچه از بین N شهر موجود یکی را هم به عنوان مبدا و هم به عنوان مقصد در نظر بگیریم و از CNN-SPSN برای یافتن کوتاهترین مسیر *N-1* شهری بین مبدا و مقصد مذکور استفاده کنیم در حقیقت شبکه به ما پاسخی برای مسأله TSP متناظر با N شهر مفروض داده است. به عبارت دیگر توانسته‌ایم از CNN-SPSN برای حل مسأله فروشنده دوره‌گرد نیز استفاده نماییم. می‌توان از این خاصیت برای بررسی دقت CNN-SPSN استفاده نمود. بدیهی است که زمانی که از CNN-SPSN برای حل TSP استفاده می‌کنیم، شبکه بیشترین خطای ممکن را از خود نشان می‌دهد؛ زیرا مسیر در این حالت دارای بیشترین تعداد شهر است.

نتایج تجربی

در ادامه مقاله، نتایج تجربی حاصل از شبیه‌سازی شبکه عصبی پیشنهادی شرح داده می‌شود.

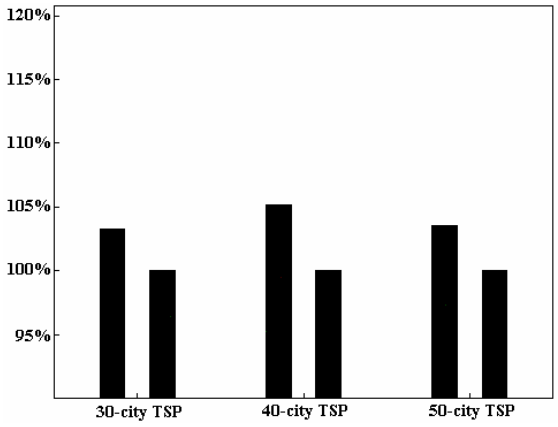
فرآیند نرمالسازی

در شبیه‌سازی‌ها از چندین الگوریتم برای حل مسائل TSP مختلف به عنوان محک استفاده شده است. یک روش برای مقایسه عملکرد الگوریتم‌های مختلف میانگین‌گیری از طول مسیرهای معرفی شده توسط آنها است. اما با توجه به اینکه طول مسیر وابسته به توزیع شهرهای آن بوده و این توزیع از مسأله‌ای تا مسأله دیگر بسیار متفاوت است لذا، باید طول مسیرها قبل از میانگین‌گیری نرمالیزه شوند.

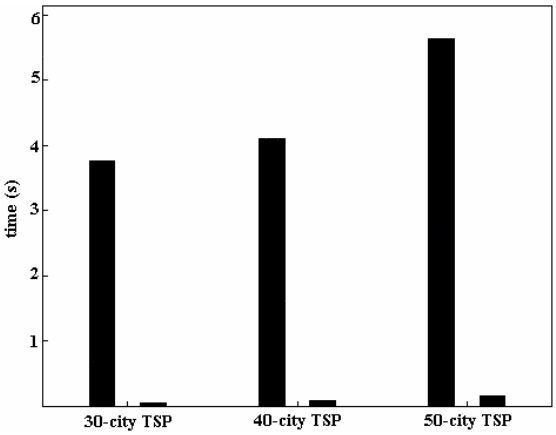
فرض کنید، ارزیابی عملکرد m الگوریتم برای n مساله TSP مختلف (به عنوان محک) مطلوب است. برای این منظور کافی است که پاسخ m الگوریتم مذکور برای هر یک از مسائل محک محاسبه شود. سپس، طول مسیرهای بدست آمده برای هر مساله، توسط روش‌های مختلف، با استفاده از طول بهترین مسیر بدست آمده نرمالیزه می‌شود. در نهایت، مطابق معادله ذیل طول متوسط نرمالیزه (یا بطور خلاصه طول متوسط) برای هر یک از الگوریتم‌ها بطور جداگانه محاسبه می‌شود:

$$\overline{e}_i = \frac{1}{n}\sum_{j=1}^{n}\frac{e_{i,j}}{\inf_{i=1}^{m}\left(e_{i,j}\right)}$$

که در آن، $e_{i,j}$ طول مسیر بدست آمده از الگوریتم i برای مساله محک j می‌باشد. بدیهی است که طول متوسط معیاری است برای مقایسه مسیرهای بدست آمده از هر روش نسبت به دیگر روش‌ها. به عنوان مثال، چنانچه معیار طول متوسط برای یک الگوریتم ۱۰۰٪ باشد، بدان معنی است که پاسخ‌های الگوریتم مذکور همواره بهتر از دیگر الگوریتم‌ها بوده است. همچنین، چنانچه این معیار مثلا برابر ۱۰۵٪ باشد، بدان معنی است که طول پاسخ‌های



شکل ۶ : طول متوسط مسیرهای پیشنهاد شده توسط شبکه کوهونن (ستون اول هر جفت)، طول متوسط مسیرهای شبکه پیشنهادی CNN-TSP (ستون دوم هر جفت) نسبت به طول مسیر کوتاهتر.



شکل ۷: زمان همگرایی شبکه کوهونن (ستون اول هر جفت) و زمان همگرایی شبکه پیشنهادی CNN-TSP (ستون دوم هر جفت) برحسب ثانیه.

