SAI引擎改善征子计算的方法

1 PUCT决策函数

在蒙特卡洛树搜索（MCTS）的计算过程中, 算法会基于2个方面进行下一步的价值评估：
未来走向终局的结果和目前状态的探索程度。

MCTS算法使用一个常数c来平衡这两者， 在有限的计算步骤或时间内作出最优选择。

AlphaZero 在这里使用多项式型上置信树 (PUCT)来决策下一步的行动。

如果我们给定局面 s 然后考虑 a 行动， 我们需要以下3个值来计算 PUCT(s, a):
* Q — a 行动的平均价值. 当前计算得到的 a 行动的平均终局结果
* P — 神经网络给出的先验概率
* N — 节点计算量(visits), 目前计算中 a 行动的计算次数

然后我们计算 PUCT(s,a) = Q(s,a) + U(s,a)， 其中U(s,a) = c_{puct} * P(s,a) * $\frac{\sqrt{\sum_b N(s,b)}}{1+N(s,a)}$
需要注意的是， 所有可选行动 b 在分子部分， a 行动的计算次数(visits)在分母部分。
我们越少尝试过此行动， 则 U 函数的结果越大。这形成一种启发式的探索模式。

通过增大c_{puct}， 使得探索过程更偏向于陌生的行动， 形成前所未见的局面； 相应的，
减小c_{puct}， 我们更偏向于对已知的局面的探索。AI模型训练对超参数的选择和不断调整，
成为一项难于掌握、 后知后觉且计算成本高昂的事情。开炉炼丹之说， 由此而生。

然而， SAI引擎引入了一种独特的方式， 对征子计算做出了显著的改善。

2 N(s,a) 中的特殊处理

本文略去介绍SAI引擎通过记录连续打吃状态(forced)， 判断局面处于征子状态的设计
在UCTNode.cpp中

```
    void UCTNode::clear_visits() {
     m_forced = 0;
   }
    void UCTNode::update(float eval) {

     if (forced) {
```

```
    m_forced++;
    }
  }
  int UCTNode::get_visits() const {
    return m_visits;
  }
  int UCTNode::get_denom() const {
    if (cfg_laddercode) {
    return 1 + m_visits - m_forced;
    } else {
    return 1 + m_visits;
    }
  }
    const auto denom = child.get_denom();
    const auto puct = cfg_puct * psa * (numerator / denom);
```

SAI引擎通过增加denom[1]函数， 额外给处于征子状态的 a 行动降低visits，
使得AI对征子行动的探索倾向大大提升。

在自对弈训练和正常比赛中， SAI引擎减少了很多征子局面。
通过加载LeelaZ权重测试， 在40K以上的计算量中， LZ可以避开征子局面，
或利用征子局面， 以数量级的计算优势领先LZ官方引擎。

SAI引入征子特征的版本号为： SAI-0.17.5

George（qq 271970125）
2020-01-03

1. denominator 分母(英文)