# 一种基于意图的设计模式排序与优化方法

关 慧[1,2],历子谦[1],吕 颖[1]
(1.沈阳化工大学 计算机科学与技术学院,辽宁 沈阳 110028;
2.辽宁省化工过程工业智能化技术重点实验室,辽宁 沈阳 110142)

**摘 要**:设计模式是对于特定软件设计问题的经过验证的解决方案,通常用来获取为解决软件设计问题所需的软件知识,但选出合适的设计模式却并非一件容易的事。设计模式意图是理解设计模式的最短路径,因此通过解析设计模式意图对设计模式排序,更利于用户获得所需设计模式。文中提出了一种基于意图的设计模式排序方法,以方便设计模式的选择,定义了相关相似度的计算方法并在文中给出相应的证明,而后通过遗传算法对所定义的参数进行了调整和优化,最后用软件实际开发中实际问题数据集和设计模式集合,在提出的方法上进行了验证和分析。实验结果表明,该方法相比其他方法在匹配率上有一定提高,且排序后使得更多正确的结果出现在靠前的位置。

**关键词**:WordNet;Stanford Parser;遗传算法;设计模式;排序
**中图分类号**:TP311 **文献标识码**:A **文章编号**:1673-629X(2021)08-0019-07
doi:10.3969/j.issn.1673-629X.2021.08.004

# A Design Pattern Ranking and Optimization Method Based on Intent

GUAN Hui[1,2],LI Zi-qian[1],LYU Ying[1]
(1. School of Computer Science and Technology,Shenyang University of Chemical Technology,
Shenyang 110028,China;
2. Liaoning Province Key Laboratory of Intelligent Technology of Chemical Process Industry,
Shenyang 110142,China)

**Abstract**:Design patterns are proven solutions to specific software design problems and often used to acquire the software knowledge needed to solve software design problems,but choosing the appropriate design patterns is not an easy task. Design pattern intent is the shortest path to understand design patterns,so sorting design patterns by analyzing design pattern intent is more beneficial for users to obtain the required design patterns. We propose an intent-based ranking method to facilitate the choice of design patterns,define the relevant similarity calculation method and give the corresponding proof. Then,the parameters defined are adjusted and optimized by genetic algorithm. Finally,the proposed method is verified and analyzed by using the data set of practical problems and design patterns in the actual software development. The experiment shows that compared with other methods,the matching rate of the proposed method is improved to a certain extent,and more correct results appear in the first place after sorting.

**Key words**:WordNet;Stanford Parser;genetic algorithm;design pattern;ranking

## 0 引 言

软件工程的发展使得设计模式的使用成为一种合理的实践,使用设计模式是解决开发团队中反复出现问题的首选方法,它可以避免“造轮子”,也可以简化针对正在解决问题部分的解决方案进行交流[1]。但是在没有任何工具的支持下,要找到正确的设计模式来解决给定的设计问题是非常困难的,因为确定设计模式对给定问题的适用性很大程度上依赖于软件开发人员的经验,而且对于不熟悉设计模式且不知道如何找到最佳模式的新手开发人员来说也是极其困难的[2]。此外还有大量的设计模式,如 Booch 2006,Coad 1995,Douglass 2002 等,对于非常了解模式的开发人员也十分具有挑战性[3-4]。设计模式主要由设计意图、适用性、结构等部分组成,H. Kampffmeyer[5] 认为设计意图部分是理解设计模式的最短路径,因此通过设计模式意图的处理对于设计模式的选择是十分有必要的。

