حل مسأله سینماتیک معکوس در رباتهای افزونه با استفاده از اتوماتاهای یادگیر

سید علیرضا متولیان محمدرضا میبدی سعید شیری

دانشکده مهندسی کامپیوتر و فناوری اطلاعات
دانشگاه صنعتی امیرکبیر
تهران ایران
(motevalian, meybodi, shiry)@ce.aut.ac.ir

چکیده: هدف از حل مسأله سینماتیک معکوس تعیین مقادیر اتصالات ربات برای قرار دادن نوک ابزار آن در یک نقطه و جهت معین از فضای کاری ربات میباشد. جز برای انواع خاصی از رباتها، مسأله سینماتیک معکوس با روشهای تحلیلی و ریاضی قابل حل نیست و به همین علت روشهای عددی گوناگونی برای حل آن ارائه شده است. در این مقاله الگوریتمی مبتنی بر *اتوماتاهای یادگیر* برای حل مسأله سینماتیک معکوس رباتهای افزونه پیشنهاد میشود. الگوریتم پیشنهادی دارای دو مرحله می‌باشد. در مرحله اول با استفاده از مجموعه‌ای از اتوماتاهای یادگیر، تقریبی از مقادیر اتصالات ربات برای قرار دادن نوک ابزار آن در یک نقطه و جهت معین از فضای کاری ربات بدست می‌آید و در مرحله دوم با استفاده از مقادیر تقریبی بدست آمده در مرحله اول مقادیر دقیق اتصالات محاسبه می‌شود. نتایج شبیه‌سازی نشان می‌دهد که با انتخاب صحیح پارامترها، این الگوریتم می‌تواند کارایی خوبی داشته و به جوابهای نزدیک به بهینه همگرا گردد. الگوریتم پیشنهادی با یک الگوریتم ژنتیکی که یکی از بهترین الگوریتمهای گزارش شده برای این منظور است، مقایسه می‌شود.

کلمات کلیدی: اتوماتاهای یادگیر، رباتیک، ربات افزونه، سینماتیک معکوس

۱- مقدمه

مسأله سینماتیک معکوس[1] [2] که عبارتست از تعیین مقادیر اتصالات ربات برای قرار دادن نوک ابزار آن در یک نقطه و جهت معین از فضای کاری ربات، یکی از مهمترین مسایل مطرح در طراحی رباتها می‌باشد. متأسفانه جز برای انواع خاصی از رباتها، مسأله سینماتیک معکوس با روشهای تحلیلی و ریاضی قابل حل نیست و به همین علت روشهای عددی گوناگونی برای حل آن ارائه شده است. البته با پیچیده‌تر شدن ساختار رباتها و افزایش درجات آزادی آنها و همچنین اعمال محدودیتهای فیزیکی بیشتر برروی فضای کاری ربات و نحوه حرکت آن، استفاده از روشهای عددی رایج بسیار مشکل و پیچیده می‌شود.

رباتهای افزونه[2] رباتهایی هستند که تعداد درجات آزادی آنها بیشتر از حداقل لازم، برای پوشش فضای کاری آنهاست. مثلاً برای قرار دادن نوک ابزار در یک نقطه معین از یک فضای کاری ۳ بعدی، ۳ درجه آزادی و برای تعیین جهت آن علاوه بر موقعیت، ۶ درجه آزادی کافی می‌باشد. بنابراین برای مورد اول، رباتهای دارای بیش از ۳ اتصال و برای مورد دوم رباتهای دارای بیش از ۶ اتصال افزونه هستند. این اتصالهای اضافی انعطاف‌پذیری بیشتری را به ربات داده و وجود چندین پاسخ برای سینماتیک معکوس آنها، امکان در نظر گرفتن معیارهای دیگر مانند جلوگیری از برخورد با موانع، میزان تغییر اتصالات و میزان انرژی مصرفی را برای کمینه نمودن هزینه فراهم می‌نماید. اما از سوی دیگر وجود اتصالات زیاد در این رباتها، یافتن پاسخهای سینماتیک معکوس آنها را بسیار مشکل می‌سازد. در حقیقت، مسأله سینماتیک معکوس رباتهای افزونه، در زمره مسایل بهینه‌سازی پیچیده می‌باشد که یافتن نقطه بهینه آن توسط الگوریتمهای تکراری و عددی رایج مانند کاهش گرادیان، به علت وجود پاسخهای نامطلوب زیاد - در اثر اعمال قیود و معیارهای متعدد در تعریف مسأله- ممکن نمی‌باشد. به همین منظور، راه‌حلهایی مبتنی بر روشهای مکاشفه‌ای مختلف مطرح گردیده است که مهمترین این روشها شبکه‌های عصبی و الگوریتمهای ژنتیکی می‌باشند.

