软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2017,28(10):2548−2563 [doi: 10.13328/j.cnki.jos.005130]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

面向软件产品线中特征选择的多目标优化算法*

连小利, 张 莉

(北京航空航天大学 计算机学院,北京 100191)

通讯作者: 张莉, E-mail: lily@buaa.edu.cn

摘 要: 软件产品线中,产品定制的核心是选择合适的特征集.由于多个非功能需求间往往相互制约甚至发生冲突,特征选择的本质是多目标优化过程.优化过程的搜索空间被特征间错综复杂的依赖和约束关系以及明确的功能需求大大限制.另外,有些非功能需求有明确的数值约束,而有些则仅要求尽可能地得到优化.多样的非功能需求约束类型也给优化选择过程带来极大的挑战.提出一种含修正算子的多目标优化算法 MOOFs.首先,设计特征间依赖和约束关系描述语言 DL-DCF 来统一规范特征选择过程中必须遵守的规则,所有的非功能需求都转化为优化目标,相关的数值约束则作为优化过程中特征选择方案的过滤器.另外,设计了修正算子用于保证选择出的特征配置方案必满足产品线的特征规则约束.通过与4种常用的多目标优化算法在4个不同规模的特征模型上的运行结果进行对比,表明该方法能够更快地产生满足约束的优化解,且优化解具备更好的收敛性与多样性.

关键词: 软件产品线;特征选择;多目标优化算法;非功能需求;功能需求

中图法分类号: TP311

中文引用格式: 连小利,张莉.面向软件产品线中特征选择的多目标优化算法.软件学报,2017,28(10):2548−2563. http://www.jos.org.cn/1000-9825/5130.htm

英文引用格式: Lian XL, Zhang L. Multi-Objective optimization algorithm for feature selection in software product lines. Ruan Jian Xue Bao/Journal of Software, 2017,28(10):2548−2563 (in Chinese). http://www.jos.org.cn/1000-9825/5130.htm

Multi-Objective Optimization Algorithm for Feature Selection in Software Product Lines

LIAN Xiao-Li, ZHANG Li

(School of Computer Science and Engineering, Beihang University, Beijing 100191, China)

Abstract: In software product lines, the core of product customization is to select appropriate features. Due to the various competing and even conflicting non-functional requirements (NFRs), feature selection, in essential, is a multi-objective optimization process. What's more, the search space in optimization is constrained largely by the relationships between features and the definitive functional requirements (FRs). Besides, some NFRs are with clear numerical limits, while others are not. These varied types of NFRs also present challenges for feature selection. To solve these problems, a novel multi-objective optimization algorithm with a feature selection reviser is proposed. Firstly, description language for the dependency and constraints relationships between features (DL-DCF) are designed to format different types of relationships between features uniformly, which stipulates the coexistence of two or more features. Next, during selection, all NFRs are transformed to optimization goals, and the quantified constraints on NFRs are used as filters to exclude invalid solutions. Furthermore, a reviser is designed to repair the configuration which violates any relation between features or FRs. Finally, the reviser is planted into the multi-objective optimization framework to form the proposed algorithm, MOOFs, to perform feature selection. Comparing with four popular baselines running on four feature models with different scales, empirical results show notable performance improvement of the algorithm on efficiency of valid solution generation and on the multiple NFRs balancing, especially when the feature models are large and complex.

* 基金项目: 国家自然科学基金(61370058)
Foundation item: National Natural Science Foundation of China (61370058)
收稿时间: 2015-07-13; 修改时间: 2015-11-18, 2016-06-14, 2016-08-18; 采用时间: 2016-09-03; jos 在线出版时间: 2016-10-11
CNKI 网络优先出版: 2016-10-12 16:26:37, http://www.cnki.net/kcms/detail/11.2560.TP.20161012.1626.007.html

Key words: software product line; feature selection; multi-objective optimization algorithm; non-functional requirement; functional requirement

软件产品线是“具有一组可管理的公共特性的软件密集型系统的集合,这些系统满足特定的市场或任务需求,并且按预定义的方式从一个公共的核心资产库中开发得到”[1,2].Boeing,Bosch Group,Nokia(产品线名人堂:http://www.splc.net/fame.html,目前包括 21 位成员,这些成员因成功采用产品线工程开发产品而被选为产品线工程的杰出案例)等多家企业已经通过实践证明,产品线工程是一种提高产品质量、缩短产品上市时间并能同时降低开发成本的有效方法.

