

دانشگاه صنعتي امیرکبیر  
(پلی­تکنیک تهران)

دانشكده ریاضی و علوم کامپیوتر

درس هوش مصنوعی و کارگاه

گزارش پروژه چهارم

نگارش

نرگس گل محمدی

40113426

استاد اول

دکتر مهدی قطعي

استاد دوم

بهنام یوسفي مهر

اذر 1403

# چكيده

در این پروژه از الگوریتم های سی اس پی یا ارضای محدودیت استفاده میکنیم که در آن ها باید مقداری برای متغیرها انتخاب کنیم که تمام محدودیت ها یا قیدهای مسئله را رعایت کند.

مسئله ای به اسم اسمان خراش ها داریم و شبیه یک سودوکو عمل میکند با این تفاوت که اعدادی کنار ان قرار میگیرد و محدودیت ها را بیشتر میکند.

باید با استفاده از مسائل ارضای محدودیت این مسئله اسمان خراش را حل کنیم و برای بهبود آن از الگوریتم هایی استفاده کنیم که انرا بهینه تر کند.

واژه کلیدی: مسئله ارضای محدودیت ، مسئله اسمان خراش ، بهینه سازی

|  |  |
| --- | --- |
| فهرست مطالب | صفحه |

[چكيده ‌أ](#_Toc179490925)

[۱. فصل اول مقدمه 4](#_Toc179490926)

[۲. فصل دوم مسئله ارضای محدودیت 5](#_Toc179490927)

[۱-۲- نحوه عملکرد 6](#_Toc179490928)

[۳. فصل سوم کد 7](#_Toc179490930)

[۱-۳- نحوه عملکرد 8](#_Toc179490931)

[۴. فصل چهارم جمع بندی 10](#_Toc179490934)

[۱-۴- نحوه عملکرد 11](#_Toc179490935)

# ۱. فصل اول مقدمه

**مقدمه**

مسئله آسمان‌خراش‌ها نوعی بازی منطقی مشابه سودوکو است که در آن هدف، پر کردن یک جدول ان در ان با اعداد یک تا ان است. به گونه‌ای که هیچ عددی در هر سطر یا ستون تکرار نشود. قیدهایی که در اطراف جدول نوشته شده‌اند، باید رعایت شوند. این قیدها نشان می‌دهند از هر سمت چند آسمان‌خراش قابل مشاهده هستند. آسمان‌خراش بلندتر می‌تواند دید ساختمان‌های کوتاه‌تر پشت خود را مسدود کند، بنابراین ترتیب چیدمان اعداد در هر ردیف یا ستون مستقیماً بر تعداد آسمان‌خراش‌های دیده‌شده از یک زاویه تأثیر می‌گذارد.

برای حل این مسئله، از الگوریتم‌های **CSP**  یا مسائل محدودیت‌ها استفاده می‌کنیم. در این روش، ابتدا هر خانه جدول به‌عنوان یک متغیر تعریف می‌شود و دامنه مقادیر هر متغیر برابر با اعداد 1 تاn  است. سپس قیدهای مسئله اعمال می‌شوند. قید اول، مشابه سودوکو، عدم تکرار اعداد در هر سطر و ستون است. قید دوم مربوط به اعداد نوشته‌شده در اطراف جدول است که تعداد آسمان‌خراش‌های قابل مشاهده را مشخص می‌کنند. درنهایت هدف چیدمانی از اعداد است که تمامی قیدها را رعایت کند.

الگوریتم حل، ابتدا متغیرها را مقداردهی می‌کند و در هر مرحله بررسی می‌کند که آیا مقداردهی انجام‌شده با قیدهای مسئله سازگار است یا خیر. اگر مقداردهی نامناسب باشد، الگوریتم به مراحل قبلی بازمی‌گردد و مقادیر دیگری را امتحان می‌کند. برای بهینه‌سازی از تکنیک‌هایی مثل انتخاب متغیر با کمترین دامنه مقادیر ممکن (MRV) و انتخاب مقدار با کمترین تأثیر بر سایر متغیرها (LCV) استفاده می‌شود. یا روش سازگاری کمانی (Arc Consistency) به کار گرفته می‌شود تا مقادیر ناسازگار از دامنه متغیرها حذف شوند و تعداد حالت‌های بررسی‌شده کاهش یابد.

