CS4244: Knowledge Representation and Reasoning **Spring 2020**

CS4244 Project: Stage 1

Lecturer: Kuldeep S. Meel *Students: Darren Ong Yun Kai, Nguyen Duy Son*

1 Introduction

Boolean Satisfiability Problem (SAT) is often used in the Computer Science academia as it is one of the fundamental problem in the academia for decades. This SAT problem is helpful in interpreting and solving various problems with a computer. These problems are often used to be encoded into some form of propositional logic and being input to the computer. So, instead of using humans' brain, which may often leads to faults and errors, a reliable computer is used. This problem has been proved to be helpful in various focus area in Computer Science such as Artificial Intelligence, Algorithmic Design and Analysis, Circuit Design and others.

In this project, a SAT Solver is implemented to solve the Einstein's Puzzle. The SAT Solver will take in a Conjunctive Normal Form (CNF) Formula and produce its assignment if the CNF formula is satisfiable, otherwise a "UNSAT" print out message. First, the encoding of Einstein's Puzzle into CNF formula will be discussed. Then, the approaches or algorithm used in the SAT Solver.

2 CNF Encoding of Einstein's Puzzle

To solve the Einstein's Puzzle using the SAT Solver, the puzzle has to be encoded into some form of propositional logic. Then, this propositional logic can be further transformed into a CNF formula which will be able to input to the SAT Solver.

2.1 Translate Einstein's Puzzle to Propositional Logic

The following table shows the encoding for various variables used in the CNF formula.

Nationality		House	H	House Color	C	Beverage	B	Cigar	S	Pet	P
Brit	B	House No.1	1	Red	1	Tea	1	Pall Mall	1	Dog	1
Swede	S	House No.2	2	Green	2	Coffee	2	Dunhill	2	Bird	2
Dane	D	House No.3	3	White	3	Milk	3	Blends	3	Cat	3
German	G	House No.4	4	Yellow	4	Beer	4	Bluemasters	4	Horse	4
Norwegian	N	House No.5	5	Blue	5	Water	5	Prince	5	Fish	5

Figure 1: Variables Identified From the Hints

Next, to represent the hints as propositional logic:

1. The Brit lives in the red house.
   B_{C_1}
2. The Swede keeps dogs as pets.
   S_{P_1}

3. The Dane drinks tea.

   D_{B_1}

4. The green house is on the left of the white house.

   $X, Y \in \{B, S, D, G, N\}$ and $X \neq Y$

   $((X_{C_2} \wedge Y_{C_3} \wedge X_{H_1}) \implies Y_{H_2})) \equiv (\neg X_{C_2} \vee \neg Y_{C_3} \vee \neg X_{H_1} \vee Y_{H_2})$

   $((X_{C_2} \wedge Y_{C_3} \wedge X_{H_2}) \implies Y_{H_3})) \equiv (\neg X_{C_2} \vee \neg Y_{C_3} \vee \neg X_{H_2} \vee Y_{H_3})$

   $((X_{C_2} \wedge Y_{C_3} \wedge X_{H_3}) \implies Y_{H_4})) \equiv (\neg X_{C_2} \vee \neg Y_{C_3} \vee \neg X_{H_3} \vee Y_{H_4})$

   $((X_{C_2} \wedge Y_{C_3} \wedge X_{H_4}) \implies Y_{H_5})) \equiv (\neg X_{C_2} \vee \neg Y_{C_3} \vee \neg X_{H_4} \vee Y_{H_5})$

