

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ



دانشکده مهندسی

پایان نامه کارشناسی ارشد در رشته مهندسی کامپیوتر (نرم افزار)

جستجوی مکاشفه‌ای با حافظه‌ی محدود

توسط:

مرتضی کشت کاران

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 عظیمی فر، استادیار بخش مهندسی کامپیوتر

دکتر محمد هادی صدرالدینی، دانشیار بخش مهندسی کامپیوتر

شهریور ماه ۱۳۸۸

به نام خدا

اظهارنامه

اینجانب مرتضی کشت کاران (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 می‌نمایم 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 و نام خانوادگی:

تاریخ و امضا:

تقدیم به پدر و مادر که بدون همراهی و پشتیبانی آن‌ها موفقیتی حاصل نمی‌شد و تقدیم به برادر و خواهر 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 گره‌ها 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 مسئله‌ی مرتب‌سازی 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 مقذور نبود، میسر می‌سازد.

فهرست مطالب

صفحه	عنوان
۱.....	مقدمه
۵.....	فصل اول: پیش زمینه
۶.....	۱-۱ فضای حالت جستجو در گراف
۷.....	۲-۱ جستجوی اول بهترین
۱۰.....	۳-۱ معیار مقایسه ی الگوریتم های جستجو در گراف
۱۰.....	۴-۱ مسائل مورد بررسی
۱۱.....	۱-۴-۱ مسئله ی کاشی های متحرک
۱۲.....	۲-۴-۱ مسئله ی مرتب سازی پنکیک ها
۱۳.....	فصل دوم: الگوریتم های کاهش حافظه ی مورد نیاز در الگوریتم * A
۱۴.....	۱-۲ الگوریتم های جستجوی خطی
۱۵.....	۱-۱-۲ الگوریتم * IDA
۱۶.....	۲-۱-۲ روش های بهبود الگوریتم * IDA
۲۳.....	۲-۲ کاهش مستقیم فضای مورد نیاز الگوریتم * A
۲۵.....	فصل سوم: تقویت تابع مکاشفه ای
۲۶.....	۱-۳ جستجوی محیطی
۲۸.....	۱-۱-۳ نتایج
۲۹.....	۲-۳ مقادیر مکاشفه ای تعریف شونده توسط تجرید
۳۲.....	۱-۲-۳ تجرید های جمع پذیر

۳۶ پایگاه الگوی داده ی جمع پذیر برای مسئله ی مرتب سازی پنکیک ها
۳۷ نتایج ۱-۲-۳
۳۹ فصل چهارم: روش های جدید ارائه شده
۴۰ ۱-۴ الگوریتم IDA^* با اعتماد بر تابع مکاشفه ای (IDA^*_WRH)
۴۳ نتایج ۲-۴
۴۳ ۱-۲-۴ نتایج بدست آمده بر روی ۱۵-پازل با مقدار مکاشفه ای Manhattan
۴۶ ۲-۲-۴ تأثیر الگوریتم پیشنهادی بر الگوریتم $BIDA^*$
۴۷ ۳-۲-۴ نتایج استفاده از پایگاه الگویی داده
۴۸ ۳-۴ کاهش فضای حالت در تولید و نگهداری پایگاه الگویی داده
۴۹ ۱-۳-۴ تقارن در مسئله ی مرتب سازی پنکیک ها
۵۰ ۲-۳-۴ کاهش فضای حالت برای ساخت و نگهداری پایگاه های الگویی داده
۵۴ نتایج ۳-۳-۴
۵۵ فصل پنجم: نتیجه گیری و پیشنهادها
۵۶ ۱-۵ نتیجه گیری
۵۷ ۲-۵ کارهای آینده

فهرست جدول ها

عنوان و شماره	صفحه
جدول (۱) مقادیر مورد نیاز در MA^* برای هر گره ۱۹	۱۹
جدول (۲) میانگین نتایج بدست آمده ی $BIDA^*$ بر روی ۱۰۰۰ نمونه ی ۱۵-پازل ارائه شده توسط Korf ۲۹	۲۹
جدول (۳) نتیجه ی اجرای الگوریتم IDA^* با تابع مکاشفه ای Manhattan و پایگاه الگویی داده ی جمع پذیر مربوط به شکل (۱۰)..... ۳۴	۳۴
جدول (۴) حل ۲۵ نمونه از مسئله ی ۲۴-پازل توسط پایگاه الگوی داده ی جمع پذیر..... ۳۵	۳۵
جدول (۵) نتایج حل مسئله ی ۱۷-پنکیک..... ۳۸	۳۸
جدول (۶) نتایج بدست آمده بر روی ۱۰۰۰ نمونه ی مسئله ی ۱۵-پازل ارائه شده توسط Korf..... ۴۷	۴۷
جدول (۷) مقایسه ی IDA^*_WRH با IDA^* بر روی ۲۵ مسئله ی ساده تر ۲۴-پازل..... ۴۷	۴۷
جدول (۸) اجرای الگوریتم IDA^* و IDA^*_WRH برای ۲۵-۲۸-پنکیک..... ۵۴	۵۴

