Additional Discussion to Reviewer 3xi4's Questions

Q3: Discussion of EcoAct

While EcoAct (Zhang et al., 2024) shares certain similarities with our approach (e.g., it is training-free and supports dynamic usage decisions), our work differs in key respects.

(1) Specifically, EcoAct does not propose an explicit algorithm for toolset optimization, nor does it provide a comprehensive "plug-and-play" framework with standardized interfaces to facilitate adding or removing tools.

(2) Furthermore, EcoAct primarily demonstrates efficiency gains on one benchmark (ToolBench), whereas our OctoTools framework is evaluated on 16 diverse tasks, reflecting broader coverage and generalization.

(3) Additionally, EcoAct is presented more as an algorithmic extension rather than a fully deployable system with open-source code and a live demo. In contrast, OctoTools offers both a public codebase and an accessible online demo, enabling the community to readily experiment with, extend, and evaluate our agentic framework.

Q4: Beneficial to have efficiency and cost analysis

We thank the reviewer for highlighting the importance of cost and efficiency in real-world deployments. Below is our concise cost analysis and its implications:

1. Small step limits still yield huge gains. Even restricting OctoTools to just one or two maximum steps (i.e., minimal multi-step planning) already yields sizable improvements—e.g., **+7.1% over GPT-4o with only a single step** (See Figure 5 in Section 5.1). This demonstrates that developers can cap the maximum number of steps and still gain substantial accuracy benefits while containing latencies and token usage. We summarize accuracies over the number maximum allowed steps as follows:

	Acc on 16 tasks
GPT-4o	around 49.2%
AutoGen	around 47.9%
GPT-Functions	around 51.0%
LangChain	around 51.2%
OctoTools (1 step)	**56.3%**
OctoTools (2 steps)	**56.4%**
OctoTools (4 steps)	**56.7%**
OctoTools (6 steps)	**58.3%**
OctoTools (8 steps)	**58.8%**
OctoTools (10 steps)	**59.1%**

2. Empirical cost analysis. As we discussed in Section 5.1, running OctoTools with up to 10 steps for **100 queries costs under $5 using GPT-4o**. With a smaller model (GPT-4-mini), the same 10-step setting costs about **$0.3 per 100 queries**. These findings suggest that even more complex, multi-step reasoning can remain practical budget-wise.

3. Adjustable toolset. When concerns about latency or token overhead arise (e.g., due to large or growing toolsets), users may limit the total number of steps or apply our optional toolset optimization to prune less beneficial tools. Alternatively, users can apply heuristics to manually configure the toolset—for example, enabling all image-related tools for visual reasoning tasks or disabling all web-access tools for mathematical reasoning tasks. This setup flexibly accommodates varying cost–accuracy trade-offs and deployment constraints.

Overall, our results indicate that OctoTools can balance the accuracy gains of multi-step reasoning with manageable computational and monetary overhead. By tuning parameters such as the maximum step limit or applying toolset optimization, users can effectively control cost while still achieving significant performance improvements over baseline systems.

Q7: Relationship between OctoTools' tool cards and Model Context Protocol

Thank you for bringing up the Model Context Protocol (MCP) and we appreciate the interest in comparing OctoTools' tool cards with Anthropic's Model Context Protocol (MCP). In essence, both approaches standardize how LLM-based systems discover and invoke external functionality, but they serve somewhat different scopes:

1. **Single-System vs. Cross-System Focus.**
   OctoTools' tool cards primarily operate *within a single agentic framework*, encapsulating each tool's name, metadata, input/output specifications, and best practices. This design enables our planner–executor pipeline to dynamically decide which tool to call (and how) for multi-step reasoning. In contrast, MCP focuses on an *open protocol for inter-system connections*, establishing uniform, bidirectional communication across multiple services or data sources.

2. **Granular Metadata vs. Protocol Standardization.**
   Tool cards place emphasis on *tool-specific best practices, usage constraints, and domain details*, so that an LLM can effectively plan and chain these tools together without retraining. MCP, on the other hand, defines how *applications and servers* should exchange context and capabilities through a standardized interface. Both offer structured ways of exposing tools, but MCP is designed to unify orchestration across *disparate systems*, whereas tool cards are designed for *intra-system* integration, extensibility, and multi-step reasoning.

3. **Complementary Potential.**
   While OctoTools and MCP were developed with different priorities in mind, they are not mutually exclusive. MCP's open standard for interoperability could complement our framework if we extend OctoTools to discover and coordinate tools across distributed networks. Meanwhile, the *tool card* abstraction would remain responsible for specifying each tool's local usage constraints and metadata within the planner–executor loop.

Overall, OctoTools' tool cards do share MCP's structured approach to describing tool capabilities, but they differ in scope and implementation details. We consider them compatible and potentially synergistic if OctoTools is extended to leverage MCP's cross-system connectivity in the future.

References

Zhang, S., Zhang, J., Ding, D., Garcia, M. H., Mallick, A., Madrigal, D., Xia, M., Rühle, V., Wu, Q., and Wang, C. Ecoact: Economic agent determines when to register what action. *arXiv preprint arXiv:2411.01643*, 2024.