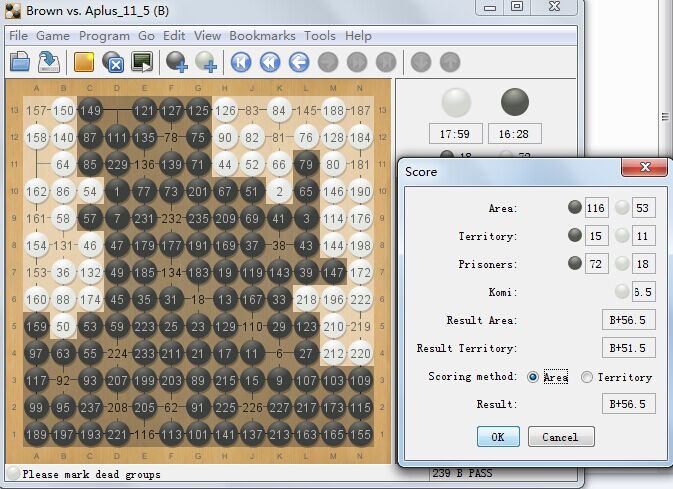
张哲慧 | 5120309056 | 2015年1月17日



人工智能课程报告

个人部分

[[1]](#footnote-1)

Contents

[前言 2](#_Toc409304767)

[个人贡献 2](#_Toc409304768)

[UCT-RAVE实现 2](#_Toc409304769)

[策略优化 2](#_Toc409304770)

[uct公式优化 2](#_Toc409304771)

[测试与调试 3](#_Toc409304772)

[结果与反思 3](#_Toc409304773)

[关于围棋AI 3](#_Toc409304774)

[关于UCT 4](#_Toc409304775)

[关于优化 4](#_Toc409304776)

[感悟与思考 4](#_Toc409304777)

[理论与实践的gap 4](#_Toc409304778)

[看淡结果 5](#_Toc409304779)

[竞争与合作 5](#_Toc409304780)

[作为娘子军 5](#_Toc409304781)

[真正的提高 5](#_Toc409304782)

[致谢 6](#_Toc409304783)

[参考文献 7](#_Toc409304784)

# 前言

个人报告重点写个人贡献，结果与反思，感悟与体会三个部分，个人贡献部分写了下自己做的事，结果与反思讲了一下对于围棋AI的一些思考，感悟部分则写了一些自己在这整个过程中产生的一些感想。由于技术内容在小组报告里都介绍的比较全面了，所以这里侧重点就写自己的感悟和体会。

# 个人贡献

## UCT-RAVE实现

在UCT之余，我们还实现了最基本的AMAF方法。在基本调研之后，我们觉得UCT-RAVE方法应该可以提高围棋的战斗力，因此决定将其添加到现有的UCT中。因为毕竟通过UCT一是不能得到很多的数据，最开始我们的UCT模拟盘数只有一万多，而且收敛的也不够好，我通过修改原有代码，得到了数组实现的RAVE，具体实现方法已经在小组报告的技术实现部分做以阐述。但是发现这样和原有的节点结构就冲突了，而且在多线程中，数组结构会因为读写冲突报错，当时检测了一下单线程状态下的UCT-RAVE的表现，差强人意，所以最后在实际对战中就放弃了这一块。

## 策略优化

策略优化是在uct搜索中，为了增强模拟对战的真实性、有效性，双方对战所采取的策略，我通过查阅相关资料，根据文献中总结的策略优化经验，敲定了hane、cut、border等pattern[2]和，atari denfense, fill board, nakade, capture等策略[1]。并实现了其中的一部分。这一部分主要是因为坐标转换部分有点棘手，基本实现就是完全的匹配问题了。关于其他的战略，由于和气的实现部分比较相关，后来改了很多遍。其实关于这些策略，比如fill board这种走法，其实在我们看来是很不合适的，因为在所有的对战里，都是棋走的越紧越好，因此我们也专门设计了紧气这样的战略，不过为了综合效果还是予以实现，但最终的版本中还是没有用到。