**收稿日期**:2020-10-13 **修回日期**:2021-02-22
**基金项目**:辽宁省自然科学基金(201602583)
**作者简介**:关 慧(1976-),女,博士,副教授,研究方向为软件安全性演化。

## 1 相关研究

很多学者就设计模式的选择做了相关的研究。M. R. J. Qureshi 等[6]在问题驱动帮助用户选择设计模式上做了一番研究,设计模式中一些特定的场景的表达,需要人工参与解释,这样有利于相对精确地选择正确设计模式,但对于问题来说,通常是静态或者通用,而且在模式数量巨大的情况下,构造问题也是一个极具挑战性的任务。Suresh S 等[7]使用文本分类和文本检索技术,为每个设计模式类训练一个分类器,但分类器的分类能力与样本容量和稀疏密度密切相关;Hussian 等[8]基于本体和文本匹配场景,基于本体方法的缺点在于构造很昂贵,并且给方法自动化带来障碍。Gupta 等[9]使用了归一化互相关帮助用户选择模式,归一化互相关技术已经广泛应用于机器视觉,用图像中的相关程度匹配引申到设计模式的匹配上,优点是给类图之间比较提供了量化的方法,但只在结构上判定两个设计是否相似,其正确性也有待斟酌。Smith 等[10]设计的选择方法,基于在设计文档或代码中检测反模式的设计模式,在代码级推荐设计模式是一个好的想法,但在代码开发阶段再考虑设计模式为时已晚,因为软件已经设计好了,还需要更改就变成很麻烦的事。Issaoui 等[11]使用了一个相似矩阵来比较从设计中提取的词语(类名、方法名等)和模式参与者扮演的角色,并与设计人员交互,获得设计片段的上下文的文本描述,通过预先设计的模式问题精炼结果,该方法改进了单一使用某个方面选择模式的缺点,但设计问题需要提前设计,当模式数量巨大时同样将面临问题。P. Gomes 等[12]采用了一种基于案例推论的方法,案例表示在过去的软件设计中已经应用了模式的情况,用类图来描述案例,但其缺点是图并不总是有用的,也不能产生相关性分数。W. Muangon 等[13-14]提出了一种基于案例推理和形式概念分析相结合的内聚技术的解决方案,这种集成使得检索组织能够构造一个完整的设计问题描述,用来寻找合适的设计模式。

## 2 基于意图的设计模式排序

本方法通过 Stanford Parser 对设计模式的意图文本和设计模式实际问题做文本预处理,得到相应的词对集,表示成相应的向量,通过定义的词对匹配评分函数,为设计模式意图和实际问题评出相似度评分,降序排序得到设计模式的排序结果。整体流程如图 1 所示。

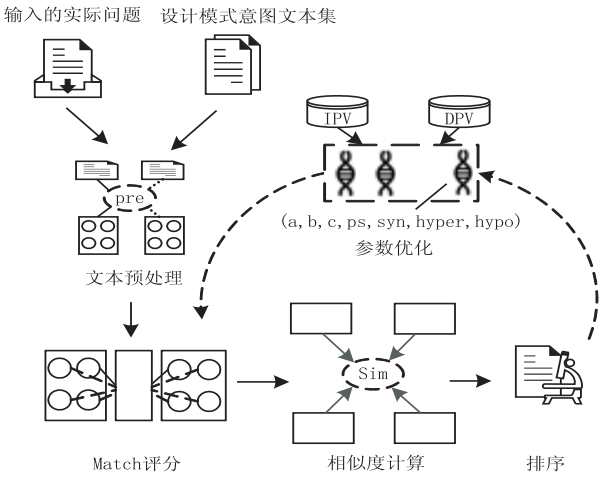


图 1 方法整体流程

### 2.1 文本预处理

从文本提取关键词使用 Stanford Parser 的 Stanford Dependency parser[15-16]生成表述每个句子中成对单词之间的语法依赖类型。在语法分析器提供的 48 个 Type dependency(Td)中,只有四个提供了与输入问题相关的实体和意图[17],分别如表 1 所示。

表 1 四种 Td 的含义与组合词性

| Td | 含义 | Gov | Dep |
|---|---|---|---|
| nusbj | 名词性主语 | V | N |
| nsubjpass | 被动名词性主语 | V | N |
| dobj | 直接宾语 | N | V |
| nmod | 复合名词修饰 | — | — |

词形还原(Lemmatization)被用来还原单词形态,其使用已知单词形式组成的字典,并考虑单词在句子中的作用,将单词标准化为词元。

例如:Decorate 模式的意图“Attach additional responsibilities to an object dynamically. Decorators provide a flexible alternative to sub classing for extending functionality.”解析见表 2。

表 2 Decorate 模式意图抽取 Td 结果

| Td | 语法关系 | 关键词 |
|---|---|---|
| nsubj | nsubj(provide-V, Decorators-N) | ‘decorator provide’ |
| dobj | dobj(Attach-V, responsibilities-N) | ‘attach responsibility’ |
| | dobj(provide-V, alternative-N) | ‘provide alternative’ |
| nmod | nmod_to(Attach-V, dynamically-N) | ‘attach dynamically’ |
| | nomd_for(classing-N, extending-V) | ‘class extending’ |

