مجلهٔ علمی- پژوهشی
«مهندسی برق مدرس»
دورهٔ یازدهم، شمارهٔ ۱، بهار ۱۳۹۰

برنامه ریزی حرکت چند روباتی با حرکات همزمان بر روی گراف

الیپس مسیحی[۱*]، هیوا صمدیان[۲]
۱- استادیار، دانشکده فنی و مهندسی، دانشگاه تربیت مدرس
۲- دانش آموخته کارشناسی ارشد مهندسی فناوری اطلاعات، دانشگاه تربیت مدرس

masehian@modares.ac.ir
(دریافت مقاله: ۸۸)

چکیده- به موازات توسعهٔ نسل روبات های هوشمند و گسترش کاربرد آنها در صنایع مختلف، حوزه برنامه‌ریزی حرکت روبات‌ها که زیر شاخهٔ مهمی از علوم کامپیوتر و مهندسی کنترل می باشد بیش از پیش اهمیت و گسترش یافته است. در این حوزه اغلب به مسائل پرهیز از موانع یا الگوریتم های بهینه‌سازی مسیرهای حرکت روبات پرداخته می شود. در این مقاله روشی جدید برای حل مسئله برنامه‌ریزی حرکت چند روبات در شرایطی که روبات‌ها مجاز به حرکت یا توقف بر روی گره‌های یک گراف همبند و مسطح هستند ارائه شده است. همچنین، شرایط دیگری مانند ویژگی‌های گراف مسیر از لحاظ طول و ظرفیت یال‌ها و امکان حرکت همزمان روبات‌ها مد نظر قرار گرفته‌اند. چالش اساسی مسائل برنامه ریزی حرکت چند روباتی، وقوع حالات قفل‌شدگی (تمایل به اشغال یک مکان واحد در زمان واحد) می باشد که همین امر آنرا در دستهٔ مسائل سخت (NP-complete) قرار داده است. روش ارائه شده ترکیبی ابتکاری از رویکردهای برنامه ریزی حرکت متمرکز و غیرمتمرکز است و بدین جهت از ویژگی‌های مثبت هر دو رویکرد بهره‌مند است. جهت سنجش کیفیت جواب‌های تولید شده از معیارهای کمینه سازی زمان حل و تعداد حرکات لازم به طور توأم استفاده شده است. الگوریتم توسعه داده شده دارای پیچیدگی زمانی چندجمله‌ای بوده و به‌وسیله نرم‌افزار MATLAB شبیه سازی شده است. نتایج حل مثال‌های متعدد با اندازه‌ها و پیچیدگی‌های مختلف نشان دادند که تحت روش ارائه شده، تعداد کل حرکات لازم جهت رفع قفل شدگی‌ها تفاوت کمی با حداقل تعداد ممکن حرکات بدون رفع قفل شدگی ها دارد.

کلید واژگان: برنامه‌ریزی حرکت چند روباتی، قفل شدگی، نظریه گراف.

[Downloaded from mjee.modares.ac.ir on 2024-12-29]

۱- مقدمه

برای آن که روبات های نسل جدید بتوانند به طور مستقل و هوشمند عمل کنند بایستی توانایی برنامه‌ریزی جهت حرکت از یک نقطه به نقطه دیگر را، بدون برخورد با موانع و ترجیحاً در حداکثر سرعت ممکن دارا باشند، به طوری که مسیر حاصل حتی‌الامکان کوتاه و هموار بوده و امکان پیمایش آن توسط روبات ممکن باشد. مسئله *برنامه‌ریزی حرکت روبات* جهت پاسخگویی به همین نیاز مطرح شد و از اواخر دهه ۱۹۷۰ میلادی به عنوان یکی از زمینه‌های مهم در علم روباتیک مورد توجه محققان قرار گرفت [۱].

مسیر یک روبات، منحنی‌ای در فضای پیکربندی آزاد (عاری از موانع) می باشد که می تواند یا به شکل ریاضی و پارامتری و یا به شکل توالی‌ای از نقاط واقع بر آن تعریف و بیان شود. *برنامه ریزی مسیر*[1] (یا مسیریابی) روبات عبارت است از یافتن مسیری عاری از تصادم با موانع (ثابت یا متحرک) میان نقاط آغازین و پایانی روبات. *خط سیر*[2] روبات، مسیری می باشد که همراه با تخصیص زمانی است؛ یعنی مکان روبات در هر لحظه از زمان توسط یک تابع یا مجموعه مشخص میشود. *برنامه ریزی حرکت*[3] لفظی است که عموماً به برنامه ریزی خط سیر اطلاق می شود. در مسائل برنامه‌ریزی حرکت معمولاً توصیف کاملی از هندسه روبات، محیط و موانع ارائه می‌شود [۲].

هر چند الگوریتم های برنامه‌ریزی حرکت از نظر جزئیات تفاوت‌های زیادی با هم دارند اما در بیشتر آنها یک چارچوب کلی وجود دارد. در گام اول، روبات – که ممکن است دارای شکل پیچیده هندسی باشد – از لحاظ نظری به یک نقطه کاهش داده شده و فضای کاری نیز به یک فضای جدید به نام *فضای پیکربندی*[4] تبدیل می‌شود. این امر باعث می‌شود تا مسئله اصلی به یک مسئله مسیریابی برای یک نقطه تبدیل شود. *پیکربندی* یک روبات (یا جسم) مجموعه‌ای از پارامترها است که به صورت یکتا موقعیت هر نقطه بر روی روبات (یا جسم) را تعیین می کند. این موضوع برای اولین بار در برنامه‌ریزی حرکت در [۳] استفاده شد. به عنوان مثال، فضای پیکربندی یک *روبات سیار*[5] مجموعه ای شامل مختصات نقطهٔ مرجع و جهت قرارگیری آن (x, y, θ) است.

مسئله برنامه‌ریزی حرکت روبات از نظر پیچیدگی زمانی و فضایی، NP-complete و PSPACE-hard بوده و با افزایش تعداد روبات‌ها به صورت نمایی رشد می کند [٤].

بسته به نوع به دست آوردن اطلاعات محیطی، روشهای برنامه‌ریزی حرکت روبات به دو گروه کلی غیربهنگام[6] و بهنگام[7] تقسیم‌بندی می‌شوند: در الگوریتم های غیر بهنگام، دانش کلی راجع به محیط در ابتدای کار و پیش از برنامه ریزی در دست است، در حالی که روش‌های بهنگام چنین اطلاعاتی را از پیش نداشته و آن را به طور جزئی و در حین اجرا از طریق حسگرها به دست می آورند [۵].

اکثر روش‌های غیر بهنگام ملهم از چند شیوه کلاسیک مانند نقشه مسیر (اسکلت یابی)، میدان‌های پتانسیل و تجزیه سلولی می باشند [۱]. در شیوه نقشه مسیر، فضای

1. Path planning
2.Trajectory
3. Motion planning

4. Configuration space
5. Mobile robot
6. Offline
7..Online

[Downloaded from mjee.modares.ac.ir on 2024-12-29]

پیکربندی، یعنی مجموعه حرکات قابل تصور، کاهش یافته و به صورت خطوط یک بعدی (گراف) رسم میشود. نقشه‌های مسیر مشهور عبارتند از *دیدنگار*[1] و *نمودار ورونویی*[2]. دیدنگار مجموعه خطوط در فضای آزاد است که هر رأس از موانع را به رئوس قابل رؤیت دیگر متصل می‌کند، و کوتاه‌ترین مسیر بین شروع و هدف از این نمودار و با استفاده از الگوریتم *A محاسبه می‌شود. نمودار ورونویی (یا محور میانی) عبارت است از مکان هندسی نقاطی که در فواصل یکسانی از حداقل دو جسم اطراف قرار دارند. در نتیجه، این نمودار امن‌ترین مسیر (دورترین تا موانع) را ایجاد می کند، و فضا را به نواحی-ای افراز می‌کند به طوری که کلیه نقاط در هر ناحیه نزدیک‌ترین فاصله را تا جسم درون آن ناحیه دارند (شکل ۱(الف)). با وصل نقاط شروع و هدف به نمودار ورونویی، یک مسیر موجه بین آنها قابل یافتن است. مزیت نمودار ورونویی این است که همبند بودن فضای اولیه مستقیماً به نمودار منتقل می‌شود [٦].