## uct公式优化

uct的公式参数调整对其的表现十分重要，也就是exploitation和exploration的比重问题，在[1]中综述了集中UCB公式，我们也是一个个试了一下，最后敲定了UCB-Tune2来作为最终的公式，在我的检测中，在exploitation方面的权重在开始时可能很大，但万步以后，就很小了。UCT公式的优化效果还是比较明显的，在我们试的4个UCT公式中，UCB-Tune2始终能稳定的战胜所有其他版本的UCT，在测试中发现，我们UCT虽然模拟盘数很多，但是很多集中在胜率高的节点上，比如在7万盘的时候，有2万多盘都在UCT中意的最优节点上，在当前最优棋比较明显的时候这样当然可以，但是在局势不明显的时候，就会导致有些节点的遍历次数有限，从而不能发现那些真正有价值的节点。

## 测试与调试

这部分工作一个是UCT的测试，为了测试UCT，我们尝试过每次输出，但是这样很影响运行效率，也通过输出文本进行过测试。进行测试一是为了检测UCT部分的模拟盘数和收敛速率，一级各项参数的变化。另一个是对战测试，为了检测版本更新后的效果。

这部分工作虽然看起来不值一提，但确实是很重要的一部分，比如说在比赛前一天的时候，我们想出来了一些小trick，排列组合得到了十几个版本，我们几个就来回测试，版本之间比，版本和上一届的程序比，和其他组的程序比，最终敲定棋力最强的一个版本参加比赛。测试这一部分其实是大家都在做的工作，有时谁更新了一个版本，大家就一起测，毕竟这样效率更高。

# 结果与反思

## 关于围棋AI

围棋作为一个博大精深的学问，实在是一门值得人投入一生的事业，虽然开始的时候，我们都是一点围棋都不知道的围棋小白，但经过了这么久的学习，也知道了怎么开局，做活，收盘。在开始的时候，我们就调研了一下已经有的开源代码和相关论文，实话说已经有的工程，比如mogo、fuego、mango、gnugo，有的是开源代码，有的不是但也有很多相关文献，这些都是很宝贵的参考资料，不过确实这些工程太过浩大，我们也难以抽离出我们想要的部分，重点还是通过论文进行理论学习，然后实现其中的一些优化。

其中主要的难点，还是在与如何实现已经如何调整，理论毕竟抽象，或许并不适合我们的项目本身，比如[2]中提到的UCB1-Tuned，我们也试图实现过这个公式，但是毕竟需要记录的数据跟我们之前的数据结构不一样，牵一发则动全身，所以没有采用；还有的优化则是效用不如预期，比如UCT-RAVE，比如模拟中的策略，比如置换表，所以最后也没有用。

## 关于UCT

其实我们的UCT每秒可以模拟上万盘，和mogo的战斗力是差不多的，但是棋力却相差很大，据说是mogo中的参数设置的很巧妙，可能还是因为我们修为不够，所以最后并没有得到最合适的公式，不过相比之下一直在提高，也算是对得起我们的努力了。

其实真说，虽然UCT算法是我们写出来的，但最后反倒是我们要学习他更多。有时候觉得UCT给出的最优位置并不好，但仔细体会之下，也发现这一步好像有着更长远的效益，毕竟在模拟中确实可以得到量化的棋子价值，而不是像我们旁观者只是看到短期的吃子救子带来的效益，其实我们在最后给吃子和救子很大的优先权，也不知是否伤害了最纯真的UCT。

其实不管是策略，还是各种快速判断的trick，本质都是为了弥补UCT在随机模拟中可能会误判权重的问题，而我们也尽最大可能去利用我们自己的经验，去设置快速判断，选择合适的一步棋。或许是我们不够信任UCT，或许是我们没把UCT发挥到最好，但实践中，结合了快速判断确实提高了棋力，所以虽然他违背了UCT的准则，但也是把短期效益发挥到了最大。

## 关于优化

其实我们后期的优化主要集中在提高UCT的效率上面，虽然试图提高其中的策略性，但发现带来的速率的降低对棋力的影响十分大，虽然不是很能理解UCT是如何从傻瓜对局中抽取出有用的步骤，但实践证明，模拟盘数的上升总能稳定的带来收益，所以就还是集中在这一块了。其实这也很奇怪，因为在测试中大概3万盘的时候，棋子的胜率已经比较稳定了，按理说如果把模拟盘数缩减到这个水准应该是没有影响的，但是在实际的测试中，模拟盘数较低的还是被模拟盘数多的完败，即使更有策略性，可能是由于测试具有偶然性，或者在局势比较复杂的时候，还是搜索深度越深越好。个人感觉我们最后的UCT公式选的还是比较巧妙的，毕竟是理论上有依据，实际效果也很好，但是事实上我们可能还是没有设计好最优的权重。