产品线工程中开发单个产品的过程称为产品定制,这其中的一个关键步骤便是根据产品需求选择合适的特征集.特征是“由一系列功能和非功能需求来规范的一个逻辑单元”[3].作为 FODA(feature-oriented domain analysis)[4]的一部分而被提出的特征模型,已成为目前表达产品线中所有特征的通用方法.特征模型主要通过描述特征以及特征间的依赖和约束关系来实现对产品线中所有产品的共性和可变性的建模.特征间的依赖和约束关系主要包括强制性(mandatory)、可选性(optional)、选择性(alternative)、多选性(or)、要求性(requires)和排他性(excludes)等.

特征选择面临着多重挑战.

1) 对特征之间的依赖和约束关系的满足,是判断特征选择方案有效的依据.任何违反特征关系的方案都是无效的.然而,这些关系往往数量庞大且错综复杂,常造成这样一种现象:最新选择的一个特征导致已选的多个特征无效,致使特征集发生冲突而不得不重新选择.这样的过程非常普遍,需耗费大量的时间;
2) 对非功能需求的满足与否,越来越成为决定产品成败的关键.大多数非功能属性间都存在着制约甚至冲突关系.众所周知,几乎对任何一个其他质量属性的满足都会造成系统性能在一定程度上的下降.因此,在选择特征的过程中必须权衡产品的多个非功能属性需求,以求达到系统整体质量最优;
3) 非功能需求往往有多种类型.有些有明确数值约束,而有些则没有.明确的数值约束,如手机价格不高于 1 000 元,能够将所有价格高于 1 000 元的配置方案过滤掉.而那些无数值约束的需求,如价格尽可能低等,则要求属性尽可能地得到优化.这些明确及隐含的约束,需要在特征选择过程中综合考虑;
4) 明确的功能需求.一个产品线中的特征数量往往成百上千,如 Linux 2.6.32 内核有 6 320 个特征,而用户明确需要的并不多.有效避免用户对规模庞大的无关特征进行逐个判断,是保证特征选择方法实用性的一个重要方面.

虽然现在已有一些特征自动选择方法,如文献[5−8]等,但是这些方法没有全面考虑以上 4 个问题,而只关注部分问题的解决.特别是在特征间依赖和约束关系的满足及无数值型约束的多非功能属性的优化方面,更是缺少综合处理.目前,针对非功能属性的处理,包括文献[5,6,9−11],都只考虑了有明确数值约束的类型.虽然 Siegmund 等人[7]能够处理无数值约束的非功能需求的优化,他们要求用户对不同非功能需求有明确的权重分配,以便于将多个需求叠加为单一明确的综合优化目标.而实际中,用户对于如性能、可靠性等属性大多只有定性的认识,很难给出明确的数值比重.Sayyad 等人[8]开创性地采用多目标优化算法 MOOA(multi-objective optimization algorithm)来处理互相制约的多个非功能属性,通过与广泛应用的 MOOAs 的对比分析可以得出,IBEA(indicator-based evolutionary algorithm)的表现最为突出.特别是当优化目标增加时,IBEA 优势更加明显.但是 IBEA 是一种相对通用的算法,并未对特征间的约束和依赖关系进行明确的处理;且文献[8]没有考虑对有数值约束的非功能需求的处理,亦没有对实现明确需求的特征集进行处理.

多目标优化算法通过 Pareto 排序等方式并行考虑多个冲突的优化目标,避免了对各正交目标之间的对比与叠加,是权衡多个非功能需求的优选方法.在应用多目标优化算法选择特征时,通常用二进制基因串表达选择方案,用 1、0 表示对应位特征的选择与否.然而在多目标优化算法中,交叉、变异算子等不可避免地会破坏特征之间的约束关系,造成在优化多个非功能属性的同时却降低了方案的正确性.本文的正确性是指选择方案对于

特征间约束和依赖关系的遵从程度.违反的关系越多,正确性越低.当满足特征间所有关系时,称该方案为正确的.

针对正确性被破坏的问题,本文在多目标优化算法框架中设计了一种修正算子,以保证优化的特征选择方案同时满足特征之间的约束.另外,由于特征间的关系有强制性、可选性等多种类型,为便于统一处理,本文设计了特征间依赖和约束关系描述语言 DL-DCF 对特征间的多种关系作统一规范.结合修正算子,本文设计了一种基于遗传算法的多目标优化特征选择算法 MOOFs.首先对特征模型作了简单介绍.第 2 节引出方法 MOOFs.并在第 3 节和第 4 节通过实验证明修正算子的必要性和 MOOFs 在正确方案的生成效率以及非功能属性优化方面的优势.最后总结当前特征选择的自动化方法,并对这些方法进行系统性的比较.