۱-۲- برنامه ریزی حرکت چند روباتی

در برنامه ریزی حرکت سیستم های چند روباتی، فضای زمان-پیکربندی[3] حرکت روبات‌ها مورد توجه قرار می گیرد. تعریف عمومی مسئله چنین است: "مجموعه ای از روبات‌های همگن یا ناهمگن $r_1, r_2, \ldots, r_n$ با حداکثر سرعت‌های $v_1, v_2, \ldots, v_n$ در فضای کاری W حرکت می-کنند. تابع هدف مسئله عبارت است از کمینه سازی مجموع اندازه مسیرهایی که هر یک از روبات‌ها از نقطه شروع تا پایان حرکت خود طی می‌کنند به نحوی که هیچ نوع برخوردی با موانع و روبات‌های دیگر نداشته باشند".

این امر به معنای حداقل کردن اندازه مسیر حرکت هر یک از روباتها به صورت مجزا می‌باشد. به عبارت دیگر، برنامه‌ریزی حرکت چند روباتی، طراحی مسیرهای قابل پیمایش توسط روبات‌ها و ارائه قوانین حرکت آنها در یک محیط کاری است، تا روبات‌ها بتوانند از یک پیکربندی اولیه به یک پیکربندی نهایی برسند.

در مسائل برنامه ریزی حرکت چند روباتی، فضا (و کمبود آن) مهمترین محدودیت را اعمال می کند؛ بدین معنی که به علت وجود روبات‌های متعدد معمولاً فضای کافی و آزاد در هر لحظه برای تردد همهٔ روبات‌ها در اختیار نیست، و در نتیجه مواردی اتفاق می افتند که دو یا چند روبات در طی پیمایش مسیرشان به سمت هدف خود، بایستی از یک محل خاص در زمان واحد عبور کنند. این پدیده که *قفل‌شدگی*[4] (تعارض، اشغال یک مکان واحد در زمان واحد) نامیده می شود، بزرگترین چالش مسئله چندروباتی است.

در ادبیات دو رویکرد اصلی در رابطه با برنامه‌ریزی حرکت سیستمهای چند روباتی پیشنهاد شده است [۱]: در اولی که اصطلاحاً *برنامه‌ریزی متمرکز*[5] نامیده می‌شود کلیهٔ روبات‌ها به صورت اجزای گسسته‌ای از یک روبات واحد مرکب (با درجه آزادی برابر با مجموع درجات آزادی همهٔ روبات‌ها) در فضای پیکربندی در نظر گرفته می شوند، و لازم است که هم پیکربندی اولیه و هم پیکربندی نهایی کامل سیستم روباتهای چندتایی معلوم باشد. در این صورت برنامه ریزی حرکت روبات‌ها در یک فضای پیکربندی مرکب انجام می گیرد [۷۸]. در رویکرد متمرکز، برنامه‌ریزی در یک فاز صورت می گیرد و

1. Visibility Graph
2. Voronoi Diagram
3. Configuration-Time Space
4. Deadlock
5. Centralized planning

[Downloaded from mjee.modares.ac.ir on 2024-12-29]

تصمیم در خصوص حرکت یک روبات با در نظر گرفتن وضعیت سایر روبات‌ها است (حل *فراگیر*[1]). مزیت این روش آن است که *کامل*[2] است؛ یعنی در صورتی که راه حلی موجود باشد، یافتن آن را تضمین می‌کند. عیب این روش نیز آن است که احتیاج به جستجو در فضای با ابعاد بالا دارد و به جهت آنکه پیچیدگی زمانی مسئله رشد می کند، نامطلوب می باشد [۱].

در رویکرد دوم که روش بسیار متداول‌تری بوده و برنامه‌ریزی *غیرمتمرکز*[3] (توزیعی) نامیده می شود، برنامه-ریزی در دو مرحله انجام می گیرد: ابتدا برنامه‌ریزی انفرادی برای هر روبات به طور جداگانه و بدون در نظر گرفتن روبات‌های دیگر انجام می گیرد، و در مرحلهٔ دوم این برنامه‌ها با در نظر گرفتن موقعیت سایر روبات‌ها جهت رفع قفل‌شدگی تصحیح می شوند [۹، ۱۰].

معمولاً تعارضات میان روبات‌ها با روشهای محلی[4] *تنظیم سرعت*[5] و یا *اولویت دهی*[6] برطرف می‌شوند. به عبارت دیگر، برای تصمیم‌گیری در خصوص حرکت یک روبات در نظر گرفتن وضعیت همه روبات‌های دیگر به طور همزمان الزامی نیست. به علت ابعاد پائینتر فضای جستجو، رویکرد غیرمتمرکز هزینه محاسباتی کمتری نسبت به برنامه‌ریزی متمرکز دارد، ولی در عوض کامل نیست، یعنی ممکن است هیچ راه حلی یافت نشود در شرایطی که حداقل یک جواب وجود داشته باشد [۱].

در مورد اهمیت برنامه ریزی حرکت چند روباتی در ادبیات موضوع و اشکال خاص آن در [۱۱] مرور و بررسی جزئی به عمل آمده است. در [۱۲] از یک رویکرد احتمالی جهت ساختن نقشه مسیری از فضای آزاد استفاده شده است و مسیرهای روبات‌ها روی این شبکه برنامه ریزی و هماهنگ می شوند. در [۱۳] از یک روش غیرمتمرکز استفاده شده است که در فاز دوم آن از روش تغییر سرعت برای جلوگیری از تصادم بهره گرفته می شود. همچنین، ضمن حل یک مسئله دو روباتی به عنوان مثال، معیارهایی برای ویژگی مقاوم[7] بودن مسئله تعریف شده است. روش معرفی شده در [۱٤] نیز مبتنی بر تنظیم سرعت روبات‌ها برای پرهیز از برخورد می باشد.

الگوریتم ارائه شده در [۱۵] یک رویکرد نوعاً غیرمتمرکز می باشد که در آن هر روبات اولویت بندی می گردد و مسیرهای کلی مطابق با روبات های با ارجحیت بالاتر برنامه ریزی می کنند.

در رویکرد اولویت دهی توسعه داده شده در [۱٦] حرکت روبات‌ها به طور ترتیبی و بر مبنای چند ارجحیت محاسبه می گردند و عدم تصادم در مورد روبات‌های با اولویت پایین بررسی نمی شود. نحوه تخصیص اولویت‌ها برای موفقیت روش‌های مبتنی بر اولویت‌دهی حائز اهمیت بسیار می باشد. اولویت‌ها ممکن است ثابت و بدون تغییر فرض شوند (مانند [۱۷])، و یا از اولویت‌دهی ابتکاری (هیوریستیک) استفاده شود (مانند [۱۸]).

۱-۳- رویکرد جدید توسعه داده شده

همان گونه که اشاره شد روش های مبتنی بر نقشه مسیر شبکه یا گرافی به هم پیوسته از مسیرهای متعدد میان نقاط شروع و پایان تولید می کنند و سپس این گراف را برای یافتن کوتاهترین مسیر جستجو می کنند. گراف‌ها با تثبیت مکان هایی از فضای پیکربندی آزاد به عنوان

1. Global
2. Complete
3. Decoupled planning
4. Local
5. Velocity tuning
6. Prioritization
7. Robust

[Downloaded from mjee.modares.ac.ir on 2024-12-29]

گره‌ها، یال‌ها و سایر اجزاء توانایی خوبی در وضع و اعمال قوانین حرکت و اولویت حرکت روبات ها دارند. ایده استفاده از نظریه گراف در تحقیقات حوزه برنامه-ریزی حرکت روز به روز تقویت شده و امروزه در حال تبدیل شدن به یک جریان تحقیقاتی می باشد. برای مثال، مقالاتی مانند [۱۷، ۱۹، ۲۰، ۲۱] به مطالعه امکان پذیری و حل مسایل برنامه ریزی حرکت چندروباتی بر روی گراف‌ها پرداخته‌اند.

در تحقیق حاضر نیز با هدف بهره گیری از پتانسیل نظریه گراف در گسسته سازی فرآیند حرکت روبات‌ها و سهولت تعریف اولویت های حرکتی آنان، از گرافی برگرفته از نمودار ورونویی به عنوان ورودی اولیه و خام سیستم استفاده می شود. معهذا، با توجه به اینکه چنین گرافی اساساً برای برنامه ریزی حرکت تنها یک روبات مناسب است و اینکه گراف ورونویی (به جهت ماهیت بیشینه سازی فواصل آن از موانع) فضاهای زیادی که جهت رفع قفل شدگی‌های میان روبات‌های معارض قابل استفاده هستند را پوشش نمی‌دهد، مفهوم جدیدی به نام *گراف بهبودیافته* را مطرح نموده و با انجام عملیاتی، گراف خام ورونویی را جهت پذیرفتن و حل مسئله چند روباتی ارتقا دادیم.

