دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده ریاضی و علوم کامپیوتر

گزارش دوم

بررسی الگوریتم A^* ، BFS و IDS

علیرضا داودی _ 4013011

دکتر مهدی قطعی _ دکتر بهنام یوسفی مهر

بهمن هزار و چهارصد و دو

چکیده

مسیر یابی یک الگوریتم برای برنامه‌های کامپیوتری است که هدف آن یافتن (غالبا) کوتاه‌ترین مسیر بین دو نقطه است. مسیر یابی یک راه کاربردی برای حل هزارتوها. است

مسیر یابی به مقدار زیادی به مسئله کوتاه‌ترین مسیر در نظریه گراف‌ها ارتباط دارد؛ که در واقع این مسئله به این موضوع می‌پردازد که چگونه سریع‌ترین، ارزان‌ترین (از لحاظ تعداد راس‌ها) و کوتاه‌ترین مسیر را بین دو نقطه در یک شبکه بزرگ بیابیم

الگوریتم های زیادی برای مسیریابی وجود دارد که بعضی از ان ها A*[1] ، BFS[2] و IDS[3] هستند. در الگوریتم BFS مسیر بهینه ای بدست می اید ولی زمان زیادی طول میکشد. IDS بهینه ترین مسیر را ارائه نمیدهد ولی بهینه تر از DFS است و زمان کمتری نسبت به BFS نیاز دارد. الگوریتم A* مسیر بهینه ای مانند BFS ارائه می دهد و زمان ان نیز از BFS بهتر است. همچنین روش هایی برای بهینه تر کردن مسیر و زمان در الگوریتم A* وجود دارد.

در ادامه به این روش ها خواهیم پرداخت.

[1] A-Star algorithm
[2] Breadth fist search
[3] Iterative deepening search

فهرست مطالب صفحه

چکیده .....أ
فصل اول مقدمه 2
1-1 الگوریتم *A..... 3
1-2 تابع هیوریستیک 4
فصل دوم توضیح الگوریتم BFS و IDS..... 9
2-1 BFS 6
2-2 IDS 7
فصل سوم بهبود *A 10
3-1 نقاط برخورد مسیر..... 12
3-2 نقاط چرخش..... 13
3-3 نقاط تغییر مسیر 14
فصل چهارم نتیجه..... 15
منابع 17
Abstract 18

مقدمه

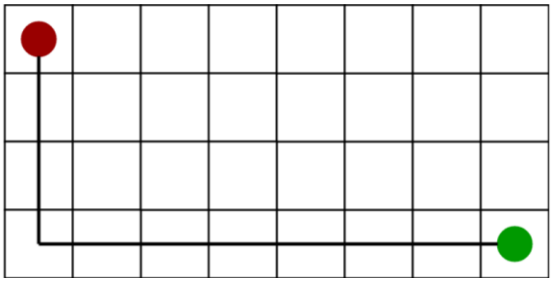
در این مقاله به بررسی الگوریتم های مسیریابی می پردازیم. برای هر الگوریتم بررسی میکنیم که ایا مسیر بهینه ای دارد یا نه ، زمانی که طول میکشد مسیر را پیدا کند زیاد است یا کم. متد مسیریابی با شروع از یک راس و جستجو در راس‌های مجاور آن تا زمان رسیدن به راس مقصد، یک گراف را جستجو می‌کند؛ که معمولاً هدف آن یافتن سریع‌ترین مسیر است. در حقیقت هدف الگوریتم مسیر یابی عموماً یافتن مسیری با کمترین تعداد راس استفاده شده (در گراف‌های فاقد وزن)، یا کمترین جمع اوزان مشاهده شده (در گراف‌های وزن‌دار) است. به عنوان مثال می‌توان گفت که الگوریتم مد نظر، همانند شخصی است که قصد دارد از نقطه‌ای به نقطه دیگری برسد؛ به جای این که این شخص تمام مسیرهای موجود را بررسی کند، در یک مسیر حرکت می‌کند و فقط زمانی از مسیر خود منحرف می‌شود که مانعی بر سر راه وی وجود داشته باشد.

همچنین روش هایی برای بهینه سازی مسیر و زمان وجود دارد که در ادامه به ان خواهیم پرداخت.

الگوریتم A*

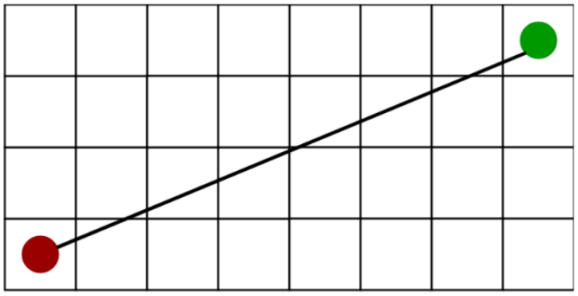
الگوریتم A* در واقع الگوریتمی است که به دنبال بهینه ترین مسیر است . در این الگوریتم از یک گره مشخص جستجو را آغاز می‌کند و هدفش پیدا کردن یک مسیر به گره نهایی یا هدف است که کمترین هزینه را دارد. این روش با ترکیب هزینه رسیدن به گره و تخمین هزینه[۴] رسیدن از گره تا گره هدف را ارزیابی می‌کند.