این رویکرد نه تنها مسئله را به طور دقیق حل می‌کند، بلکه با استفاده از تکنیک‌های بهینه‌سازی، زمان حل را نیز به حداقل می‌رساند و امکان رسیدن سریع‌تر به جواب صحیح را فراهم می‌کند.

# ۲. فصل دوم مسئله ارضای محدودیت

## ۱-۲- نحوه عملکرد

الگوریتم CSP  یا مسئله ارضای محدودیت‌ها یک چارچوب برای حل مسائل است که در آن‌ها باید مقادیر مشخصی به متغیرها اختصاص داده شود، به‌گونه‌ای که یک مجموعه از محدودیت‌ها رعایت شود. مسئله به سه بخش اصلی تقسیم می‌شود:

متغیرها: مجموعه‌ای از عناصر که باید برای آن‌ها مقادیری انتخاب شود. به هر متغیر Xi​ یک دامنه مقادیر ممکن D(Xi) اختصاص داده می‌شود.

دامنه: مجموعه مقادیر مجاز برای هر متغیر. دامنه می‌تواند شامل مقادیر عددی، نمادین یا هر نوع داده دیگر باشد.

محدودیت‌ها: قوانینی که روابط میان متغیرها را مشخص می‌کنند و تعیین می‌کنند که کدام ترکیب از مقادیر برای متغیرها معتبر است. این محدودیت‌ها می‌توانند شامل محدودیت‌های دوتایی (مانند رابطه بین دو متغیر) یا چندتایی باشند.

هدف در یک مسئله CSP این است که برای همه متغیرها مقادیری از دامنه‌هایشان انتخاب کنیم، به‌گونه‌ای که تمامی محدودیت‌ها رعایت شوند. الگوریتم‌های CSP معمولاً از سه مرحله تشکیل می‌شوند:

جستجوی مقدار:  انتخاب مقادیر برای متغیرها به‌گونه‌ای که محدودیت‌ها رعایت شوند. این مرحله اغلب با روش‌های جستجوی بازگشتی و مقداردهی آزمون و خطا انجام می‌شود.

بررسی سازگاری: حذف مقادیر ناسازگار از دامنه‌ها و اطمینان از اینکه هر مقدار انتخاب‌شده با سایر متغیرها سازگار است. روش‌هایی مانند سازگاری کمانی (Arc Consistency) در این مرحله به کار می‌روند.

بازگشت: اگر در فرآیند جستجو بن‌بست ایجاد شود (هیچ مقداری برای متغیری قابل تخصیص نباشد)، به مراحل قبلی بازمی‌گردیم و مسیرهای دیگر را امتحان می‌کنیم.

در مسئله آسمان‌خراش‌ها، جدول n×n شامل متغیرهایی است که هر خانه Xi,j جدول به‌عنوان یک متغیر تعریف می‌شود. دامنه مقادیر برای هر متغیر برابر با اعداد 1 تا n است. دو دسته محدودیت باید رعایت شوند:

محدودیت‌های سودوکویی: هر عدد بین 1 تا n دقیقاً یک بار در هر سطر و ستون ظاهر شود. این محدودیت مانع از تکرار اعداد در هر ردیف یا ستون می‌شود.

