عامل های حل‌کننده مسائل
NP Complete

ارائه دهندگان: مهدی نائینی و بابک حسینی محتشم و محمد طاها مجلسی
اساتید: جناب آقای دکتر محمد جواد دوستی و یدالله یعقوب زاده

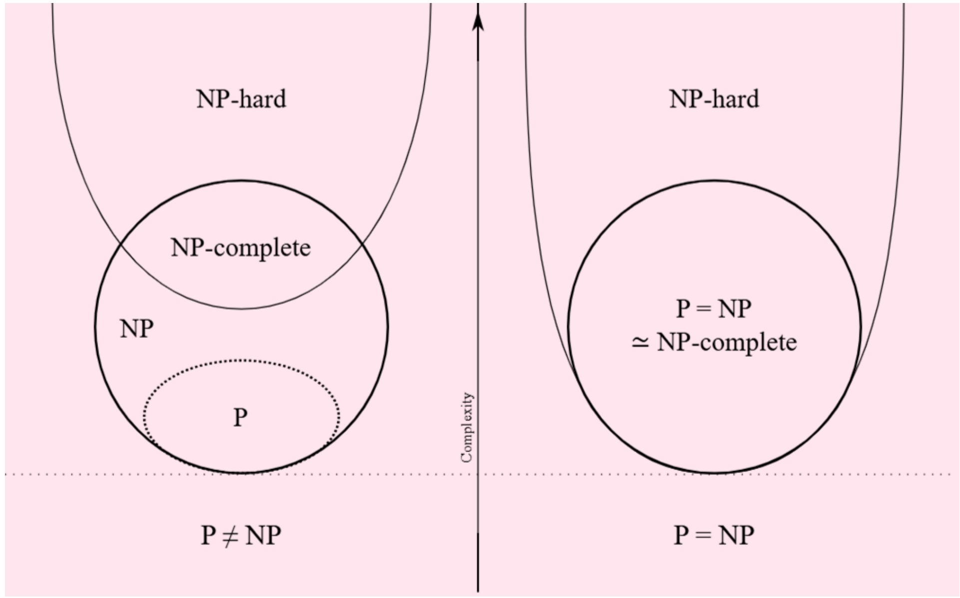
شهریور ۱۴۰۴

فهرست مطالب

عامل های حل کننده مسائل
NP Complete

مسائل NP:

مسائل NP یا (Nondeterministic Polynomial time) دسته ای از مسائل در علوم کامپیوتر هستند که بررسی صحت یک جواب پیشنهادی برای آن‌ها در زمان چندجمله ای (Polynomial time) امکان پذیر است ولی پیدا کردن خود جواب ممکن است بسیار سخت و زمان بر باشد. به عبارت دیگر اگر کسی یک پاسخ برای چنین مسائلی به ما بدهد، می‌توانیم به‌سرعت (در زمان چندجمله ای) بررسی کنیم که آیا آن پاسخ درست است یا خیر. بسیاری از مسائل معروف مانند مسئله فروشنده دوره‌گرد و مسئله 3SAT در این دسته قرار دارند. اهمیت مسائل NP در این است که ارتباط نزدیکی با مفهوم NP-Complete دارند. یعنی مسائلی که اگر حتی یکی از آن‌ها در زمان چندجمله‌ای حل شود، تمام مسائل NP نیز به‌طور کارآمد حل خواهند شد.



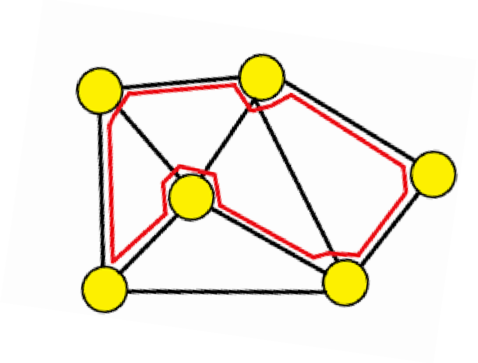
Comparison between P, NP, NP-Hard and NP-Complete

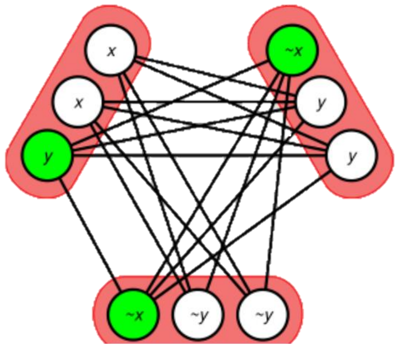
Problem Type	Verifiable in P time	Solvable in P time
P	Yes	Yes
NP	Yes	Yes or No *
NP-Complete	Yes	Unknown
NP-Hard	Yes or No **	Unknown

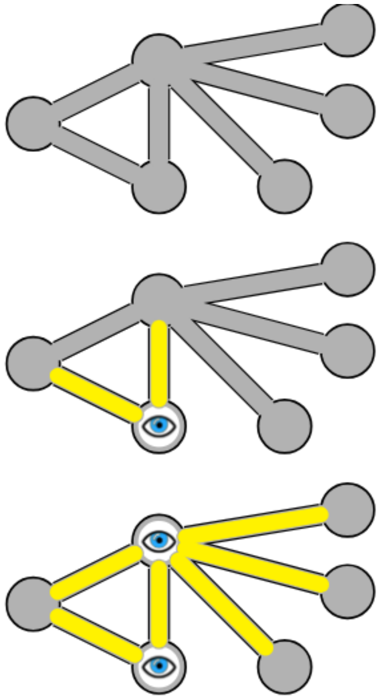
- * An NP problem that is also P is solvable in Polynomial time.
- ** An NP-Hard problem that is also NP-Complete is verifiable in P time.

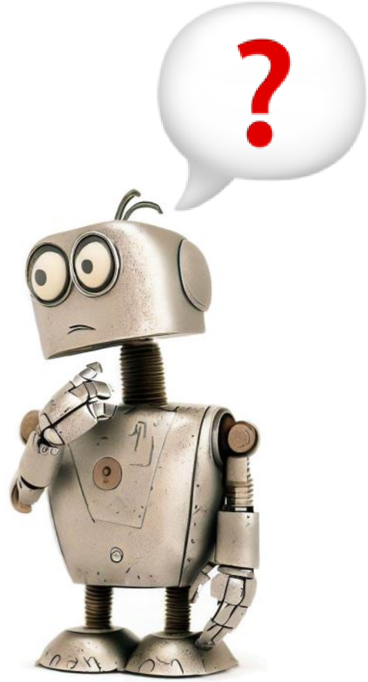
حل مسائل NP:

مدل‌های زبانی بزرگ (LLMs)در سال‌های اخیر توانایی بالایی در تشخیص الگو و استدلال نشان داده‌اند. در این پروژه، ما به بررسی امکان استفاده از این مدل‌ها برای حل مسائل NP-Complete مانند مسیر همیلتونی (HAM-PATH) و مسئله 3SAT می‌پردازیم. همچنین روش‌هایی برای بهبود عملکرد آن‌ها بررسی می‌کنیم از جمله استفاده از ابزارهای کمکی و ریزتنظیم مدل های کوچک‌تر. نتایج نشان می‌دهد که اگرچه مدل های آماده در حل این مسائل با محدودیت مواجه اند، اما می توانند چشم‌اندازی نو برای حل مسائل محاسباتی پیچیده و توسعه حل‌کننده های قدرتمندتر فراهم کنند.









حل مسائل NP:

بسیاری از چالش های مهم محاسباتی در حوزه‌هایی مانند لجستیک، زمان‌بندی، راستی‌آزمایی سخت‌افزار و تخصیص منابع در دسته ی مسائل NP-Complete قرار می‌گیرند. این مسائل به دلیل رشد نمایی پیچیدگی با افزایش اندازه‌ی ورودی، به سختی حل می‌شوند.

اهمیت عملی مسائل NP-Complete باعث شده است که ابزارهای پیشرفته‌ای مانند SMT Solver ها مانند Z3 توسعه پیدا کنند. این ابزارها، که حتی در زبان‌هایی مثل Python نیز یکپارچه‌سازی شده‌اند، حل نمونه‌های پیچیده را ساده‌تر می‌سازند. با این حال، علی‌رغم کارایی بالا و سربار کم، پیچیدگی زمانی آن‌ها همچنان نمایی باقی می‌ماند.

هدف این پروژه، ابتدا تحلیل توانایی LLM ها در حل مسائل NP-Complete و سپس بهبود آن‌ها با روش‌های مختلف است؛ از جمله تجهیز عامل‌های زبانی به ابزارهای مفید یا ریزتنظیم مدل‌های کوچک‌تر. فرضیه‌ی ما این است که مدل‌های زبانی بزرگ به دلیل قابلیت تشخیص الگو و استنتاج در زمان چندجمله‌ای می‌توانند یاد بگیرند که راهبردها و روش‌های ابتکاری موثری بسازند.

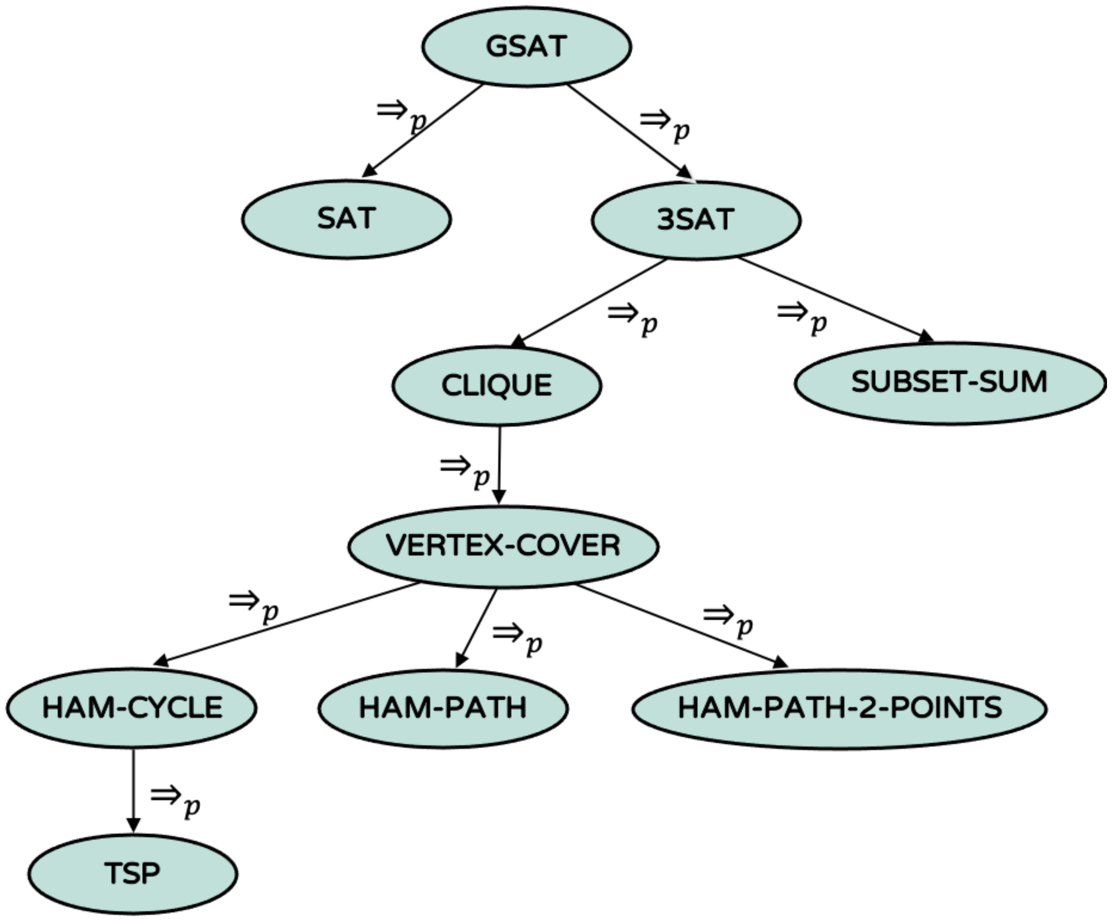
پرسش‌های اصلی پژوهش ما:

