

More with Less: An Empirical Study of Turn-Control Strategies for Efficient Coding Agents

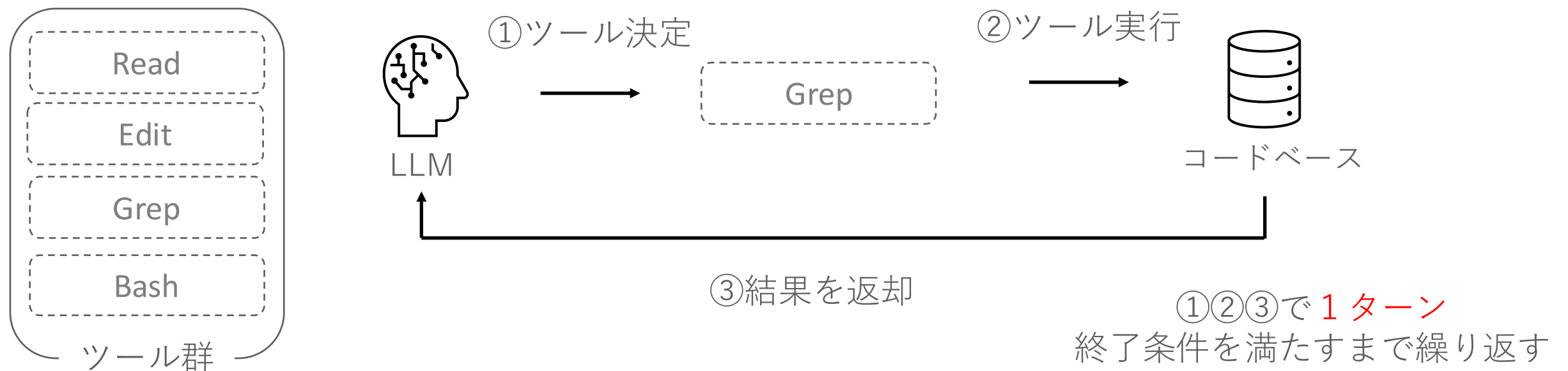
Pengfei Gao
ByteDance
Beijing, China
gaopengfei.se@bytedance.com