الگوریتم مورد نظر بطور متوسط ۵٪ بیشتر از طول بهترین مسیر بدست آمده می‌باشد. به عبارت دیگر، کوچک بودن (نزدیک بودن به ۱۰۰٪) معیار طول متوسط بیانگر عملکرد مناسب الگوریتم مورد نظر و برعکس، بزرگ بودن آن نشان دهنده کارایی نامناسب الگوریتم مذکور می‌باشد.

مقایسه شبکه‌های CNN-TSP و کوهونن

شبیه‌سازی‌های این بخش بر روی یک کامپیوتر شخصی AMD Athlon 1600MHz تحت نرم‌افزار Matlab انجام شده است. به منظور تعیین دقت CNN-TSP در مقایسه با کوهونن، شبکه شرح داده شده در بخش دوم پیاده‌سازی شده است. برای ۱۰۰ توزیع ۳۰ شهری، ۱۰۰ توزیع ۴۰ شهری و ۱۰۰ توزیع ۵۰ شهری، پاسخ‌های هر دو شبکه و زمان همگرایی آنها محاسبه شده است. پارامترهای شبکه کوهونن چنین انتخاب شده‌اند:

$\eta_0 = \sqrt{30}$, $G_0 = 10$, $Tr = 0.01$, $\alpha = 0.01$

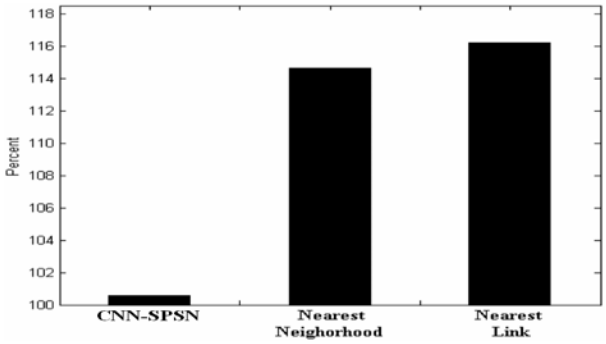
بنا به نظر Kim و Choi شبکه با این پارامترها بهترین پاسخ‌ها را ارائه می‌دهد [۲۹]. شبیه‌سازی‌ها نیز این نکته را تأیید می‌کند. در شکل (۶) عملکرد شبکه عصبی پیشنهادی و نگاشت خودسازمانده کوهونن با استفاده از معیار طول متوسط (که در بخش قبل معرفی گردید)، مقایسه شده است. همچنین در شکل (۷) زمان همگرایی متوسط دو شبکه مذکور، برای مسائل با تعداد شهر یکسان، نشان داده شده است.

همانطور که در شکل (۷) مشاهده می‌شود، شبکه عصبی کوهونن بسیار کندتر از شبکه پیشنهادی است؛ به طـــوری که سرعت همگرایی شبکه پیشنهادی حداقل ۲۰ برابر آن می‌باشد. همچنین نمودارهای شکل (۶) حاکی از آن است که تقریبا در تمام موارد مسیرهای شبکه پیشنهادی کوتاهتر نیز بوده است، زیرا در این شکل تقریبا همه ستون‌های مربوط به شبکه پیشنهادی، ۱۰۰٪ را نشان می‌دهند. مسیرهای بدست آمده از شبکه کوهونن به طور متوسط به اندازه بیش از ۲/۵٪ از مسیرهای ارائه شده توسط شبکه عصبی پیشنهادی بلندتر بوده‌اند.

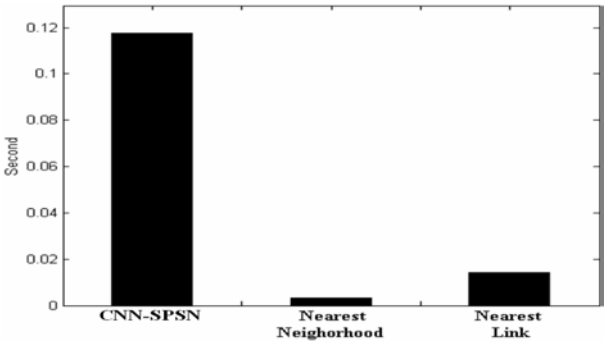
بررسی عملکرد شبکه CNN-SPSN

در این بخش، در ابتدا پاسخ‌های الگوریتم CNN-SPSN را برای یافتن بهترین مسیر ۵۰ شهری از بین ۱۰۰ شهر، با پاسخ‌های الگوریتم‌های کلاسیک 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 از مسیر، تا مراکز کمان‌های مسیر محاسبه می‌شود و شهری که کمترین فاصله را دارد، در مسیر قرار داده می‌شود. برای ۲۰۰ توزیع مختلف ۱۰۰ شهری، پاسخ‌ها و زمان همگرایی هر سه الگوریتم محاسبه گردید. نتایج حاصل در شکل‌های (۸) و (۹) نشان داده شده است. همانطور که در شکل (۸) مشاهده می‌کنید، پاسخ‌های