- تا چه حد LLM های پیشرفته قادرند مستقیماً نمونه‌هایی از مسائل NP-Complete را حل کنند؟
- چه رابطه‌ای میان ویژگی‌های کلیدی یک نمونه از مسئله و احتمال حل درست آن توسط LLM وجود دارد؟
- آیا در برخی دسته‌های خاص از مسائل، LLM ها می‌توانند سریع‌تر از حل‌کننده‌های بهینه‌ای مثل Z3 به پاسخ برسند؟
- آیا ریزتنظیم یک مدل زبانی عمومی روی مجموعه‌ای از مسائل NP-Complete باعث افزایش تعمیم‌پذیری آن در نمونه‌های جدید می‌شود و می‌تواند به ایجاد یک حل‌کننده تخصصی و کارآمدتر بینجامد؟

مسائل NP-Complete:

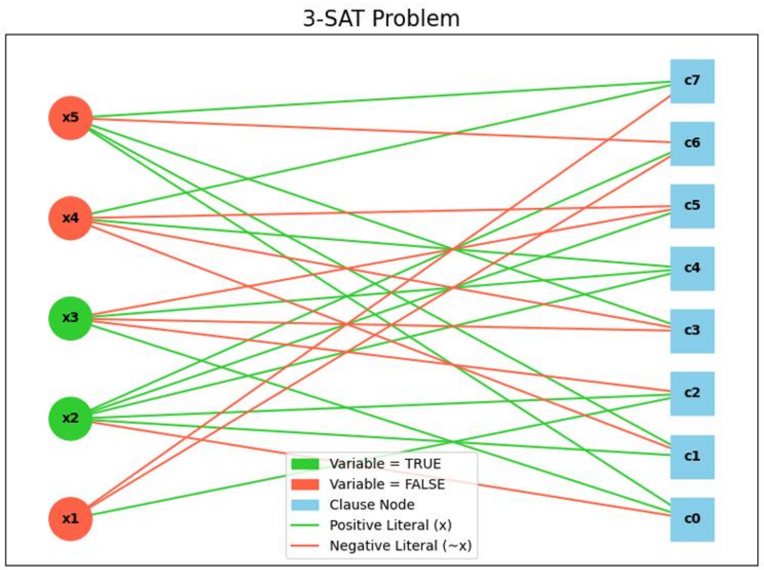
در نظریه پیچیدگی محاسباتی، کلا NP شامل مسائلی است که برای آن‌ها، یک راه حل پیشنهادی می‌تواند در زمان چندجمله ای توسط یک ماشین تورینگ قطعی بررسی شود. یک مسئله زمانی به عنوان NP_Complete (NPC) طبقه‌بندی می‌شود که هم در NP باشد و هم هر مسئله دیگری در NP بتواند در زمان چندجمله ای به آن کاهش یابد. این ویژگی باعث می‌شود مسائل NP-Complete سخت ترین مسائل محاسباتی در کلاس NP باشند. به این معنا که یافتن یک الگوریتم چندجمله ای برای حتی یک مسئله NP_Complete، به معنای وجود الگوریتم چندجمله ای برای همه مسائل NP خواهد بود.

در ادامه ی این ارائه ما شش مسئله NP_Complete را که برای پژوهش خود انتخاب کرده ایم را معرفی خواهیم کرد.



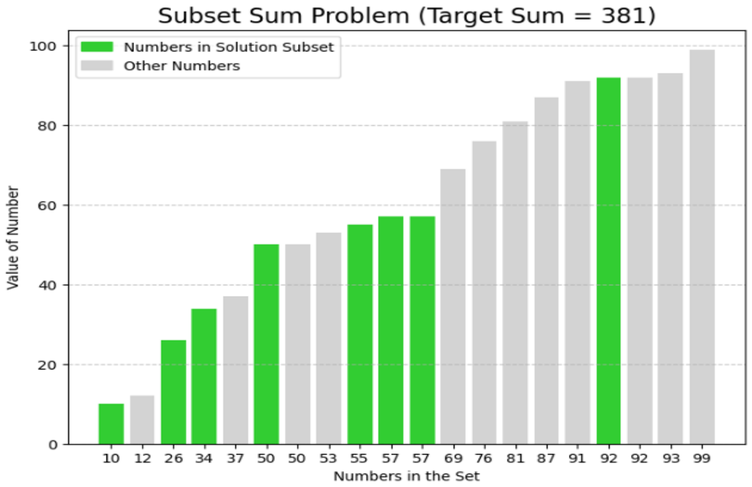
۱. مسئله 3SAT:

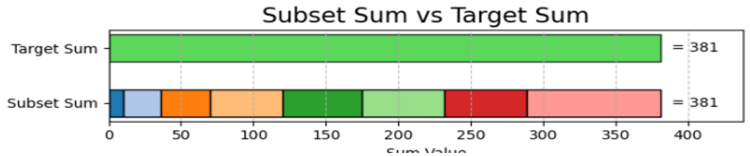
مسئله 3SAT اولین مسئله‌ای است که توسط قضیه کوک لوین به عنوان NP-Complete اثبات شد و به همین دلیل یکی از پایه‌های نظریه پیچیدگی محاسباتی به شمار می‌رود. این مسئله به این شکل تعریف می‌شود که با توجه به یک فرمول بولی ϕ شامل متغیرها و عملگرهای AND (∧)، OR (∨) و NOT (¬)، آیا تخصیصی از مقادیر TRUE و FALSE به متغیرها {X1, X2, ..., Xn} وجود دارد که کل فرمول ϕ را صحیح کند؟ اگر چنین تخصیصی موجود باشد، فرمول قابل رضایت است. در غیر این صورت، غیرقابل رضایت است.



۲. مسئله Subset Sum:

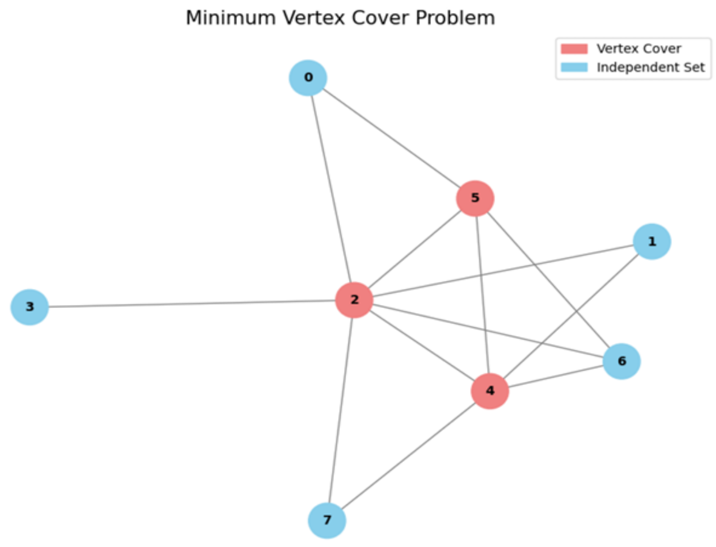
مسئله مجموع زیرمجموعه ها (Subset Sum Problem – SSP) یکی دیگر از مسائل NP-Complete است. این مسئله به این صورت تعریف می‌شود: با توجه به یک مجموعه (یا چندمجموعه) از اعداد صحیح S={a1, a2, ..., an} و یک عدد هدف T، آیا زیرمجموعه‌ای S′⊆S وجود دارد که مجموع عناصر آن دقیقا برابر با T باشد؟





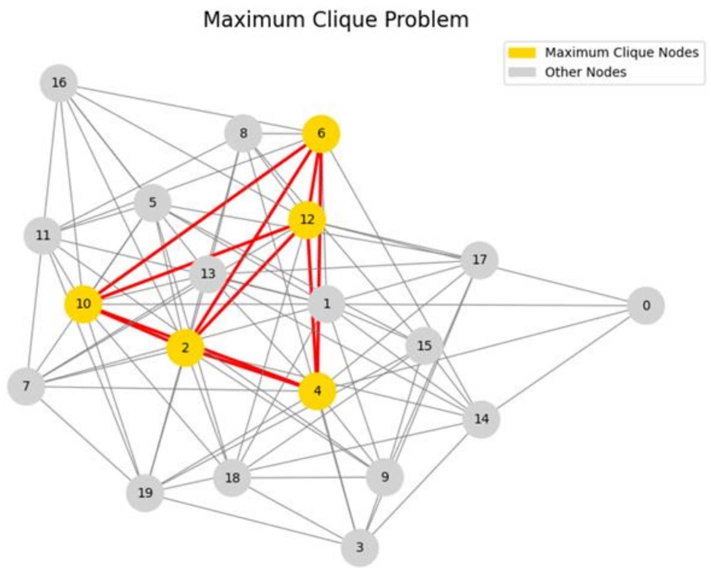
۳. مسئله Minimum Vertex Cover:

در نظریه گراف ها، مسئله پوشش راس (Vertex Cover) یعنی یک گراف بدون جهت $G = (V,E)$ با مجموعه ای از رئوس $C \subseteq V$ است به طوری که برای هر یال $(u, v) \in E$ ، حداقل یکی از رئوس انتهایی آن در C قرار داشته باشد. به زبان ساده مجموعه C تمام یال‌های گراف را پوشش می‌دهد.



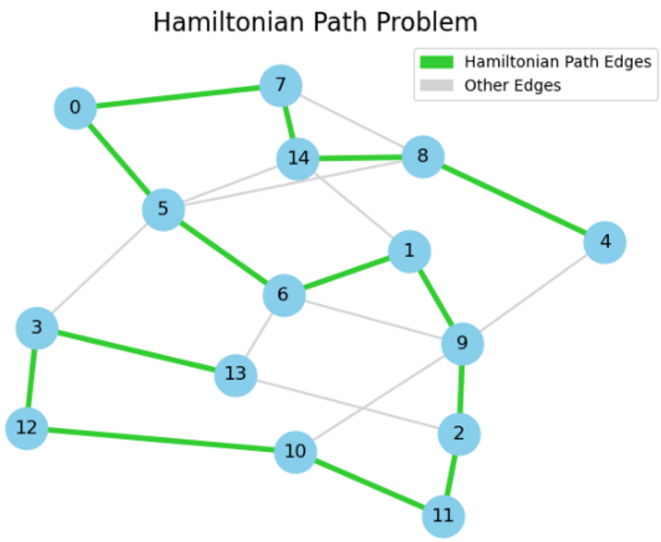
۴. مسئله Maximum Clique:

در نظریه گراف ها، کلیک (Clique) زیرمجموعه ای از رئوس یک گراف بدون جهت $G = (V,E)$ است که هر دو رأس متمایز در آن به هم متصل باشند. یعنی زیرگراف القا شده کامل است. کلیکی با اندازه k معمولا به نام k-Clique شناخته می‌شود. مسئله به این صورت تعریف می‌شود که با توجه به یک گراف بدون جهت $G = (V,E)$ و یک عدد صحیح مثبت $k \leq |V|$، آیا مجموعه ای از رئوس $C \subseteq V$ با اندازه $|C| \geq k$ وجود دارد که یک کلیک تشکیل دهد؟



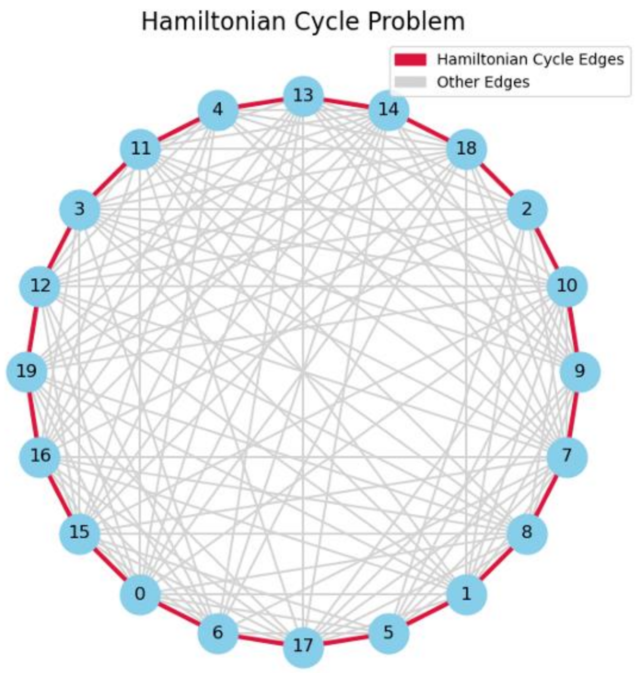
۵. مسئله Hamiltonian Path :

