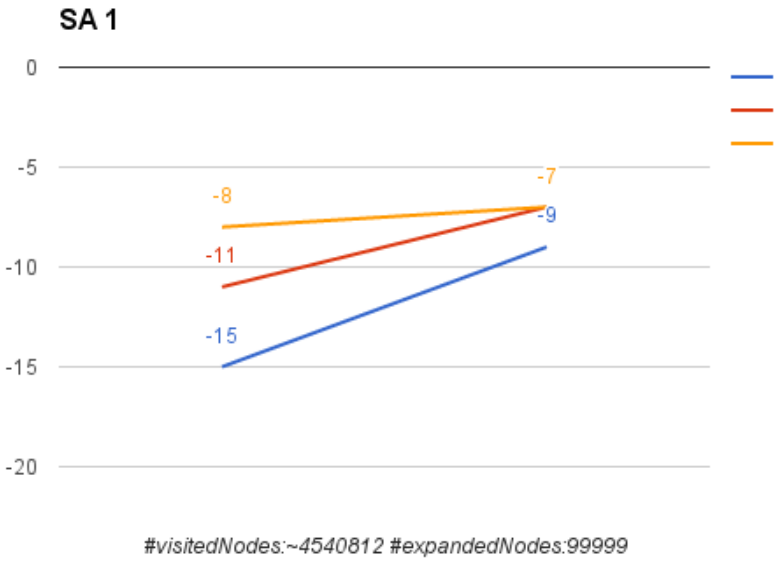
گزارش پروژه دوم برنامه نویسی درس هوش مصنوعی حجت ایمانی ۹۲۳۱۰۶۲

مساله ۸ وزیر را به دو صورت مدل کردم. یکی به این صورت که وزیرها بتوانند در هر حالتی قرار بگیرند (هر وزیر در یک خانه) و حالات بعدی با انتخاب یک وزیر و حرکت آن به یکی از خانه های مجاور ایجاد شود. دوم به این صورت که هیچ دو وزیری نتوانند در یک ستون قرار بگیرند و حالات بعدی با انتخاب یک وزیر و حرکت آن به هر یک از خانه های ستون خود ایجاد شود. بطور کلی تمام الگوریتم ها با روش دوم پاسخ های بهتری را بدست می‌آوردند. در ادامه نتایج مدل کردن با روش اول آورده شده است.

برای هر کدام از الگوریتم‌ها برنامه سه مرتبه اجرا شده و نمودارها ارزش جواب اولیه و جواب نهایی را در هر بار اجرای برنامه نشان می‌دهند. جواب بهینه رسیدن به ارزش صفر است.

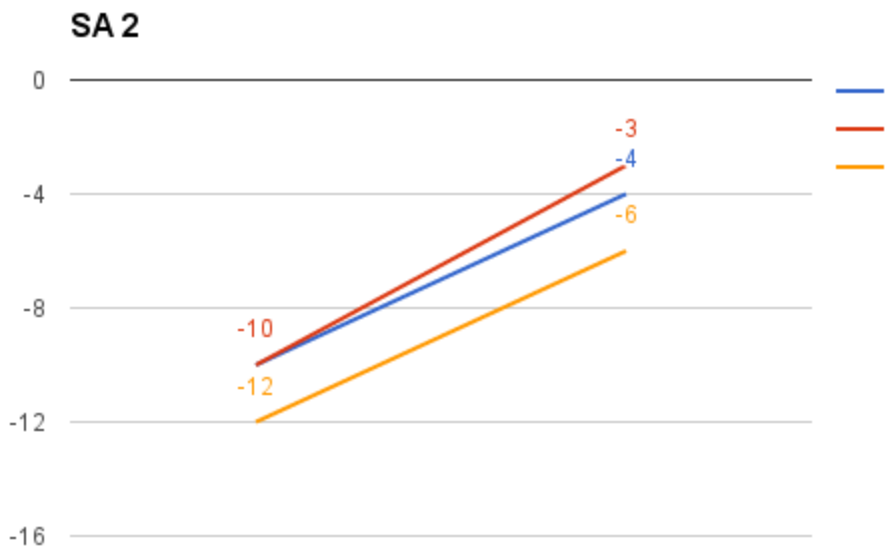
حل مساله ۸ وزیر با simulated annealing :
از سه روش مختلف برای کاهش دما استفاده کردم.

