数码谜题求解的算法设计及其扩展研究

冯晓辉,马光思
(西安建筑科技大学 信息与控制工程学院,陕西 西安 710055)

摘 要:数码谜题是人工智能领域中的经典问题。结合低阶数码谜题求解的具体实现过程,分析了求解高阶数码谜题的存储机制设计、可解性判断、移动规则设计及搜索策略。与传统搜索算法相比,A*算法具有求解快、占用内存小的特点,研究了该算法框架下的启发函数改进及向高阶数码谜题求解的扩展。实验结果证明了该算法的有效性。

关键词:数码谜题;可解性判断;A*算法;启发式搜索
中图分类号:TP18 **文献标识码**:A **文章编号**:1673-629X(2009)08-0110-03

Algorithm Design and Extension Research of N-Puzzle Problem

FENG Xiao-hui, MA Guang-si
(School of Info. & Control Eng., Xi'an Univ. of Architecture & Tech., Xi'an 710055, China)

Abstract: N-puzzle is a classic problem in artificial intelligence. Combining with the implementation process of solving 8-puzzle, analyzed the designing of the storage mechanism, judging of the solvability, designing of moving rules, and search strategy of solving n-puzzle problem. Compared with traditional search algorithm, A* algorithm has faster speed and less memory usage. Based on the framework of A* algorithm, researched the improvement on the heuristic function and the extension toward solving the n-puzzle. The experiment results have illustrated the effectiveness of this algorithm.

Key words: n-puzzle; solvability; A* algorithm; heuristic search

0 引 言

8数码谜题又称重排九宫,参见图1。问题描述为:8个用数字1~8标注的棋子摆放在一个3×3共9个格子的棋盘上,空出一个格子使棋子能在盘内水平滑动,以形成不同的棋局。求从任一初始棋局变化到另一目标棋局是否有解,以及有解时的解法。其初始、目标棋局皆可人为构造或随机生成。

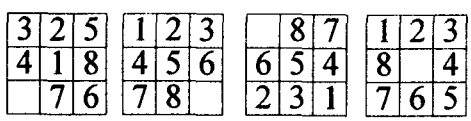

图1 8数码谜题

关于8数码谜题已有不少文献讨论过,且有确定的结论。文献[1]从可解性和算法等方面探讨了问题的表示和求解;文献[2,3]分别使用C++与Java语言求解问题;文献[4]分别使用深度、广度优先搜索与A*算法解题,分析比较了三者的性能和应用范围;文献[5]比较了8数码中三种不同启发函数的搜索效率,并指出选择最佳启发函数的原则。文中结合笔者实现数码重排的实验,开展对相关问题的理论分析及算法设计,其中包括一些扩展性讨论。

1 存储机制设计

根据图1,8数码谜题的棋局显然可用矩阵或二维数组存储。按照对应关系,棋局也能以向量或一维数组形式存放。

(1) 按从左向右、从上到下的顺序,任意棋局可惟一对应一序列 $P = a_1a_2a_3a_4a_5a_6a_7a_8a_9$。其中 $a_i(i = 1 \sim 9)$ 为0,1,…,8九个数中的一个,0表示空格,称序列 P 为棋局对应的状态序列。该序列可采用一维数组 $S[1 \sim n \times n]$ 存放,例如,图1中初始棋局1可存储为:

$S(1) \leftarrow 3, S(2) \leftarrow 2, S(3) \leftarrow 5, S(4) \leftarrow 4, S(5) \leftarrow 1, S(6) \leftarrow 8, S(7) \leftarrow 0, S(8) \leftarrow 7, S(9) \leftarrow 6$

(2) 棋局还能以 S 型顺序来存放。即对第一行元素按从左到右的顺序依次存放,然后对第二行以从右向左顺序存放,第三行又与第一行存放方式相同,依此类推。

收稿日期:2008-11-25;**修回日期**:2009-02-09
基金项目:陕西省教育专项科研基金(07JK306)
作者简介:冯晓辉(1984-),男,陕西西安人,硕士研究生,研究领域为人工智能;马光思,教授,研究领域为计算机软件与理论、信息安全。