我们后期对于优化其实还有一些考虑，但是却没有时间去做这些优化，或者去测试这些方法是否有优化效果了，这些优化大致有:

* GPU编程

这部分我们构想已久，但苦于没有途径，但大概在比赛前两天的时候，陈悦莹对这方面有了一些眉目，也辛辛苦苦把环境搭建了起来，但最后还是由于实现过于繁杂而作废，这也算是一个遗憾吧。

* 收官部分

在之前我们讨论过是否应该通过一些方式在即使输的情况下也能尽量是输的目数最少，我们也尝试了通过把原有的get\_score函数变成非布尔的函数，返回赢或者输的目数，但是发现这样反而降低胜率，大致是因为，目数这一量化标准不能合适的反映棋子的价值。而且毕竟比赛最后也只看胜负，但在实际比赛中，小负还可以有加分，所以收官部分确实是可以有些调整，从而最大化己方的领域的。

* UCT公式的调整

没有实现UCB-TUNE1公式算是比较遗憾的一点，由于之前一直集中在策略方面下功夫，我们大概比赛前两天才想到通过修改UCT中的权重来调整，最后虽然得到了一个相对较好的公式，但毕竟权重变化那么多，这其中肯定还有很多变化我们没有涉及到。

* 策略的效果

实现阶段很懊恼的一点就是很辛苦写的策略效果却不好，因为加了策略过后，棋反而走的更散，最后只能用快速判断的方式去弥补，我们也尝试过把这些候选点加权送到UCT中，增大这些位置的初始权重来得到最优位置，但效果并不好，最后其实也有想法，就是根据UCT最终得到节点的胜率和访问次数，确定这个点的真正价值，然后在与快速判断得到的位置结合，选出比较好的位置。但这只是想法，却遗憾的没有实现。

# 感悟与思考

人工智能这门课可以说是这个学期最有趣的课了，之前选这门课的时候就知道最后会有期末大作业，每个组做围棋AI最后还有对战！虽然也能想象出来为了优化最后会很辛苦，但还是坚定的选了这门课。

## 理论与实践的gap

平时的理论课程看起来和大作业部分没有什么关系，但是也是很有用的，尤其是前一部分，启发式搜索相关的内容在最后大作业中都有体现，不过理论和实践毕竟风格不一样、侧重点不一样，中间还有有个很大的gap，所以最后实现起来很多都要依靠自己去思考如何降低复杂度，如何提高效率和效果。

其实感觉大学学了这么多课，很多都是考前背背书就过了，虽然也有大作业，但一般也不是竞技性的，不会有个量化排名，大家自己做自己的，辛苦一段时间做出来，展示一下，最后一般给分也都不错。感觉这次大作业最大的特点就是，不是做得多就做的好，经常辛辛苦苦实现一些东西，最后用了这些优化方法的版本反而不如没有经过优化的，所谓事倍功半，也就是这种感觉了。

## 看淡结果

我们组算是比较提前做的了，之前uct做出来，就写好了吃子救子pattern等策略优化，在比赛前一周的时候跟关系比较好的两个组比了一下，能赢个60子+的，感觉很是不错，不过当时这两组的AI也都不是很完善，过了几天过后，其中的一组就过来草虐我们了。再有就是比赛前两三天的时候，跟”天元圣手”他们组比了一下，被屠城了，当时心情差的都不想复习了，所以后面也试图做了很多像和他们对抗一下，但最后也没有的到很好的成果，在最终的比赛中也确实输了，虽然很伤心，但也算意料之中。

其实做优化比较难的一点也就是不知道怎么去衡量，优化了的是不是就是好的，比如在11版本上，我们做了9个版本，做出各种调试，其中大概四五个版本都是超过前一个版本的，但在对战其他组的时候，表现相差无益，虽然有时能赢天元圣手，或者上届的undecidable，但是也都不稳定。

