基于树的多因素平衡组卷模型及算法设计①

Multiple Factors Based on Tree of Test Paper ConstructionModel and Algorithm Design

黄 河 （温州职业技术学院 计算机系 浙江温州 325035）

摘 要： 高智能自动组卷算法要求找到最大程度地满足多重约束条件的试题组合。本文提出一种高智能自动组卷的算法模型。它利用树做为基本数据结构，充分考虑了多重组件因素之间的平衡，并实现了算法和数据存储的分离。此算法模型满足多种约束因素、多种策略选择的组卷选择，并提供了高可扩展性。

关键词： 组卷 树 模型 策略 可扩展 算法框架

按照一定的组卷策略从建成的试题库中抽取试题，组成符合要求的试卷，是实现考试规范化和科学化的重要手段，在各种计算机辅助考试系统的软件开发中，高智能自动组卷一直是一个重要的研究课题。

自动组卷算法要求在给定一种组卷策略的条件下，求解出最大程度满足多重约束条件的试题组合。由于组卷策略的复杂性，包含了题型、知识点分布、难度区间、离散度、信度、有效度等多种指标。所以，算法的难度在于如何从多重约束中取得平衡，本质是解决多重约束的组合优化问题。

在国内外组卷算法实现上，常见的有随机组卷法、回溯试探组卷法和遗传组卷法等。但各种算法都存在这样或那样的缺点，比如随机组卷法只考虑随机性，无法体现各种组卷参数对结果的影响。回溯试探法和遗传算法虽然考虑参数平衡但是效率低下，且不易扩展。

本文提出的基于树的多因素平衡组卷算法首先将试题库建构为一棵树；再依据组卷策略，进行归类、分组、抽题、复查、调整，形成试卷。采用此模型能适应多种约束因素、多种策略选择。并在多种因素中依据策略要求取得平衡，最大限度的满足组卷要求。

同时，本文提出一种灵活可扩展的算法框架。在此算法框架下，不仅组卷约束因素可以参数化定制（比如区分度、难度值、题量等）。并且实现策略算法与数据存储分离，策略算法的变化不影响数据存储，同样数据存储的变化也无需修改策略算法的实现。大大提高了可扩展性。易于把算法模型运用在各种不同系统实现中。

1 模型设计

1.1 题库设计及基本属性

在本模型中，将题库设计为模块（Module）、考核块（CheckBlock）、考核面（CheckSurface）、考核点（CheckSpot）、考题（Subject）的五级的严格父子结构。各个级别可以扩展出自定义的组卷约束因素属性已用于定义不同的组卷策略。

以下的讨论基于一个实用的组卷策略。

此策略要求为：在题库中抽取出给定模块的、给定数量的题目形成一份试卷。要保证有一定的覆盖面（即离散性），决定考核点层抽题数量时受重要度的约束，并且整份试卷要达到或接近一定的难度系数比。即本策略靠考虑在覆盖度、重要度和难度这三个约束条件中取得平衡。

所以，在考核点上定义重要度（Importance）属性，此属性决定该考核点被抽到题目的概率的大小。此值越大，此考核点下被抽到的题目的概率越大。

在考题上定义难度（Difficulty）属性，在选择题目时要使得整份试卷尽量接近一定的难度系数比。（如要求难度为 0.0 ~ 03 的题目要抽 20%，难度为 0.3 ~ 0.7

① 基金项目：温州职业技术学院院级重点科研项目（wzy2007003）

的题目要抽 50%，难度为 0.7 ~1.0 的题目要抽 30%；如果试卷有 40 道题目则难度为 0.0 ~0.3，0.3 ~0.7，0.7 ~1.0 的题目抽取的数量分别为 8，20，12）。