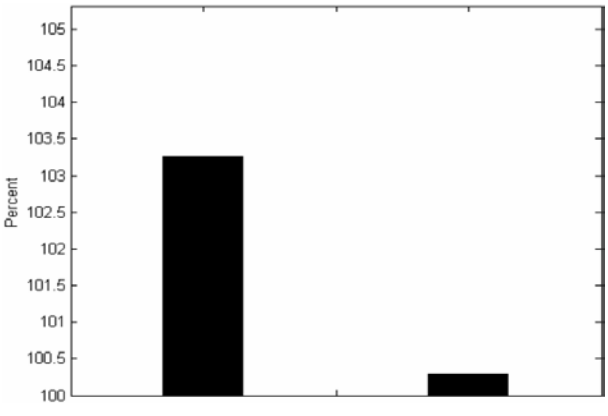
شبکه به مراتب از دو الگوریتم دیگر بهتر است و تقریبا عدد ۱۰۰٪ را نشان می‌دهد؛ در حالی که نزدیکترین ستون که مربوط به الگوریتم نزدیکترین همسایه است بیش از ۱۱۴٪ را نشان می‌دهد. البته در ازای این بهبود ما هزینه‌ای هم پرداخته‌ایم و آن افزایش زمان محاسباتی است که در شکل (۹) نشان داده شده است. البته زمان محاسباتی ما کمتر از ۰/۱۲ ثانیه است که هنوز برای بسیاری از کاربردهای بلادرنگ قابل قبول است. اکنون از همان توزیع‌های ۱۰۰ شهری قبلی، برای مقایسه عملکرد CNN-TSP و CNN-SPSN استفاده می‌کنیم. اینبار هدف یافتن بهترین پاسخ برای TSP متناظر با هر توزیع است. نتایج در شکل‌های (۱۰)، (۱۱) و (۱۲) نشان داده شده است. همانطور که در شکل (۱۰) مشاهده می‌کنید، طول مسیرهای معرفی شده توسط CNN-SPSN تقریبا به اندازه ۳٪ از طول مسیرهای CNN-TSP بلندتر است. همچنین با مراجعه به شکل (۱۱) در می‌یابیم که زمان محاسباتی CNN-SPSN تقریبا ۰/۱ ثانیه طولانی‌تر بوده است. دو عامل سبب ایجاد چنین اختلافی شده است: ۱) مسیر اولیه چهار شهری که CNN-TSP بوسیله آن مقدار دهی اولیه می‌شود و ۲) قابلیت جابجایی تمام شهرها (در فاز بهینه‌سازی) در CNN-TSP؛ درحالی‌که در CNN-SPSN شهرهای مبدا و مقصد (که در اینجا یکسان هستند) ثابت می‌باشند. در حقیقت CNN-TSP برای حل TSP بهینه گردیده و بیشتر بودن دقت آن نسبت به CNN-SPSN، امری قابل پیش‌بینی بود. طولانیتر بودن زمان همگرایی CNN-SPSN نیز به دلیل بیشتر بودن تعداد تکرارهای شبکه در فاز بهینه‌سازی نسبت به CNN-TSP است. این مطلب در شکل (۱۲) نشان داده شده است.



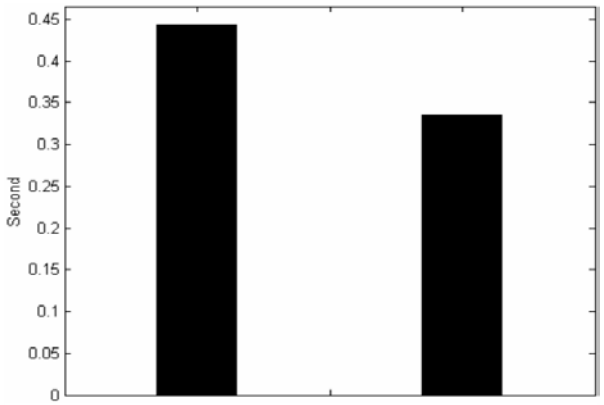
شکل ۸ : طول متوسط مسیرهای ۵۰ شهری پیشنهاد شده توسط الگوریتم‌های CNN-SPSN (ستون اول)، نزدیکترین همسایه (ستون دوم) و نزدیکترین کمان (ستون سوم) نسبت به طول مسیر کوتاهتر برای ۲۰۰ توزیع ۱۰۰ شهری.



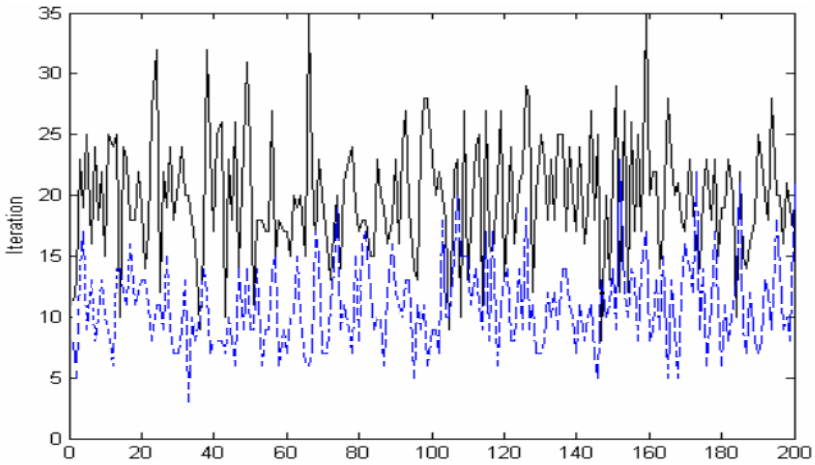
شکل ۹ : زمان همگرایی متوسط الگوریتم‌های CNN-SPSN (ستون اول)، نزدیکترین همسایه (ستون دوم) و نزدیکترین کمان (ستون سوم) برای یافتن مسیرهای ۵۰ شهری از بین ۱۰۰ شهر بر حسب ثانیه.