هزینه رسیدن به گره فعلی، مسیری است که از گره شروع تا این گره رسیده است که ان را با g(n) نشان می دهیم. تخمین هزینه رسیدن از گره فعلی به گره مقصد را میتوان فاصله گره فعلی تا گره مقصد در نظر گرفت که برای بدست اوردن ان دو روش وجود دارد. اگر صفحه به این صورت باشد که حرکت به صورت بالا، پایین، چپ و راست مجاز باشد تخمین به صورت زیر است:



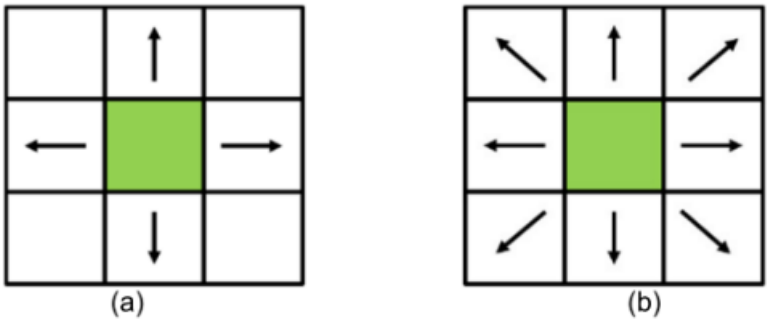
$$h(n) = (x_a - x_b) + (y_a - xy_b)$$

و اگر حرکت به صورت قطری امکان پذیر باشد به صورت زیر خواهد شد:



$$h(n) = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$$

[4] Heuristic



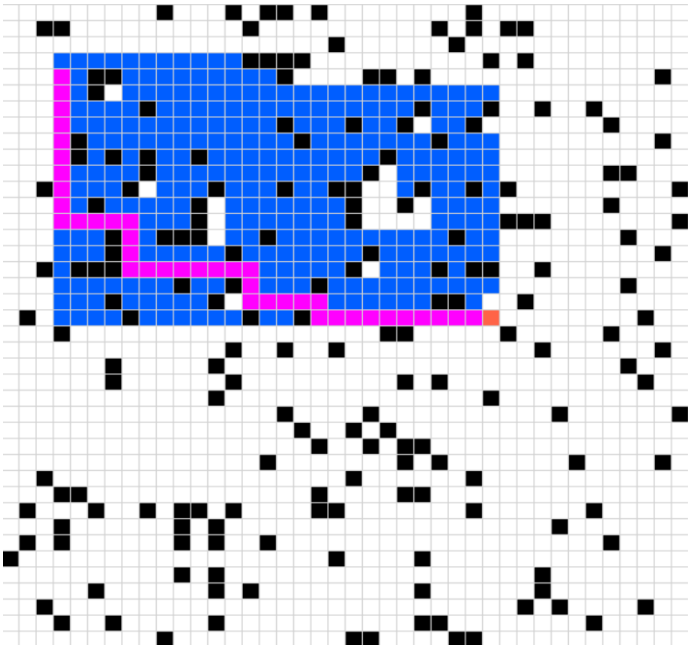
حال هزینه رسیدن از گره فعلی به گره مقصد به صورت زیر است:

$$f(n) = g(n) + h(n)$$

بنابراین اگر به دنبال بهینه ترین مسیر هستیم ، باید به گره هایی برویم که کمترین هزینه را دارند. بنابراینن وقتی از از نقطه شروع بخواهیم حرکت کنیم باید به همسایه ای از ان برویم که کمترین هزینه را دارد همین کار را برای هر گره ای که رفتیم انجام میدهیم.

برای پیاده سازی این الگوریتم نیاز به یک صف الویت[5] دار داریم. الویت این صف نیز هزینه هر گره هست و به ترتیب در ان گره ها قرار می گیرند. در هر مرحله گره فعلی که در ان قرار داریم از صف خارج میشود و همسایه های ان به ترتیب الویت در صف قرار خواهند گرفت. در مرحله بعد به گره ای با کمترین هزینه که در صف قرار دارد می رویم و ان را از صف خارج میکنیم و همسایه های ان را به ترتیب الویت در صف قرار خواهیم داد. این کار را تا زمانی انجام میدهیم تا به گره مقصد برسیم یا صف ما خالی شود که به این معنی است که مسیری وجود ندارد. همجنین یک لیست به نام visited وجود دارد که بررسی میکند که یک گره را چند بار بررسی نکند.