همچنین در این مقاله روشی جدید از ترکیب دو رویکرد متمرکز و غیر متمرکز برنامه ریزی حرکت چند روباتی توسعه داده شده است که از ویژگیهای مثبت و مزایای هر دو رویکرد بهره می برد: از یک طرف به خاطر تفکیک مراحل برنامه ریزی مسیر و برنامه ریزی حرکت در دو فاز مجزا، روش ارائه شده یک روش غیر متمرکز محسوب می شود، که چنین تفکیکی امکان بهره‌مندی از الگوریتم های موفق و مطرح در برنامه ریزی حرکت روبات را فراهم کرده است. برای مثال، جهت مرحله تشخیص موانع و استخراج گراف مسیر (برنامه ریزی مسیر) از روش ورونویی استفاده می شود و جهت یافتن کوتاهترین مسیر روی گراف از الگوریتم دایکسترا بهره گرفته می شود. از طرف دیگر، در فاز دوم برای رفع تعارضات بین روباتها از روش متمرکز استفاده می شود که در آن ارائه برنامه هر روبات با در نظر گرفتن وضعیت سایر روبات‌ها در کل فضای حل مسئله انجام می شود.

در نتیجهٔ چنین رویکردی، هم پیچیدگی حل توسط روش جدید کمتر از حل متمرکز است (که این خود منجر به کاهش زمان حل می شود)، و هم برای تصمیم‌گیری در خصوص یک روبات وضعیت سایر روبات‌ها نیز در نظر گرفته می شود که این خود منجر به رفع تعارضات، جلوگیری از ورود روبات به مسیری که بعدها منشأ قفل شدگی با سایر روبات‌ها خواهد شد، و نیز کاهش یافتن تعداد حرکتهای غیر ضروری می شود.

۲- تعاریف اولیه

در جداول ۱ و ۲ مفاهیمی که در مقاله به کار گرفته شده‌اند تعریف شده اند. برخی از این تعاریف مفاهیم موجود و رایج در ادبیات موضوع هستند که جهت سهولت مراجعه عیناً آورده شده‌اند، و برخی دیگر نیز تعاریفی هستند که در گذشته به کار رفته‌اند ولی در این مقاله به شکل دیگری بازتعریف شده‌اند (مانند تعریف قفل‌شدگی). دسته دیگری از تعاریف نیز هستند که کاملاً جدید بوده و جزء روش حل جدید می‌باشند (مانند تعریف ماتریس مسیر).

[Downloaded from mjee.modares.ac.ir on 2024-12-29]

جدول ۱ تعاریف مربوط به گراف

اصطلاح/ نشانه	توضیحات
پیکربندی	آرایشی از روبات‌ها روی گره‌های گراف به طوری که هیچ گرهی دارای بیش از یک روبات نباشد
n یا $\|G\|$	تعداد رئوس گراف $G=(V,E)$
$deg(v)$	درجه گره یا تعداد یال‌های متصل به یک گره
دو رأس مجاور	دو گرهی که حداقل با یک یال بدون واسطه به هم متصل هستند
یال‌های موازی	یال‌هائی که دو گره مجاور را به هم متصل می‌کنند
گراف اولیه	گرافی که از اجرای روش ورونوئی حاصل شده است. این گراف با ابعاد $G_{n1\times n1}$ نمایش داده میشود و مقدار درایه (i,j) آن صفر است اگر گره های i و j مجاور نباشند، و مقدار آن k است اگر با k عدد یال به هم متصل باشند
گره میانی	برای هر گره دلخواه در گراف اولیه، با حرکت روی یال‌ها به اندازه یک واحد سرعت یک گره میانی تعریف می شود
گراف بهبود یافته	گرافی است که با انجام عملیاتی روی گراف اولیه حاصل می شود و آنرا تا حد ممکن بزرگ می کند یعنی حداکثر گره‌ها و یال‌ها و دورها را دارد، و نیز یال‌ها و گره‌های اضافی را حذف می کند و طول یال‌ها در آن به اندازه حداقل طولی که به صورت خط مستقیم می تواند طی شود (در یک واحد سرعت) تبدیل شده است. این گراف با ابعاد $G_{n2\times n2}$ نمایش داده می شود و مقدار درایه (i,j) آن صفر است اگر گره های i و j مجاور نباشند، و مقدار آن ۱ است اگر به هم متصل باشند

جدول ۲ تعاریف مربوط به روش حل

اصطلاح/ نشانه	توضیحات
$\mathrm{PL}^0(r_i)$	*برنامه اولیه روبات* r_i برای هر روبات r_i کوتاهترین مسیرش از گره شروع تا گره هدف را به صورت دنباله $\mathbf{PL^0}(r_i)$ زیر در نظر می گیریم: $\langle v_1 \mid a'_1\ a'_2 \dots a'_{k_1} \mid v_2 \mid b'_1\ b'_2 \dots b'_{k_1} \mid \dots \mid v_n \rangle$ که در آن a'_i و b'_j به ترتیب گره های میانی بین گره های مجاور اولیه v_1 و v_2 و v_2 و v_3 هستند. تعداد عناصر بزرگترین دنباله ای که از این طریق حاصل می شود را Mمی نامیم، و چنانچه طول دنبالهٔ روباتی کمتر از M باشد آخرین عنصر آن دنباله را تا رسیدن به M تکرار میکنیم. در واقع M نشان دهنده تعداد مراحل برنامه است
PM	*ماتریس مسیر*: ماتریسی با ابعاد $G_{m\times M}$ است (m تعداد روبات‌هاست) که هر سطر آن شامل برنامهٔ هر روبات می‌باشد. اولین ماتریس مسیر از $\mathbf{PL^0}(r_i)$–ها تشکیل شده است. این ماتریس در طول اجرای الگوریتم مرتباً به روز آوری می شود (شکل ۵(ک))
S_j	*مرحله*: هر آرایش جدید از روبات‌ها روی گره‌های گراف که از حرکت حداقل یک روبات به اندازهٔ یک گره ایجاد شود یک مرحله جدید نامیده می شود. در واقع ستون j–ام ماتریس **PM** نشان دهنده مرحله j–ام بوده و با S_j نشان داده می شود
CH	*کانال*: عبارت است از یک دنباله از گره‌های درجه دو که به طور مشترک در مسیر دو یا چند روبات قرار دارد
قفل شدگی	هرگاه در یک برنامه دو روبات r_i و r_j وجود داشته باشند که در یک مرحله مانند S_k یکی از دو حالت زیر پیش آید آن برنامه دارای قفل‌شدگی است: $\mathrm{DL1}: \ a_{i,k}=a_{j,k+1} \ \wedge \ a_{j,k}=a_{i,k+1}$ $\mathrm{DL2}: \ a_{i,k}=a_{j,k}$

[Downloaded from mjee.modares.ac.ir on 2024-12-29]

۲-۱- گراف اولیه

از آنجا که گراف اولیه از اجرای الگوریتم ورونویی حاصل می شود، یال‌های آن از حداقل دو مانع به یک فاصله بوده و مسیرها دقیقاً در وسط فضای آزاد میان موانع قرار دارند (شکل ۱(الف)). از طرفی در به دست آوردن این گراف اندازه روبات‌ها در نظر گرفته نشده‌اند، و لذا ممکن است یالی وجود داشته باشد که به خاطر قرارگیری در یک گذرگاه باریک اجازه عبور به روباتی را ندهد. نکتهٔ دیگر آنست که برای استفاده بهینه از فضای کاری در جاهایی که فاصله دو مانع از هم زیاد است (به گونه ای که چند روبات همزمان می توانند از کنار هم عبور کنند) بایستی یال‌های دیگری (به صورت موازی) داشته باشیم که این گراف اولیه فاقد آن است.

دو موضوع فوق نیاز به تکمیل و بهبود گراف اولیه را در راستای رفع این دو اشکال ایجاب می کند. گراف‌های اولیه می توانند دارای یال‌های موازی بین دو گره مجاور باشند. در واقع چنین گرافی خروجی مرحله برنامه ریزی مسیر است که در آن با حذف موانع از فضای کاری به گراف اولیه دست یافته ایم.

۲-۲- گراف بهبود یافته

گراف بهبود یافته از تغییر گراف اولیه حاصل می شود به طوری که یال‌های عبوری از میان گذرگاه های تنگ موانع به دلیل بلا استفاده بودن حذف شده و در نتیجه سبب کاهش حجم پردازش می شود. در عین حال، از حداکثر فضای ممکن (به ویژه اطراف موانع) استفاده شده و تا حدی که با توجه به قطر روبات‌ها به آنها امکان جابجایی بدهد گره ها، یال‌ها و دورها به گراف اولیه افزوده می شوند. همچنین با قرار دادن چند یال موازی به جای یک یال با ظرفیت بیش از یک روبات، امکان حرکت چند روبات در امتداد یک یال (و به عبارت دیگر امکان داشتن ظرفیت بیش از یک روبات برای هر یال اولیه) لحاظ می شود (شکل ۱(ب)).

