
بسم الله الرحمن الرحيم

**دانشکده مهندسی**

پایان‌نامه کارشناسی ارشد در رشته مهندسی کامپیوتر (نرم افزار)

# جستجوی مکاشفه‌ای با حافظه‌ی محدود

**توسط:**

**مرتضی کشت‌کاران**

**استاد راهنما:**

**دکتر کورش زیارتی**

**شهریور ۱۳۸۸**

به نام خدا

**جستجوی مکاشفه ای با حافظه ی محدود**

به وسیله ی:
**مرتضی کشت کاران**

**پایان نامه**
**ارائه شده به تحصیلات تکمیلی دانشگاه به عنوان بخشی**
**از فعالیت‌های تحصیلی لازم برای اخذ درجه کارشناسی ارشد**

در رشته:
**مهندسی کامپیوتر-نرم افزار**

**از دانشگاه شیراز**
**شیراز**

**جمهوری اسلامی ایران**

**ارزیابی شده توسط کمیته پایان نامه با درجه: عالی**

دکتر کورش زیارتی، استادیار بخش مهندسی کامپیوتر (رئیس کمیته) .....

دکتر سیده زهره عظیمی‌فر، استادیار بخش مهندسی کامپیوتر .....

دکتر محمد هادی صدرالدینی، دانشیار بخش مهندسی کامپیوتر .....

**شهریور ماه ۱۳۸۸**

به نام خدا

# اظهارنامه

اینجانب مرتضی کشت‌کاران (۸۵۰۱۰۷)
دانشجوی رشته‌ی مهندسی کامپیوتر گرایش نرم‌افزار دانشکده‌ی مهندسی
اظهار می‌کنم که این پایان‌نامه حاصل پژوهش خودم بوده و در جاهایی که از منابع دیگران استفاده کرده‌ام، نشانی دقیق و مشخصات کامل آن را نوشته‌ام. همچنین اظهار می‌کنم که تحقیق و موضوع پایان‌نامه‌ام تکراری نیست و تعهد می نمایم که بدون مجوز دانشگاه دستاوردهای آن را منتشر ننموده و یا در اختیار غیر قرار ندهم. کلیه حقوق این اثر مطابق با آیین‌نامه مالکیت فکری و معنوی متعلق به دانشگاه شیراز است.

نام ونام خانوادگی:

تاریخ و امضا:

تقدیم به پدر و مادرم که بدون همراهی و پشتیبانی آن ها موفقیتی حاصل نمی شد و تقدیم به برادر و خواهرم که همیشه دوست و پشتیبان من بودند.

# سپاسگزاری

پس از شکر خدا بر خود لازم می دانم از زحمات تمام اساتیدی که در این مقطع مرا یاری رساندند به ویژه از آقای دکتر زیارتی که در طول این دوره همراه، پشتیبان و راهنمای من بودند، قدردانی و سپاسگزاری نمایم.

همچنین از تمام دوستانی که در طول این دوره مرا صمیمانه همراهی و راهنمایی نمودند قدردانی می نمایم.

# چکیده

## جستجوی مکاشفه‌ای با حافظه‌ی محدود

به کوشش

**مرتضی کشت‌کاران**

الگوریتم‌های جستجو در گراف در حل بسیاری از مسائل مهندسی و علوم کامپیوتر مورد استفاده قرار می‌گیرند. از جمله‌ی این الگوریتم‌ها، الگوریتم *A می‌باشد که به دلیل نیاز به حافظه‌ی بسیار زیاد، کارایی خود را برای حل مسائل بزرگ از دست می‌دهد. روش‌های بسیاری در جهت کاهش حافظه‌ی مورد نیاز در این الگوریتم ارائه گردیده‌اند که در این پایان‌نامه تعدادی از آن‌ها مورد بررسی قرار گرفته و در پایان دو راهکار جدید در این راستا ارائه می‌گردد.

الگوریتم *IDA از جمله الگوریتم‌های ارائه شده در جهت خطی کردن حافظه‌ی مورد نیاز الگوریتم *A می‌باشد. از مشکلات این الگوریتم تولید گره‌های تکراری برای رسیدن به جواب می‌باشد. راهکار اول ارائه شده در این پایان‌نامه اختصاص به ارائه‌ی روشی کم هزینه در جهت جلوگیری از تولید مجدد برخی از گره ها در این الگوریتم دارد.

