بسمه تعالی

سیستم‌های توزیع‌شده — نیمسال دوم ۰۰-۹۹
گزارش پروژه تحقیقاتی عملی

محمدحسین زارعی، سید محمد حسینی
شماره دانشجویی: ۹۶۱۰۱۷۰۸، ۹۶۱۰۱۴۸۷

پیاده‌سازی الگوریتم GHS به صورت توزیع شده

۱ مقدمه

۱.۱ تعریف مسئله و کاربرد

در مسئله‌ي یافتن درخت پوشای بهینه به صورت توزیع‌شده، یک گراف بدون جهت و همبند با N گره و E یال داریم. در این گراف، هر گره نماینده‌ي یک پردازنده بوده که کد الگوریتم روی آن اجرا می‌شود و هر یال نماینده‌ی یک کانال ارتباطی دوطرفه بوده که پیام‌ها به صورت مستقل در هر دو طرف منتقل می‌شوند. در حالت توزیع‌شده، هر گره اطلاعات محلی از گراف را دارد. یعنی فقط از همسایه‌های خودش اطلاع دارد. همچنین فرض شده‌است که هر پیام پس از تاخیری محدود اما نامشخص و بدون خطا و به ترتیب (FIFO) به مقصد می‌رسد. همچنین هر یال یک وزن محدودی هم دارد که حتما باید از سایر یال‌ها متمایز باشد. یکی از مهمترین کاربردهای MST (Minimum Spanning Tree) ، در broadcasting پیام در شبکه است. یعنی هر گره، پیام خودش را برای آنکه به تمامی گره‌ها برساند، کافی است تا از روی یال‌های درخت پوشای بهینه، برای بقیه ارسال کند. سایر گره‌ها هم باید پیام دریافتی را روی سایر یال‌های درخت ارسال کنند. به این ترتیب، هم کمترین تعداد ممکن پیام در شبکه جاری می‌شود و ترافیک زیادی مثل الگوریتم flooding ایجاد نمی‌کند و هم کمترین هزینه‌ی ممکن در اثر ارسال پیام‌ها ایجاد می‌شود.

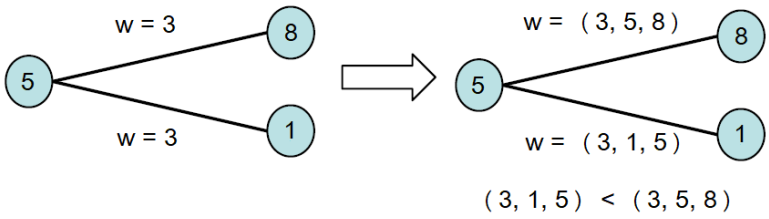
۲.۱ مقدمات الگوریتم

۱.۲.۱ ویژگی‌های درخت پوشای بهینه

یک زیردرخت از MST را به عنوان یک fragment تعریف می‌کنیم. یک یال خروجی برای این fragment، یالی از گراف اصلی است که فقط یک سر آن در این fragment قرار داشته‌باشد. حال پس از این تعاریف، به بیان ویژگی‌های MST می‌پردازیم: ویژگی اول: اگر یال e، یال خروجی با کمترین وزن در یک fragment باشد، از ترکیب کردن دو fragment دو سر یال e، یک fragment جدیدی به می‌آید. یعنی با ترکیب این دو fragment، یک component جدیدی حاصل می‌شود که بازهم زیرمجموعه‌ی MST است. اثبات این موضوع را می‌توانید در مقاله‌ی GHS [۱] بخوانید. ویژگی دوم: اگر تمامی یال‌های یک گراف همبند، وزن‌های متفاوتی داشته‌باشند، MST یکتا خواهدبود. اثبات این موضوع را می‌توانید در مقاله‌ی GHS [۱] بخوانید

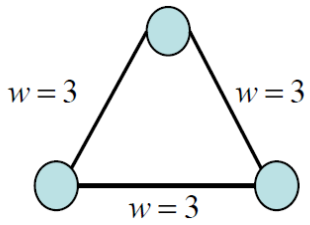
۲.۲.۱ متمایز کردن وزن یال‌ها

همان‌طور که در ویژگی دوم دیدیم، برای آنکه درخت پوشای بهینه یکتا باشد، باید وزن تمام یال‌ها متمایز باشد، به همین منظور، از آنجا که id گره‌های مختلف شبکه متمایز است، به کمک آن‌ها یال‌های شبکه را هم متمایز می‌کنیم. به این طریق که به جای استفاده از یک عدد که نمایانگر وزن یال باشد، از یک سه‌تایی مرتب برای بیان وزن یال استفاده می‌کنیم. عدد اول همان وزن یال است. اعداد دوم و سوم هم به ترتیب، id گره کوچکتر و id گره بزرگتر دو سر یال هستند. برای مقایسه‌ی وزن دو یال، ابتدا عدد اول را مقایسه کرده و در صورت برابری، سپس عدد دوم و بعد از آن عدد سوم را مقایسه می‌کنیم. به این ترتیب، وزن یال‌ها متمایز می‌شود.



شکل ۱: نحوه مشخص شدن وزن هر یال

دلیل دیگر این تمایز یال‌ها، جلوگیری از ایجاد دور در طی اجرای الگوریتم است. اگر وزن یال‌ها متمایز نباشد، هر گره به صورت تصادفی می‌تواند یکی از دو یال با وزن برابر را برای ادامه‌ی اجرای الگوریتم انتخاب کند. به دلیل deterministic نبودن انتخاب یال، امکان به وجود مشکل در الگوریتم وجود دارد. برای مثال شکل زیر را در نظر بگیرید. چون احتمال انتخاب هر کدام از دو یال متصل به یک گره، توسط آن گره به یک اندازه است، امکان دارد که هر کدام از این گره‌ها یال متصل به گره سمت راست خود را انتخاب کرده و به این طریق، هر سه یال در MST نهایی وجود داشته‌باشد که با تعریف درخت در تناقض است.

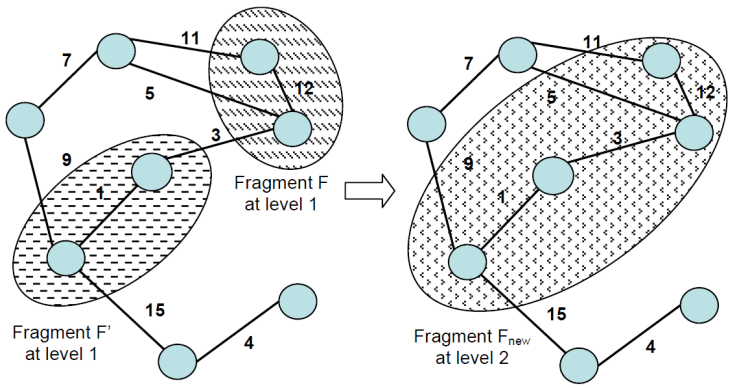


شکل ۲: ضرورت متمایز کردن وزن یال‌ها

۲ توصیف الگوریتم

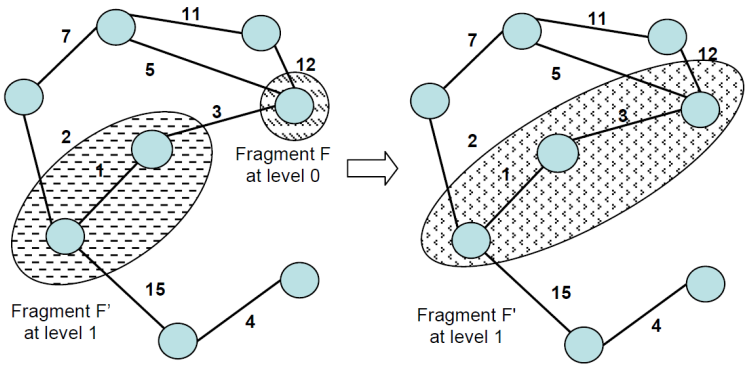
۱.۲ ایده‌های اصلی

ایده‌های الگوریتم، بر اساس دو ویژگی بیان‌شده هستند. به این ترتیب که الگوریتم با fragment N که هرکدام فقط شامل یک گره هستند، آغاز می‌شود. مطابق با ویژگی اول، ما هر fragment را در راستای یال خروجی با کمترین وزن بزرگ می‌کنیم. همچنین بر اساس ویژگی دوم می‌دانیم که چون MST نهایی یکتا است. پس fragment حاصل از ترکیب دو fragment همچنان جزئی از یک MST خواهدبود و چون تنها یک MST وجود دارد، پس این fragment جدید، همچنان جزئی از آن خواهدبود.
در این الگوریتم، هر fragment یال خروجی با کمترین وزن خود را پیدا کرده و با fragment طرف دیگر این یال ترکیب می‌شود. نوع این ترکیب به سطح (level) این fragment ها بستگی دارد. در آغاز الگوریتم، سطح همه‌ی fragment ها برابر صفر است. در هنگام ترکیب دو اتفاق ممکن است رخ دهد.
اتفاق اول: اگر دو fragment با سطح یکسان، یال خروجی با کمترین وزن یکسانی هم داشته‌باشند، به عبارت دیگر این دو fragment بخواهند با دیگری ترکیب شوند، در این حالت این دو با هم ترکیب شده و یک fragment با یک سطح بالاتر می‌سازند. به این عملیات merge، به یال خروجی با کمترین وزن مشترک آن‌ها، هسته‌ی (core) fragment جدید و به دو گره دو سر این هسته، گره‌های هسته می‌گویند.



شکل ۳: merge در الگوریتم GHS

اتفاق دوم: اگر یک fragment برای ترکیب شدن با یک fragment با سطح بالاتر تلاش بکند، یعنی سر دیگر یال خروجی با کمترین وزن آن، یک fragment با سطح بالاتر باشد، این fragment در fragment با سطح بالاتر جذب می‌شود و سطح fragment دیگر را به خود می‌گیرد. به این عملیات absorb می‌گویند.



شکل ۴: absorb در الگوریتم GHS

توجه: در سایر حالات این دو اتفاق، هیچ عملیات merge یا absorb رخ نمی‌دهد. برای مثال اگر یک fragment بخواهد در یک fragment با سطح پایین‌تر ترکیب شود، هیچ اتفاقی نمی‌افتد تا سطح fragment دیگر به سطح این fragment و یا بالاتر از آن برسد.

۲.۲ تعاریف اولیه

حالت‌های گره: هر گره می‌تواند سه حالت Sleeping ، Find و Found داشته‌باشد. حالت ،Sleeping حالت اولیه‌ی همه‌ی گره‌ها به هنگام آغاز الگوریتم است. الگوریتم با بیدار کردن یک یا چندین گره آغاز می‌شود. یعنی از خارج از محیط گره‌ها، فرمان بیدار شدن به یک یا تعدادی از گره‌ها و یا همه‌ي آن‌ها داده شده و الگوریتم آغاز می‌کنند. سایر گره‌ها با دریافت پیام از گره‌های بیدار، بیدار خواهندشد. حالت بعدی Find است که مختص زمانی است که گره به دنبال یافتن یال خروجی با کمترین وزن fragment خود است. در سایر حالات، در حالت Found است.