[5] Priority queue

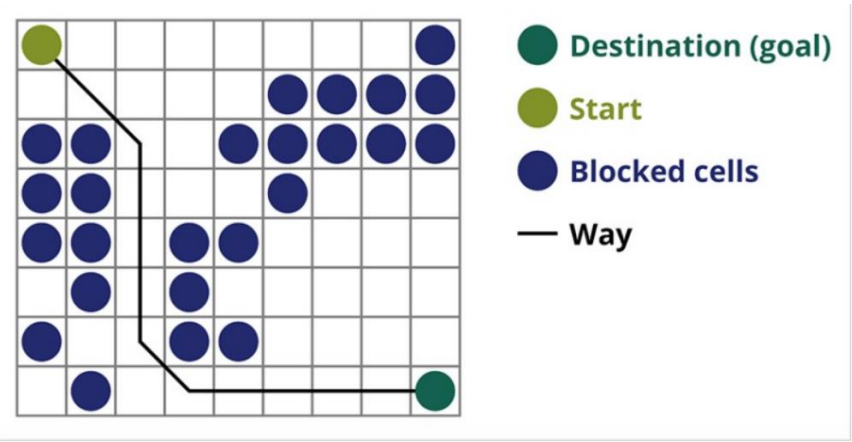


همانطور که در شکل پیداست، این مسیر ، بهینه ترین مسیر است ولی این الگوریتم برای فواصل بیشتر کند است. به طوری که از الگوریتم DFS[۶] کند تر ولی مسیر بهینه تری دارد. ولی زمان ان از BFS بهتر است و مانند BFS مسیر بهینه دارد.

همانطور که در شکل می بینیم مسیر حرکت جوری است که به سمت گوشه ها نمی رویم و برایند حرکت به سمت به هدف است. این به خاطر تابع تخمین و هزینه ای است که ما در نظر گرفته ایم.

البته اگ ر از فاصله اقلیدسی استفاده می شد در تابع تخمین ان، مسیر حرکت فرق می کرد.

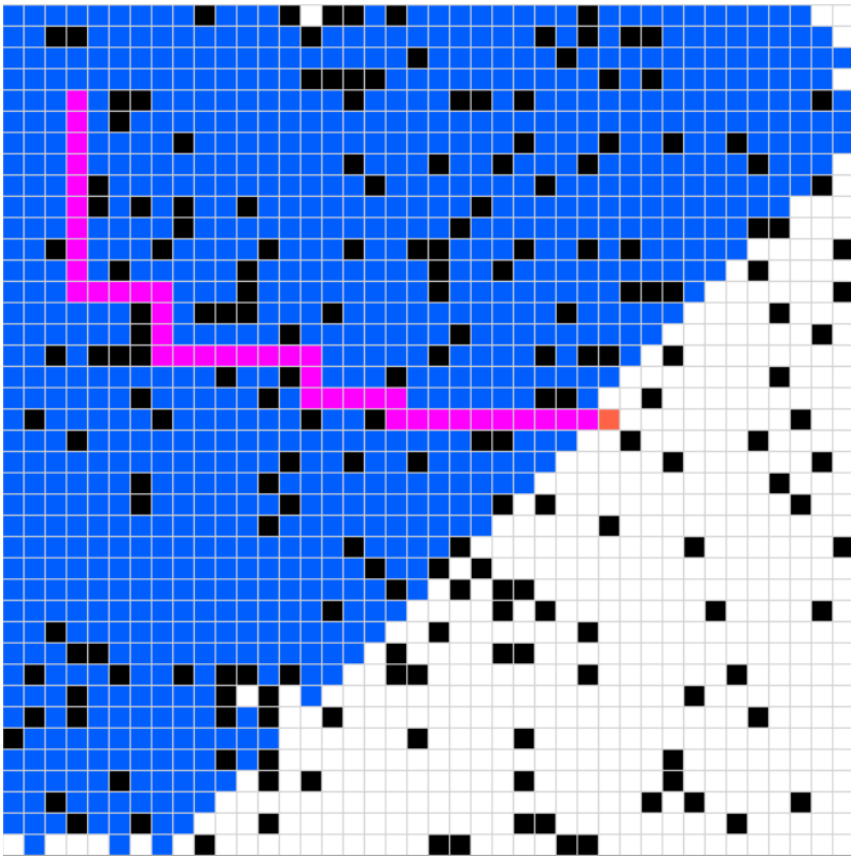
[6] Depth first search



در این مسیر ، میبینیم به صورت قطری نیز حرکت کرده ایم. اگر حرکت قطری مجاز باشد بهتر است از فاصله اقلیدسی استفاده کنیم. ولی اگر حرکت به چهار جهت فقط مجاز باشد بهتر است از فاصله منهتنی استفاده کنیم.

الگوریتم BFS

الگوریتم BFS به صورت سطحی حرکت می کند ، پس وقتی به مقصد می رسد بهینه ترین مسیر را پیدا کرده زیرا کمترین طول مسیر را طی کرده است. ولی این کار باعث می شود زمان زیادی طول بکشد.