با تجزیه یک یال به مجموعه‌ای از قطعه یال‌های کوچکتر که همگی خطوط مستقیم هستند اولاً روبات‌ها یال‌های منحنی شکل را با دقت بالاتر و خطای کمتری می‌پیمایند، و ثانیاً امکان ایجاد توقف روی یک یال در بین راه به کمک گره‌های میانی فراهم می شود. ثالثاً، به گرافی می رسیم که یال‌های موازی و حلقه (لوپ) بلافصل (که در نمودار ورونویی اتفاق می افتند) نداشته و با یک ماتریس با عناصر ۰ و ۱ قابل نمایش است.

۳- رویکرد حل

مراحل کلی الگوریتم برنامه ریزی حرکت چند روباتی بدین صورت می باشد:

(۱) ابتدا فضای کاری و موقعیت موانع، تعداد و اندازه روبات‌ها و پیکربندی‌های اولیه و نهائی آنها به عنوان ورودی دریافت می شود.

(۲) توسط الگوریتم ورونویی گرافی در فضای آزاد و عاری از موانع رسم می شود (گراف اولیه).

(۳) گراف اولیه طی عملیات اضافه نمودن یال‌های موازی و گره‌ها و نیز هرس کردن یال‌های بلا استفاده به گراف بهبود یافته تبدیل می شود.

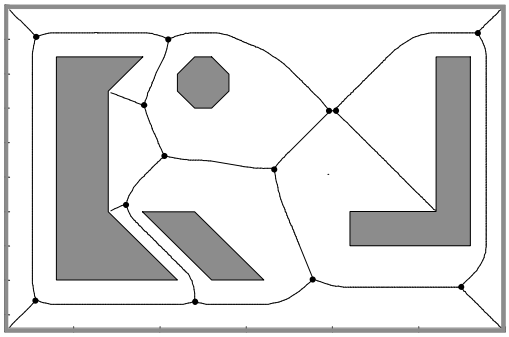
(۵) الگوریتم برنامه ریزی حرکت چند روباتی روی گراف[1] (MMPG) اجرا می شود. در این مرحله، زیرمسیرهایی (توالی گره‌های مرتبط) که مورد معارضه دو یا چند روبات هستند شناسایی می شوند، که آنان را *کانال*

1. Multi-robot Motion Planning on Graphs

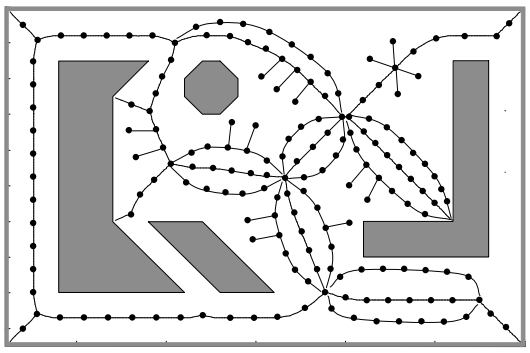
[Downloaded from mjee.modares.ac.ir on 2024-12-29]

یا گذرگاه می نامیم. سپس قفل شدگی‌ها شناسایی شده و در انتها برنامه ریزی برای رفع آنها انجام می گیرد.

با توجه به تعاریف ارائه شده در جداول ۱ و ۲، چنانچه همهٔ قفل‌شدگی‌ها در ماتریس مسیر رفع شوند به حل مسأله برنامه ریزی حرکت چند روباتی دست یافته ایم، یعنی به برنامه‌ای (ماتریسی) رسیده‌ایم که مشخص می کند هر روبات در هر لحظه در کجای فضای حل قرار دارد و نهایتاً چه مسیری را باید طی کند تا بدون تصادم با موانع و سایر روبات‌ها به هدفش برسد.



(الف)



(ب)

شکل ۱ (الف) گراف ورونویی اولیه؛ (ب) گراف بهبود یافته

(٤) برای هر روبات، گراف بهبود یافته بدون در نظر گرفتن وضعیت سایر روبات‌ها جستجو شده و کوتاهترین مسیر هر روبات محاسبه می شود.

۳-۱- معیارهای بهینگی

همانند بسیاری از مسائل بهینه سازی ریاضی، معیارهای موجود برای تعیین کیفیت یک حل (برنامه حرکت) را می‌توان دو معیار (الف) کمینه‌سازی زمان حل (پردازش)، و (ب) کمینه‌سازی تعداد حرکات کل روبات-ها در نظر گرفت.

معیارهای دیگری مثل طول کمتر مسیر پیموده شده و تعداد کمتر مراحل حل معادل معیار زمان حل کمتر هستند. جالب توجه است که کاهش تعداد حرکات همواره باعث کاهش زمان حل نمی شود (و بالعکس)؛ و به همین دلیل این دو معیار جزء دو دسته مجزا قرار می گیرند. ما در این مقاله کاهش تعداد مراحل حل را به عنوان معیار اصلی بهینگی در نظر گرفته ایم که جزء دسته معیارهای کمینه‌سازی زمان حل است.

برای رسیدن به کمترین تعداد مرحله، هنگام رفع قفل شدگی بین دو روباتی که هر دو اولویت حرکت دارند، روباتی بایستی از مسیر اصلی اش منحرف شود که طول کوتاهترین مسیرش کوتاهتر است. این در نهایت به کاهش تعداد مراحل در کل مسأله منجر می شود (با توجه به همزمانی حرکت روبات‌ها، کاهش تعداد مراحل معادل است با کم کردن تعداد مراحل یا گام‌های روباتی که دارای طویل ترین کوتاهترین مسیر است). در انجام این انحرافات، کم کردن تعداد حرکت‌های اضافی (جهت رفع تعارض) نیز لحاظ شده است.

جلوگیری از ورود روبات به کانال در مواقع لازم و متوقف کردن آن سبب کاهش چشمگیر تعداد حرکت‌ها شده و از وقوع قفل‌شدگی‌های احتمالی در آینده پیشگیری

[Downloaded from mjee.modares.ac.ir on 2024-12-29]

می کند. همچنین، تلفیق دو معیار تعداد حرکات کمتر و زمان حل کمتر یکی از ویژگی‌های روش حاضر است.

۳-۲- اولویت دهی

به زبان ساده، اولویت دهی یعنی انتخاب یک روبات از بین چند روبات برای حرکت و انتخاب یک مقصد محلی برای روباتی که حرکت آن قطعی شده است از بین چند گره در دسترس. بدیهی است که این انتخاب ها بر اساس معیارهای مختلفی انجام می‌گیرند. معیارهای تعیین اولویت (هم برای انتخاب روبات و هم برای انتخاب گره) متأثر از دو عامل است: (الف) مفروضات مسئله؛ و (ب) معیار بهینگی.

در مسائلی که حرکت روبات‌ها صرفاً به صورت نوبتی انجام می شود (غیر همزمان) اولویت بندی معادل نوبت دهی است. زمانی که گراف مسیر یک مالتی گراف است (یعنی یال‌های موازی دارد) تعیین اولویت بین چند مسیر واصل دو گره اهمیت خواهد داشت در غیر این صورت چنین معیار اولویت دهی وجود نخواهد داشت (در این تحقیق حرکت همزمان روبات‌ها مجاز فرض شده است).

با توجه به اینکه کاهش تعداد مراحل حل و در نتیجه کم کردن زمان حل به عنوان معیار اصلی بهینگی تعیین شده است معیارهای تعیین اولویت متناسب با این معیار بهینگی خواهند بود. معیارهای دیگر جهت اولویت دهی عبارتند از: معیارهای امکان پذیری (شامل بررسی وجود گره های خالی مجاور گره‌های مورد تعارض و تعداد آنها، و بررسی گره‌هایی که ورود به آنها قفل‌شدگی جدید ایجاد می کند)، معیارهای کانال (شامل بررسی سر راه بودن مقصد یک روبات، گره‌های پر ترافیک (این معیار بر اساس دادن یک کد اولیه برابر با حاصل تقسیم درجه هر گره بر دفعات حضور آن در مسیر روبات‌های دیگر مشخص می شود)، و معیار بهینگی طویل‌ترین مسیر (این معیار با انجام انحراف یا توقف در مسیرهای کوتاهتر اعمال می شود).

