خوشه‌بندی خود-پایاساز با قابلیت محدودسازی خطا در شبکه‌های حسگر بی‌سیم

وصال حکمی[1]، نستوه طاهری جوان[2] و مسعود صبایی[3]
[1]دانشگاه صنعتی امیرکبیر، vhakami@aut.ac.ir
[2]دانشگاه صنعتی امیرکبیر، nastooh@aut.ac.ir
[3]دانشگاه صنعتی امیرکبیر، sabaei@aut.ac.ir

چکیده - در این مقاله، مسأله «خوشه‌بندی» شبکه حسگر با ساخت «مجموعه مستقل ماکسیمال» در نظریه گراف معادل‌سازی شده و یک الگوریتم خوشه-بندی با ویژگیهای «خود-پایاسازی» و «محدودسازی خطا»- که از قابلیتهای کلیدی در بحث تحمّل‌پذیری خطا ویژه سیستمهای توزیعی بشمار می‌آیند- پیشنهاد می‌شود. روشهای قابل مقایسه موجود یا به‌کلی از ویژگی «محدودسازی خطا» بی‌بهره‌اند و یا طراحی آنها از اساس با فرض وجود یک «زمانبند متمرکز» صورت گرفته است. الگوریتم پیشنهادی ضمن اینکه از پیکربندی‌های تک‌خطایی با پیچیدگی زمانی و مکانی $O(1)$ ترمیم می‌شود، تحت سیاست زمانبندی «توزیعی ناعادلانه»- که بیشترین تطبیق را با محیط عملیاتی شبکه‌های حسگر دارد- کار می‌کند. برخورداری الگوریتم از مشخصه‌های «خود-پایاسازی» و «محدودسازی خطا» با استدلالِ رسمی نشان داده می‌شود؛ نتایج شبیه‌سازی نیز حاکی از آن است که صرف نظر از تعداد و تراکم گره‌ها، روش پیشنهادی علاوه بر ترمیم سریع در مقابل خطاهای مقیاس کوچک، زمانِ رسیدن به پایداری با شروع از پیکربندی دل‌خواه اولیه را نیز نسبت به روش‌های قبلی بهبود می‌دهد. تحقق ساختار خوشه‌بندی کارآمدتر، کاهش تعداد پیامهای بروزرسانی و پایدارسازی با حداقل تغییر در ساختار توپولوژیکی خوشه‌بندی از دیگر مزایای الگوریتم می‌باشند.

کلید واژه- خود-پایاسازی، خوشه‌بندی، شبکه‌های حسگر بی‌سیم، صرفه‌جویی انرژی، محدودسازی خطا.

۱- مقدمه

معماریهای ساده و غیرسلسله‌مراتبی در «شبکه‌های حسگر بی‌سیم»، متشکل از تعداد زیادی گره، مقیاس‌پذیر نبوده و به لحاظ مصرف انرژی نیز کارآمد نیستند. در این راستا، «خوشه-بندی» شبکه به منظور فراهم نمودن قابلیت خود-سازماندهی و میسّر ساختن عملیات مسیریابی سلسله‌مراتبی یکی از راهکارهای موفق و متداول محسوب می‌گردد. به طور کلی، در نبود یک زیرساختار ثابت و با توجه به پویایی شبکه حسگر و لزوم برقراری ارتباطاتِ چندگامه، روشهای خوشه‌بندی که منجر به پایداری بالاتر سیستم و برخورداری آن از قابلیت بازپیکربندی بدون دخالت خارجی شوند، مطلوب‌تر خواهند بود. خوشه‌بندی با ویژگی «خود-پایاسازی» [1] علاوه بر بی‌نیاز ساختن شبکه از پیکربندی اولیه، امکان ترمیم خودکار از خطاهای گذرا، ناشی از تغییرات محیطی، خرابی حسگرها یا تغییر وضعیت داخلی آنها، گسستگی ساختار ارتباطی و بالأخره تغییرات ناهمگام پیکربندی را فراهم می‌سازد. «خود-پایاسازی» از رویکردهای مطرح در راستای ایجاد قابلیت تحمّل‌پذیری خطا، به ویژه در سیستمهای توزیعی پویا می‌باشد که اخیراً در حوزه پژوهشی شبکه‌های حسگر نیز مورد توجه ویژه واقع شده است [2]. بنا به تعریف، یک سیستم «خود-پایاساز» بدون توجه به حالت اولیه تضمین می-کند که بعد از طی تعداد متناهی گام به صورت خودکار و بدون دخالت عوامل خارجی (نیروی انسانی) به یک حالت معتبر همگرا شود. تصحیح کار سیستم پس از هرگونه خطای «گذرا» تنها با اتکاء به دانش محلی و بی نیاز به اطلاعات سراسری از ویژگی‌های قابل ملاحظه سیستمهای خود-پایاساز محسوب می‌گردد [1].