حالت‌های یال: هر یال می‌تواند از نظر گره متصل به آن سه حالت داشته‌باشد. Rejected که یال مربوط به MST نیست. Branch که یال جزوی از MST است. Basic که هنوز تکلیف این یال مشخص نشده‌است. دقت شود که در یک لحظه، یک یال می‌تواند از نظر دو گره دو سر آن، حالت‌های متفاوتی داشته‌باشد. اما هیچگاه با هم در تناقض نیستند.

۳.۲ پیدا کردن یال خروجی با کمترین وزن

در آغاز الگوریتم که همه‌ي گره‌ها در سطح صفر هستند و هر گره خودش به تنهایی یک fragment است، به محض بیدار شدن، ابتدا یال با کمترین وزن خودش را پیدا کرده و حالت آن را به Branch تغییر داده و سپس پیام (۰)Connect را روی این یال می‌فرستد. سپس خودش هم به حالت Found می‌رود. در واقع در این حالت می‌خواهد با fragment دیگری ترکیب شود. با توجه به ویژگی‌های اول و دوم، می‌دانیم که یال با کمترین وزن هر گره، به MST یکتا تعلق دارد.

اما هنگامی که دو fragment با سطح یکسان و غیر صفر باهم merge می‌شوند، اولا سطحشان یکی بالا رفته و یال بین دو fragment سطح قبل، به عنوان هسته‌ي fragment جدید شناخته می‌شوند که همان شناسه‌ی این fragment است. پس از ایجاد fragment جدید، این fragment به دنبال یافتن یال خروجی با کمترین وزن خواهد بود. به همین منظور گره‌های هسته (دو گره دو سر یال هسته) یک پیام initiate حاوی سطح fragment جدید، وزن هسته و حالت Find را روی شاخه‌های Branch خود می‌فرستند. سپس این پیام برای همه‌ی گره‌های fragment جدید ارسال می‌شود و بدین ترتیب، همه‌ی گره‌ها از فرایند merge رخداده مطلع می‌شوند و سطح و fragment خود را به‌روزرسانی می‌کنند.

همچنین اگر فرایند absorb هم رخ دهد، گرهی که یک fragment با سطح پایین‌تر به آن متصل شده‌است، ک پیام initiate برای آن fragment ارسال می‌کند تا اطلاعات خود را به اطلاعات fragment جدید به‌روز کنند و این پیام initiate در کل fragment جذب‌شده، منتقل می‌شود تا به گوشی همه‌ی گره‌های آن fragment برسد.

حال اگر این پیام initiate حاوی حالت Find باشد، یعنی اینکه fragment به دنبال یافتن یال خروجی با کمترین وزن است. به همین منظور، هر گره با دریافت چنین پیامی، یک پیام test روی یال با کمترین وزنی که هنوز تکلیفش مشخص نشده، می‌فرستد. یعنی روی یال Basic با کمترین وزن که این پیام حاوی سطح و fragment id است. هدف از این کار این است که این گره متوجه شود که آیا با گره همسایه‌اش در یک fragment قرار دارند یا خیر.

با دریافت پیام test توسط یک گره، اگر fragment گره فرستنده با fragment خودش یکسان باشد، پیام rejected را در جوابش ارسال می‌کند. زیرا متعلق به یک fragment هستند. اگر هم در fragment یکسانی نباشند، پیام accept را در پاسخ ارسال می‌کند. در اینجا نکته‌ای را باید در نظر گرفت. امکان دارد که گره گیرنده‌ي پیام test هنوز اطلاعاتش به‌روز نباشد. یعنی هنوز پیام initiate را از گره‌های هسته دریافت نکرده‌باشد. اما گره فرستنده‌ی پیام test به‌روز بوده و اطلاعات fragment جدید را دارد. اگر گیرنده بخواهد صرفا بر اساس مقایسه‌ي fragment ها پاسخ بدهد، در اینجا به او accept داده ولی چون در یک fragment هستند، در ادامه مشکل ایجاد می‌شود. به همین منظور، در هنگام دریافت پیام ،test ابتدا بررسی می‌کنیم که سطح گره فرستنده از سطح ما بیشتر نباشد. اگر بیشتر نبود، صرفا با مقایسه‌ی fragment می‌توان پیام accept یا reject را در پاسخ ارسال کرد. اما اگر سطح فرستنده بیشتر بود، به دلیل اینکه امکان دارد هنوز اطلاعات ما به‌روز نباشد، به آن پاسخ نمی‌دهیم و آن را به بعدا موکول می‌کنیم.

در هنگام دریافت پیام ،test یک استثنا وجود دارد. وقتی که دو گره در یک ،fragment یک پیام test برای یکدیگر ارسال می‌کنند، در این حالت پیام reject برای یکدیگر ارسال نمی‌کنند. هنگامی که دو گره متوجه می‌شوند که به یک fragment تعلق دارند و هر دو برای هم test فرستاده‌اند، بدون آنکه برای هم reject بفرستند، متوجه می‌شوند که در یک fragment قرار دارند و یک عملیات test جدید را آغاز می‌کنند.

با دریافت پیام ،reject گره گیرنده، حالت یالی که پیام را از آن دریافت کرده، به Rejected تغییر می‌دهد و سراغ یال Basic با وزن کمینه‌ي بعدی می‌رود. پس از یافتن کم‌وزن ترین یال (یا فهمیدن این موضوع که هیچ همسایه‌ی Basic ندارد)، گره منتظر دریافت report از گره‌هایی که برای آن‌ها پیام initiate فرستاده، می‌ماند. پس از دریافت همه‌ي report ها، خود این گره یک report برای گرهی که از آن پیام initiate دریافت کرده، می‌فرستد. این پیام ،report حاوی وزن کم‌وزن‌ترین یال خروجی بین یالی که خودش پیدا کرده و یال‌هایی که برای این گره report شده، است. این روند ارسال و دریافت ،report تا گره‌های هسته ادامه دارد. این دو گره هم اطلاعات خود را از طریق پیام report به یکدیگر منتقل کرده و پس از این لحظه، هر دو گره هسته، متوجه محل یال خروجی با کمترین وزن fragment خود می‌شوند.

۴.۲ تغییر هسته

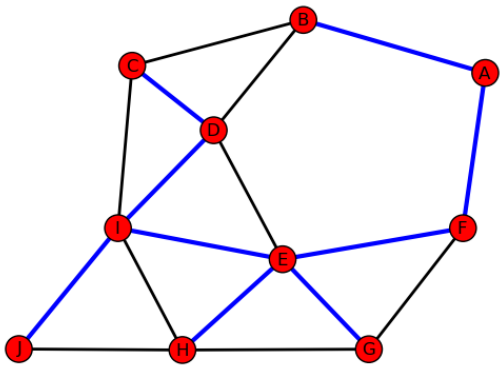
پس از اینکه هر دو گره هسته، متوجه یال خروجی با کمترین وزن شدند، آن گرهی که به آن یال نزدیکتر است، پیام core change از طریق تمامی گره‌های روی مسیر به یال خروجی با کمترین وزن ارسال می‌شود. وقتی که این پیام به گره متصل به یال خروجی با کمترین وزن می‌رسد، این گره یک پیام connect را روی این یال ارسال کرده که این پیام حاوی سطح این fragment است. از طریق این پیام، این fragment به fragment طرف دیگر این یال خبر می‌دهد که قصد ترکیب شدن با آن را دارد.

۵.۲ اتصال fragment ها

برای اتصال دو fragment چندین حالت وجود دارد:
- اگر دو fragment با سطح یکسان، یال خروجی با کمترین وزن مشترکی هم داشته‌باشند، هر کدام پیام connent را برای دیگری ارسال کرده و این یال که پیام connect روی آن ارسال می‌شود، به عنوان هسته‌ي fragment جدید انتخاب می‌شود و فرایند یافتن یال خروجی با کمترین وزن بر روی این fragment جدید آغاز می‌شود. یعنی initiate با سطح و fragment جدید و حالت Find ارسال می‌شود.
- حال اگر پیام connect از یک fragment با سطح پایین‌تر دریافت شود، گره گیرنده بدون توقف، پیام initiate را برای آن fragment ارسال کرده و عملیات absorb انجام می‌گیرد.
این پیام initiate می‌تواند دارای حالت Find یا Found باشد که بستگی به گره فرستنده‌ي آن دارد. اگر این گره تاکنون مقدار خودش را report نکرده‌باشد، یعنی در حالت Find باشد، این پیام initiate باعث می‌شود که fragment جذب‌شده، وارد عملیات جست‌وجو برای یال خروجی با کمترین وزن شود.
اما اگر این گره report خود را فرستاده‌باشد، یعنی حالت آن Found باشد، امکان ندارد که یال خروجی با کمترین وزن fragment جدید، از fragment جذب‌شده باشد. به همین دلیل نیازی نیست که این fragment جذب‌شده وارد عملیات جست‌وجو شود. زیرا اولا خود این یال بین این دو fragment، یعنی آن یالی که fragment جذب‌شده روی آن connect فرستاده، نمی‌تواند یال خروجی با کمترین وزن باشد. زیرا fragment سمت مقابل آن، دارای سطح کمتری بوده و تا بالا رفتن سطحش به پیام test پاسخ نمی‌دهد. ولی این گره، مقدار خودش را report کرده، پس این یال نمی‌تواند کمترین وزن را داشته‌باشد. ثانیا هیچکدام از یال‌های خروجی fragment جذب‌شده هم نمی‌تواند به عنوان یال خروجی با کمترین وزن fragment جدید باشد. زیرا یالی که پیام connect روی آن ارسال شده، کمترین وزن را در آن fragment داشته و چون این یال نمی‌توانست کمترین وزن را در fragment جدید داشته‌باشد، پس سایر یال‌های خروجی fragment جذب‌شده هم نمی‌تواند کمترین وزن را داشته‌باشد.

۶.۲ درستی الگوریتم

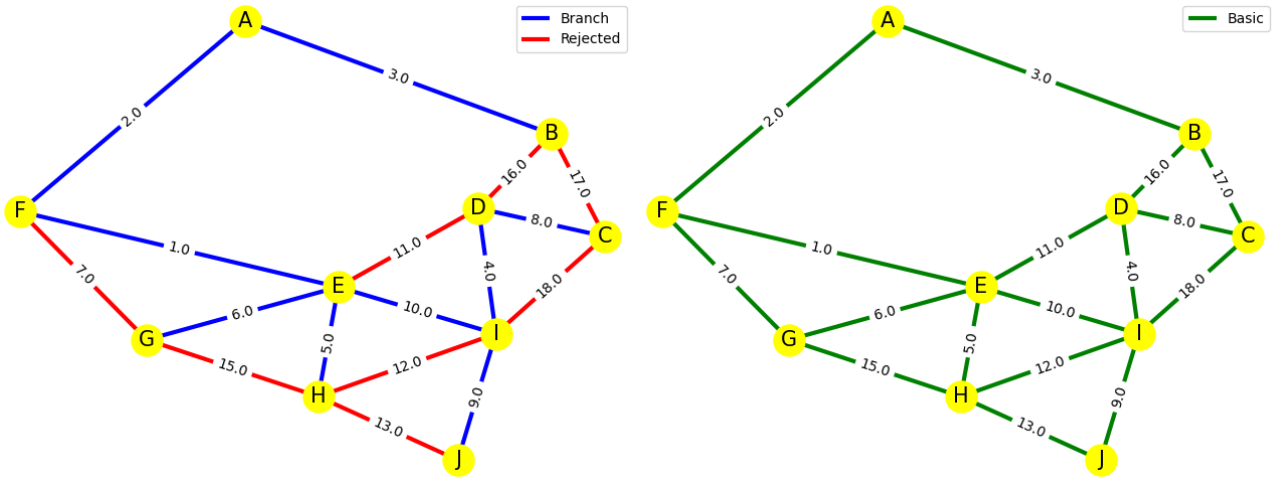
متناسب با توضیحات داده‌شده، کد الگوریتم نوشته شده‌است. برای بررسی درستی عملکرد کد، از چندین مثال آماده استفاده کرده‌ایم.
مثال اول: گراف این مثال و یال‌های MST با رنگ آبی نشان داده شده‌است.