٤- کانال‌ها

کانال‌ها (CH) زیرمسیرهایی (توالی گره‌های مرتبط) از برنامه حرکتی روبات‌ها هستند که مورد معارضه یک یا چند روبات دیگر قرار گرفته اند. کانال ها بسته به عوامل زیر تنوع و گوناگونی دارند:

هم جهت بودن یا نبودن جهت حرکت روبات های درگیر در کانال (۲ حالت ممکن است)،

وجود مقصد یک روبات بر سر راه دیگری، یا وجود مقصد هر دو بر سر راه همدیگر، یا هیچکدام (۳ حالت ممکن است)،

وجود مبداء یکی یا هر دو یا هیچکدام از روبات‌ها در کانال (۳ حالت ممکن است).

با در نظر گرفتن تمامی این حالات، در کل بایستی ۲×۳×۳ = ۱۸ نوع کانال وجود داشته باشد، ولی چون بعضی از این حالت‌ها وضعیت‌های تکراری ایجاد می کنند و برخی از حالت‌ها امکان وقوع ندارند و برخی هم کانال های بدون تأثیر ایجاد می کنند، تنها ۹ حالت متمایز از کانال‌ها می توانند در برنامه ریزی حرکت مشکل ساز باشند، که در شکل ۲ دسته بندی شده‌اند.

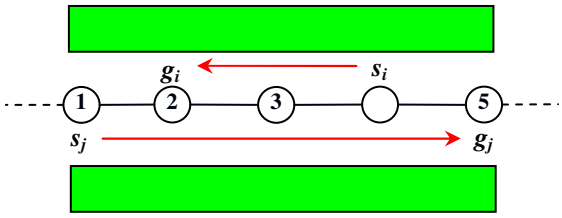
با اعمال قوائد منطقی مبتنی بر اصولی مانند شرایط حرکت روی مسیرهای آزاد، چگونگی انحراف از کوتاهترین مسیر به نفع طرف معارض (رقیب)، و تخلیهٔ مسیرهایی که پر تردد هستند، برای هر کدام از انواع کانال‌ها راه حل‌های زیر بایستی به کار گرفته شوند:

[Downloaded from mjee.modares.ac.ir on 2024-12-29]

CH1: توقف روبات خاطی (یعنی روباتی که مقصدش سر راه دیگری است) قبل از گره بحرانی (گره ابتدای کانال) تا ورود روبات دیگر به این گره. در این صورت یک اولویت کانالی ایجاد می شود..

CH2: توقف روبات خاطی قبل از گره بحرانی (گره ابتدای کانال) تا ورود روبات دیگر به این گره (یعنی خروجش از کانال). در این صورت یک اولویت کانالی ایجاد میشود.

(الف)



(ب)

شکل ۲ (الف) انواع کانال؛ (ب) شماتیکی از کانال CH6

CH3: توقف یکی از دو روبات بر اساس معیار کُد گره‌های مجاور مقصدها در گره بحرانی تا حضور گره دیگر در مقصدش و سپس حرکت آن و رفع DL1 ایجاد شده با انحراف (رجوع به جدول ۲).

CH4: توقف یکی از دو روبات بر اساس معیار کوتاهترین مسیر قبل از گره بحرانی تا رسیدن روبات دیگر به گره بحرانی (یعنی خروج روبات دیگر از کانال) و سپس وارد شدنش به کانال.

CH5: توقف روبات خاطی در نقطه مبداء یا مقصدش بر اساس معیار کُد گره‌های مجاور این دو گره سپس رفع DL1 ایجاد شده با رسیدن گره دیگر به کمک انحراف.

CH6: توقف روبات خاطی در نقطه مبداء یا مقصدش بر اساس معیار کُد گره‌های مجاور این دو گره و سپس رفع DL1 ایجاد شده با رسیدن گره دیگر به کمک انحراف.

CH7: توقف یکی از دو روبات در مبداء بر اساس معیار کُد گره‌های مجاورش تا رسیدن دیگری و سپس رفع DL1 ایجاد شده با رسیدن گره دیگر به کمک انحراف.

CH8: توقف روبات غیر خاطی (روبات خاطی در این حالت، روباتی است که وارد کانال شده است) تا رسیدن روبات خاطی به گره بحرانی (یعنی خروج روبات دیگر از کانال).

CH9: توقف یکی از دو روبات در مبداء بر اساس معیار کُد گرههای مجاورش تا رسیدن دیگری و سپس رفع DL1 ایجاد شده با رسیدن گره دیگر به کمک انحراف.

در الگوریتم توسعه داده شده پس از تشکیل ماتریس مسیرهای اولیه، کانال‌ها شناسایی می شوند. این کار با مقایسه سطرهای ماتریس صورت می گیرد: چنانچه اولویت‌های کانالی در این ماتریس رعایت شده باشند،

[Downloaded from mjee.modares.ac.ir on 2024-12-29]

تغییری صورت نمی گیرد اما چنانچه این اولویت‌ها در این ماتریس رعایت نشده باشد، قبل از ورود به مرحله قفل شدگی ها ماتریس باید بر اساس این اولویت‌ها اصلاح شود و با ماتریس اصلاح شده ای که در آن احتمال ورود به هر نوع کانالی منتفی است وارد مرحله قفل شدگی ها شویم. نکته حائز اهمیت دیگر آن است که در طول اصلاح ماتریس مسیرهای اولیه با اعمال اولویت های کانالی ممکن است برای اعمال اولویت در یک سطر، اولویت کانالی سطری دیگر به هم بخورد. برای جلوگیری از این اشتباه، ترتیب خاصی از اعمال اولویت‌ها بر اساس نوع کانال‌ها وجود دارد که رعایت کردن آن الزامی است.

۴-۱- اقدامات اصلاحی کانال

پس از شناسائی کانال‌ها و پیش از اعمال اولویت‌های کانالی تشخیص داده شده بر اساس نوع کانال‌های موجود، لازم است برخی اقدامات اصلاحی در مورد کانال‌ها اجرا شود. این اقدامات عبارتند از حذف کانال‌های زیر مجموعه کانال‌های طولانی دیگر، حذف کانال‌های کوتاه (با طول ۳ گره یا کمتر) و هرس کردن.

در این بین، هرس کردن علاوه بر اینکه ممکن است باعث کاهش تعداد کانال‌ها و در کل باعث کاهش زمان حل و جلوگیری از توقف‌های طولانی شود، در بحث رفع DL2 نیز کمک شایانی می کند. یک دنباله از گره‌های مشترک بین دو روبات، یا به عنوان کانال شناخته می شود که در این صورت روبات‌ها با رعایت اولویت کانالی واردش می شوند، و یا در اثر هرس کردن جزء کانال‌ها به حساب نخواهد آمد که در این صورت حتماً در طول مسیر مشترک این دو روبات گره های با درجه بیش از ۲ به اندازه کافی و در فاصله های حداکثر سه گره وجود داشته اند. این مسئله باعث می شود تا در صورت ورود دو روبات به یک کانال بدون رعایت اولویت کانالی، برنامه رفع DL2 بتواند به جز در موارد خاص، با حداکثر دو لایه جستجو، گرهی برای انحراف پیدا کند.

۴-۲- تفاوت برنامه و مسیر روبات ها

برای نمایش یک *مسیر*[1] ناگزیریم مشخص کنیم از کدام یک از یال‌های بین دو گره برای حرکت از یکی و رسیدن به دیگری استفاده می شود، در صورتیکه *برنامه*[2] فقط در مورد گرافی قابل طرح است که یال موازی نداشته باشد، مانند گرافی با ویژگی‌های گراف بهبود یافته. در یک برنامه بیان یال واسط دو گره مجاور ضروری نیست چراکه فقط یک یال وجود دارد. به عبارت دیگر، برنامه دنباله ای است که امکان تکرار عناصر آن وجود دارد، اما مسیر طبق تعریف گرهٔ تکراری ندارد (در نظریه گراف هر مسیر یک *گشت*[3] می باشد). در برنامه، دو عنصر متوالی می توانند یکسان باشند و معنای آن این خواهد بود که حرکت روبات روی این عنصر (گره) متوقف شده است.