اگرچه مسأله خوشه‌بندی با خصوصیت خود-پایاسازی در شبکه‌های حسگر کمابیش مورد مطالعه قرار گرفته ولی به قابلیت پایداری توأم با ویژگی «محدودسازی حوزه خطا» [3] که مناسبت بیشتری با مشخصه‌های این نوع شبکه‌ها دارد، به ندرت پرداخته شده است. به منظور بهبود کارایی روشهای تحمل‌پذیری خطا، تضمین ترمیم سریعتر از کلیه پیکربندیهای تک‌خطایی بدون اینکه قابلیت بازیابی نهایی سیستم از خطاهای گسترده‌تر تحت تأثیر قرار گیرد، اهمیت ویژه‌ای دارد. سیاست برخورد یکسان الگوریتم‌های خوشه‌بندی خود-پایاساز با انواع مختلف خطا صرف نظر از میزان گستردگی آنها، تبعات ناخوشایندی نظیر بازه زمانی طولانی-مدت خارج از سرویس، اتلاف انرژی بیشتر جهت بازگشت به حالت پایدار و یا تغییر گسترده‌تر ساختار توپولوژیکی خوشه‌بندی را در پی دارد.

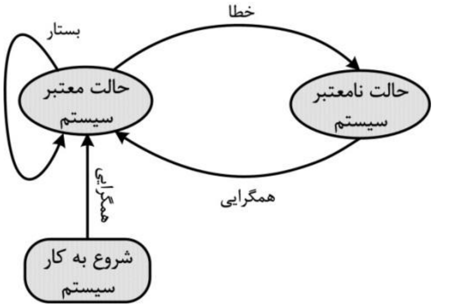
در این مقاله، هدف، ارائه الگوریتمی با هر دو ویژگی «خود-

-پایاسازی» و «محدودسازی خطا» جهت خوشه‌بندی شبکه حسگر از طریق ساخت مجموعه «مستقل ماکسیمال» در گراف نظیر شبکه می‌باشد. الگوریتم پیشنهادی تحت سیاست زمانبندی توزیعی، در شرایط وقوع خطاهای مقیاس کوچک از گسترش خطا در سطح شبکه جلوگیری می‌کند. برای این منظور، از نظریه «محدودسازی خطا» [3, 9] که رویکردی «پوششی» در بحث تحمّل‌پذیری خطاست در ترکیب با خود-پایاسازی جهت محدود ساختن حوزه مکانی و زمانی تأثیر خطاهای مقیاس کوچک استفاده شده است. این امر، موجب کاهش تعداد بِروزرسانیها و تغییر حالتها و در نتیجه کاهش تعداد «همه‌پخشی»ها و صرفه-جویی توان و بالا رفتن عمر شبکه می‌گردد. از سوی دیگر، با جلوگیری از انتشار خطا در سطح شبکه، زمان رسیدن به پایداری نیز کاهش می‌یابد. میزان «قابلیت دسترسی سیستم» با استفاده از این الگوریتم، از طریق حذف حالت میانی که در روشهای پیشین تعبیه شده و نیز کاهش زمان رسیدن به پایدرای، بهبود می‌یابد. کاهش تعداد تغییر حالتها از بهم‌ریختگی آرایش توپولوژیکی شبکه جلوگیری نموده و گرهها حتی‌الامکان در خوشه خود باقی می‌مانند. تدوین قوانین پایداری در طراحی الگوریتم پیشنهادی بر مبنای محدودسازی حوزه خطا در حالت کلی نیز با هدف تشکیل توپولوژی خوشه‌بندی کارآمدتر نسبت به روشهای پیشین صورت گرفته است؛ در واقع، ماهیت قوانین تعبیه شده برای گرهها به گونه‌ایست که تعداد گرههای سرخوشه بی‌فایده در توپولوژی حاصل حتی‌الامکان کاهش یابد.

سازماندهی مطالب مقاله به این شرح است: در بخش ۲، ضمن معرفی اجمالی مفاهیم پایه به مرور کارهای پیشین خواهیم پرداخت. در بخش ۳، الگوریتم پیشنهادی تشریح شده و بُرهان‌های مربوط به بررسی صحت کارکرد آن ارائه می‌گردد. بخش ۴ نیز به ارزیابی و مقایسه عملکرد الگوریتم پیشنهادی در قالب مجموعه آزمایشهای شبیه‌سازی اختصاص داده شده است. ارائه خلاصه مطالب و نتیجه‌گیری پایان‌بخش مطالب مقاله خواهد بود.

۲- پیشینه نظری و کارهای مرتبط

شرط لازم و کافی برای برخورداری یک سیستم از قابلیت خود-پایاسازی مبتنی بر دو ویژگی می‌باشد: تضمین همگرایی سیستم به یک پیکربندی مجاز سراسری (مطلوب طراح) با شروع از هر وضعیت اولیه دلخواه و تضمین استقرار سیستم در پیکربندی مجاز مادامی که خطایی رخ ندهد (بَستار) [1, 8]؛ بر اساس این دو ویژگی، یک سیستم خود-پایاساز نیازی به راه-اندازی (یا مقداردهی) اولیه نداشته و می‌تواند مشروط برآنکه دستخوش خطای بیشتری قرار نگیرد، به صورت خودکار و بدون هر نوع دخالت خارجی از یک یا چند خطای گذرا (موقتی) ترمیم شده و مجدداً به یک پیکربندی مجاز همگرا شود (شکل ۱).