1.2 题库树设计

依据以上分析，将题库建构为一棵树。其根节点为模块，第一层子节点为考核块，第二层节点为考核面，第三层节点为考核点，叶子节点为考题。如图 1 所示：

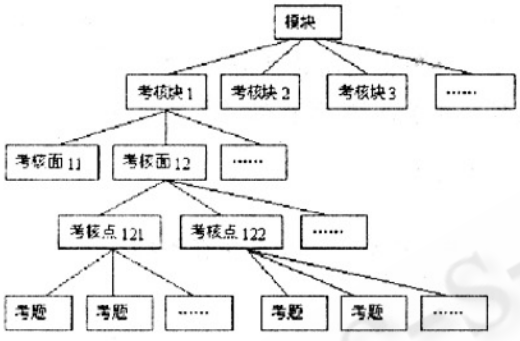

图 1　题库树结构

用以下可变记录类型（TNodeData）来表示节点：

```
TNodeType= (ntBlock,ntSurface,ntSpot,ntSubject,ntUnknown);
PTNodeData = ^TNodeData;
TNodeData = packed record
  case TNodeType of
    ntBlock,ntSurface,ntSpot,ntSubject: (
      Code: ShortString;
      case TNodeType of
        ntBlock,ntSurface,ntSpot: (
          TotalCount: Integer;
          SelectedCount: Integer;
          case TNodeType of
            ntSpot:(Importance: Integer));
        ntSubject: (
          Selected: Boolean;
          Difficulty: Integer));
    ntUnknown: ();
end;
```

1.3 组卷模型设计

由于不同的科目、不同考试，所要求的组卷策略都不同。所以要求设计的组卷模型要足够的灵活。因此在本设计中，采用两个原则来达到此目的：一是将数据存储与组卷算法的进行分离。二是题库各个级别的属性可扩展。图 2 是设计的类关系图.

类 TTacticTree 只用来存储数据，它不对数据做任何操作。若扩展了题库中的属性，可以从此类继承子类即可。

类 TSubjectStrategy 对算法进行封装，它是每个算法的基类，提供最一般的组卷行为 DoStrategy( )。要实现自定义的组卷算法，从其派生的子类即可，但需对 DoStrategy( ) 加以实现，以实行各自的组卷策略。

类 TMiracleTactic 只是一个外壳，它聚合了 TTacticTree 和 TSubjectStrategy 两个对象。它有两个接口 Iinit( ) 和 DoTactic( )，调用 Init( )，初始化有关数据，创建一题库树；调用 DoTactic( ) 来执行策略算法，它调用了 TSubjectStrategy 的 DoStrategy( ) 方法。

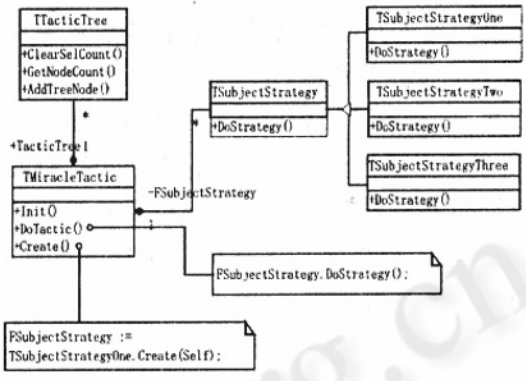

图 2　类　图

将来，如果算法改变，只需从 TSubjectStrategy 派生一个子类，对 DoStrategy( ) 方法加以实现，再在 TMiracleTacTic 的 Create( ) 构造函数中，创建该类的对象即可。

如果数据存储改变（题库属性变化），只需从 TtacticTree 派生一个子类，实行构造树的各个方法。再在 TMiracleTacTic 的 Create( ) 构造函数中，创建该类的对象即可。

2 算法设计

2.1 题库树的生成

在抽题之前根据题库中的信息，把题库中指定模块的题目生成一棵树（TacticTree）。该树有 5 个层次，为根节点、考核块节点、考核面节点、考核点节点、试题节点。

每个节点的 Data 都指向一片数据区域，该数据区

域是 TNodeData 类型的可变记录，它随节点层次的不同而不同。具体如下：

1）根节点存放的是所有考题数量信息和抽题数量信息。

2）块节点存放的是该考核块下所有考题数量、抽题数量以及考核块代码等信息。

3）面节点存放的是该考核面下所有考题数量、抽题数量以及考核面代码等信息。

4）点节点存放的是该考核点下所有考题数量、抽题数量、考核点代码以及重要度等信息。

5）试题节点存放的是试题代码、难度以及是否被选中等信息。

2.2 组卷算法

1）考核块 CheckBlock 的抽题数量

考核块的抽题数量是由该考核块所拥有的考题数量占给定模块下所有试题数量的比值决定的。如，我们被要求生成 Window98 的试卷，抽 40 道题目，题库中总共有 100 道 Windows98 的题目，分 5 个块核块 BlockA、BlockB、BlockC、BlockD、BlockE，它们分别拥有的题目数量为 30、20、15、8、27。那么，