文本预处理通过上述两个方法,首先将设计模式意图解析成语法关系,提取所需的单词对,然后通过词形还原获得标准化后的单词。

### 2.2　文本向量

设计模式的意图通过解析成单词对,表示成为设计模式向量(DPV),当输入问题时,产生输入问题向量(IPV),每种模式的 DPV 和 IPV 之间需要计算一个相似度,根据计算得出的相似度得分进行排序。由于不同 Td 之间相应位置词的词性有所不同,组合的方式不同,所以将文本预处理获得的单词对分为多个集合,分别为 nsubj,nsubjp,dobj 和 nmod。

DPV 与 IPV 之间的关系如图 2 所示,$X_i$,$Y_i$ 分别表示 DPV 与 IPV 的 TdSet 中的元素。

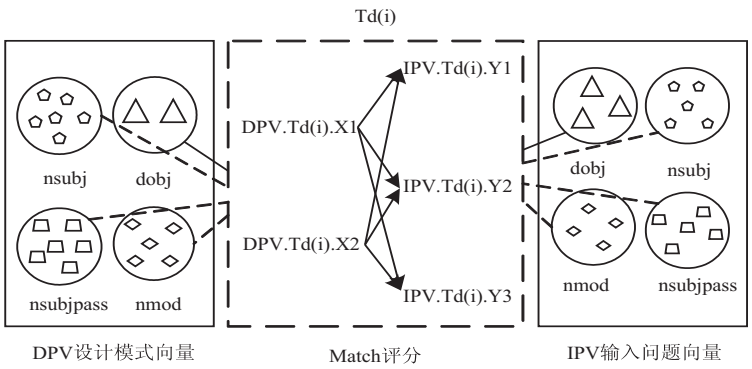


图 2　文本向量之间的关系

### 2.3　Match 评分

为了增加关键词的匹配,考虑 WordNet 中同义词(synonyms)、上位词(hypernyms)、下位词(hyponyms)的匹配,当匹配的是词汇的上位词和下位词时,从匹配得分中扣除惩罚项(Ps),由于单词对比单个单词所蕴含的语义信息更丰富,所以当两个单词都得到匹配时,则获得一定奖励。例如“provide alternative”和“supply alternative”的计算匹配得分为:(1)“supply”是“provide”的上位词,所以匹配的分为 1 - Ps;(2)“alternative”和“alternative”是相同的词,匹配得分为 1;(3)两个词都匹配上,额外获得奖励 1。所以总分为(1-Ps)+1+1。文中定义算法如下:

以下伪代码中 synonyms(a,b),hypernym(a,b)和 hyponym(a,b)在返回 a,b 在 WordNet 中是否为同义/相等,a 是否为 b 的上位词和 a 是否为 b 的下位词,返回的是 boolean 型。

输入:A,B 分别是设计模式和问题相应 TdSet 中的一个元素,Ps 是惩罚项,syn,hyper,hypo 分别为同义、上位和下位词的匹配是否开启的参数,取值 boolean 型。

输出:Score 是 A,B 两个 Td 的相似性评分。

步骤:

```
初始化:govscore=0,depscore=0
if synonyms(A.gov,B.gov) && syn==true then
govscore+=1
end if
if(hypernym(A.gov,B.gov)&&hyper==true) ||
(hyponoym(A.gov,B.gov)&&hypo==true) then
govsocre+=1-Ps
end if
if synonyms(A.dep,B.dep) && syn==true then
depscore += 1
end if
if (hypernym(A.dep,B.dep)&&hyper==true) ||
(hyponoym(A.dep,B.dep)&&hypo==true) then
depsocre+=1-Ps
end if
Score=govscore+depscore
if depscore>0 && govscore>0 then
score+=1
end if
Return scorsse
```

WordNet 中的 Td 类型不同,搭配的词性不同,所以匹配时将不同集合分别计算,计算公式如下:

$$\mathrm{Match}_{Td_i}(\mathrm{DpSet},\mathrm{IpSet})=\sum_{m=0}^{\mathrm{DpSet}_{Td_i}\mathrm{length}}\mathrm{Score}_m$$

其中,$\mathrm{Td}_i \in \{\mathrm{nsubj},\mathrm{nsubjp},\mathrm{dobj},\mathrm{nomd}\}$。