شکل ۱: دیاگرام حالت سیستمی که بصورت خود-پایاساز اجرا می‌شود.

تحقیقات بسیاری تابحال در زمینه طراحی الگوریتمهای خود-پایاساز برای ساخت مجموعه‌های مستقل و غالب در حوزه نظریه گراف ارائه شده [5]، امّا بخش قابل ملاحظه‌ای از این کارها بر مبنای مدلهای ارتباطاتی مطمئن (مانند حافظه اشتراکی) و با فرض اِعمال سیاست زمانبندی متمرکز [6] طراحی شده‌اند که در واقع شرایط محدودکننده‌ای را برای کارکرد در شبکه‌های حسگر تحمیل می‌نمایند. علاوه بر این، برخی از این الگوریتمها اساساً با دید کاربردی خاصی (مثلاً: خوشه‌بندی) طراحی نشده و بعضاً شکل توپولوژیکی حاصل از آنها مطلوب شبکه حسگر نیست [7]. در حوزه کارهای مرتبط با ایده مطرح در این مقاله، تنها الگوریتمی که از پیچیدگی زمانی «خطی» برخوردار بوده و با فرض سیاست زمانبندی توزیعی به حل مسأله می‌پردازد، روش ارائه شده در [4] می‌باشد. قابلیت خود-پایاسازی در [4] از طریق تعبیه حالتهای میانی در کارکرد الگوریتم صورت گرفته که البته این امر نیز خود به صورت بالقوه منبع خطا محسوب می‌گردد و می‌تواند منجر به کاهش « قابلیت دسترس‌پذیری» سیستم شود.

از دیگر ایراداتی که به کلیه الگوریتمهای موجود در این زمینه وارد است، عدم تمایز در چگونگی مدیریت خطاها بر مبنای گستردگی آنهاست. رخداد تنها یک خطا در پیکربندی معتبر می‌تواند منجر به واکنش الگوریتم در قالب بروزرسانیهای متعدد سرتاسری شده و در نتیجه خطای مزبور مقیاس وسیعی از شبکه را تا رسیدن به پایداری تحت تأثیر قرار دهد. هر بروزرسانی در شبکه‌های بی‌سیم در واقع معادل یک «همه-پخشی» به گرههای همسایه است که با توجه به محدودیت انرژی گره‌ها منجر به کوتاه‌شدن عمر آنها و در نتیجه تنزّل بازدهی شبکه خواهد شد. مسأله دیگر، زمان طولانی صرف شده تا رسیدن به پایداری است که طی آن عملاً سیستم مطابق

چارچوب مدّنظر طراح کار نمی‌کند و تا لحظه رسیدن به پایداری در یک حالت نامعتبر قرار دارد. تعداد بالای بروزرسانیها همچنین منجر به بهم‌ریختگی آرایش شبکه شده و ساختار توپولوژیکی خوشه‌بندی را دچار سلسله تغییرات نامطلوب می‌نماید: سرخوشه‌ها به‌ناگاه تعویض شده و گرهها ناگزیر می‌بایست به خوشه‌های جدید بپیوندند.

۳- الگوریتم پیشنهادی

در این بخش، یک توصیف سطح بالا از الگوریتم خوشه‌بندی مبتنی بر ساخت «مجموعه مستقل ماکسیمال» که دارای هر دو ویژگی خود-پایاسازی و محدودسازی خطاست (با نام اختصاری MIS*fc*) ارائه می‌شود.

۳-۱- شرحی بر طراحی و نحوه کارکرد الگوریتم

اگر فرض شود حالت ۱-خطایی با خطا در گره v تنها با یک تغییر نامطلوب در مقدار یکی از متغیرهای گره v حاصل شده، به سادگی می‌توان نشان داد که در چنین حالتی دو شرط برقرار است: (۱) یکی از قانون‌ها در گره v فعال خواهد بود. (۲) می‌توان با اجرای یکی از قانون‌ها تنها در گره v به پایداری رسید (در برخی موارد، اجرای قانون در یکی از همسایه‌ها نیز منجر به پایداری می‌گردد). هدف، شناسایی و رفع حالت‌های ۱-خطایی با اجرای قانون تنها در همان گره خطادار، و ممانعت از انتشار خطا با اجرای ناخواسته در گره‌های همسایگی v، $N(v)$، است. برای سادگی در درک الگوریتم پیشنهادی، ابتدا به تعریف تعدادی گزاره- که در واقع، پیش‌شرطهای اجرای عملیات تغییر وضعیت در گرهها می‌باشند- و تعدادی مجموعه برای خوانایی بهتر شِبه-کُد الگوریتم می‌پردازیم (شکل ۲).