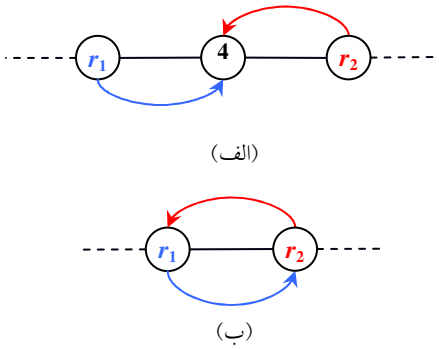
۵- قفل‌شدگی ها

طبق تعریفی که قبلاً ارائه شد قفل شدگی را به دو نوع تقسیم بندی کردیم: نوع اول با نام DL1 قفل شدگی‌ای است که در آن دو روبات یک گره را به اشتراک می گذارند، و نوع دوم با نام DL2 قفل شدگی‌ای است که در آن دو روبات یک یال را به اشتراک می گذارند. به عبارت دیگر در DL1 در یک مرحله S_k دو روبات همزمان می خواهند وارد یک گره شوند، و در DL2 در

1. Path
2. Plan
3. Walk

[Downloaded from mjee.modares.ac.ir on 2024-12-29]

یک مرحله S_k دو روبات همزمان می خواهند از یک یال به سمت هم عبور کنند (جدول ۲ و شکل ۳).



شکل ۳ قفل شدگی (الف) نوع اول DL1؛ (ب) نوع دوم DL2

حالتهای مختلف برخورد دو روبات، همگی، یکی از این دو حالت است. راه حل کلی برای رفع DL1، *توقف* است و راه حل کلی برای حل DL2، *انحراف* است. انحراف تا رسیدن به یک دور یا سه شاخه باید ادامه پیدا کند چرا که رفع قفل شدگی از راه انحراف لزوماً با در اختیار داشتن یکی از این دو محقق می شود.

این دو نوع قفل شدگی قابل تبدیل به یکدیگر هستند: همه DL2–ها با توقف یکی از دو روبات به DL1 تبدیل می شود، و بنابراین DL2–ها با توقف حل نمی شوند. همچنین، به عنوان نمونه،DL1–ی که در آن دو روبات معارض پس از عبور از گره مشترک به محل استقرار روبات دیگر بروند بعد از توقف یکی از آنها به DL2 تبدیل می شود. آنچه که به DL2 تبدیل شود و یا از ابتدا DL2 باشد به ناچار از راه انحراف باید برطرف شود، اما DL1 را علاوه بر توقف می توان از راه انحراف نیز حل کرد. با اینحال به دلیل پرهیز از پیمودن مسیرهای اضافی، برای این دسته از قفل شدگی ها حتی المقدور راه حل توقف اجرا می شود. راه حل توقف یا منجر به حل می شود و یا قفل شدگی را به DL2 تبدیل می کند که در این صورت راه حل انحراف به کار خواهد رفت.

DL1 را می توان به لحاظ رفتاری که از خود نشان می دهد به چند نوع دسته بندی کرد:

یکی از روباتها از قبل در گره مشترک باشد و بخواهد در مرحله بعد در آن گره بماند (با توجه به اینکه در ماتریس مسیر اولیه هر سطر یک مسیر بدون دور می باشد، و رفع قفل شدگی خاصیت مسیر بودن سطر را در هر مرحله و مراحل بعدیش بر هم نمی زند، می توان شرط ماندن یا نماندن روبات در مرحله بعد را معادل این دانست که آیا مقصد نهائی روبات در گره مورد تداخل هست یا خیر.)

یکی از روبات‌ها از قبل در گره مشترک باشد و بخواهد در مرحله بعد از آن گره برود.

یکی از روبات‌ها از قبل در گره مشترک نباشد و بخواهد در مرحله بعد در آن گره بماند.

۱. هر دو روبات از قبل در گره مشترک نباشند و بخواهند در مرحله بعد از آن گره بروند.

در نوع اول، برای رفع، حتماً همین روبات باید منحرف شود.

در نوع دوم، برای رفع، ابتدا باید بررسی شود که در صورت برگشت این روبات به موقعیت قبلی آیا در مراحل پیشین قفل شدگی ایجاد می شود یا خیر؟ در صورتیکه ایجاد نمی شود، راه حل بهتر آنست که این روبات به موقعیت قبلی اش منحرف شود و روبات دیگر وارد گره مشترک شود. اما در صورت ایجاد قفل شدگی در مراحل قبل، راه حل توقف روبات دیگر است.

در نوع سوم، برای رفع، همین روبات باید متوقف شود.

[Downloaded from mjee.modares.ac.ir on 2024-12-29]

در نوع چهارم، برای رفع، یکی از روباتها باید بر اساس سایر اولویت‌ها متوقف شود.

چهار حالت فوق پس از بررسی کلیه ۱۶ حالت ممکن که ترکیبات مختلف همین چهار حالت پایه‌ای هستند به دست آمده‌اند (البته وجود بعضی ترکیبات امکانپذیر نیست). راه حل‌های فوق نیز پس از بررسی تمامی ترکیبات مختلف این حالتها به دست آمده اند، که در صورت ترکیب دو حالت مختلف با هم سازگاری دارند.

با توجه به قضایائی که در ادامه ثابت خواهند شد رفع DL1 در یک ستون ماتریس PM باعث ایجاد قفل شدگی در ستون قبلی نمی شود. همچنین رفع DL2 در یک ستون باعث ایجاد قفل شدگی در همان ستون و ستونهای قبل از آن نمی شود. بدیهی است که در ستون اول، DL1 وجود ندارد. بنابراین می توان نتیجه گرفت که با شروع از ستون اول و رفع DL2 در آن و سپس به ترتیب رفع DL1 و بعد DL2 در ستونهای بعدی می توان قفل شدگی‌ها را به جلو راند. به این ترتیب پس از آنکه آخرین قفل شدگی‌ها در ستون آخر رفع شد به جواب کلی مسأله رسیده ایم. البته چنانچه در طول حل به حالت‌های رفع نشدنی برخورد کنیم الگوریتم، از برنامه خارج می شود و مسأله به عنوان مسأله غیر قابل حل اعلام می شود. همین موضوع دلیل کامل بودن الگوریتم توسعه داده شده است.

۵-۱- قضایای مربوط به قفل‌شدگی ها

قضیه ۱: رفع DL1 در یک مرحله باعث ایجاد DL2 در مراحل قبلی نمی شود.

برهان: *حالت (۱): اگر DL1 بتواند از راه توقف حل شود:*

فرض کنیم روباتهای r_i و r_j در مرحله k یک گره را به اشتراک بگذارند، در این صورت $a_{i,k} = a_{j,k}$. پس از توقف یکی از آنها (مثلاً r_i) تنها تغییری که خواهیم داشت عبارت است از $a_{i,k} = a_{i,k-1}$. به این ترتیب تمامی $a_{j,k}$ ها به جز $j = i$ بدون تغییر می مانند.

برای شناسائی DL2 در هر ستون، هر مؤلفه آن ستون مثل $a_{i,k-1}$ با مؤلفه های ستون بعدی برای سایر روباتها (تمامی $a_{j,k}$ ها به جز $j = i$) مقایسه می شوند. از آنجا که ابتدا DL2 در ستون $k-1$ برطرف شده سپس به ستون k برای رفع DL1 رسیده ایم و برای رفع DL1 در ستون k تمامی $a_{j,k}$ ها به جز $j = i$ بدون تغییر مانده اند، برای کنترل DL2 در ستون $k-1$ کافیست تمامی $a_{i,k-1}$ ها به جز $j = i$ با $a_{i,k}$ مقایسه شوند. سپس در صورت مشاهده یک تساوی مانند $a_{i,k-1} = a_{i,k}$، تساوی $a_{i,k-1} = a_{1,k}$ بررسی میشود و چنانچه این تساوی برقرار باشد DL2 وجود دارد. اما تساوی $a_{1,k-1} = a_{i,k}$ هرگز درست نیست چون در این صورت خواهیم داشت:

$$\left.\begin{aligned} a_{i,k-1} &= a_{i,k} \\ a_{1,k-1} &= a_{i,k} \end{aligned}\right\} \Rightarrow a_{i,k-1} = a_{1,k-1}$$

با توجه به اینکه DL1 در ستون $k-1$ (بعد از رفع DL2 در ستون $k-1$ و قبل از وارد شدن به رفع قفل شدگی در ستون k) رفع شده است، این تساوی امکان پذیر نیست. از آنجا که توقف عبارت است از تبدیل $a_{i,k}$ به $a_{i,k-1}$، لذا توقف تغییری در ستونهای $j-2$ و قبل از آن ایجاد نمی کند.

حالت (۲): اگر DL1 بتواند از راه انحراف حل شود:

این حالت نوع اول از چهار حالت ممکن برای DL1 بوده و مربوط به زمانی است که یکی از روبات ها از قبل در گره مورد تداخل قرار گرفته و در مرحله بعد (و تا انتها) قرار است در این گره بماند. در این حالت این گره باید منحرف شود.

