# Report

**Mingbo Liu**

**U7517737**

## 1. Exercise 10: Using control knowledge in SAT

In order to reduce the runtime of the planner and restrict its search space, various knowledge rules were explored in this exercise. Unfortunately, due to time constraints and my limited planning expertise, most of the attempted rules did not improve the runtime and, in many cases, even increased it. However, I have retained three rules that showed relatively better performance in the program.

The attempted rules included, but were not limited to:

1. If a package is at its destination, it cannot leave.
2. If a package is in the city where its destination is located, it cannot be loaded onto an airplane.
3. If a package was just loaded onto a vehicle in the previous step, it must be moved.
4. If a vehicle carrying packages was just moved, it must be unloaded.
5. If a package is in the city where its destination is located and it is loaded, it must be moved towards its destination.
6. If a package is not in the city where its destination is located, not at an airport, and already loaded, it must be sent to the airport.

Most of these rules did not yield positive effects and, in some cases, even had negative impact on the runtime. Hence, I have chosen to retain rules 1, 3, and 4 from the above list.

After reasoning and experimentation, I have realized that adding clauses itself increases the computational complexity of the planner. To minimize the runtime of the program as much as possible, it is crucial to restrict the inference paths in as simple clauses as possible. It is evident that the fifth and sixth rules mentioned earlier are highly inefficient, as they increase the workload for both humans and the computer.

On the other hand, the first rule stands out as a simple yet effective constraint that can easily restrict all actions related to a package once it reaches its destination. While I have discovered some patterns, my limited abilities have prevented me from finding more efficient rules.

Overall, I acknowledge the challenge of optimizing the planner's runtime through the addition of control knowledge and constraints. Despite my efforts, I have not been able to identify sufficiently efficient rules within the given time frame and my skill set.

1. If a package is at its destination, it cannot leave.
2. If a package was just loaded onto a vehicle in the previous step, it must be moved.
3. If a vehicle carrying packages was just moved, it must be unloaded.

For the above three rule, their CNF clause are:

1. $\neg p@s \lor p@s+1$
2. $\neg a1@s \lor a2@s+1$
3. $\neg a1@s \lor a2@s+1$

The runtime result is shown in tables below, to avoid the disturbance by the condition of computer, the result is the mean value of three times.

Table 1 Solution time for first step with a valid plan

| Problem | First step of plan | Basic encoding (s) | With Control Knowledge (s) |
|---|---|---|---|
| Logistics03 | 10 | 8.4 | 10.2 |
| Logistics04 | 10 | 9.4 | 9.0 |
| Logistics07 | 10 | 1.7 | 1.5 |
| Logistics09 | 11 | 18.2 | 18.6 |

Table 2 Total time

| Problem | First step of plan | Basic encoding (s) | With Control Knowledge (s) |
|---|---|---|---|
| Logistics03 | 10 | 8.9 | 11.1 |
| Logistics04 | 10 | 10.4 | 10.2 |
| Logistics07 | 10 | 2.3 | 2.3 |
| Logistics09 | 11 | 27.7 | 32.8 |

Based on the results, it is evident that my approach to control knowledge has been ineffective, resulting in minimal or even negative improvements compared to the baseline program. The differences in runtime are within the error margin, indicating that the control knowledge introduced did not significantly reduce the computational time, mainly due to the low efficiency of the knowledge rules themselves. As mentioned earlier, the time spent on applying these constraints may even outweigh the reduction in branching caused by limiting the search space.

However, the positive aspect is that at least the implemented knowledge rules did not result in UNSAT (unsatisfiable) outcomes. This indicates that the rules were correctly formulated and did not introduce contradictions.

Overall, while my control knowledge approach did not yield significant improvements in terms of computational time, the implemented rules were valid and consistent.

## 2. Exercise 11: Understanding Planning Problems

The program was executed for ten problem instances in three domains under eight different configurations. A total of 240 data points were collected. After excluding the timeout data, there were 133 data points remaining. To analyze the performance of the program under different conditions, I extracted the data on the total runtime from the logs and created a line graph. The data is presented in the following graph:

The three domains are depot, miconic and rovers. No UNSAT or error reported among the 240 samples of the experiment. Those problems which are time out are assigned as 100s.