برای مجموعه مستقل ماکسیمال دو حالت ۱-خطایی خواهیم داشت: اول خطای خارج شدن یک گره عضو از مجموعه (یعنی IN به OUT)، و دوم خطای ورود یک گره غیرعضو به داخل مجموعه (یعنی OUT به IN)؛ از ویژگیهای قابل ملاحظه الگوریتم پیشنهادی حذف حالت میانی WAIT است که در [30] مطرح شده بود. قوانین الگوریتم MIS*fc* با هر دو ویژگی خود-پایداری و محدودسازی خطا نیز در شکل ۳ نمایش داده شده‌اند.

قانون‌های ۱ تا ۳ ویژه مدیریت تغییر وضعیت گرهها در سناریوی تک‌خطای IN به OUT تعبیه شده‌اند. دو قانون اول، مبتنی بر تشخیص حالت ۱-خطایی بوده و قانون ۳ شرایط بیش از یک خطا را در گره v و همسایگانش کنترل می‌کند.

$\boldsymbol{inNeighbor(v)} \equiv \exists w \in N(v){:}\, w.state = IN$

$\boldsymbol{conflictIn(v)} \equiv v.state = IN \wedge \exists w \in N(v){:}\, w.state = IN$

$\boldsymbol{Pending(v)} \equiv state.v = OUT \ \wedge\ \sim inNeighbor(v)$

$\boldsymbol{pendigNeighbors(v)} \equiv w \in N(v){:}\, Pending(w)$

$\boldsymbol{inNeighborWithLowerId(v)} \equiv \exists w \in N(v){:}\, w.state = IN \wedge w.id < v.id$

$\boldsymbol{solePending(v)} \equiv Pending(v) \wedge \forall w \in N(v){:} \sim Pending(w)$

$\boldsymbol{canOut(v)} \equiv conflictIn(v) \wedge execution\ of\ (v.state \coloneqq OUT) \Longrightarrow \{\sim pendingNeighbors(v) \wedge \sim conflictIn(v) \wedge (\forall w N(v){:} \sim conflictIn(w))\}$

شکل ۲: گزاره‌ها و مجموعه‌های مورد استفاده در شِبه‌کُد الگوریتم MIS*fc*.

R1. $canOut(v) \wedge \big(\forall w \in N(v){:} \sim canOut(w)\big) \rightarrow v.state \coloneqq OUT$

R2. $canOut(v) \wedge \big(\forall w \in N(v){:}\, (canOut(w) \wedge v.id > w.id)\big) \rightarrow v.state \coloneqq OUT$

R3. $conflictIn(v) \wedge \sim canOut(v) \wedge \big(\forall w \in N(v){:} \sim canOut(w)\big) \wedge inNeighborWithLowerId(v) \rightarrow v.state \coloneqq OUT$

R4. $Pending(v) \wedge \big(|pendingNeighbors(v)| > \max_{\forall w \in pendingNeighbors(v)} |pendingNeighbors(w)|\big) \vee \Big(\big(|pendingNeighbors(v)| = \max_{\forall w \in pendingNeighbors(v)} |pendingNeighbors(w)|\big) \wedge (v.id < w.id)\Big) \rightarrow v.state \coloneqq IN$

R5. $soloPending(v) \rightarrow state.v \coloneqq IN$

شکل ۳: قوانین تعبیه شده در طراحی الگوریتم MIS*fc*.

گزاره $confilictIn(v)$، شرط لازم برای برقراری هر یک از این سه قانون است. در قانون ۱، با بررسی $canOut(v)$، حالت ۱-خطایی را تنها در گره v، و نه در هیچ یک از همسایه‌هایش، یعنی: $\forall w \in N(v){:} \sim canOut(w)$، تشخیص می‌دهیم که تحت این شرایط برای رسیدن به حالت پایدار تنها کافی است گره v متغیر وضعیتش را تغییر حالت بدهد و از مجموعه خارج شود. قانون ۲ هنگامی فعال می شود که حالت ۱-خطایی هم در گره v و هم در گره(های) همسایه v، مثل w، داریم. بدین معنی که هر کدام از دو گره v یا w از مجموعه خارج شوند، به شرایط پایداری می‌رسیم، اما باید توجه شود که خروج همزمان این دو گره با هم شرایط ناپایداری را برای ما رقم می‌زند، از این رو به منظور شکست تقارن بوجود آمده و جلوگیری از اجرای همزمان، در قانون ۲ اگر گره v شماره شناسه‌ی بزرگتری نسبت به گره(های) همسایه‌اش داشت، از مجموعه خارج می‌شود. در حقیقت، اولویت بقای گرهها در مجموعه ماکسیمال مستقل در شرایط مساوی با گره‌هایی است که شماره شناسه کوچکتری

دارند. در ادامه، قانون‌های ۴ و ۵ ویژه مدیریت تغییر وضعیت گرهها در سناریوی تک‌خطای OUT به IN تعبیه شده‌اند. برقراری گزاره $Pending(v)$ شرط لازم جهت فعالسازی این دو قانون است. در قانون ۴، گره v زمانی عضو مجموعه می‌شود که تعداد اعضای مجموعهٔ $pendingNeighbors$ گره از گره(های) همسایه‌اش بیشتر باشد. به طور شهودی، در واقع، بررسی می‌شود تا گره‌ای حرکت کند که بیشترین مشکل را بین خود و همسایه‌هایش رفع کند (بیشترین تعداد گره را از حالت $pending$ خارج کند). در صورتی که تعداد اعضای این مجموعه برای گره جاری و گره همسایه‌اش هر دو همزمان بزرگترین مقدار باشند، از شماره شناسه‌ی گرهها برای شکست تقارن کمک می‌گیریم. قانون ۵ حالتی را بررسی می‌کند که گره v تنها گره $Pending$ در بین خود و همسایه‌هایش است. در صورت فعال شدن این قانون، گره مورد نظر بلافاصله وارد مجموعه می‌شود.