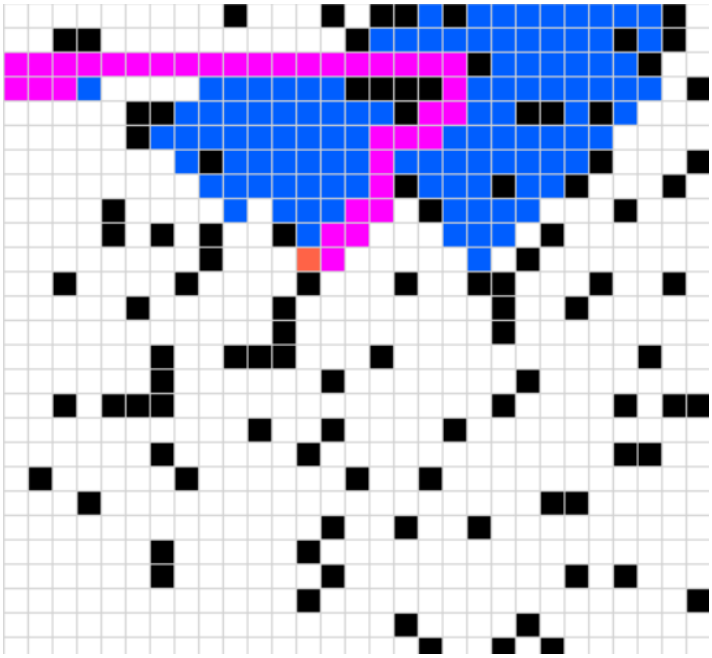


همانطور که در شکل می بینیم خانه ها و مسیر های بیشتری نسبت به الگوریتم A^* چک شده اند که همین باعث کند شدن برنامه می شود. A^* و BFS هر دو بهینه ترین مسیر را دارند.

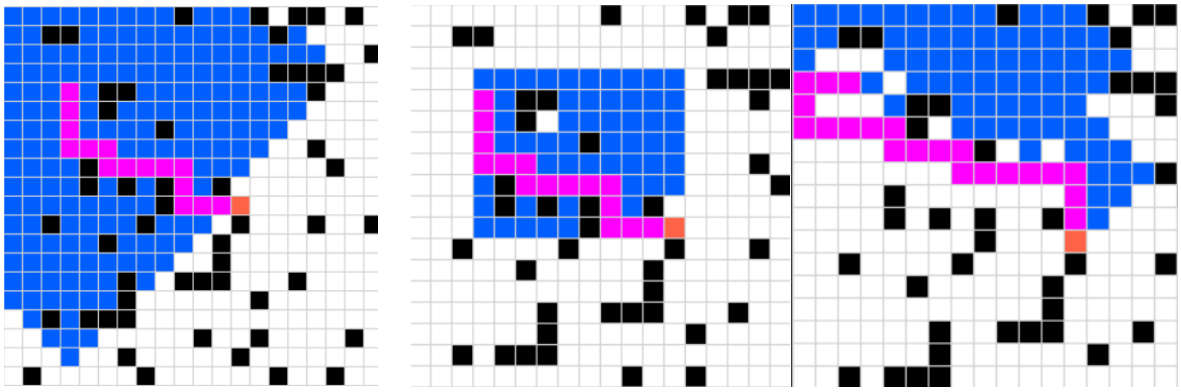
الگوریتم BFS برای فاصله کم خوب است ولی برای مسیر طولانی زمان زیادی می برد.

الگوریتم جست و جوی تعمیق تکراری

این الگوریتم به این صورت عمل میکند که در هر لایه از اول DFS می زند و مسیر را پیدا میکند. در واقع ادغام شده DFS و BFS است. این مسیر بهینه تر از مسیر DFS است ولی بهینه ترین مسیر لزوما نیست.



این تصویر نشان می دهد مسیر بهینه ترین نیست ولی از DFS بهینه تر است. همچنین این تصویر مسیر و خانه های نهایی است در حالی که در مراحل قبل خانه های زیادی بررسی شدند و بعضی خانه ها چندین بار بررسی شده اند که این باعث کند بودن این الگوریتم نسبت به DFS می شود.



الگوریتم A* الگوریتم BFS الگوریتم IDS

شکل های بالا ، الگوریتم های مختلف با شرایط یکسان را نشان می دهد. زمان اجرای برنامه در الگوریتم IDS 27.307 ثانیه طول می کشد. در الگوریتم BFS 3.143 ثانیه طول می کشد. در الگوریتم A* 1.126 ثانیه طول میکشد. نتیجه میشود الگوریتم A* کمترین زمان را می برد و سپس BFS و بعد از ان IDS . همچنین مسیر BFS و A* بهینه ترین مسیر هستند. اگر الگوریتم DFS را بررسی کنیم بد ترین مسیر را دارد و زمان ان نیز خیلی زیاد است برای مسیر های کوتاه. ولی بریا مسیر های دور تر زمان ان از بعضی الگوریتم ها بهتر است.

بهبود الگوریتم A^* [7]

ابتدا به الگوریتم ان می پردازیم:

1. 8 گره مجاور گره شروع را در لیست باز قرار میدهد و اگر گره ای مانع باشد انرا از لیست باز حذف میکند .

2. تابع $f(x)$ را برای گره های داخل لیست باز محاسبه میکنهیم و سهپس کهم هزینهه تهرین گهره را انتخاب میکنیم و گره شروع را در لیست بسته ها قرار میدهیم. تهابع $f(x) = g(x) + h(x)$ اسهت که در آن $g(x)$ هزینه رسیدن به گره بعدی و $h(x)$ تخمین یاصله گره بعدی تا گره پایانی است.

3. نود انتخابی را n در نظر میگیریم، گره های مجاور آنرا برسی میکنیم اگر در لیست باز ها نبود آنرا به عنوان یرزند n به لیست اضایه میکنیم، اگر در لیست باز ها بود تابع $g(n)$ آنرا از مسیر گره n برسی میکنم اگر از مسیر قبلی کمتر بود آنرا به عنوان یرزند n اضایه مهیکنیم و قبلهی را حهذف میکنیم و دوباره تابع $f(n)$ را برای گره هایی که در لیست باز هستند محاسبه میکنیم.