شکل ۱۰: طول متوسط مسیرهای ۱۰۰ شهری پیشنهاد شده توسط شبکه‌های CNN-SPSN (ستون اول) و CNN-TSP(ستون دوم) نسبت به طول مسیر کوتاهتر، برای ۲۰۰ توزیع ۱۰۰ شهری.



شکل ۱۱: زمان همگرایی متوسط شبکه‌های CNN-SPSN (ستون اول) و CNN-TSP (ستون دوم) برای یافتن مسیرهای ۱۰۰ شهری از بین ۱۰۰ شهر بر حسب ثانیه.



شکل ۱۲: تعداد تکرارهای CNN-SPSN (منحنی توپر بالا) و CNN-TSP (منحنی خط چین پایین) در فاز بهینه‌سازی برای هر توزیع ۱۰۰ شهری.

پاسخ‌های CNN-TSP و CNN-SPSN به مسائل کتابخانه TSPLIB

در این بخش پاسخ‌های CNN-TSP و CNN-SPSN را به بعضی از مسائل ذکر شده در کتابخانه TSPLIB [۳۱] بدست آورده و از این راه دقت آنها را محاسبه کرده‌ایم. نتایج در جدول (۳) ذکر شده است. در ستون‌های این جدول، به ترتیب نام مسأله، تعداد شهرهای آن، بهترین پاسخی که برای آن در کتابخانه مذکور ذکر شده است، پاسخ بدست آمده از CNN-TSP، خطای نسبی CNN-TSP نسبت به بهترین پاسخ و زمان محاسباتی شبکه تا همگرایی، پاسخ بدست آمده از CNN-SPSN، خطای نسبی آن و در نهایت زمان محاسباتی آن ذکر شده است. برای محاسبه خطای متوسط هر دو الگوریتم از مقادیر خطا در ستون‌های پنجم و هشتم، به طور جداگانه میانگین می‌گیریم. همانطور که می‌بینید، اختلاف متوسط CNN-TSP و CNN-SPSN به ۲۹ مسأله‌ای که در کتابخانه TSPLIB پاسخی برای آنها معین شده است، به ترتیب ۳/۸۱٪ و ۷٪ است. با مقایسه ستون‌های ششم و نهم، می‌توان نتیجه گرفت که سرعت همگرایی CNN-TSP از CNN-SPSN بیشتر است. در حقیقت با اینکه CNN-TSP و CNN-SPSN هر دو دارای یک ساختار هستند، اما CNN-TSP برای حل مسأله فروشنده دوره‌گرد بهینه گردیده و مسیر اولیه چهار شهری آن و عدم محدودیتش در جابجایی همه شهرها (شهر مبداء یا مقصد وجود ندارد)، سبب کاهش تعداد تکرارها (و در نتیجه کاهش حجم محاسبات) و افزایش دقت آن شده است. با وجود این، CNN-SPSN برای حل مسأله کوتاهترین مسیر با تعداد شهر مشخص طراحی شده است. دقت CNN-TSP در مقایسه با سرعت آن بسیار خوب و دقت CNN-SPSN نیز در حد قابل قبولی ارزیابی می‌شود. در جدول (۴)، پاسخ‌های CNN-TSP و CNN-SPSN برای ۱۰ مسأله از کتابخانه TSPLIB، با دو روش محک متداول شامل آبکاری شبیه‌سازی شده [۳۲] و نگاشت خودسازمانده Budinich's SOM [۳۳] مقایسه شده است. نتایج بیانگر آن است که دقت CNN-TSP در مقایسه با این دو الگوریتم، به ترتیب ۰/۷۷٪ و ۰/۴۶٪ بهبود داشته است. همچنین دقت CNN-SPSN نیز به ترتیب تنها ۰/۳٪ و ۳/۳٪ کمتر از دو الگوریتم محک مذکور بوده است. بعلاوه، مطابق جدول (۴)، CNN-TSP به ۶ مسأله از ۱۰ مسأله مذکور، پاسخی بهتر از روش‌های محک داده است که این خود دلیلی دیگر بر کارایی این الگوریتم است. این در حالی است که سرعت CNN-TSP و CNN-SPSN، همانطور که قبلا نشان داده شد، بسیار بیشتر از الگوریتم‌هایی چون آبکاری شبیه‌سازی شده و شبکه‌های خود سازمانده است.

نتیجه‌گیری

در این مقاله یک شبکه عصبی سازنده جدید برای حل مسأله TSP ارائه شده است. در طراحی این شبکه سعی گردیده با بهره‌گیری از ویژگی‌های اساسی شبکه‌های عصبی هاپفیلد و کوهونن، مزایای هر یک حفظ و از معایب آنها پرهیز شود. ساختار فیدبکی-رقابتی شبکه، انعطاف زیادی به آن بخشیده است و از این رو می‌توان مشابه با

جدول ۳: پاسخ شبکه‌های CNN-TSP و CNN-SPSN به تعدادی از مسائل TSPLIB.