BlcokA 的抽题数量为 Trunc(40 * 30/100) = 12

BlcokB 的抽题数量为 Trunc(40 * 20/100) = 8

BlcokC 的抽题数量为 Trunc(40 * 15/100) = 6

BlcokD 的抽题数量为 Trunc(40 * 8/100) = 3

BlcokE 的抽题数量为 Trunc(40 * 27/100) = 10

总计为 39 题，还差 40 − 39 = 1 道题，在算法的具体实现中差额的题量被随机分配到这 5 个考核块中。

2）考核面 CheckSurface 的抽题数量

考核面的抽题数量的确定与考核块抽题数量的确定类似，由该考核面下拥有的考题数量占所属的考核块的试题数量的比值决定，再由这个比值乘以所属的考核块的抽题数量得到该考核面的抽题数量。具体的做法同上，不再累赘。

3）考核点 CheckSpot 的抽题数量

考核点有重要度属性。把所属考核面下所有的考核点按重要度分组，然后按由重要度决定的概率将所属考核面的抽题数量分配到各组中。

首先定义一个从一个重要度组里面随机抽一道题的过程。若考核点的重要度为 10、20、30，则把该所属考核面下的所有考核点分为 3 组，按从每组里面抽得一道题分别为 1/6，2/6 和 3/6 的概率调用该过程，直到所属考核面下的抽题数量全被分配完为止。

4）题目 Subject 的抽取

Subject 有难度属性。首先对每个试题节点按难度分组，取名为 DiffGroup。根据要求，抽题算法生成的试卷不同难度的题目要符合一定的比例。如难度为 0.0 ~ 0.3 的题目要抽 40 × 20% = 8 道题，难度为 0.3 ~ 0.7 的题目要抽 40 × 50% = 20 道题，难度为 0.7 ~ 1.0 的题目要抽 40 × 30% = 12 道题。根据这个信息，对每个 DiffGroup 进行抽题，当抽够了所要求的题目个数或该难度下的题目没有时退出。

当对每个 DiffGroup 抽题后，很有可能还未抽满 40 个，因为供选择的 Spot 中某一难度的题目数量没有到达要求。漏抽的题目随机分配到有剩余可抽的 Spot 中。

3 试验及分析

为了验证本算法框架的正确性，在题库中加入随机题目信息，但作出有如下前提：

- 各个 CheckSurface 下都有题目。
- 重要度属性各个考核点随机取 10，20，30。
- 难度属性各个题目在 0.0 − 1.0 之间随机取值。
- 组卷题数是 40，难度要求为 0.0 ~ 0.3：0.3 ~ 0.7：0.7 ~ 1.0 = 2：5：3。

表 1 是试验结果。

表 1 试验结果

题库量	组卷份数	满足覆盖度要求份数	平均重要度符合率	难度平均值（0 ~ 0.3：0.3 ~ 07：0.7 ~ 1.0）
80	20	20	80%	2.7：4.1：3.2
160	20	20	92%	2.1：4.8：3.1
320	20	20	100%	2：5：3
640	20	20	100%	2：5：3

我们可以对结果进行分析：

- 由于本组卷优先考虑的覆盖度，而且题库中题目是均匀分布的（随机）。故覆盖度将得到满足。

（下转第 26 页）

(上接第 29 页)

- 难度随着题库量的增大,会越来越接近难度要求,而且当题目量够大,将始终满足要求。原因在于,在题量少的情况下,对不满足难度要求要进行随机分配,影响了难度平均值。
- 重要度随者题库量的增大而接近要求,而当题目量够大,将始终满足要求。原因在于,在题量少的情况下,不满足难度要求的题目要随机再分配,造成重要度分布发生变化。随着难度要求得到满足,题目不再随机分配,重要度也将符合要求。

4 结论

本文提出的组卷模型以及算法,充分考虑了组卷问题的复杂性,利用树结构构造题库树,使用可定义组卷策略在题库树上选择题目进行组卷。通过试验,在利用默认的组卷策略组卷时,能较好的达到组卷参数要求,随着题库中题量的增加,组卷参数的符合度越好。

并且此模型实现了算法和数据存储相分离,可以方便的派生出自定义组卷策略并应用与不同的题库数据结构。具有较好可重用性和可扩展性。

参考文献

1 乔兹德克(Drozdek,A.) 数据结构与算法 - - C + + 版[M] 郑岩,战晓苏. 北京:清华大学出版社,2006.

2 伽玛. 设计模式:可复用面向对象软件的基础[M] 李英军. 北京: 机械工业出版社,2000.

3 徐士良. 常用算法程序集(C 语言描述)(第三版)[M]. 北京:清华大学出版社,2004.

4 边肇祺. 模式识别(第二版)[M]. 北京:清华大学出版社,2000.

5 DELPHI 7.0 英文版 帮助文档 [CP/DK]