1. Uct

对于博弈类游戏，很容易想到的方法就是最大最小树，并用合理的方法剪枝。但这方法并不适用与围棋这类复杂性如此高的棋类游戏。通过上网查阅资料，我们锁定了蒙特卡洛随机搜索方法，简称MC方法。

我们如何才能知道哪一步棋很优呢？蒙特卡洛的方法的核心，就是通过对某一棋局随机模拟很多盘得到其胜率。随机模拟很多盘的主要思想，就是在某一个棋局之上，让黑白子采用随机下法对战，直至终盘，如此往复，模拟多盘，我们就能得到这一棋局的胜率。这一棋局是通过原始棋局下某一步得到的，所以这个胜率也就是这一步棋的胜率。胜率最高的那步棋，就是我们接下来要下的那步棋。

虽然蒙特卡洛方法简单易懂，但它的缺点也很明显，在一定时间内只能模拟出一定棋局数，而且没有将对方下一手考虑进去，等于只搜索了一步，而对于博大精深的围棋来说，只考虑一步是远远不够的。

进一步，我们发现结合树蒙特卡洛搜索和UCB公式的UCT方法。UCB是一个表示结点价值的公式，不仅包含结点的胜率也涉及到了结点被访问的次数，UCB公式是对胜率和被访问次数的平衡。胜率越高UCB分数越高，被访问的次数越少UCB分数也越高，简单理解就是胜率越高说明该结点的赢的可能越高，而被访问的次数少的结点需要给它机会来证明的自己的价值。

随着进度的推进，我们发现最初是的UCB公式似乎并不那么“好”，于是我们去寻求其他paper的帮助，最后通过论文[1]，我们更换了UCB的公式。如下图所示：



* 1. Uct 剪枝

如果事先在树搜索时砍去一些预计不会很好的枝，可以减少uct的遍历，对预计优秀的结点多一些模拟和搜索。我们尝试只将一些我们认为好的着棋作为uct的结点。包括上一步棋周围的某一个范围内的空点，上一步被吃掉的棋串的位置等等，将其中合法的结点作为uct的结点。但尝试下来的结果都不尽人意，都不如之前的uct棋力高。围棋博大精深，限制条件太过绝对，反而效果不佳。

但是后来我们想到可以砍掉一些显而易见不够好的下棋点，比如周围8个点都没有棋的位置（不让棋下在空旷地方），一定步数下棋谱边缘的点等等。在uct不够强的情况下，剪掉一些这类的结点，可以提高一定的棋力。

* 1. Uct结点排序

我们在尝试剪枝的时候发现单纯的去除一些结点效果并不是很好，所以我们想到是否能将uct的结点做一个评分，也就是对每一个可以选择的着棋方法评分，按照这个评分给uct结点排序，分值越高越先拓展该子节点。我们考虑到的有几下几个方面：吃子、救子、紧气、做真眼、连接、切断以及避免填眼，对于符合这些情况的点我们会赋予一定的权重，具体分数由该点的贡献来决定。

这样可以让uct更有针对性，对一些我们认为较好的点进一步拓展搜索。我们也有试过给uct子节点的uctvalue直接赋值，但是效果并不是很好，所以我们仍采用排序的方法。

1. 其他优化方法
   1. 并查集维护棋串及气

虽然这个方法很多组都用到，但是作为提高棋盘速度很重要的一个方法，我们还是想在这里做一下阐述。因为每次判断棋串是否为某一个位置的子提供气时，需要遍历整个棋串，这就导致了很多重复浪费的操作，所以用并查集，为每一个棋串维护一个气，是非常重要的。同时，我们在全局棋盘变量维护的是真气，在uct 模拟时维护的是伪气。前者是为了能为为uct结点排序提供正确的信息，因为其中会涉及到吃只剩一口气的子，需要提供气的具体数目。而在uct 中模拟时，我们没有用到真气的必要，用伪气已经足够了。由于维护真气需要很多操作，所以不再模拟中用真气可以大大增加模拟的局数。

1. 前言

本次报告中，我们会详细阐述我们的技术路线，以及整个实现过程。

主要的思想仍是使用uct，在这之上，我们又做了很多改进。比如剪枝，排序等，除此以外我们还在棋盘上做了一些优化，增加了棋盘的速度。