## 竞争与合作

像这种竞技性大作业，不仅仅是闭门造车那么简单，比如说，我们还需要时时关注别的组的进展，了解其他组在做什么优化，当前棋力如何，如何针对别人的战略思考对策，通过对战来相互提高。比如说跟”神之一手”，跟他们的交流算是相对比较多的。在我们version4.x的时候，他们还没有做很多优化，跟他们比，可以赢60子+，过了几天，他们棋力有了比较大的提升后，跑过来要求跟我们一战的时候，就可以赢我们30子+了，当时心里虽然很不爽，但是这种受挫也给了我们继续优化的动力，等到我们version7.x的时候，就可以先后手稳赢他们了。有时候赢了别人也会觉得不好意思，输了也会不开心，但这些小情绪都会被继续努力变得更强的动力取代，大家一起讨论，研究，虽然有些核心技术没办法共享，但能够沟通的也是帮助很大了。

## 作为娘子军

作为一个纯妹子组，实话说相比汉子组还是有点劣势的，比如说熬不起夜，但是相信我们的优点也有很多，比如说认真仔细，准备充分。其实我们能选这门课也都是做好了辛苦码代码的准备，而且作为CSer，说起来我们也都挺喜欢编程，所以倒不至于像大多数人以为的妹子就是拖后腿的。其实最后的对战成绩也证明了我们的实力也是还可以的，虽然相信我们还有很多可以继续优化的地方，但是毕竟不能尽善尽美，而且对战总有输有赢，看淡点结果，重视点过程，感觉学习了很多，收获也有很多，这些应该就够了。

## 真正的提高

经过这次大作业，我相信每个人应该收获都很多，应该也不尽相同，就我个人来说，主要的提高应该还是在实践方面，下面简单说说：

* 理论方面：
  + 在课程里，感觉学的最有用的就是搜索部分，因为最大最小树，还有一些基本的剪枝思想都在大作业中有所涉及，还有就是关于推断部分，贝叶斯网络的建立，滤波，像卡尔曼滤波在之前也是有用过。不过确实很多部分也没能吃透，感觉有点羞愧，但是真的学到了一些东西，总归会有用的。
  + 在完成大作业的过程中，很有用的一方面资料就是前人文献，我们的主要路线还是参考mogo的，但其实fuego和mogo很像，而且也有开源代码，所以参考价值也很高，不得不说，如果没有这门课，我应该永远都不会知道一个AI可以经过这么多代的优化，那么多group一起对战、提高，感觉十分奇妙。在我们的竞技中其实也有些这种感觉。
* 实践方面：
  + 编程能力的提高，这是毋庸置疑的，整个AI项目代码量应该相对来说是所有C/C++项目里最强的了，之前开发的安卓代码虽然长但是很多事UI相关并不麻烦，而这个则是相对来说全靠自己架构的“干货”。有时出现了bug，需要一层一层去测试才能发现到底是哪里出错，有时改了一个地方，另一个部分却受到影响，这些经历对我们的能力提高都是大有裨益。
  + 对c的熟悉也是一方面，以前并不熟悉c语言，因为基本都是用python或者C++。我们第一次用C语言开发项目，平台并不难搭建，交互平台、框架、也都给了我们，其实我们要做的也就是实现自己的AI。通过这次大作业，也算对C有了更多了解，其实也没有什么大的差别，但是有时候想用一些库的时候，会突然发现这个库在C里面没有。
  + 团队精神与竞技精神。这并不是我们第一次团队开发项目，但应该是最纠结的一次，为了优化我们走了太多的弯路，这其中有很多的失落，也有很多的惊喜，每次棋力的提高，每次版本的更新，赢了输了又赢了，这些都给过我们快乐和忧伤。通过这次比赛，也更能感受到同学间的友谊，虽然大家处在竞争的位置上，但都还能愉快的合作，拿出最强战力的exe比拼，并共同提高，也是极好的了。

# 致谢

最后，还要感谢张老师的悉心教诲，助教的热心指导，和同学们的关心与帮助，如果没有这些师长，同学的帮助，我们肯定无法取得现在的成绩，谢谢你们！

# 参考文献

**[1]Chaslot G, Chatriot L, Fiter C, et al. Combining expert, offline, transient and online knowledge in Monte-Carlo exploration. 2008.**

**[2]Gelly, S., Wang, Y., Munos, R., & Teytaud, O. (2006). Modification of UCT with patterns in Monte-Carlo Go (Technical Report 6062). INRIA.**

**[3] Gelly S, Silver D. Monte-Carlo tree search and rapid action value estimation in computer Go. Artificial Intelligence, 2011, 175(11): 1856-1875.**

1. 封面图片为Aplus11\_5与上届undecidable【第一】的对战 [↑](#footnote-ref-1)