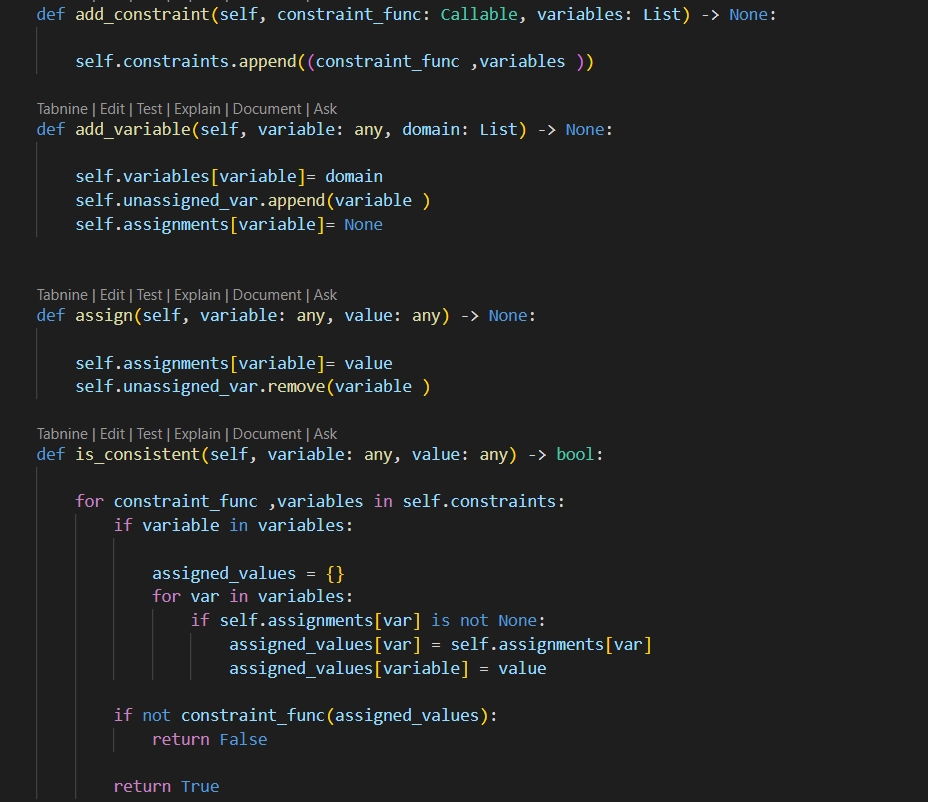
محدودیت‌های دید: اعدادی که در اطراف جدول قرار گرفته‌اند، نشان‌دهنده تعداد آسمان‌خراش‌های قابل مشاهده از آن زاویه هستند. این قید به معنی رعایت ترتیب ارتفاع ساختمان‌ها در ردیف‌ها و ستون‌هاست. برای مثال، اگر از یک زاویه عدد 3 داده شود، یعنی باید دقیقاً سه ساختمان دیده شوند و ساختمان‌های بلندتر مانع دید ساختمان‌های کوتاه‌تر می‌شوند.

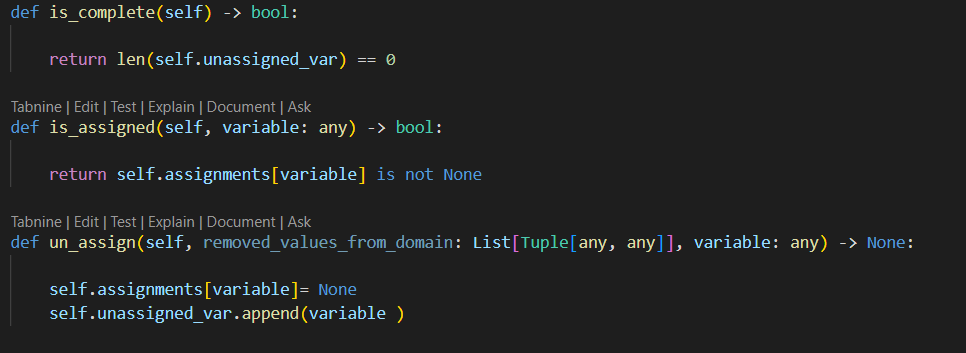
برای حل این مسئله با CSP، الگوریتم ابتدا خانه‌های جدول را مقداردهی می‌کند و بررسی می‌کند که آیا مقداردهی انجام‌شده با محدودیت‌های سودوکویی و دید سازگار است یا خیر. در صورت عدم سازگاری، به مراحل قبلی بازمی‌گردد و مقادیر دیگری را امتحان می‌کند. برای بهینه‌سازی این فرآیند، روش‌های انتخاب متغیر با کمترین دامنه (MRV) و انتخاب مقدار با کمترین تأثیر منفی (LCV) استفاده می‌شوند. همچنین، روش سازگاری کمانی (Arc Consistency) به حذف مقادیر ناسازگار از دامنه متغیرها کمک می‌کند.

این تطبیق باعث می‌شود مسئله آسمان‌خراش‌ها به طور کامل حل شود، به‌گونه‌ای که تمامی قیدها رعایت شده و جدول نهایی به دست آید.

# ۳. فصل سوم کد

## ۱-۳- نحوه عملکرد





متد add\_constraint

این متد محدودیت‌های مسئله را اضافه می‌کند. هر محدودیت شامل یک تابع (که بررسی می‌کند آیا محدودیت رعایت شده است یا خیر) و متغیرهایی است که تحت تأثیر آن محدودیت قرار می‌گیرند.

محدودیت به لیست self.constraints اضافه می‌شود.

متد add\_variable

این متد متغیرهای مسئله و دامنه مقادیر ممکن آنها را تعریف می‌کند.

