

文章编号:1001-9081(2006)02-0462-03

机器博弈游戏在单片机上实现

杨 军, 张 波

(南开大学 软件学院, 天津 300071)

(Jy820314@eyou.com)

摘 要:介绍了博弈程序在单片机环境下的实现方法,讨论了单片机运行速度和空间容量与博弈程序匹配问题的解决方案。通过对程序及数据占用空间的优化以及提高程序执行效率,达到适应单片机的运行环境的目的。结果显示,经过优化的博弈程序在单片机环境下可以达到相当的智能水平。

关键词:单片机;机器博弈;人工智能;搜索技术

中图分类号: TP18 **文献标识码:** A

Implement of adversarial game on the Single Chip Micryoco

YANG Jun, ZHANG Bo

(College of Software, Nankai University, Tianjin 300071, China)

Abstract: An implement of adversarial game on the single chip micryoco was presented, and the solutions about the match between SCM's speed was discussed, space and adversarial program. To adapt the adversarial program to SCM environment, the solution optimized the space that occupied by program and data, made the execution more efficient. The experimental results suggested that the optimized adversarial program under SCM environment can achieve a good intelligence.

Key words: SCM; machine adversarial; artificial intelligence; adversarial search

1 博弈游戏的主要组成部分

1.1 搜索技术

棋盘和棋子的表示主要是指用数据结构表示棋盘上的信息,通常用一个二维数组表示棋盘和棋子的信息。在人机博弈时,机器需要找到一个合理有效的动作,即走步,这种寻找的过程称为搜索,在搜索的过程中需要用数据结构表示整个过程。一般来说,用游戏树表示这个过程。以九格棋为例,游戏树如图 1 所示。

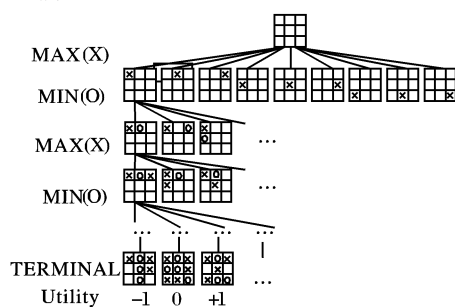


图 1 九格棋的游戏树表示

在机器准备搜索时,当前棋盘的状态用树的根节点表示。在根节点下机器可能有很多可以选择的走步,并由此产生新的棋盘状态,这些棋盘的状态用根节点的子节点表示,然后对于每一个节点,机器的对手也将有很多种选择,依次延伸下去,可以得到如图 1 所示的游戏树。

游戏树中的每一个节点都可以用估值函数对局面进行评估,返回整数,数值越大表示对机器越有利,反之对机器越不利。

搜索技术就是找到这样的一个走步,由它产生的局面对

机器最有利,同时也要考虑到对手走出对机器最不利的一个走步的可能性,从游戏树中看,也就是找到最底层叶子节点所对应局面的估值情况,若最底层处于根节点下的奇数层,则搜索该层的最大值,即找到使得产生的局面对机器最有利的一个走步,若最底层处于根节点下的偶数层,则搜索该层的最小值,即考虑到对手走出对机器最不利的一个走步的可能性。不管是奇数层还是偶数层,两种搜索都将找到的最值返回到上一层,再依次向上递推,当返回到根节点的下一层时,取该层所有局面评估值的最大值所对应的局面为最优的局面,那么机器将选择产生该局面的走步,作为最优的走步。

1.2 可走步的产生

在讨论搜索技术的过程中,机器要搜索所有可能产生的局面,所以不可避免的要涉及到当前局面下可走步的产生。各种机器博弈游戏的规则不同,走法产生的复杂程度也有较大的区别。例如,五子棋的棋盘上的任意空白都是合法的下一步,这样五子棋的走法产生模块里,只要扫描棋盘,寻找到所有的空白,就可以罗列出所有符合规则的下一步;而在中国象棋里,走法的产生就要复杂一些,马走日、象走田等等复杂规则都需要考虑。

1.3 估值函数

先回到图 1 的游戏树,对于九格棋的游戏树所有叶子节点的个数是 $9 \times 8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1 = 362880$,对这种简单的游戏,如果不加任何的搜索优化技术,需要搜索 362880 个节点。对于较复杂的棋,以中国象棋为例,每一个局面可有 20~60 种走法,以平均 40 步走法计,一般双方各走 50 步就可以分出胜负,那么这棵游戏树将有 10^{160} 个节点,这将远远超出计算机的处理能力,所以我们不可能搜索到游戏的结束。为此我们在搜索时需要指定搜索的深度,在指定深

