Universitat de Lleida

Advanced Programming in Artificial Intelligence

Local Search Report (Research Paper?)

Atakan Göl

Hatice Hüma Kalaycı

Table of Contents

1. Local Search

1.1. What is Local Search in SAT

Local Search is an incomplete method. It is used for finding a solution to a problem.
It is based on iteratively improving an assignment of the variables until all constraints are satisfied. In particular, local search algorithms typically modify the value of a variable in an assignment at each step. The new assignment is close to the previous one in the space of assignment, hence the name *local search*.

The satisfiability problem in propositional logic (SAT) is the task to decide for a given propositional formula whether it has a model. This problem plays a promi- nent role in various areas of computer science, as it is one of the conceptual- ly simplest -complete problems.

SAT local search algorithms generally work as follows: An initial truth assignment is generated (usually randomly). Then, at each iteration, a variable is selected and flipped until either a satis- fying solution is found, or some termination condition applies.

1.2. Aim of the Project

In this Project, our goal was to design and develop a Local Search Solver for SAT.

2. Implementation

2.1. Code and Approach

We used Python 3 to implement our Project.
While testing our Project, we used benchmarks that are generated with rnd-cnf-gen.py.

2.1.1. Algorithm

Our program follows the same basic algorithm as any local search program. However at the start our program doesn't choose the first random solution it finds. Instead, for 0,5 seconds it finds more and more random solutions and evaluates them in the end returns the best 2 solutions to be put into the local search algorithm. This way we aimed to reduce the risk of local maximums/minimums.

2.1.2. Functions

```
def readData(name):
#reads the file and returns the relevant data

def evaluate(clauses,solution):
#returns the number of clauses this solution satisfies
```

def random_guesser(var_count):
#returns a random guess of a specified length

def try_and_remember(var_count,clause_count, clauses, start, cutoff=2, best_count=5):
#returns either the solution (unlikely) or the top solutions with the wrong count

def search_local_random(var_count,sol,offset,clauses,limit):
#runs the local search algorithm, offset is the current number of wrong clauses, limit is the time limit

def get_specific_neighbour(temp,change):
#returns the neighbour of original solution

def print_sol(sol):
# if a solution is found prints out the solution and exits the program

def unlikely(offset,clause_count):
# if a solution isn't found prints out the number of wrong clauses and exits the program

2.2. Running

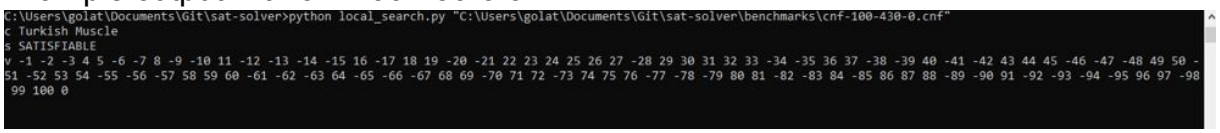
python solver.py "path\benchmark.cnf"

3. Results

Example outputs with cnf-100-200-1.cnf :

```
Huma-MacBook-Pro:satsolvermaster humakalayci$ python3 local_search.py "/Users/humakalayci/Desktop/satsolvermaster/benchmarks/cnf-100
-200-1.cnf"
c Turkish Muscle
s SATISFIABLE
v -1 -2 3 -4 5 -6 7 -8 -9 -10 -11 -12 -13 14 -15 -16 17 18 -19 20 21 22 23 -24 -25 -26 -27 -28 29 -30 31 -32 -33 -34 -35 36 -37 -38
-39 40 -41 -42 43 44 -45 46 47 -48 49 50 51 -52 53 -54 55 -56 57 -58 59 60 -61 -62 -63 -64 -65 -66 67 68 69 70 -71 72 -73 74 -75 -76
 77 -78 79 -80 -81 -82 83 -84 -85 86 -87 88 -89 90 91 -92 93 -94 95 -96 97 98 -99 100 0

c Found solution in 0.7873 seconds

Huma-MacBook-Pro:satsolvermaster humakalayci$ python3 local_search.py "/Users/humakalayci/Desktop/satsolvermaster/benchmarks/cnf-100
-200-1.cnf"
c Turkish Muscle
s SATISFIABLE
v 1 2 -3 4 -5 -6 7 -8 -9 -10 -11 -12 13 -14 -15 16 -17 18 19 20 21 22 -23 -24 -25 -26 -27 28 -29 30 -31 -32 -33 34 -35 -36 37 -38 -3
9 40 -41 -42 -43 44 45 46 -47 48 -49 50 51 52 -53 -54 -55 -56 -57 -58 -59 -60 -61 62 -63 -64 65 66 67 -68 69 -70 -71 72 -73 -74 75 7
6 77 -78 79 80 81 82 83 -84 85 -86 -87 -88 -89 -90 91 92 -93 -94 95 -96 -97 98 -99 -100 0

c Found solution in 0.6317 seconds

Huma-MacBook-Pro:satsolvermaster humakalayci$ python3 local_search.py "/Users/humakalayci/Desktop/satsolvermaster/benchmarks/cnf-100
-200-1.cnf"
c Turkish Muscle
s SATISFIABLE
v 1 2 3 -4 5 -6 7 -8 9 10 11 -12 -13 -14 -15 -16 -17 18 19 -20 21 22 -23 -24 -25 -26 -27 -28 29 -30 31 -32 33 34 -35 36 37 38 -39 40
 41 42 43 44 45 -46 -47 -48 -49 50 51 52 -53 54 55 -56 -57 -58 -59 60 -61 -62 63 -64 65 -66 67 -68 69 -70 -71 72 73 -74 -75 76 77 -7
8 79 -80 -81 82 83 -84 -85 -86 -87 -88 -89 -90 -91 92 93 -94 -95 -96 -97 -98 99 -100 0

c Found solution in 0.7085 seconds
```

Example output with cnf-100-430-0.cnf :



5. References

(https://en.wikipedia.org/wiki/Local_search_(constraint_satisfaction))

(https://www.cs.ubc.ca/~hoos/Publ/jar00.pdf)

(Efficient Implementations of SAT Local Search)

(GUNSAT: A Greedy Local Search Algorithm for Unsatisfiability)