راهکار دوم ارائه‌ی روشی در جهت کاهش فضای مورد نیاز برای تولید و نگهداری پایگاه الگویی داده‌ی جمع‌پذیر، از جمله روش‌های تقویت تابع مکاشفه‌ای، برای مسئله ی مرتب سازی پنکیک‌ها می‌باشد. این روش با کاهش فضای حالت در ساخت این پایگاه‌های الگویی داده، حل مسائل با تعداد پنکیک بیشتر را که پیش از این به دلیل محدودیت حافظه مقدور نبود، میسر می‌سازد.

فهرست مطالب

عنوان صفحه

# فهرست جدول ها

عنوان و شماره صفحه

فهرست شکل ها و تصویرها

عنوان و شماره صفحه

# مقدمه

# مقدمه

بسیاری از مسائل مهندسی و مهم علمی می‌توانند با مسئله‌ی یافتن کوتاهترین مسیر در گراف مرتبط شوند. تعیین مسیر ارتباطات تلفنی، مسیریابی در معمای Maze و طراحی بردهای الکترونیکی نمونه‌هایی از این مسائل هستند. در سال ۱۹۶۸ الگوریتم A* [۱] از ادغام دو دیدگاه ریاضیاتی و مکاشفه ای[1] که برای حل این مسائل استفاده می شدند، ارائه گردید. در این الگوریتم نشان داده می‌شود که چگونه می توان اطلاعاتی از مسئله‌ی مورد نظر را در شیوه‌های ریاضیاتی که برای یافتن کوتاهترین مسیر در گراف ها استفاده می شوند به کار برد تا جستجو هدفمند گشته و با پردازش تعداد گره‌های کمتری بتوان جواب بهینه را پیدا نمود. به عنوان مثال برای یافتن کوتاهترین مسیر بین دو شهر می توان از این خصوصیت که فاصله‌ی بین دو شهر مجاور هیچگاه از فاصله‌ی مستقیم آن دو کمتر نیست استفاده نمود تا با پردازش شهرهای کمتری این فاصله ی بهینه یافت شود. به چنین خصوصیاتی که از مسئله گرفته شده و در حل آن مورد استفاده قرار می گیرند اطلاعات مکاشفه ای گفته می شود. الگوریتم های جستجوی اول بهترین[2] که A* نیز از آن دسته می‌باشد، با نگهداری حالاتی[3] که در طول اجرای الگوریتم مشاهده می‌شوند، از گسترش یافتن یک حالت برای چندین مرتبه جلوگیری می‌کنند. در نتیجه‌ی این ذخیره کردن حالات، این الگوریتم‌ها با کمبود حافظه مواجه می شوند. به همین دلیل تا کنون برای مسائل گوناگون تلاش‌های بسیاری در جهت بهبود این الگوریتم ها از لحاظ حافظه و همچنین از لحاظ تقویت تابع تخمین مورد استفاده در آنها، که خود منجر به کاهش

[1] Heuristic Approach
[2] Best-First Search
[3] States

حافظه‌ی مورد نیاز می‌گردد، صورت پذیرفته است. هدف از انجام این پایان‌نامه بررسی چنین الگوریتم هایی می باشد. در ادامه محتوای هر فصل به اختصار آورده شده است.

در فصل اول پس از تعریف فضای جستجو و مطالب مرتبط با یافتن کوتاهترین مسیر در گراف، به شرح الگوریتم های اول بهترین که A* از آن دسته می باشد پرداخته می شود. سپس دو مسئله‌ی کاشی های متحرک[1] و مرتب سازی پنکیک ها[2] که نتایج و مقایسه‌ی الگوریتم-های مختلف این پایان نامه بر روی آن ها ارائه شده است، تعریف خواهند شد.