۱. تابع T = Tmax - time
با این تابع دما از یک مقدار مشخص به صورت خطی و با شیب ثابت کم می‌شود. نتیجه بصورت زیر بود. نمودار زیر نتیجه ۳ مرتبه اجرای الگوریتم sa با این تابع سرد کردن با دمای اولیه ۱۰۰۰۰۰ میباشد. که همانطور که مشخص است جوابهای خوبی بدست نمیدهد.



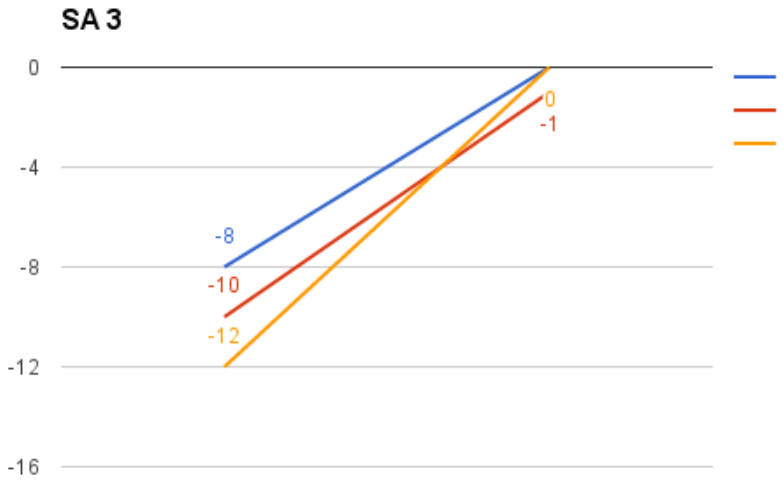
#visitedNodes:~4540812 #expandedNodes:99999

۲. تابع T = Tmax / time
نتایج ۳ مرتبه اجرای الگوریتم sa با این تابع سرد کردن بصورت زیر است. مشخص است که با این تابع الگوریتم نتایج بهتری بدست می‌دهد اما تقریبا هر دفعه در حوالی ۴- همگرا شده و به جواب نهایی نمی‌رسد.



#visitedNodes:~4963456 #expandedNodes:100000

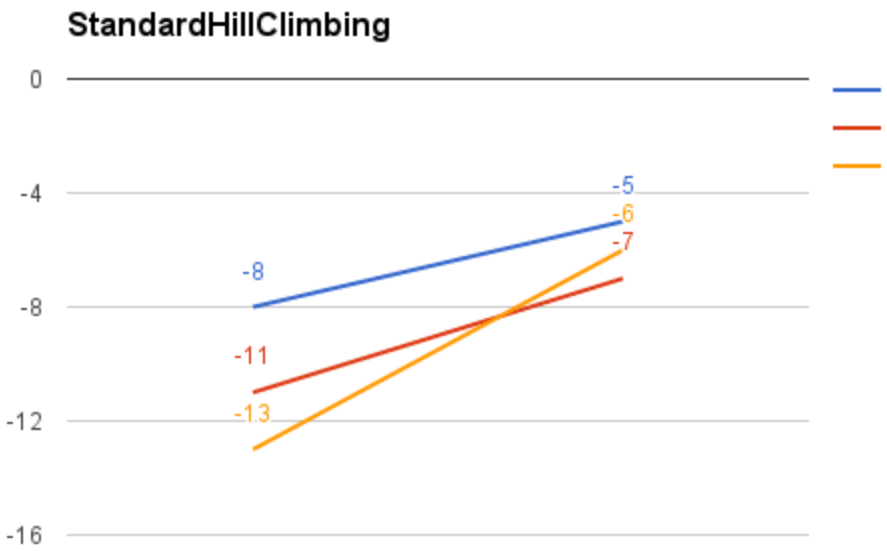
۳. بعنوان تابع سوم از یک تابع که بر مبنای stablizing کار می‌کند استفاده کردم. این تابع هر دما را چندین مرتبه به الگوریتم می‌دهد و این تعداد را با کاهش دما افزایش می‌دهد. (دماهای پایین تر تعداد دفعات بیشتری به الگوریتم داده میشوند) برای آنکه این تابع را بتوانم به ازای پارامتر های مختلف stablizing و کاهش دما آزمایش کنم آن را بصورت یک کلاس قابل پارامتریزه شدن پیاده‌سازی کردم. (الگوریتم کاهش دما در کلاس SimulatedAnnealing.StabilizerSchedule پیاده شده است.)
نمودار زیر نتیجه الگوریتم sa را به ازای دمای اولیه 35 و ضریب stablize اولیه 35 نمایش می‌دهد. همان طور که مشخص است این روش گاهی اوقات جواب بهینه را بدست می‌آورد. (با روش بهینه تر مدل کردن مساله این روش در اکثر اوقات جواب بهینه را بدست می‌آورد.)