در نظریه گراف ها، مسیر همیلتونی (Hamiltonian Path) مسیری در یک گراف بدون جهت یا جهت‌دار G=(V,E) است که هر راس دقیقا یک بار بازدید شود. تعیین وجود چنین مسیری یکی از مسائل کلاسیک و دشوار محاسباتی است. مسئله به این صورت تعریف می‌شود که با توجه به گراف G=(V,E)، آیا ترتیب رئوسی v1 تا vn وجود دارد که هر رأس دقیقا یک‌بار در آن ظاهر شود و هر جفت متوالی رئوس (vi, vi+1) یک یال در E باشد؟



۶. مسئله Hamiltonian Cycle :

مسئله چرخه همیلتونی (Hamiltonian Cycle) مسئله ای نزدیک به مسیر همیلتونی است. در نظریه گراف ها، چرخه همیلتونی مسیری در گراف بدون جهت یا جهت‌دار G=(V,E) است که هر رأس دقیقا یک بار بازدید شود و در نهایت به رأس شروع بازگردد. مسئله به این صورت تعریف می‌شود که با توجه به گراف G=(V,E)، آیا چرخه‌ای وجود دارد که هر رأس در V را دقیقا یک‌بار شامل شود (به جز رأس شروع و پایان که یکسان‌اند)؟



تولید داده‌ها برای مسائل NP-Complete:

یکی از چالش‌های اصلی در ارزیابی مدل‌های زبانی بزرگ روی مسائل NP-Complete، کمبود داده‌های معیار بزرگ و متنوع است. برای هر یک از شش مسئله انتخاب شده، ما مجموعه‌ای از نمونه‌ها به همراه راه‌حل‌های متناظر آن‌ها ایجاد کردیم. توجه داشته باشید که راه‌حل ایجاد شده الزاما تنها راه‌حل مسئله نیست. برای اطمینان از قابل حل بودن نمونه‌ها، ابتدا یک راه‌حل تصادفی ساخته شده و سپس نمونه‌ای تولید می‌شود که توسط آن راه‌حل برآورده شود.

روش تولید نمونه‌ها:

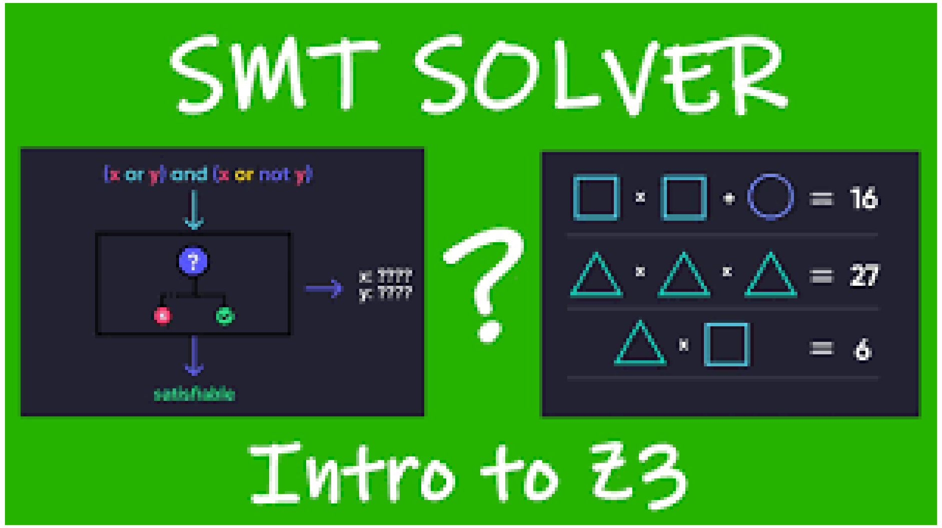
برای هر مسئله، الگوریتم‌های خاصی طراحی شد. به عنوان مثال در 3SAT ابتدا یک تخصیص تصادفی به متغیرها داده شد و سپس هر بند شرطی (clause) طوری تنظیم شد که با راه‌حل مطابقت داشته باشد. در Subset Sum، زیرمجموعه‌ای به عنوان راه‌حل انتخاب شد و اعداد باقی‌مانده تصادفی تولید شدند تا مجموعه کامل شکل بگیرد. برای مسائل گرافی مثل Minimum Vertex Cover، Maximum Clique، Hamiltonian Path و Hamiltonian Cycle، الگوریتم‌ها شامل ایجاد ساختار پایه گراف (backbone) مطابق با راه‌حل و سپس افزودن یال‌های اضافی برای کنترل پیچیدگی و اتصال گراف بودند.

کنترل و پارامترها:

این روش‌ها امکان کنترل دقیق دشواری مسئله و ویژگی‌های نمونه‌ها را از طریق پارامترهایی مانند تعداد متغیرها یا رئوس، نسبت زیرمجموعه‌ها یا کلیک‌ها، چگالی یال‌ها و محدوده اعداد فراهم می‌کنند. همچنین برای هر مسئله، تابعی برای بررسی صحت راه‌حل، تابعی برای نمایش آن و امکان حل نمونه با استفاده از Z3 آماده شد تا ارزیابی مدل‌ها به شکل کامل و استاندارد انجام شود.

Z3 Theorem Prover:

Z3 یک حل‌کننده قدرتمند SMT (Satisfiability Modulo Theories) است که توسط مایکروسافت توسعه یافته و برای حل محدودیت های پیچیده طراحی شده است. این ابزار در زمینه‌هایی مانند راستی‌آزمایی نرم‌افزار، اجرای سمبولیک و تحلیل پیشرفته برنامه‌ها کاربرد دارد. Z3 به عنوان یک حل‌کننده دقیق، تضمین می‌کند که اگر راه‌حلی وجود داشته باشد آن را پیدا کند یا ثابت کند که هیچ راه‌حلی وجود ندارد. برای شش مسئله NP-Complete مورد مطالعه، مدل‌هایی با استفاده از API پایتون Z3 ساخته شد تا محدودیت های هر نمونه به دقت کدگذاری شود. با این حال، پیچیدگی زمانی آن در بدترین حالت همچنان نمایی است. فرضیه اصلی ما این است که LLM ها با قابلیت استنتاج در زمان چندجمله ای می‌توانند نقش حل‌کننده‌های ابتکاری (heuristic) را داشته باشند و افزایش دقت آن‌ها می‌تواند منجر به مدل‌هایی با دقت بالا و زمان اجرا چندجمله‌ای شود.

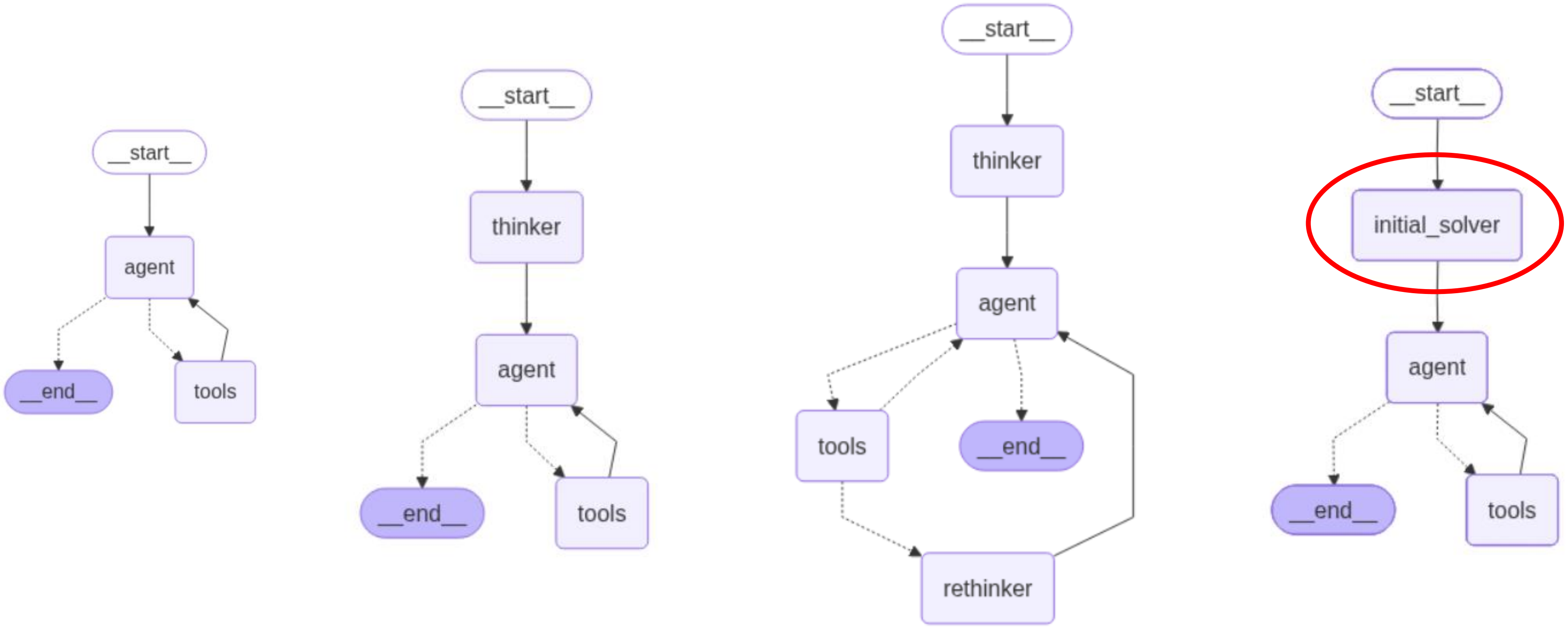


:Base LLM (Gemini 2.5 Flash)

Baseline دوم ما یک مدل زبانی بزرگ پیشرفته است که تعادل مناسبی بین عملکرد، هزینه و سرعت استنتاج دارد. ما مدل Gemini 2.5 Flash را انتخاب کردیم، زیرا مدل‌های قوی‌تر مانند Gemini 2.5 Pro یا GPT-4o با هزینه محاسباتی و تاخیر بسیار بالا مناسب این کار نیستند، در حالی که Z3 می‌تواند مسائل را سریع‌تر حل کند. برای ارزیابی این baseline، از استراتژی Zero-Shot Prompting استفاده شد: برای هر نمونه مسئله، یک دستور طبیعی طراحی شد که مشکل را توصیف کرده و به مدل دستور می‌دهد پاسخ نهایی را در قالب ساختاریافته و قابل پردازش ارائه دهد. عملکرد این baseline نشان می‌دهد که حتی یک LLM عمومی قدرتمند بدون آموزش تخصصی یا ابزار کمکی توانایی حل مسائل محاسباتی پیچیده را دارد.

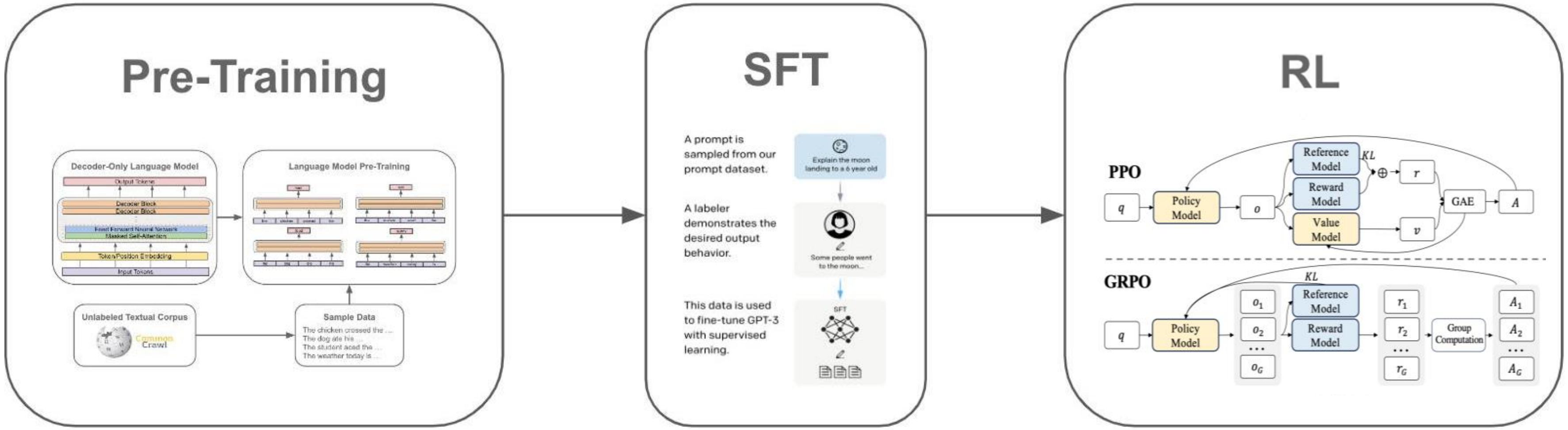
:Agent

پس از ساخت دیتاست مصنوعی همراه با راه‌حل‌های زیر بهینه، هدف ما استفاده از توان مدل‌های زبانی بزرگ (LLMها) در محیط‌های عامل‌محور (Agentic) است تا شباهت پاسخ آن‌ها را با راه‌حل‌های ما اندازه‌گیری کنیم. برای این منظور از API رایگان مدل Gemini به عنوان عامل اصلی استفاده کردیم. با طراحی پرامپت‌های مختلف، ابزارهای ابتکاری و قالب‌بندی متفاوت در اختیار عامل قرار داده شد تا خود تصمیم بگیرد چه زمانی از هر ابزار استفاده کند. پس از تولید پاسخ، صحت آن بررسی می‌شود تا مطمئن شویم راه‌حل معتبر است و حداقل به خوبی راه‌حل زیر بهینه ما عمل می‌کند.