شکل ۵: گراف مثال اول

با اجرای الگوریتم روی این گراف به نتایج زیر رسیدیم که با شکل بالا همخوانی دارد. (این نتایج پس از اجرا در فایل GHS.log در پوشه‌ی result قابل مشاهده است.)

number of nodes = 10
number of edges = 17
number of levels = 2

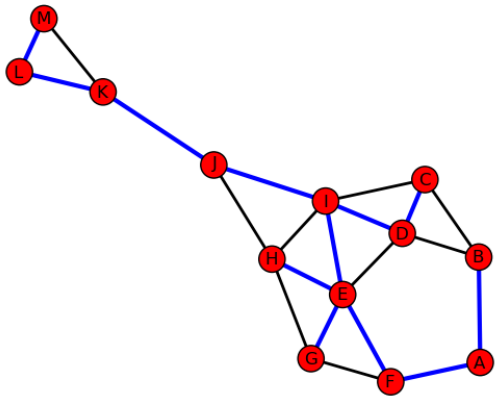


شکل ۶: اجرای الگوریتم بر روی مثال اول GHS در شبیه‌ساز

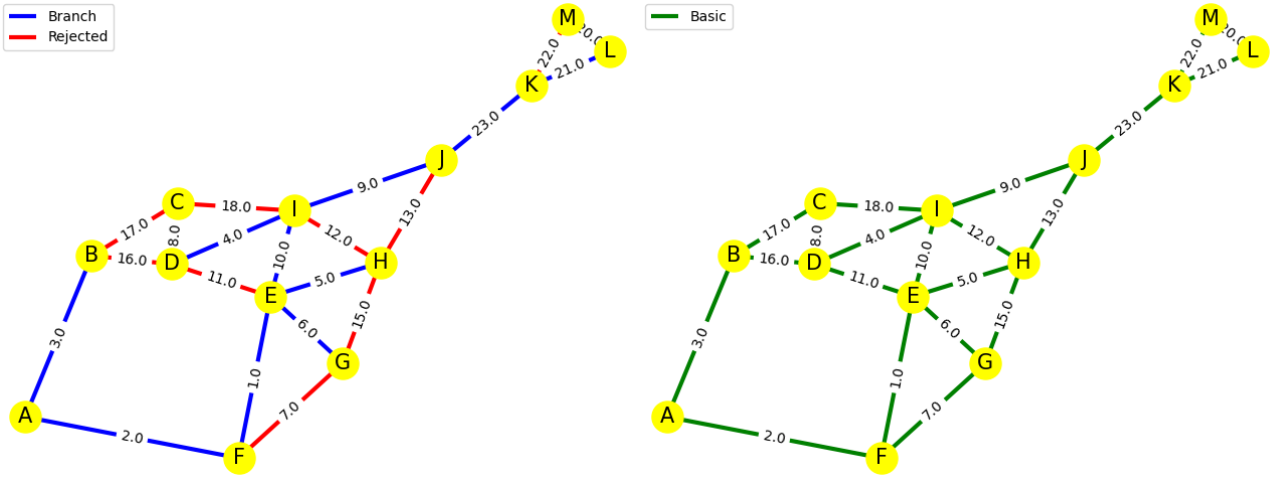
total time of execution = 11.06
average time of execution per level = 5.53
total number of messages = 92
average number of messages per node = 9.20
average number of messages per level = 46.00
total number of bits = 5932
average number of bits per node = 593.20
average number of bits per message = 64.48
branches:

(A,B)
(A,F)
(C,D)
(D,I)
(E,F)
(E,G)
(E,H)
(E,I)
(I,J)

دو تصویر شکل ۶، تصویر گراف اولیه و گراف نهایی است که یال‌های درخت روی آن مشخص شده‌است. مثال دوم: گراف این مثال و یال‌های MST با رنگ آبی نشان داده شده‌است.



شکل ۷: گراف مثال دوم

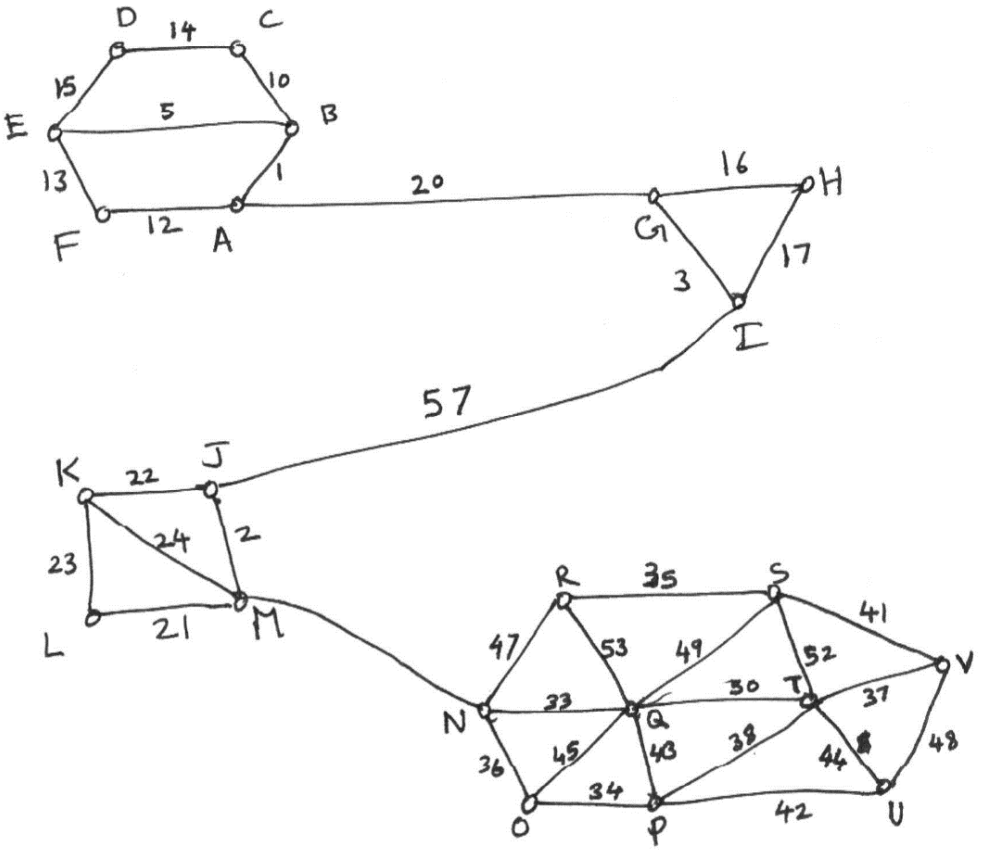


شکل ۸: اجرای الگوریتم بر روی مثال دوم GHS در شبیه‌ساز

با اجرای الگوریتم روی این گراف به نتایج زیر رسیدیم که با شکل بالا همخوانی دارد.

number of nodes = 13
number of edges = 21
number of levels = 2
total time of execution = 17.60
average time of execution per level = 8.80
total number of messages = 115
average number of messages per node = 8.85
average number of messages per level = 57.50
total number of bits = 7585
average number of bits per node = 583.46
average number of bits per message = 65.96
branches:
(A,B)
(A,F)
(C,D)
(D,I)
(E,F)
(E,G)
(E,H)
(E,I)
(I,J)
(J,K)
(K,L)
(L,M)

دو تصویر شکل ۸، تصویر گراف اولیه و گراف نهایی است که یال‌های درخت روی آن مشخص شده‌است.
مثال سوم: گراف این مثال در شکل زیر نشان داده شده‌است.



شکل ۹: گراف مثال سوم

نتیجه‌ی این مثال به شکل زیر است:

Solution: (A, B), (A, F), (A, G), (B, C), (B, E), (C, D), (G, H), (G, I), (I, J), (J, K), (J, M), (L, M), (M, N), (N, O), (N, Q), (O, P), (P, T), (P, U), (R, S), (S, V), (T, V)

با اجرای الگوریتم روی این گراف به نتایج زیر رسیدیم که با شکل بالا همخوانی دارد.

number of nodes = 22
number of edges = 34
number of levels = 3
total time of execution = 65.38
average time of execution per level = 21.79
total number of messages = 240
average number of messages per node = 10.91
average number of messages per level = 80.00
total number of bits = 16568
average number of bits per node = 753.09
average number of bits per message = 69.03
branches:

(A,B)
(A,F)
(A,G)
(B,C)
(B,E)
(C,D)
(G,H)
(G,I)

(I,J)
(J,K)
(J,M)
(L,M)
(M,N)
(N,O)
(N,Q)
(O,P)
(P,T)
(P,U)
(R,S)
(S,V)
(T,V)

به علت بزرگ بودن گراف، از نمایش آن خودداری کردیم.

۷.۲ پیچیدگی زمانی، تعداد و طول پیام الگوریتم GHS

۱.۷.۲ پیچیدگی تعداد پیام

در الگوریتم GHS حداکثر تعداد پیام‌ها با رابطه‌ی زیر باند می‌شود:

$$Messages\ Upper\ Bound : 4|E| + 5nlogn$$

- تعداد $4|E|$ پیام از نوع test و reject :این دو نوع پیام در هر یال حداکثر دوبار در دو جهت فرستاده می‌شوند.
- تعداد n پیام از نوع initiate در هر سطح broadcast) شده بر روی درخت هر (component
- تعداد n پیام از نوع report در پاسخ به initiate (convergecast)
- تعداد $2n$ پیام از نوع test و accept در هر سطح
- تعداد n پیام از نوع changeroot و connect در هر سطح

با توجه به این‌که تعداد سطح‌ها حداکثر $\log n$ تا است؛ رابطه ذکر شده همیشه برقرار است.

۲.۷.۲ پیچیدگی زمانی

پیچیدگی زمان اجرای هر سطح الگوریتم GHS به صورت $O(n(l+d))$ است که n تعداد گره‌ها و l و d به ترتیب باند بالای تاخیر ارسال لینک‌ها و یال‌ها هستند. در نتیجه، پیچیدگی زمانی کلی به صورت $O(n \log n(l+d))$ خواهد بود.