4. مرحله سوم را تا اضایه شدن نقطه هدف به لیست بسته ها ادامه میدهیم، سهپس از نقطهه هدف شروع میکنیم با دنبال کردن نقاط پدر به نقطه شروع رسیدن تا یک مسیر از نقطه شروع تا پایهان پیدا کنیم .

5. بعد از اضایه شدن گره هدف کل گره های داخل لیست بسته توسط توابع $p(x, y)$ و $w(x, y)$ چک میشوند تا گره هایی که شرایط را برآورده نمیکنند را حذف کنند.

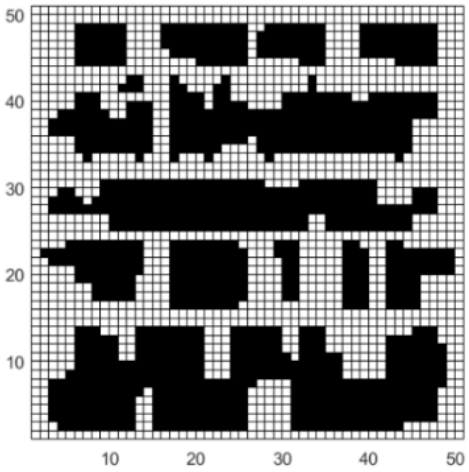
۶ چک میکنیم که ایا تعداد گره ها در لیست بسته در مرحله 5 کاهش مییابد یا خیهر اگهر کهاهش یایته بودن به مرحله 5 ادامه میدهیم در غیر این صورت از حلقه خارج شده و گره های باقی مانده را در لیست نهایی ذخیره میکنیم .

7. گره های باقی مانده را بهم وصل میکنیم تا یهک مسهیر جدیهد بدسهت بیایهد و همزمهان مسهیر چرخش را هموار میکنیم.

[7] Geometric A-Star algorithm

1. تابع تخمین:

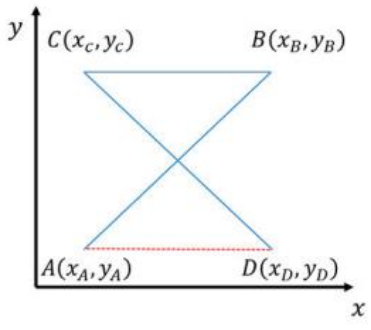
میتوان محیط را با استفاده از گوگل مپ دریافت کرد و مسیر ها و موانع را به صورت یک صفحه ماز شبیه سازی کرد. برای مثال یک اسکله را شبیه سازی می کنیم.

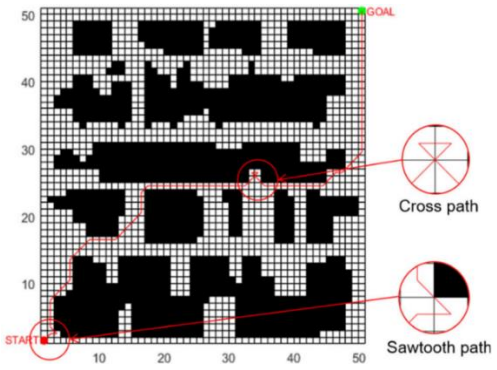


همانطور که قبلا گفته شد، برای تابع تخمین دو فرمول اویلری[8] و منهتنی[9] وجود دارد. وقتی که مسیر ما از موانع زیادی برخوردار است و باید تغییر مسیر های بیشتری داشته باشیم بهتر است از فرمول اویلری استفاده کنیم.

2. نقاط برخورد در یک مسیر[10]:

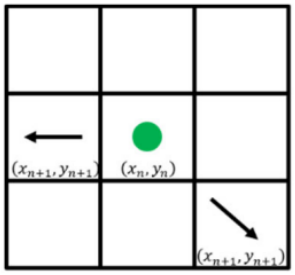
در مسیر ممکن است یک مسیر خودش را قطع کند. برای مثال در اسکله بالا:





[8] Euler formula
[9] Manhattan
[10] Optimization Of Cross Paths

اگر ما در گره nام باشیم برای فرار از این برخورد باید یا به سمت چپ برویم یا پایین راست.



پس هنگامی که به نود n+1ام می رویم چنین رابطه ای داریم:

$$\begin{cases} x_n > x_{n+1} \\ \quad or \\ y_n > y_{n+1} \end{cases}$$

پس فرمول p(x,y) را میتوان به صورت زیر تعریف کرد:

$$P(x_n, y_n) = \begin{cases} \emptyset, & x_n > x_{n+1} \cup y_n > y_{n+1} \\ (x_n, y_n), & otherwise \end{cases}$$

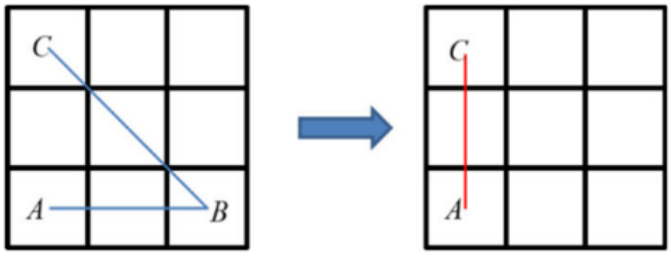
برای مثال در همان مثال برخورد در اسکله، $x_B > x_C$ پس گره B حذف می شود و گره A به گره B وصل می شود. همچنین $y_C > y_D$ پس گره C حذف خواهد شد و گره A به گره D وصل میشود.