Chao Peng*
ByteDance
Beijing, China
pengchao.x@bytedance.com

ICSE '26, April 12–18, 2026, Rio de Janeiro, Brazil

背景

コーディングエージェント



- バグ修正・リファクタリングなどのタスクで利用される
- 自律的にタスクを遂行
- Cursor^[4], Claude Code^[1]などの製品が存在

[4] Cursor. 2025. <https://cursor.com>

[1] Anthropic. 2025. Claude Code. <https://www.anthropic.com/claude-code>

エージェントの課題点

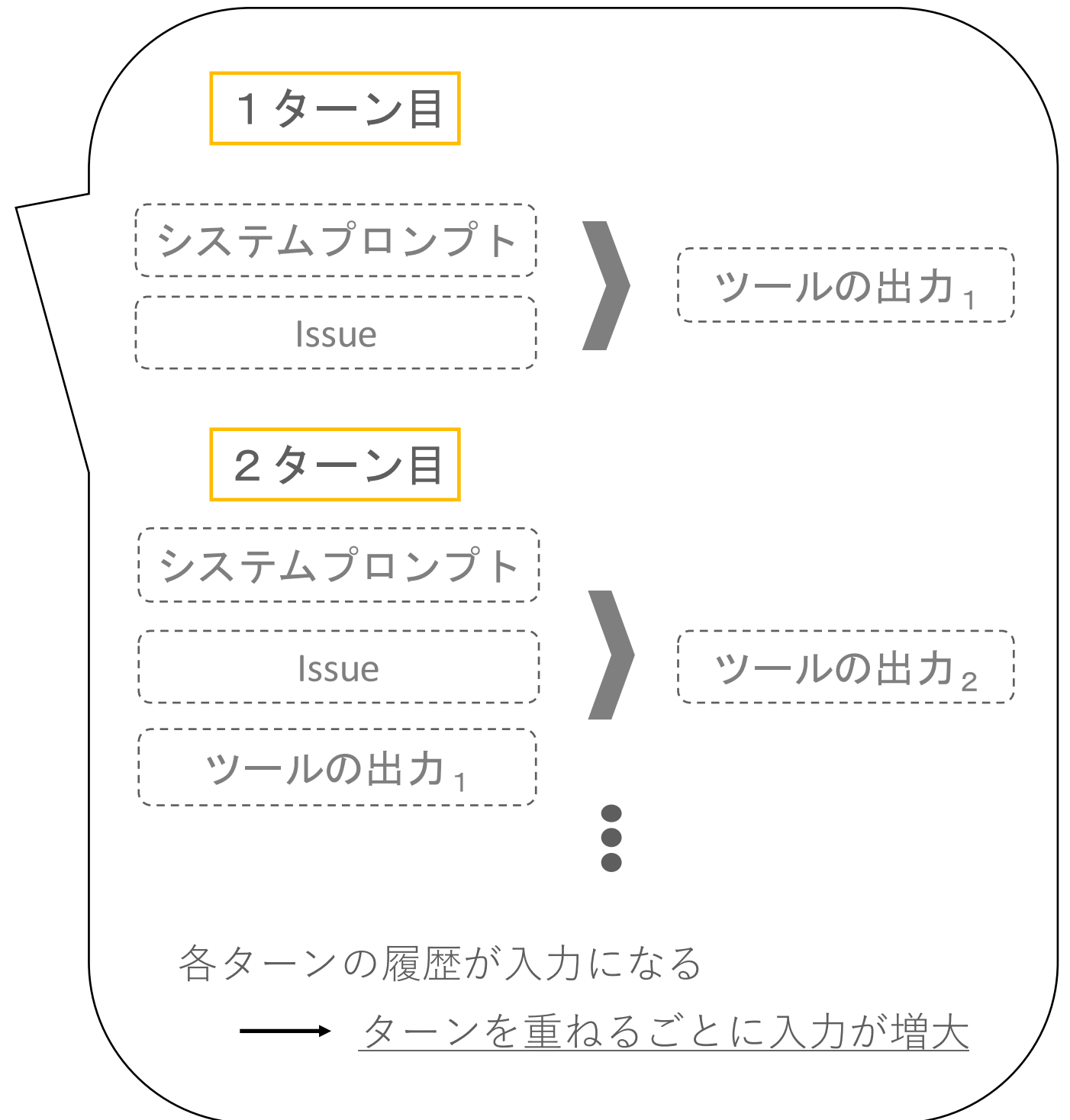
運用コストが高い

要因 1：入力トークンの増大

要因 2：高性能モデルの価格

要因 3：多くのターンが必要

要因 4：無駄なターンを使う



エージェントの課題点

運用コストが高い

要因 1：入力トークンの増大

要因 2：高性能モデルの価格

要因 3：多くのターンが必要

要因 4：無駄なターンを使う

(例)

claude 4 Sonnet

入力：\$3 出力：\$15

claude4 Opus

入力：\$15 出力：\$75

エージェントの課題点

運用コストが高い

要因 1：入力トークンの増大

要因 2：高性能モデルの価格

要因 3：多くのターンが必要

要因 4：無駄なターンを使う

SWE-bench リーダーボード^[22]
上位 5 エージェントのターン数

中央値 41 ~ 58 ターン

175 ターンを超える場合も

[22] SWE-bench. 2025. SWE-bench Verified Leaderboard. <https://www.swebench.com/>

エージェントの課題点

運用コストが高い

要因 1：入力トークンの増大

要因 2：高性能モデルの価格

要因 3：多くのターンが必要

要因 4：無駄なターンを使う

cursorのフォーラム^[5]上で

- ・ 重複した関数, 冗長なコードを追加した
- ・ 要求していない機能追加を試みた

という報告

[5] Cursor Community Forum. 2024. Claude 3.7 Booby Traps.
<https://forum.cursor.com/t/claude-3-7-booby-traps/63114>

エージェントの課題点

運用コストが高い

要因 1：入力トークンの増大

要因 2：高性能モデルの価格

要因 3：多くのターンが必要

要因 4：無駄なターンを使う



ターン数の制御が有効ではないか？

目的・RQs

目的

様々なターン制御戦略がエージェントの性能・コストに与える影響を調べる

RQ

- ターン数の制限がない時の性能とコストは？
- リマインダ付きの固定ターン制限時の性能とコストは？
- 動的なターン増加戦略の性能とコストは？

実験設定

タスク

- SWE-bench Verified 100件サブセット

モデル

モデル名	入力 (\$/Mtok)	出力 (\$/Mtok)
Claude 4 Sonnet	3	15
Gemini 2.5 Pro	1.25/2.50(20万tok超)	10/15
GPT4.1	2	8

