A k-armed bandit problem is the approach to the objective of maximizing the total reward. By always selecting the highest estimated value, such actions are called greedy actions (exploitation). Since the estimated values are uncertain, by practicing non-greedy action (exploration) repeatedly, the expected reward might be higher than first time exploiting, and then the optimal solution would be exploiting the actions discovered in long term exploration. Methods of exploring include epsilon-greedy, Upper-Confidence-Bound Action Selection, and Gradient Bandit Algorithms.

Epsilon-greedy method sets up a random probability denoted as epsilon. By most of the time, the greedy actions are performed. With a probability of the value epsilon, greedy behavior is violated and actions are selected randomly. The advantage of epsilon-greedy method is when performed numerous times, all actions would be tried enough times to provide enough evidences to judge the optimal approach. However, the optimal value obtained by this method is only asymptotic, and lack any practical effectiveness. This method is useful for stationary problems. For non-stationary problems, a constant step-size parameter (alpha) is used.

The 10-armed testbed is used to determine the effectiveness of different epsilon values. Higher epsilon values explored more and usually find the optimal action quicker than the lower epsilon values. But higher epsilon values are less likely to select greedily, thus eventually lower epsilon values performs better.

Gradient Bandit Algorithm has a different approach. Not by calculating the action values, but rather calculating action preferences. Initially, all actions have the same preference to be selected. By using the estimate at time t of the expected reward as a baseline, if the reward is higher than baseline, the preference is increased, and vice versa.

It is hard to determine the best method among exploration. 10-armed testbeds could be run to compare performances, but to get a meaningful result we must consider the performance as a function of their parameters (alpha, epsilon). A better method is to summarize a learning curve of each algorithm.