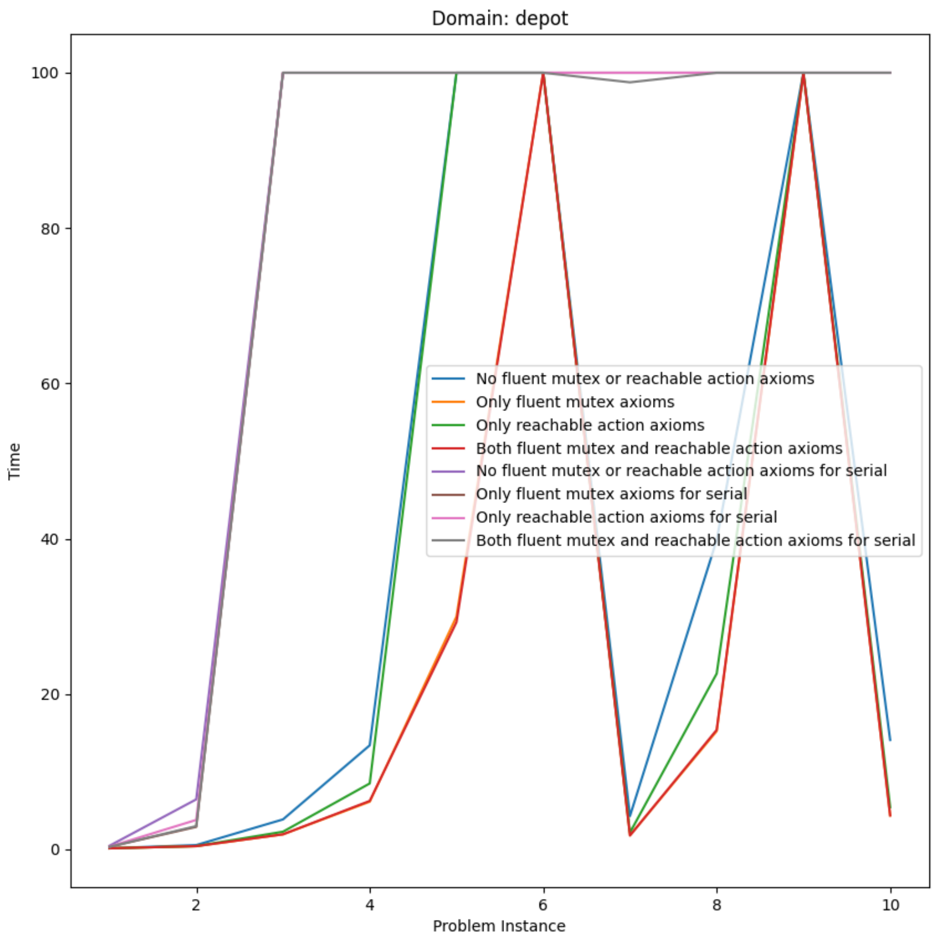

Figure 1 running time-problem number line chart of domain depot

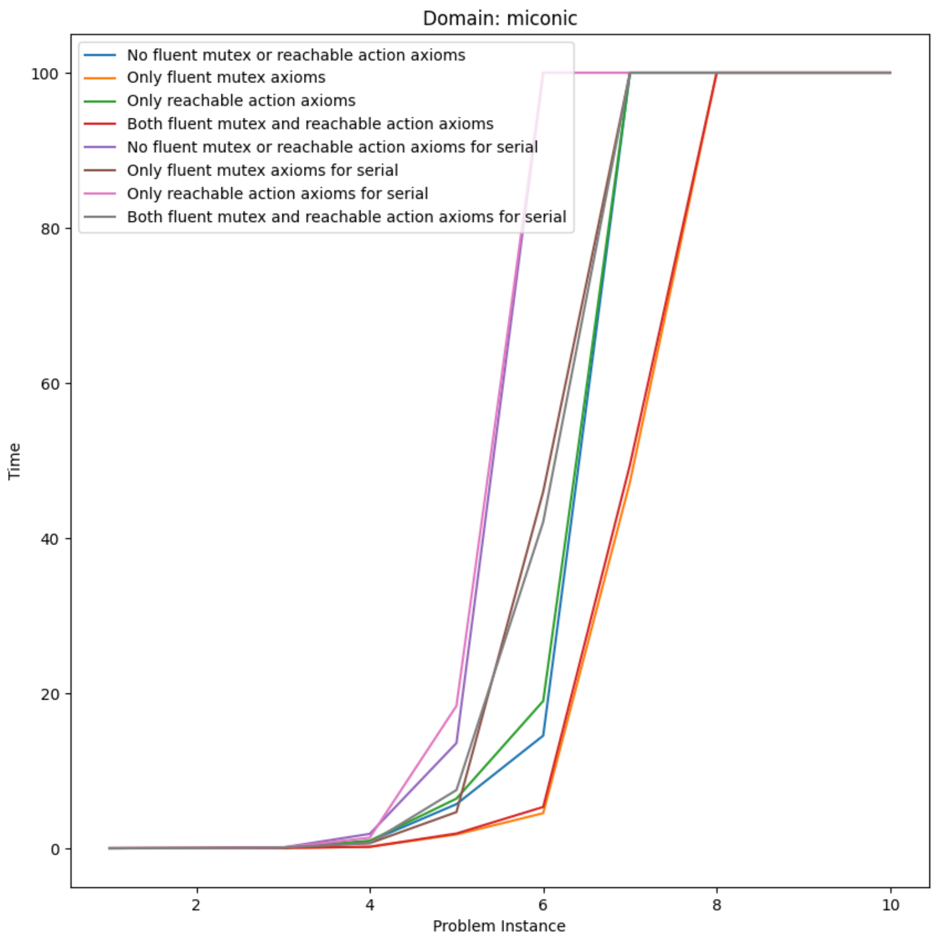

Figure 2 running time-problem number line chart of domain miconic

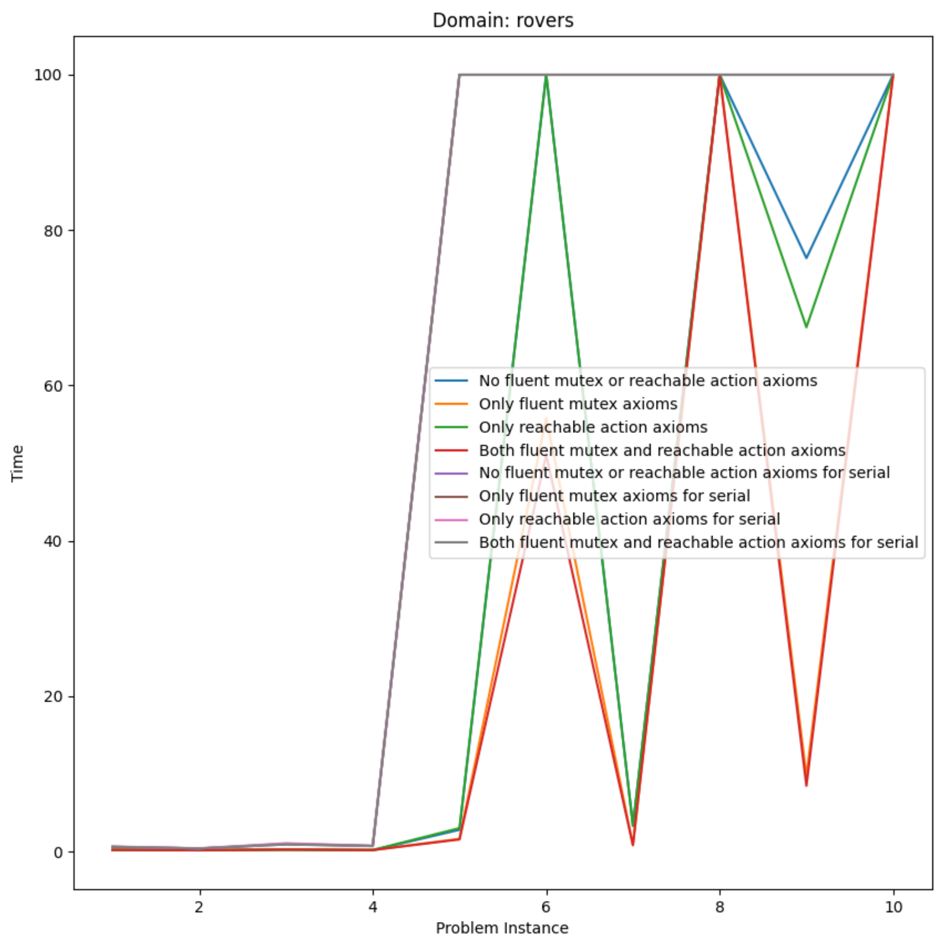

