Shakey the Robot
http://oai.dtic.mil/oai/oai?
verb=getRecord&metadataPrefix=html&identifier=ADA458918

From 1966 through 1972 the Artificial Intelligence Center at the Stanford Research Institute worked on a robot called "Shakey". This project resulted in the advancement of several techniques in the field of Artificial Intelligence including Planning and Search algorithms. The team decided to minimize the focus on hardware and focus their attention to solving the planning problem. This means that they didn't try to miniaturize the components to make a more human looking robot with a small head, or worry about the ability to identify complex objects visually. The team realized that these issues would one day be solved by other advances in hardware and chose to focus their efforts on making an algorithm to solve a domain specific problem that we now call a planning search. The Shakey programmers used a new language called STRIPS (Stanford Research Institute Problem Solver) to define it's world. STRIPS defined Shakey's world as a series of variables representing states and the actions that change states. After the problem is defined in STRIPS it can be solved using predicate calculous. Once the world state, goal state, and actions are defined in a language like strips, a program can execute the potential actions and see the resulting states. This is then repeated until a solution is found. This is the forerunner to the the Planning Domain Definition Languages (PDDL) that are used today to solve planning problems. This was a big advancement in the field of AI. It created a general way to define a problem, actions, and goal so that it could be applied to different problems. The same techniques used to move a robot around a room could be used to solve other problems like the classic airplane cargo problem.

Graphplan
https://www.cs.cmu.edu/~avrim/Papers/graphplan.pdf

Graphplan is an algorithm that's input is a domain definition language like STRIPS to find a planning solution. Graphplan creates a planning graph that contains the constraints inherit in the problem such as mutually exclusive actions at each level. These relationships are called "mutex" as in mutually exclusive. A mutex relation can be one of the following; inconsistent effects when the effect of one action is the negation of another action, interference when the effect of an action is the negated precondition of another, and competing needs when a precondition of one action is mutually exclusive with a precondition of another. Once the plan graph is created, Graphplan searches backward from the goal to the initial state to generate the plan.

Heuristic Search Planner
http://citeseerx.ist.psu.edu/viewdoc/
download;jsessionid=F2B484FB69AF828141076CA76EFA14F4?
doi=10.1.1.43.246&rep=rep1&type=pdf

Heuristic search planners perform a search starting at an initial state towards a goal using a heuristic function. The heuristic function gives a score to each node which indicates how promising it is toward reaching a solution. The node with the best score is searched first. This technique is similar to the A* search we saw with the original search a map problem. In the case of the map search, a manhattan distance between a point on the graph and the solution location could be used to estimate the fitness of a given node. For example, if you are on a point on a map where the distance "as the crow flies" is one mile away from the goal, it would rank higher than a point where the distance is two miles away from the goal. These distances might be irrelevant if there are no straight paths toward the goal, but the calculation can still sever as a good estimate of which point is better. (If we knew the true distance from a point to the goal, we would use it, but since the search isn't complete, we need to estimate.) In the same spirit, with a search plan we can for example count the number of attributes that still need to change before a goal is reached. For example, a node with only one wrong state would be ranked better than a node with two wrong states even though they might not require 1 and 2 actions respectively to be solved. The heuristic function simply gives a quick way of determining what is best mid search to speed up the search.

The research at the Stanford Research Institute was groundbreaking in the field of Artificial Intelligence. The researchers developed STRIPS which was a precursor to Planning Domain Definition Languages used today. These languages enabled the description of problems, solutions, and actions which could be fed into different algorithms to solve the problem. This enabled different planning solvers to be developed independently of the actual problem then used by others. The number of variables needing change greatly increases the amount of time and memory needed to find an optimal solution. Graphplan was created to solve these problems in polynomial time because it creates a planning graph including mutex relationships to cut down on options that need searching. In addition, it starts with the solution and works backward to find an optimal path. Heuristic search planners were also created to increase the speed of the planning solver by adding heuristic functions to evaluate each node. For example, counting the number of variables that are not in a goal state can be used to find the approximate number of actions needed to solve the problem. The heuristic search planner can use this number to analyze which path to search next. The techniques started in the Stanford Labs with Shakey the robot were the groundwork for future developments in the search and planning domain. Advancements like Graphplan and Heuristic Search Planners increase the speed and efficiency of solving these

problems.