### 2.4　相似度函数

$$\mathrm{Sim}(\mathrm{DPV},\mathrm{IPV})=\frac{\alpha\,\mathrm{Match}_{\mathrm{dobj}}+\beta\,\mathrm{Match}_{\mathrm{nmod}}+\gamma(\mathrm{Match}_{\mathrm{nsubj}}+\mathrm{Match}_{\mathrm{nsubjp}})}{[\alpha\,\mathrm{Num}(\mathrm{Set}_{\mathrm{dobj}})+\beta\,\mathrm{Num}(\mathrm{Set}_{\mathrm{nmod}})+\gamma(\mathrm{Num}(\mathrm{Set}_{\mathrm{nsubj}})+\mathrm{Num}(\mathrm{Set}_{\mathrm{nsubjp}}))]\cdot 3}$$

其中,$\alpha$,$\beta$,$\gamma$ 为权重参数;DPV 为设计模式向量;IPV 为问题向量;$\mathrm{Num}(\mathrm{Set}_{\mathrm{Td}_i})$ 为设计模式 Td 集合中元素的数量;$\mathrm{Match}_{\mathrm{Td}_i}(\mathrm{DpSet},\mathrm{IpSet})$ 为设计模式与问题在 $\mathrm{Td}_i$ 上的匹配评分。

相似度函数性质:

(a)相似度的值应在[0,1];

(b)当进行比较的两个向量属性相同时,相似度为 1,完全不同时,相似度为 0。

证明:

设:设计模式向量为 DPV,输入问题向量为 IPV。

(1)当 DPV≠IPV 时,Sim(DPV,IPV)= 0。

∵ DPV ≠ IPV

∴ $Match_{Td_i}(DpSet,IpSet)=0$,

$Td_i \in \{nusbj,nsubjp,dobj,nmod\}$

∵ $[\alpha Num(Set_{dobj})+\beta Num(Set_{nmod})+\gamma(Num(Set_{nsubj})+Num(Set_{nsubjp}))]*3\neq 0$

$Td_i \in \{nusbj,nsubjp,dobj,nmod\}$

∵ $[\alpha Num(Set_{dobj})+\beta Num(Set_{nmod})+\gamma(Num(Set_{nsubj})+Num(Set_{nsubjp}))]\cdot 3\neq 0$

∴ Sim(DPV,IPV) = 0

$Td_i \in \{nusbj,nsubjp,dobj,nmod\}$

∵ $[\alpha Num(Set_{dobj})+\beta Num(Set_{nmod})+\gamma(Num(Set_{nsubj})+Num(Set_{nsubjp}))]\cdot 3\neq 0$

∴ Sim(DPV,IPV)= 0

(2)当 DPV=IPV 时,Sim(DPV,IPV)= 1。

∵ DPV = IPV

∴ $Match_{Td_i}(DpSet,IpSet)=Num(Set_{Td_i})*3$

$Td_i \in \{nusbj,nsubjp,dobj,nmod\}$

∴ $\alpha Match_{dobj}+\beta Match_{nmod}+\gamma(Match_{nsubj}+Match_{nsubjp})=$

$$\alpha Num(Set_{d0bj})\div 3+\beta Num(Set_{nmod})*3+\gamma(Num(Set_{nsubj})+Num(Set_{nsubjp}))*3$$

∴ Sim(DPV,IPV) = 1

(3)当 DPV 与 IPV 部分相似时,0<Sim(DPV,IPV)<1

∵ DPV 与 IPV 部分相似

∴ $Match_{Td_i}(DpSet,IpSet)\leqslant Num(Set_{Td_i})*3$

∴ Sim(DPV,IPV) ≤ 1

又∵(1)(2)

∴ 0 < Sim(DPV,IPV) < 1

综上,证毕。

### 2.5 示 例

有文本两段如下:

设计模式“Visitor”意图:“Represent an operation to be performed on the elements of an object structure. Visitor lets you define a new operation without changing the classes of the elements on which it operates.”

实际设计问题:“Many distinct and unrelated operations need to be performed on node objects in a heterogeneous aggregate structure. You want to avoid " polluting" the node classes with these operations. And, you don´t want to have to query the type of each node and cast the pointer to the correct type before performing the desired operation.”