۳.۷.۲ پیچیدگی طول پیام

در الگوریتم GHS انواع مختلفی از پیام‌ها وجود دارند که شامل چندتا از بخش‌های زیر هستند. تعداد بیت‌های لازم برای هر بخش پیام به صورت زیر است

- ۱۶ بیت برای id هر گره
- ۳ بیت برای نوع پیام (test, initiate, report, ...)
- ۴ بیت برای مشخص کردن سطح ($level < \log_2 16 = 4$)
- ۳۲ + ۲×۱۶ بیت برای مشخص کردن هسته (۳۲ بیت برای وزن یال از نوع float و ۲ × ۱۶ بیت برای گره‌های دو طرف آن)
- ۲ بیت برای مشخص کردن وضعیت یال‌ها (rejected, branch, basic)

۳ آزمایش‌های انجام شده

در این بخش آزمایش‌های انجام شده و نحوه شبیه‌سازی آن‌ها در برنامه را شرح داده و نحوه اندازه‌گیری پارامترهای خروجی مانند تعداد پیام‌ها و زمان اجرا را بررسی می‌کنیم.

۱.۳ قابلیت‌های برنامه شبیه‌ساز

برای بررسی نحوه عملکرد الگوریتم GHS لازم است حالت‌های مختلف ممکن برای قرارگیری گره‌ها در شبکه، قرارگیری یال‌های بین آن‌ها و وزن و تاخیر یال‌ها در نظر گرفته شده و به ازای تمامی حالت‌ها بررسی‌ها انجام شود تا نتایج قابل استناد و تعمیم باشند. به این منظور در برنامه شبیه‌ساز ۴ گزینه متفاوت برای توپولوژی شبکه در نظر گرفته شده و تعداد گره‌ها و یال‌ها نیز قابل تنظیم است:

- **گراف دلخواه**: تعداد گره‌ها و یال‌ها به همراه وزن و تاخیر هر یال به عنوان ورودی از کاربر دریافت می‌شود. این کار تا زمانی که یک گراف متصل (هم‌بند) تشکیل شود، ادامه می‌یابد.
- **گراف با توپولوژی حلقه**: در واقع گرافی با کمترین یال ممکن برای اجرای الگوریتم GHS است و تعداد گره‌ها به عنوان ورودی دریافت می‌شود.
- **گراف کامل**: همه گره‌ها از طریق یال‌هایی با وزن‌های تصادفی با یک‌دیگر ارتباط دارند و تعداد گره‌ها به عنوان ورودی دریافت می‌شود.
- **گراف تصادفی:** تعداد گره‌ها و یال‌ها به عنوان ورودی دریافت می‌شود و یال‌ها به صورت کاملا تصادفی بین گره‌ها با وزن‌های تصادفی قرار می‌گیرد. البته شرط هم‌بند بودن گراف نیز بررسی می‌شود و در صورت ناهم‌بند بودن گراف، با استفاده از الگوریتم DFS ناحیه‌های جداازهم شناخته شده و یال جدیدی بین آن‌ها برقرار می‌شود تا شبکه هم‌بند باشد.

۲.۳ بررسی صحت عملکرد شبیه‌ساز

به منظور سنجش اعتبار برنامه شبیه‌ساز طراحی شده، شبکه‌هایی با ساختار و وزن‌ها و تاخیرهای شبکه مشخص بر روی شبیه‌ساز اجرا شده و نتایج به دست آمده با نتیجه موردانتظار مقایسه می‌شود.

۳.۳ سناریوهای مورد بررسی

در ادامه با استفاده از شبیه‌ساز قصد داریم تاثیر تعداد گره‌ها، تعداد یال‌ها، توپولوژی شبکه و تاخیر شبکه را بر روی نحوه عملکرد الگوریتم GHS بررسی کنیم. به این منظور ۴ سناریو متفاوت برای ارزیابی الگوریتم طراحی شده‌اند:

۱. **اثر تعداد یال‌ها؛ تعداد گره ثابت**
در این حالت، تعداد گره‌ها ثابت باقی می‌ماند و به ازای تعداد یال‌های متفاوت عملکرد الگوریتم بررسی می‌شود. به این منظور به ازای تعداد گره n تعداد یال‌ها از n (توپولوژی حلقه) تا $\frac{n(n-1)}{2}$ (گراف کامل) تغییر می‌کند.

۲. **اثر تعداد گره‌ها؛ تعداد یال ثابت**
در این حالت، تعداد یال‌ها ثابت باقی می‌ماند و به ازای تعداد گره‌های متفاوت عملکرد الگوریتم بررسی می‌شود. به این منظور به ازای تعداد یال $|E|$ تعداد گره‌ها به گونه‌ای تغییر می‌کند که گراف کامل تا گراف با توپولوژی حلقه که شامل $|E|$ است، تحت پوشش قرار می‌گیرد.

۳. **گراف کامل**
به ازای تعداد گره‌ها متغیر، یک شبکه کامل تشکیل داده شده و الگوریتم مورد بررسی قرار می‌گیرد. هم‌چنین وزن یال‌ها به صورت تصادفی انتخاب شده و باند بالایی برای تاخیر شبکه در نظر گرفته می‌شود.

۴. **توپولوژی حلقه**
این توپولوژی نیز به عنوان یک حالت خاص بر روی الگوریتم به ازای تعداد گره‌های متغیر اجرا می‌شود و پارامترهای به دست آمده مورد بررسی قرار می‌گیرند.

۴.۳ پارامترهای مورد بررسی

سناریوهای ذکر شده در بخش قبل، به صورت جداگانه اجرا شده و پارامترهای زیر در هر حالت بر حسب متغیرها مورد بررسی قرار می‌گیرند:

- تعداد پیام‌ها در شبکه
- زمان اجرا
- حداکثر تعداد سطح‌ها (level) حین اجرا
- تعداد بیت‌های ارسالی
- متوسط تعداد پیام‌های ارسالی هر گره
- متوسط تعداد پیام‌های ارسالی در هر سطح
- متوسط زمان هر سطح
- متوسط طول پیام‌ها

۴ نتایج آزمایش

۱.۴ اثر تعداد یال‌ها؛ تعداد گره ثابت

در این آزمایش، تعداد گره‌ها عدد ثابت ۲۰ در نظر گرفته شده و به ازای تعداد یال‌های متفاوت عملکرد الگوریتم بررسی می‌شود. به این منظور تعداد یال‌ها از ۲۰ تا ۱۹۰ یال تغییر می‌کند.

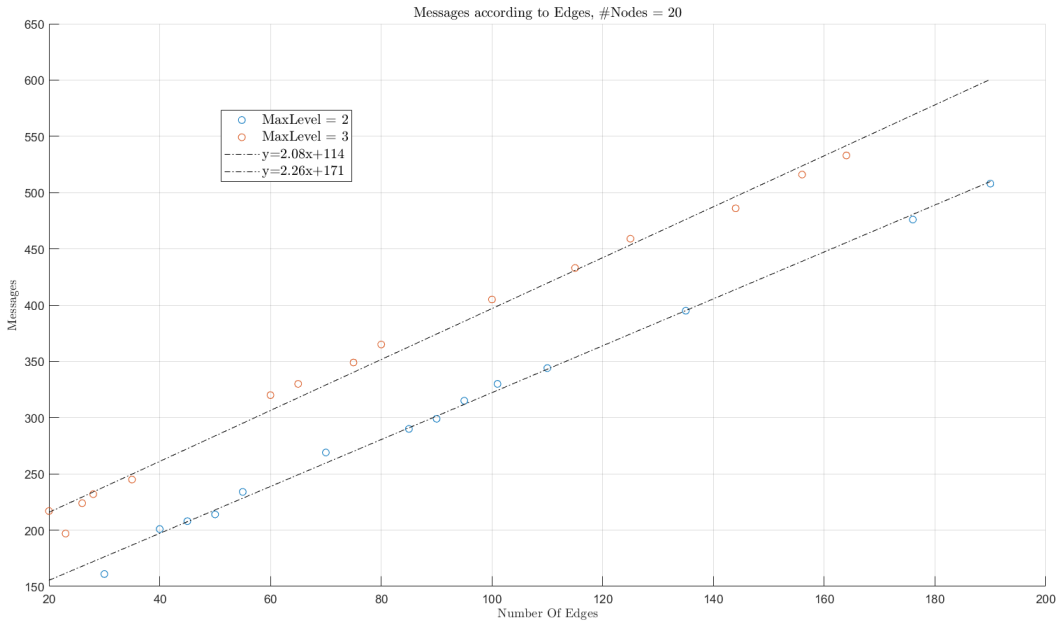
۱.۱.۴ بررسی پیچیدگی پیام

در الگوریتم GHS پیچیدگی تعداد پیام‌های ارسالی بر حسب تعداد گره و یال به صورت $O(|E| + nlogn)$ است. حال با ثابت بودن تعداد گره‌ها انتظار می‌رود که تعداد پیام‌ها تابعیت خطی از تعداد یال‌ها داشته باشد. همان‌طور که در شکل ۱۰ ملاحظه می‌شود، به طور کلی با افزایش تعداد یال‌ها، تعداد یال‌ها نیز به صورت خطی با شیب ثابتی افزایش می‌یابد.

البته تعداد پیام‌های ارسال شده به تعداد سطح‌های اجرای الگوریتم نیز وابسته است. تعداد سطح‌ها همان‌طور که انتظار می‌رفت در محدوده $O(logn)$ قرار گرفته است. با افزایش تعداد سطح‌ها، در واقع تعداد حداقل $2n$ پیام initiate و report در هر سطح از اجرای الگوریتم به کل تعداد پیام‌ها افزوده می‌شود و به همین خاطر به ازای تعداد سطح بیشتر، یک بایاس به نمودار پیام مربوط به آن سطح متناسب با تعداد سطح‌ها افزوده می‌شود.

خط‌های درجه اول به هر دسته از داده‌ها طبق معیار mse منطبق شده است که معادله دو خط به دست آمده به صورت زیر است:

$$M_1 = 2.08|E| + 114$$
$$M_2 = 2.26|E| + 171$$



شکل ۱۰: تعداد پیام‌ها بر حسب تعداد یال در شبکه به ازای $n = 20$

حال تاثیر تعداد سطح‌ها را بررسی می‌کنیم و یک رابطه برای تعداد پیام (M) بر حسب تعداد گره‌ها (N)، تعداد یال‌ها ($|E|$) و تعداد سطح‌ها (L) به دست می‌آوریم. طبق رابطه‌ی پیچیدگی پیام می‌توانیم این رابطه را به صورت زیر بنویسیم:

$$M = a_M NL + b_M|E|$$

حال با استفاده از معیار کمینه mse ضرایب a_M و b_M را به دست می‌آوریم؛ ضرایب به دست آمده به صورت زیر است:

$$a_M = 2.96, b_M = 2.12, \frac{\sigma}{\bar{x}} = 3.59\%$$

تعداد پیام‌های ارسال شده از باند بالای تعداد پیام‌ها که به صورت

$$M_{max} = 5NL + 4|E|$$

است کمتر است و باند بالا به درستی در نظر گرفته شده است. هم‌چنین ضریب مربوط به تعداد یال‌ها ۴۷ درصد و ضریب مربوط به گره‌ها ۴۰ درصد پایین‌تر از ضریب حد بالای آن‌ها است؛ یعنی در حالت کلی و به صورت میانگین، تعداد پیام‌ها به ازای شبکه‌های مختلف، تقریبا نصف باند بالای در نظر گرفته شده است. هم‌چنین دقت ضرایب به دست آمده بالا بوده و نسبت انحراف معیار به میانگین برای تابع به دست آمده برابر ۵۹.۳ درصد است.