收稿日期:2005-08-29;修订日期:2005-10-31

作者简介:杨军(1982-),男,天津人,硕士研究生,主要研究方向:人工智能博弈算法、语音信号处理;张波(1972-),男,陕西安康人,副教授,主要研究方向:语音信号处理。

度的最后一层是不能分出胜负的,这就需要估值函数,对当前的局面进行评估,根据值的大小表示对哪方有利。

估值函数的设计与特定的游戏规则紧密相关,而且主观性强,很难做到对局面精确地估值,但一般来说,可以通过以下几个方面考虑局面的优劣:

1) 棋子的数量和价值,对于不同的棋子价值肯定有所不同,以中国象棋为例两个马要优于一个马,一个车要优于一个马或一个炮。

2) 棋子的位置,除了棋子的价值外,棋子所在的位置也至关重要,例如处于边角的马与处于中央位置的马,肯定价值不同。

3) 棋子对棋盘的控制能力,加入棋子对整个棋盘上的关键位置都具有控制能力,那么相应的对方的棋子就处于被动局面。

4) 棋子之间的协调,明显地,如果各个棋子的搭配得当,那么会产生很强的攻击力或者很坚稳的防守。

5) 其他特殊的优势(劣势),这部分与具体的游戏有关。

2 具体的实践——中国跳棋的例子

下面用中国跳棋博弈程序在单片机上的实现这个实例来进行具体说明。

首先,选取的单片机芯片是凌阳 SPCE061A,它是一款 16 位结构的微控制器,内嵌 32K 字的闪存 (FLASH),主要的性能参数有: CPU 时钟 0.32MHz ~ 49.152MHz,内置 2K 字 SRAM,32K 字 FLASH。和 PC 机的 CPU 芯片相比,单片机芯片在速度上和存储器容量上都显不足,尤其是在运行较大规模计算和较大数据的存储上,需要设计一种优化时间和空间的解决方案。

其次,中国跳棋是一种规则较简单的博弈游戏,可以有 2 个人到 6 个人同时进行,棋盘为六星型,棋子分为六种颜色,每种颜色 10 个棋子(3 个以下玩家每人可用 15 个棋子),每一位玩家只一个角,拥有一种颜色的棋子,该游戏最大的特点是整个过程中所有棋子都在不停被俘获或移动。

2.1 棋盘和棋子存储结构的修改和优化

出于空间和时间的考虑,跳棋棋盘的表示不能采用传统的二维数组,因为单片机芯片是按字进行存储的,二维数组的表示将占用不少的空间,更糟的是单片机的 SRAM 只有 2K 字,我们不可能将棋盘和棋子的二维数组表示放入 RAM 中,只能放在 FLASH 中,但是 FLASH 的写操作是很消耗时间的。如果采用二维数组表示,那么在每一次棋盘上的变动都会产生 FLASH 写操作,时间消耗很大,另外中国跳棋只有少量的棋子(双人游戏时,只需要 20 个棋子),而且棋子之间是没有差别的。所以如果用一个字的空间表示棋子号,显然有很多空间的浪费。所以设计出一种新的结构,来优化二维数组的存储结构。

棋盘和棋子的表示肯定要放到 FLASH 中的,FLASH 的读操作与 ROM 的读操作的时间差别不大,所以对 FLASH 中棋盘只要求读操作。棋盘上共有 121 个点,首先将所有的点编一个唯一的序号。

新的结构命名为序号/位置映射结构,它就是将二维数组的下标的位置信息用一个结构体表示,该结构体包含行号和列号两部分。由于棋盘是不规则的,所以需要将棋盘规则化,使它能节省大量的空间和方便地使用坐标表示,图 2 所示,将棋子重新排布后存入棋盘点的序号/位置映射表,这是一个结

构体数组,每一个元素包含该位置所在的行号和列号,它的长度是 121。

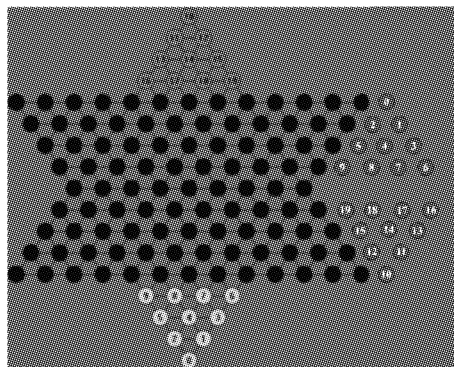


图2 中国跳棋棋盘表示的规则化

由图2可以看出该数组可以定义成 9 行 14 列。简单地说,就是将位置坐标信息放入数组的存储空间,而位置序号信息放入数组的下标,如果需要对棋盘上的棋子位置进行更新,只需要更改放在 RAM 中的棋子号/序号映射表,这样一来,就不需要对放在 FLASH 中的棋盘进行写操作,节省了 FLASH 写操作与 RAM 写操作的时间差。因为博弈程序包含搜索过程,棋盘必然会多次更改,所以节省的时间是很可观的。