متغیر و دامنه آن به دیکشنری self.variables اضافه می‌شوند و بعد متغیر به لیست self.unassigned\_var (متغیرهای تخصیص‌نیافته) اضافه می‌شود و مقدار اولیه آن در self.assignments برابر None قرار می‌گیرد.

متد assign

این متد یک مقدار مشخص را به یک متغیر تخصیص می‌دهد.

مقدار متغیر در دیکشنری self.assignments به‌روزرسانی می‌شود و متغیر از لیست self.unassigned\_var حذف می‌شود.

متد un\_assign

این متد مقدار تخصیص‌یافته به یک متغیر را حذف می‌کند و آن را به لیست متغیرهای تخصیص‌نیافته برمی‌گرداند.

مقدار متغیر در دیکشنری self.assignments به None تغییر داده می‌شود و متغیر دوباره به لیست self.unassigned\_var اضافه می‌شود.

متد is\_consistent

این متد بررسی می‌کند که مقدار انتخاب‌شده برای یک متغیر با محدودیت‌های موجود سازگار است یا نه

محدودیت‌هایی که متغیر در آنها دخیل است شناسایی می‌شوند. مقادیر فعلی متغیرها و مقدار جدید متغیر جاری در یک دیکشنری (assigned\_values) جمع‌آوری می‌شود.

تابع محدودیت بررسی می‌کند که آیا تخصیص با محدودیت‌ها سازگار است یا خیر و اگر حتی یکی از محدودیت‌ها نقض شود، مقدار False برمی‌گردد؛ در غیر این صورت True.

متد is\_complete

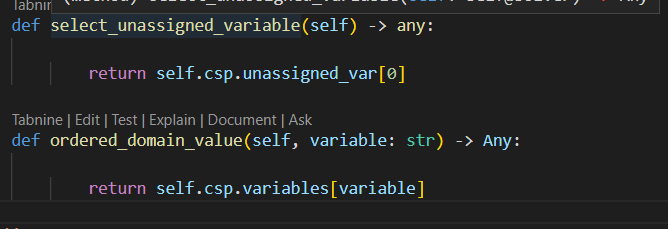
این متد بررسی می‌کند که آیا همه متغیرها مقدار گرفته‌اند یا خیر.

طول لیست self.unassigned\_var بررسی می‌شود. اگر این لیست خالی باشد، یعنی همه متغیرها مقدار گرفته‌اند، مقدار True برمی‌گردد؛ در غیر این صورت False.

متد is\_assigned

این متد بررسی می‌کند که آیا به یک متغیر خاص مقدار تخصیص داده شده است یا خیر.

مقدار متغیر در دیکشنری self.assignments بررسی می‌شود. اگر مقدار None نباشد، مقدار True برمی‌گردد.



متد select\_unassigned\_variable

این متد برای انتخاب یکی از متغیرهایی که هنوز مقدار به آن تخصیص داده نشده است، استفاده می‌شود.

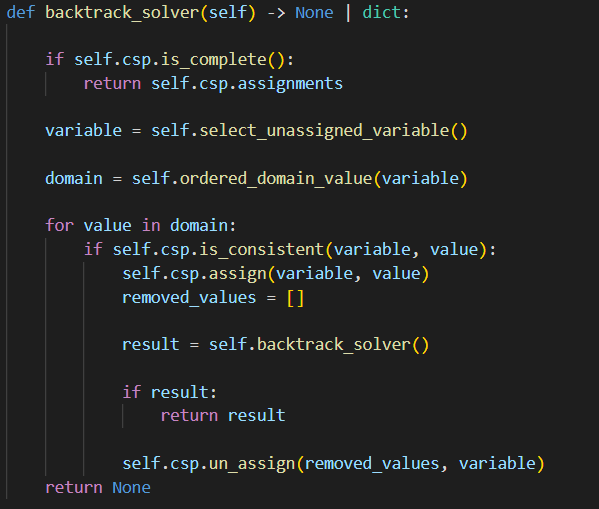
لیست self.csp.unassigned\_var شامل متغیرهایی است که هنوز مقدار نگرفته‌اند. اولین متغیر از این لیست انتخاب و برگردانده می‌شود

می‌توان متغیر را با استراتژی‌ پیچیده‌تری مثل کمترین دامنه باقی‌مانده (MRV) یا بیشترین محدودیت انتخاب کرد.

متد ordered\_domain\_value

این متد برای بازگرداندن لیست مقادیر ممکن یک متغیر خاص از دامنه آن استفاده می‌شود.