   In total: $5(X) * 4(Y) * 4(clauses) = 80$ clauses

5. The green house's owner drinks coffee.

   $X \in \{B, S, D, G, N\}$

   $(X_{C_2} \implies X_{B_2}) \equiv (\neg X_{C_2} \vee X_{B_2})$

   In total: $5(X) * 1(clauses) = 5$ clauses

6. The person who smokes Pall Mall rears birds.

   $X \in \{B, S, D, G, N\}$

   $(X_{S_1} \implies X_{P_2}) \equiv (\neg X_{S_1} \vee X_{P_2})$

   In total: $5(X) * 1(clauses) = 5$ clauses

7. The owner of the yellow house smokes Dunhill.

   $X \in \{B, S, D, G, N\}$

   $(X_{C_4} \implies X_{S_2}) \equiv (\neg X_{C_4} \vee X_{S_2})$

   In total: $5(X) * 1(clauses) = 5$ clauses

8. The man living in the center house drinks milk.

   $X \in \{B, S, D, G, N\}$

   $(X_{H_3} \implies X_{B_3}) \equiv (\neg X_{H_3} \vee X_{B_3})$

   In total: $5(X) * 1(clauses) = 5$ clauses

9. The Norwegian lives in the first house.

   N_{H_1}

10. The man who smokes Blends lives next to the one who keeps cats.

    $X, Y \in \{B, S, D, G, N\}$ and $X \neq Y$

    $((X_{S_3} \wedge X_{H_1} \wedge Y_{P_3}) \implies Y_{H_2})) \equiv (\neg X_{S_3} \vee \neg X_{H_1} \vee \neg Y_{P_3} \vee Y_{H_2})$

    $((X_{S_3} \wedge X_{H_2} \wedge Y_{P_3}) \implies (Y_{H_1} \oplus Y_{H_3}))$

    $((X_{S_3} \wedge X_{H_3} \wedge Y_{P_3}) \implies (Y_{H_2} \oplus Y_{H_4}))$

    $((X_{S_3} \wedge X_{H_4} \wedge Y_{P_3}) \implies (Y_{H_3} \oplus Y_{H_5}))$

    $((X_{S_3} \wedge X_{H_5} \wedge Y_{P_3}) \implies Y_{H_4})) \equiv (\neg X_{S_3} \vee \neg X_{H_5} \vee \neg Y_{P_3} \vee Y_{H_4})$

    In total: $5(X) * 4(Y) * 5(clauses) = 100$ clauses

11. The man who keeps the horse lives next to the man who smokes Dunhill.
    $X, Y \in \{B, S, D, G, N\}$ and $X \neq Y$
    $((X_{P_4} \wedge X_{H_1} \wedge Y_{S_2}) \implies Y_{H_2})) \equiv (\neg X_{P_4} \vee \neg X_{H_1} \vee \neg Y_{S_2} \vee Y_{H_2})$
    $((X_{P_4} \wedge X_{H_2} \wedge Y_{S_2}) \implies (Y_{H_1} \oplus Y_{H_3}))$
    $((X_{P_4} \wedge X_{H_3} \wedge Y_{S_2}) \implies (Y_{H_2} \oplus Y_{H_4}))$
    $((X_{P_4} \wedge X_{H_4} \wedge Y_{S_2}) \implies (Y_{H_3} \oplus Y_{H_5}))$
    $((X_{P_4} \wedge X_{H_5} \wedge Y_{S_2}) \implies Y_{H_4})) \equiv (\neg X_{P_4} \vee \neg X_{H_5} \vee \neg Y_{S_2} \vee Y_{H_4})$
    In total: $5(X) * 4(Y) * 5(clauses) = 100$ clauses
12. The owner who smokes Bluemasters drinks beer.
    $X \in \{B, S, D, G, N\}$
    $(X_{S_4} \implies X_{B_4}) \equiv (\neg X_{S_4} \vee X_{B_4})$
    In total: $5(X) * 1(clauses) = 5$ clauses
13. The German smokes Prince.
    G_{S_5}
14. The Norwegian lives next to the blue house.
    $X \in \{B, S, D, G\}$
    $((N_{H_1} \wedge X_{C_5}) \implies X_{H_2}) \equiv (\neg N_{H_1} \vee \neg X_{C_5} \vee X_{H_2})$
    $((N_{H_2} \wedge X_{C_5}) \implies (X_{H_1} \oplus X_{H_3}))$
    $((N_{H_3} \wedge X_{C_5}) \implies (X_{H_2} \oplus X_{H_4}))$
    $((N_{H_4} \wedge X_{C_5}) \implies (X_{H_3} \oplus X_{H_5}))$
    $((N_{H_5} \wedge X_{C_5}) \implies X_{H_4}) \equiv (\neg N_{H_5} \vee \neg X_{C_5} \vee X_{H_4})$
    In total: $4(X) * 5(clauses) = 20$ clauses
15. The man who smokes Blends has a neighbour who drinks water.
    $X, Y \in \{B, S, D, G, N\}$ and $X \neq Y$
    $((X_{S_3} \wedge X_{H_1} \wedge Y_{B_5}) \implies Y_{H_2})) \equiv (\neg X_{S_3} \vee \neg X_{H_1} \vee \neg Y_{B_5} \vee Y_{H_2})$
    $((X_{S_3} \wedge X_{H_2} \wedge Y_{B_5}) \implies (Y_{H_1} \oplus Y_{H_3}))$
    $((X_{S_3} \wedge X_{H_3} \wedge Y_{B_5}) \implies (Y_{H_2} \oplus Y_{H_4}))$
    $((X_{S_3} \wedge X_{H_4} \wedge Y_{B_5}) \implies (Y_{H_3} \oplus Y_{H_5}))$
    $((X_{S_3} \wedge X_{H_5} \wedge Y_{B_5}) \implies Y_{H_4})) \equiv (\neg X_{S_3} \vee \neg X_{H_5} \vee \neg Y_{B_5} \vee Y_{H_4})$
    In total: $5(X) * 4(Y) * 5(clauses) = 100$ clauses

