minimax的算法优化也同样适用于Negamax。Alpha-beta剪枝可以减少negamax算法在搜索树中评估的节点数，其方式与使用minimax算法类似。

使用alpha-beta修剪的深度受限negamax搜索伪代码如下

α和β表示给定树深度下子节点值的下界和上界。Negamax将根节点的参数α和β设置为可能的最低值和最高值。其他搜索算法，如negascout和MTD（f），可以使用交替值初始化α和β，以进一步提高树搜索性能。

当negamax遇到alpha/beta范围之外的子节点值时，negamax搜索将切断，从而从探索中剪除游戏树的部分。基于节点返回值，截断是隐式的。在其初始α和β范围内发现的节点值是节点的精确（或真）值。该值与negamax基本算法返回的结果相同，没有截止值，也没有任何α和β边界。如果节点返回值超出范围，则该值表示上限（If）值≤ α） 或更低（如果值≥ β） 绑定到节点的精确值。Alpha-beta修剪最终会丢弃任何值绑定结果。这些值不会影响根节点上的negamax值。

该伪代码显示了alpha-beta修剪的失效软变化。Fail soft从不将α或β直接作为节点值返回。因此，节点值可能超出通过negamax函数调用设置的初始α和β范围边界。相反，失败硬alpha-beta修剪总是将节点值限制在α和β的范围内。

此实现还显示了在计算子节点的foreach循环之前的可选移动顺序。移动排序[2]是对alpha-beta修剪的优化，尝试猜测产生节点得分的最可能的子节点。该算法首先搜索这些子节点。良好猜测的结果是更早和更频繁地发生alpha/beta截止，从而从搜索树中剪除额外的游戏树分支和剩余的子节点。