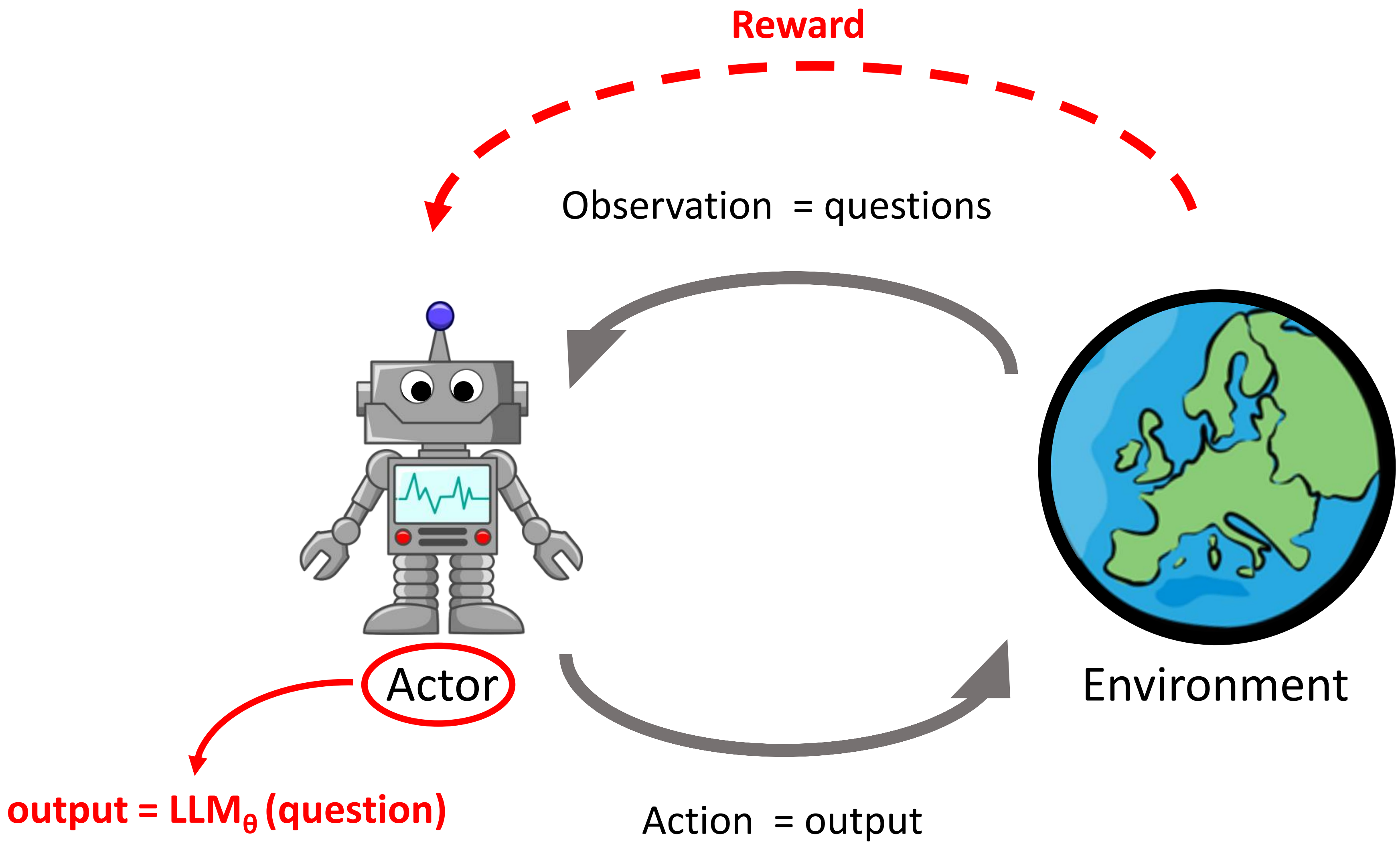


:SFT + RL

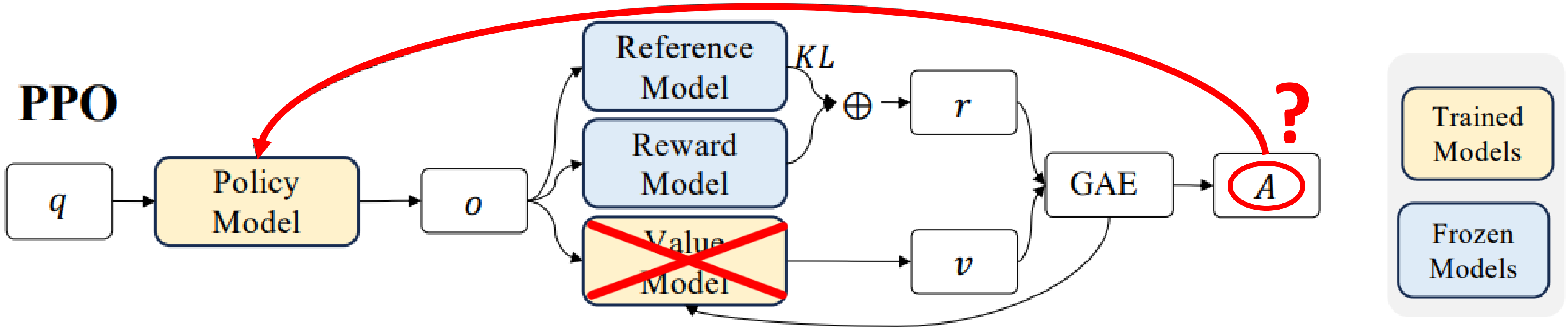
برای افزایش دقت مدل، بدون کاهش سرعت استنتاج یا افزایش تعداد توکن خروجی، روش‌های پرامپتینگ مختلف مانند تشویق به استدلال گام‌به‌گام یا تفکر الگوریتمی آزمایش شدند. همچنین یک مدل کوچکتر با SFT (Supervised Fine-Tuning) برای یادگیری ساختار و فرمت کلی و سپس با RL مانند روشGRPO برای تولید راه‌حل‌های معتبر (نه لزوما مشابه پاسخ ما) آموزش داده شد. برای هر مسئله، ۱۰۰ نمونه با مقادیر متغیر تولید و پرامپت‌های مربوطه به صورت دسته ای به مدل داده شدند. سپس خروجی ها استخراج و صحت آن ها بررسی شد.



Reinforcement learning:



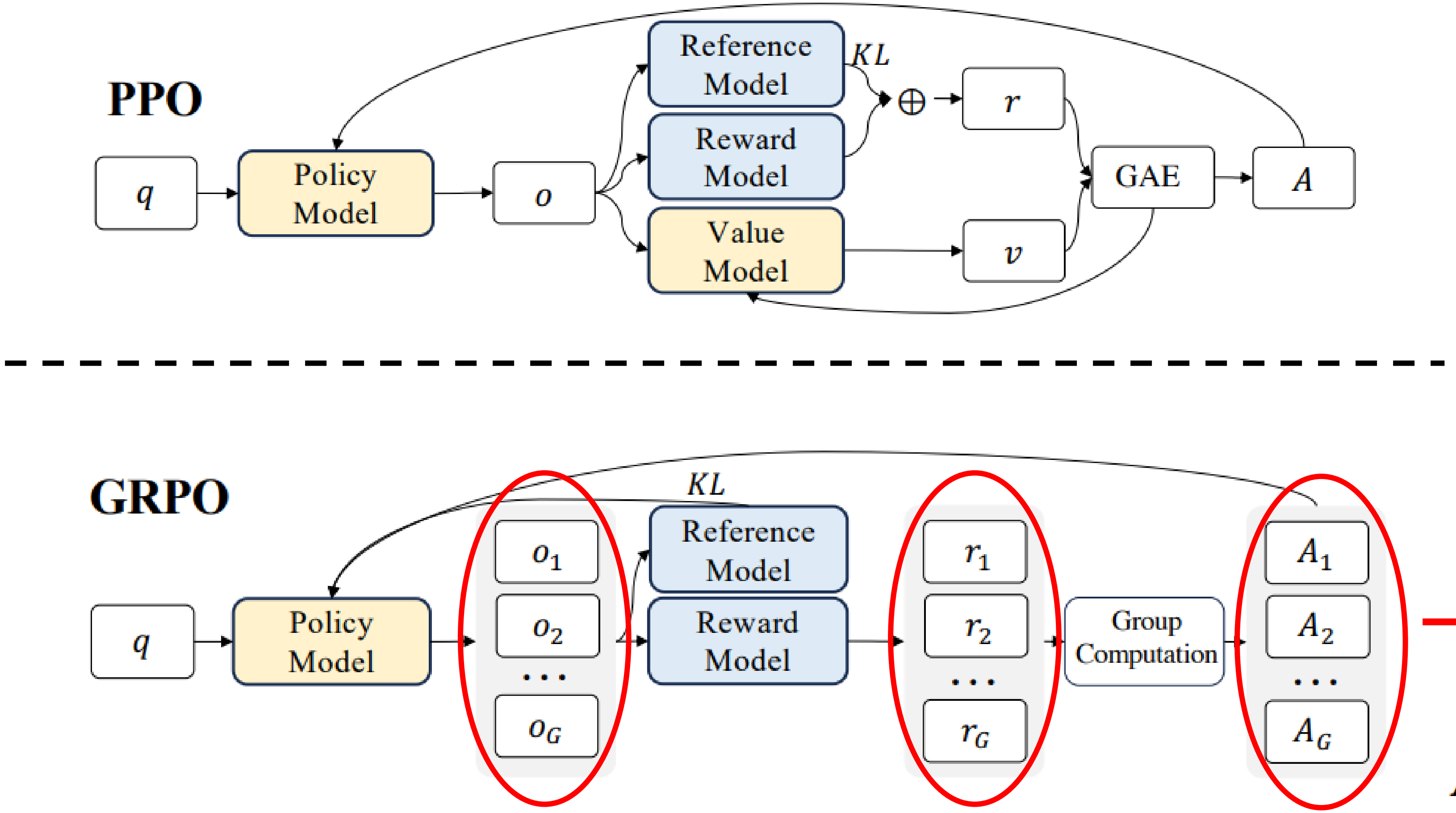
:PPO

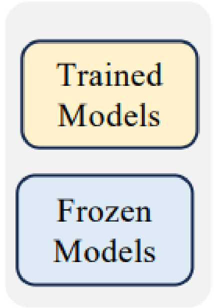


$$\mathcal{J}_{PPO}(\theta) = \mathbb{E}[q \sim P(Q), o \sim \pi_{\theta_{old}}(O|q)] \frac{1}{|o|}\sum_{t=1}^{|o|} \min\left[\frac{\pi_\theta(o_t|q,o_{<t})}{\pi_{\theta_{old}}(o_t|q,o_{<t})}A_t, \text{clip}\left(\frac{\pi_\theta(o_t|q,o_{<t})}{\pi_{\theta_{old}}(o_t|q,o_{<t})}, 1-\varepsilon, 1+\varepsilon\right)A_t\right],$$

$$r_t = r_\varphi(q, o_{\le t}) - \beta \log \frac{\pi_\theta(o_t|q,o_{<t})}{\pi_{ref}(o_t|q,o_{<t})},$$