More possibly necessary constraints:

16. Each owner is the resident of exactly one different house.
    Example: If the Brit already owns house no.1, he can't be the owner of any other houses.
    $B_{H_1} \implies (\neg B_{H_2} \wedge \neg B_{H_3} \wedge \neg B_{H_4} \wedge \neg B_{H_5})$
    $\equiv \neg B_{H_1} \vee (\neg B_{H_2} \wedge \neg B_{H_3} \wedge \neg B_{H_4} \wedge \neg B_{H_5})$
    $\equiv (\neg B_{H_1} \vee \neg B_{H_2}) \wedge (\neg B_{H_1} \vee \neg B_{H_3}) \wedge (\neg B_{H_1} \vee \neg B_{H_4}) \wedge (\neg B_{H_1} \vee \neg B_{H_5})$
    Similarly, we can have $B_{H_2}, B_{H_3}, B_{H_4}$ or B_{H_5} as the owner's house.
    Same for S, D, G or N as the owner's nationality.

17. Each owner's house has exactly one different colour.
    Example: If the Brit's house colour is red, it cannot be any other colours.
    $B_{C_1} \implies (\neg B_{C_2} \wedge \neg B_{C_3} \wedge \neg B_{C_4} \wedge \neg B_{C_5})$
    $\equiv (\neg B_{C_1} \vee \neg B_{C_2}) \wedge (\neg B_{C_1} \vee \neg B_{C_3}) \wedge (\neg B_{C_1} \vee \neg B_{C_4}) \wedge (\neg B_{C_1} \vee \neg B_{C_5})$
    Similarly, we can have $B_{C_2}, B_{C_3}, B_{C_4}$ or B_{C_5} as the owner's house colour.
    Same for S, D, G or N as the owner's nationality.

18. Each owner drinks exactly one different type of drink.
    Example: If the Brit's drinks tea, he cannot drink any other types of drink.
    $B_{B_1} \implies (\neg B_{B_2} \wedge \neg B_{B_3} \wedge \neg B_{B_4} \wedge \neg B_{B_5})$
    $\equiv (\neg B_{B_1} \vee \neg B_{B_2}) \wedge (\neg B_{B_1} \vee \neg B_{B_3}) \wedge (\neg B_{B_1} \vee \neg B_{B_4}) \wedge (\neg B_{B_1} \vee \neg B_{B_5})$
    Similarly, we can have $B_{B_2}, B_{B_3}, B_{B_4}$ or B_{B_5} as the owner's drink.
    Same for S, D, G or N as the owner's nationality.

19. Each owner smokes exactly one different type of cigar.
    Example: If the Brit smokes Pall Mall, he cannot smoke any other brands.
    $B_{S_1} \implies (\neg B_{S_2} \wedge \neg B_{S_3} \wedge \neg B_{S_4} \wedge \neg B_{S_5})$
    $\equiv (\neg B_{S_1} \vee \neg B_{S_2}) \wedge (\neg B_{S_1} \vee \neg B_{S_3}) \wedge (\neg B_{S_1} \vee \neg B_{S_4}) \wedge (\neg B_{S_1} \vee \neg B_{S_5})$
    Similarly, we can have $B_{S_2}, B_{S_3}, B_{S_4}$ or B_{S_5} as the owner's cigar.
    Same for S, D, G or N as the owner's nationality.

