- تابع **heuristic** (سوال ۱)

این تابع در الگوریتم min-max زمانی استفاده می‌شود که الگوریتم تا عمق مورد نظر پیش رفته است و یا حرکتی که انجام می‌شود حرکت پایانی بازی است در این حالت ما نیاز به این تابع داریم که مشخص کند که چقدر وضعیت موجود ما با توجه به حرکتی که انجام شده قابل قبول است و در واقع امتیازی که گرفته‌ایم را به ما می‌دهد. در بازی ۴connect از آنجایی که هدف ابتدایی جور کردن چهار مهره پشت سر هم است نحوه امتیاز دهی ما به اینگونه‌ای است که هرچه به این هدف نزدیک‌تر باشیم امتیاز بیشتری کسب خواهد کرد و همینطور چون نمی‌خواهیم حریف به این هدف برسد، یک امتیاز منفی برای هر موقع که حریف به این هدف نزدیک بود در نظر گرفتیم.

در کلاس ConnectSin علاوه بر متد heuristic که امتیاز بازیکنی که آیدی آن را گرفته را، با توجه به حالت حال حاضر برد، برمی‌گرداند، یک متد کمکی به نام evaluate_window نیز تعریف کردیم که کار آن است که یک لیست به طول ۴ دریافت میکند و با توجه به اینکه بازیکن چقدر به هدف نزدیک است یا حریف آن چقدر نزدیک است امتیاز را برمیگرداند. در متد heuristic کار ما این است که پنجره‌های در جهت‌های مختلف (ستون وسط، عمودی، افقی و قطری در هر دو جهت) را جدا کرده و با استفاده از تابع evaluate_window امتیاز دهیم.

برای تکمیل هر چهار مهره امتیاز ۱۰۰، برای ۳ مهره پشت سرهم و خالی بودن جایگاه بعد امتیاز ۵ و ۲ مهره امتیاز ۲ در نظر گرفته شده همچنین اگر بازیکن حریف ۳ مهره پشت سر هم داشت امتیاز -۴ تعلق میگیرد. همه‌ی امتیازات داده شده در پنجره‌های کوچک باهم جمع شده و مجموعه‌ی کل صفحه در نظر گرفته میشود.

- توابع **min-max**

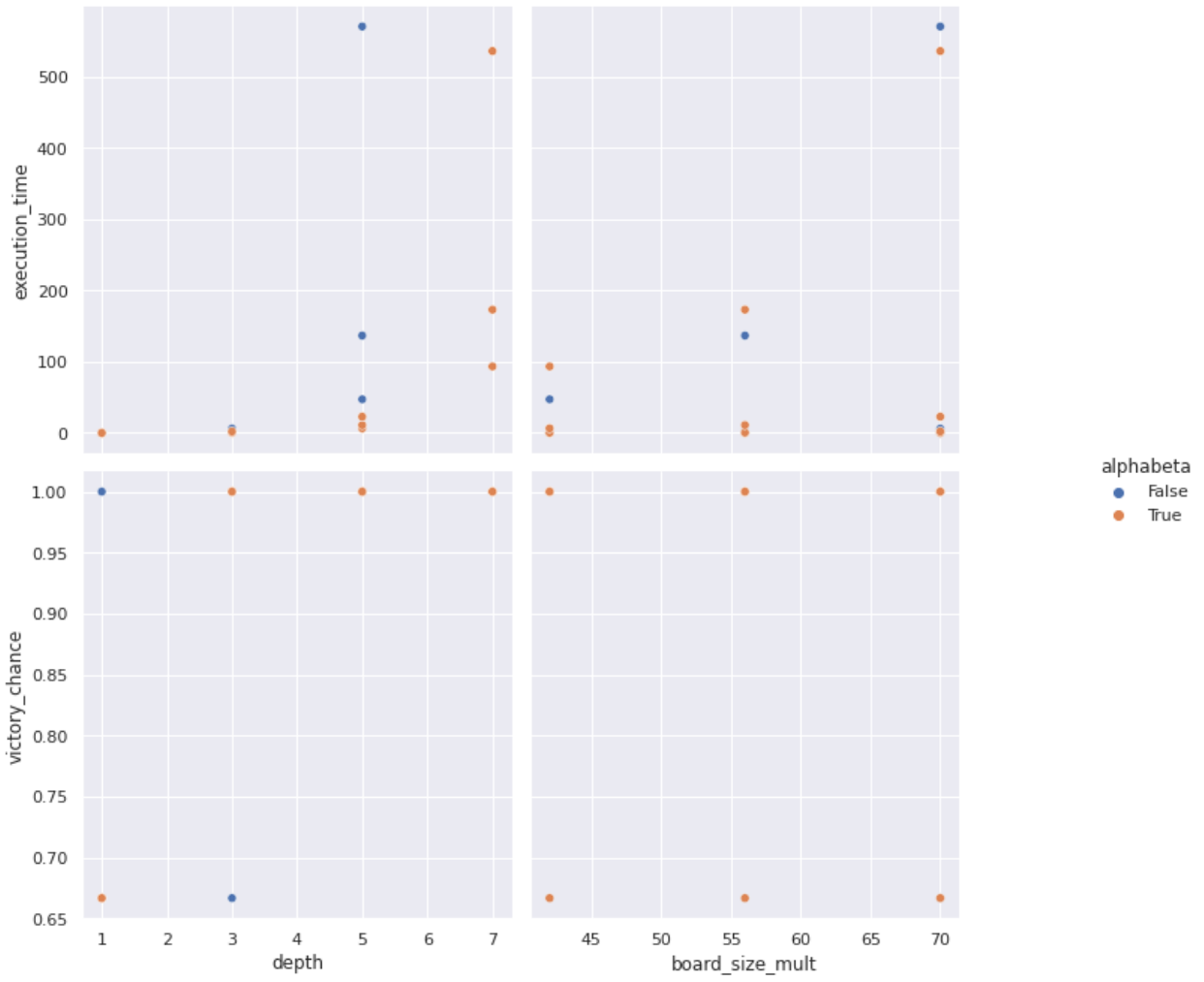
دو متد min_max و min_max_alphabeta برای پیاده سازی الگوریتم minmax تعریف شده که همانطورکه از اسمشان پیداست تابع min_max_alphabeta هرس alpha , beta را نیز اجرا میکند و سرعت بیشتری دارد. این توابع با توجه به عمقی که به آنها داده میشود به صورت recursive نودهای درخت تصمیم‌گیری را تشکیل میدهند و مسیر با بیشترین امتیاز ممکن برای بازیکن و کمترین امتیاز برای حریف را پیدا می‌کنیم.