进一步,对放在 RAM 中的棋子号/序号映射表可以进行优化。单片机芯片是按字进行存储的,棋子号的空间是 0 ~ 19(双人游戏时,只需要 20 个棋子),序号的空间是 0 ~ 120(棋盘上共有 121 个点),每个字的占 16 位。所以我们可以建立 20 个空间的一维数组,数组的每个元素表示两个序号,用高 8 位和低 8 位分别表示。

综上所述,传统二维数组结构和规则化的双映射表结构空间占用的比较如表 1 所示。

表1 传统二维数组结构和规则化的双映射表结构空间占用的比较

空间占用 (单位:字)	二维数组	双映射结构	
		序号/位置映射	棋子号/序号映射
RAM	0	0	2月20日
FLASH	17 * 25	9 * 14	0

2.2 搜索层次选择

如前所述,我们不可能建立起完整的一棵游戏树,也就是无法一直搜索到游戏的结束,对于中国跳棋来说,平均每一个局面都有 60 ~ 100 种走法,平均双方各走 50 步左右就可分出胜负(双人博弈的情况),那么需要搜索的节点就是 80^{100} 个,需要的时间就是天文数字了。所以我们只能预先指定一个搜索的深度。首先要明确的是,这个深度直接关系到机器的智能性,如果搜索的层次较小那么机器只能对当前局面以后较少的回合进行分析,必然影响到机器的棋力,因此我们要在层次和单片机的处理能力上进行权衡。

表2 层次与运行时间的关系(t:s)

	SPCE061A (49.152MHz, 2K RAM)	PC (2.6GHz, 512M RAM)
二层	3s ± 0.2s	时间极短, < 0.2s
三层	≈ 100s	时间极短, < 0.2s
四层	≈ 3000s	2.6s
五层	很大	7.98s
六层	很大	127s
七层	很大	很大

表 2 先给出一个大致的层次与运行时间的关系(博弈程序是未经任何优化的)。

由表 2 可见,PC 机与单片机的处理能力的差别还是非常明显的,在单片机看来 2 层、3 层和 4 层是可以接受的深度,而 5 层以及 5 层以上就无法实时处理了。再从棋力来说 2 层的棋力有限,只能对付一般的中级棋手,不能满足棋力的要求,所以在深度和处理能力之间权衡的结果是选择 3 层或者 4 层。

2.3 可走步的压缩和转移

表 2 中显示 3 层需要 100 多秒,4 层更需要 3000 多秒,所以要达到实时的效果,至少需要将时间降低到 15 秒以下。

机器博弈者根据当前棋盘上的局面,获得所有机器博弈者能够行进的合法走步,然后对每一种走步进行试探,即产生新的局面,然后机器博弈者再在新局面下产生对手可能会出现的所有走步,并对对手的每一可能的走步产生新的局面,如此反复,当达到系统指定的层次时,停止试探,并对局面进行评估,选择对机器博弈者最有利或者选择对机器博弈者损害最小的一步。注意:在当前局面产生所有新局面时,我们需要保存到达这些新局面的可走步,走步的表示为起始位置序号和终止位置序号,需要 2 个字,大致需求的空间是平均每层的步数 $80 \times 2 \times 4$ (层数) = 640 字,这部分如果放入 FLASH,写操作必然会对性能有很大影响,如果放入 RAM,640 字将占去 2K RAM 的 31.25%。

解决的办法就是将可走步进行压缩,即用一个字表示一个走步,高 8 位表示起始位置序号,低 8 位表示终止位置序号,然后将压缩后的可走步栈放入 RAM,以避免 FLASH 的写操作。表 3 将压缩前和压缩后做一下比较。

表 3 中各项的含义:搜索,整个智能走子的搜索过程所消耗的时间;产生可走步,产生所有可走步所消耗的时间;估值,所有的局面估值所消耗的时间;入栈,所有的 flash 写所消耗的时间;行进一步,所有用于产生新的可能局面的走步所消耗的时间;回退一步,所有用于恢复上一局面的走步所消耗的时间。

表 3 可走步栈压缩前后(flash 写操作对时间的影响)