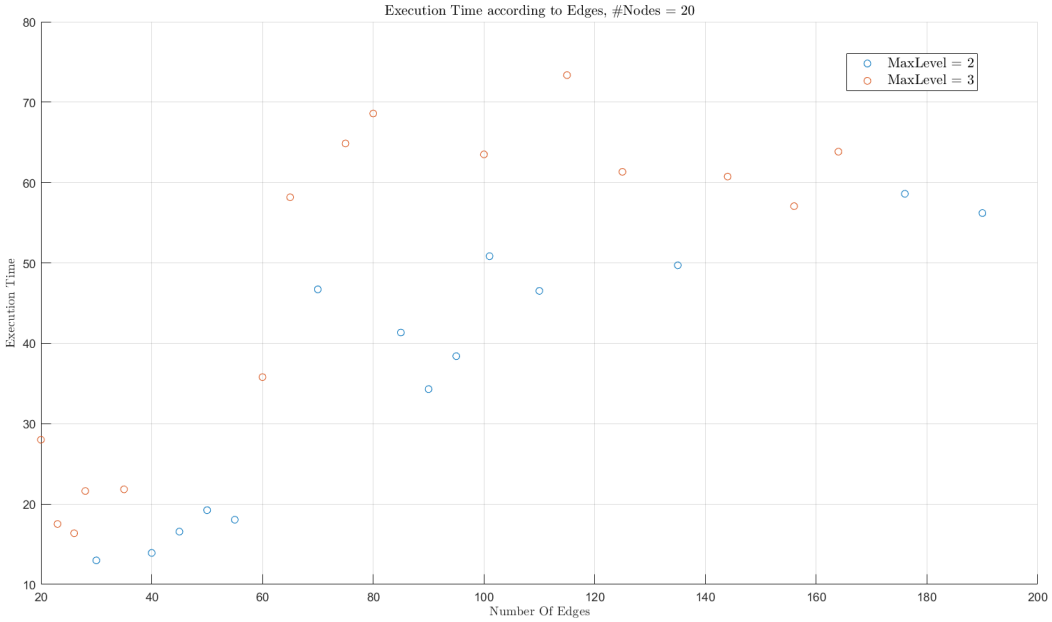
۲.۱.۴ بررسی پیچیدگی زمانی

در الگوریتم GHS پیچیدگی زمانی بر حسب تعداد گره‌ها به صورت $O(nlogn(l+d))$ است. هم‌چنین زمان اجرا برای هر سطح با عبارت $O(n(l+d))$ باند می‌شود. با ثابت بودن تعداد گره‌ها انتظار می‌رود که زمان اجرا ثابت باشد؛ زیرا در رابطه تابعیتی از تعداد یال‌ها وجود ندارد. با این حال، با توجه به رندوم بودن شبکه ممکن است تعداد سطح‌ها در هر مرحله از اجرا متفاوت باشد و به وضوح زمان اجرا با تعداد سطح‌ها رابطه‌ی مستقیم دارد. همان‌طور که در شکل ۱۱ ملاحظه می‌شود، به ازای تعداد نود ۲۰، ممکن است تعداد سطح‌ها در برخی از گراف‌ها برابر ۲ و در برخی برابر ۳ باشد. با زیاد شدن تعداد سطح‌ها ملاحظه می‌شود که زمان اجرا نیز افزایش می‌یابد. در واقع می‌توان زمان اجرا را به صورت

$$T = a_T NL + b_T$$

در نظر گرفت و وابستگی زمان اجرا به تعداد سطح‌ها را بررسی کرد. ضریب $a_T = 0.57$ به دست می‌آید و در واقع با افزایش یک سطح در حین اجرا، به طور متوسط زمان اجرا $0.57 \times 20 = 11.4s$ افزایش می‌یابد.

نکته قابل ملاحظه دیگر آن است که تعداد یال‌ها تاثیری بر تعداد سطح‌های اجراهای الگوریتم GHS ندارد و عموما به تعداد گره‌های شبکه وابسته است و با رابطه $O(logn)$ باند می‌شود. هم‌چنین در شکل ۱۱ ملاحظه می‌شود که زمان اجرا نوسان بالایی در هر اجرا دارد که این پدیده در واقع به دلیل عملکرد پردازنده کامپیوتر شخصی است که برنامه بر روی آن اجرا می‌شود. در واقع، به نظر با افزایش یال‌ها زمان اجرا در حال افزایش است؛ این پدیده می‌تواند ناشی از پیچیده شدن گراف شبکه و تقسیم بیشتر توان پردازنده بین هر ترد برنامه باشد. در نتیجه، زمان‌های اجرا نوسان بالایی دارند و نمی‌توان نتیجه‌گیری دقیقی از آن داشت.



شکل ۱۱: زمان اجرا بر حسب تعداد یال در شبکه به ازای $n = 20$

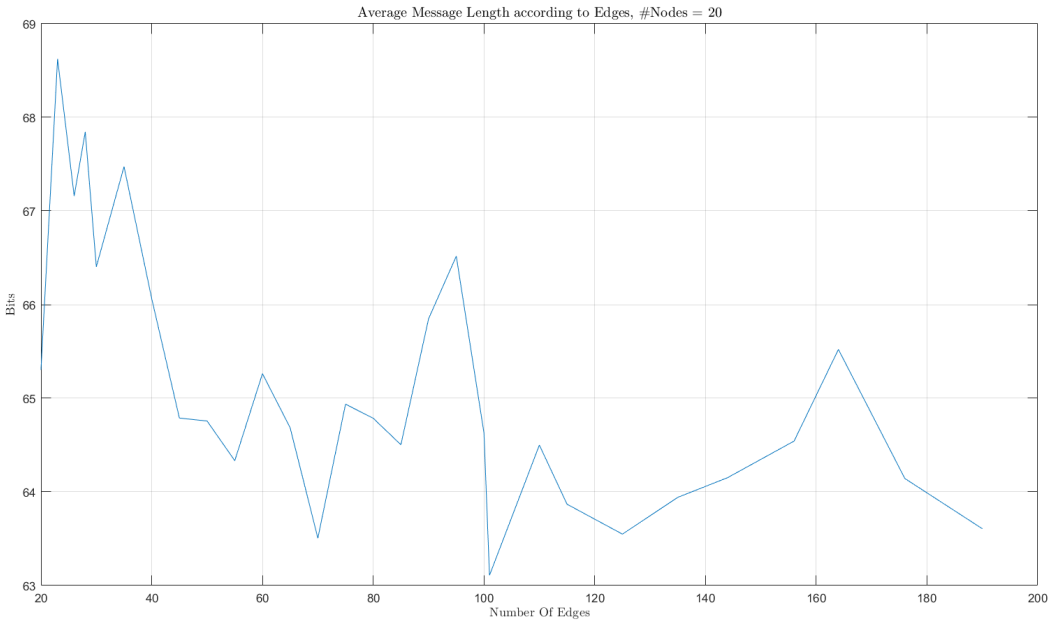
اگر مانند رابطه‌ی در نظر گرفته شده برای تعداد پیام‌ها، رابطه زمان اجرا به صورت $T = a_T NL + b_T|E|$ در نظر گرفته شود، ضرایب تخمین زده شده عبارتند از:

$$a_T = 0.364, b_T = 0.283, \frac{\sigma}{\bar{x}} = 26.3\%$$

دقت تابع برازش شده پایین است و نمی‌تواند به خوبی وابستگی زمان اجرا را تخمین بزند.

۳.۱.۴ بررسی پیچیدگی طول پیام

هنگام اجرای الگوریتم GHS در هر سطح همه انواع پیام‌ها، از جمله initiate ، report ، test ، accept ، connect ، changeroot رد و بدل می‌شوند. همان‌طور که گفته شد، میانگین طول پیام‌ها برابر ۶۶ بیت است. در اجراهای مختلف میانگین طول پیام بر حسب تعداد یال‌ها به دست آمده است. میانگین طول پیام در هر اجرا تقریبا روند ثابتی دارد. میانگین طول پیام‌ها در کل برابر 65.11 است و انحراف معیار این مقادیر برابر 1.38 است که نشان از عدم تغییر چندان طول پیام‌ها با تغییر گراف شبکه و سایر پارامترهای آن دارد. هم‌چنین تعداد کل بیت‌های ارسالی نیز، به طبع روندی مشابه تعداد پیام‌ها (شکل ۱۰) خواهد داشت.



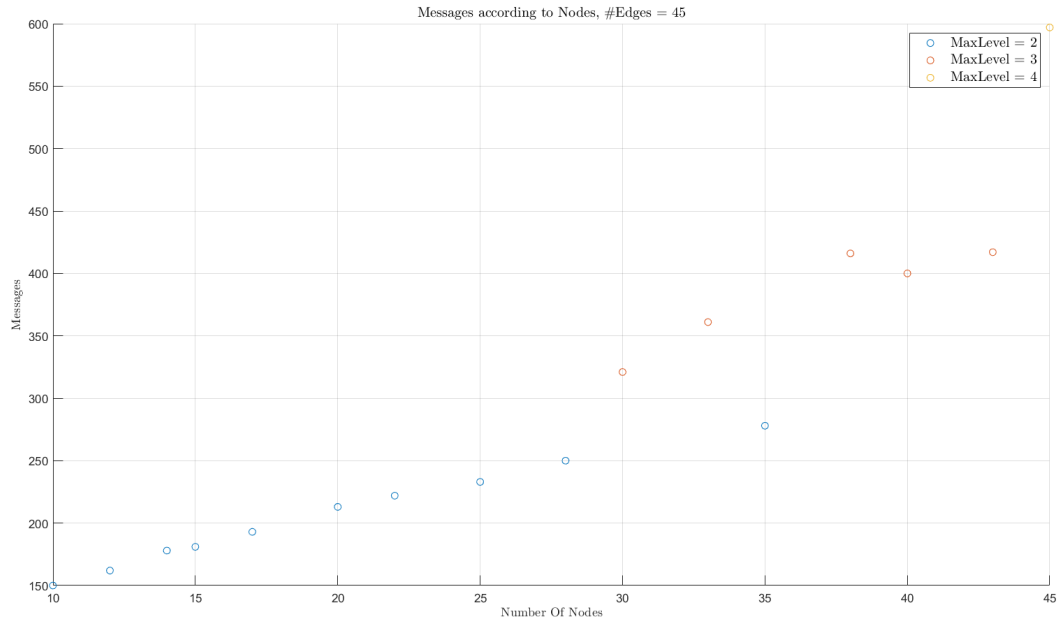
شکل ۱۲: میانگین طول پیام‌ها بر حسب تعداد یال در شبکه به ازای $n = 20$