فصل دوم و سوم اختصاص به مروری بر کارهای گذشته دارد. مطالب فصل دوم در دو قسمت ارائه می شود. در قسمت اول این فصل الگوریتم IDA*، از جمله الگوریتم های جستجو با حافظه‌ی خطی، ارائه می شود. الگوریتم های جستجوی خطی از حافظه ی بسیار کمی بهره می‌برند و در نتیجه با مشکل حافظه روبرو نیستند. اما در این الگوریتم ها قسمت بسیار زیادی از حافظه در طول جستجو بدون استفاده باقی می‌ماند. این الگوریتم ها با خطی کردن حافظه‌ی مورد، نیاز توانایی جلوگیری از تولید و گسترش گره‌های تکراری را ندارند و در نتیجه کارایی خود را برای مسائلی که مسیرهای متعددی تا گره‌های مختلف در آن‌ها وجود دارد، از دست می‌دهند. به همین دلیل الگویتم هایی ارائه شده‌اند که از باقیمانده‌ی حافظه جهت تشخیص گره‌های تکراری بهره می‌برند که در این فصل به بررسی تعدادی از این الگوریتم ها نیز پرداخته خواهد شد. در قسمت دوم این فصل یکی از روش‌های کاهش مستقیم فضای مورد نیاز الگوریتم A* شرح داده خواهد شد که با حذف یکی از لیست‌های مورد نیاز در این الگوریتم آن را قادر به حل مسائل بزرگ‌تر می‌گرداند.

برخی دیگر از الگویتم‌ها از حافظه جهت تقویت تابع مکاشفه‌ای موجود و یا نگهداری مقادیر مکاشفه‌ای دقیق تر بهره می‌برند که فصل سوم به این دسته از الگوریتم ها می پردازد.

و در نهایت فصل چهارم به ارائه ی روش های جدید ارائه شده در این پایان نامه و نتایج حاصل از آن ها اختصاص دارد. روش اول ارائه شده در این فصل راهکاری در جهت کاهش تولید گره‌های تکراری در الگوریتم IDA* می باشد. روش دوم ارائه شده، در جهت کاهش فضای مورد نیاز برای ساخت و نگهداری مقادیر مکاشفه‌ای برای مسئله ی مرتب سازی پنکیک

[1] Sliding Tile Puzzle
[2] Pancake Sorting Puzzle

ها مورد استفاده قرار می‌گیرد که حل این مسئله را برای تعداد پنکیک های بیشتر که تا قبل از این مقدور نبود، میسر می‌سازد.

# فصل اول

## پیش زمینه

# ۱- پیش زمینه

## ۱-۱ فضای حالت[1] جستجو در گراف

فضای حالت جستجو در گراف یک قالب متداول در حل مسائل هوش مصنوعی می باشد. در حالت کلی فضای حالت جستجو را می توان با گراف وزن دار $G =< N, E, c >$ نشان داد که در آن تعریف‌های زیر ارائه می شوند:

- $n \in N$: گره متناظر با حالتی از مسئله
- $(n, n') \in E$: یال متناظر با حرکتی که حالت مسئله را از $n$ به $n'$ تغییر می دهد
- $c : E \to R^+$: تابع هزینه که مقدار حقیقی و مثبت $c(n, n')$ را به هر حرکت $(n, n') \in E$ انتساب می‌دهد.

به گرافی که هزینه‌ی تمام یال‌ها (حرکات) در آن یکسان باشند **گراف یکنواخت** گفته می-شود.

هر نمونه[2] ای از مسئله را می‌توان با چندتایی $< N, E, c, n_0, T >$ نشان داد که در آن تعاریف زیر ارائه می‌شوند:

- $N$، $E$ و $c$ همانطور که در بالا تعریف شدند
- $n_0 \in N$: گره متناظر با حالت اولیه‌ی مسئله
- $T \subseteq N$: گره‌های متناظر با حالات پایانی مسئله

[1] State Space
[2] Instance

جواب یک مسئله ی جستجو، مسیری از گره اولیه ی $n_0$ به گره هدفی چون $n_t \in T$ می-باشد که به صورت $\pi = \{n_0, n_1, ..., n_t\}$ نشان داده می شود و در آن : $\forall_{0<i\leq t} (n_{i-1}, n_i) \in E$.

**عمق جواب** که به شکل $|\pi|$ نشان داده می شود تعداد حرکاتی است که این مسیر شامل می‌شود. هزینه‌ی جواب به شکل $c(\pi)$ نشان داده می‌شود که مجموع هزینه‌های یال‌های موجود در مسیر می باشد یعنی:

$$c(\pi) = \sum_{i=1}^{t} c(n_{i-1}, n_i)$$

**جواب بهینه** ( $\pi^*$ ) جوابی با کمترین هزینه می باشد، یعنی:

$$\pi^* = \arg \min_{\pi} c(\pi)$$

با توجه به تعریف هایی که در بالا ارائه شدند، حل یک مسئله را می توان با یافتن کوتاهترین مسیر در گراف متناظر کرد. به طور معمول فضای حالت به طور ضمنی با گره اولیه و مشخص کردن روال تولید گره‌های مجاور یک گره در آن نشان داده می شود.