与大多数讨论相同,文中采用(1) 中存储方式。

2　可解性判断

由于重排问题中有些初始棋局无法变化到目标棋局,因此求解前必须先讨论其可解性。对于 8 数码,可考察从初局到终局能否偶数次交换棋局的任意两个数,若能,则问题有解;否则无解。用代数学术语可描述为考察从初局到终局所作置换是否为偶置换,若是,则问题有解[6]。具体处理过程是通过计算棋局状态序列的逆序数来判断的。

2.1　序列的逆序数

在序列 P 中对 $i < j$,若有 $a_i > a_j$,则称 a_j 出现了一个逆序,元素 a_j 的逆序数记为 $R(a_j)$[6]。于是序列 P 的逆序数记为 $R(P)$,定义为除元素 0 外所有元素的逆序个数之和。序列 P 的元素存放在数组 $S[1 \sim n \times n]$ 中,则求其逆序数的算法如下:

Step1 $t = 0$;
Step2 $i = 1$;
Step3 当 $i \leqslant n \times n$ 时反复执行
　Step3.1 If $S[i] \neq 0$ then
　　对 $j = 1$ 到 $i - 1$ 步长取 1
　　If $S[j] > S[i]$ then $t = t + 1$
　Step3.2 $i = i + 1$
Step4 $R(P) = t$

2.2　可解性判断

对于任意 n 阶棋局,判断可解性需要考虑 n 的奇偶性。

引理 1　若 n 为奇数,只要按规则移动,则状态序列逆序数的奇偶性不变。

证明:因元素 0 对序列求逆序数没有影响,为了方便证明,状态序列中可不考虑 0。当数码左右移动时,状态序列没有变化,故其逆序数奇偶性不变;考察数码上下移动。当数码 a_i 下移时,相当于数码往后挪了 n 个位置,序列从 $a_1a_2\cdots a_ia_{i+1}a_{i+2}\cdots a_{i+n-1}\cdots a_{n^2-1}$ 变为 $a_1a_2\cdots a_{i+1}a_{i+2}\cdots a_{i+n-1}a_i\cdots a_{n^2-1}$,即状态序列中 a_i 与 a_{i+1},a_i 与 a_{i+2},$\cdots$,a_i 与 a_{i+n-1},依次进行了 $n-1$ 次相邻对换。因为序列经过偶数次邻换后,其逆序数的奇偶性不变[1],所以问题得证。同理可证数码上移。引理 1 得证。

引理 2　若 n 为偶数,只要按规则移动,则状态序列的逆序数加空格所在行数的奇偶性不变。

证明:情况 1:当数码左右移动时,状态序列没有变化,故其逆序数奇偶性不变。因移动前后空格均在同一行,显见状态序列的逆序数加空格所在行数后,奇偶性不变。情况 2:考察数码上下移动。当数码 a_i 下移时,也意味着位于棋盘第 m 行的空格上移到第 $m-1$ 行,设原序列为 P_0,移动后的序列为 P_1。与引理 1 证明类似,数码下移时状态序列中 a_i 与 a_{i+1},a_i 与 a_{i+2},$\cdots$,a_i 与 a_{i+n-1},依次进行了 $n-1$ 次相邻对换。由于假设前提中 n 为偶数,故 $n-1$ 为奇数。因此这一下移变换改变了原序列逆序数的奇偶性,用公式可表示为 $(-1)^{R(P_0)} = (-1)^{R(P_1)\pm 1}$。此时,若把空格所在的行数考虑进去,即对原序列 P_0 的逆序数加 m,对移动后新序列 P_1 的逆序数加 $m-1$,因 m 与 $m-1$ 奇偶性相反,可得 $(-1)^{R(P_0)+m} = (-1)^{R(P_1)+m-1}$。可见数码下移,状态序列逆序数加上空格所在行数后的奇偶性保持不变。同理可证数码上移。综合情况 1 与情况 2,引理 2 得证。

n 阶数码重排问题中,任意初始状态可到达的状态数为 $n^2!/2$ 个[7]。这样恰把棋局状态空间分为两个等价类。亦即,只有等价类内的状态彼此可达。

综上分析,8 数码谜题扩展到任意 n 阶时,可解性判断定理如下:

(1)对奇数阶棋盘,只有当初始状态序列与目标状态序列二者的逆序数奇偶性相同时问题有解。

(2)对偶数阶棋盘,只有当初始状态序列与目标状态序列二者各自的逆序数加其空格所在行数所得结果的奇偶性相同时问题有解。

3　移动规则及相应操作

设计算法求解数码问题前,必须预先给定移动规则及相应的操作。数码的上、下、左、右移动都是围绕着棋盘空格进行的,即棋盘空格与相邻的数码互换位置。

用 j 表示上述一维数组中元素 0 的下标,则 $S[j]$ 就表示棋盘的空格。对 8 数码而言,以 $1 \leqslant j \leqslant 6$ 作为上移限制规则,即当且仅当空格位于 8 数码上部两行中时,才允许其下方的数码上移。扩展到 15 数码时,以 $1 \leqslant j \leqslant 12$ 作为上移前提,即当且仅当空格处在 15 数码上部三行中时,才允许其下方的数码上移。

同理,8 数码的下移、左移、右移规则分别为 $4 \leqslant j \leqslant 9$、$j \bmod 3 \neq 0$、$j \bmod 3 \neq 1$;扩展到 15 数码的相应规则分别为 $5 \leqslant j \leqslant 16$、$j \bmod 4 \neq 0$、$j \bmod 4 \neq 1$。

综上可构造出 n 阶棋盘的移动规则,以及满足某种(上、下、左、右) 移动规则时执行的相应操作:

上移操作:if $1 \leqslant j \leqslant n \times (n-1)$, then $S[j] = S[j+n]$, $S[j+n] =$ '0';

下移操作:if $n+1 \leqslant j \leqslant (n \times n)$, then $S[j] = S[j-n]$, $S[j-n] =$ '0';

左移操作:if $j \bmod n \neq 0$, then $S[j] = S[j+1]$,

$S[j+1]$ = '0';

右移操作:if $j \bmod n \neq 1$, then $S[j] = S[j-1]$, $S[j-1]$ = '0';

例如'上移操作',若条件满足,先将表示上移数码的数组元素 $S[j+n]$ 赋予 $S[j]$(空格),让后者表示上移数码;再给 $S[j+n]$ 赋值 0,使其表示空格,就此完成数码上移。

4 搜索策略

数码重排通常采用状态空间搜索作为求解方法,但广度和深度优先搜索都是盲目展开子结点,这样就可能出现指数级的时间复杂性,所以最好使用启发式搜索求解。

4.1 启发式搜索

启发式搜索就是在搜索时通过一个估价函数来对每个结点进行评估,选择代价最少、重要性较高的作为下一步的搜索结点,然后跳转其上(若有一个以上代价最少结点,通常选择距离当前搜索点最近一次展开的结点进行下一步搜索)。

启发式搜索算法通常由两部分组成:估价函数和使用该函数搜索状态空间的算法。估价函数定义为从初始结点经过 n 结点到达目标结点的最小代价路径的代价估计值。它的一般形式是 $f(n) = g(n) + h(n)$,其中 $g(n)$ 是从初始结点到 n 结点的实际代价,$h(n)$ 是从 n 结点到目标结点的最优估计代价,主要是 $h(n)$ 体现了搜索的启发信息[8]。

4.2 算法估价函数

数码重排最简单的启发函数 $h(n)$(见图 2)就是计算当前状态与目标状态相比位置不符的数码数目。该方法虽然计算量较小,但其结点扩展量大,搜索效率低。

所以将 $g(n)$ 定义为从初始结点到 n 结点移动的数码数目,即搜索树中 n 结点的深度;$h(n)$ 定义为在 n 结点状态下各数码到达目标状态的距离之和,即 $h(n) = \sum(|x_{i1} - x_{i2}| + |y_{i1} - y_{i2}|)$,$x_{i1}$、$x_{i2}$ 分别为数码 i 在当前状态和目标状态下的横坐标,同理 y 为纵坐标。

