A PairCoder: A Multi-Agent Framework for Code Generation

Taylan Bapur, Jan Härtrich

Summer 2025 – Seminar on Machine Learning in Software Engineering

Agenda

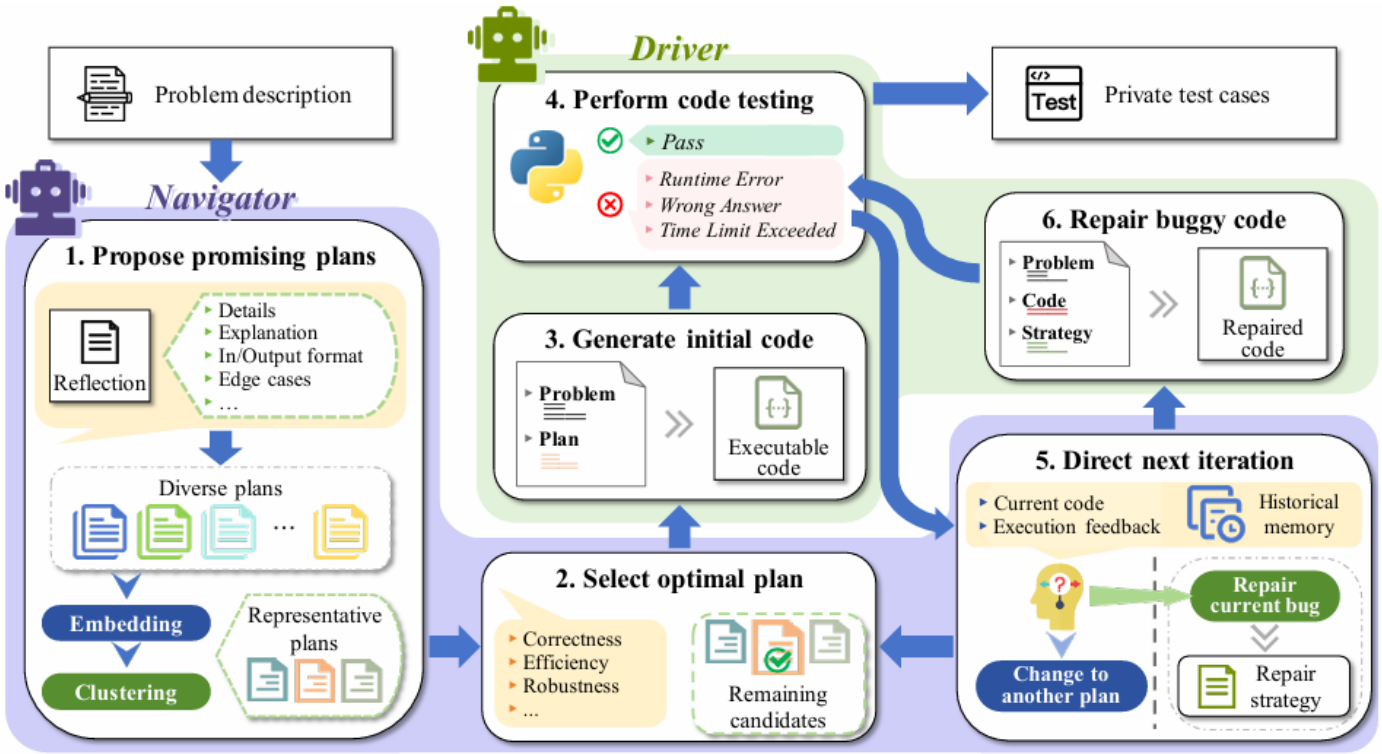
Motivation

- LLMs are effective at code completion and simple programming tasks.
- Struggle with complex logic, dependencies, and iterative problem-solving.
- Multi-agent frameworks offer collaboration, adaptability, and robustness.

Overview of Frameworks

- **PairCoder**: Navigator and Driver collaborate iteratively.
- **Guided Code Generation**: Decomposes tasks hierarchically.
- **MapCoder**: Simulates full development cycle with 4 agents.

PairCoder Architecture



- Navigator plans, Driver implements and tests.
- Iterative loop mimics human pair programming.

Navigator Role

- Reflects on problem and generates multiple solution plans.
- Uses clustering to pick diverse, promising strategies.
- Analyzes feedback to refine or switch plans.