فهرست شکل ها و تصویرها

عنوان و شماره	صفحه
شکل (۱) هدف نهایی در ۱۵-پازل	۱۱
شکل (۲) حالت اولیه و هدف در پازل ۵-پنکیک	۱۲
شکل (۳) شبه کد الگوریتم IDA*	۱۵
شکل (۴) شبه کد الگوریتم MA*	۲۱
شکل (۵) مثالی از ناپایدار شدن مقادیر مکاشفه ای در الگوریتم MA*	۲۲
شکل (۶) جستجوی محیطی	۲۷
شکل (۷) تجرید ۸-پازل بر مبنای محل خانه ی خالی	۳۰
شکل (۸) نمونه ای از تجرید ۸-پازل	۳۱
شکل (۹) نمونه ای از تجرید جمع پذیر برای ۸-پازل	۳۳
شکل (۱۰) تقسیم بندی ۵-۵-۵ برای ۱۵-پازل	۳۴
شکل (۱۱) تقسیم بندی ۶-۶-۶ برای ۲۴-پازل به همراه حالت انعکاسی آن نسبت به قطر	۳۵
شکل (۱۲) مثالی از مسئله ی ۴-پنکیک	۳۶
شکل (۱۳) تقسیم بندی های مختلف ۱۷-پنکیک برای ساخت پایگاه های الگویی داده ی جمع پذیر	۳۸
شکل (۱۴) شبه کد الگوریتم IDA*_WRH	۴۲
شکل (۱۵) درصد کاهش میانگین تعداد گره های تولیدی توسط الگوریتم IDA*_WRH نسبت به الگوریتم	
IDA* برای ۱۰۰ نمونه ی Korf	۴۴
شکل (۱۶) درصد کاهش میانگین تعداد گره های تولیدی توسط الگوریتم IDA*_WRH نسبت به الگوریتم	
IDA* برای ۱۰۰۰ نمونه ی Korf	۴۵
شکل (۱۷) نمودار تخمین مقدار پارامتر LOW در الگوریتم IDA*_WRH	۴۶
شکل (۱۸) دو نمونه جداسازی ارائه شده برای ۱۷-پنکیک	۴۸
شکل (۱۹) جابجایی تجرید میانی به بعد از اولین فضای خالی با هزینه ی صفر	۴۹

- شکل (۲۰) نتیجه‌ی حاصل از معکوس کردن از پنجمین فضای خالی بعد از تجرید در شکل (۱۹) بعد از اولین معکوس انجام شده ۵۰
- شکل (۲۱) تجرید حالت قبل و بعد از حذف فضاهای خالی اضافی با پنکیکی در انتهای پشته ۵۱
- شکل (۲۲) تجرید حالت قبل و بعد از حذف فضاهای خالی اضافی بدون پنکیکی در انتهای پشته ۵۱
- شکل (۲۳) انتقال یکی از دو فضای خالی پشت سرهم به ابتدای پشته بدون تغییر در ترتیب پنکیک‌ها ۵۳

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 مکاشفه‌ای^۱ که برای حل این مسائل استفاده می‌شدند، ارائه گردید. در این الگوریتم نشان داده می‌شود که چگونه می‌توان اطلاعاتی از مسئله‌ی مورد نظر را در شیوه‌های ریاضیاتی که برای یافتن کوتاهترین مسیر در گراف‌ها استفاده می‌شوند 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 می‌توان 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 فاصله‌ی بهینه یافت شود. به چنین خصوصیتی که از مسئله گرفته شده و در حل آن مورد استفاده قرار می‌گیرند اطلاعات مکاشفه‌ای گفته می‌شود. الگوریتم‌های جستجوی اول بهترین^۲ که A^* نیز از آن دسته می‌باشد، با نگهداری حالاتی^۳ 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 می‌شوند. به همین دلیل تا کنون برای مسائل گوناگون تلاش‌های بسیاری در جهت بهبود این الگوریتم‌ها 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 الگوریتم‌هایی می‌باشد. 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 الگوریتم‌های اول بهترین که A^* از آن دسته می‌باشد پرداخته می‌شود. سپس دو مسئله‌ی کاشی‌های متحرک^۱ و مرتب‌سازی پنکیک‌ها^۲ که نتایج و مقایسه‌ی الگوریتم‌های مختلف این پایان‌نامه بر روی آن‌ها ارائه شده است، تعریف خواهند شد.