[Downloaded from mjee.modares.ac.ir on 2024-12-29]

فرض کنیم گره مورد تداخل $a_{i,j}$ باشد. برای انحراف، گره‌های خالی مجاور $a_{i,j}$ بررسی شده و هر گرهی که در ستون $j-1$ نیز پر بوده است از مجموعه گره‌های ممکن برای انحراف کنار گذاشته می شود. فرض وجود DL2 در ستون $j-1$ پس از انحراف $a_{i,j}$ به گرهی مانند $a_{t,j}$، به این معناست که گرهی مثل $a_{1,j-1}$ وجود دارد که $a_{1,j-1} = a_{t,j}$ و این با کنار گذاشتن گره‌های ستون $j-1$ از مجموعه گره-های ممکن برای انحراف در تناقض است. در صورت پر بودن همه گره‌های مجاور $a_{i,j}$ و استفاده از لایه بعدی گره-های مجاور نیز همین استدلال برای انحراف از ستون $j-1$ و عدم برخورد با DL2 در ستون $j-2$ تکرار می شود.

استدلال دیگر برای قسمت دوم قضیه:

همانطور که در تعاریف مربوط به قفل‌شدگی آمده است این نوع از DL1 رفتاری مشابه DL2 دارد و رفع آن نیز معادل رفع یک DL2 در ستون قبلی است و در رفع DL2 عدم ایجاد DL2 در همان ستون بررسی میشود. پس امکان ایجاد DL2 در ستون قبلی در این حالت هم وجود ندارد. درصورت پر بودن همه گره‌های مجاور گره مورد تداخل و استفاده از لایه بعدی گره‌های مجاور نیز همین استدلال برای انحراف از ستون قبلی و عدم برخورد با DL2 در ستون قبل از آن تکرار می شود.

نتیجه ۱: با توجه به اینکه بدیهی است رفع DL1 در یک ستون باعث ایجاد DL1 در ستونهای قبلی نمی شود می توان نتیجه گرفت که رفع DL1 در یک ستون کلاً باعث ایجاد قفل شدگی در ستون‌های قبلی نمی شود.

قضیه ۲: رفع DL2 در یک مرحله باعث ایجاد قفل شدگی در آن مرحله نمی شود.

برهان: در رفع DL2 قبل از انتخاب گرهی که باید به آن منحرف شد بررسی می شود که ورود به این گره باعث ایجاد DL2 دیگری در همین ستون نشود. از طرفی در رفع DL2 در یک ستون، تغییرات ایجاد شده در ستون بعدی ظاهر خواهد شد لذا در ستون حاضر که DL1 ندارد تغییری نداریم. لذا پس از رفع DL2 در همین ستون DL1 نیز نخواهیم داشت.

نتیجه ۲: با توجه به اینکه رفع DL2 در یک ستون باعث ایجاد قفل‌شدگی در ستون‌های قبلی نمی‌شود می‌توان نتیجه گرفت که رفع DL2 در یک ستون باعث ایجاد قفل‌شدگی در آن ستون و ستون‌های قبلی نمی شود.

۵-۲- حالت‌های رفع نشدنی قفل‌شدگی ها

در دو حالت قفل‌شدگی های رفع نشدنی وجود دارد: یکی حالتی است که در یک DL1 روباتی که باید منحرف شود هیچ گرهی (حتی در بین گره‌های همسایه گره‌های مجاورش) برای انحراف پیدا نمی کند. حالت دیگر زمانی است که در یک DL2 روباتی که باید منحرف شود هیچ گرهی (حتی در بین گره‌های همسایه گره های مجاورش) برای انحراف پیدا نمی کند. در واقع در حالتی که قفل شدگی بایستی از راه انحراف حل شود ولی گرهی برای انحراف در کل گراف (تا عمق D که در ابتدای کار مشخص می کنیم) وجود نداشته باشد، اجرای برنامه خاتمه یافته و مسأله به عنوان مسأله غیر قابل حل اعلام می شود. این دو حالت معادل وضعیتی است که مثلاً دو روبات در یک کانال یا دور که هیچ شاخهٔ متصل به آن وجود ندارد قرار گیرند و بخواهند جایشان را با هم تعویض کنند.

تعبیر دیگر از حالت رفع نشدنی این است که بگوئیم زمانی که دور یا سه شاخه‌ای برای رفع یک قفل‌شدگی موجود نباشد حالت رفع نشدنی داریم. لازم به ذکر است که در صورت افزایش عمق D مسئله ممکن است حل داشته باشد؛ بنابراین می توان D را به

[Downloaded from mjee.modares.ac.ir on 2024-12-29]

تدریج (تا هنگامی که منابع زمانی و محاسباتی اجازه دهد) افزایش داد تا مسئله حل شود. در صورتی که با افزایش عمق جستجو به اندازهٔ کل گراف، راه حلی پیدا نشود، مسئله قطعاً جواب ندارد.

۶- شبیه سازی الگوریتم

شبه‌کد الگوریتم MMPG در شکل ٤ ارائه شده است.

Procedure MMPG

Make the initial Graph (*G*)

Assign the start and goal of robots to the nodes of graph *G*

For all robots do

Find the shortest path of each robot

End

Form the Path Matrix (PM)

Allocate codes to the nodes of graph *G* (for applying two optimality criteria)

Find the Channels (CH)

Prune the Channels

Apply Channel priorities (to avoid redundent motions during the problem solving)

Find DL2 in the first column of PM and resolve it

For column *i* of PM (*i* = 2, …., *M*) do

Find DL1 and resolve it

Find DL2 and resolve it

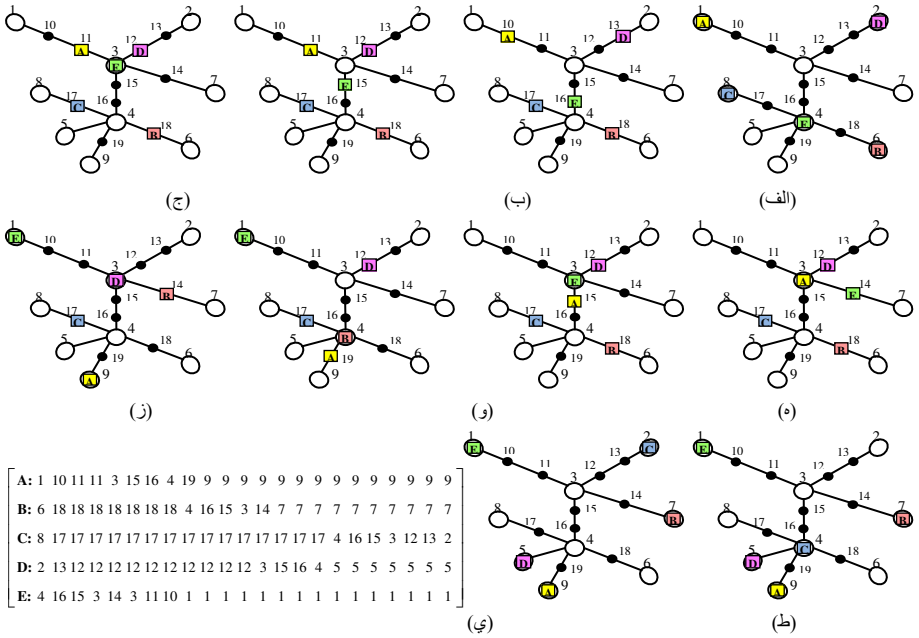
End

شکل ٤ شبه کد روش MMPG

در شکل ۵ یک مثال برنامه ریزی حرکت پنج روباتی حل شده است. شکل ۵(الف)، پیکربندی اولیه است. در شکل ۵(ب) حرکت همزمان همه روباتها با هم دیده می شود. شکل ۵(ج) نشان می دهد که سه روبات همزمان به یک گره رسیده اند. شکل ۵(د) نشان می دهد که ابتدا کدام روبات باید وارد شود؟ در شکل ۵(ه) انحراف روبات E برای ورود روبات A انجام می شود، و در شکل ۵(و) برگشت روبات E و عبور روبات A انجام شده است. باید توجه داشت که روباتهای B و C و D متوقفند. زمان ورود روبات D مهم است: در حالی که روبات D متوقف است روبات‌های A و E از گره بحرانی کانال می گذرند. در شکل ۵(ز) عبور روبات A از گره ٤ نشان داده شده است و دیده می شود که روبات B و نه C، بلافاصله پس از خروج روبات A وارد کانال شده است. در شکل ۵(ح)، روبات D که برای ورود به گره ۳ منتظر روبات B بوده است در تعارض بین سه روبات A و D و E بر سر گره ۳، وضعیت روبات B را که یک روبات غیر درگیر در وضعیت فعلی است برای تصمیم گیری در نظر دارد، یعنی وضعیتی که در آینده اتفاق خواهد افتاد بر تصمیم فعلی روبات D مؤثر بوده است. این نمایشی است از شیوه حل متمرکز. در این شکل روبات D بلافاصله پس از عبور روبات B وارد گره ۳ می شود. شکل ۵(ط) ورود روبات C به کانال را پس از عبور روبات D نشان می دهد و در شکل ۵(ی) پیکربندی نهائی دیده می شود.