Figure 3 running time-problem number line chart of domain rovers

The three images above are simply too chaotic, making it difficult to discern any patterns or comparisons from them. However, overall, it can be observed that questions with higher numbers have greater computational complexity, which aligns with the nature of the tasks. Detailed comparisons will be presented in the following two sets of images.

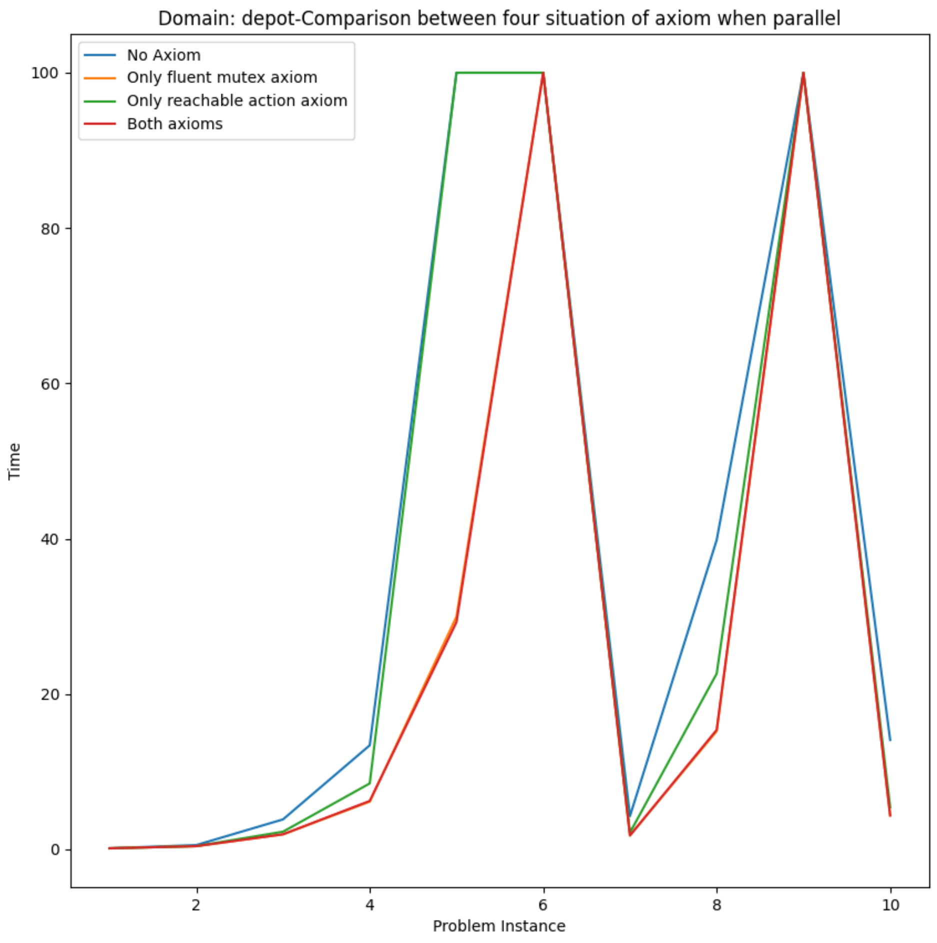

Figure 4 Comparison between four situations of axiom when parallel of domain depot