Tour name	No. of Cities.	Opt. answer reported at TSPLIB	CNN-TSP answers	%Dif. of CNN-TSP answers	Com. Time of CNN-TSP (s)	CNN-SPSN answers	%Dif. of CNN-SPSN answers	Com. Time of CNN-SPSN (s)
burma14	14	-	30.88	-	0.006	31.45	-	0.006
ulysses16	16	74.00	74.00	0.0%	0.006	74.20	0.27%	0.008
ulysses22	22	75.31	76.50	1.58%	0.009	75.31	0.0%	0.012
bays29	29	9074.15	9077.92	0.0415%	0.020	9309.79	2.60%	0.022
dantzig42	42	-	687.81	-	0.029	731.02	-	0.038
att48	48	33523.71	34394.14	2.60%	0.035	34153.19	1.88%	0.044
eil51	51	426	443.44	4.09%	0.046	450.99	5.87%	0.054
Berlin52	52	7544.37	8333.74	10.46%	0.045	8483.52	12.44%	0.061
st70	70	678.60	697.99	2.56%	0.098	705.17	3.92%	0.139
eil76	76	545.39	561.51	2.96%	0.129	578.17	6.01%	0.176
pr76	76	108159.44	112858.39	4.34%	0.118	112351.44	3.88%	0.165
gr96	96	512.31	537.65	4.95%	0.251	561.17	9.54%	0.389
rat99	99	-	1224.10	-	0.300	1291.03	-	0.377
kroA100	100	21282	21831.73	2.58%	0.326	23211.55	9.07%	0.389
kroB100	100	-	22716.22	2.61%	0.227	24531.03	10.80%	0.385
kroC100	100	20750.76	21065.76	1.52%	0.211	22833.17	10.03%	0.437
kroD100	100	21294.29	21634.06	1.60%	0.216	22859.42	7.35%	0.395
kroE100	100	-	22540.71	-	0.225	24062.33	-	0.395
rd100	100	7910.40	8264.57	4.48%	0.350	8592.31	8.62%	0.373
eil101	101	629	669.71	6.47%	0.341	663.34	5.46%	0.416
lin105	105	14383.00	14439.70	0.39%	0.378	15717.41	9.28%	0.476
pr107	107	-	45530.50	-	0.382	49200.34	-	0.492
gr120	120	1610.31	1649.00	2.40%	0.617	1673.48	3.92%	0.704
pr124	124	-	60054.98	-	0.638	60443.36	-	0.711
bier127	127	-	123753.16	-	0.705	134421.42	-	0.895
ch130	130	6110.86	6332.89	3.63%	0.835	6637.55	8.62%	0.917
pr136	136	-	98432.77	-	0.962	102817.53	-	1.023
gr137	137	698.53	720.32	3.12%	0.929	785.68	12.48%	1.121
pr144	144	-	60949.38	-	1.047	68002.54	-	1.38
ch150	150	6532.28	6838.65	4.69%	1.376	6811.18	4.27%	1.803
kroA150	150	26524	28454.71	7.28%	1.240	28802.85	8.59%	1.484
kroB150	150	-	27031.66	-	1.306	27711.55	-	1.685
pr152	152	-	74604.65	-	1.256	83497.80	-	1.680
u159	159	-	46812.24	-	1.426	46778.21	-	1.863
rat195	195	-	2486.40	-	3.010	2485.34	-	3.696
d198	198	-	16763.22	-	3.310	16626.31	-	4.181
kroA200	200	29368	30961.89	5.43%	3.47	31853.12	8.46%	4.376
kroB200	200	-	30700.21	-	3.735	31940.13	-	4.27
gr202	202	486.90	501.86	3.07%	3.906	531.04	9.07%	4.867
ts225	225	-	134677.11	-	6.459	127283.07	-	6.019
tsp225	225	3859	4101.98	6.30%	5.438	4044.10	4.80%	6.369
pr226	226	-	82179.05	-	5.448	88539.92	-	6.369
gr229	229	-	1766.28	-	6.179	1715.27	-	6.82
gil262	262	-	2531.47	-	10.044	2617.82	-	11.246
pr264	264	-	50567.74	-	10.315	56335.80	-	13.189
a280	280	2586.77	2700.48	4.40%	12.188	2732.15	5.62%	14.761
pr299	299	-	50763.26	-	15.292	53061.27	-	18.547
lin318 or linhp318	318	42029	45015.90	7.11%	19.017	45945.51	7.11%	23.714
rd400	400	-	16194.20	-	46.006	16205.49	-	52.065
fl417	417	-	12447.28	-	49.511	12622.99	-	56.902
gr431	431	-	2136.99	-	60.847	2055.55	-	75.058
pr439	439	-	113655.42	-	61.118	119848.10	-	73.475
pcb442	442	50779	53709.98	5.77%	58.644	55895.12	10.08%	72.504
d493	493	-	36882.30	-	98.75	37761.81	-	112.59
att532	532	87550	91103.99	4.06%	125.43	95977.84	9.63%	153.71
ali535	535	-	2155.12	-	134.31	2192.45	-	158.04
pa561	561	-	16039.90	-	153.53	16213.35	-	182.62
u574	574	-	39171.04	-	165.58	41090.75	-	198.43
rat575	575	-	7278.90	-	161.54	7268.50	-	202.29
p654	654	-	36133.26	-	253.23	38589.38	-	286.18
d657	657	-	52351.80	-	288.74	53601.94	-	326.11
gr666	666	-	3366.85	-	300.05	3376.18	-	342.94
u724	724	-	45376.08	-	376.21	45998.42	-	460.97

Opt. = Optimum, Com. = Computational, %Dif. = Percent Difference

جدول ۴: پاسخ شبکه‌های CNN-TSP، CNN-SPSN، SA و Budinich's SOM به ۱۰ مسأله از کتابخانه TSPLIB، بهترین پاسخ‌ها با فونت تیره مشخص شده است.