根据语法拆分可获得各自 Td 集,如图 3 所示。

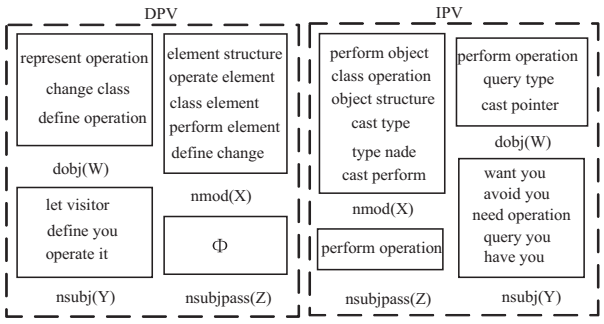


图 3 拆分 Td 集结果

首先确定参数,不妨假设参数组合为:$\alpha=1$,$\beta=1$,$\gamma=1$,Ps = 0.4,syn = true,hyper = true,根据各自 Td 集内的单词对计算相应的 Score,相应求和计算 Td 集的 Match 分数,匹配上的如表 3 所示。

表 3 Match 结果

| IPV | DPV | Match | IPV | DPV | Match |
|---|---|---|---|---|---|
| DPV. W1 | IPV. W1 | 0.0\|1.0 | DPV. Y1 | IPV. Y2 | 0.0\|1.0 |
| DPV. W1 | IPV. W3 | 0.0\|1.0 | DPV. Y2 | IPV. Y2 | 0.0\|1.0 |
| DPV. X1 | IPV. X4 | 1.0\|0.0 | DPV. Y4 | IPV. Y2 | 0.0\|1.0 |
| DPV. X2 | IPV. X3 | 1.0\|0.0 | DPV. Y5 | IPV. Y1 | 1.0\|0.0 |
| DPV. X3 | IPV. X1 | 0.0\|1.0 | DPV. Y5 | IPV. Y2 | 0.0\|1.0 |

由于 DPV. Z=Ø,所以相应的 $Match_{nsubjpass}(DpSet,IpSet)=0$;DPV. W1 匹配 DPV. W1 和 DPV. W3 都等于 1,则取最大值 1,则 $Match_{dobj}(DpSet,IpSet)=1$;DPV. X1,DPV. X2,DPV. X3 分别匹配上了 IPV. X4,IPV. X3,IPV. X1,所以 $Match_{nmod}(DpSet,IpSet)=3$;同理,$Match_{nsubj}(DpSet,IpSet)=4$。由图 3 得,$Num(Set_{dobj})=3$ 即在参数为(1,1,1,0.4,true,true,false)时,实际问题与设计模式之间的相似度为 24.2%。

## 3 参数调优

参数随着问题实例的增加,精确求解方法变得非常耗时,另外参数确定的状态空间随着参数取值范围和精度的增加而增加,就文中参数而言,共 $2^{25}$ 种组合方式。参数调优模块采用遗传算法的框架,将要解决的问题模拟成生物进化过程,进行多次迭代后就有可能会进化出相对表现较好的参数方案,然而所有的遗

传算法都必须具有良好的问题遗传表示,合适的适应度函数才能有效的工作,因此定义规则与适应度函数如下。

### 3.1　编　码(Encode)

编码是应用遗传算法时要解决的首要问题,也是设计遗传算法的一个关键步骤。需要确定的参数有 $\alpha$ , $\beta$ , $\gamma$ ,Ps,syn,hyper,hypo 共 7 个。$\alpha$ , $\beta$ , $\gamma$ 是三种词集的相应的权重,对不同设计模式数据集[17-18]经过 10 轮,每轮 200 代的遗传测试显示,在数值在 0 ~ 10 之间浮动会对相应适应度值有较显著的影响,其他数值几乎无明显影响,所以文中参数设置取[0,10]之间,Ps 是惩罚项,取值范围为(0,1),精确到小数点后 2 位,syn,hyper,hypo 分别是同义词,上位词,下位词的开关参数,取布尔值,各参数取值范围如表 4 所示。

表 4　参数取值范围

| 参数 | $\alpha,\beta,\gamma$ | Ps | Syn | Hyper | Hypo |
|---|---|---|---|---|---|
| 参数含义 | 权值 | 惩罚项 | 同义词 | 上位词 | 下位词 |
| 取值范围 | [0,10] | (0,1) | {0,1} | {0,1} | {0,1} |
| 二进制位 | 7 | 7 | 1 | 1 | 1 |

为了表示以上参数,文中采用二进制编码的编码方式。二进制编码具有编解码操作方便,交叉编译便于实现,符合最小字符集等特点。假设某一参数的取值范围为 $[U_{\min},U_{\max}]$ ,用长度为 $l$ 的二进制编码符号串表示该参数,则 $\delta$ 为该二进制编码的精度。

$$\delta = \frac{U_{\max} - U_{\min}}{2^l - 1}$$

例如:$\alpha$ 的取值范围为[0,10],若用 7 位长的二进制表示,则精度 $\delta = \frac{10 - 0}{2^7 - 1} \approx 0.079$。