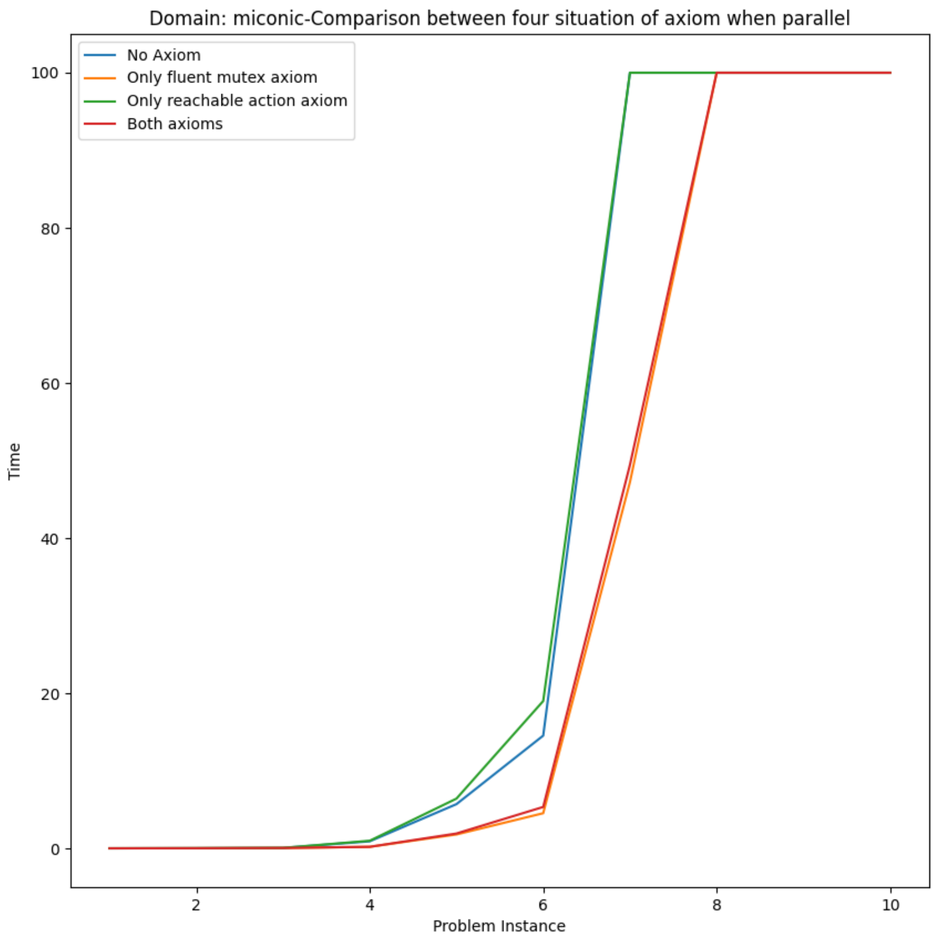

Figure 5 Comparison between four situations of axiom when parallel of domain miconic