20. Each owner owns exactly one different type of pet.
    Example: If the Brit keeps a dog as pet, he cannot keep any other types of pets.
    $B_{P_1} \implies (\neg B_{P_2} \wedge \neg B_{P_3} \wedge \neg B_{P_4} \wedge \neg B_{P_5})$
    $\equiv (\neg B_{P_1} \vee \neg B_{P_2}) \wedge (\neg B_{P_1} \vee \neg B_{P_3}) \wedge (\neg B_{P_1} \vee \neg B_{P_4}) \wedge (\neg B_{P_1} \vee \neg B_{P_5})$
    Similarly, we can have $B_{P_2}, B_{P_3}, B_{P_4}$ or B_{P_5} as the owner's pet.
    Same for S, D, G or N as the owner's nationality.

21. Someone owns fish.
    $B_{P_5} \vee S_{P_5} \vee D_{P_5} \vee G_{P_5} \vee N_{P_5}$

22. Each house belongs to exactly one owner.
    Example: If House 1 belongs to the Brit, it cannot belong to any other owners.
    $B_{H_1} \implies (\neg S_{H_1} \wedge \neg D_{H_1} \wedge \neg G_{H_1} \wedge \neg N_{H_1})$
    $\equiv (\neg B_{H_1} \vee \neg S_{H_1}) \wedge (\neg B_{H_1} \vee \neg D_{H_1}) \wedge (\neg B_{H_1} \vee \neg G_{H_1}) \wedge (\neg B_{H_1} \vee \neg N_{H_1})$
    Similarly, we can have $B_{H_2}, B_{H_3}, B_{H_4}$ or B_{H_5} as the house.
    Same for S, D, G or N as the owner's nationality.

23. Each house colour belongs to exactly one owner.
    Example: If red is the Brit's house colour, it cannot be any other owners' house colours.
    $B_{C_1} \implies (\neg S_{C_1} \wedge \neg D_{C_1} \wedge \neg G_{C_1} \wedge \neg N_{C_1})$
    $\equiv (\neg B_{C_1} \vee \neg S_{C_1}) \wedge (\neg B_{C_1} \vee \neg D_{C_1}) \wedge (\neg B_{C_1} \vee \neg G_{C_1}) \wedge (\neg B_{C_1} \vee \neg N_{C_1})$
    Similarly, we can have $B_{C_2}, B_{C_3}, B_{C_4}$ or B_{C_5} as the house colour.
    Same for S, D, G or N as the owner's nationality.

24. Each type of drink belongs to exactly one owner.
    Example: If tea is the Brit's drink, it cannot be any other owners' types of drink.
    $B_{B_1} \implies (\neg S_{B_1} \wedge \neg D_{B_1} \wedge \neg G_{B_1} \wedge \neg N_{B_1})$
    $\equiv (\neg B_{B_1} \vee \neg S_{B_1}) \wedge (\neg B_{B_1} \vee \neg D_{B_1}) \wedge (\neg B_{B_1} \vee \neg G_{B_1}) \wedge (\neg B_{B_1} \vee \neg N_{B_1})$
    Similarly, we can have $B_{B_2}, B_{B_3}, B_{B_4}$ or B_{B_5} as the type of drink.
    Same for S, D, G or N as the owner's nationality.

25. Each type of cigar belongs to exactly one owner.
    Example: If Pall Mall is the Brit's brand of cigar, it cannot be any other owners' brands of cigar.
    $B_{S_1} \implies (\neg S_{S_1} \wedge \neg D_{S_1} \wedge \neg G_{S_1} \wedge \neg N_{S_1})$
    $\equiv (\neg B_{S_1} \vee \neg S_{S_1}) \wedge (\neg B_{S_1} \vee \neg D_{S_1}) \wedge (\neg B_{S_1} \vee \neg G_{S_1}) \wedge (\neg B_{S_1} \vee \neg N_{S_1})$
    Similarly, we can have $B_{S_2}, B_{S_3}, B_{S_4}$ or B_{S_5} as the type of cigar.
    Same for S, D, G or N as the owner's nationality.

26. Each type of pet belongs to exactly one owner.
    Example: If dog is the Brit's pet, it cannot be any other owners' types of pets.
    $B_{P_1} \implies (\neg S_{P_1} \wedge \neg D_{P_1} \wedge \neg G_{P_1} \wedge \neg N_{P_1})$
    $\equiv (\neg B_{P_1} \vee \neg S_{P_1}) \wedge (\neg B_{P_1} \vee \neg D_{P_1}) \wedge (\neg B_{P_1} \vee \neg G_{P_1}) \wedge (\neg B_{P_1} \vee \neg N_{P_1})$
    Similarly, we can have $B_{P_2}, B_{P_3}, B_{P_4}$ or B_{P_5} as the type of pet.
    Same for S, D, G or N as the owner's nationality.