以上参数用 31 位的二进制符号串表示,其分布如图 4 所示。

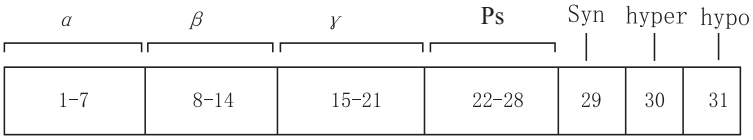


图 4　31 位染色体的遗传表示

### 3.2　解　码(Decode)

从编码后的基因型向可用参数的表现型转换的方式称为解码。假设某一参数的编码为 $b_k b_{k-1} b_{k-2} \cdots b_2 b_1$ ,则解码公式为:

$$X_i = U_{\min} + \frac{U_{\max} - U_{\min}}{2^m - 1} \sum_{i=1}^{m} b_i \times 2^{i-1}$$

### 3.3　适应度函数(Fitness function)

遗传算法中,群体的进化过程就是以群体中各个个体的适应度为依据,通过一个反复迭代的过程,不断寻找求出适应度较大的个体。所以适应度函数的确定也十分重要。设 Rank 为相似度排名后设置展示的数量, RankNum[$i$] 为问题对应的设计模式在 Rank 结果中位于第 $i$ 位的总数量,则适应度函数定义如下:

$$\text{Feritnes}(\alpha,\beta,\gamma,\text{Ps},\text{syn},\text{hyper},\text{hypo}) = \sum_{i=1}^{\text{Rank}} \text{RankNum}[i] * (\text{Rank} - i)$$

当排名越靠前的数量越多时,Fitness 的值越大,并且适应度缓慢增加,有利于寻找到最优解。

## 4　实　验

### 4.1　实验数据集

文中提供两组设计模式数据集与一组安全设计模式数据集。

设计模式数据集:

安全设计模式样本集合(46)[17]。被分为 8 类,系统访问与控制体系结构,访问控制模型,身份识别与认证,操作系统访问控制,安全互联网应用,防火墙体系结构,企业安全与风险管理,账单。

Gof 模式组[18]是设计模式主要参考书籍之一,面向对象设计模式样本集合(23),被分为 3 个粗粒度类别,创建型、行为型、结构型。

实际问题数据集:

为了测试所提出的排序方法的有效性,从不同的资源中检索出实际问题,根据专家意见,检索出已经解决和映射的设计问题

(1)46 个安全设计模式集合的问题[17]。

问题 1:"When objects are created we define the rights processes have to them. These authorization rules or policies must be enforced when a process attempts to access an object."

(2)24 个设计模式集合的问题[19]。

问题 2:"One of the dominant strategies of object oriented design is the "open closed principle". Figure

demonstrates how this is routinely achieved encapsulate interface details in a base class, and bury implementation details in derived classes. Clients can then couple themselves to an interface, and not have to experience the upheaval associated with change: no impact when the number of derived classes changes, and no impact when the implementation of a derived class changes."

(3)7 个面向对象设计模式的问题[20]。

问题 3:"Design a system for drawing graphic images: a graphic image is composed of lines, rectangles, texts, and images. An image may be composed of other images, lines, rectangles, and texts."

### 4.2 对比实验

#### 4.2.1 46 个安全设计模式集合的问题

文中使用三组参数在安全设计模式与实际问题的数据集上进行测试,实验结果如表 5 所示。

表 5 不同参数在前 rank5 中的匹配情况

| 实验组 | 参数组合 | 匹配率/% |
|---|---|---|
| 参数组 1 | 1,1,1,true,true,false,0.4 | 67.39 |
| 参数组 2 | 3,2,1,true,true,false,0.3 | 76.09 |
| 参数组 3 | 3,2,1,true,false,false,0.6 | 67.39 |
| 优化组 | 9.8,5.6,2.5,true,true,false,0.02 | 76.09 |