1 特征模型

为了直观地了解特征模型,图 1 给出了一个文献中常用的手机产品线特征模型[8,12].从图中可以看出,特征模型是由特征(矩形)以及特征之间的关系(特征之间的连线)组成的层次化树状结构.为了表示产品的非功能特性,也可以给特征赋予各种属性[5−7].

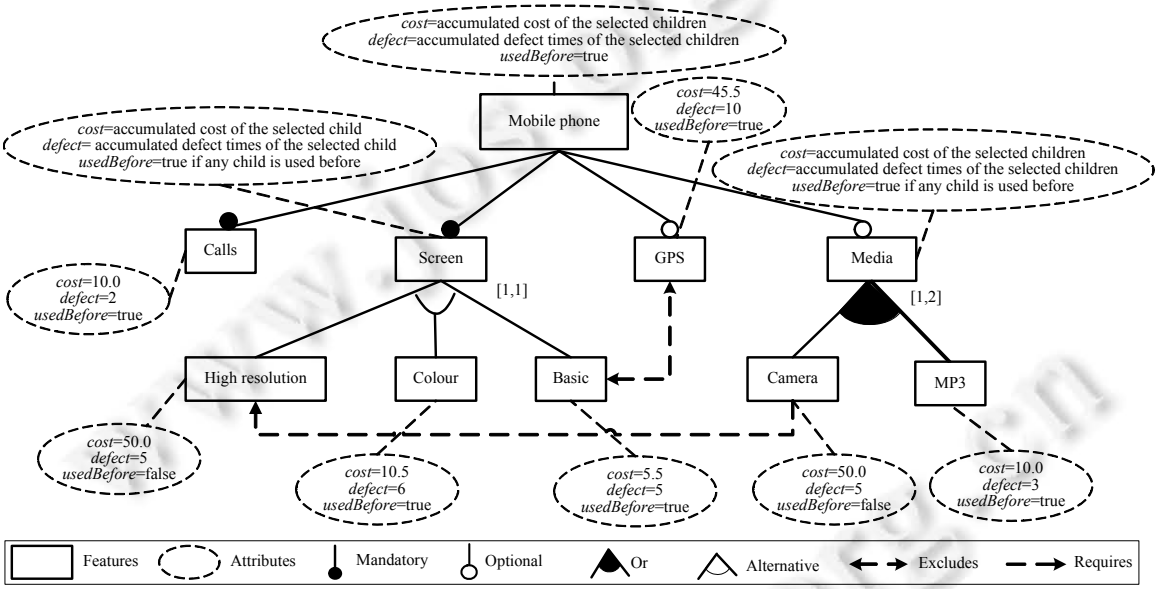


Fig.1 An feature model of mobile product line

图 1 手机产品线特征模型

特征之间的关系可以有如下总结[12].

1) 父子特征之间的关系(实线)

主要的关系类型有强制性、可选性、选择性和多选性.其中,强制性关系表示无论何种情况下一定要选择的特征;可选性表示当父特征被选择后,子特征可以选择或不选;选择性表示当父特征被选择后,子特征集合中只有一个特征可以被选择;多选性表示当父特征被选择后,所有的子特征都可以独立地选或不选.除了这些基本的约束关系外,还可以添加[*n*…*m*]形式的基数来约束允许被选择的特征数量.对于单个特征,基数表示可以实例化的个数.对于特征组来说,表示当父特征被选择后,至少有 *n* 个、最多有 *m* 个子特征可以被选择.

2) 跨层次的约束关系(虚线)

主要的关系类型有要求性(requires)和排他性(excludes)两种.

- 要求性关系表示依赖关系,即:当一个(或一组)特征被选择后,另一个(或一组)特征必须被选中;

- 排他性关系表示互斥关系,即:当一个(或一组)特征被选择后,另一个(或一组)特征必须被排除.

基于图符(即虚线)的跨层次约束关系表达能力有限,现有特征模型往往采用合取范式 CNF(conjunctive normal form)的方式来补充描述跨层次约束.Mendonca 等人[13]研究证明,实践中的特征模型的跨层次约束需要由二元合取范式 2-CNF 和三元合取范式 3-CNF 组合才足以表达.