#visitedNodes:~6955206 #expandedNodes:~150387

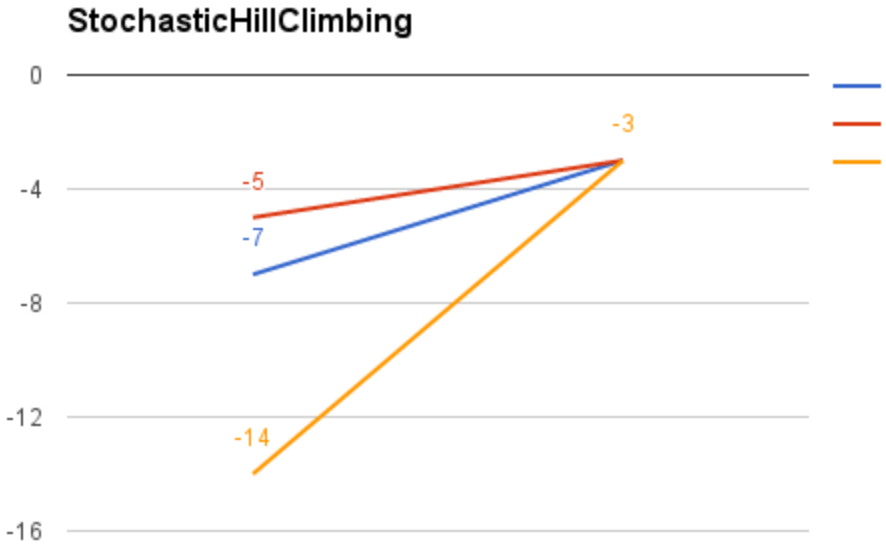
حل مساله ۸ وزیر با استفاده از نسخه های مختلف تپه نوردی

