Lillian Walker

Khazhismel Kumykov

Project 4 - CSP

CS 4341 - AI

# How to Run Program

`java -jar lgwalker\_kkumykov\_csp.jar <input.txt> [search type]`

search type is an optional argument:

0 - Just run backtracking search

1 - Backtracking + Heuristics

2 - Backtracking + Heuristics + Forward Checking (Default)

3 - MinConflicts (Note: because of local minima, may need to run multiple times)

Backtracking search variants will output results to “output\_date”, and a log to “log\_file\_date”, as well as printing results to stdout

MinConflicts will only print to stdout

Was built using Java1.8 but I don’t believe it uses any 1.8 features.

## Code Organization

CSPApp contains the Main function

CSP is the Backtracking + extras implementation

MinConflictsSearch is the MinConflicts implementation

Reader parses the input file.

Constraints and the package constraints handle the constraints.

# Approach

Backtracking

Backtracking is a form of depth first search performed by making assignments of values to variables (bags to items) and checking to make sure the assignment is within the specified constraints. If the assignment is valid, the next variable is assigned a value and so on. If an assignment is not valid, backtracking occurs and the program backs up the search tree to the last satisfied assignment and makes another assignment.

Heuristics

We implemented several heuristics. We implemented the minimum remaining values heuristic, which chooses the next variable to assign by choosing the only with the minimum number of possible values. In the case of a tie, we also implemented a degree heuristic which chooses the variable with the fewest associated constraints. We also implemented the least constraining value heuristic for ordering the values to be tried in the variable chosen in the MRV heuristic.

Forward checking used Arc Consistency

Forward checking was accomplished using arc consistency algorithm AC-3. Arc consistency checks an assignment by determining whether the assignment changed the domains of the variables in the associated arcs. If any of those domains is reduced to [], then the arcis not valid as it would result in an unassignable variable. If all of the arcs from the assigned variable are invalid, then the assignment is invalid and backtracking can begin.

MinConflicts

MinConflicts was implemented using the standard MinConflicts algorithm, checking each loop if the solution was found, and if not selecting a random unsolved variable, and finding a spot where it conflicted with the fewest number of constraints.

# Tests Done on Program

Ran against the sample files. We also compared different algorithms with each other to make sure they were self-consistent.

# Strengths and Weaknesses

## Backtracking

### Strengths:

Backtracking performs a depth first search over the variables (in this case items). When an assignment (item to bag) is made, backtracking checks the assignment against the constraints and if the assignment is not consistent “backs up” the tree. This reduces the number of nodes produced compared to trying every combination.

### Weaknesses:

Still has to try a lot of nodes to complete the problem.

## Backtracking + Heuristics

### Strengths:

There are several heuristics which can help the CSP solver choose which nodes to expand. The heuristics we implemented reduce branching, so there are fewer assignments to check.

### Weaknesses:

Heuristics are still not guaranteed to expand correct nodes.

## Backtracking + Heuristics + Forward Checking

### Strengths:

Forward checking reduces the number of checks to the constraints required in backtracking. Our forward checking is accomplished through arc consistency AC-3. The arc consistancy function checks an assignment and checks all of the domains of the values for the variables generated in the arcs from the assignment. If any of the arcs generates an empty domain for any value, the assignment is invalid and backtracking begins.

### Weaknesses:

Does involve increased computation. The worst case scenario for an arc consistency check with a maximum domain of d and c binary constraints is O(cd^3).

## MinConflicts

### Strengths:

Rather simple to implement. When it did succeed (didn’t get caught in a local minima) it did so in a decent number of steps. Our inputs had smaller problems that backtracking was able to deal with quickly, perhaps with bigger problems this would outperform the backtracking.

### Weaknesses:

This method failed a lot due to local minima, and on average did as well or worse than BT + Heuristic + Forward Checking.

# Comparison Table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Problem | Backtracking | BT+Heuristic | BT + Heuristic + Forward Checking | Min Conflicts |
| input1.txt | 3 | 3 | 3 | 1 |
| input4.txt | 15 | 63 | 34 | 5 (2-51) |
| input5.txt | 1879 | 75 | 66 | 194 (88-473) |
| input6.txt | 31 | 47 | 21 | 72 (27-167) |

Since MinConflicts can run into local minima, it can fail a lot. The numbers given for MinConflicts are for when it succeeded. (Each run was limited to max of 5000 checks)