1. **Learning Rate**: You use lr=0.0003. How sensitive is training to this? Did you do hyperparameter sweep?
2. **PPO Hyperparameters**: n\_steps=64, batch\_size=32. Why these values? Any ablation studies?
3. **Entropy Coefficient**: ent\_coef=0.01. Is exploration encouraged sufficiently? Should you increase this?
4. **Multi-Task Learning**: Can one policy handle multiple domains? Or do you need domain-specific policies?
5. **Training Stability**: Do rewards converge? What is the learning curve? Smooth or noisy?
6. **Benchmark Selection**: Which domains should you test on? IPC benchmarks? New domains?
7. **Problem Difficulty**: How do you select easy vs. hard problems? Is difficulty measured by baseline FD time?
8. **Train-Test Split**: What is the split ratio? Are test problems from different domains?
9. **Metric Selection**: Should you track plan cost, search time, or both? Are they orthogonal?
10. **Expansion Count**: Is the number of A\* expansions a good proxy for heuristic quality? Why or why not?
11. **Plan Quality**: Do learned strategies produce better plans (lower cost) or faster planning?
12. **Baseline Fairness**: Are baselines (SCC-DFP, MIASM) run with the same time limits? Same shrinking configs?
13. **Statistical Significance**: How many runs per problem? Confidence intervals? T-tests?
14. **Timeout Handling**: Problems that timeout—how do you score them? Partial credit or 0?
15. **Scaling**: How do metrics change as problem size increases? Polynomial, exponential, logarithmic?
16. **Ablation Studies**: What components are most critical? GNN vs. random features? Reward variant?
17. **Domain Generalization**: Test on unseen domains. How much does performance drop?
18. **Computational Cost**: Total training time? Inference time per merge? Is ROI positive?
19. **Coverage**: What fraction of problems is the policy better than baselines on?
20. **Failure Analysis**: On which problems does the learned policy fail? Root causes?
21. **Visualization**: Can you visualize the TS graph and learned policy decisions? For papers/debugging?
22. **Version Control**: FD version? Python version? NumPy/PyTorch versions? Do these affect results?
23. **Documentation**: Is code well-commented? Can newcomers understand the architecture?
24. **Error Recovery**: What happens if env.step() raises an exception? Retry or terminate?
25. **Related Work**: How does this compare to other learned heuristics? Other merge strategies?
26. **Theoretical Analysis**: Is there any theory about optimality of learned merge strategies?
27. **Empirical Claims**: What are the main claims? (speed, plan quality, generalization, sample efficiency)
28. **Tables & Figures**: What plots would best convey results? (learning curves, performance matrix, speedup)
29. **Limitations**: What are honest limitations? When does the approach fail?
30. **Future Work**: What are natural extensions? (hierarchical merging, multi-policy ensemble, better reward)
31. **Reproducibility**: Can readers reproduce your experiments? Code/data availability?
32. **Comparison Fairness**: Are you comparing against strongest baselines? Any weak comparisons to strengthen your claims?
33. **Generalization Limits**: On what domains/problems does it fail? Any systematic failure modes?
34. **Computational Analysis**: What is the break-even point? When is learned strategy worth the overhead?
35. **Graph Attention**: Should you use attention mechanisms to focus on critical edges?
36. **Constraint Satisfaction**: Can you encode domain-specific constraints (e.g., "don't merge TSs far apart in causal graph")?
37. כרגע איך שזה יוצא, אנחנו בונים בעצם סוכן שיודע לבחור מיזוג באופן גרידי, כי זה מיזוג אחד נבחר בכל פעם, הוא לא מסתכל קדימה על האם המיזוג הזה יתאים או יהרוס, וגם לא מסתכל אחורה לצורך העניין, האם זאת הגישה הנכונה, או צריך לשנות גישה?
38. האם הדרך בה הREWARD בנוי מבטיחה שאם MERGE טוב יקבל REWARD גבוה (וחיובי?) וMERGE רע יקבל REWARD נמוך (ושלילי?), איך האיכות של MERGE מתבטאת בREWARD?
39. אנחנו צריכים לצמצם היפר פרמטרים בהיבט של פונקציית הREWARD שלנו?
40. כמה זמן אנחנו רוצים להשקיע באימון המודלים שלנו?