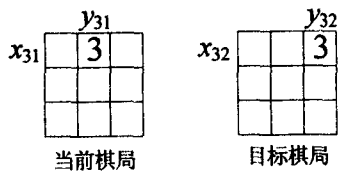

图 2 启发函数 h(n)

4.3 A* 算法框架

A* 算法是指满足条件 $h(n) \leqslant h^*(n)$ 的启发式搜索,$h^*(n)$ 为从 n 结点到目标结点的实际最优代价,多数情况下 $h^*(n)$ 无法计算,但要判定 $h(n)$ 是否大于 $h^*(n)$ 是可能的。

A* 算法采用两个称为 Open 表和 Closed 表的动态数据结构,前者记录当前未考察的结点,后者记录考察过的结点。扩展所得子结点按其估值的大小插入 Open 表中合适的位置,每次从中选出估值最小的加以扩展。每个结点都保留父结点的信息,这保证了搜索完毕后,能通过 Closed 表返回完整的搜索路径[8]。求解数码重排的算法框架如下:

Step1 判断问题可解性;若无解,则退出。

Step2 初始化,将初始结点放入 Open 表,令 Closed 表为空。

Step3 若 Open 表不为空,循环执行以下操作:

Step3.1 选取 Open 表的表头作为当前结点;若该结点是目标结点,跳转至 Step4。

Step3.2 将当前结点从 Open 表中移除,放入 Closed 表中。

Step3.3 求出当前结点的所有子结点。

Step3.4 计算子结点的估价值。

Step3.5 如果子结点不在 Open 表和 Closed 表中,将其插入到 Open 表。

Step3.6 按照结点的估价值,以递增顺序重排 Open 表。

Step4 回溯得解。

如图 3 所示(字母表示结点,数字表示该结点的估价值):假设初始结点 A 产生三个子结点 B、C、D,则 Open 表与 Closed 表的当前状态为 Open = [B4, C4, D6], Closed = [A5]。然后选择 Open 表中估值最小的结点 B 扩展后, Open = [C4, E5, F5, D6], Closed = [B4, A5]。但启发信息也可能出错,故算法不丢弃其它状态而是将其保留在 Open 表。当某一启发信息将搜索导向错误路径时,可从 Open 表中选取先前生成的"次优"状态进行搜索。此例中结点 B 扩展所得结点 E、F 的启发性较差,算法自动纠错,将搜索转移到 C 结点。

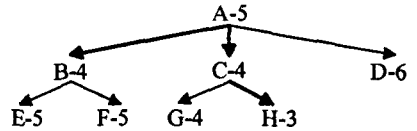

图 3 算法搜索示意

5 结束语

最差情况下 A* 算法的时间复杂度为 $O(b^m)$,其中 b 为搜索树的最大分枝因子,m 为状态空间的最大

(下转第 116 页)

VSM 进行测试,最后得到不同的阈值下传统模型与改进模型的查全率、查准率及综合评估率 F 的结果(见表 3)。

通过实验结果可知,通过改进索引项权重和相似度的计算方法,使基于改进模型的查全率和查准率得到一定的改善。

表 2 从本系统收集的用户问句中抽取了 387 条问句(问句未加改动)

编号	问 句
1	什么溢出错误!
2	frx,scc 文件怎么来的,有什么用啊?
3	怎么做才能让文本框字以彩色显示!?
⋮	⋮
317	图像控件部分也要考吗?