文中设置三组参数和一组经过遗传算法调优后得到的参数组,经过计算得相应的匹配率结果和各 TopN-Rank 的总匹配占比,如图 5 所示。表中反映了在 Rank 为 5 时,所有实际问题中正确匹配的设计模式出现在前 TopN-rank 的个数和所有出现在 Rank5 的实际问题中在实际问题集中的占比率(即匹配率)。在经过遗传算法调优的参数组合后,出现于前 $n$ 的匹配数目明显上升。Top1-Rank 下,相对参数组 1 和参数组 3,优化组的匹配个数上升至 19,匹配率由原来的 67.39% 上升至 76.09%,涨幅 8.7%。相同匹配率的参数组 2 与优化组相比,优化组的 Rank 匹配率整体上涨了 5.72%。

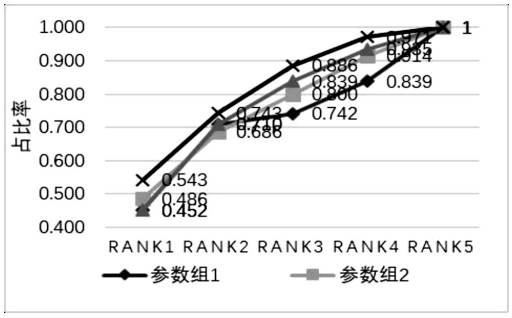


图 5 Top-n Rank 匹配在总匹配中占比

整体上,优化组相比其他参数组的匹配率上升了 5.72% 至 9.13% 不等,匹配率占比也高于其他参数组,经过遗传算法的调优参数,使得准确的结果明显前移,让更多的正确结果出现在了靠前的位置。

#### 4.2.2 ChannaBou 2018 的对比

在设计模式数量 14,实际问题数量 24 的情况下,对比如表 6 所示。

表 6 参数设置(syn,hyper,hypo)为(true,true,false)其他方法[16]与文中方法的对比

| Ps | The percentage of correct answers(1,1,0) | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Top1-Rank | | | Top2-Rank | | | Top3-Rank | | | Top4-Rank | | | Top5-Rank | | |
| | Other (%) | Article (%) | Differ (%) | Other (%) | Article (%) | Differ (%) | Other (%) | Article (%) | Differ (%) | Other (%) | Article (%) | Differ (%) | Other (%) | Article (%) | Differ (%) |
| 0.2 | 29.17 | 37.50 | 8.33 | 37.50 | 41.67 | 4.16 | 54.17 | 58.33 | 4.16 | 66.67 | 66.67 | 0.00 | 70.83 | 79.17 | 8.33 |
| 0.4 | 25.00 | 33.33 | 8.33 | 37.50 | 37.50 | 0.00 | 58.33 | 58.33 | 0.00 | 62.50 | 62.50 | 0.00 | 70.83 | 79.17 | 8.33 |
| 0.6 | 25.00 | 33.33 | 8.33 | 33.33 | 33.33 | 0.00 | 66.67 | 66.67 | 0.00 | 66.67 | 70.83 | 4.16 | 70.83 | 79.17 | 8.33 |
| 0.8 | 25.00 | 33.33 | 8.33 | 41.67 | 41.67 | 0.00 | 66.67 | 66.67 | 0.00 | 66.67 | 70.83 | 4.16 | 70.83 | 79.17 | 8.33 |

通过选取相同的设计模式集合[7]和实际问题集[19-20],在经过参数调优后获得的数据如表 6 所示,Other 和 Article 分别为其他方法和文中方法的表现,Differ 表示两种方法表现的差值。在参数 syn,hyper,hypo 设置为 true,true,false 时,Ps 分别在 0.2,0.4,0.6,0.8 的情况下,使用文中的参数调优方法后,使得各 TopN-Rank 的数量明显增加,Top1-Rank 下,各 Ps 的值下涨幅 8.33%,Top5-Rank 下涨幅 8.33%,整体涨幅 4.16% 至 8.33% 不等,效果可观。

受人工设置参数的局限,通过遗传算法寻找相对较优解替代人工设置参数。实验一的结果表明,经过调优后文中方法得到的结果相对人工定义的参数组合表现较好,使匹配率得到了相应的提升,同时也使得更多正确的结果出现在了靠前的位置。实验二的结果显示,通过与本方法相近的方法的对比,在经过遗传算法调优后,匹配率都得到了相应的提升,相同参数的情况下,Top1-Rank 的匹配率均上调 8.33%,Top5-Rank 的匹配率相应的上调 8.33%,比同类的方法能获得更多的匹配结果,本方法的实验表现总体较优。