برای اینکه کد ما کلی تر باشد دو متغیر به کلاس ConnectSin به صورت دستی پیش از فراخوانی تابع run به آبجکت ساخته شده از این کلاس باید بدهیم، یکی مقدار depth (عمق) مورد نظر و دیگری use_alphabeta که مشخص میکند که آیا میخواهیم از هرس استفاده کنیم یا خیر.

- هرس **alpha** و **beta** (سوال ۲)

هنگامی که از هرس alpha و beta استفاده میکنیم در ابتدا این دو مقدار را به ترتیب منفی بینهایت و مثبت بینهایت قرار می‌دهیم، پارامتر آلفا با مقایسه با بیشترین امتیاز بازیکن و پارامتر بتا با کمترین امتیاز حریف مشخص می‌شود و اگر پارامتر آلفا از بتا بیشتر شد یعنی حرکت ما بهتر از حرکت بهینه‌ی حریف خواهد بود و آن حرکت انتخاب شده و در درخت تصمیم گیری قرار می‌گیرد.

نتایج



board_size	depth	alphabeta	steps	victory_chance	execution_time	board_size_mult
(6, 7)	1	FALSE	3	1	0.03971147537	42
(6, 7)	3	FALSE	3	0.6666666667	1.458905697	42
(6, 7)	5	FALSE	3	1	47.27308416	42
(7, 8)	1	FALSE	3	0.6666666667	0.04305028915	56
(7, 8)	3	FALSE	3	1	2.728847027	56
(7, 8)	5	FALSE	3	1	136.6059222	56
(7, 10)	1	FALSE	3	0.6666666667	0.07495546341	70
(7, 10)	3	FALSE	3	1	6.215614796	70
(7, 10)	5	FALSE	3	1	569.8818951	70
(6, 7)	1	TRUE	3	0.6666666667	0.05291795731	42
(6, 7)	3	TRUE	3	1	0.4395396709	42
(6, 7)	5	TRUE	3	1	6.548993349	42
(6, 7)	7	TRUE	3	1	93.30747938	42
(7, 8)	1	TRUE	3	0.6666666667	0.03943681717	56
(7, 8)	3	TRUE	3	1	0.6098015308	56
(7, 8)	5	TRUE	3	1	11.13418078	56
(7, 8)	7	TRUE	3	1	172.9335907	56
(7, 10)	1	TRUE	3	0.6666666667	0.05987286568	70
(7, 10)	3	TRUE	3	1	2.212001085	70
(7, 10)	5	TRUE	3	1	22.8338084	70

					2	
(7, 10)	7	TRUE	3	1	535.4146729	70

- **تحلیل نتایج (سوال ۳)**

با توجه به شکل و جدول مشاهده میکنیم که عمق الگوریتم بر روی شانس پیروزی تاثیر مثبت داشته مخصوصا زمانی که از هرس استفاده می‌کنیم، البته نتایج ما با توجه به اینکه اجرای کد زمان بر است بر روی ۳ ران در هر یک از شرایط ذکر شده گزارش شده و اگر تعداد ران‌ها بالا برود نتایج قابل اعتمادتری می‌توان کسب نمود.