PairCoder: Agent Collaboration for Smarter Code Generation

Taylan Bapur

Machine Learning in Software Engineering Seminar

2025

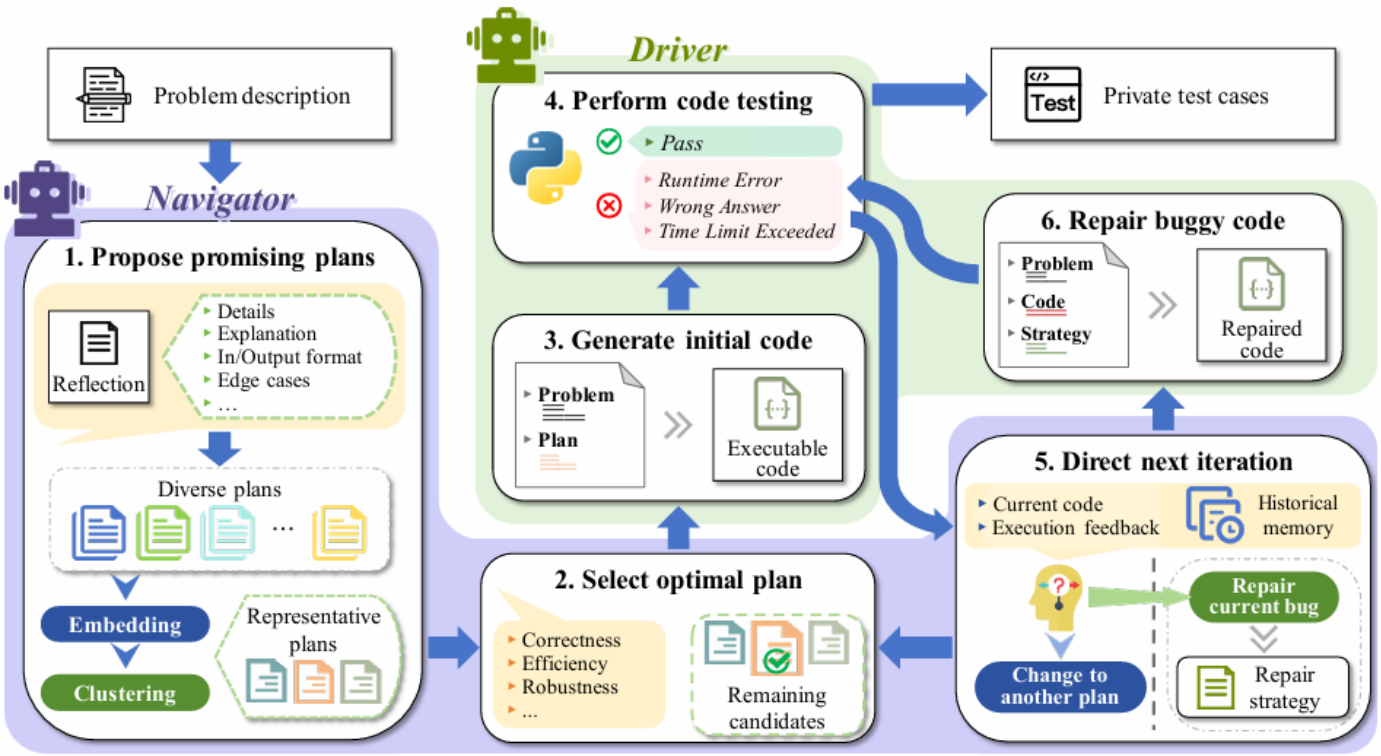
Motivation

- LLMs can generate code but struggle with complex logic
- Rigid, single-path plans limit robustness
- Human pair programming inspires agent collaboration

What is PairCoder?

- Two LLM agents: **Navigator** and **Driver**
- Navigator reflects, plans, selects strategies
- Driver implements, tests, and refines code
- Agents work in an iterative feedback loop

PairCoder Architecture



Source: Zhang et al. (2024)

Key Techniques

- **Multi-Plan Exploration**: Generate, cluster, and select diverse plans
- **Feedback-Driven Refinement**: Use test results to refine or switch strategies
- Historical memory to avoid redundant attempts

Example: Initial Code by Driver

```
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
```

Listing 1: Driver's original code output

Refined Code after Feedback

```python
def solve_fixed(arr):
    arr = arr[:]
    i = 0
    ops = 0
    while i < len(arr):
        if arr[i] > i + 1:
            arr.insert(i, i + 1)
            ops += 1
        else:
            i += 1
    return ops
```

Listing 2: Code after Navigator refinement

Results and Benchmarks

- Datasets: HumanEval, MBPP, CodeContest
- Up to 162% improvement in pass@1 over baselines
- PairCoder excels at complex tasks requiring logical precision

Accuracy Gains (pass@1)

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%

Source: Zhang et al. (2024)

Ablation and Efficiency

- Removing Multi-Plan drops accuracy by 6–8%
- Removing Feedback Loop drops it by 10–12%
- Balance of performance vs. token/API cost

PairCoder vs. MapCoder

- **PairCoder**: 2 tightly-coupled agents for focused refinement
- **MapCoder**: full pipeline (retrieval, planning, generation, debugging)
- PairCoder = Agile; MapCoder = Comprehensive

Limitations and Future Work

- Performance depends on initial plan quality
- Computationally expensive in real-time contexts
- Extend with human-in-the-loop or automated test generation

Conclusion

- PairCoder leverages agent collaboration for robust code generation
- Outperforms state-of-the-art baselines
- Paves the way for agentic software engineering tools

Questions?

Thank you!
Questions and discussion welcome.