Driver Role

- Implements code from selected plan.
- Runs public test cases.
- Applies Navigator's suggested fixes.

Iterative Workflow

1. Navigator proposes a plan.
2. Driver generates code.
3. Code is tested and classified.
4. Navigator reacts to feedback.

Scenario: Greedy Array Fix with Insertions

- Goal: Ensure each $arr[i] \leq i + 1$.
- Method: Insert values to the array as needed.
- Input: `[1, 3, 4]`
- Expected: Insert to satisfy index constraints with minimal operations.

Navigator's Plan (Logical Flaw)

```
1   def solve(arr):
2       operations = 0
3       for i, val in enumerate(arr):
4           if val > i + 1:
5               operations += val - (i + 1)
6       return operations
7
```

Flaw: Counts how many insertions would be needed, but *does not simulate actual insertions*.

Fixed instertion

```
1   def solve_fixed(arr):
2       for i, val in enumerate(arr):
3           if val > i + 1:
4             arr.insert(i, i + 1)
5           else:
6               i += 1
7       return arr
8
```

Fix: fixing by acutally inserting value 2

Execution: Sample Test Case

```
1    input_sequence = [1, 3, 4]
2    result = solve_fixed(input_sequence)
3    print("Output:", result)
4
```

Output 1: 2
Output 2: [1, 2, 3, 4]

Demo

- Live Demo

Key Techniques

- Multi-plan exploration using clustering.
- Feedback-driven refinement loop.
- State tracking to avoid repetition.

Evaluation Results

Approaches	GPT-3.5-Turbo						DeepSeek-Coder					
	HumanEval		MBPP		CodeContest		HumanEval		MBPP		CodeContest	
	Orig	Plus	Orig	Plus	Valid	Test	Orig	Plus	Orig	Plus	Valid	Test
Direct prompting	67.68	60.98	66.80	66.42	6.84	6.06	76.22	67.78	66.40	64.41	5.98	6.67
CoT prompting	68.90	62.80	69.00	67.17	5.13	5.45	77.27	68.90	67.60	67.67	5.45	6.67
SCoT prompting	68.29	61.59	62.60	61.40	5.98	4.24	73.17	65.85	61.80	60.15	4.27	6.06
Self-planning	72.56	64.63	69.60	67.67	7.69	6.06	74.39	68.90	65.80	68.17	6.84	9.09
Self-collaboration†	74.40	-	68.20	-	-	-	-	-	-	-	-	-
Self-repair	73.17	65.24	70.60	69.42	7.69	6.67	77.44	70.12	70.00	70.43	5.98	7.27
Self-debugging	78.05	72.56	72.80	70.43	9.40	11.52	79.27	73.78	72.20	71.12	8.55	13.33
INTERVENOR	77.44	69.51	73.40	71.93	8.55	9.09	79.88	72.56	72.60	72.43	7.69	10.30
Reflexion	69.57	-	67.76	-	-	-	81.99	-	74.31	-	-	-
LDB†	82.90	-	76.00	-	-	-	-	-	-	-	-	-
PairCoder (ours)	**87.80**	**77.44**	**80.60**	**77.69**	**17.95**	**15.15**	**85.37**	**76.22**	**78.80**	**75.69**	**13.68**	**14.55**
Relative improvement ↑	29.73%	26.99%	20.66%	16.97%	162.43%	150.00%	12.00%	12.45%	18.67%	17.51%	128.76%	118.14%

- Up to 87.8% on HumanEval.
- Outperforms CoT, Self-debugging.
- Gains most from refinement loop.

What did we get?