特征属性可以表示产品的非功能属性,如图 1 中虚线矩形所示.该图中列出了 3 种属性:*cost*,*defect* 和 *usedBefore*.其中,*cost* 是一个浮点数变量,表示该特征的经济属性;*defect* 是整型变量,表示该特征对应组件的出错次数,也表示该特征的实现属性;*usedBefore* 是布尔变量,表示在配置产品的过程中,该特征是否曾被选择过,表示配置属性.通常,叶节点的属性会直接赋值,再通过某种计算得到非叶节点的属性值,如累加相应子节点的属性值[5].在图 1 示例的特征模型中,各特征的属性值均由分布函数随机生成.

产品的非功能属性可以分为定量与定性两类.文献[5−7]等研究了非功能属性的分类以及定性属性的定量化方法.本文将关注点放在如何基于特征已有的属性进行优化选择,而不将产品非功能属性与特征属性之间的映射纳入研究范围.本文假设非功能属性已通过某种方式映射到相关特征的属性,且属性也已经被赋值.

2 一种面向特征选择的多目标优化算法 MOOFs

本节详细介绍本文提出的面向特征选择的多目标优化算法 MOOFs(multi-objective optimization algorithm for feature selection).由于选择过程中需要考虑特征之间的依赖和约束关系以及功能、非功能需求等多个不同方面,选择之前必须将这些因素进行统一表达以实现与多目标优化算法的融合.特别是功能需求和特征之间的关系这两个方面,虽然性质不同,但本质而言都是对选择过程中特征存在与否的限制.为便于处理,本文设计了一种简洁的描述方式 DL-DCF(description language for the dependency and constraints relationships between features).

2.1 相关描述

2.1.1 非功能需求及对非功能需求约束的表达

如上节所述,本文假设非功能属性均已经转化为特征的相关属性且属性得到赋值.非功能需求的约束已表示为相应的等式或不等式,如 *cost*≤￥1000.本文只关注在这些特征及数值基础上进行的多目标优化过程,而不将属性的度量纳入研究范围.

由所选特征属性推导定制产品的属性有多种计算方法,如累加、特定计算公式等[5].由于本文的重点在于如何基于已有的多重特征属性值进行优化的特征选择,故而采用最简单的累加方法,即,定制产品的属性值由所有选择的特征的相关属性累加得到.实际应用中,可以根据需要灵活更换父特征及定制产品属性值的计算方法,而不影响本文特征选择算法的应用.

2.1.2 特征约束和功能需求的表达

由特征模型的介绍可知:特征间存在着多种类型的依赖和约束关系,以不同形式限制选择方案中特征的存在与否.另外,由于功能需求必须通过一系列特征来实现,因此须将其转化为明确的特征需求.当前,有关从需求中抽取特征的工作已比较成熟[14,15],本文不作此方面的研究,假设已存在明确的强制满足的特征集.

为便于对不同类型的特征关系及必须包含的特征集作统一处理,本文定义了特征间依赖和约束关系描述语言 DL-DCF,其乔姆斯基范式见定义 1.

DL-DCF 由多个特征规则组成.一条特征规则可以描述为一个有序对⟨*A*,*B*⟩,其中,*A* 称为特征规则前项,*B* 为特征规则后项,*A* 和 *B* 统称为特征规则项.特征规则项可以是单个特征或者特征组合.单个特征由特征的唯一标识表示,本文为深度优先遍历特征模型得到的特征序列号 *featureID*.当一组特征的存在个数有限制时,可以表达为特征组合,其形式为[α,β,γ,…]{min,max},其中,α,β,γ为同一个特征组合中的特征元素,min 和 max 分别表示组合中可以被同时选中的特征个数的上限和下限.如图 1 所示,Screen 与 High resolution,Color 和 Basic 为特征组合,可以表示为⟨3,[4,5,6]{1,1}⟩,即:设置手机的屏幕时,必须唯一指定屏幕类别.