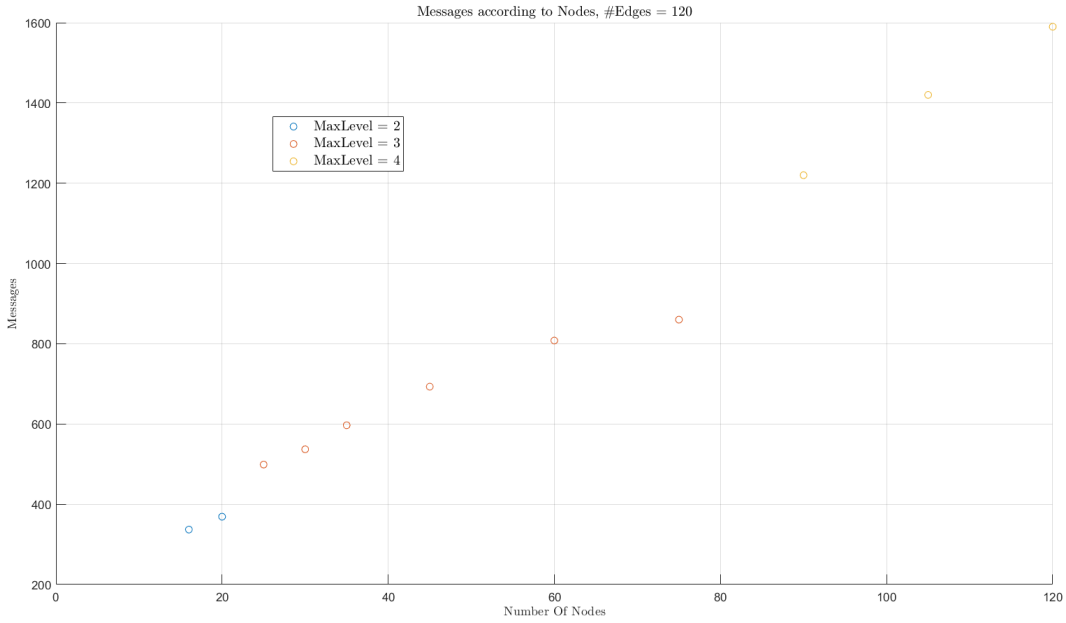
۲.۴ اثر تعداد گره‌ها؛ تعداد یال ثابت

در این آزمایش، تعداد یال‌ها عدد ثابت ۴۵ و ۱۲۰ در نظر گرفته شده و به ازای تعداد گره‌های متفاوت عملکرد الگوریتم بررسی می‌شود. به این منظور تعداد گره‌ها به ترتیب در دو آزمایش از ۱۰ تا ۴۵ و از ۱۶ تا ۱۲۰ گره تغییر می‌کند.

۱.۲.۴ بررسی پیچیدگی پیام

در الگوریتم GHS پیچیدگی تعداد پیام‌های ارسالی بر حسب تعداد گره و یال به صورت $O(|E| + nlogn)$ است. حال با ثابت بودن تعداد یال‌ها انتظار می‌رود که تعداد پیام‌ها تابعیت خطی از حاصل‌ضرب تعداد گره‌ها در تعداد سطح هنگام اجرای الگوریتم داشته باشد. (ترم $O(nlogn)$) همان‌طور که در شکل ۱۳ ملاحظه می‌شود، به طور کلی دو پارامتر تعداد گره‌ها و تعداد سطح‌ها، در تعداد پیام‌های ارسال شده نقش دارند. به ازای تعداد سطح ثابت، طبق رابطه تعداد پیام‌ها تابعیت خطی از تعداد گره‌ها را خواهد داشت. هم‌چنین با افزایش تعداد سطح‌ها، هم‌چنان تعداد پیام‌ها به صورت خطی و با شیب بزرگتری نسبت به تعداد گره‌ها تغییر می‌کند.





شکل ۱۳: تعداد پیام‌ها بر حسب تعداد گره در شبکه به ازای $|E| = 45$ و $|E| = 120$

طبق رابطه‌ی پیچیدگی پیام می‌توانیم این رابطه را به صورت زیر بنویسیم:

$$M = a_M NL + b_M |E|$$

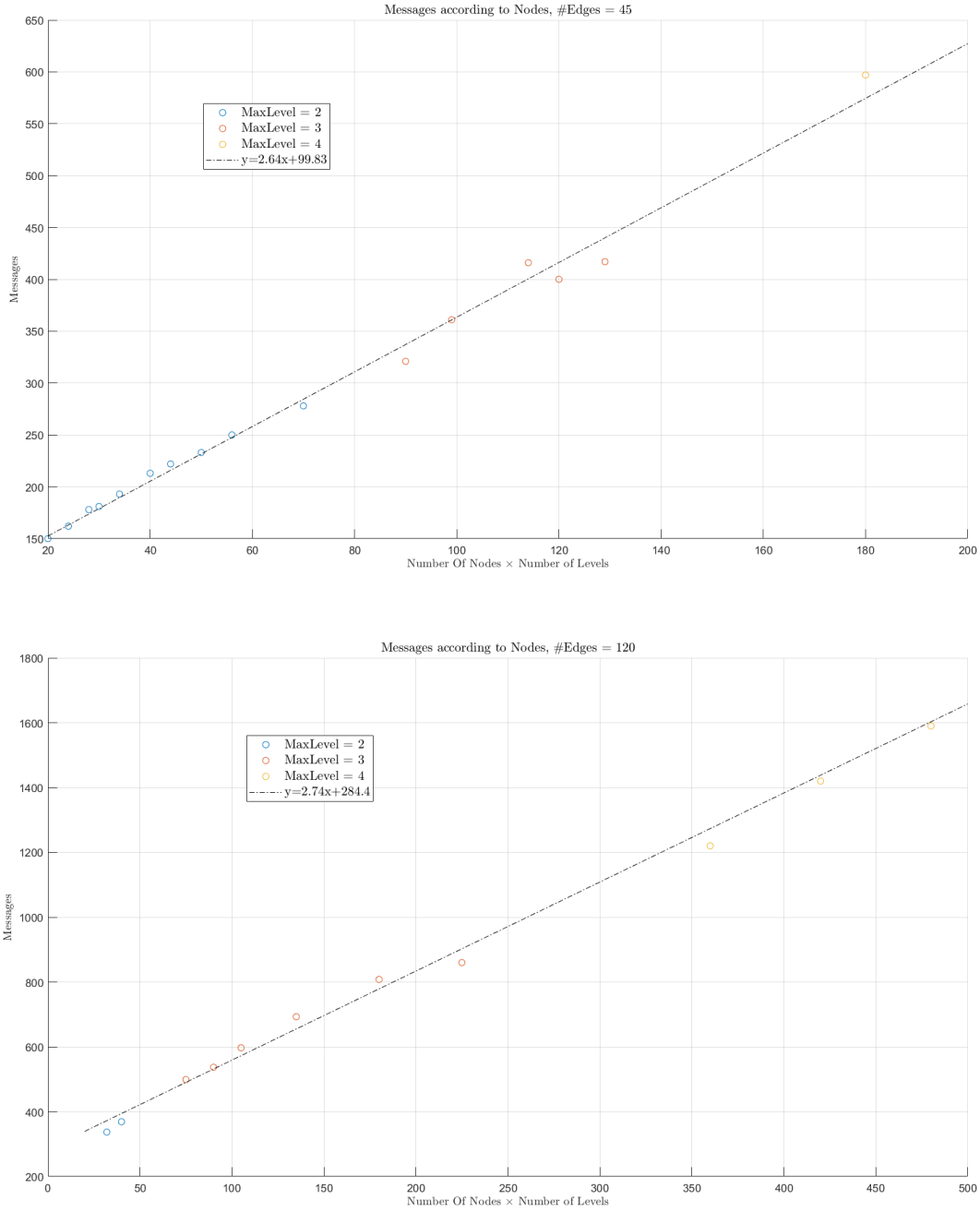
حال با استفاده از معیار کمینه mse ضرایب a_M و b_M را برای حالت $|E| = 45$ به دست می‌آوریم؛ ضرایب به دست آمده به صورت زیر است:

$$a_M = 2.64, b_M = 2.21$$

هم‌چنین برای حالت $|E| = 120$ ضرایب به دست آمده به صورت زیر است:

$$a_M = 2.74, b_M = 2.37$$

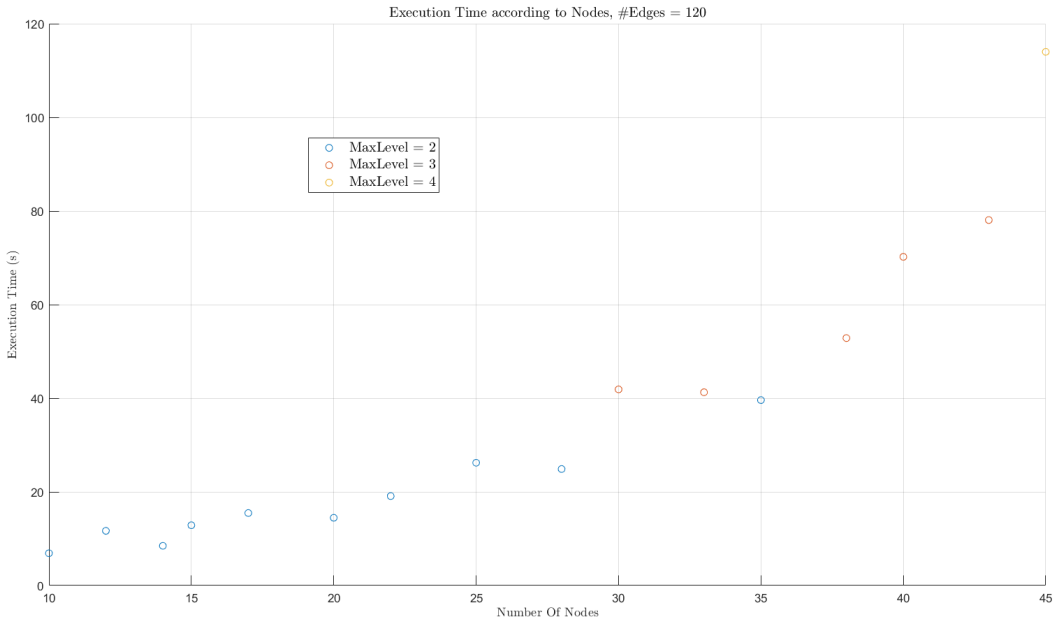
همان‌طور که در شکل ۱۴ مشاهده می‌شود، دو خط برازش شده با دقت خوبی داده‌ها را نمایش می‌دهند. ضرایب مانند نتیجه به دست در بخش۱.۱.۴ کوچک‌تر از باند بالای در نظر گرفته شده هستند و حدود ۴۰ درصد پایین‌تر از حد بالا بر حسب تعداد یال و تعداد نودها به دست آمده‌اند.



