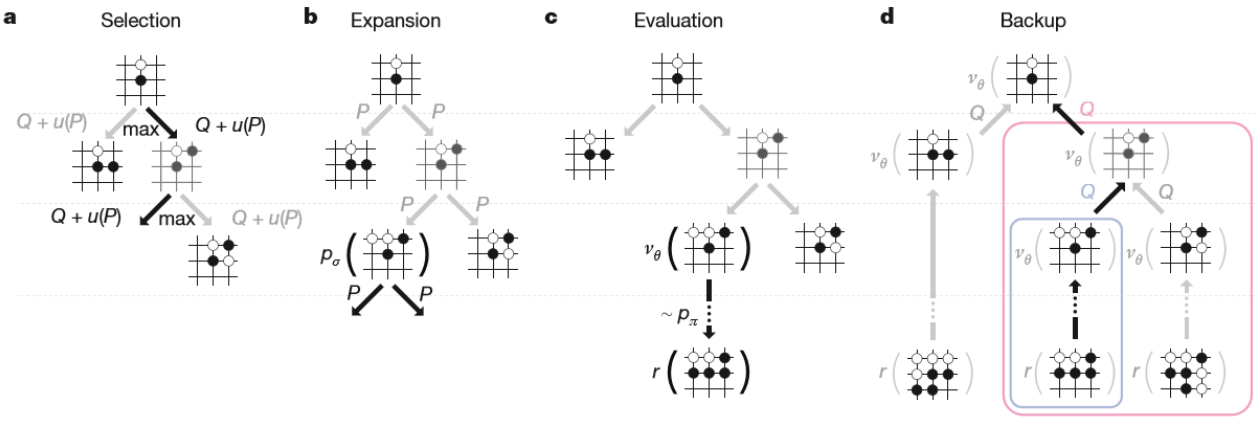
Summary of Alpha Go paper:

This paper summarizes the design principles behind the AlphaGo program, which defeated European Go champion Fan Hui. This was a major milestone in AI field, because scientists thought that superhuman performance in the game of Go was at least a decade away.

The reason why the game Go is particularly difficult for traditional game AI is that the search space is too large. For example, if we want to apply a simple minimax search algorithm to search the endgame in Go the number of possible moves would be 250^{150}. To solve this problem in games like Go, Monte Carlo Tree Search(MCTS) is used.



The backbone of this design is shown at the top (this is figure 3 in the paper). In a traditional depth-limited tree search, what we do is to expand the tree up to a certain depth and use a heuristic function to evaluate the leaves and then propagate this information up to the initial nodes to select an action. What differentiates MCTS is that during the evaluation stage of the algorithm a limited set of nodes are played to the end. This allows us to sample some of the leaves in the end game and propagate this information up the tree. Actually best Go programs before this paper were already using this technique. What makes Alpha Go MCTS successful is how the nodes to be expanded are selected and evaluated.

During the execution of MCTS following four formulas are the key:

$$a_t = \underset{a}{\operatorname{argmax}}(Q(s_t,a) + u(s_t,a))$$

Formula1

$$u(s,a) \propto \frac{P(s,a)}{1+N(s,a)}$$

Formula2

$$V(s_L) = (1-\lambda)v_\theta(s_L) + \lambda z_L$$

Formula3

$$N(s,a) = \sum_{i=1}^{n} 1(s,a,i)$$

$$Q(s,a) = \frac{1}{N(s,a)} \sum_{i=1}^{n} 1(s,a,i) V(s_L^i)$$

Formula4

I think that explaining formulas going from last to first makes more sense. Formula 4 is the simplest. N(s,a) is the visit count to keep track of how many times a node is visited

during simulation. Q(s,a) is the average value of evaluation. During each visit of a particular state action pair a new evaluation result(V_{SL} is obtained and Q(s,a) is updated accordingly. How do we get V_{SL} ?This is formula 3. Evaluation function is a combination of evaluation value coming from neural network $V_{\theta(SL)}$ and z_L which is the result of the end game (win or loss) during a particular simulation. Formula 2 (u(s,a) is a prior probability that comes from RL network, but decay as a node is visited more. Basically, we select nodes initially according to predictions of RL network, but gradually we switch to Q(s,a) values. Each time a decision needs to be made we select an action that maximizes Q(s,a)+u(s,a).

The way evaluation values are obtained from SL and RL networks are also unique. AlphaGo uses deep neural networks in order to evaluate the whole board and assign a probability value to each next possible action. These neural networks consisted of multiple layers of convolutions and non-linear activation functions. Initial neural network called SL(supervised learning) are trained on 30 million positions made by human players and achieved a success rate of around 55% on test data. In the next stage of training RL(reinforcement learning) networks are created as a copy of SL network and initialized the same way, but this time RL networks are allowed to play the game to the end and weights are updated according to the results of the game.

RL policy network were already very good in predicting good moves such that it won most of the games against existing Go programs. In the final stage of training, they trained a value network that is similar in architecture to RL network but outputs a single prediction(win or loss) rather than a probability distribution($V_{\theta(SL)}$ from above). Value network is also trained through self-play.

In conclusion AlphaGo program achieved superhuman performance by combining two methods: computer vision through deep neural networks :to see" and evaluate the moves and monte carlo tree search in order to simulate and search the end game.