エージェント

- TRAE Agent^[8]

ターン制御戦略

無制限（RQ1）

- エージェントの「task_done」のみ

固定＋リマインダ（RQ2）

- 毎ターン「残り〇ターン」を入力する。上限到達で強制終了

動的（RQ3）

- 低い上限で開始してパッチ未生成であれば、一度だけ上限引き上げ

上限・延長幅は無制限時のターン数分布から決定（RQ2,3共通）

例）上限：25パーセンタイル 延長幅：50パーセンタイルまで

結果 | RQ1 (ターン無制限)

LLM	解決率	総ターン数	コスト
Claude 4 Sonnet	75%	5,704	\$585
Gemini 2.5 Pro	63%	3,925	\$407
GPT 4.1	62%	5,641	\$322

- 解決率

claudeがトップだがコストも一番高い

- ターン数

geminiが最も少ないが最安ではない

- コスト

GPTが最安

結果 | RQ1（ターン無制限）

LLM	解決率	総ターン数	コスト
Claude 4 Sonnet	75%	5,704	\$585
Gemini 2.5 Pro	63%	3,925	\$407
GPT 4.1	62%	5,641	\$322

ターン数の分布（パーセンタイル） RQ2,3で利用

LLM	ターン数（25 th ）	ターン数（50 th ）	ターン数（75 th ）
Claude 4 Sonnet	44	52	63
Gemini 2.5 Pro	19	28	44
GPT 4.1	38	49	66

結果 | RQ2（固定+リマインダ）

25パーセンタイル制限

モデル	解決率	コスト
claude	60% (-20%)	\$312 (-47%)
gemini	26% (-59%)	\$51 (-87%)
GPT	48% (-21%)	\$120 (-63%)

50パーセンタイル制限

モデル	解決率	コスト
claude	63% (-16%)	\$384 (-34%)
gemini	52% (-17%)	\$82 (-79%)
GPT	57% (-8%)	\$163 (-49%)

75パーセンタイル制限

モデル	解決率	コスト
claude	71% (-5%)	\$446 (-23%)
gemini	65% (+3%)	\$129 (-68%)
GPT	57% (-8%)	\$198 (-38%)

結果 | RQ2 (固定+リマインダ)

25パーセンタイル制限

モデル	解決率	コスト
claude	60% (-20%)	\$312 (-47%)
gemini	26% (-59%)	\$51 (-87%)

- ・ ターン数を制限することで性能とコスト両方低下
- ・ 75th制限で解決率一桁低下 & コスト削減でバランス良
- ・ リマインダによって解決率を改善

75パーセンタイル制限

モデル	解決率	コスト
claude	71% (-5%)	\$446 (-23%)
gemini	65% (+3%)	\$129 (-68%)
GPT	57% (-8%)	\$198 (-38%)

結果 | RQ3 (動的)

上限固定 (50パーセンタイル)

モデル	解決率	コスト
claude	63%	\$384
gemini	52%	\$82
GPT	57%	\$163

動的 (25th → 50th)

モデル	解決率	コスト
claude	64% (+1.6%)	\$324 (-16%)
gemini	55% (+5.8%)	\$78 (-6%)
GPT	57% (0%)	\$123 (-24%)

上限固定 (75パーセンタイル)

モデル	解決率	コスト
claude	71%	\$446
gemini	65%	\$129
GPT	57%	\$198

動的 (50th → 75th)

モデル	解決率	コスト
claude	70% (-1.4%)	\$387 (-13%)
gemini	64% (-1.5%)	\$98 (-24%)
GPT	59% (+3.5%)	\$174 (-12%)

性能同等 + コスト低下を実現

まとめ

コーディングエージェントのターン数を制御することによる性能とコストの変化を調査

単純にターン数削減 → 解決率低下 + コスト削減

リマインダー追加 → 解決率改善

動的 → 解決率同等 + コスト削減

適切なターン管理で費用対効果を最適化可能