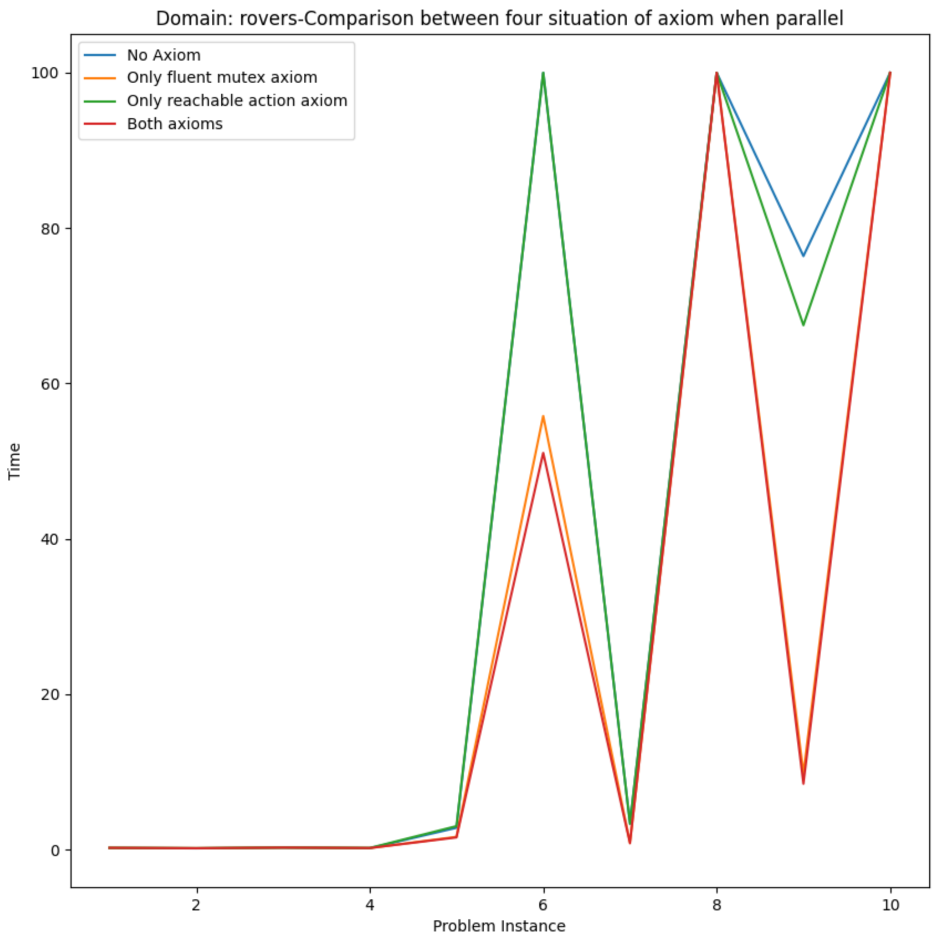

Figure 6 Comparison between four situations of axiom when parallel of domain rovers

Figure 4-6 illustrates the performance of different domains in terms of runtime when actions are executed concurrently. Compared to the previous three images, these three images provide clearer insights. For all the problems, the runtime of "No axiom" and "Only reachable action axiom" is similar, while the runtime of "Only fluent mutex axiom" and "Both axiom" is also similar. However, the latter two exhibit significantly lower runtime than the former two.

This indicates that, among the two types of axioms, the "fluent mutex axiom" has a more notable ability to improve runtime efficiency, while the "reachable action axiom" offers almost no improvement in terms of speed. Combining this observation with the analysis in question 10, this phenomenon can be attributed to the relative computational overhead, or the limitation on expansions is not substantial. For certain specific problems, the use of the reachable action axiom may even result in negative improvements (see Figure 5).