۳-۲- اثبات درستی الگوریتم

لِم ۱ (شرط بَستار): *در پیکربندی‌ای که هیچ گره‌ای فعال نمی‌باشد، مجموعه* $I = \{v|v.state = \mathrm{IN}\}$ *یک مجموعه مستقل ماکسیمال می‌سازد.*

این لِم بسادگی از طریق برهان خلف اثبات می‌گردد؛ به‌دلیل محدودیت فضا از تشریح آن در این بخش صرف نظر شده است.

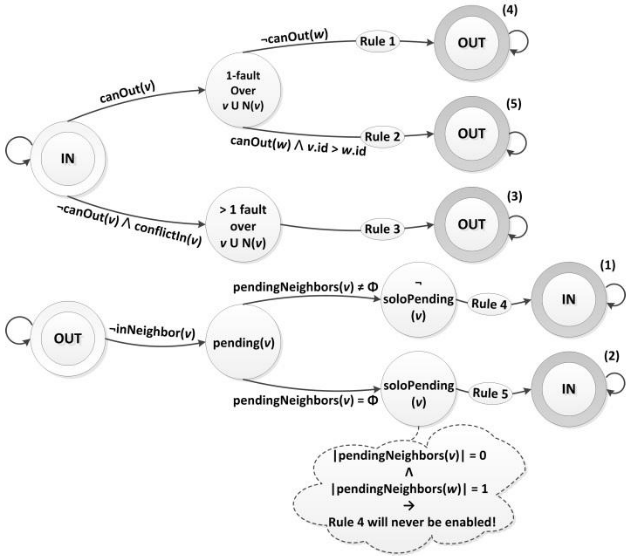
لِم ۲: *در حین اجرای الگوریتم، گره v با حالت اولیه دلخواه، یکی از دنباله های* $\mathrm{IN} \rightarrow \mathrm{OUT}$ *،* $\mathrm{OUT} \rightarrow \mathrm{IN}$ *و یا* $\mathrm{IN} \rightarrow \mathrm{OUT} \rightarrow \mathrm{IN}$ *را اجرا خواهد کرد، و بعد از آن هیچ قانون دیگری را اجرا نمی‌کند.*

اثبات: حالت‌های ممکن در شکل ۴ نشان داده شده‌اند که به تفصیل در زیر بررسی می‌شوند.

وضعیت (۱). روال رسیدن گره v به این وضعیت همان‌گونه که در شکل ۴ نشان داده شده با شروع از حالت اولیه OUT و تحت شرایط برقراری $Pending(v)$ و عدم برقراری $soloPending(v)$ می‌باشد. بعد از این دور، با عضو شدن (IN کردنِ) گره v، با توجه به اینکه گره v هیچ همسایه‌ای با وضعیت IN نداشته است ($\sim inNeighbor(v)$)، دیگر هیچ‌گاه $confilictIn(v)$ برقرار نخواهد شد. به عبارت دیگر، کلیه همسایگان v، $N(v)$، همواره در حالت OUT باقی می‌مانند.

وضعیت (۲). روال رسیدن گره v به این وضعیت همان‌گونه که در شکل ۴ نشان داده شده با شروع از حالت اولیه OUT و تحت شرایط برقراری هر دو گزاره $Pending(v)$ و $soloPending(v)$ است. بعد از این دور، وضعیت گره v به IN تغییر کرده و کلیه همسایه‌های آن دارای وضعیت OUT با حداقل دو گره همسایه IN می‌باشند (خارج از مجموعه با حداقل دو همسایه عضو)؛ بنابراین گرههایِ مجموعه $\{v\} \cup N(v)$ هرگز قانون دیگری برای تغییر حالت اجرا نخواهند کرد.

وضعیتهای (۳)، (۴) و (۵). با شروع از حالت اولیه IN، اجرای قوانین متناظر با دیاگرام حالت بالایی در شکل ۴ منجر به قرارگیری در یکی از وضعیتهای (۳)، (۴)، یا (۵) با حالت OUT برای گره مورد نظر خواهد شد. در این وضعیتها، خاتمه‌پذیری روتینِ پایداری گره v این بار به ازای شروع از حالت OUT با توجه به تغییر گذرهای قسمت پایینی شکل، یعنی همان روال نظیر وضعیتهای (۱) و (۲)، استنتاج می‌شود؛ البته به طور دقیقتر، می‌توان نشان داد که وضعیت‌های (۴) و (۵) با توجه به اینکه ناشی از رخداد حالت‌های ۱-خطایی هستند، پایانی می‌باشند (لِم ۳) و هیچ قانونی بعد از آن‌ها فعال نمی‌شود و تنها در وضعیت (۳) امکان دور مجدد $\mathrm{OUT} \rightarrow \mathrm{IN}$ وجود دارد.