3. مسیر های اره ای[11]:

ممکن است در مسیر به نود هایی برویم که هزینه بهتری دارند ولی باعث شوند دوباره برگردیم و مسیر کلی دور شود. برای مثال در عکس اسکله sawtooth مثالی از این مورد است.

$$W(x_n, y_n) = \begin{cases} \emptyset, & K_{n,n+1} < 0 \\ (x_n, y_n), & otherwise \end{cases} \qquad K_{n,n+1} = \frac{y_{n+1} - y_n}{x_{n+1} - x_n}$$

[11] Optimization Of Sawtooth Paths



طبق فرمول $w(x , y)$ اگر K را برای نقطه B حساب کنیم، $K_{B,C} < 0$ در نتیجه گره B حذف خواهد شد.

پس گره A به گره C وصل خواهد شد.

4. بهینه سازی در دور زدن[۱۲]:

مرحله ۱: دسترسی به لیست نهایی؛

مرحله ۲: به دلیل اینکه گره‌های مجاور در لیست نهایی ممکن است پس از بهینه‌سازی مسیر دورتر از هم باشند. صاف کردن مسیر ممکن است با موانع برخورد کند. لیست نهایی را طی کرده، و زاویه چرخش α که توسط گره فعلی $n(n > 1)$، گره قبلی n-1، و گره بعدی در لیست نهایی n+1 تشکیل شده است را بر اساس فرمول زیر محاسبه کنید. اگر $\alpha \neq \pi$ ، مختصات گره فعلی n به لیست نهایی اضافه می‌شود، و مختصات سایر گره‌ها با فضا نشان داده می‌شود. اگر $\alpha = \pi$، ادامه دهید تا گره‌های باقی‌مانده را فیلتر کنید؛

$$\alpha = \arccos\left[\frac{(x_{n-1} - x_n)(x_{n+1} - x_n) + (x_{n-1} - x_n)(x_{n+1} - x_n)}{\sqrt{(x_{n-1} - x_n)^2 + (y_{n-1} - y_n)^2}\sqrt{(x_{n+1} - x_n)^2 + (y_{n+1} - y_n)^2}}\right].$$

مرحله ۳: بر اساس رابطه مختصات بین گره فعلی n و گره قبلی n-1 در لیست بسته؛ مختصات گره‌های n-1 و n+1 در لیست نهایی با استفاده از فرمول‌های زیر به‌روزرسانی می‌شوند:

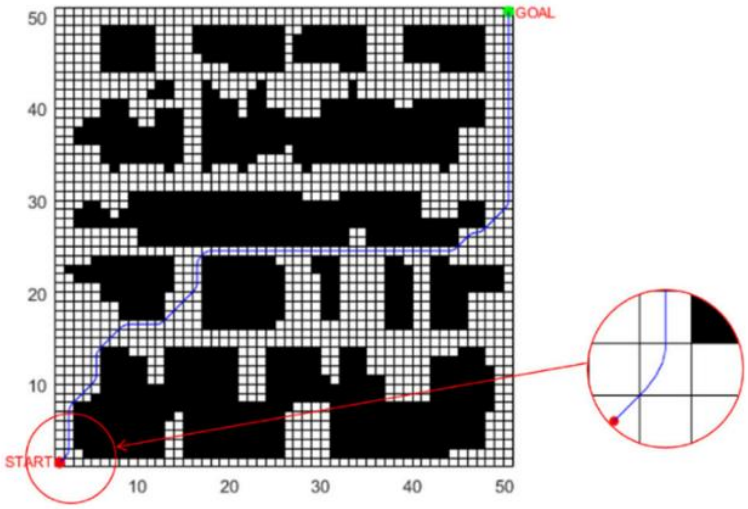
[12] Sawtooth Turning Path

$$if\ x_{n-1} \neq x_n, therefore \begin{cases} x_{n-1} = x_n - 1 \\ y_{n-1} = y_n - 1 \\ x_{n+1} = x_n + 1 \\ y_{n+1} = y_n \end{cases},$$

$$if\ x_{n-1} = x_n, therefore \begin{cases} x_{n-1} = x_n \\ y_{n-1} = y_n - 1 \\ x_{n+1} = x_n + 1 \\ y_{n+1} = y_n + 1 \end{cases};$$