۱. تپه نوردی استاندارد



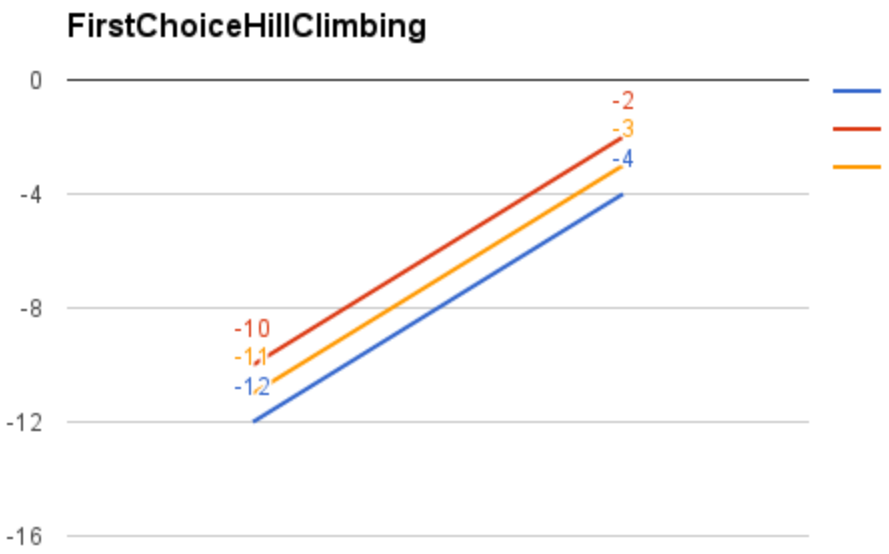
#visitedNodes:~96 #expandedNodes:~2

۲. تپه نوردی تصادفی



#visitedNodes:~250 #expandedNodes:~3

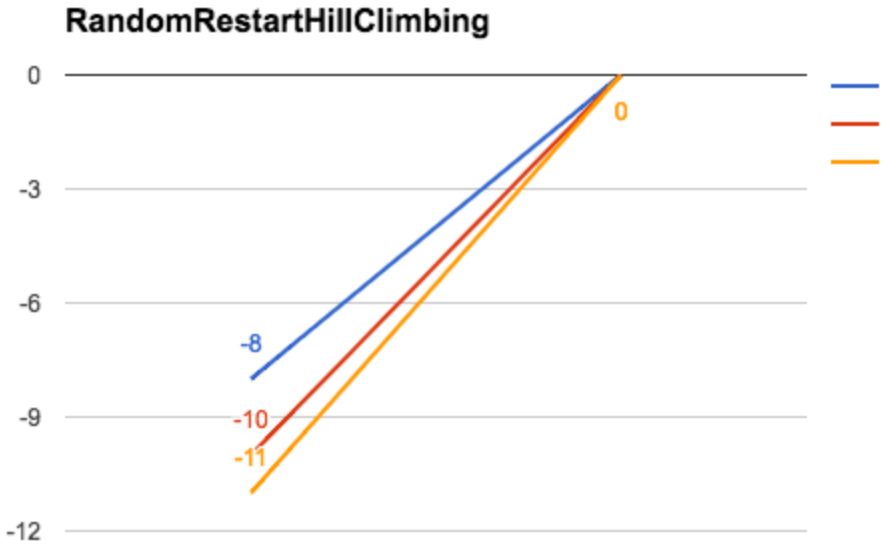
۳. تپه نوردی اولین انتخاب



#visitedNodes:~107 #expandedNodes:~6

۴. تپه نوردی با شروع مجدد تصادفی

نتایج زیر مربوط به حالتی است که تعداد دفعات شروع مجدد محدود نباشد. در اینصورت الگوریتم همواره جواب بهینه را بدست می‌آورد ولی تعداد نودهای مشاهده شده و بسط داده شده خیلی زیاد است. اگر تعداد دفعات شروع مجدد را محدود کنیم بسته به تعداد پاسخ‌هایی بین رنج پاسخ تپه‌نوردی استاندارد تا جواب بهینه را بدست می‌آوریم.



#visitedNodes:~336269 #expandedNodes:~5041

مقایسه و نتیجه‌گیری:

در بین الگوریتم‌های مطرح شده الگوریتم SimulatedAnnealing اگر با تابع کاهش دمای مناسب استفاده شده و زمان کافی به آن داده شود می‌تواند جواب بهینه را پیدا کند. همچینین RandomRestartHillClimbing اگر به اندازه کافی عمل شروع مجدد را انجام دهد به جواب بهینه می‌رسد. سایر الگوریتم های HillClimbing معمولا به جواب بهینه نمی‌رسند و به یک ماکسیمم محلی همگرا می‌شوند و جواب‌های نه چندان خوبی بدست می‌دهند. البته جواب‌هایی که نسخه تصادفی و اولین انتخاب تپه نوردی بدست می‌آورند به نسبت از تپه نوردی استاندارد بهتر است. نکته قابل قیاس دیگر تعداد گره‌های مشاهده و گسترش داده شده است که برای SA و RandomRestartHillClimbing این مقدار خیلی بالاست. اما این اعداد برای نسخه‌های دیگر تپه نوردی بسیار کم است که نشان می‌دهد تپه نوردی خیلی زود همگرا می‌شود.