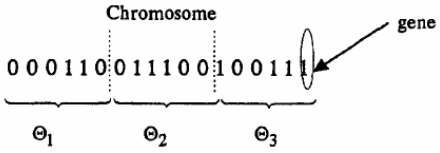
اغلب روشهای مبتنی بر شبکه‌های عصبی از شبکه‌های انتشار خطا به عقب (BP) و یادگیری با نظارت استفاده کرده‌اند[4][5][9][10]. در دسته اول از این روشها سعی شده است تا با ورود مختصات فضایی تعدادی نقطه به عنوان خروجی‌های مطلوب شبکه، و نیز مقدار متغیرهای

[1] Inverse Kinematics Problem

[2] Redundant Robots

اتصالات ربات برای قرار دادن نوک ابزار در این نقاط، شبکه بگونه‌ای مورد آموزش قرار بگیرد که خطای آن از حد معینی کمتر باشد[4]. دسته دوم روشهای مبتنی بر شبکه BP، از شبکه‌های BP معکوس[3] استفاده می‌کند. در این شبکه‌ها، پس از اعمال یک ورودی و محاسبه خطای ایجاد شده در خروجی، براساس خطای بدست آمده علاوه بر وزنها، ورودی نیز بروز رسانی می‌گردد. در این شبکه‌ها، فضای جواب همان فضای ورودیهاست و هدف یادگیری بهترین ورودی می‌باشد. در [10]، از این شبکه‌های معکوس برای یادگیری مقادیر اتصالات استفاده شده است. علاوه بر روشهای فوق، شبکه‌های هاپفیلد هم به عنوان یک ابزار بهینه‌سازی، برای حل مسأله سینماتیک معکوس بکار رفته‌اند. در [3]، از شبکه هاپفیلد برای تخمین شبه‌معکوس[4] ماتریس ژاکوبین استفاده شده است.

در روشهای مبتنی بر الگوریتمهای ژنتیکی، از منظری دیگر با مسأله سینماتیک معکوس برخورد شده است. در این روشها مسأله سینماتیک معکوس بصورت یک مسأله بهینه‌سازی تعریف شده و سپس با یک الگوریتم ژنتیکی پاسخ نزدیک به بهینه برای آن بدست می‌آید[1][7][8]. در این روشها، ابتدا فضای مقادیر هریک از متغیرهای اتصالات به مقادیر گسسته‌ای تبدیل می‌شود. سپس این مقادیر، بصورت باینری درآمده و از کنار هم گذاردن این مقادیر باینری رشته کروموزومهای الگوریتم ژنتیکی بدست می‌آید(شکل ۱).



شکل ۱) یک کروموزوم

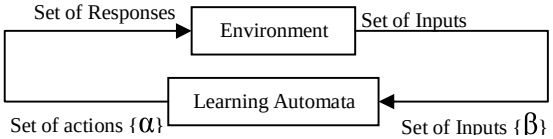
در این مقاله الگوریتمی مبتنی بر *اتوماتاهای یادگیر* برای حل مسأله سینماتیک معکوس رباتهای افزونه پیشنهاد می‌شود. الگوریتم پیشنهادی دارای دو مرحله می‌باشد. در مرحله اول با استفاده از مجموعه‌ای از اتوماتاهای یادگیر، تقریبی از مقادیر اتصالات ربات برای قرار دادن نوک ابزار آن در یک نقطه و جهت معین از فضای کاری ربات بدست می‌آید و در مرحله دوم با استفاده از مقادیر تقریبی بدست آمده در مرحله اول مقادیر دقیق اتصالات محاسبه می‌شود. نتایج شبیه‌سازی نشان می‌دهد که با انتخاب صحیح پارامترها، این الگوریتم می‌تواند کارایی خوبی داشته و به جوابهای نزدیک به بهینه همگرا گردد. الگوریتم پیشنهادی با یک الگوریتم ژنتیکی که یکی از بهترین الگوریتمهای گزارش شده برای حل این مسأله است مقایسه می‌شود.

ادامه گزارش بدین صورت سازماندهی شده است. در بخش ۲ به معرفی اجمالی اتوماتاهای یادگیر می‌پردازیم. بخش ۳ الگوریتم پیشنهادی ارایه می‌گردد. نتایج شبیه سازیها در بخش ۴ آمده است و بخش نهایی نتیجه گیری می‌باشد.

۲- اتوماتاهای یادگیر[5]

یک اتوماتای یادگیر یک مدل انتزاعی است که تعداد محدودی عمل را می‌تواند انجام دهد. هر عمل انتخاب شده توسط محیطی احتمالی ارزیابی می‌گردد و پاسخی به اتوماتای یادگیر داده می‌شود. اتوماتای یادگیر از این پاسخ استفاده نموده و عمل خود را برای مرحله بعد انتخاب می‌کند. اتوماتاهای یادگیر به دو دسته عمده اتوماتای یادگیر با ساختار ثابت و اتوماتای یادگیر با ساختار متغیر تقسیم می‌شوند.