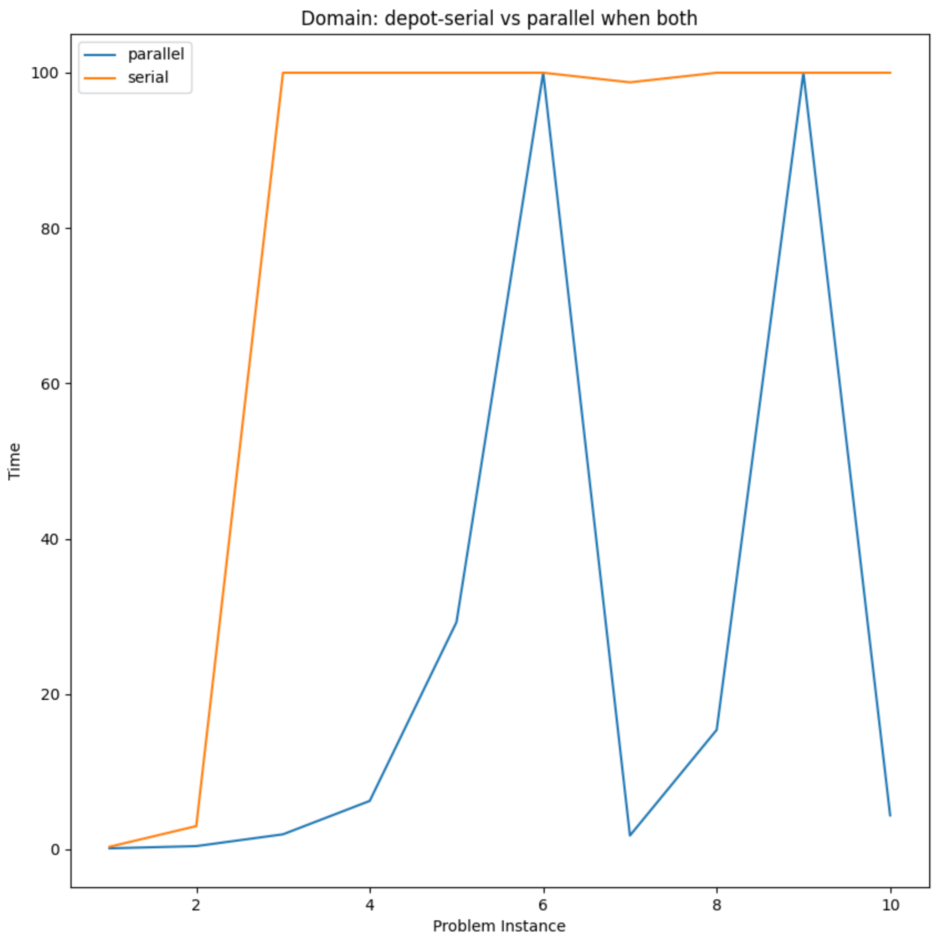

Figure 7 serial vs. parallel of domain depot

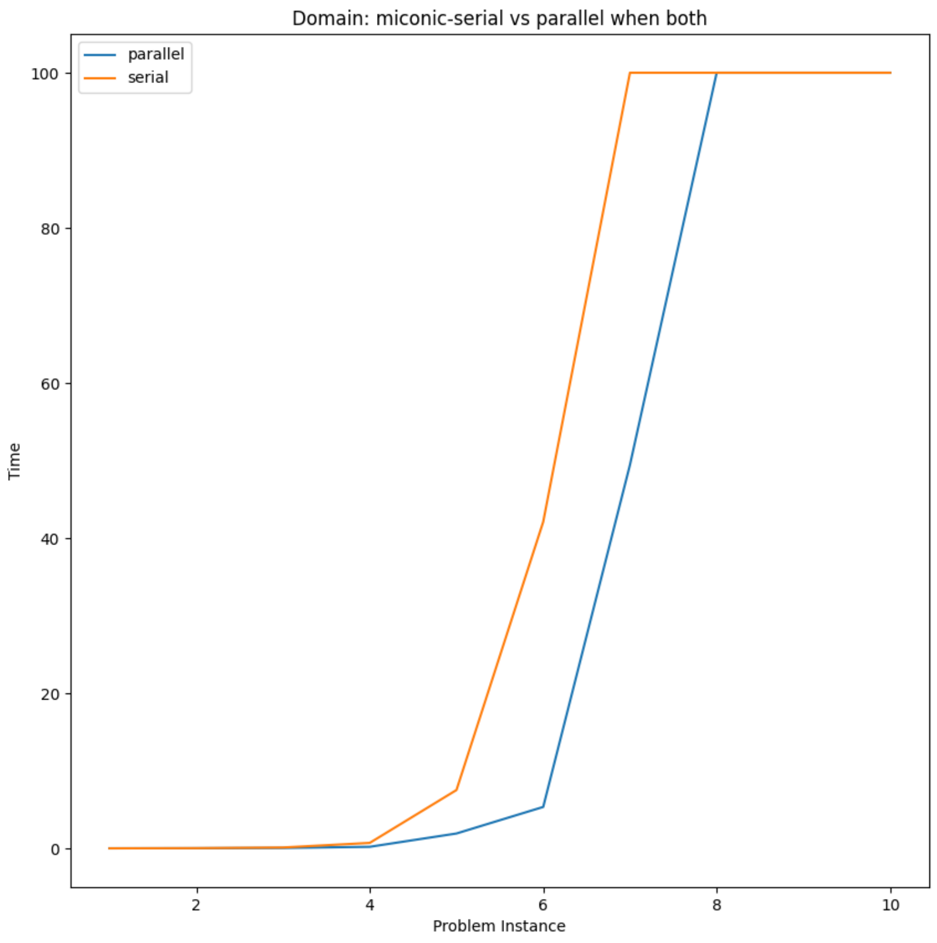

Figure 8 serial vs. parallel of domain miconic

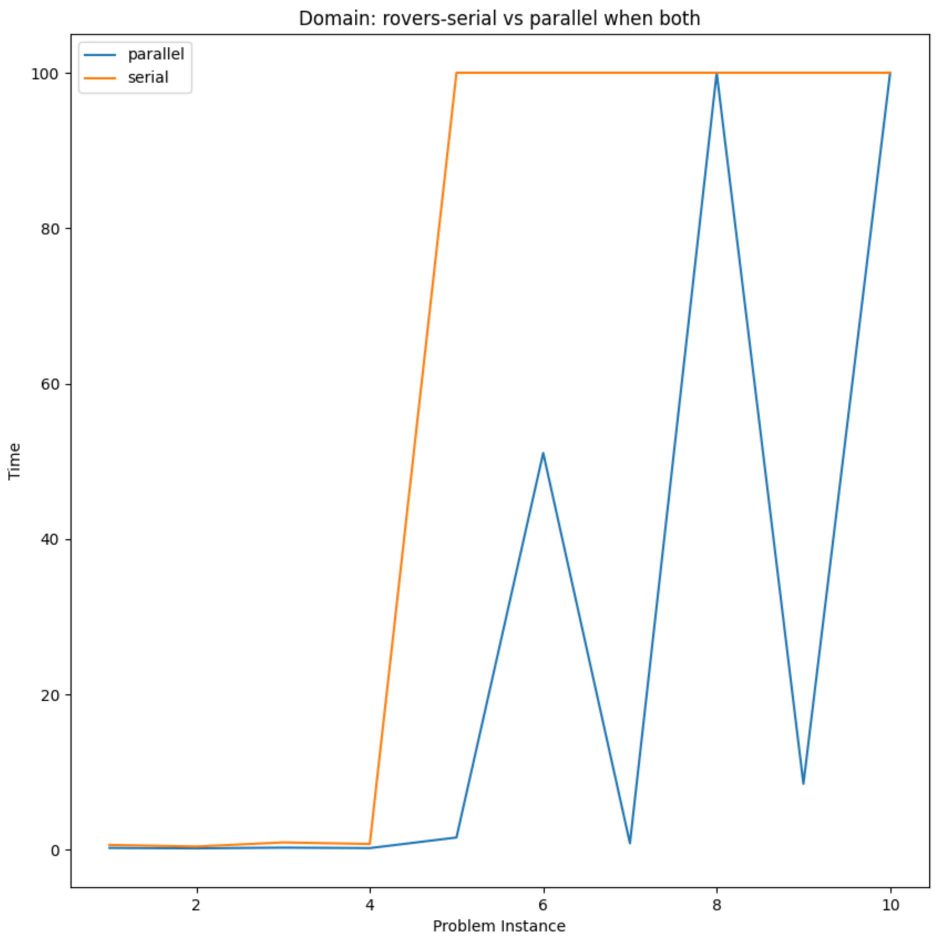

Figure 9 serial vs. parallel of domain rovers

Figure 7-9 illustrates the comparison between serial and parallel strategies when using both axioms in three different domains. It is evident that the serial strategy takes significantly more time, especially for complex problems. The reason is simple: the serial strategy can only take action once in a single turn, while the parallel strategy allows for multiple actions. This results in the serial strategy requiring more steps in all problems where parallel execution is possible. Since the program incrementally tries from one step to 30 steps, it is easy to understand why the serial strategy takes much longer than the parallel strategy.

From an efficiency perspective alone, there seems to be no reason to choose serial strategy execution over parallel strategy execution. Especially in this exercise, where parallel strategy has already taken conflicts and interference into account, there should be no scenarios where only the serial strategy can solve the problems.

In summary, during the search process, using the planning graph, especially with the fluent mutex axiom, can effectively prune the search space and reduce the runtime of the program. In most search problems, parallel planning has a significant advantage over serial planning.