GRPO:

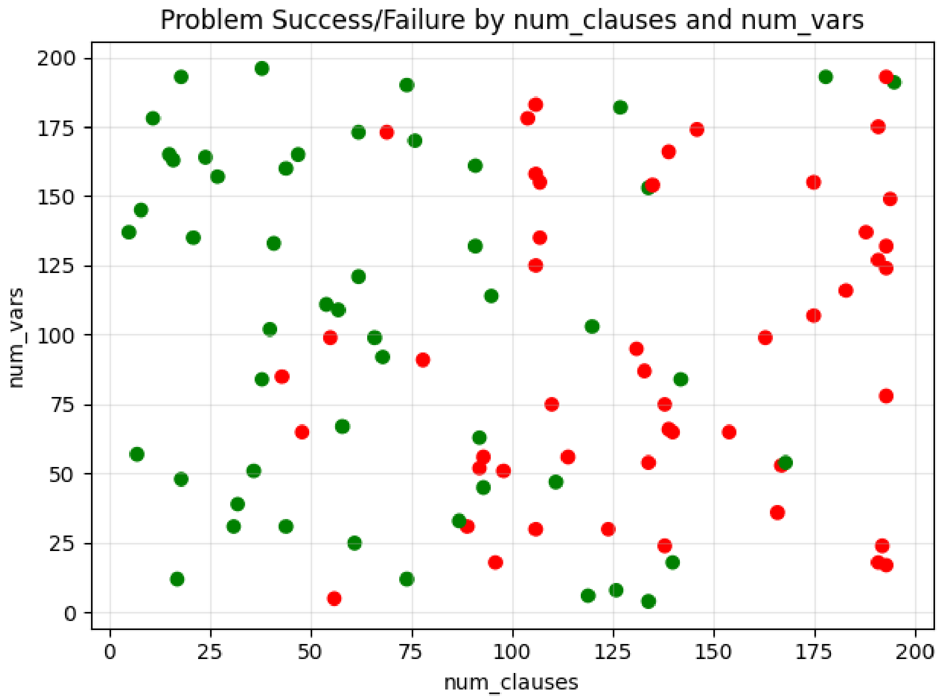


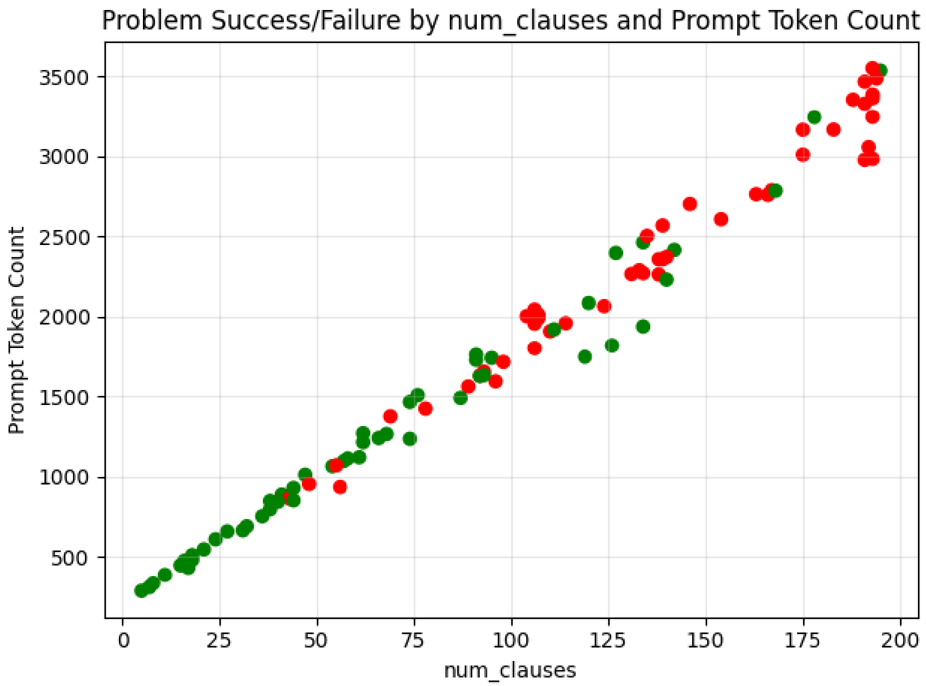


$$\hat{A}_{i,t} = \frac{r_i - \mathrm{mean}(\mathbf{r})}{\mathrm{std}(\mathbf{r})}$$

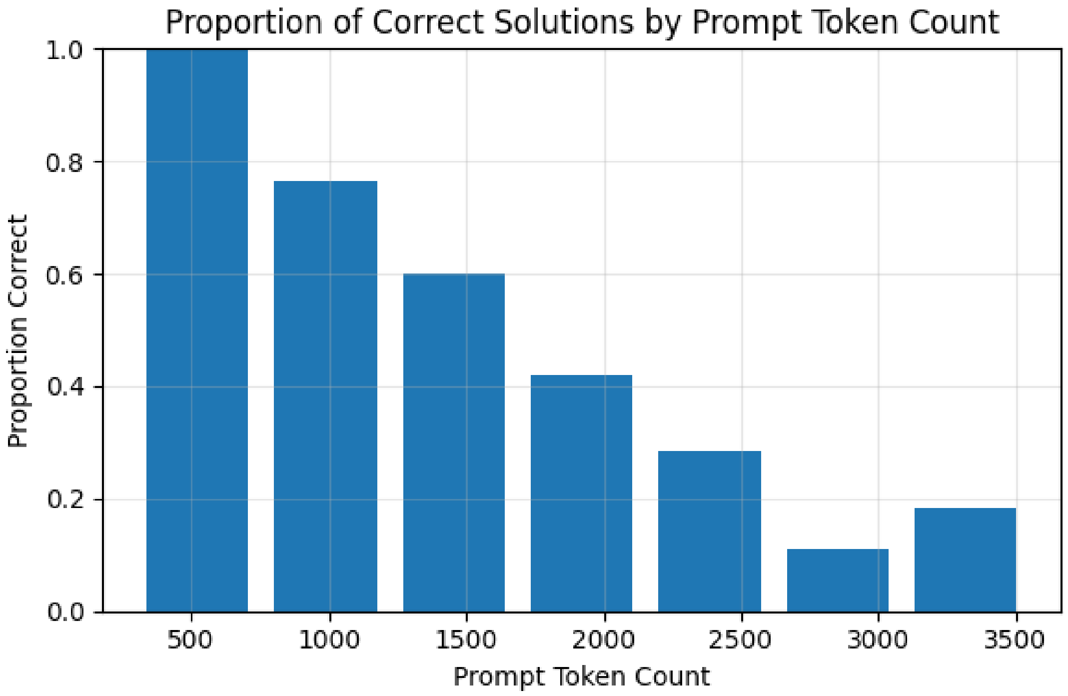
:Base LLM (Gemini 2.5 Flash)

مدل پایه مسائل را با چه دقت و سرعتی می‌تواند حل کند؟





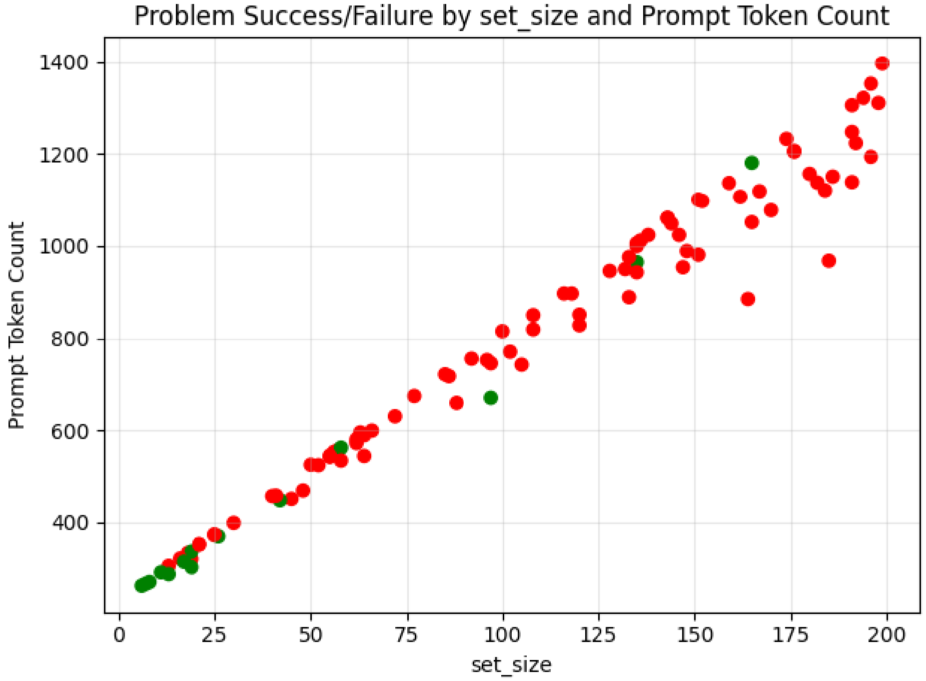
:Base LLM (Gemini 2.5 Flash)
مسئله 3-SAT:

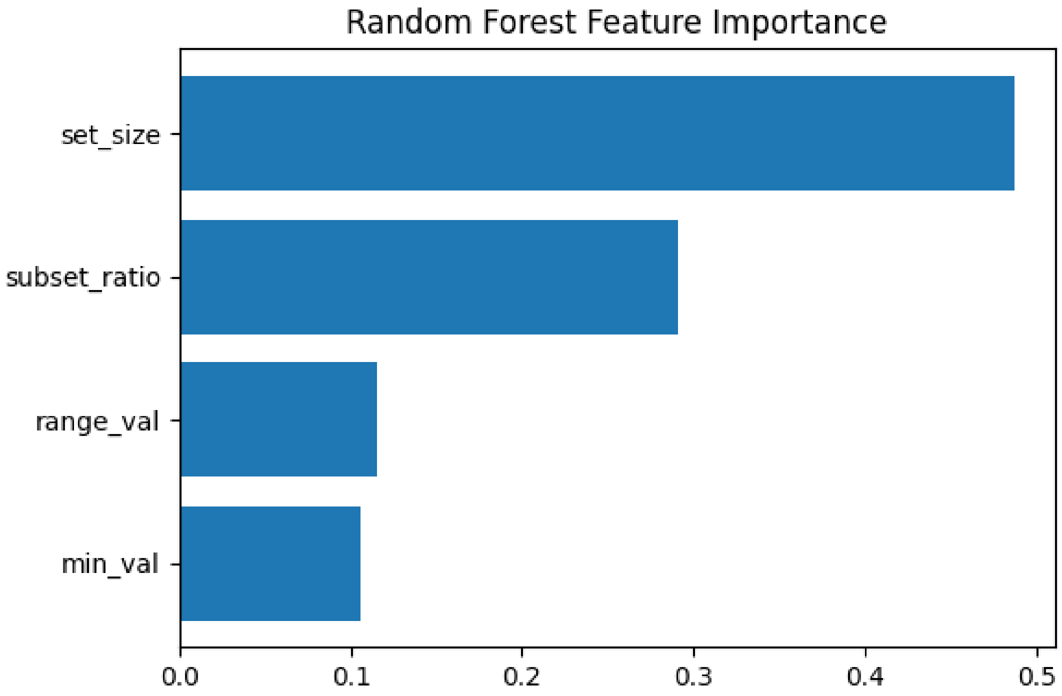


با اینکه تعداد متغیرها تاثیر بر فضای جستجوی پاسخ دارد ولی عملکرد مدل رابطه بیشتری با تعداد کلازها یا همان طول پرامپت دارد!