جهت ارزیابی عملکرد الگوریتم ۲۰ مسئله حل شد که نتایج آن در جدول ۳ گزارش شده است. با مقایسه تعداد حرکات پیدا شده و تعداد حرکات مینیمم مطلق (برابر با جمع کوتاهترین مسیرها و بدون رفع قفل شدگی) دیده می شود که فاصله جواب‌های روش جدید از مقدار مینیمم مطلق به طور متوسط حدود ۱۱٪ است که این حرکات اضافی صرف رفع قفل‌شدگی ها شده‌اند.

[Downloaded from mjee.modares.ac.ir on 2024-12-29]



شکل ۵ شبیه سازی حل مسأله ترکیبی؛ (الف) پیکربندی اولیه؛ (ب) – (ی) مراحل میانی تا نهائی؛ (ک) ماتریس مسیر PM

جدول ۳ نتایج محاسباتی

زمان حل ارائه شده	زمان بدون در نظر	تعداد حرکتها در	تعداد حرکتها	نسبت تعداد	تعداد	تعداد گره-	شماره
۱۷۲	۲۱۰	۲۰۱	۱۸۹	۰/۷۰	۲۱	۳۰	۱
۱۳۹	۱۵۳	۱۴۳	۱۳۶	۰/۵۷	۱۷	۳۰	۲
۷۸	۱۰۴	۱۰۷	۹۱	۰/۴۳	۱۳	۳۰	۳
۱۹۹	۲۲۱	۲۲۷	۱۹۲	۰/۷۹	۱۹	۲۴	۴
۹۷	۱۰۰	۱۰۲	۸۶	۰/۵۰	۱۲	۲۴	۵
۲۱	۴۴	۳۷	۳۷	۰/۲۹	۷	۲۴	۶
۴۸	۶۵	۶۷	۵۸	۰/۷۰	۷	۱۰	۷
۲۲	۵۱	۵۱	۴۶	۰/۵۰	۵	۱۰	۸
۱۴	۲۵	۲۲	۲۲	۰/۳۰	۳	۱۰	۹
۲۲	۴۰	۴۲	۳۵	۰/۵۶	۵	۹	۱۰
۱۹	۴۷	۴۹	۴۳	۰/۴۴	۴	۹	۱۱
۱۱	۳۲	۳۰	۲۸	۰/۴۴	۴	۹	۱۲
۱۱	۲۰	۱۹	۱۷	۰/۳۳	۳	۹	۱۳
۱۵	۴۴	۴۷	۴۰	۰/۵۷	۴	۷	۱۴
۱۰	۲۳	۲۲	۲۰	۰/۴۳	۳	۷	۱۵
۸	۱۴	۱۴	۱۲	۰/۲۹	۲	۷	۱۶
۱۳	۳۵	۴۱	۳۰	۰/۸۳	۵	۶	۱۷
۱۱	۳۱	۳۰	۲۷	۰/۶۷	۴	۶	۱۸
۹	۱۸	۱۵	۱۵	۰/۵۰	۳	۶	۱۹
۸	۱۳	۱۲	۱۱	۰/۳۳	۲	۶	۲۰
۴۶/۳۵	۶۴/۵	۶۳/۹	۵۶/۷۵	میانگین			

[Downloaded from mjee.modares.ac.ir on 2024-12-29]

۷- نتیجه‌گیری

در این مقاله روش جدیدی با ترکیب رویکردهای متمرکز و غیرمتمرکز جهت حل مسائل برنامه ریزی حرکت چند روباتی ارائه شده است. فرآیند حل با پذیرش یک گراف اولیه مانند دیاگرام ورونویی به عنوان ورودی شروع شده، آنرا به گراف بهبود یافته تبدیل کرده، و سپس با شناسایی محل های قفل شدگی (کانال) و نوع تعارض، نسبت به حل قفل‌شدگی‌ها از طریق توقف یا انحراف روبات های معارض اقدام می شود. در این زمینه دو قضیه مطرح و اثبات شدند. نتایج محاسباتی و مقایسات با بهینه های مطلق مؤید کارایی الگوریتم ارائه شده می باشند.

جهت توسعه آتی الگوریتم می توان از ترکیب روشهای تغییر سرعت و اولویت دهی برای حل مسئله هایی که روباتها دارای سرعتهای متفاوت هستند بهره برد. همچنین می توان روش را برای حل مسائل در شرایط محیط ناشناخته (بهنگام) توسعه داد.

۸- منابع

[1] Latombe J.C.; *Robot Motion Planning*, Kluwer Academic Publishers, 1991.

[2] Hwang Y.K., Ahuja N.; "Gross Motion Planning–A Survey", *ACM Computation Surveys*, Vol. 24, No. 3, 1992, pp. 219-291.

[3] Lozano–Perez T., Wesley M.A.; "An algorithm for planning collision–free paths among polyhedral obstacles", *Communications of ACM*, Vol. 22, 1979, pp. 560-570.

[4] Canny J.F.; *The Complexity of Robot Motion Planning*, MIT Press, 1988.

[5] Choset H., Lynch K.M., Hutchinson S., et al.; *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005.

[6] Aurenhammer F.; "Voronoi Diagrams - A Survey of a Fundamental Geometric Data Structure", *ACM Computing Surveys*. Vol. 23, No. 3, 1991, pp. 345-405.

[7] Aronov B., de Berg M., van der Stappen A.F., Svestka P., Vleugels J.; "Motion planning for multiple robots" in *Proc. Int. Symp. Comp. Geom.* 1998, pp. 374-382.

[8] LaValle S.M., Hutchinson S.; "Optimal motion planning for multiple robots having independent goals", *IEEE Trans. on Rob. & Autom.*, Vol. 14, No. 6, 1998, pp. 912-925.

[9] Lumelsky V.J., Harinarayan K.R.; "Decentralized motion planning for multiple mobile robots: the cocktail party model", *Autonomous Rob.*, Vol. 4, 1997, pp. 121-135.

[10] Tanner, H.G., Kumar, A.; "Formation stabilization of multiple agents using decentralized navigation functions", *in Proc. Robotics: Sci. & Sys.*, Cambridge, USA, 2005.

[11] Parker, L.E.; "Path Planning and Motion Coordination in Multiple Mobile Robot Teams", *in Encyc. Complexity and System Science*, R.A. Meyers (ed.), Springer, 2009.

[12] Clark C.; "Probabilistic Road Map sampling strategies for multi-robot motion planning", *Robotics and Autonomous Systems*, Vol. 53, No.3, 2005, pp. 244-264.

[13] Ferrari E., Pagello J.; "A framework for robust multiple robots motion planning", *in Proc. IEEE IROS*, 1997, pp. 1684-1690.

[14] Kant K., Zucker S.W.; "Planning collision-free trajectories in time-varying environments: a two-level hierarchy", *The Visual Computer,* Vol. 3, No. 5, 1988, pp. 304-313.

[15] Cao Y., Fukunaga A., et al.; "Cooperative Mobile Robotics: Antecedents and Directions", *Autonomous Robots*, Vol. 4, No. 1, 1997, pp. 301-309.

[16] Erdmann M., Lozano-Pérez T.; "On multiple moving objects", *Algorithmica*, Vol. 2, No. 1, 1987, pp. 477-521.

[17] Bennewitz M, Burgard M, Thrun S.; "Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots", *Robotics and Autonomous Systems*, Vol. 41, No. 2, 2002, pp. 89-99.

[18] van der Stappen A.F., Overmars M.H., et al.; "Motion Planning in Environments with Low Obstacle Density", *Discrete and Computational Geometry*, Vol. 20, No. 4, 1998, pp 561-587.

[19] Pereira G., Kumar V. et al.; "Closed loop motion planning of cooperating mobile robots using graph connectivity", *Robotics and Autonomous Systems;* Vol. 56, No. 4, 2008, pp. 373-384.

[20] Ryan M.R.K.; "Exploiting subgraph structure in multi-robot path planning", *J. Artificial Intelligence Research*, Vol. 31, 2008, pp. 497-542.

[21] Masehian E., Samadian H., Daneshzand F.; "Analysis of Motion Feasibility of Multiple Mobile Agents on Graphs", *in Proc. Int. Conf. Information Technology* 2009, Amman, Jordan.

[Downloaded from mjee.modares.ac.ir on 2024-12-29]