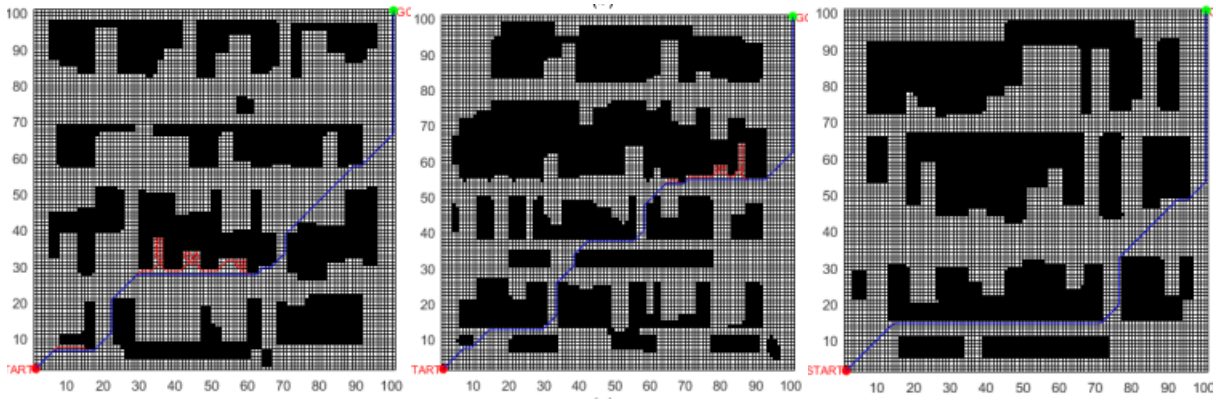
مرحله ۴: تابع تقریب B-spline مکعبی در فرمول * برای تطابق گره‌ها در لیست نهایی استفاده می‌شود؛

پس از تطابق مسیر خطی با یک منحنی B-spline مکعبی، مسیر چرخش صاف شده که در شکل زیر نشان داده شده است، به دست می‌آید.



بررسی محیط های متفاوت:

اگر محیط ها فرق کنند نتیجه الگوریتم ها چه خواهد شد؟

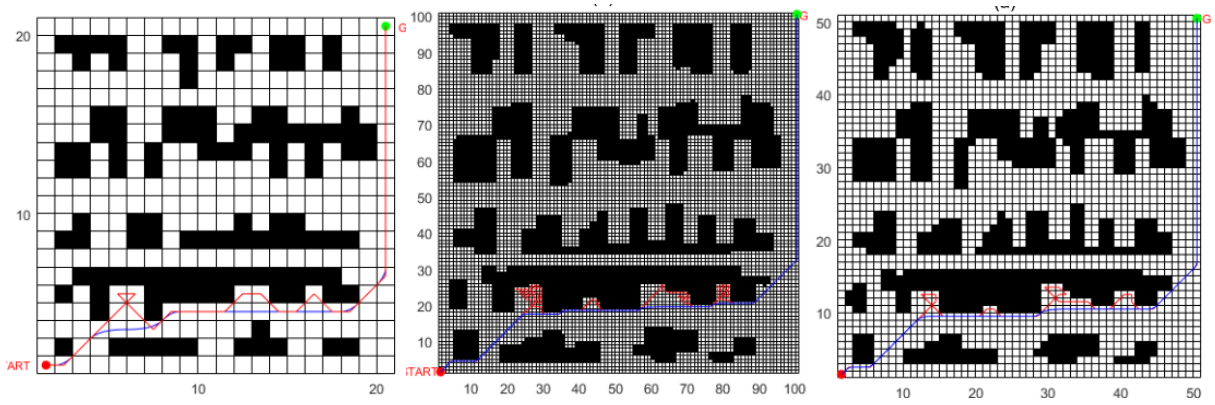


Map number	Path parameters	Traditional A-star	Geometric A-star	Reduced proportion
3	Running time/s	351.5	349.67	0.18%
	Number of nodes	160	160	0%
	Number of turns	7	7	0%
	Max turning angle	45°	45°	0%
	Total distance/m	175.15	174.835	0.18%
4	Running time/s	453.58	336.24	25.9%
	Number of nodes	193	110	43%
	Number of turns	60	9	85%
	Max turning angle	135°	45°	66.7%
	Total distance/m	226.79	168.12	25.9%
5	Running time/s	609.198	358.69	41.1%
	Number of nodes	409.22	345.264	15.6%
	Number of turns	182	138	24.2%
	Max turning angle	39	17	56.4%
	Total distance/m	135°	45°	66.7%

طبق جدول، در می یابیم که هرچه مسیر پیچیده تر و موانع بیشتری داشته باشد الگوریتم بهینه شده بهتر است. همچنین به این در می یابیم هرچه مسیر ساده تر باشد الگوریتم A* بهتر است.

تاثیر ابعاد بر الگوریتم ای استار :