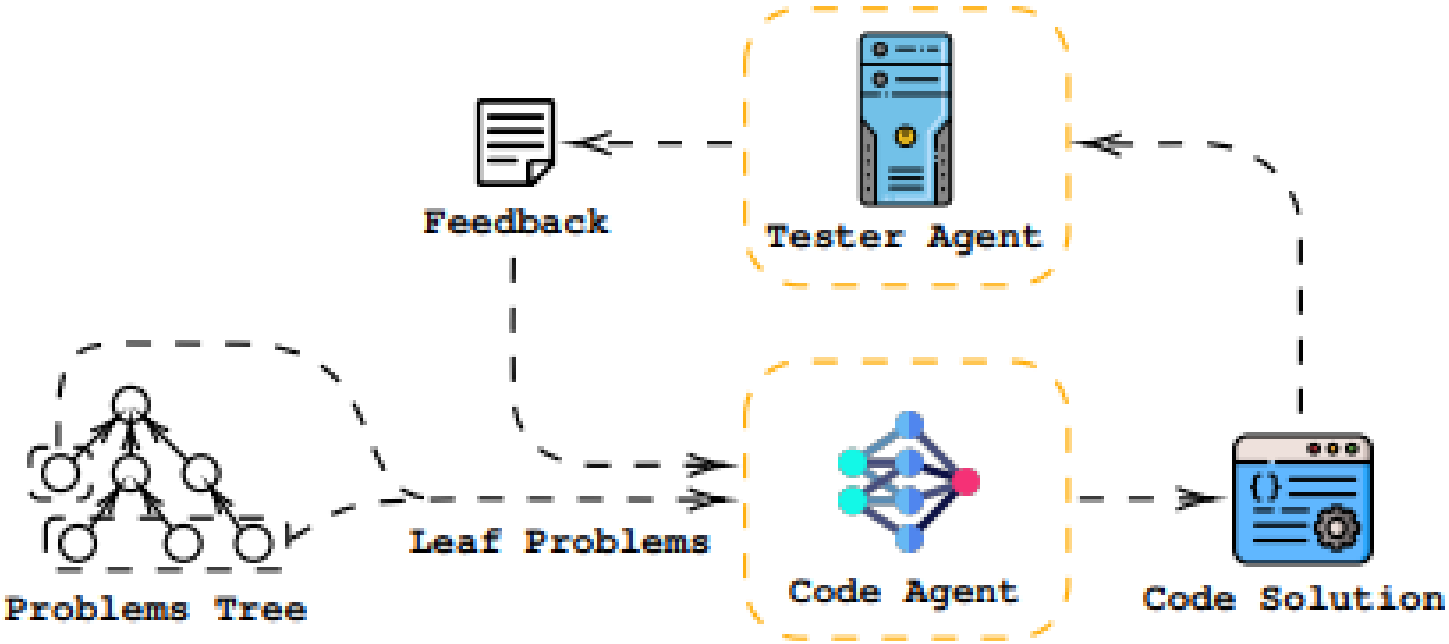
Benchmark	Split	Score (%)	Overall (%)
mbpp	test	76.74	79.37
mbpp	plus	85.00	
humaneval	raw	90.00	67.50
humaneval	plus	45.00	
codecontest	test	0.00	27.78
codecontest	valid	38.46	

- mbpp 63 tests
- humaneval 40 tests
- codecontest 18 tests
- https://github.com/ELSATOAH/A-Pair-Coder

Guided Code Gen Overview

- Generalist Agent decomposes problem.

Guided Code Gen Overview



- Code Agents solve atomic tasks.
- Tester Agent validates components.