دامنه مقادیر متغیر مورد نظر که قبلاً در دیکشنری self.csp.variables ذخیره شده است، بازگردانده می‌شود. می‌توان مقادیر دامنه را بر اساس استراتژی‌هایی مثل کمترین محدودیت باقی‌مانده (LCV) مرتب کرد.



متد backtrack\_solver

یک الگوریتم جستجوی بک‌ترکینگ (Backtracking) برای حل مسئله ارضای محدودیت‌ها (CSP) است. این الگوریتم به صورت بازگشتی عمل می‌کند و هدف آن پیدا کردن یک تخصیص معتبر برای تمام متغیرهای مسئله است که تمام محدودیت‌ها را رعایت کند. در صورتی که تخصیص نادرست باشد، به عقب باز می‌گردد و سعی می‌کند گزینه‌های دیگر را بررسی کند.

بررسی کامل بودن مسئله:

اولین کاری که انجام می‌شود این است که متد is\_complete بررسی می‌کند آیا تمام متغیرها تخصیص داده شده‌اند یا خیر.

اگر همه متغیرها تخصیص داده شده باشند، self.csp.assignments که شامل تخصیصات نهایی است، برگردانده می‌شود و الگوریتم خاتمه می‌یابد.

متد select\_unassigned\_variable برای انتخاب یک متغیر تخصیص‌نیافته فراخوانی می‌شود. این متغیر قرار است در ادامه مقداردهی شود.

با استفاده از متد ordered\_domain\_value(variable), دامنه مقادیر ممکن برای متغیر انتخاب‌شده بازگردانده می‌شود.

الگوریتم شروع به بررسی هر یک از مقادیر در دامنه متغیر می‌کند.

برای هر مقدار، ابتدا با استفاده از متد is\_consistent(variable, value) بررسی می‌شود که آیا تخصیص این مقدار به متغیر باعث نقض محدودیت‌ها نمی‌شود.

اگر مقدار انتخاب‌شده با محدودیت‌ها سازگار بود، متغیر با آن مقدار تخصیص داده می‌شود (self.csp.assign(variable, value))

سپس الگوریتم وارد مرحله بازگشتی می‌شود و دوباره به backtrack\_solver فراخوانی می‌شود تا بررسی کند که آیا تخصیص جدید موجب حل کامل مسئله می‌شود یا نه.

اگر پس از فراخوانی بازگشتی، نتیجه‌ای برگردانده نشد تخصیص قبلی به متغیر حذف می‌شود و الگوریتم تلاش می‌کند مقدار دیگری را برای متغیر انتخابی بررسی کند.

برای این کار، از متد un\_assign استفاده می‌شود که تخصیص متغیر را حذف می‌کند و به لیست متغیرهای تخصیص‌نیافته اضافه می‌کند.

اگر الگوریتم هیچ‌گونه تخصیص معتبری پیدا نکرد، None باز می‌گرداند که نشان‌دهنده عدم موفقیت در یافتن راه‌حل است.

اگر تخصیصات معتبر پیدا شوند، دیکشنری self.csp.assignments که شامل تخصیصات تمام متغیرهاست، برگردانده می‌شود.

اگر هیچ راه‌حلی پیدا نشود، None برمی‌گردد.

# ۴. فصل چهارم جمع بندی

# جمع بندی

با توجه به تمام نکات گفته شده نتیجه میگیریم که الگوریتم مسئله ارضای محدودیت کاربرد زیادی دارد و میتواند در انجام بهترین حرکت در هوش مصنوعی کمک ما کند. همچنین برای بهینه سازی این الگوریتم میتوانیم از روش‌های انتخاب متغیر با کمترین دامنه (MRV) و انتخاب مقدار با کمترین تأثیر منفی (LCV) استفاده کنیم و سرعت را بالا ببریم.

منابع

<https://courses.cs.washington.edu/courses/cse573/04au/Project/mini1/RUSSIA/miniproject1_vaishu_muthu/Paper/Final_Paper.pdf>

<https://www.geeksforgeeks.org/expectimax-algorithm-in-game-theory/>

<https://www.youtube.com/watch?v=zp3VMe0Jpf8>

<https://www.youtube.com/watch?v=SLgZhpDsrfc>

<https://www.youtube.com/watch?v=Q7Xz4ZpCgUY&t=6s>

<https://www.youtube.com/watch?v=xBXHtz4Gbdo&t=249s>

<https://www.youtube.com/watch?v=fY-9Kcf9ycI>