فصل دوم و سوم اختصاص به مروری بر کارهای گذشته دارد. مطالب فصل دوم در دو قسمت ارائه می‌شود. در قسمت اول این فصل الگوریتم IDA^* ، از جمله الگوریتم‌های جستجو با حافظه‌ی خطی، ارائه می‌شود. الگوریتم‌های جستجوی خطی از حافظه‌ی بسیار کمی بهره می‌برند و در نتیجه با مشکل حافظه روبرو نیستند. اما در این الگوریتم‌ها قسمت بسیار زیادی از حافظه در طول جستجو بدون استفاده باقی می‌ماند. این الگوریتم‌ها 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 الگوریتم‌هایی ارائه شده‌اند که از باقیمانده‌ی حافظه جهت تشخیص گره‌های تکراری بهره می‌برند که در این فصل به بررسی تعدادی از این الگوریتم‌ها 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 الگوریتم‌ها از حافظه جهت تقویت تابع مکاشفه‌ای موجود و یا نگهداری مقادیر مکاشفه‌ای دقیق‌تر بهره می‌برند که فصل سوم به این دسته از الگوریتم‌ها می‌پردازد.

و در نهایت فصل چهارم به ارائه‌ی روش‌های جدید ارائه شده در این پایان‌نامه و نتایج حاصل از آن‌ها اختصاص دارد. روش اول ارائه شده در این فصل راهکاری در جهت کاهش تولید گره‌های تکراری در الگوریتم IDA^* می‌باشد. روش دوم ارائه شده، در جهت کاهش فضای مورد نیاز برای ساخت و نگهداری مقادیر مکاشفه‌ای برای مسئله‌ی مرتب‌سازی 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^۱ 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 $G = \langle N, E, c \rangle$ 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
 - $c: E \rightarrow R^+$: تابع هزینه که مقدار حقیقی و مثبت $c(n, n')$ را به هر حرکت $(n, n') \in E$ انتساب می دهد.
- به گرافی که هزینه‌ی تمام یال‌ها (حرکات) در آن یکسان باشند **گراف یکنواخت** گفته می-شود.

هر نمونه^۲ ای از مسئله را می توان با چندتایی $\langle N, E, c, n_0, T \rangle$ نشان داد که در آن تعاریف زیر ارائه می شوند:

- E, N 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 $\pi = \{n_0, n_1, \dots, n_t\}$ 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 می شود. هزینه ی جواب به شکل $c(\pi)$ نشان داده می شود که مجموع هزینه های یال های موجود در مسیر می باشد یعنی:

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

جواب بهینه (π^*) 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 گره های مجاور یک گره در آن نشان داده می شود.

۲-۱ جستجوی اول بهترین

الگوریتم های جستجوی اول بهترین از ایده های مشترک جهت یافتن کوتاهترین مسیر در گراف بهره می برند. این الگوریتم ها شامل دو لیست Open و Closed می باشند. منظور از گسترش یک گره در این الگوریتم ها، تولید تمام گره های مجاور آن می باشد. به گره ای که تمام گره های مجاور آن تولید شده باشند، گره گسترش یافته^۱ گفته می شود. لیست Open شامل گره های کاندید گسترش می باشد که در ابتدای جستجو تنها شامل گره متناظر با حالت اولیه ی مسئله می باشد. گره های کاندید، گره هایی هستند که تولید شده^۲ اما هنوز گسترش نیافته اند. لیست Closed گره های گسترش یافته را در بر می گیرد. در این الگوریتم ها به هر گره هزینه ای انتساب داده می شود که تفاوت الگوریتم های جستجوی اول بهترین در چگونگی تعریف این

^۱ Expanded Node

^۲ 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 گره‌های 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 می‌توان از دنبال کردن اشاره گر ذخیره شده در هر گره تا رسیدن به گره اولیه بدست آورد. این اشاره گرها نشان می‌دهند که گره‌های مربوطه از چه گره‌ای تولید شده‌اند و یا به تعبیری دیگر گره والد آن‌ها 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 گره‌هایی که به طور واضح در مسیر بهینه قرار نمی‌گیرند نیز گسترش یابند که باعث هدر رفتن زمان و فضای جستجو می‌گردد. از طرفی اگر این معیار باعث شود که از گره‌هایی که امکان حضور در مسیر بهینه را دارا می‌باشند صرف نظر شود، این امر می‌تواند منجر به از دست رفتن بهینگی جواب بدست آمده گردد.

از جمله حالات خاص جستجوی اول بهترین می‌توان الگوریتم‌های جستجوی سطحی^۱، دیکسترا^۲ و A^* را نام برد. همانطور که گفته شد، تنها تفاوت این الگوریتم‌ها در نحوه‌ی تعریف تابع هزینه‌ی آن‌ها می‌باشد. اگر مقدار هزینه‌ی گره‌ها در جستجوی اول بهترین عمق گره باشد این الگوریتم همان الگوریتم جستجوی سطحی می‌باشد. این مقدار در الگوریتم دیکسترا، $g(n)$ ، مجموع هزینه‌ی یال‌هایی که از گره اولیه تا گره n طی شده‌اند تعریف می‌شود و در نهایت $f(n) = g(n) + h(n)$ مقدار هزینه برای گره n در الگوریتم A^* می‌باشد که در آن $h(n)$ تخمین هزینه باقیمانده تا گره هدف می‌باشد و از آن به تابع مکاشفه‌ای نام برده می‌شود.

¹ Breath-First Search

² Dijkstra