定义 1. DL-DCF 以四元式(N,Σ,P,S)表示,其中,非终结符集合 N={*F-Rule*,*A*,*B*,*G*,Min,Max,N,N_+},终结符集合

Σ={ε,[,],,,<,>,{,},*,0,1,2,3,4,5,6,7,8,9,*featureID*},开始符集合 S={*F-Rule*}.相应的推演规则 P 为

$$F\text{-}Rule ::= \langle A, B\rangle$$
$$A ::= \varepsilon \mid B$$
$$B ::= featureID \mid G$$
$$G ::= [featureID(, featureID)^{+}]\{\text{Min}, \text{Max}\}$$
$$\text{Min} ::= 0 \mid N_{+}N^{*}$$
$$\text{Max} ::= \text{Min} \mid *$$
$$N ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$
$$N_{+} ::= 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

另外,定义特殊规则前项ε,其对应的后项一定要被满足.此定义便于对排他性类型的跨层次约束及其他强制存在的特征项进行表达.如:GPS excludes Basic 可以表示为⟨ε,[6,7]{0,1}⟩,表示在任何选择方案中,Basic(No.6)和 GPS(No.7)不能同时存在.又如:根节点 Mobile Phone 属于必选节点,可以表示为⟨ε,0⟩.

在定制具体产品时,功能需求对应的特征一定要被选择,其可以用特征前项为ε的规则来描述.以手机产品线为例,对于需求:要求手机必须有 GPS 功能,则其特征规则描述为⟨ε,7⟩.

特征规则含义为:若前项 A 被满足,则后项 B 必须满足.为便于说明,本文对特征规则项的“满足”作了进一步阐述,见定义 2.

定义 2. 当符合以下任意一种情况时,称特征规则项被满足.

1) 若特征规则项为单个特征,且此特征已被选中;

2) 若特征规则项为特征组合,且组合中已被选中的特征数目满足组合的上下限要求.

各条特征规则之间为合取关系,即,一个正确的特征配置方案必须同时符合所有的特征规则.

与 CNF 相比,DL-DCF 对特征间约束和依赖关系的描述可以大大减少约束条目.如特征规则⟨3,[4,5,6]{1,1}⟩对应的 CNF 范式为(–3∨4∨5∨6)∧(–4∨–5∨–6)∧(–4∨–5)∧(–5∨–6)∧(–4∨–6),其中,‘–’表示排除该 ID 所示特征.该简单示例说明,1 条特征规则与 5 条简单析取式的表达能力相当.而一个特征模型中存在着大量类似的约束关系,故特征规则描述数量将大为减少.以第 3 节实验中规模最小、复杂度最低的模型 WebPortal 为例,本文描述的约束为 51 条,CNF 描述的为 84 条,比本文多 65%.特征选择过程伴随着大量对特征规则的逐个检查,以统计特征选择方案违反的特征规则数量,因此,特征关系数目将直接影响选择方法的计算速度.另一方面,本文特征规则的描述是特征模型中约束关系的直观映射,决定了本描述方法的易用性.

按照 DL-DCF 的文法及特征模型的约束类型分析,特征规则包括⟨ε,单个特征⟩,⟨ε,特征组合⟩,⟨单个特征,单个特征⟩,⟨单个特征,特征组合⟩,⟨特征组合,单个特征⟩及⟨特征组合,特征组合⟩这 6 类,这里需要对⟨特征组合,特征组合⟩作特别说明.特征间的约束关系有两种来源:基于图符的特征关系及额外的跨层次约束描述.前者中,无论是父子特征间依赖关系还是跨层次约束,都不会出现⟨特征组合,特征组合⟩的模式;后者基于 CNF 表达的跨层次约束具有更强的表达能力,虽然可能出现⟨特征组合,特征组合⟩形式,但是特征组合往往只规定了可选元素的数目区间,具体单个元素的选择首先取决于与组合外特征及特征组间的依赖关系,最后受制于组合元素个数的约束,因此,特征组合的存在,增加了选择过程中的不确定性.又由于真实特征模型的跨层次约束由 2-CNF 和 3-CNF 共同表达[13],这两种类型的 CNF 均可以用⟨ε,单个特征⟩,⟨ε,特征组合⟩,⟨单个特征,单个特征⟩,⟨单个特征,特征组合⟩和⟨特征组合,单个特征⟩这 5 种规则类型描述,因此,为了提高算法的效率,减少对不确定性的处理,下文只针对前 5 种基本的特征规则类型进行讨论.

为了叙述方便,本文称完全满足所有特征规则的选择方案为正确方案;反之,则不正确.

2.1.3 特征选择方案的基因表达

本文将特征选择方案表达为二进制串,每一位对应特征模型中的一个特征.特征按照深度优先遍历特征模型的顺序排列.当某特征被选择时,对应位为 1;反之为 0.