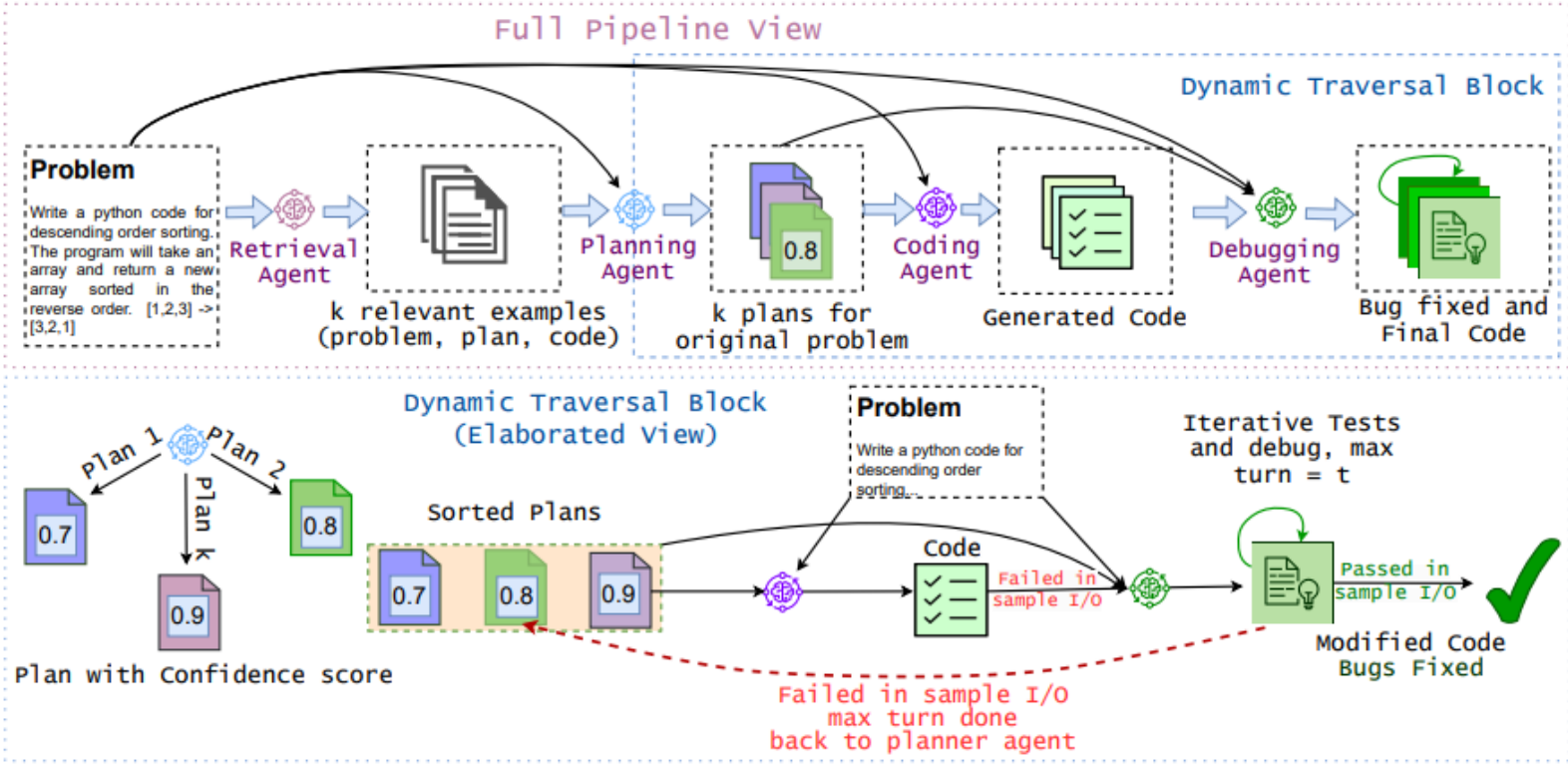
Comparison with PairCoder

- Guided Code is modular and hierarchical.
- PairCoder is iterative and flexible.
- Trade-off: structure vs. adaptability.

MapCoder Overview

- Simulates full development cycle.
- 4 agents: Retrieval, Planning, Coding, Debugging.
- Inspired by human-like software pipelines.

MapCoder Architecture



- Pipeline with dynamic traversal.
- Switches plans on failure.

Agent Roles in MapCoder

- Retrieval Agent: Fabricates examples.
- Planning Agent: Generates multiple plans + scores.
- Coding Agent: Implements plan.
- Debugging Agent: Refines code iteratively.

MapCoder Evaluation

- 93.9% pass@1 on HumanEval.
- SOTA results across benchmarks.
- Best suited for hard problems.

Threats to Validity

- **Dataset Leakage:** Possible training overlap with benchmarks like HumanEval; may inflate accuracy.
- **Prompt Sensitivity:** Results depend heavily on prompt design, sampling temperature, and clustering.
- **Model Size Differences:** Comparisons use different model scales (e.g., LLaMA 8B vs. GPT-3.5), affecting fairness.
- **Evaluation Bias:** partial correctness or code quality; setups vary between frameworks.
- **Reproducibility:** Practical replication (e.g., via GitHub) may fail due to setup issues or lack of robustness.

Framework Comparison

- **PairCoder:** Agile, mid-sized tasks.
- **Guided Code:** Best for structured problems.
- **MapCoder:** Full pipeline, highest accuracy.

Conclusion

- Multi-agent systems improve LLM coding.
- Each method has strengths.
- Hybrid approaches are promising.
- Tool reproducibility is a challenge.

Questions?