:Base LLM (Gemini 2.5 Flash)

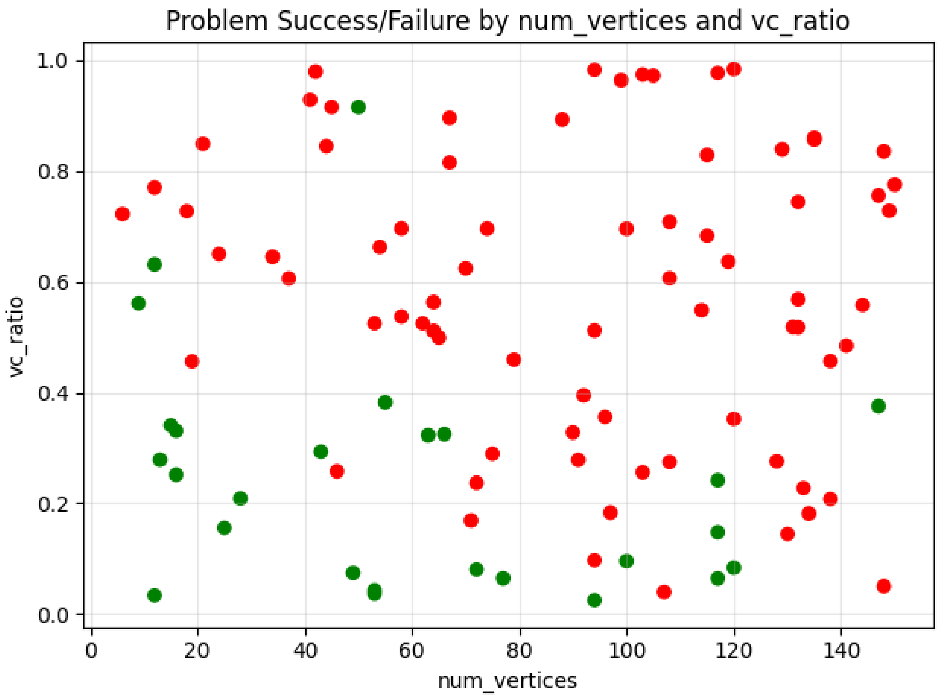
:Subset Sum مسئله

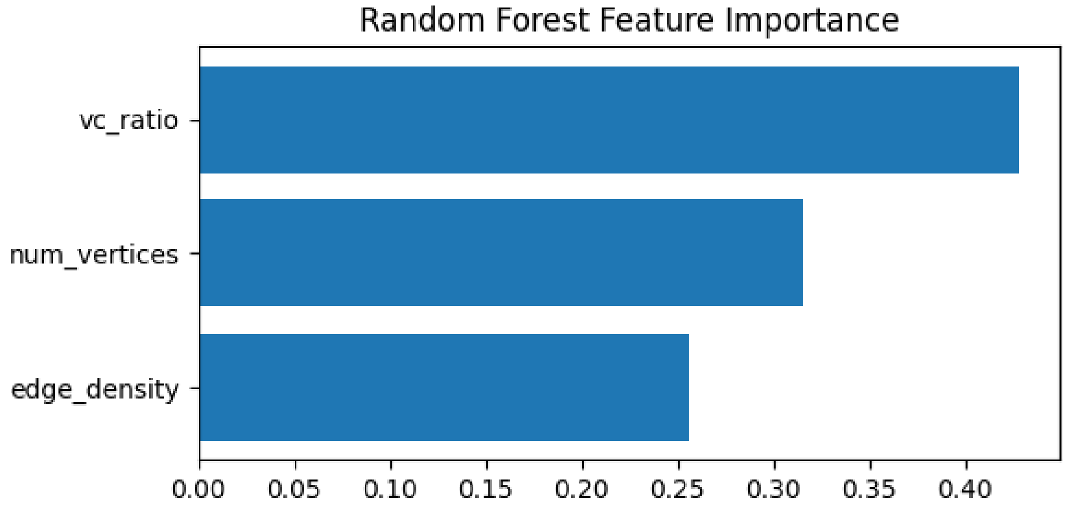


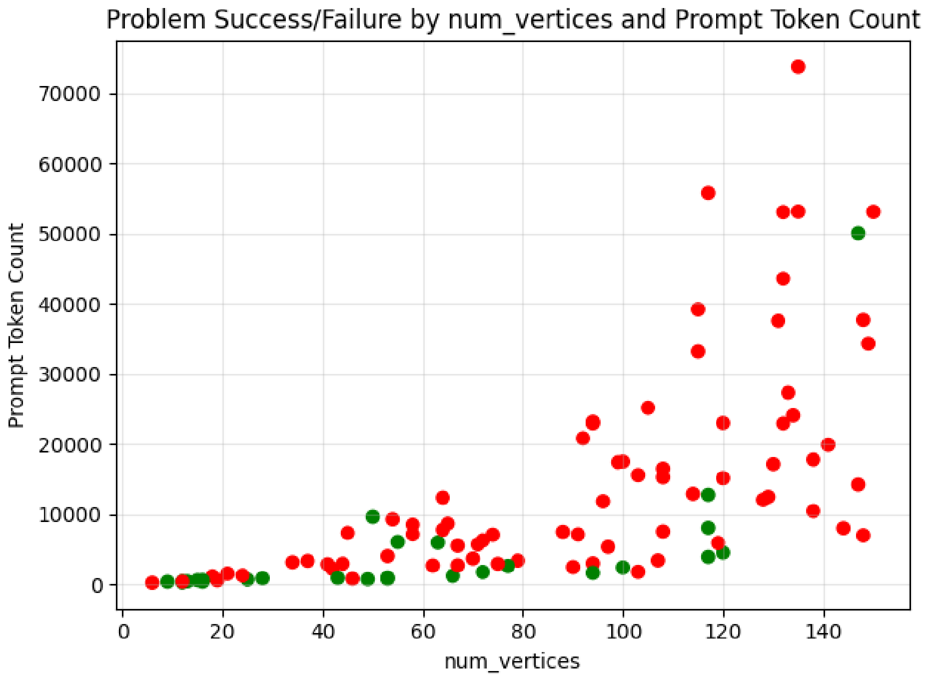


به طور مشابه در مسئله SSP هم پارامتر همبسته با طول پرامپت تاثیر بیشتری دارد.
عملکرد مدل ضعیف‌تر است و بیشتر مسائل حتی با پراپمت با توکن کمتر نسبت به 3-SAT نمی‌تواند حل کند.

:Base LLM (Gemini 2.5 Flash)
مسئله Vertex Cover:

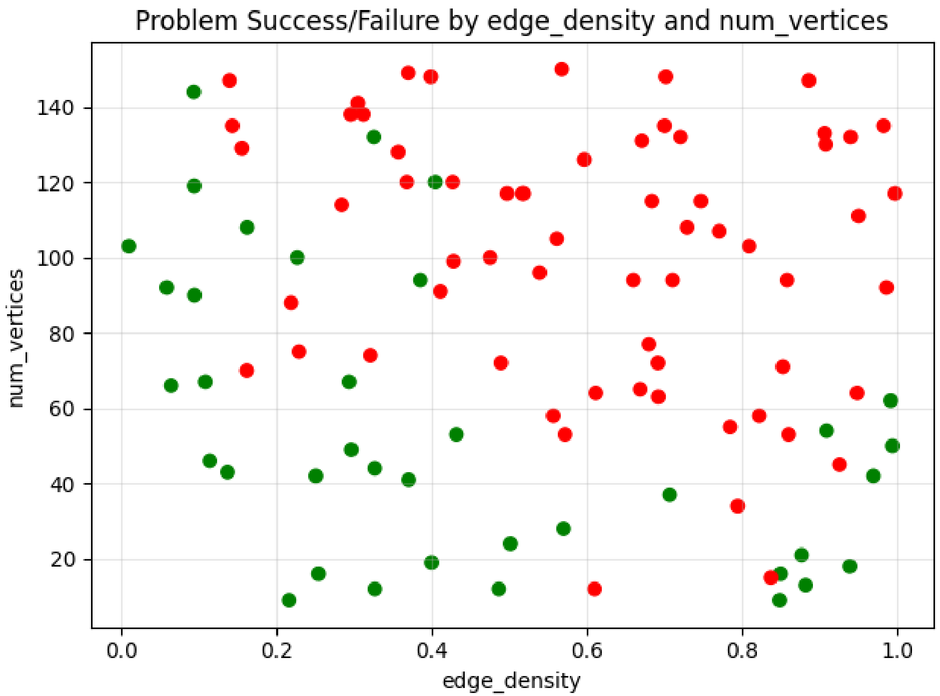


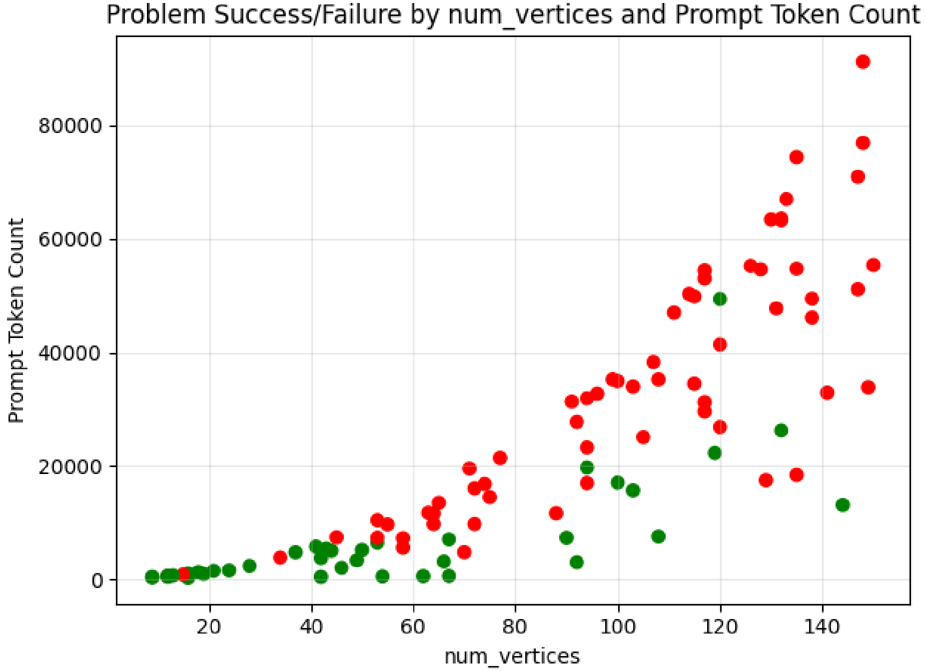


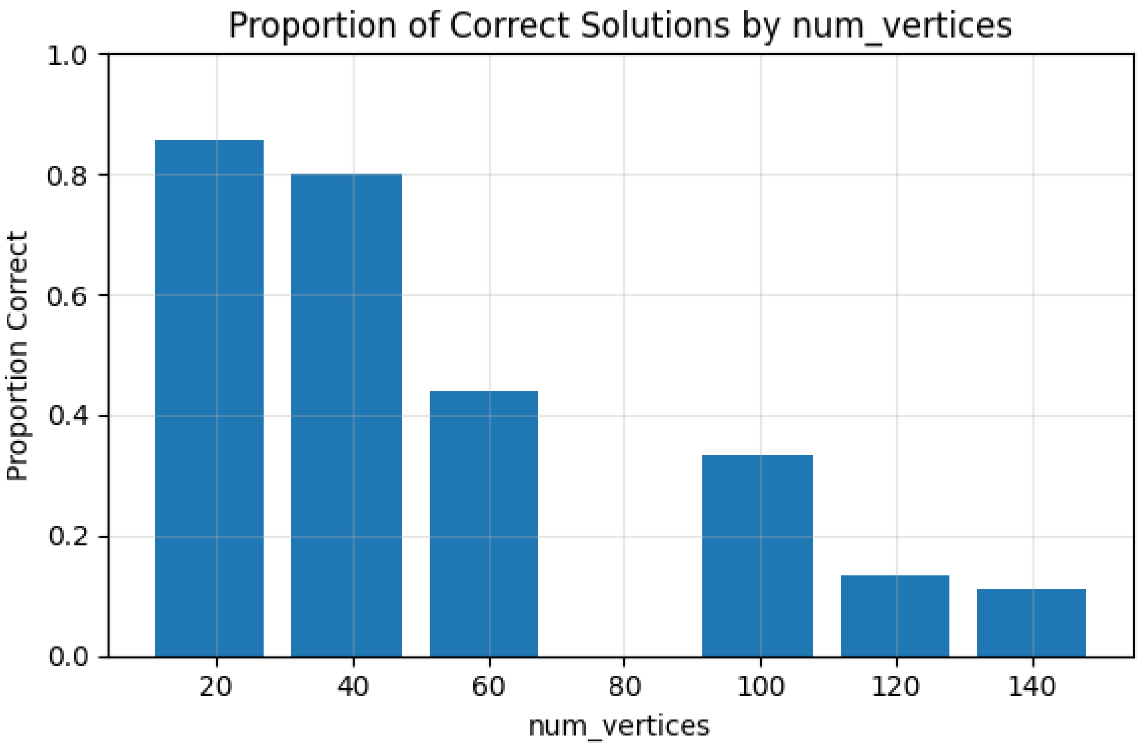


در Vertex Cover اندازه زیرگراف پاسخ بیشترین تاثیر را در حل دارد.