شکل ۱۴: تعداد پیام‌ها بر حسب تعداد گره در تعداد سطح (ترم $O(nlogn)$) در شبکه به ازای $|E| = 45$ و $|E| = 120$

۲.۲.۴ بررسی پیچیدگی زمانی

در الگوریتم GHS پیچیدگی زمانی بر حسب تعداد گره‌ها به صورت $O(nlogn(l+d))$ است. با ثابت بودن تعداد یال‌ها و افزایش تعداد گره‌ها انتظار می‌رود که زمان اجرا به صورت خطی در صورت ثابت بودن تعداد سطح‌ها افزایش یابد. با این حال، با توجه به رندوم بودن شبکه ممکن است تعداد سطح‌ها در هر مرحله از اجرا متفاوت باشد. تغییرات زمان اجرا بر حسب تعداد گره‌ها به ازای $|E| = 120$ در تصویر ۱۵ قابل مشاهده است.



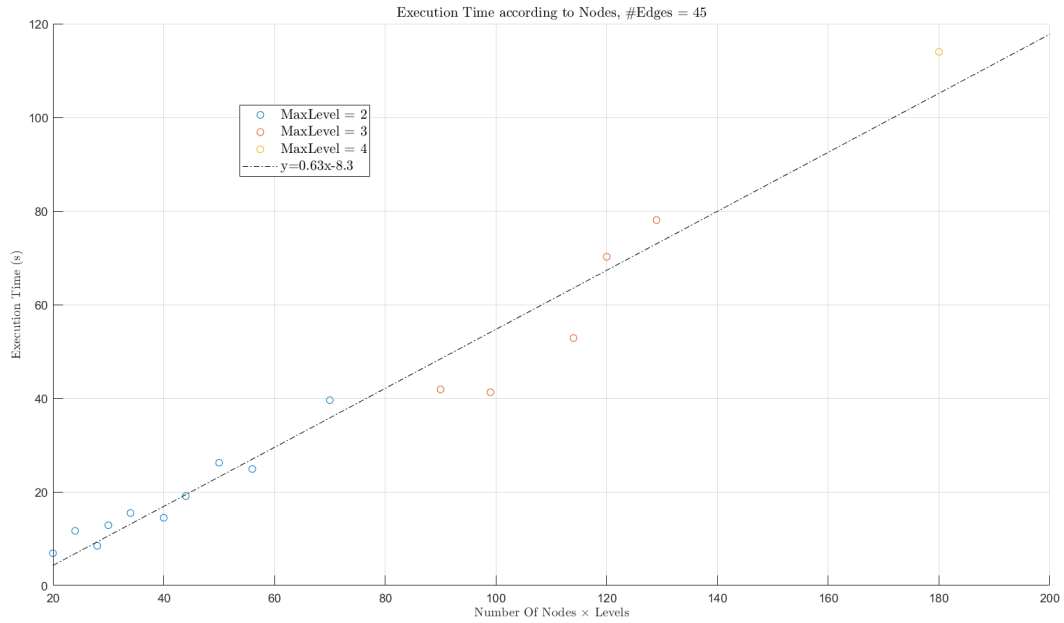
شکل ۱۵: زمان اجرا بر حسب تعداد گره در شبکه به ازای $|E| = 120$

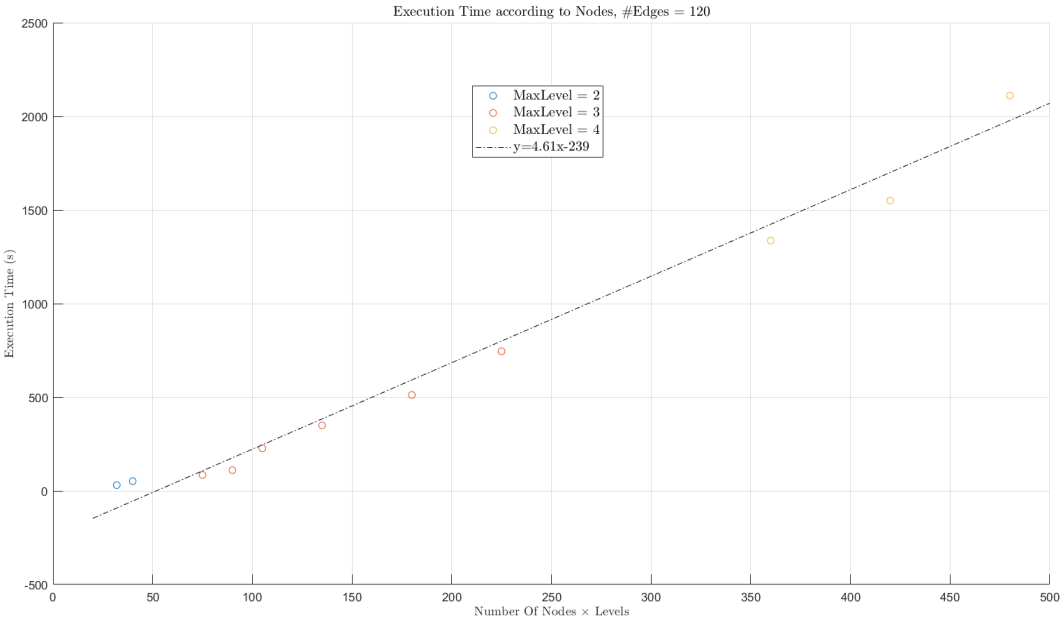
همان‌طور که در شکل ۱۶ ملاحظه می‌شود، به ازای تعداد گره‌های متغیر، زمان اجرا تقریبا تابعیت خطی از تعداد گره‌ها دارد. سطح‌ها در برخی از گراف‌ها برابر ۲ و در برخی برابر ۳ و ۴ می‌باشد. با زیاد شدن تعداد سطح‌ها ملاحظه می‌شود که زمان اجرا نیز افزایش می‌یابد. در واقع می‌توان زمان اجرا را به صورت

$$T = a_T NL + b_T$$

در نظر گرفت و وابستگی زمان اجرا به تعداد سطح‌ها را بررسی کرد. ضریب a_T به ازای دو حالت $|E| = 45$ و $|E| = 120$ به ترتیب برابر $a_T = 0.63$ و $a_T = 4.61$ به دست می‌آید و که در واقع نشان می‌دهد با افزایش گره‌ها، به طور متوسط زمان اجرا چقدر افزایش می‌یابد.

نکته قابل ملاحظه دیگر آن است که این ضریب در دو آزمایش تفاوت زیاد و غیرقابل توجیهی از نظر شبکه دارند. این تغییر زمان زیاد در واقع به دلیل تقسیم توان بین گره‌ها هنگام اجرای الگوریتم رخ داده و به ازای تعداد گره‌های بالا، شبیه‌سازی بیش از حد کند می‌شود.

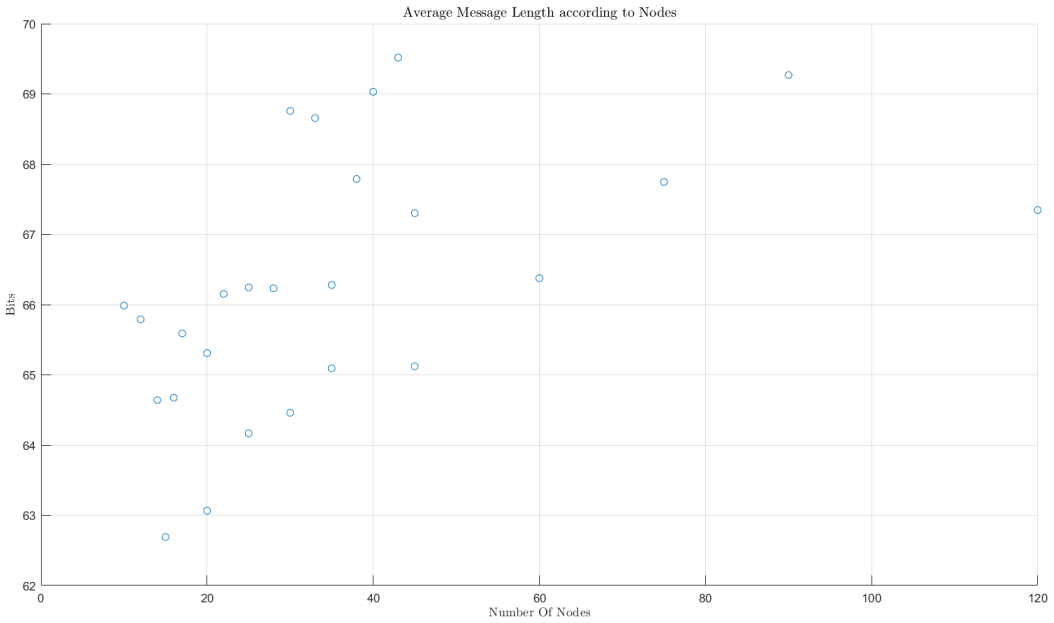




شکل ۱۶: زمان اجرا بر حسب تعداد گره در شبکه به ازای $|E| = 120$

۳.۲.۴ بررسی پیچیدگی طول پیام

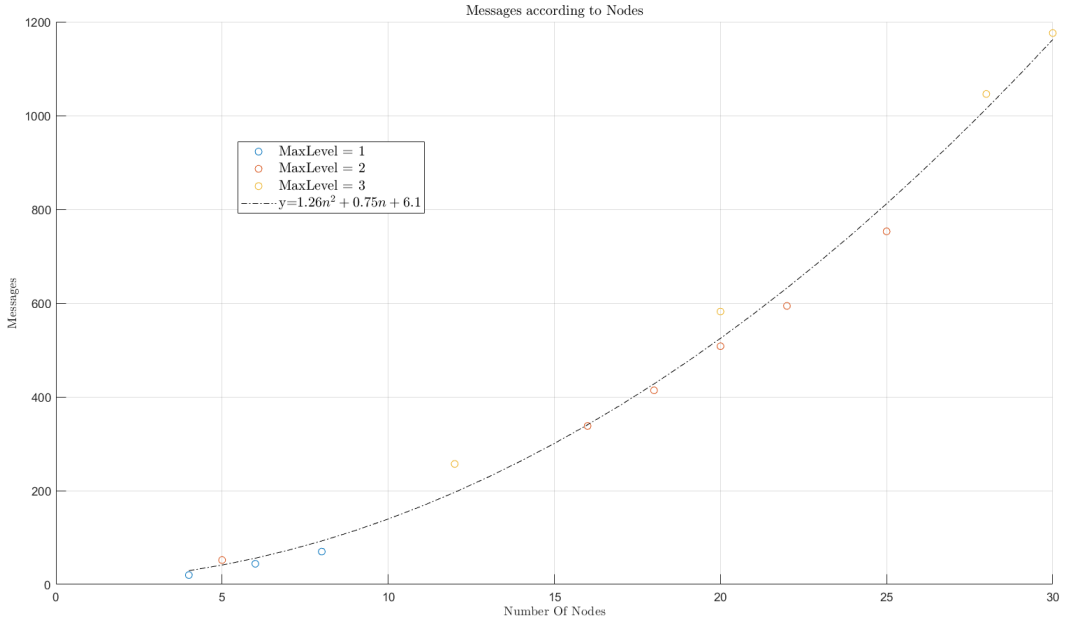
همان‌طور که گفته شد، میانگین طول پیام‌های مختلف برابر ۶۶ بیت است. در اجراهای مختلف میانگین طول پیام بر حسب تعداد گره‌ها به دست آمده است. میانگین طول پیام در هر اجرا تقریبا روند ثابتی دارد. میانگین طول پیام‌ها در کل برابر 66.28 است و انحراف معیار این مقادیر برابر 1.86 است که نشان از عدم وابستگی این پارامتر به تعداد گره‌ها و تغییر نه‌چندان طول پیام‌ها با تغییر گراف شبکه و سایر پارامترهای آن دارد. هم‌چنین تعداد کل بیت‌های ارسالی نیز، به طبع روندی مشابه تعداد پیام‌ها (شکل ۱۳) خواهد داشت.