Tour name	No. of Cities	Optimum Answer	Difference of CNN-TSP (%)	Difference of CNN-SPSN (%)	Difference of SA (%)	Difference of Budinich's SOM (%)
eil51	51	426	4.09%	5.87%	**2.33%**	3.10%
gr96	96	512.31	4.95%	9.54%	4.12%	**2.09%**
kroA100	100	21282	**2.58%**	9.07%	5.94%	3.68%
eil101	101	629	6.47%	5.46%	5.74%	**5.24%**
gr137	137	698.53	**3.12%**	12.48%	8.45%	8.61%
kroA150	150	26524	7.28%	8.59%	**4.31%**	4.36%
kroA200	200	29368	**5.43%**	8.46%	5.61%	6.13%
lin318	318	42029	**7.11%**	9.32%	7.56%	8.19%
Pcb442	442	50779	**5.77%**	10.08%	9.15%	8.43%
att532	532	87550	**4.06%**	9.63%	5.38%	5.67%
Average			**4.57%**	8.85%	5.86%	5.55%

آنچه در بخش ۸ ذکر گردید، با توسعه ساختار، از آن برای حل سایر مسائل پیچیده ترکیبی سود جست. به عنوان مثال، شبکه عصبی CNN-SPSN قادر است مسأله کوتاهترین مسیر با تعداد شهر مشخص را حل نماید. شبیه‌سازی‌ها نشان می‌دهد که خطای نسبی متوسط CNN-TSP و CNN-SPSN نسبت به بهترین پاسخ‌های گزارش شده در کتابخانه TSLIB به ترتیب ۳/۸۱٪ و ۷٪ است. همچنین، شبکه پیشنهادی در مقایسه با دیگر روش‌های محک متداول شامل آبکاری شبیه‌سازی شده و نگاشت خودسازمانده Budinich's SOM عملکرد قابل قبولی از خود نشان می‌دهد. سرعت و دقت مناسب، CNN-TSP و CNN-SPSN را به ابزاری مناسب برای کاربردهای بلادرنگ تبدیل نموده است.

قدردانی

در اینجا از اعضای گروه روباتیک دانشگاه فردوسی مشهد که با طرح صورت مسأله ما را یاری نمودند، قدردانی می‌شود.

مراجع

1 - Crescenzi, P. and Kann, V. (1995). "A compendium of NP optimization problems." *Available: http://www.zvne.fer.hr*

2 - Fujimura, K., Obu-Cann, K. and Tokutaku, H. (1999). "Optimization of surface component mounting on the printed circuit board using SOM-TSP method." *9th International Conference on Artificial Neural Networks*, Vol. 2.

3 - Fujimura, K., Fujiwaki, S., Kwaw, O. C. and Tokutaka, H. (2001). "Optimization of electronic chip-mounting machine using SOM-TSP method with 5 dimensional data." *International Conferences on Info-tech and Info-net (ICII)*, Vol. 4, PP. 26-31.

4 - Mehmet M. K. and Kamoun, A. F. (1993). "Neural networks for shortest path computation and routing in computer networks." *IEEE Trans. Neural Networks*, Vol. 4, No. 6.

5 - Onoyama, T., Maekawa, T., Kubota, S., Taniguchi, Y. and Tsuruta, S. (2002). "Intelligent evolutional algorithm for distribution network optimization." *Proceedings of the International Conference on Control Applications*, Vol. 2, PP. 802 -807.

6 - Banaszak, D., Dale, G. A., Watkins, A. N. and Jordan, J. D. (1999). "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.

7 -Barrel, D., Perrin, J. P., Dombre, E. and Liengeois, A. (1999). "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.

8 - Cheng, C. H., Lee, W. K. and Wong, K. F. (2002). "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.

9 - Gu, J. and Huang, X. (1994). "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.

10 - Tian, F. and Wang, L. (2000). "Chaotic simulated annealing with augmented Lagrange for solving combinatorial optimization problems." *26th Annual Conference of the IEEE Industrial Electronics Society (IECON)*, Vol. 4, PP. 2722 -2725.

11 - Jin, H.D., Leung, K. S., Wong, M. L. and Xu, Z. B. (2003). "An efficient self-organizing map designed by genetic algorithms for the traveling salesman problem." *IEEE Trans. Systems, Man, and Cybernetics-Part B: Cybernetics*, Vol. 33, No. 6, PP. 877-888.

12 - Takahashi, Y. (1998). "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.

13 - Abe, S. and Gee, A. H. (1995). "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.

14 - Bhide, S., John, N. and Kabuka, M. R. (1993). "A real-time solution for the traveling salesman problem using a Boolean neural network." *International Conference on Neural Networks (ICNN'93)*, Vol. 2, PP. 1096-1103.