شکل ۴: دیاگرام حالت الگوریتم MISfc.

قضیه ۱ (شرط همگرایی): *با شروع از هر پیکربندی دل‌خواه و طی تعداد متناهی گام، دیگر هیچ گره‌ای فعال نخواهد بود و مجموعه* $I = \{v|v.state = IN\}$ *مستقل ماکسیمال است.*

اثبات: در شروع از هر پیکربندی دل‌خواه در لِم ۲ نشان داده شد که هر گره بعد از حداکثر ۳ گام دیگر قانونِ فعالی نخواهد داشت. از سویی، طبق لِم ۱، در پیکربندی‌ای که در آن هیچ گره‌ای فعال نباشد، I یک مجموعه مستقل ماکسیمال خواهد بود.

لِم ۳ (قابلیت محدودسازی خطا): *در یک پیکربندی مجاز، در صورت وقوع یک خطا در وضعیت گره دلخواه j، تنها با* $O(1)$ *حرکت به وضعیت مجاز برمی‌گردیم.*

اثبات: دو حالت می‌تواند رخ دهد:

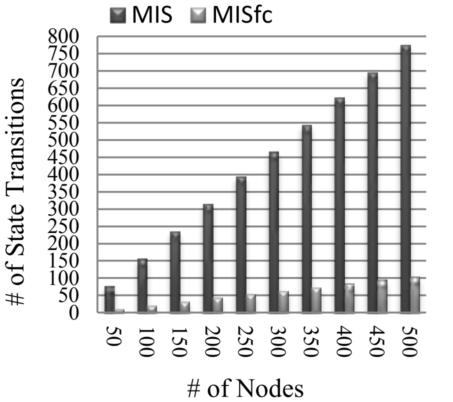
(الف) با بروز خطا، گره j از وضعیت OUT به وضعیت IN تغییر حالت داده است (($confilictIn(j)$ برقرار است): (الف)-۱ j بیش از یک همسایه IN دارد: در این حالت $canOut(j)$ برقرار است ولی به ازای هیچ کدام از همسایه‌های j، مثل i، $canOut(i)$ برقرار نخواهد بود چرا که علی‌رغم خروج i از مجموعه، $confilictIn(j)$ همچنان برقرار خواهد ماند. از این رو در این حالت تنها قانون ۱ برای گره j برقرار است و با یک تغییر وضعیت به حالت مجاز برمی‌گردیم. (الف)-۲ j تنها یک همسایه IN، مثل i، دارد: در این حالت، با توجه به اینکه OUT کردن j لزوماً منجر به برقراری حالت مجاز می‌شود، کلیه گزاره‌های مقابل در این وضعیت برقرار خواهند بود: $pendingNeighbors(j) = \emptyset$ و $\sim confilictIn(j)$ و $\forall k \in N(j): \sim confilictIn(k)$، که در نتیجه، برقراری $canOut(j)$ را می‌رساند. در این شرایط، در صورتی که $canOut(k)$ برقرار نباشد، گره j تنها گره‌ای خواهد بود که با توجه به قانون ۱ به وضعیت OUT تغییر حالت خواهد داد و سیستم به حالت مجاز برخواهد گشت. در شرایطی که $canOut(k)$ هم برقرار باشد، قانون ۲ صرفاً برای یکی از دو گره j یا k (گره‌ای که شماره شناسه‌ی بزرگتری دارد)، فعال خواهد شد و بنابراین تنها با یک تغییر وضعیت به پیکربندی مجاز بازخواهیم گشت.

(ب) با بروز خطا، گره j از وضعیت IN به وضعیت OUT تغییر حالت داده است (($Pending(j)$ برقرار است): (ب)-۱ تمام همسایگان j همسایه‌ای با وضعیت IN دارند: در این حالت، گزاره $Pending(j)$ برقرار است و در نتیجه، تنها قانون ۵ برای گره j فعال خواهد شد و بنابراین تنها با یک تغییر وضعیت به پیکربندی مجاز بازخواهیم گشت. قانون ۴ به این دلیل فعال نخواهد شد که $|pendingNeighbors(v)|$ همواره کوچکتر از $|pendingNeighbors(w)|$، $w \in N(v)$، است. (ب)-۲ همسایه یا همسایگانی از j وجود دارند که آنها نیز با خروج j، به حالت $Pending$ می‌روند؛ برای سادگی فرض کنیم تنها گره i در همسایگی j این گونه است. در این حالت، گزاره‌های $Pending(j)$ و $Pending(i)$ برقرار خواهند بود. سه حالت داریم: ۱- $|pendingNeighbors(j)| > |pendingNeighbors(i)|$: تنها در گره j قانون ۴ فعال خواهد شد و با یک حرکت به حالت مجاز می‌رسیم. ۲- $|pendingNeighbors(j)| < |pendingNeighbors(i)|$: تنها در گره i قانون ۴ فعال خواهد شد و با یک حرکت به حالت مجاز می‌رسیم. ۳- $|pendingNeighbors(j)| = |pendingNeighbors(i)|$: گره‌ای که شماره شناسه کوچکتری داشته باشد قانون ۴ در آن فعال شده و با یک حرکت به پیکربندی مجاز باز می‌گردیم.