محیط: محیط را می‌توان توسط سه‌تایی $E \equiv \{\alpha, \beta, c\}$ تعریف نمود که در آن $\alpha \equiv \{\alpha_1, \alpha_2, ..., \alpha_r\}$ مجموعه ورودیها، $\beta \equiv \{\beta_1, \beta_2, ..., \beta_r\}$ مجموعه خروجیها و $c \equiv \{c_1, c_2, ..., c_r\}$ مجموعه احتمالهای جریمه شدن می‌باشد. هرگاه β_i دو مقداری باشد، محیط از نوع P می‌باشد. در چنین محیطی $\beta_i = 1$ به عنوان جریمه و $\beta_i = 0$ به عنوان پاداش در نظر گرفته می‌شود. در محیط از نوع Q، $\beta_i(n)$ می‌تواند بطور گسسته یکی از مقادیر محدود در فاصله [0,1] را اختیار کند و در محیط از نوع S، $\beta_i(n)$ متغیر تصادفی در فاصله [0,1] است، یعنی $\beta_i(n) \in [0,1]$.



شکل ۲) ارتباط بین اتوماتای یادگیر و محیط

c_i احتمال اینکه عمل α_i نتیجه نامطلوب داشته باشد، می‌باشد. در یک محیط پایدار مقادیر c_i بدون تغییر باقی می‌ماند. حال آنکه در یک محیط ناپایدار این مقادیر در طی زمان تغییر می‌کنند. شکل ۲ ارتباط بین اتوماتای یادگیر و محیط را نشان می‌دهد.

اتوماتای یادگیر با ساختار متغیر: اتوماتای یادگیر با ساختار متغیر توسط ۵ تایی $LA \equiv \{\alpha, \beta, p, T, c\}$ که در آن $\alpha \equiv \{\alpha_1, \alpha_2, ..., \alpha_r\}$ مجموعه عمل‌های اتوماتا، $\beta \equiv \{\beta_1, \beta_2, ..., \beta_r\}$ مجموعه ورودیهای اتوماتا، $p \equiv \{p_1, p_2, ..., p_r\}$ بردار احتمال انتخاب عمل،

[3] Inverted BP Neural Networks
[4] Pseudo-Inverse
[5] Learning Automata (LA)

$T \equiv p(n+1) = T[\alpha(n), \beta(n), p(n)]$ الگـــــوریتم یـــــادگیری و $c \equiv \{c_1, c_2, ..., c_r\}$ مجموعه احتمالهای جریمه‌شدن مـی‌باشـند. هرگـاه β مجموعه دو عـضوی باشـد، محـیط از نـوع P مـی‌باشـد. در چنـین محیطی $\beta_1 = 0$ به عنوان پاداش و $\beta_2 = 1$ به عنوان جریمه در نظر گرفته می‌شود. در محیط از نوع Q، $\beta(n)$ می‌تواند به طور گسسته یک مقدار از مقادیر محدود در فاصله [0,1] و در محـیط از نـوع S، $\beta(n)$ متغیر تصادفی در فاصله [0,1] است. در این نوع از اتوماتاهـا، اگـر عمـل α_i در مرحله nام انتخاب شود و پاسخ مطلوب از محیط دریافـت نمایـد، احتمال $p_i(n)$ افزایش و سایر احتمالها کاهش مـی‌یابنـد. بـرای پاسـخ نامطلوب $p_i(n)$ کاهش و سایر احتمالها افزایش می‌یابند. در هـر حـال، تغییرات بگونه‌ای صورت می‌پذیرد که حاصل جمـع $p_i(n)$ هـا همـواره ثابت و مساوی یک باقی بماند. الگوریتم زیر یـک نمونـه از الگـوریتم‌هـای یادگیری خطی در محیط نوع P می‌باشد:

الف- پاسخ مطلوب:

$$p_i(n+1) = p_i(n) + a[1 - p_i(n)]$$
$$p_j(n+1) = (1-a)p_j(n) \qquad \forall j,\ j \neq i$$

ب- پاسخ نامطلوب:

$$p_i(n+1) = (1-b)p_i(n)$$
$$p_j(n+1) = \frac{b}{r-1} + (1-b)p_j(n) \qquad \forall j,\ j \neq i$$

در روابط فوق، a پارامتر پاداش و b پارامتر جریمه مـی‌باشـد. بـا توجه به مقادیر a و b سه حالت را می‌توان در نظر گرفت. اگـر a و b بـا هم برابر باشند، الگوریتم را L_{RP}، چنانچه b از a خیلی کـوچکتر باشـد، الگـوریتم را $L_{RεP}$ و اگـر b صـفر باشـد، الگـوریتم را L_{RI} مـی‌گوینـد. الگوریتم یادگیری فوق را می‌توان به محیط نوع S که در آن پاسخ محیط، $\beta(n)$، مقـداری پیوسـته اسـت، تعمـیم داد. الگـوریتم یـادگیری زیـر الگوریتم یادگیری L_{RI} در محیط S(SL_{RI}) می‌باشد:

$$p_i(n+1) = p_i(n) - a(1-\beta(n))p_i(n) \qquad \alpha(n) \neq \alpha_i$$
$$p_i(n+1) = p_i(n) + a(1-\beta(n))\sum_{j \neq i} p_j(n) \qquad \alpha(n) = \alpha_i$$

برای اطلاعات بیشتر در بـاره اتوماتاهـای یـادگیر مـی‌تـوان بـه [6] مراجعه کرد.