15 - He, Y. and Wang, L. (2000). "Chaotic neural networks and their applications," *Proceedings of the 3rd World Congress on Intelligent Control and Automation*, Vol. 2, PP. 826-830.

16 - Jiao, L. and Wang, L. (2000). "A novel genetic algorithm based on immunity." *IEEE Trans. on Systems, Man, and Cybernetics-Part A: Systems and Humans*, Vol. 30, No. 5.

17 - Baraglia, R., Hidalgo, J. I. and Perego, R. (2001). "A hybrid heuristic for the traveling salesman problem." *IEEE Trans. on Evolutionary Computation*, Vol. 5, No. 6.

18 - Fogel, D. B. (1993). "Applying evolutionary programming to selected traveling salesman problems." *Cybernetics and Systems*, Vol. 24, PP. 27-36.

19 - Stützle, T. and Dorigo, M. (1999). "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.

20 - Dorigo, M. and Gambardella, L. M. (1997). "Ant colonies for the traveling salesman problem." *BioSystems*, Vol. 43, PP. 73-81.

21 - He, Z., Wei, C., Jin, B., Pei, W. and Yang, L. (1990). "A new population-based incremental learning method for the traveling salesman problem." *Proceedings of Congress on Evolutionary Computation*.

22 - Coy, S. P., Golden, B. L., Runger, G. C. and Wasil, E. A. (1998). "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.

23 - Boettcher, S. and Percus, A. (2000). "Research note: Nature's way of optimizing." *Artificial Intelligence,* Vol. 119, PP. 275-286.

24 - Saadatmand-T., E. A., Akbarzadeh-T., M. R. and Khademi, M. (2001). "A novel hybrid neural network for the traveling salesman problem (TSP)." *9th Iranian Conference on Electrical Engineering (ICEE2001)*, PP. (29-1)-(29-8). (in Farsi).

25 - Saadatmand-T., M. and Akbarzadeh-T., M. R. (2003). "A novel constructive hybrid neural network for the symmetric shortest path problem with specified city number." *5th Iranian Conference on Intelligent Systems (CIS2003)*, PP. 236-243. (in Farsi).

26 - Smith, K. (1996). "An argument for abandoning the traveling salesman problem as a neural network benchmark." *IEEE Trans. Neural Networks*, Vol. 7, No. 6.

27 - Gowda, S. M., Lee, B. W. and Sheu, B. J. (1989). "An Improved Neural Network Approach to the Traveling Salesman Problem." *4th IEEE Region 10 International Conference (TENCON '89)*, PP. 552-555.

28 - Lee, B. W. and Sheu, B. J. (1991). "Modified Hopfield neural networks for retrieving the optimal solution." *IEEE Trans. Neural Networks*, Vol. 2, No. 1.

29 - Kim, S. J. and Choi, H. M. (1993). "An efficient algorithm for traveling salesman problems based on self-organizing feature maps." *2nd IEEE Int. Conference on Fuzzy Systems*.

30 - Vieira, F., Neto, A. and Costa, J. (2002). "An efficient approach of the SOM algorithm to the traveling salesman problem." *Proceedings of 7th Brazilian Symposium on Neural Networks (SBRN'02)*, PP. 152.

31 - Bixby, B. and Reinelt, G. (1995). *TSPLIB-A library of traveling salesman and related problem instances*. Available: ftp://softlib.rice.edu/pub/tsplib/tsp/

32 - Jin, H. D., Leung, K. S., Wong, M. L. and Xu, Z. B. (2003). "An efficient self-organizing map designed by genetic algorithms for the traveling salesman problem." *IEEE Trans. Systems, Man, and Cybernetics-Part B: Cybernetics*, Vol. 33, No. 6, PP. 877-888.

33 - Budinich, M. (1996). "A self-organizing neural network for the traveling salesman problem that is competitive with simulated annealing." *Neural Computation*, Vol. 8, No. 2, PP. 416-424.

واژه های انگلیسی به ترتیب استفاده در متن

1 - Combinatorial Optimization Problems
2 - Traveling Salesman Problem (TSP)
3 - Shortest Path Problem (SP)
4 - Data Partitioning
5 - Deterministic
6 - Probabilistic
7 - Simulated Annealing (SA)
8 - Self Organizing Feature Maps (SOFM)
9 - Boolean
10 - Chaotic
11 - Genetic Algorithms (GA)
12 - Evolutionary Programming
13 - Ant Colony Optimization Systems (ACO)
14 - Population-Based Incremental Learning
15 - Fine-Tuned Learning
16 - Offline
17 - Real Time
18 - Shortest Path Problem with Specified City Number
19 - Symmetric TSP (STSP)
20 - Constructive Neural Network
21 - Constructive Neural Net to Solve TSP
22 - Shortest Path (SP)

ضمیمه الف

در این بخش پایداری شبکه عصبی پیشنهادی و همگرایی آن به یک میمنیمم محلی اثبات شده است. برای این منظور کافی است که نامعادله $e(t+1) \le e(t)$ در فاز بهینه‌سازی آموزش، اثبات شود؛ به این ترتیب تابع انرژی شبکه در طول فاز بهینه‌سازی آموزش سیر نزولی داشته و شبکه پایدار به مفهوم لیاپانوف است. با توجه به اینکه در فاز سازنده، در هر مرحله، تنها یک شهر وارد مسیر می‌شود و تعداد شهرها محدود است، بنابراین این بخش از فرآیند آموزش نمی‌تواند سبب ناپایداری الگوریتم شود. در حقیقت فاز سازنده در هر مرحله، حکم تابع مقداردهی اولیه فاز بهینه‌ساز را ایفا می‌کند. فرض کنید:

۱- در پایان مرحله t، نرون q برنده شده است و خروجی آن p است.

$$ {}^{q}x^{4} = p \qquad (۳۴) $$

۲- شهر q، در مسیر $\vec{\varphi}^{t}$ در محل p_0 بوده است.

$$ \varphi^{t}_{p_0} = q $$

از معادله‌های (۲۰)-(۲۲)، (۲۵) و (۳۴) نتیجه می‌شود:

$$ \begin{cases} {}^{q}v^{3} = {}^{q}w^{3}_{p} \\ {}^{q}k^{3} = p \end{cases} \qquad (۳۵) $$

$$ {}^{q}w^{3}_{p} = \sum_{j=1}^{N} d_{q,j}\ {}^{p}x^{2}_{j} - \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} d_{i,j}\ {}^{p}x^{2}_{i}\ {}^{p}x^{2}_{j} \qquad (۳۶) $$

از (۱۹) و دو معادله قبل می‌توان نتیجه گرفت:

$$ {}^{q}v^{3} = \sum_{i=1}^{N}\sum_{j=1}^{N}\left(d_{q,i} + d_{q,j} - d_{i,j}\right)^{p}x^{1}_{i}\ {}^{\mathrm{p+1}}x^{1}_{j} \qquad (۳۷) $$

می‌دانیم که شهر q، در محل p_0 مسیر $\vec{\varphi}^{t}$ قرار دارد. لذا، با توجه به معادلات (۲۳)، (۲۶) و (۳۷) داریم:

$$ {}^{q}w^{4} = \sum_{i=1}^{N}\sum_{j=1}^{N}\left(d_{q,i} + d_{q,j} - d_{i,j}\right)^{(p_0+1)}x^{1}_{i}\ {}^{(\mathrm{p_0-1})}x^{1}_{j} $$

$$ {}^{p_0}x^{1}_{q} = 1 \qquad (۳۸) $$

$$ {}^{k}v^{4} = \sum_{i=1}^{N}\sum_{j=1}^{N}\left[\left(d_{k,i} + d_{k,j} - d_{i,j}\right)^{p}x^{1}_{i}\ {}^{\mathrm{p+1}}x^{1}_{j}\right] - \left[\left(d_{k,i} + d_{k,j} - d_{i,j}\right)^{(p_0+1)}x^{1}_{i}\ {}^{(\mathrm{p_0-1})}x^{1}_{j}\right] \qquad (۳۹) $$

فرض کنید در مسیر $\vec{\varphi}^{t}$ داشته باشیم:

$$ \varphi^{t}_{p_0-1} = a \qquad (۴۰) $$

$$ \varphi^{t}_{p_0+1} = b \qquad (۴۱) $$

$$ \varphi^{t}_{p} = c \qquad (۴۲) $$

$$ \varphi^{t}_{p+1} = d \qquad (۴۳) $$

در نتیجه:

$$ {}^{q}v^{4} = (d_{q,c} + d_{q,d} - d_{c,d}) - (d_{q,a} + d_{q,b} - d_{a,b}) < 0 \qquad (۴۴) $$

$$ d_{q,c} + d_{q,d} + d_{a,b} < d_{q,a} + d_{q,b} + d_{c,d} \qquad (۴۵) $$

انرژی شبکه در مرحله tام برابر است با:

$$ e(t) = \left[\sum_{r=1}^{N}\sum_{i=1}^{N}\sum_{\substack{j=1 \\ r\neq p_0, p_0+1, p+1}}^{N} d_{i,j}\ {}^{r}x^{1}_{i}\ {}^{(r-1)}x^{1}_{j}\right] + d_{q,a} + d_{b,q} + d_{d,c} \qquad (۴۶) $$

$$ e(t) = C_1 + d_{q,a} + d_{q,b} + d_{c,d} \qquad (۴۷) $$

انرژی شبکه در مرحله $(t+1)$ نیز بروش مشابه محاسبه می‌شود:

$$ e(t+1) = C_2 + d_{q,c} + d_{q,d} + d_{a,b} \qquad (۴۸) $$

تفاوت دو مسیر $\vec{\varphi}^{t}$ و $\vec{\varphi}^{t+1}$ در مکان شهر q است. q در $\vec{\varphi}^{t}$، بین a و b قرار دارد در حالی که در $\vec{\varphi}^{t+1}$، بین c و d قرار گرفته است. اگر از مسیر $\vec{\varphi}^{t}$ کمان‌های (a,q)، (q,b) و (c,d) حذف شوند و کمان‌های (c,q)، (q,d) و (a,b) به آن اضافه گردند، مسیر $\vec{\varphi}^{t+1}$ حاصل می‌شود. بنابراین سایر قسمت‌های این دو مسیر مشترکند. C_1 و C_2 نیز برابر با طول همین قسمت‌های مشترک هستند. بنابراین از معادله‌های (۴۵)، (۴۷) و (۴۸) می‌توان نامعادله $e(t+1) \le e(t)$ را نتیجه گرفت.