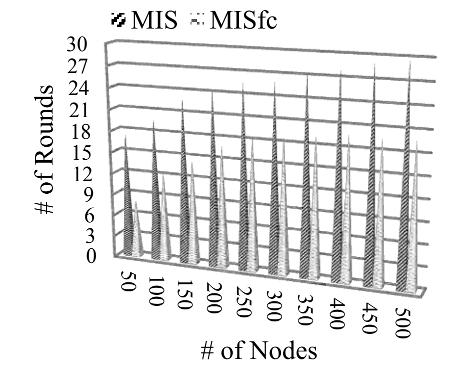
۴- ارزیابی کارایی

در این بخش، کارایی الگوریتم خوشه‌بندی MIS*fc* به لحاظ «تعداد تغییر حالتها»، «تعداد سیکلهای زمانی لازم تا پایداری» و «ساختار توپولوژیکی خوشه‌بندی» با الگوریتم ارائه شده در [4] (MIS) مقایسه می‌شود. روال کلی انجام شبیه‌سازیها بر دو مبنای شروع از پیکربندی اولیه (همه گرهها در حالت OUT) و شروع از پیکربندی چندخطایی بوده که تحت سیاست زمانبندی «توزیعی ناعادلانه» تحت آزمایش قرار می‌گیرند. کلیه آزمایشات شبیه‌سازی با استفاده از نرم‌افزار MATLAB صورت گرفته و نتایج به ازای ۱۰۰ مرتبه آزمایش در هر سناریو گزارش می‌شود.

شکلهای ۵ و ۶، مقایسه عملکرد MIS و MIS*fc* تحت سیاست زمانبندی «توزیع‌شده ناعادلانه» با پیکربندی اولیه تماماً غیرعضو را نمایش می‌دهد که در آن متوسط «درجه همبندی» به طور ثابت 7 فرض شده و تعداد گرهها متغیّر می‌باشد.



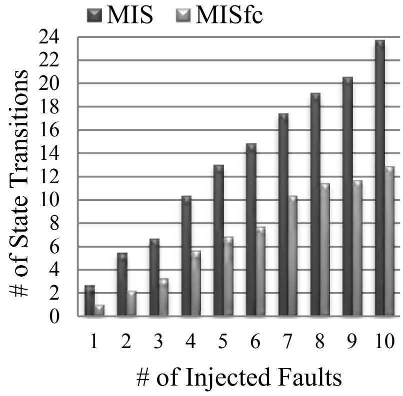
شکل ۵: مقایسه تعداد «تغییر حالتها» در MIS*fc* و MIS.



شکل ۶: مقایسه تعداد «سیکلهای زمانی» در MIS*fc* و MIS.

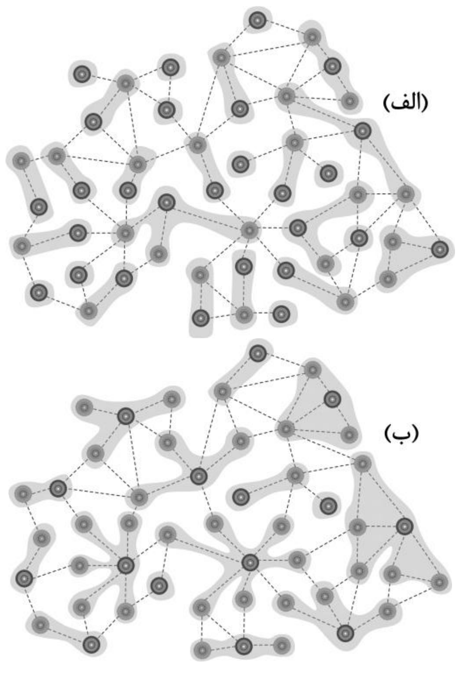
با توجه به نمودارها، می‌توان نتیجه گرفت که نسبت کارایی MIS*fc* به MIS با افزایش «تعداد گرهها» بیشتر می‌گردد. علاوه بر این، صرف نظر از مقدار «درجه همبندی»، MIS همواره به تعداد ثابت $2n$ «تغییر حالت» خواهد داشت (n: تعداد گرهها)؛ در حالیکه MIS*fc* در توپولوژیهای متراکم‌تر کارایی بالاتری دارد.