	压缩前 (走步存储 FLASH 中)		压缩后 (走步存储 RAM 中)	
搜索	9387	100%	3579	100%
可走步	4608	49.0%	1879	52.5%
估值	2298	24.5%	901	25.3%
入栈	1580	16.8%	148	4.1%
行进一步	563	6.0%	295	8.2%
回退一步	441	4.7%	289	8.1%

注:表格内的数字单位是 click,每个 click 是 2048 个时钟周期

由表 3 可以看出经过压缩,放入 RAM 中后,入栈的时间明显减少,这也说明了 FLASH 的写操作比 RAM 的写操作要费时得多。

相应地,单片机上搜索的层数和时间的关系将会改变,如表 4 所示。

表 4 经过可走步栈压缩后的层次与运行时间的关系(t:s)

	SPCE061A (49.152MHz, 2K RAM)	PC (2.6GHz, 512M RAM)
三层	27	时间极短, 小于 0.2
四层	600	2.6

由表 4 可见,仍然不能做到实时的产生智能的走步,于是需要进一步优化。

2.4 优化方向的选择

表 3 显示出产生可走步的时间占了整个时间一半多,另外局面评估的时间也占整个时间的四分之一。

因为中国跳棋的规则比较简单,所以单从可走步产生的算法上很难找出改进点;同样地,规则简单也会使得估值函数的设计也不那么复杂,对跳棋来说,只需将己方棋子所在棋盘位置的权重累加起来并减去对手棋子所在棋盘位置的权重即可,这已经是最简单的估值函数设计了。

那么缩减时间的可能性就集中在搜索算法本身的优化上了。目前的对搜索算法的优化主要有置换表(transposition tables)、驳斥表(refutation tables)、窄窗搜索(aspiration search)和最小窗口搜索(minimal window)、启发式搜索(heuristic)……这些优化都是从空间上牺牲换取时间上的改善,从前面的分析看,这不适用于空间有限的单片机。

根据中国跳棋的特殊性,可以有下面的判断,在搜索每一层的所有可走步的时候,每一层的所有可走步中将会有相当一部分的回退步,很明显,中国跳棋中棋子向回退的走步是对己方不利的,也就是说搜索函数无需搜索这种向回退的走步所产生的新局面,从而节省时间。

2.5 基于知识的优化

由上面的经验可以想到,对于二人博弈游戏,产生的所有可走步是所有合法的可走步,而不是合理的可走步。所以如果对当前层次的所有局面的进行估值并排序,就会从两方面提高性能:增加剪枝的可能性;如果将当前层最差的若干步删除,将直接减少搜索的时间。与此同时,产生可走步部分的时间会因为最差步的删除,新局面的减少而减少,估值函数的调用也会因为新局面的减少而减少。如此获得的性能改善将是明显的,如表 5 所示。

表 5 经过排序优化和智能选择优化后的关系(t:s)

优化方法	目前的优化	加入排序	基于知识的优化			
			删除最差的 10 步	删除最差的 20 步	删除最差的 30 步	删除最差的 40 步
三层	27	15.58	18.32	15.57	12.85	10.21
四层	600	370	420	380	320	250

3 结语

在空间和处理速度都有限的单片机上实现博弈游戏程序,需要考虑游戏的表示所占用的空间;搜索过程占用的空间;搜索层数与单片机的计算量的匹配;搜索时间的优化从而达到游戏的实时性要求。

参考文献:

- [1] NILSSON NJ. Artificial Intelligence[M]. Beijing: China Machine Press, 1999. 195-213.
- [2] GILLOGLY J. Performance analysis of the technology chess program [D]. Dep. Comput. Sci., Carnegie-Mellon Univ., 1978.
- [3] SCHAEFFER J. The History and Alpha-Beta Search Enhancements in Practice[J]. IEEE: Pattern Analysis and Machine Intelligence, 1989, 11: 1203-1212.
- [4] 廖家平, 舒军, 王粟. 基于 PDC-PROLOG 自学习机器博弈[J]. 湖北工学院学报, 1997, 12(4): 8-12.
- [5] 田胜丰, 黄厚宽. 人工智能与知识工程[M]. 北京: 中国铁道出版社, 1999. 64-103.
- [6] 李兵. 搜索算法的改进及其在国际象棋中的应用[A]. 全国人工智能机器应用学术会议论文集[C]. 1989. 364-371.
- [7] 李晶皎. 嵌入式语音技术及凌阳 16 位单片机应用[M]. 北京: 北京航空航天大学出版社, 2003. 120-201.
- [8] SAHNI S. 数据结构算法与应用[M]. 北京: 机械工业出版社, 2000. 248-363.