۳- الگوریتم پیشنهادی

الگوریتم پیشنهادی دارای دو مرحله می‌باشد. در مرحله اول تقریبی از مقادیر اتصالات ربات برای قرار دادن نوک ابزار آن در یک نقطه و جهت معین از فضای کاری ربات بدست مـی‌آیـد. در مرحلـه دوم بـا اسـتفاده از مقادیر تقریبی بدست آمده از مرحله اول مقادیر دقیق اتـصالات محاسـبه می‌شود. الگوریتم در مرحلـه اول از یـک مجموعـه از اتوماتاهـای یـادگیر استفاده می‌کند. در این مجموعه، به تعداد اتصالات ربات، اتوماتای یادگیر وجود دارد. هر اتوماتای یادگیر برای یکی از متغیرهای مربوط بـه یکـی از اتصالها در نظر گرفته شده است و عملهای آن برابر مقادیری است که این متغیر ممکن است اختیار کند. در این الگوریتم، فرض برایـنست که فـضای مقادیر متغیرهای اتصالات، گسسته و تعداد مقادیر ممکن برای هر متغیر محدود می‌باشد. فرض گسسته بودن، بخصوص در کنترلهای دیجیتال کـه اساساً با مقادیر گسسته سروکار دارند کاملاً منطقی و مفید است. علاوه بر این، بدون از دست دادن کلیت مسأله، فـرض مـی‌کنـیم اتـصالهای ربـات همگی چرخشی[6] می‌باشند.

عملکرد مرحله اول الگوریتم بدین شرح است: در ابتدا بـا اسـتفاده از تابع سینماتیک مستقیم، ربات محل جاری نـوک ابـزار را بدسـت آورده و فاصله اقلیدسی آن تا نقطه مقصد (نقطه‌ای که می‌خـواهیم نـوک ابـزار در آنجا قرار بگیرد) را محاسبه می‌کند. با نرمال‌سـازی ایـن فاصـله (تبـدیل مقیاس به محدوده [۰,۱])، یک سیگنال امتیازدهی محاسبه و به تمـامی LAها داده می‌شود. هر اتوماتای یادگیر با استفاده از این سـیگنال، بـردار احتمال عمل خود را بروز نمـوده و سـپس عمـل دیگـری(براسـاس بـردار احتمال عمل) را انتخاب می‌نماید. این عمل بروزرسـانی، انتخـاب عمـل و تعیین امتیاز آنقدر تکرار می‌شود تا فاصله میان نوک ابزار و نقطه مقصد از یک حد آستانه از پیش تعیین شده کمتر گشته و یا تعداد تکرارها از حـد معینی فراتر رود که در این صورت مرحله اول الگوریتم همگرا نشده است.

ممکن است در نظر اول این الگـوریتم کامـل بنظـر برسـد و انتظـار داشته باشیم به علت این طراحی خاص سیگنال امتیازدهی- بگونه‌ای کـه هرچه نوک ابزار به نقطه مقصد نزدیک‌تر باشد پاداش بیـشتری(سـیگنال نزدیک‌تر به صفر) داده شود- اتوماتاهای یادگیر سعی خواهنـد نمـود ایـن فاصله را به حداقل رسانند (پاداش مجموع را بیشینه کنند). اما باید توجه داشت که در این الگوریتم، تعداد عملهای **LA**ها می‌تواند بسیار زیـاد (بـا فرض قابلیت چـرخش ۳۶۰ درجـه اتـصالات و دقـت 0.1mm، تـا ۳۶۱۰ عمل) شود و زیاد بودن تعـداد عملهـا، مـانع از رسـیدن بـه پاسـخ بهینـه سراسری گشته و همگرایی الگوریتم را بسیار ضعیف می‌کند. دو مرحله‌ای بودن الگوریتم برای رفع همین مشکل بـوده اسـت. در مرحلـه اول، ابتـدا دقت[7] فضا مقـادیر اتـصالات و حدآسـتانه الگـوریتم را تـا حـدی کـاهش می‌دهیم. این امر منجر به کاهش تعداد عملهای LAهـا تـا حـد مناسـبی می‌شود. مثلاً بجای در نظر گرفتن دقت ۰/۱ درجه، دقت را تـا ۱۰ درجـه کاهش می‌دهیم که در این صورت تعـداد عملهـای هـر **LA** حـداکثر ۳۷