2.2 面向特征选择的多目标优化算法MOOFs

Sayyad等人[8]通过实验表明:相对于NSGA-II在内的多个应用广泛的多目标优化算法,IBEA在面向多目标优化的特征选择过程中表现得最为突出.据文献[8]分析,IBEA 的主要优势在于其适应度函数集成了用户基于多目标的优化偏好,该偏好反映为对两两解集进行 Pareto 对比的二进制指标,计算公式见算法 1,具体解释见文献[16].本文采用 IBEA 的适应度函数,并引入修正算子设计面向特征选择的多目标优化算法 MOOFs,基本流程见算法 1.

由于多目标优化算法中,交叉和变异操作随机地发生在基因的任意位置中,不可避免地会破坏特征之间的约束关系,进而阻碍正确方案的产生.为此,本文设计了特征选择方案修正算子 *SolutionRevise*,用以将特征选择方案转化为满足特征规则的方案,增加每一代进化开始时种群中正确解的比重.具体地,首先,MOOFs 随机生成一批 0,1 字符串,字符串的长度为特征总个数.0 和 1 分别代表当前位对应的特征的状态:被排除和被选择.然后评估这些随机方案对特征规则的遵守与否,修正违反特征规则的方案,更新初始种群,开始遗传过程.其间,将每一次交叉、变异产生的子代方案都转化为正确方案,以用于下一次迭代.修正算子与多目标优化过程的融合,使得在优化多个非功能属性的同时,也能确保对特征间关系以及功能需求的满足.

参与遗传过程的配置方案可能不满足某非功能属性的数值约束.但 MOOFs 直到优化结束后的 Pareto 排序时,才判断生成方案是否违反数值约束:若违反,则标注无效,并从最终选择的配置方案中排除.原因在于:在随机的遗传过程中,无效方案也极有可能产生出有效的优化配置方案.

算法 1. MOOFs.

输入:α(群体规模);

N(最大遗传代数);

R(特征规则集);

n(特征总数);

输出:A(Pareto 优化特征选择方案集).

1) 初始化,随机生成规模为α的特征选择方案集 P,每个方案的基因长度为 n;
2) 修正 P 中的每个方案,得到修正后的方案集 P^R;
3) 重复以下过程,直至达到最大遗传代数或其他停止条件被满足为止:
   a) 计算 P^R 中每个个体的适应度函数.对于 P^R 中任意一个方案 x_1,其适应度函数为
   $$F(x_1)=\sum_{x_2\in P^R\setminus\{x_1\}}-e^{-\frac{I(\{x_2\},\{x_1\})}{k}}$$,其中,$I(\cdot)$为可选的二进制支配关系指标;
   b) 根据适应度函数值,在群体集 P^R 中进行环境选择;
   c) 从 P^R 中随机选择两个个体,作为父代个体;
   d) 对父代个体执行交叉和变异操作,产生子代个体 O;
   e) 根据 R 修复 O 产生 O',将 O'置于群体集 P^R 中;
4) 对群体集合 P^R 进行 Pareto 排序,排除无效个体,得到优化选择方案集 A.

2.2.1 修正算子 *SolutionRevise*

由第 2.1.2 节的分析可知,本文研究的特征规则包括〈ε,单个特征〉,〈ε,特征组合〉,〈单个特征,单个特征〉,〈单个特征,特征组合〉和〈特征组合,单个特征〉这 5 类.基于此,本文定义了两种法则来决定特征的选择或排除,分别见定义 3 和定义 4.

定义 3(特征选择法则). 某特征 f 尚未被选中也未被排除,当以下任意一个条件为真时,该特征 f 被选择.

1) f 所在的特征规则中尚未有特征被选中;
2) f 作为单个特征是一条特征规则的后项,其对应前项已被满足;
3) f 存在于一个为特征组合的后项中,其对应前项被满足,且特征组合中没有被选中的特征元素个数恰为特征组合的约束下限.

定义 4(特征排除法则). 某特征f既没有被选中也没有被排除,当以下任意一个条件为真时,该特征f被排除.

1) f作为单个特征是某特征规则的前项,其后项为单个特征,且被排除;
2) f 作为单个特征是某特征规则的前项,其后项为特征组合,且该特征组合中没有被排除的特征个数小于特征组合的约束下限;
3) f存在于一则为特征组合的前项中,且该特征组合中尚未被排除的特征个数不小于此特征组合的约束下限要求,且其后项为单个特征,已被排除;
4) f 存在于一则为特征组合的后项中,且该特征组合中已被选中的特征个数等于此特征组合的约束上限,且特征规则前项被满足.