در سناریویی دیگر، عملکرد این دو الگوریتم با تزریق خطا به پیکربندی پایدار مقایسه شده است؛ در این آزمایش که نمودار مربوط به آن در شکل۷ نشان داده شده، تعداد خطاها متغیّر و از ۱ تا ۷ خطا روی یک توپولوژی متشکل از ۱۰۰ گره و متوسط «درجه همبندی» ۷ می‌باشد.



شکل ۷: تعداد «تغییر حالتها» در *MISfc* و MIS (با تزریق خطا به پیکربندی).

الگوریتم MIS*fc* حالتهای تک‌خطایی را تنها طی یک «تغییر حالت» و «یک سیکل زمانی» به پایداری می‌رساند؛ در حالیکه، MIS به طور متوسط به سه «تغییر حالت» و سه «سیکل زمانی» نیازمند است. با افزایش تعداد خطاها نسبت کارایی MIS*fc* به MIS کاهش یافته و به حدود ۲ برابر برای تعداد «تغییر حالتها» و ۱/۵ برابر به ازای تعداد «سیکلهای زمانی» می‌رسد.



شکل ۸: مقایسه توپولوژی حاصل از MIS و MIS*fc*؛ (الف): MIS، (ب): MIS*fc*

شکل ۸ ساختار خوشه‌بندی حاصل از اجرای دو الگوریتم MIS*fc* و MIS را روی یک توپولوژی اولیه ۵۰ گرهی نمایش می‌دهد. تعداد خوشه‌های حاصل از اجرای الگوریتم MIS، ۲۷ می‌باشد که از این میان تعداد ۹ خوشه تک‌عضوی بوده و تنها متشکل از خود گره «سرخوشه» می‌باشند. این در حالیست که در نتیجه اجرای الگوریتم MIS*fc* تعداد کل خوشه‌ها به ۱۵ عدد کاهش یافته و نیز تنها ۲ خوشه تک‌عضوی خواهیم داشت.

۵- نتیجه‌گیری

با توجه به مستعد خطا بودن گرهها، امکان تغییرات محیطی و عدم دسترسی به ساختار بعد از «کارگذاری» شبکه‌های حسگر، روشهای خوشه‌بندی که منجر به پایداری بالاتر سیستم و برخورداری آن از قابلیت بازپیکربندی بدون دخالت خارجی شوند، مطلوب‌تر خواهند بود. در این مقاله، یک روش خوشه‌بندی مبتنی بر ساخت «مجموعه مستقل ماکسیمال» با هر دو قابلیت «خود-پایاسازی» و «محدودسازی حوزه خطا» ارائه شده که علاوه بر پایداری سریع در مقابل خطاهای مقیاس کوچک، زمانِ رسیدن به پایداری با شروع از پیکربندی دل‌خواه اولیه را نیز بهبود می‌دهد. در دیدگاه دیگری که مبتنی بر ساخت خوشه‌ها از طریق تشکیل «کوچکترین مجموعه غالب» می‌باشد، شرط عدم مجاورت گرههای «سرخوشه» الزامی نبوده و به این ترتیب در سناریوهایی که به دلیل تغییر مکان گرهها بعد از «کارگذاری»، امکان قرارگیری «سرخوشه»ها در همسایگی هم وجود دارد، تعداد تغییر حالات کمتری تا پیکربندی مجاز نیاز خواهد داشت. قابل ذکر است طراحی این الگوریتم نیز انجام گرفته و در کارهای آتی گزارش خواهد شد.

مراجع

[1] S. Tixeuil, "Algorithms and Theory of Computation Handbook, Second Edition, chapter Self-stabilizing Algorithms," CRC Press, Taylor & Francis Group, 2009.

[2] V. Turau and C. Weyer,"Fault tolerance in wireless sensor networks through self-stabilisation," *Int. J. Communication Networks and Distributed Systems*, Vol.2, No. 1, pp. 78-98, 2009.

[3] S. Ghosh, A. Gupta and S.V. Pemmaraju, "Fault-containing network protocols," *Proc. of the ACM Symposium on Applied Computing*, pp. 431-437, USA, 1997.

[4] V. Turau, "Linear self-stabilizing algorithms for the independent and dominating set problems using an unfair distributed scheduler," *Information Processing Letters*, Vol. 103, pp. 88-93, 2007.

[5] N. Guellati and H. Kheddouci, "A Survey on Self-Stabilizing Algorithms for Independence, Domination, Coloring, and Matching in Graphs," *J. of Parallel and Distributed Computing*, Vol. 70, No. 4, pp. 406-415, 2010.

[6] J. C. Lin and T. C. Huang, "An Efficient Fault-Containing Self-Stabilizing Algorithm for Finding a Maximal Independent Set," *IEEE Trans. Parallel and Distributed Systems*, Vol. 14, pp. 742-754, 2003.

[7] S. M. Hedetniemi, et al., "Self-stabilizing Algorithms for Minimal Dominating Sets and Maximal Independent Sets," *Computers & Mathematics with Applications*, Vol. 46, pp. 805-811, 2003.

[8] S. Dolev, "Self-stabilization," MIT Press, 2000.

[9] A. Dasgupta, "Extensions and Refinements of Stabilization," PhD Dissertation, University of Iowa, 2009.