[6] Revolute
[7] Precision

خواهد بود. سپس الگوریتم اشاره شده در بالا را با این حدآستانه و دقت اجرا کرده و در طول اجرا بهترین جواب بدست آمده را ذخیره می‌کنیم. در مرحله دوم، از یک مجموعه اتوماتای یادگیر دیگر استفاده کرده و سعی می‌کنیم فاصله نوک ابزار بدست آمده در مرحله اول را تا نقطه مقصد کاهش دهیم. در این مرحله، برخلاف مرحله اول، عملهای LAها برابر مقادیر متغیرهای اتصالات نمی‌باشد. بلکه هر عمل معادل یک تغییر مقدار برای متغیر اتصال مربوطه نسبت به مقدار بدست آمده در مرحله اول می‌باشد. تعداد عملهای هر **LA** در اینجا برابر است با دقت در نظر گرفته شده در مرحله اول، تقسیم بر دقت نهایی مورد نظر. به عنوان نمونه برای مثال فوق، تعداد عملها ۱۰۰ خواهد بود. در حقیقت، در مرحله دوم، سعی می‌کنیم با ایجاد یک تغییر مکان در نوک ابزار- نسبت به مکان بدست آمده در مرحله اول- به نقطه مقصد نزدیک‌تر شویم. در ادامه به شرح دقیق‌تر مراحل اول و دوم الگوریتم پیشنهادی می‌پردازیم.

For Each LA A_i **in** the set of Learning Automata **do in Parallel**
$r_i = ceiling(\frac{U_i - L_i}{precision})$
End For
RandomInit(Θ)
$\Theta_{best} = \Theta$
tooltip = *f*(Θ) //Forward Kinematics Function.
curDist = Distance(*tooltip*, *destination*)
minDist = *curDist*
n = 1
While (*n* <= MaxIteration And curDist > τ_1)
For Each LA A_i **in** the set of Learning Automata **do in Parallel**
Choose Action $\theta_i(n)$ according to $\underline{p}_i(n)$
End For
tooltip = *f*(Θ)
curDist = Distance(*tooltip*, *destination*)
If *curDist* < *minDist* **Then**
$\Theta_{best} = \Theta$
End If
$\beta(n) = \frac{curDist}{totalRobotLength + Dis\tan ce(destination,[0\,0\,0]^T)}$
For Each LA A_i **in** the set of Learning Automata **do in Parallel**
$p_{i\theta_i(n)}(n) = p_{i\theta_i(n)}(n) + a(1-\beta(n))(1-p_{i\theta_i(n)}(n))$
For Each *j* **in** α_i **do in Parallel**
$p_{ij}(n) = p_{ij}(n) - a(1-\beta(n))p_{ij}(n)$
End For
End For
End While

شکل ۳) شبه کد مرحله اول الگوریتم پیشنهادی

فرض کنید m تعداد اتصال‌های ربات، $\Theta = \{\theta_1,...,\theta_m\}$ مجموعه متغیرهای اتصالات، r_i تعداد مقادیر ممکن برای اتصال iام و $\alpha_i = \{\alpha_{ij(i)} \mid j(i) = 1,...,r_i\}$ مجموعه مقادیر ممکن U_i و L_i بترتیب حدود پایین و بالای محدوده مقادیر ممکن برای متغیر i ($i = 1,...,m$) باشند. در این صورت مجموعه اتوماتاهای یادگیر $A = \{A_1, A_2,..., A_m\}$، دارای m اتوماتای یادگیر هریک با r_i عمل خواهد بود. $\underline{p}(n)$ بیانگر عمل جاری اتوماتای یادگیر A_i و $\theta_i(n)$ بردار احتمال عمل آن در زمان n می‌باشد. علاوه براین، τ_1 و τ_2 بترتیب دو آستانه مرحله اول و مرحله دوم الگوریتم می‌باشند که عبارتند از حداکثر خطای قابل قبول در مرحله اول و مرحله دوم. با توجه به نکات فوق، شبه کد الگوریتم مرحله اول در شکل ۳ آمده است. در این الگوریتم *totalLength* مجموع طول تمام بازوهای ربات می‌باشد. در انتهای اجرای مرحله اول، بهترین مقادیر بدست آمده برای اتصالات در Θ_{best} قرار دارد. همانگونه که از شبه کد پیداست، سیگنال امتیازدهی از رابطه $\beta(n) = \frac{curDist}{totalRobotLength + Dis\tan ce(destination,[0\,0\,0]^T)}$ بدست می‌آید که هم همواره مقداری در محدوده [۰,۱] است و هم هرچه نوک ابزار به مقصد نزدیکتر باشد، این مقدار کوچکتر می‌شود (در مقصد صفر است).