基于特征选择和排除这两个法则,本文设计了特征选择方案的修正算子 *SolutionRevise*.该算子中设计了 3 个集合 S_I,S_E和 R_G.其中,S_I用于存储被选择的特征,S_E是被排除的特征集,R_G用于存储所有后项为特征组合且要求至少有一个特征元素被选择的特征规则.

修正过程从任何配置都必须包含的特征入手,这些特征位于前项为ε的规则中.由于特征组合中可能存在的元素具有不确定性,因此首先选择要求强制满足的单个特征,同时根据规则对有依赖关系的相关特征执行选择和排除操作.然后依次判断待修正方案中被选择的特征,若某特征既不在 S_I 中也不在 S_E 中,则执行选择操作,并判断该操作对 R_G 中规则的影响.当对待修正方案中已选特征判断完毕后,可能会有某些特征尚未明确状态.这些特征必属于待修正方案被排除的特征集,同时,这些特征必独立于S_I和S_E中已完成判断的特征.为了尽可能地最大化修正方案中所选特征数量,对这些特征执行选择操作.理论上,由于特征关系的复杂性,无法保证有效配置的唯一性.修正算子从配置方案中的必选特征开始,生成一种包含尽可能多特征的有效配置方案.同时,并非所有的部分特征配置方案都存在对应的完整有效配置,修正算子也无法保证此点.事实上,只要优化过程中能够产生足够的优化配置即可,无需保证所有的解集都得到修正.Sayyad 等人[17]证明:即使仅在初始群体中添加 30 个有效方案,也可以大大增加有效解产生的效率.修正算子的有效性将在实验部分给出说明.

算法 2. *SolutionRevise*(修正算子).

输入:S(待修正的特征选择方案);

R(特征规则集);

输出:修正后的特征选择方案 S'.

1. $S_I\leftarrow\varnothing$,$S_E\leftarrow\varnothing$,$R_G\leftarrow\varnothing$;
2. 对于 R 中的每条特征规则 r,若后项是特征组合,且下限 min>0,$R_G\leftarrow r$;
3. 对于前项为ε的每条规则 r,若后项为单个特征 f,*IncludeFeature*(f);
4. 依次取 S 中每个被选择的特征 f:
   a) 若 $f\notin S_I$ 且 $f\notin S_E$,*IncludeFeature*(f);
   b) 对于 R_G 中的每个特征规则 g,记后项已被选择的元素个数为 N,若 N<min,重复以下步骤至 $N\geqslant$min:
      i) 对于后项中的每个特征 f',若 $f'\notin S_I$ 且 $f'\notin S_E$,*IncludeFeature*(f');
      ii) 计算被选择的元素个数 N;
5. 依次取 S 中每个被排除的特征 f,若 $f\notin S_I$ 且 $f\notin S_E$,*IncludeFeature*(f);
6. 映射 S_I 与 S_E 到 S'

算法 3. *IncludeFeature*(特征选择算子).

输入:f(当前被选择的特征);

R(特征规则集);

R_G(后项为特征组合的规则集,且每个特征组合的下限均大于 0);

输出:S_I,S_E.

1. $S_I\leftarrow f$;

2. 对于 R_G 中的每条特征规则,记后项 B 中当前被选择的特征数为 N.若 N 等于上限 max,则对于 B 中的每个特征 f',若 $f'\notin S_I$ 且 $f'\notin S_E$,*ExcludeFeature*(f');
3. 对于特征集 R 中的每条特征规则 r:
   a) 若后项 B 为单个特征 f',且前项已被满足,则 *IncludeFeature*(f');
   b) 若后项 B 为特征组合,记 B 中元素个数为 M,被选择的特征数为 N,被排除的特征为 E:
      i) 若 N 等于上限 max,则对于 B 中的每个特征 f',若 $f'\notin S_I$ 且 $f'\notin S_E$,*ExcludeFeature*(f');
      ii) 若 $M–E$ 等于下限 min,则对于 B 中的每个特征 f',若 $f'\notin S_I$ 且 $f'\notin S_E$,*IncludeFeature*(f').

算法 4. *ExcludeFeature*(特征排除算子).

输入:f(当前被排除的特征);

R(特征规则集);

R_G(后项为特征组合的规则集,且每个特征组合的下限均大于 0);

输出:S_I,S_E.