表 3 用户问句与 FAQ 库中问题相似度测试实验结果

阈值	查全率(%)		查准率(%)		F(%)	
	查全率传统	查全率改进	查准率传统	查准率改进	F 传统	F 改进
0.1	68.13	77.42	66.34	74.65	67.22	76.01
0.2	54.12	61.24	72.76	86.34	64.79	73.18
0.3	38.93	45.54	89.56	94.77	56.64	65.17
0.4	22.89	31.74	96.87	97.39	37.03	47.88
0.5	14.65	22.67	98.37	100.00	25.92	34.52
0.6	8.13	14.27	100.00	100.00	15.78	24.56

5 结束语

文中提出的基于改进的向量空间模型的权重和辅助关键词距离和次序的句子相似度算法,在特定领域文档查询中的查全率、查准率方面有明显的改进。本检索模型是针对特定领域,对分词词典有特别要求:针对特定领域的专业词典;带词性标注的普通分词词典。由专业词典标出专业关键词,由普通分词词典分出非专业关键词(名、动、形和限定性副词)和一般的普通词。

另外,在文中提出的索引项权重基础上,也可以构造包含同义、词义信息[7,8]或领域本体概念[9]的索引项权重函数,以进一步提高 VSM 的性能。

参考文献:

[1] Salton G, Wong A. On the Specification of Term Value in Automatic Indexing[J]. Journal of Documentation, 1973, 29(4): 351 - 372.

[2] Salton G. The SMART Retrieval System - Experiments in Automatic Document Processing[M]. Englewood Cliffs, NJ: Prentice Hall Inc, 1971.

[3] 王 宇,战学刚,蔡建山. 基于网络的中文问答系统的研究[J]. 计算机工程与应用, 2006(7):162 - 165.

[4] 闫宏飞,陈羽中. 词汇与中心词的距离信息对问句相似度匹配的影响[J]. 清华大学学报:自然科学版, 2005, 45(S1):1873 - 1877.

[5] 苗夺谦,卫志华. 中文文本信息处理的原理与应用[M]. 北京:清华大学出版社, 2007.

[6] 庞剑锋,卜东波,白 硕. 基于向量空间模型的文本自动分类系统的研究与实现[J]. 计算机应用研究, 2001(9):23 - 26.

[7] 董振东,董 强. 知网[EB/OL]. 2007 - 12 - 10. http://www.keenage.com.

[8] 梅家驹. 同义词词林[M]. 上海:上海辞书出版社, 1983.

[9] 朱礼军,淘 兰,刘 惠. 领域本体中的概念相似度计算[J]. 华南理工大学学报:自然科学版, 2004, 32(增刊):147 - 150.

(上接第 112 页)

深度。为了存储产生的结点,此时相应算法空间复杂度也为 $O(b^m)$[8]。随着问题变复杂,其状态空间规模也随之增大,8 数码约为 1.3×10^6,15 数码为 10^{13},24 数码为 10^{25}。因此搜索所需时间、空间都急剧增加,这也解释了一般在实验中 15、24 数码求解为何特别耗时。笔者目前正基于多线程和多核技术在作相关理论研究和实验测试,期望能通过并行处理得到好的结果。

参考文献:

[1] 黄沛杰. 重排九宫问题的分析与实现[J]. 现代计算机, 2003, 12:74 - 77.

[2] 朱永红,张燕平. 用 VC++ 实现基于 A* 算法的八数码问题[J]. 计算机技术与发展, 2006, 16(9):32 - 34.

[3] 方贤进. 建立状态图启发式搜索的面向对象模型[J]. 沈阳工业大学学报, 2003, 25(4):334 - 337.

[4] 詹志辉,胡晓敏,张 军. 通过八数码问题比较搜索算法的性能[J]. 计算机工程与设计, 2007, 28(11):2505 - 2508.

[5] 许精明. 智能搜索中启发函数的选择及启发能力分析[J]. 昆明理工大学学报, 2007, 32(5):31 - 34.

[6] 马光思. 组合数学[M]. 西安:西安电子科技大学出版社, 2002.

[7] 宋 文,伊良忠,牟行军. 15 - 谜问题的可达性判定[J]. 电子科技大学学报, 2004, 33(5):604 - 607.

[8] 陈世福,陈兆乾. 人工智能与知识工程[M]. 南京:南京大学出版社, 1997.