For Each LA A_i **in** the set of Learning Automata **do in Parallel**
$r_i = ceiling(\frac{precision}{0.1})$
End For
RandomInit(ΔΘ)
$\Delta\Theta_{best} = \Delta\Theta$
tooltip = *f*(Θ_{best}+ΔΘ) //Forward Kinematics Function.
curDist = Distance(*tooltip*, *destination*)
minDist = *curDist*
n = 1
While (*n* <= MaxIteration And curDist > τ_2)
For Each LA A_i **in** OSCLA **do in Parallel**
Choose Action $\Delta\theta_i(n)$ according to $\underline{p}_i(n)$
End For
tooltip = *f*(Θ_{best}+ΔΘ)
curDist = Distance(*tooltip*, *destination*)
If *curDist* < *minDist* **Then**
$\Delta\Theta_{best} = \Delta\Theta$
End If
$\beta(n) = \min(1, \frac{\text{curDistance}}{\text{k}\times\text{totalLength}})$
For Each LA A_i **in** the set of Learning Automata **do in Parallel**
$p_{i\theta_i(n)}(n) = p_{i\theta_i(n)}(n) + a(1-\beta(n))(1-p_{i\theta_i(n)}(n))$
For Each $j \neq \theta_i(n)$ in α_i **do in Parallel**
$p_{ij}(n) = p_{ij}(n) - a(1-\beta(n))p_{ij}(n)$
End For
End For
End While
Output Θ_{best}+$\Delta\Theta_{best}$ as the final Result

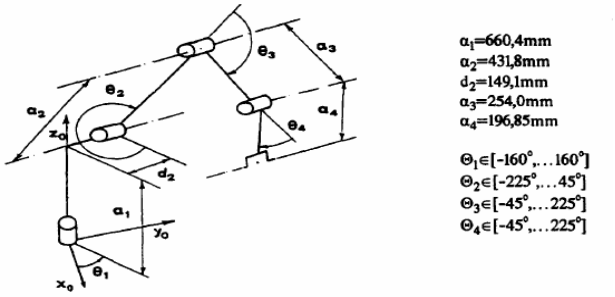
شکل ۴) شبه کد مرحله دوم الگوریتم پیشنهادی

مرحله دوم الگوریتم همانند مرحله اول الگوریتم از m اتوماتای یادگیر استفاده می‌کند با این تفاوت که مقادیر انتخاب شده توسط LAها بیانگر تغییرات مقادیر متغیرها نسبت به مقادیر بدست آمده در مرحله اول(Θ_{best}) هستند. شکل ۴ شبه کد این مرحله را نشان می‌دهد. در این

مرحله، امتیاز LAها از رابطه $\beta(n) = \min(1, \frac{curDist}{k * totalRobotLength})$ بدست آمده است که در آن k پارامتری است که با آزمون و خطا بگونه‌ای تعیین می‌شود که مقادیر $\frac{curDist}{k * totalRobotLength}$ به ازاء $curDist$های مختلف تقریباً بطور یکنواختی در محدوده [۰،۱] توزیع شوند.

۴- نتایج شبیه‌سازی

در این بخش به بررسی نتایج بدست آمده از اجرای الگوریتم بر روی یک ربات نمونه که در [8] گزارش شده است می‌پردازیم. این ربات که در شکل ۵ نشان داده شده است دارای ۴ اتصال چرخشی می‌باشد. چون برای قرار دادن نوک ابزار در هر نقطه دلخواهی از فضا(بدون در نظر گرفتن جهت آن) فقط به ۳ اتصال چرخشی نیاز می‌باشد، این ربات یک ربات افزونه است. پارامترهای سینماتیکی(غیر صفر) این ربات و نیز محدوده مقادیر ممکن برای هر متغیر اتصال در شکل ۵ مشخص شده‌اند.



شکل ۵) ربات افزونه نمونه با ۴ اتصال چرخشی

در شبیه‌سازیهای انجام گرفته، از اتوماتاهای یادگیر $S\text{-}L_{RI}$ با نرخ یادگیری ۰/۰۱ استفاده شده است. علاوه براین، مقادیر آستانه مراحل اول و دوم الگوریتم بترتیب $\tau_1 = 10mm$ و $\tau_2 = 3mm$ در نظر گرفته شده است. در این گزارش، برای بررسی کارایی الگوریتم از داده‌های آزمایشی جدول ۱ استفاده شده است[8].

