Basic overview: come up with a high level plan using greedy hill climbing and an alternative set of rules; then have A\* fill in the details of moving blocks from one spot to another.

Algorithm can be compared to having a Foreman on a construction site giving out basic instructions (*“move this bundle of rebar to the fourth floor, then move these cinderblocks to the fifth floor*”) and workers who figure out the specifics of executing those instructions without worrying about the big picture (*“to move the rebar I will need to go down to the truck, pick up the rebar, go to the elevator and go up to the fourth floor”*)

Planner uses different rules for moving blocks. Any block with nothing above it can be moved on top of any stack or any open ground. An action for the planner is of the form: move block at position (x1,y1,z1) to position (x2,y2,z2)

Heuristic explanation: distance of blocks from their destination encourages search to move blocks towards their goals. Any block in the correct place doesn’t need to be moved, so it doesn’t contribute any to the penalty. Any block that has an obstacle preventing it from being moved to its goal contributes an extra penalty to the heuristic so that the search is enticed to free blocks that are trapped under others or to move blocks that are preventing others from reaching their goal destination

**Planner heuristic:**

**Heuristic = 0**

**For every block in the wrong place:**

**Heuristic += euclidean distance of blocks current position from its goal position**

**+ large penalty for each block above it**

**+ large penalty for occupying the destination of some other block**

In practice these large penalties were equal to the maximum distance possible between two blocks, because it is always preferable for a block to be on the complete opposite side of the world than to be preventing the block below it from moving. I do believe that more theory could be used in this heuristic; what we did worked for most problems, but may run into trouble in specific areas and may lead to inefficient or sub-optimal solutions.

Find a plan to transition from a start state to a goal state using the planner rules for the simulator (basically, ignoring specifics of how a block gets from one location to another) with a greedy hill climbing search that evaluates each step using the above planner heuristic

**Make plan:**

**Plan = []**

**State = start**

**While state != goal:**

**Create a list of all valid planner actions from current state**

**Find best action by evaluating every subsequent state using planner heuristic and add to plan**

**State = best state found**

**Return plan**

We used a basic greedy hill climbing search because it was perfectly effective, however other search algorithms may either find solutions with fewer step, may run in less time, or may be more robust in the face of unusual circumstance that would thwart the basic hill climbing algorithm. Another possible refinement might involve calculating the distance of the drone from the current position of the block to be moved when evaluating what the best possible action is at each step so that the drone does not spend a whole bunch of time zipping back and forth across the search space

Moving the drone from place to place uses the normal rules of the simulator

**Move drone around using the A\* search algorithm with euclidean distance as heuristic (as documented in other papers)**

**Execute plan:**

**Plan = make plan**

**For every step in plan:**

**Move drone above step.block with A\* search**

**Attach drone**

**Move drone above step.destination with A\* search**

**Detach drone**

This portion of the algorithm is the bridge from the abstract plan created with the hill climbing algorithm to a concrete set of steps with the A\* search.