2.2 Translate the Propositional Logic to CNF Formula

In conclusion, we have **125 variables**, with **931 clauses** in total.

The Einstein puzzle's CNF formula, $F = \bigwedge$ (all clauses in section 2.1). This CNF formula is encoded into a .cnf file named **einstein.cnf** following DIMACS format. The file can be found inside project folder.

3 SAT Solver

The SAT Solver is implemented with Conflict Driven Clause Learning (CDCL) Algorithm. The CDCL algorithm has 5 major parts which are Unit Propagation, Conflict Analysis, Branch Picker, Get All Assigned Variables and Backtracking. In this section, we are going to discuss the heuristics on Branch Picking and Conflict Analysis. A general approach of CDCL algorithm implementation is shown below. [MSLM09]

Algorithm 1 CDCL (φ, ν)

```
if UnitPropagation(φ, ν) == CONFLICT then
  return UNSAT
end if
decisionLevel ← 0
while not AllVarsAssigned(φ, ν) do
  (x, v) = PickBranchingVariable(φ, ν)
  decisionLevel ← decisionLevel + 1
  ν ← ν ∪ (x, v)
  if UnitPropagation(φ, ν) == CONFLICT then
    β = ConflictAnalysis(φ, ν)
    if β < 0 then
      return UNSAT
    end if
  else
    Backtrack(φ, ν, β)
    decisionLevel ← β
  end if
end while
return SAT
```

Figure 2: General Approach for CDCL Algorithm

3.1 Branch Picking

Branch Picking is a process where picking a variable from the given list of variables and assign it with a truth value, true or false. This process may seem to be simple and direct. However, choosing a correct variable and assigning it with a correct truth value is important, and is counterintuitive. The correct assignment of truth value to a variable would impact the number of satisfied clauses containing the variable. This could further reduce the process of CDCL and improve the efficiency of the SAT Solver. Thus, the problem lies within the question on how to efficiently assign a truth value to a variable. This is an ongoing problem in Computer Science that is still being researched on since the invention of the CDCL algorithm.

In this project, there are a few heuristics for branch picking has been implemented and used. The approaches and strategies used in different heuristics will be compared and discussed.

1. **Linear Picker**

   Linear Picker picks the variable in its order, eg: 1, 2, 3, ..., n. Each variable picked is randomly assigned to a truth value, true. If the variable that is next in order already implied by other clauses, this variable will be skipped.

2. **Random Picker**

Random Picker picks the variable in an arbitrary order. Each variable picked is randomly assigned to a truth value, true. If the variable that is next in order already implied by other clauses, this variable will be skipped and another variable is randomly chosen.

3. **VSIDS Picker**

   VSIDS Picker picks a variable with the highest score for branching over a period of time. The score of a variable is calculated based on the number of occurrences over all clauses, including new clauses learnt from conflict analysis. The order of the variable regards to the highest score is maintained using a maximum Priority Queue. This period of time can be the number of times of conflicts occurred. Let the number of time of conflict occurs, n, be 256. Whenever the number n is a factor of 256, the scores of the variables in the Priority Queue will be cut by half. This is to ensure that the variables that involves in the most recent learnt clauses get prioritised.

3.1.1 Findings

The differences between the 3 has been measured and compared in terms of execution time, number of times a backtracking has been invoked. The measurement is done by running different cnf files where it is considered to be complicated enough to be a benchmark. (Note: Random Picker is tested for 3 times for each cnf files to get an average value. Detailed Data can be found in `input/sat_solver_stats.txt`.)

Execution Time Comparison

Pickers	Linear Picker	Random Picker	VSIDS Picker
`count_9_3.cnf`	125 ms	200 ms	121 ms
`php_5_4.cnf`	110 ms	142 ms	126 ms
`kcolor_3_grid_15_15.cnf`	721 ms	890 ms	653 ms

Figure 3: Execution Time Comparison between Different Pickers

Backtracking Invocation Comparison

Pickers	Linear Picker	Random Picker	VSIDS Picker
`count_9_3.cnf`	0	0	0
`php_5_4.cnf`	11	11	7
`kcolor_3_grid_15_15.cnf`	0	5	0

Figure 4: Backtracking Invocation Comparison between Different Pickers

3.1.2 Analysis

In this analysis, we will compare the findings on the 3 different heuristics and suggest which heuristic has the highest efficiency.