## ۱-۲ جستجوی اول بهترین

الگوریتم‌های جستجوی اول بهترین از ایده‌ای مشترک جهت یافتن کوتاهترین مسیر در گراف بهره می‌برند. این الگوریتم‌ها شامل دو لیست Open و Closed می‌باشند. منظور از گسترش یک گره در این الگوریتم‌ها، تولید تمام گره‌های مجاور آن می‌باشد. به گره‌ای که تمام گره‌های مجاور آن تولید شده باشند، گره گسترش یافته[1] گفته می‌شود. لیست Open شامل گره های کاندید گسترش می باشد که در ابتدای جستجو تنها شامل گره متناظر با حالت اولیه‌ی مسئله می‌باشد. گره‌های کاندید، گره‌هایی هستند که تولید شده[2] اما هنوز گسترش نیافته اند. لیست Closed گره های گسترش یافته را در بر می‌گیرد. در این الگوریتم‌ها به هر گره هزینه ای انتساب داده می شود که تفاوت الگوریتم‌های جستجوی اول بهترین در چگونگی تعریف این

[1] Expanded Node
[2] Generated

هزینه می‌باشد. در هر مرحله‌ی جستجوی اول بهترین، الگوریتم گره‌ای از لیست Open که کمترین هزینه را داشته باشد انتخاب کرده و آن را با تولید و ارزیابی گره های مجاورش گسترش می‌دهد. سپس این گره را از لیست Open خارج کرده و به لیست Closed منتقل می‌کند. هر گره جدید تولید شده اگر حالت متناظر با آن در لیست Open یا Closed نباشد، در لیست Open وارد می‌شود و در غیر اینصورت فقط گره با هزینه‌ی کمتر در لیست Open قرار می‌گیرد. زمانی الگوریتم جستجو خاتمه می‌یابد که گره هدف برای گسترش انتخاب شود و یا در صورتی که مسئله جواب نداشته باشد لیست Open خالی شده باشد. پس از انتخاب گره هدف برای گسترش، مسیر جواب را می توان از دنبال کردن اشاره گر ذخیره شده در هر گره تا رسیدن به گره اولیه بدست آورد. این اشاره گرها نشان می دهند که گره های مربوطه از چه گره ای تولید شده اند و یا به تعبیری دیگر گره والد آن ها کدام گره است.

برای اینکه تا حد امکان گره‌های کمتری برای یافتن مسیر بهینه گسترش یابند، الگوریتم جستجو نیاز به معیاری مناسب جهت انتخاب گره بعدی برای گسترش دارد. چرا که اگر معیار مناسبی در اختیار نباشد ممکن است گره هایی که به طور واضح در مسیر بهینه قرار نمی-گیرند نیز گسترش یابند که باعث هدر رفتن زمان و فضای جستجو می گردد. از طرفی اگر این معیار باعث شود که از گره‌هایی که امکان حضور در مسیر بهینه را دارا می باشند صرف نظر شود، این امر می تواند منجر به از دست رفتن بهینگی جواب بدست آمده گردد.

از جمله حالات خاص جستجوی اول بهترین می‌توان الگوریتم‌های جستجوی سطحی[1]، دیکسترا[2] و A* را نام برد. همانطور که گفته شد، تنها تفاوت این الگوریتم ها در نحوه ی تعریف تابع هزینه‌ی آن‌ها می‌باشد. اگر مقدار هزینه‌ی گره ها در جستجوی اول بهترین عمق گره باشد این الگوریتم همان الگوریتم جستجوی سطحی می‌باشد. این مقدار در الگوریتم دیکسترا، $g(n)$، مجموع هزینه‌ی یال هایی که از گره اولیه تا گره $n$ طی شده اند تعریف می شود و در نهایت $f(n)=g(n)+h(n)$ مقدار هزینه برای گره $n$ در الگوریتم A* می‌باشد که در آن $h(n)$ تخمین هزینه‌ی باقیمانده تا گره هدف می باشد و از آن به تابع مکاشفه‌ای نام برده می-شود.

[1] Breath-First Search
[2] Dijkstra