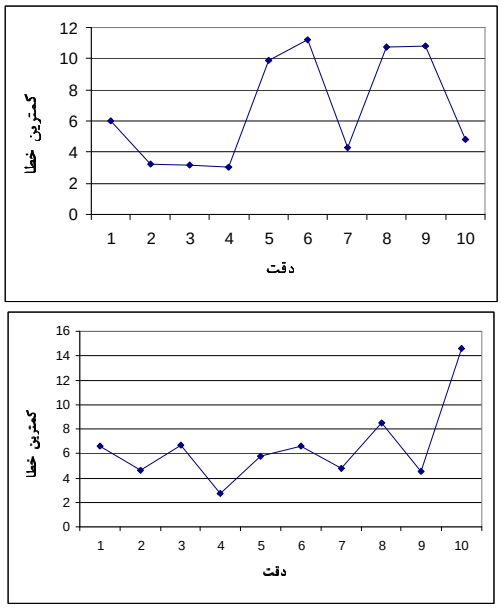
جدول ۱) داده‌های آزمایشی

نقاط مقصد(mm)	نقطه ۱	نقطه ۲
X	۶۸۵/۸	-۲۵/۴
Y	۱۵۲/۴	۳۵۵/۶
Z	۶۶۰/۴	۶۳۵/۰

جدول ۲ نتیجه اجرای الگوریتم را بر روی داده‌های آزمایشی به ازاء مقادیر مختلف پارامتر دقت(*precision*) نشان می‌دهد. هر یک از مقادیر گزارش شده در این جدول متوسط خطای ۲۰ اجرای الگوریتم به ازاء مقدار معینی از پارامتر دقت می‌باشد. همانطور که در این جدول نشان داده شده است کمترین خطا برای هر دو نقطه نمونه به ازاء دقت ۴ بدست آمده است و همچنین میانگین خطا برای نمونه دوم نیز به ازاء همین مقدار دقت، کمینه می‌باشد. با مشاهده نمودارهای ۱ و ۲ که تغییرات خطا را نسبت به تغییرات پارامتر دقت نشان می‌دهد می‌توان گفت، برای مسأله در نظر گرفته شده، مقدار مطلوب برای پارامتر دقت ۴ می‌باشد. آنچه که از مشاهده نتایج در جدول ۲ می‌توان نتیجه گرفت این است که با کاهش زیاد دقت، جوابهای بدست آمده در مرحله اول آنقدر از نقطه مقصد دور می‌شوند که مرحله دوم الگوریتم نمی‌تواند کمکی به یافتن پاسخهای قابل قبول بنماید و خطای بالایی ایجاد می‌شود. از سوی دیگر با بالا بردن بیش از حد دقت، فضای جستجوی بسیار بزرگ می‌شود و مرحله اول الگوریتم در کمینه‌های محلی گیر می‌افتد و پاسخ بدست آمده آنقدر دور از مقصد خواهد بود که مرحله دوم الگوریتم قادر به یافتن جواب مطلوب نمی‌باشد. اما با تنظیم مناسب پارامتر دقت، الگوریتم در مرحله اول به نزدیکی مقصد رسیده و در مرحله دوم نیز بخوبی فاصله باقیمانده را کاهش می‌دهد.

جدول ۲) متوسط و کمترین خطای بدست آمده برحسب میلیمتر برای دو نقطه نمونه به ازاء مقادیر مختلف دقت

دقت الگوریتم		۱	۲	۳	۴	۵	۶	۷	۸	۹	۱۰
نمونه ۱	کمترین خطا	۸/۵۱	۱۰/۳	۱۰/۶	۳۸/۲	۹/۹	۱۲/۹	۴/۳	۲۳/۲	۱۶/۱	۴/۸
مرحله ۱	متوسط خطا	۲۰/۳	۲۱/۱	۲۷/۰	۴۲/۹	۲۴/۸	۲۰/۶	۲۰/۰	۲۶/۸	۲۲/۹	۱۰/۳
نمونه ۱	کمترین خطا	۶/۰	۳/۳	۳/۲	**۳/۱**	۹/۹	۱۱/۲	۴/۳	۱۰/۸	۱۰/۸	۴/۸
مرحله۲	متوسط خطا	۱۸/۲	**۸/۸**	۱۲/۷	**۱۱/۳**	۱۷/۴	۱۸/۰	۱۷/۰	۲۰/۷	۱۹/۰	۹/۸
نمونه ۲	کمترین خطا	۱۲/۵	۱۷/۴	۱۰/۲	۵/۶	۷/۸	۶/۶	۱۳/۰	۲۶/۲	۱۴/۰	۲۰/۳
مرحله ۱	متوسط خطا	۲۲/۸	۲۴/۲	۲۶/۰	۹/۵	۲۳/۰	۲۵/۰	۲۴/۱	۳۴/۰	۲۲/۵	۲۸/۶
نمونه ۲	کمترین خطا	۶/۶	۴/۶	۶/۸	**۲/۷**	۵/۸	۶/۶	۴/۸	۸/۵	۴/۵	۱۴/۶
مرحله ۲	متوسط خطا	۱۸/۵	۹/۳	۱۳/۷	**۸/۶**	۱۵/۳	۱۶/۱	۱۵/۷	۲۰/۴	۲۰/۳	۲۵/۹