41. איך ניתן לשפר משמעותית את הייצוג הגרפי של הבעיה, כי בסוף זה רשת GNN, זה כל הרעיון, על ידי בחירת הקשת הנכונה למיזוג? בסוף כל האינפורמציה צריכה להתנקז לבחירה של ה-GNN, וצריך שכל הלמידה וכל המידע שאנחנו מקבלים כפידבק מ-FD, יהיה מקודד ב-GNN, כמה שיותר ממנו יהיה מקודד ב-GNN, אם ה-GNN חסר משמעות, אז איך נצפה שהוא ילמד משהו? האם יש מנגנון ATTENTION שניתן להחיל ב-GNN כדי שיהיה ATTENTION על הפיצ'רים שאנחנו רוצים עליהם תשומת לב שהם מרמזים / מצביעים על MERGE טוב, על בחירה נכונה של קשת?

* **Q24 (CRITICAL): How to generate problems and domains that we know for sure are valid, are solvable, and are scaled in size and complexity? What does problem size mean? Do we have examples for problem sizes, a problem of different sizes? What is the maximal "hardest" problem we can allow ourselves to deal with for the training of our models?**
* **Q26 (IMPORTANT): How does the shrinking hyperparameter and the max states argument affect our training?**
* **Q26.2: How do max\_states and max\_states\_before\_merge interact?**
* **Q26.4: How do these parameters affect the reward signal we get?**
* **Q11 (CRITICAL): How do we evaluate the success of our training or the quality of our training? Is average reward informative? Or average reward along the last steps of the training which is more indicative? Or the reward compared to baselines? Or the amount of problems it can solve, maybe under a certain time size? Or maybe faster than baselines?**
* **Q11.1: What is the relationship between training reward and final performance?**
* **Q5.3: Is there a minimum number of steps needed?**
* **How does the GCN network depth (2 layers, 3 layers, 4 layers... affect the merges learned?, because graph diameter is probably... dozens if not hundreds wide)**
* **Q14 (IMPORTANT): How does the GNN configuration, the depth, the feature definitions of the GNN, affect the learning of a merge policy?**
* **Q14.1: What is the optimal GNN depth? (2 layers? 5? 10?)**
* **Q14.2: How does hidden dimension affect learning?**
* **Q14.3: What node features are most important?**
* **Q14.4: Should we use attention mechanisms?**
* **Q14.5: How do we validate the feature engineering?**
* **Q15 (IMPORTANT): What other factors maybe influence the learned merge policy, and maybe also hinder us from actually learning a good quality merging strategy with the GNN?**
* **Q15.1: Does graph size (number of nodes) affect learning?**
* **Q15.2: How does graph connectivity affect the GNN?**
* **Q23 (CRITICAL): What is the best way to configure the GNN to output the decisions or the merge strategy? Currently is it step by step communication between fd and python, each merge step? Is it possible to export a sequence of merging decisions and halt? Is it making sense? Does the merging always stop when we are left with one final transition system in the downward framework?**
* **Q17 (OPERATIONAL): How much time will it take to perform our experiments?**
* **Q20 (IMPORTANT): How to bring full explainability to our pipeline and framework of our project to the average joe?**
* **Q20.1: What diagrams/visualizations help explain the system?**
* **Q20.2: How do we visualize GNN decisions?**
* **Q20.3: Can we show which graph features influenced a decision?**
* **Q20.4: How do we explain why a merge is good?**
* **Q33: Is the merge problem even learnable? (What is the theoretical learning complexity?)**
* **Q34: What is the sample complexity? (How many merges to learn good policy?)**
* **Q35: Is there an optimal merge strategy? (Or is it problem-dependent?)**
* **Q36: Can we prove anything about GNN expressiveness for this task?**
* **Q37: What is the relationship between graph structure and optimal merges?**
* **Q38: How well does a trained GNN transfer to different domains?**
* **Q39: How well does it transfer to different solvers?**
* **Q40: Can we use one model for multiple problem sizes?**
* **Q41: What causes poor transfer learning?**
* **Q42: Should we use transfer learning or retrain from scratch?**
* **Q43: Which IPC domains should we use?**
* **Q44: Should we use synthetic or real-world domains?**
* **Q45: How do we select "representative" problems?**
* **Q46: Should we test on domains the model wasn't trained on?**
* **Q47: How do we handle problem size imbalance in benchmarks?**
* **Q63: What are the ways training can fail?**
* **Q64: What are the ways the GNN decision can fail?**
* **Q65: How do we detect and debug these failures?**
* **Q66: Are there safety mechanisms we need?**