شکل ۱۷: میانگین طول پیام‌ها بر حسب تعداد گره در شبکه

۳.۴ توپولوژی گراف کامل

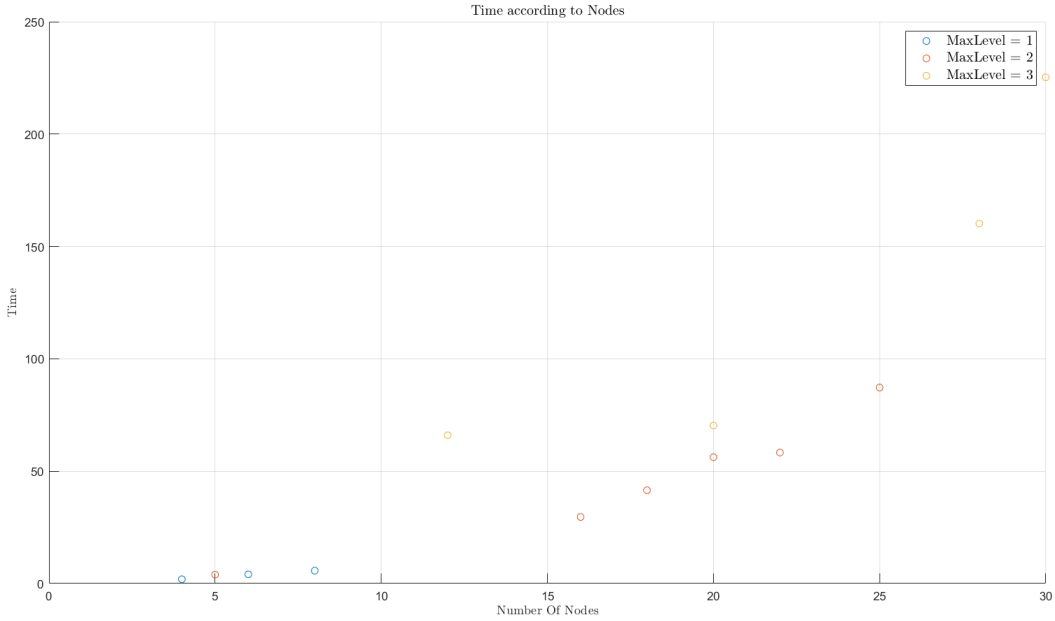
در این آزمایش، عملکرد الگوریتم بر روی گراف کامل (با حداکثر تعداد یال) بررسی می‌شود. به این منظور تعداد گره‌ها از ۴ تا ۳۰ تغییر می‌کند. برای گراف کامل، $|E| = \frac{n(n-1)}{2}$ و پیچیدگی تعداد پیام به صورت $O(n^2)$ در می‌آید. شکل ۱۸ نشان می‌دهد که تابعیت تعداد پیام‌ها از تعداد گره‌ها به صورت سهمی است و ضریب درجه دو، ضریب غالب به حساب می‌آید. در نتیجه افزایش تعداد یال‌ها نقش مهمی در افزایش بیش از حد تعداد پیام‌ها دارد و در شبکه‌های کامل این امر می‌تواند باعث ازدحام در شبکه شود.



شکل ۱۸: تعداد پیام‌ها بر حسب تعداد گره در شبکه کامل

پیچیدگی زمانی نیز به صورت $O(nlogn)$ است و با حاصل‌ضرب تعداد گره‌ها در تعداد سطح‌ها رابطه دارد. همان‌طور که در شکل ۱۹ مشاهده می‌شود، به ازای تعداد سطح ثابت، رابطه به صورت خطی است و با افزایش تعداد سطح‌ها شیب نمودار افزایش می‌یابد. (برای

داده‌های قرمز رنگ شیب بزرگتر از داده‌های آبی رنگ است و برای داده‌های زرد رنگ بیشترین شیب مشاهده می‌شود) البته هم‌چنان مانند آزمایش‌های گذشته، زمان اجرا می‌تواند به قدرت پردازش پردازنده شبیه‌ساز وابسته باشد و در برخی موارد فاقد اعتبار است.

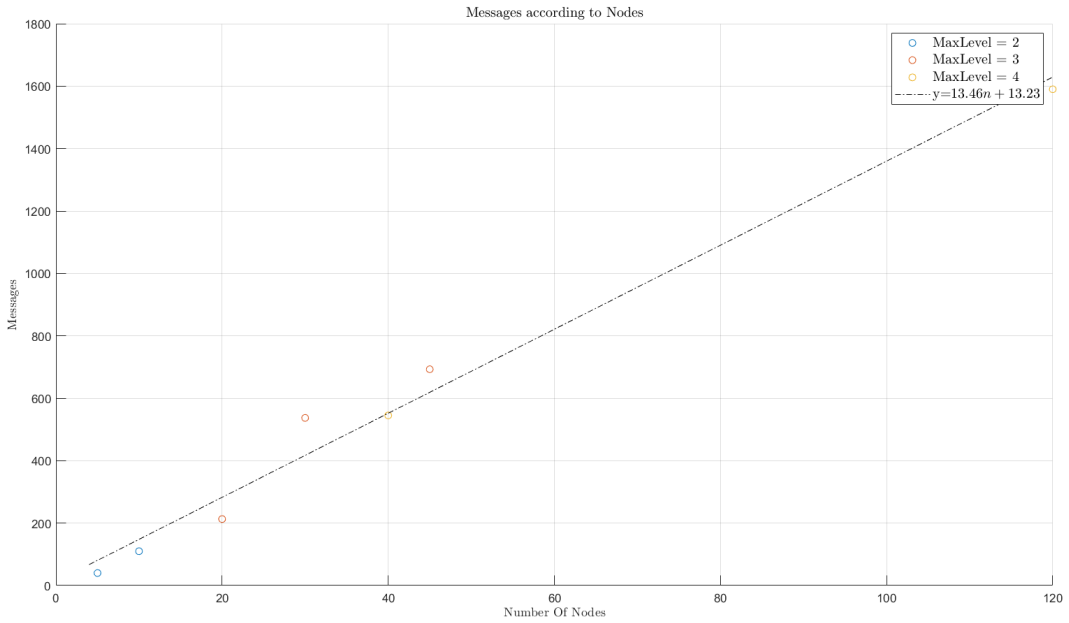


شکل ۱۹: تعداد پیام‌ها بر حسب تعداد گره در شبکه کامل

مانند آزمایش‌های قبلی متوسط طول پیام‌های ارسالی هم‌چنان در حدود 66 بیت و برابر 64.04 است و وابستگی چندانی به گراف شبکه ندارد.

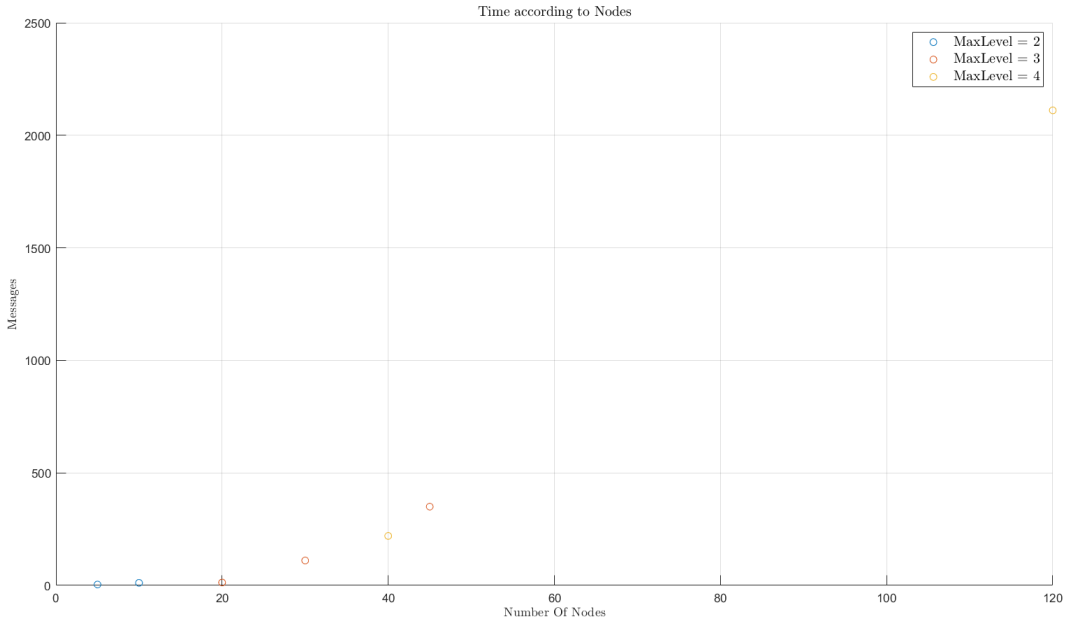
۴.۴ توپولوژی گراف حلقه

در این آزمایش، عملکرد الگوریتم بر روی گراف حلقه (با حداقل تعداد یال) بررسی می‌شود. به این منظور تعداد گره‌ها از ۴ تا ۳۰ تغییر می‌کند. برای گراف کامل، $|E| = n$ و پیچیدگی تعداد پیام به صورت $O(nlogn)$ در می‌آید. شکل ۲۰ نشان می‌دهد که تابعیت تعداد پیام‌ها از تعداد گره‌ها به صورت خطی است. البته به دلیل زمان اجرای بسیار طولانی به ازای تعداد گره‌های بالاتر، تمرکز داده‌های به دست آمده در بازه ۰ تا ۶۰ گره است.



شکل ۲۰: تعداد پیام‌ها بر حسب تعداد گره در شبکه کامل

پیچیدگی زمانی نیز به صورت $O(nlogn)$ است و با حاصل‌ضرب تعداد گره‌ها در تعداد سطح‌ها رابطه دارد. با این حال همان‌طور که در شکل ۲۱ مشاهده می‌شود، زمان اجرا وابستگی زیادی به قدرت پردازنده شبیه‌ساز داشته و داده‌ها تا حدی اعتبار هستند.



شکل ۲۱: تعداد پیام‌ها بر حسب تعداد گره در شبکه کامل

مانند آزمایش‌های قبلی متوسط طول پیام‌های ارسالی هم‌چنان در حدود 66 بیت و برابر 65.72 است و وابستگی چندانی به گراف شبکه ندارد.

۵ مراجع

1. R. G. Gallager, P. A. Humblet and P. M. Spira, A distributed algorithm for minimum-weight spanning tree, 1983

2. N. Lynch, Distributed Algorithms, 1996