نمودار ۱) تغییرات خطای کمینه نسبت به تغییرات پارامتر دقت برای نقاط نمونه آزمایشی ۱(نمودار بالا) و ۲(نمودار پایین)

در جدول ۳ نتایج الگوریتم پیشنهادی با نتایج الگوریتم ژنتیکی گزارش شده در [8] مقایسه شده است. همانطور که در جدول آمده است الگوریتم پیشنهادی مقادیر کمینه بهتری در مقایسه با الگوریتم ژنتیکی پیدا می‌کند. برای نمونه دوم آزمایشی الگوریتم پیشنهادی خطایی تقریباً نزدیک به خطای الگوریتم ژنتیکی تولید می‌کند(حدود ۰/۵ میلیمتر) اما برای نمونه اول آزمایشی خطا برای الگوریتم پیشنهادی بسیار کمتر از مقدار خطا برای الگوریتم ژنتیکی می‌باشد(حدود ۳/۶ میلیمتر) ولی در عوض، متوسط خطا برای الگوریتم ژنتیکی کمتر است.

جدول ۳) مقایسه نتایج با الگوریتم ژنتیکی [8]

		الگوریتم پیشنهادی	الگوریتم ژنتیکی
نمونه ۱	کمترین خطا(mm)	3.1	6.7
	متوسط خطا(mm)	11.3	7.1
نمونه ۲	کمترین خطا(mm)	2.7	2.2
	متوسط خطا(mm)	8.6	4.6

۵- نتیجه گیری

در این مقاله یک الگوریتم مبتنی بر اتوماتاهای یادگیر برای حل مسأله سینماتیک معکوس رباتهای افزونه ارائه شد. الگوریتم پیشنهادی دارای دو مرحله می‌باشد. در مرحله اول با استفاده از مجموعه‌ای از اتوماتاهای یادگیر تقریبی از مقادیر اتصالات ربات بدست می‌آید و در مرحله دوم با استفاده از این مقادیر تقریبی، مقادیر دقیق اتصالات محاسبه می‌شود. نتایج شبیه‌سازیها نشان می‌دهد که با انتخاب صحیح پارامتر دقت، این الگوریتم می‌تواند کارایی خوبی داشته و به جوابهای نزدیک به بهینه همگرا گردد. یکی از مشکلات الگوریتم پیشنهادی پیدا کردن مقدار مطلوب پارامتر دقت از طریق آزمایش و خطا می‌باشد. امکان تنظیم خودکار پارامتر دقت توسط روشهای هوشمند (مانند اتوماتاهای یادگیر) و همچنین امکان استفاده از دقتهای متفاوت برای اتصالهای متفاوت تحت مطالعه و بررسی می‌باشد.

۷- مراجع

[1] K. A. Buckley, S. H. Hopkins, B. C. H. Turton; "Solution of Inverse Kinematics Problems of a Highly Kinematically Redundant Manipulator using Genetic Algorithms"; *Genetic Algorithms in Engineering Systems: Innovations and Applications*, p.p. 264-269, 1997.

[2] J. J. Craig; "Introduction to Robotics: Mechanics and Control"; 2nd Edition, Addison-Wesley, 1989.

[3] J. Guo, V. Cherkassky; "A Solution to the Inverse Kinematic Problem in Robotics using Neural Network Processing"; *Proceeding of IEEE International joint Conference on Neural Networks*, vol 2, p.p. 299-304, 1989.

[4] R. Koker, C. Oz, T. Cakar, H. Ekiz; "A Study of Neural Network based Inverse Kinematics Solution for a three-joint Robot "; *Robotics and Autonomous Systems*, vol 49, Issues 3-4, p.p. 227-234, 2004.

[5] Y. Kuroe, Y. Nakai, T. Mori; " A New Neural Network Learning of Inverse Kinematics of Robot Manipulator"; *International Conference on Neural Networks*, vol V, p.p. 2715-2720, Orlando, Florida, 1994.

[6] K. S. Narendra, M. A. L. Thathachar; "Learning automata: an introduction"; Prentice-Hall, Englewood Cliffs, 1989.

[7] A. C. Nearchou; "Solving the Inverse Kinematics Problem of Redundant Robots operating in Complex Environments Via a Modified Genetic Algorithm"; 1997.

[8] J. Parker, A. Khoogar, D. Goldberg; "Inverse Kinematics of Redundant Robots using Genetic Algorithms"; *Proceeding of IEEE International Conference on Robotics and Automation*, p 271, 1989.

[9] F. Pourboghrat; "Neural Networks for Learning Inverse-Kinematics of Redundant Manipulators"; 1990.

[10] S. Tejomurtula, S. Kak; "Inverse Kinematics in Robotics using Neural Networks"; *Information Sciences*, 116, p.p. 147-164, 1990.