From Figure 3 shown above, we could observe that linear picker being the second slowest heuristic among the three. This is mainly due to it strictly follows the ascending order of the variables and only skips when the variable is already implied by an assignment.

Random Picker is the slowest heuristics among the 3. This is due to it chooses variable randomly and by luck, it could choose the most correct variable to be assigned with. In the test run we did, we got random as the slowest as this could be random picker being unlucky. This gives an inconsistency of running time and uncertain amount of backtracking invocation over multiple runs using the same formula.

Nonetheless, VSIDS Picker is the fastest heuristics among the 3. It has spent the least amount of time on solving CNF formulas and invoked the least backtracking process. It is also considered to be consistent on its backtracking invocation process.

Therefore, we can conclude that VSIDS Picker is the best heuristics among the 3 heuristics has been implemented. However, there could be other heuristics which may be better that we did not explore, eg: Dynamic Largest Individual Sum (DLIS) heuristic. There may even be a possibility that there exists a heuristics which will be the most efficient heuristics but not yet discovered by Computer Scientists.

3.2 Conflict Analysis

Conflict Analysis is a process to identify the conflict assignment occurred and decide the level to trace back. This process is one the most detrimental part of CDCL algorithm where it takes the output of branch picking and unit propagation to analyse the conflicts and decide the backtracking decision level. The decision made on the backtracking decision level plays an important as it can decide that the CNF formulas is UNSAT or determining the correct level to trace back. If a correct decision level, which may also be the root of all conflicts, to be traced back is identified, it could reduce the occasion of backtracking being invoked.

However, only one heuristic for Conflict Analysis is implemented in this project which is Unit Implication Points (UIPs) to analyse a conflict. A UIP is where the implication made is inevitable by any other assignments made at the current decision level. This implication graph can help to identify which latest assignment made implies the conflict occur at this level and backtrack to that level.

3.2.1 Findings

During the implementation, we found out that the unit implication graph may lead to learn a new clause which is already existed in the existing clauses. This has lead to an extra check is needed to prevent the same new clauses being added and added again which will grow the size of clauses significantly. Furthermore, this heuristic does not pin point to the root problem where clauses has a conflict. This could be an issue for repeating its assignment on some other variables where the source of conflict does not resolve and causes large amount of time spent on backtracking.

3.2.2 Improvement

Unfortunately, we did not have enough time to figure out any other heuristics can be used for Conflict Analysis. One possible improvement we can think of is, instead of backtracking to the latest conflict decision level, backtracking to the source of the conflicts. This may gradually reduce the invocation of backtracking.

3.3 Summary

Our SAT Solver seems to fail on solving certain CNF formulas which might be caused that the complexity, the high coupling and low cohesion of our code. These affect our code quality overall and make it hard for us to debug. But, it still manage to pass the CNF Formulas that we included.

4 Solution for Einstein's Puzzle

After the Einstein's Puzzle is encoded as a CNF Formula and SAT Solver is completed, we can use the SAT Solver to solve the Einstein's Puzzle. The solution for Einstein's Puzzle is German owns a Fish. Unfortunately, this solution is gotten from the assignments produced by other SAT Solver.

5 Conclusion

The discovery of SAT Solver has been proven to be a critical component in both Computer Science academia and industry. In the academia, researchers have been using SAT Solver to as a fundamental theorem to explore more undiscovered theory or solving problems like P = NP. In the industry, SAT Solver has been used for solving real world problem like assigning dispatch of package.

This project has given us an insight of how the SAT Solver works, the implementation, different heuristics or approaches used. We really appreciate the hard work and the researches done by the computer scientists in the past decades. Without them, we could still be in stone age era where there are no Artificial Intelligence or Automated Problem Solver to help us with our daily life problem.

Finally, we really hope this project could help us in developing an AI to solve a real world problem or even doing a research in computing field in the next few years after we graduated.

We grade our project as grade A-.

6 Reference

[MSLM09] J. Marques-Silva, I. Lynce and S. Malik, "Conflict-Driven Clause Learning SAT Solvers," *Handbook of Satisfiability*, 2009, pp. 136.