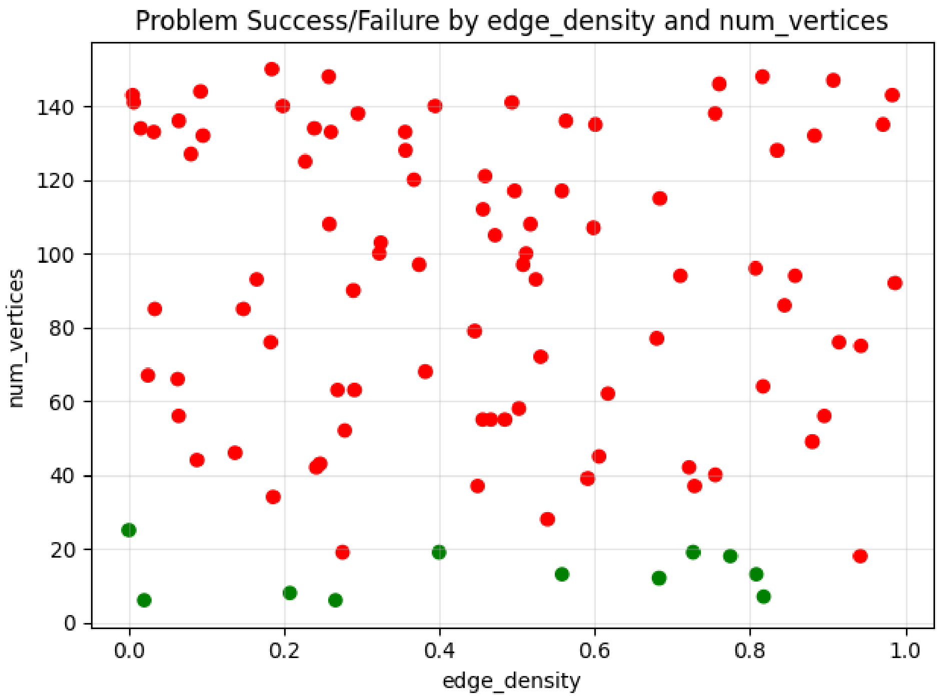
Base LLM (Gemini 2.5 Flash):
مسئله Maximum Clique:

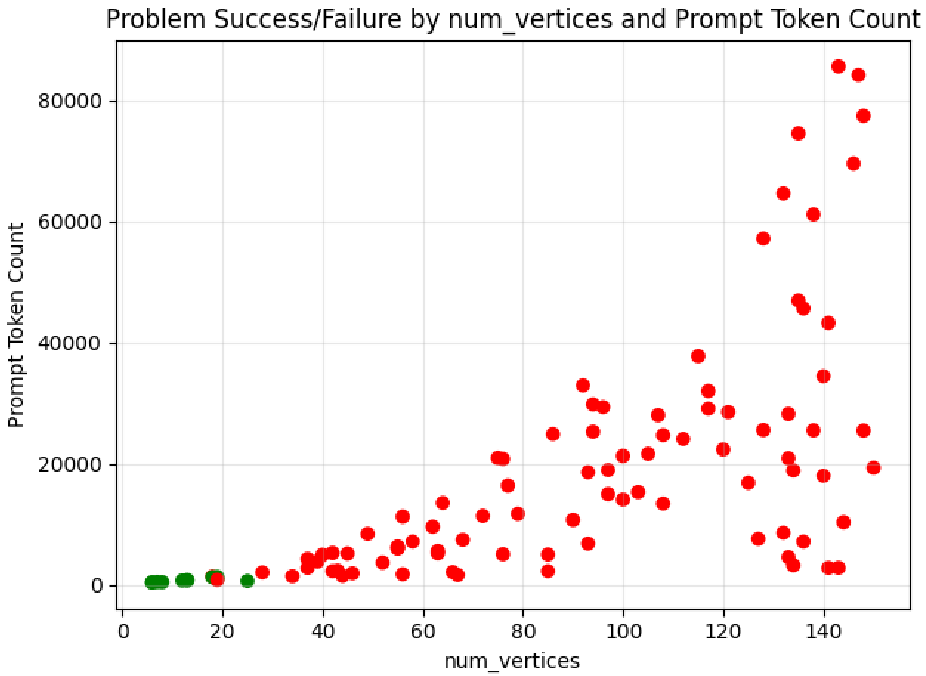




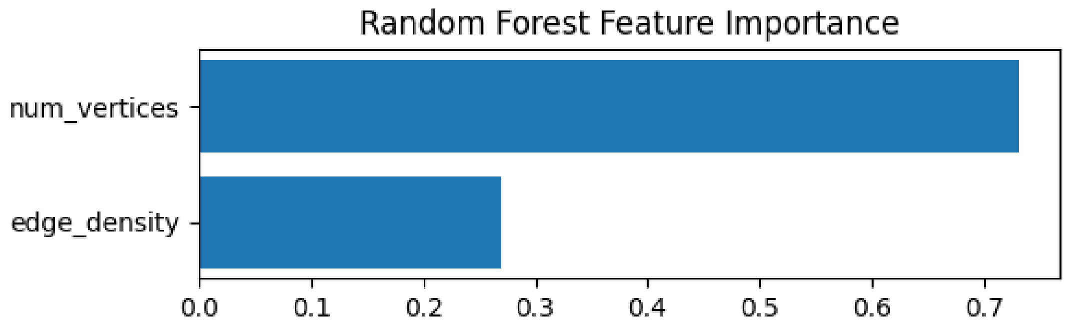


در Clique تعداد رئوس بیشترین تاثیر را در حل دارد.



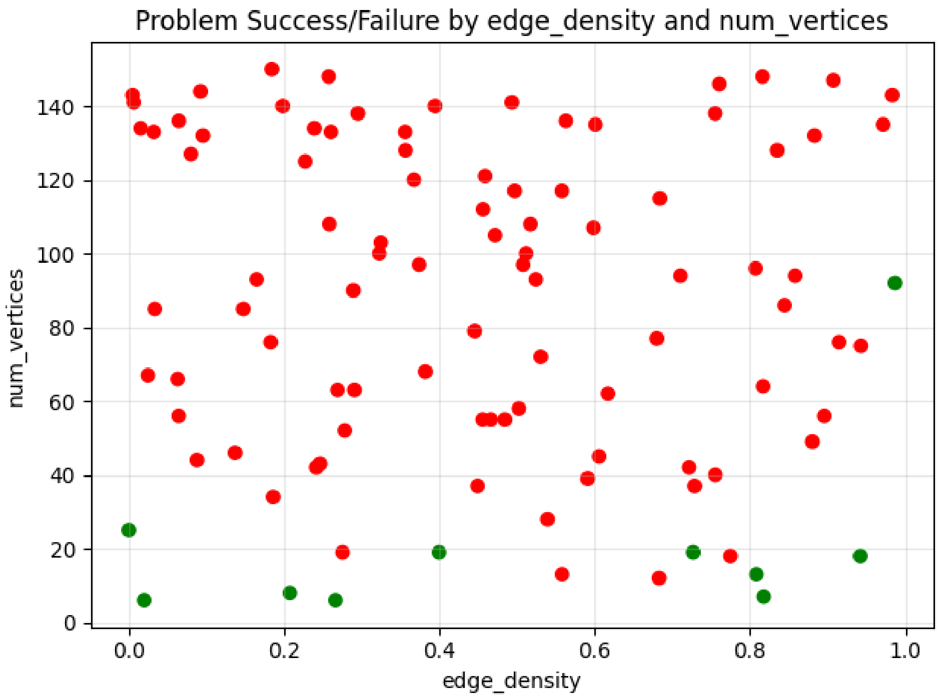


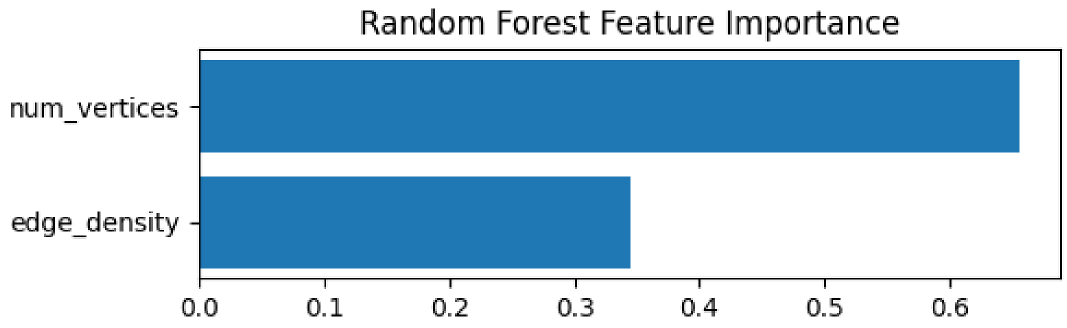
:Base LLM (Gemini 2.5 Flash)
مسئله Hamiltonian Path:

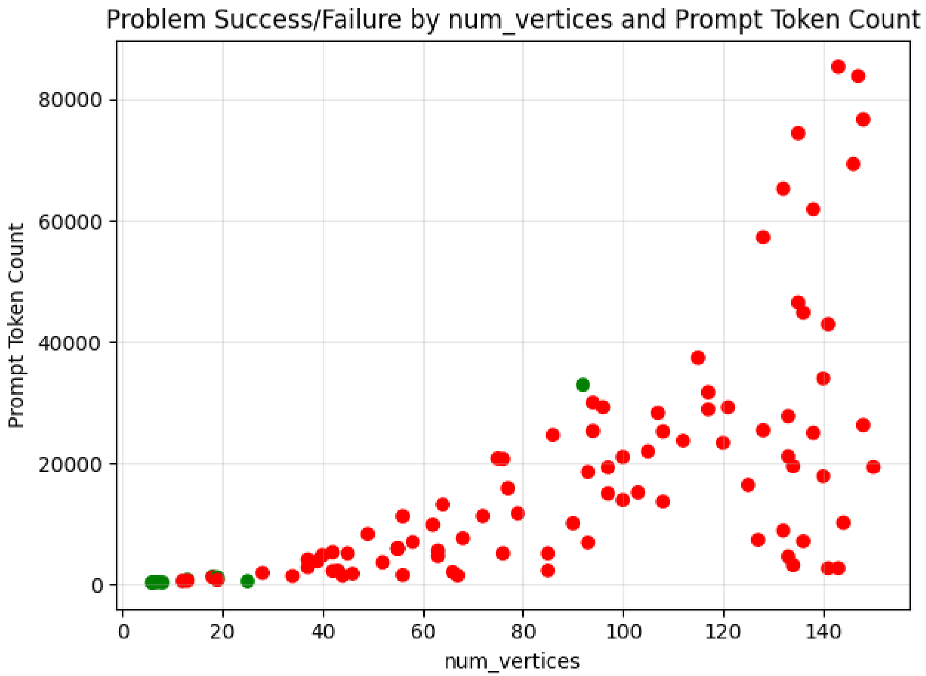


در Hamiltonian Path تعداد رئوس تاثیر بسیار زیادی در حل دارد.
پیدا کردن مسیر همیلتونی حتی برای گراف‌های کوچک هم برای مدل دشوار است.

:Base LLM (Gemini 2.5 Flash)
مسئله Hamiltonian Cycle:







نتایج مسئله دور همیلتونی بسیار مشابه مسئله مسیر همیلتونی است.