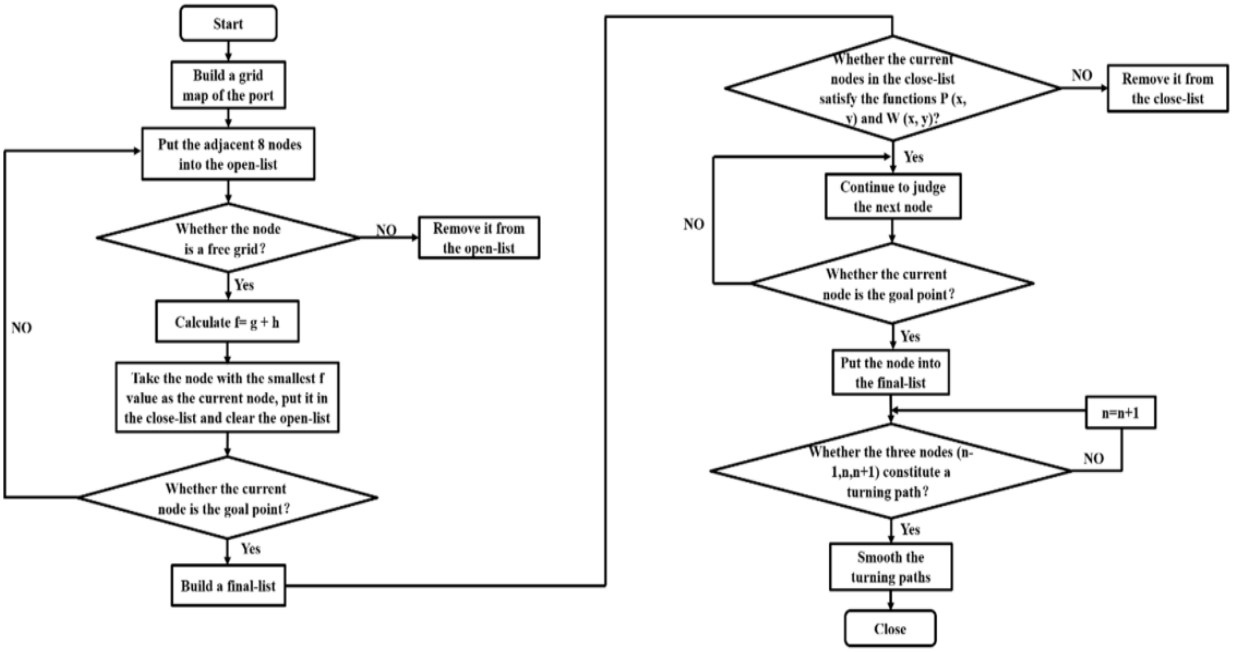
می خواهیم اسکله را به سه شبکه ۲۰×۲۰، ۵۰×۵۰ و ۱۰۰×۱۰۰ تبدیل کنیم. نقشه‌های شبکه تبدیل شده در شکل زیر نشان داده شده‌اند.



همانطور که معلوم است هرچه شبکه به تعداد بیشتری تقسیم شود تاثیر بهبود ها بیشتر است.

و هرچه شبکه به مربع های کمتری تقسیم شود این تاثیر کمتر می شود.

Map size	Path parameters	Traditional A-star	Geometric A-star	Reduced proportion
20×20	Running time/s	80.768	69.602	13.8%
	Number of nodes	36	29	19.4%
	Number of turns	14	6	57.1%
	Max turning angle	135°	45°	66.7%
	Total distance/m	201.92	174.005	13.8%
50×50	Running time/s	215.68	177.796	17.6%
	Number of nodes	96	71	26%
	Number of turns	25	7	72%
	Max turning angle	135°	45°	66.7%
	Total distance/m	215.68	177.796	17.6%
100×100	Running time/s	609.198	358.69	41.1%
	Number of nodes	230	141	38.7%
	Number of turns	69	11	84.1%
	Max turning angle	135°	45°	66.7%
	Total distance/m	304.599	179.345	41.1%



فلوچارت نهایی الگوریتم بهبود یافته.

نتیجه گیری

الگوریتم های زیادی وجود دارند که مسیریابی انجام می دهند. هدف ما از این الگوریتم ها یافتن مسیر های بهینه و همینطور زمان بهینه است.

باید در گام اول محیط را بررسی کنیم و با در نظر گرفتن ان الگوریتم مناسب با ان را انتخاب نماییم.

همچنین فاصله شروع و هدف بسیار مهم است زیرا بعضی الگوریتم ها در مسیر های کوتاه تر بهتر هستند.

همچنین به الگوریتم ای استار پرداختیم که چگونه کار میکند همچنین راه هایی برای بهبود عملکرد ان را شرح دادیم. این راه ها نیز بستگی به محیط و شرایط دارند که ایا کمکی می کنند یا نه.

منابع و مراجع

GANG TANG , CONGQIANG TANG , CHRISTOPHE CLARAMUNT , XIONG HU , AND PEIPEI ZHOU,G. Tang et al.: Geometric A-Star Algorithm: Improved A-Star Algorithm for AGV Path Planning in Port Environment

Abstract

Route finding is an algorithm for computer programs whose goal is to find the (often) shortest path between two points. Route finding is a practical way to solve mazes. Route finding is closely related to the shortest path problem in graph theory; which actually addresses the issue of how to find the fastest, cheapest (in terms of the number of vertices), and shortest path between two points in a large network.

There are many algorithms for routing, some of which are A*, BFS, and IDS. In the BFS algorithm, an optimal path is obtained, but it takes a long time. IDS does not provide the most optimal path but is more optimal than DFS and requires less time than BFS. The A* algorithm provides an optimal path similar to BFS, and its timing is also better than BFS. There are also methods for optimizing the path and time in the A* algorithm.

We will continue to discuss these methods.

Amirkabir University of Technoligy

(Tehran Polytechnic)

Facolty MCS

Second Project

Analys A*, IDS, BFS

4013011 _ Alieza Davoudi

Dr.Yousefi Mehr _ Dr.Ghatee

Winter, 2024