## 5 结束语

提出了一种基于意图的设计模式排序方法,定义

了相关的词对 match 规则,并定义了相关的相似度比较办法,随后针对参数取值依赖人工的问题,引入了遗传算法对相应的参数计算相对优值,最后通过实际问题的数据集对所提出的方法进行了验证和分析。实验结果表明,该方法在经过遗传算法的调优后,相对其他的方法有一定的提高,在其他相关设计模式上也表现较好。由于硬件和时间的限制,文中测试的程序和数据集有限,结论有一定局限性。下一步工作,一方面提高数据集的种类和数量,另一方面对意图不易表达的设计模式进行深入的研究。

**参考文献:**


[1] MARKO V. Template based, designer driver design pattern instantiation support[C]//Advances in databases and information systems. Budapest, Hungary: Springer, 2004: 144-158.

[2] KIM D K, EL KHAWAND C. An approach to precisely specifying the problem domain of design patterns[J]. Journal of Visual Languages & Computing, 2007, 18(6): 560-591.

[3] BOOCH G. Handbook of software architecture book[M/OL]. 2006. http://www. booch. com/ architecture/patterns. jsp.

[4] GUSTAVSSON A, ERSSON M. Formalizing the intent of design patterns[M]. [s. l.]: [s. n.], 1999.

[5] KAMPFFMEYER H, ZSCHALER S. Finding the pattern you need: the design pattern intent ontology[C]//International conference on model driven engineering languages and systems. Berlin: Springer, 2007: 211-225.

[6] QURESHI M R J, AL-GESHARI W. Proposed automated framework to select suitable design pattern[J]. International Journal of Modern Education and Computer Science, 2017, 9(5): 43.

[7] SURESH S S, NAIDU M M, KIRAN S A. Design pattern recommendation system (methodology, data model and algorithms)[C]//ICCTAI. [s. l.]: [s. n.], 2011.

[8] HUSSAIN S, KEUNG J, KHAN A A. Software design patterns classification and selection using text categorization approach[J]. Applied Soft Computing, 2017, 58: 225-244.

[9] GUPTA M, PANDE A, RAO R S. Design pattern detection by normalized cross correlation[C]//2010 international conference on methods and models in computer science (ICM2CS-2010). New Delhi, India: IEEE, 2010: 81-84.

[10] NAHAR N, SAKIB K. ACDPR: a recommendation system for the creational design patterns using anti-patterns[C]//Proceedings of 23rd international conference on software analysis, evolution, and reengineering (SANER). Osaka, Japan: IEEE, 2016: 4.

[11] ISSAOUI I, BOUASSIDA N, HANÊNE B A. A new approach for interactive design pattern recommendation[J]. Lecture Notes on Software Engineering, 2015, 3(3): 173-178.

[12] GOMES P, PEREIRA F C, PAIVA P, et al. Using CBR for automation of software design patterns[C]//European conference on case-based reasoning. Berlin: Springer, 2002: 534-548.

[13] MUANGON W, INTAKOSUM S. Case-based reasoning for design patterns searching system[J]. International Journal of Computer Applications, 2013, 70(26): 16-24.

[14] ALHUSAIN S, COUPLAND S, JOHN R, et al. Towards machine learning based design pattern recognition[C]//13th UK workshop on computational intelligence (UKCI). Guildford, UK: IEEE, 2013: 244-251.

[15] DE MARNEFFE M C, MACCARTNEY B, MANNING C D. Generating typed dependency parses from phrase structure parses[C]//LREC. [s. l.]: [s. n.], 2006: 449-454.

[16] BOU C, LAOSEN N, NANTAJEEWARAWAT E. Design pattern ranking based on the design pattern intent ontology[M]//Intelligent information and database systems. Berlin: Springer, 2018: 25-35.

[17] SCHUMACHER M, FERNANDEZ-BUGLIONI E, HYBERTSON D. Security patterns: integrating security and systems engineering[M]. [s. l.]: John Wiley & Sons, 2013.

[18] GAMMA E. Design patterns: elements of re-usable object-oriented software[M]. [s. l.]: Pearson Education India, 1995.

[19] SHVETS A, PAVLOVA M, FREY G. Design patterns explained simply[EB/OL]. 2015. https://sourcemaking.com.

[20] BOUHOURS C, LEBLANC H, PERCEBOIS C. Spoiled patterns: how to extend the GoF[J]. Software Quality Journal, 2015, 23(4): 661-694.