:Base LLM (Gemini 2.5 Flash)

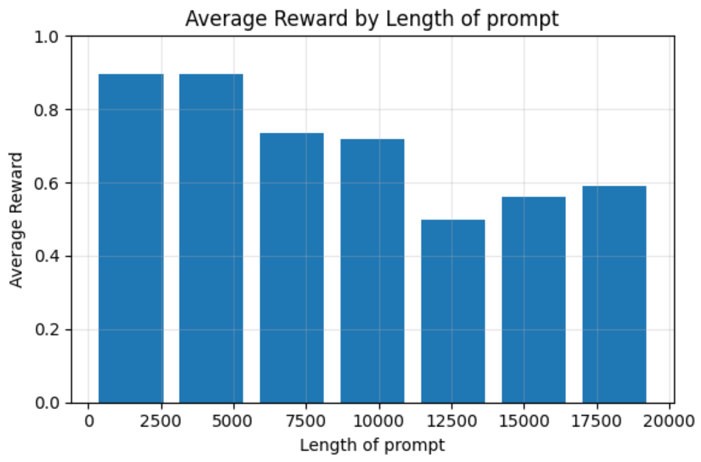
مقایسه سرعت:

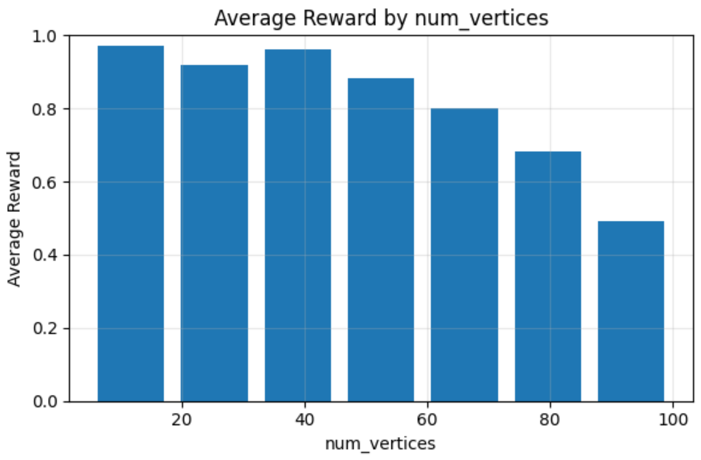
Problem	Abg. LLM	Avg Z3	Std. Z3
3-SAT	18.090	0.025	0.012
Subset Sum	14.012	0.055	0.039
Vertex Cover	22.781	5.930	54.387
Clique	28.519	0.394	1.419
Hamiltonian Path	22.306	-	-
Hamiltonian Cycle	22.785	-	-

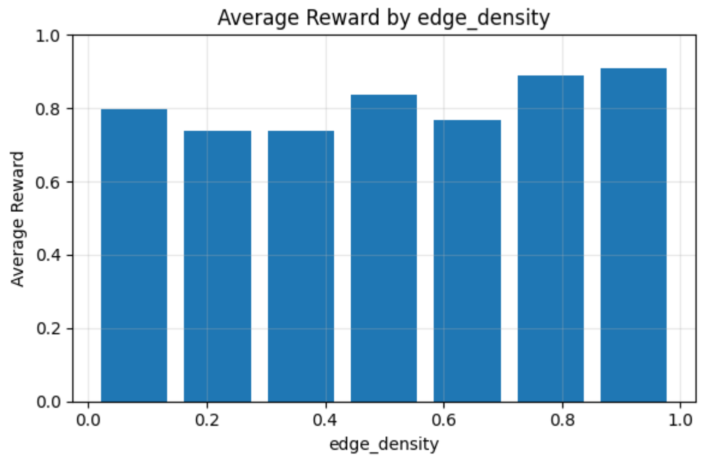
- Z3 نتوانست ۱۰ نمونه از مسائل مسیر یا دور همیلتونی را پس از چندین ساعت حل کند.
- در مسائل دیگر سرعت حل Z3 بسیار سریع‌تر است که حتی در صورت بهبود عملکرد مدل، از لحاظ سرعت نمی‌تواند از Z3 سریع‌تر باشد.
- به همین دلیل مسیر همیلتونی را به عنوان مسئله اصلی برای بهبود انتخاب می‌کنیم.

آموزش مدل:

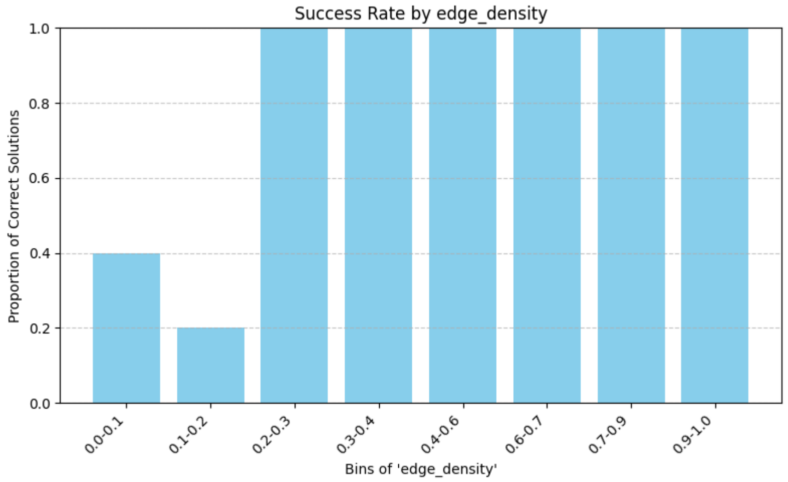
- ما از روش آموزشی ترکیب SFT و GRPO استفاده کردیم تا مدل Llama 3.2 Instruct را با تنظیمات مختلف برای مسئله مسیر همیلتونی آموزش دهیم.
- مدل را با 380 نمونه SFT سپس 570 نمونه مرتب شده GRPO با تابع ریوارد ساده و سپس 570 نمونه GRPO با تابع ریوارد سخت‌گیرانه آموزش دادیم.
- مدل‌های آموزش دیده بسیار عملکرد ضعیفی داشتند.
- با این حال ریوارد مدل برای مسائل بالا می‌شد که نشان می‌دهد مسیرهای یافت شده تعداد کمی راس اشتباه دارند.
- دلایل احتمالی ضعف پس از آموزش می‌تواند به خاطر کوچک بودن مدل و همچنین آموزش با داده کم باشد.

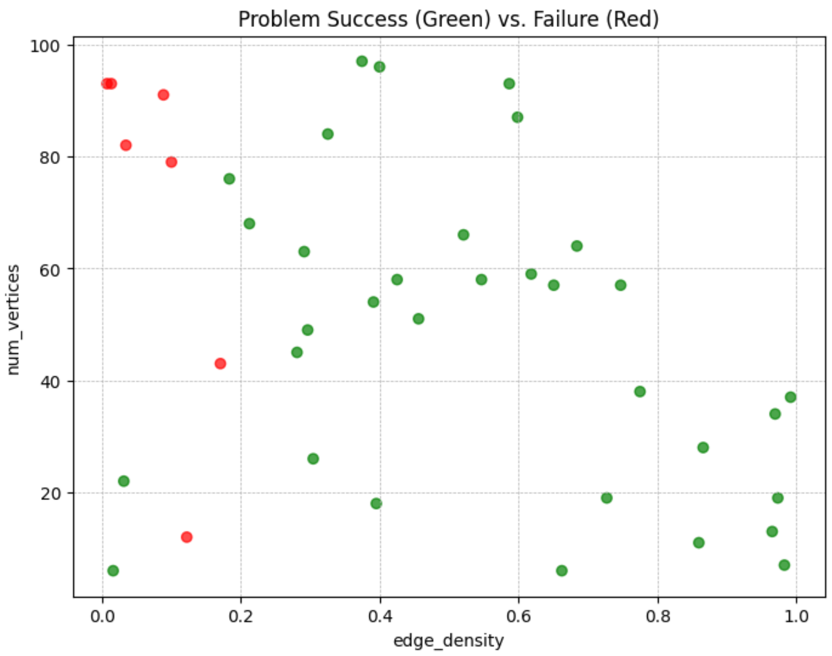


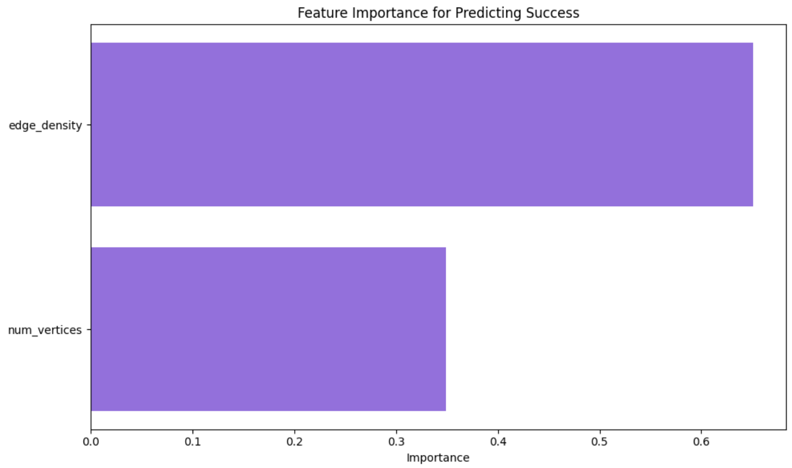


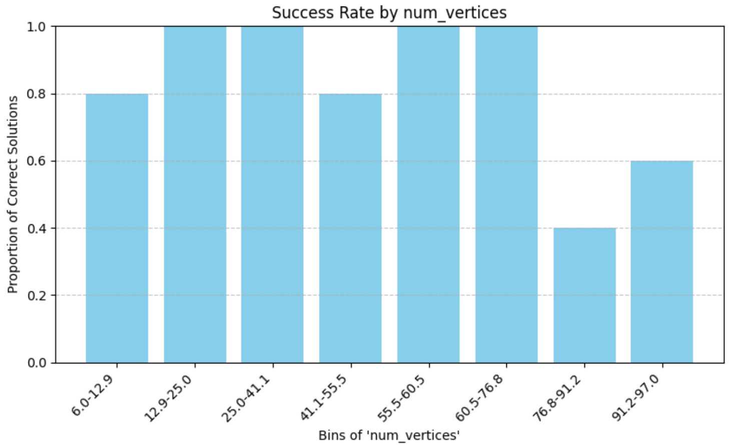


پیاده‌سازی ایجنت:









با دریافت ابزار بازسازی مسیر و مدل grpo آموزش دیده شده ایجنت توانست ۸۲.۵ درصد مسائل را به طور میانگین در زمان ۱۲۹.۵۷ ثانیه حل کند.

همچنین بر خلاف مدل پایه، در ایجنت تاثیر تعداد یال‌ها در حل بسیار بیشتر از تعداد رئوس است که این ممکن است نتیجه دهد که سختی یک مسئله ذاتی نیست بلکه وابسته به نحوه حل کردن آن است.

1. Chang Gong, Wanrui Bian, Z. Z. and Zheng, W. (2025). Pseudocode-injection magic: Enabling llms to tackle graph computational tasks.
2. Chengrun Yang, Xuezhi Wang, Y. L. H. L. Q. V. L. D. Z. and Chen, X. (2024). Large language models as optimizers.
3. DeepSeek-AI (2025). Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning.
4. Goldberg and Lingle (1985). Comparative study of different representations in genetic algorithms for job shop schedul ing problem.
5. Grefenstette, R. G. and Gucht, D. V. (1985). Comparative study of different representations in genetic algorithms for job shop scheduling problem.
6. Hanjun Dai, Elias B. Khalil, Y. Z. B. D. and Song, L. (2018). Learning combinatorial optimization algorithms over graphs.
7. Haoran Ye, Jiarui Wang, Z. C. F. B. C. H. H. K. J. P. and Song, G. (2024). Reevo: Large language models as hyper heuristics with reflective evolution.
8. Jason Wei, Xuezhi Wang, D. S. M. B. B. I. F. X. E. H. C. Q. V. L. and Zhou, D. (2023). Chain-of-thought prompting elicits reasoning in large language models.
9. LeonardodeMoura, N.B.(2008). Z3: anefficientsmtsolver.
10. Marcelo Prates, Pedro Avelar, H. L. L. C. L. and Vardi, M. Y. (2018). Learning to solve np-complete problems: A graph neural network for decision tsp.
11. Mostafa Elhoushi, Akshat Shrivastava, D. L. B. H. B. W. L. L. A. M. B. A. S. A. A. R. A. A. A. B. C. C. J.-W. (2024). Layerskip: Enabling early exit inference and self speculative decoding.
12. Zhihong Shao, Peiyi Wang, Q. Z. R. X. J. S. X. B. H. Z. M. Z. Y. L. Y. W. D. G. (2024). Deepseekmath: Pushing the limits of mathematical reasoning in open language mod els.
13. Shunyu Yao, Jeffrey Zhao, D. Y. N. D. I. S. K. N. Y. C. (2023). React: Synergizing reasoning and acting in lan guage models.

با تشکر از توجه شما