1. $S_E\leftarrow f$;
2. 对于 R_G 中的每条特征规则,记后项 B 中的元素个数为 M,当前被排除的特征数记为 E,若 $M–E$ 等于下限 min,则对于 B 中的每个特征 f',若 $f'\notin S_I$ 且 $f'\notin S_E$,*IncludeFeature*(f');
3. 对于特征集 R 中的每个特征规则 r:
   a) 若后项是单个特征且为 f,且前项为单个特征,记为 f',则 *ExcludeFeature*(f');
   b) 若后项是单个特征且为 f,且前项为特征组合,记为 s_g.记 s_g 中的元素个数为 M,当前被排除的特征个数为 E,且 $M–E\geqslant$min,则重复以下步骤至 $M–E<$min.
      i) 对于 B 中的每个特征 f',若 $f'\notin S_I$ 且 $f'\notin S_E$,*ExcludeFeature*(f');
      ii) 更新被排除的特征个数 E;
   c) 若后项是特征组合 s_g,且 f 是 s_g 的组合元素,且前项为单个特征 f'.记 s_g 中的元素个数为 M,当前被排除的特征个数为 E,且 $M–E<$min,则 *ExcludeFeature*(f').

2.2.2 算法复杂度分析

记特征模型中所含的特征总数为 n,特征规则数为 m,后项为下限大于 0 的特征组合的规则数为 p,特征组合成员平均个数为 c,遗传代数为 g,群体规模为 s.这些参数中,特征总数 n 与特征规则数 m 是一个特征模型复杂度的表征,特征总数与特征规则能够唯一确定一个特征模型,因此,算法复杂度以这两个参数为基本描述单位;而 p,c,g,s 在每次针对固定模型的优化选择中均可视作常数.据分析,*IncludeFeature* 和 *ExcludeFeature* 的时间复杂度均为 $O(p\times c\times m\times\log n)$,故修正算子的复杂度 $O(SolutionRevise)=O(m)+n\times p\times c\times m\times\log n+p^2\times c^2\times m\times\log n=(n\times p\times c\times m+p^2\times c^2\times m)\times\log n$,为特征数 n 的近多项式时间算子、特征规则数 m 的多项式时间.除修正算子外,MOOFs 算法的时间复杂度还与群体初始化算子以及交叉和变异算子相关.实验部分基于 IBEA 算法定制得到的两种算法的时间复杂度均为 $O(s^2)+g\times(n\times p\times c\times m+p^2\times c^2\times m)\times\log n$,为特征数 n 的近多项式时间、特征规则数 m 的多项式时间.

3 实验设置

3.1 特征模型及特征属性设置

为了尽可能地重现Sayyad等人[8]的实验结果以便于与IBEA对比,本文采用与文献[8]同样的特征模型及特征属性,即 SPLOT(http://www.splot-research.org/,这是一个公用的特征模型库平台)中两个真实的特征模型(EShop 和 WebPortal).每个特征有 3 个属性,分别是 *COST*,*USED_BEFORE* 和 *DEFECTS*.其中,*COST* 是区间[5.0,15.0]内满足正态分布的实数值,*USED_BEFORE* 是符合均匀分布的布尔值,*DEFECTS* 是区间[0,10]内符合正态分布的自然数.另外,*USED_BEFORE* 与 *DEFECTS* 之间存在着一定的关系:如果一个特征的 *USED_BEFORE* 为 false,则 *DEFECTS* 必为 0.

为了验证算法的适用性及可扩展性,本文又选择了 SPLOT 中两个规模较大的自动生成的特征模型(FM500

和FM1000)进行实验.4个模型的具体参数见表1.其中,*CTCs*是跨层次约束数目,*CTCR*[5]是跨层次约束中的特征数目与特征总数的比值.*CTCs* 和 *CTCR* 都是特征模型复杂度的评价指标.

Table 1 Parameters of the case feature models

表1 特征模型配置参数

特征模型	特征总数	规则总数	*CTCs*	*CTCR* (%)	Limits
WebPortal	43	51	6	2.6	*COST*≤400
EShop	290	333	21	6.8	*COST*≤2900
FM500	500	572	50	7.8	*COST*≤4000
FM1000	1 000	1 146	100	8.0	*COST*≤9000

由表1可见:本文选择的特征模型随着特征数目的增加,模型的复杂度也在增大,便于以更有力的方式对特征选择方法进行评估.

为了验证 MOOFs 对于无数值约束的质量需求的处理能力,本文在文献[8]中描述的属性的基础上,为每个特征模型的 *COST* 属性设置了上限,见表1.

每种算法都有5个非功能属性优化目标:*COST* 尽可能小、所选特征数尽可能多、*DEFECTS* 尽可能小、未选择过的特征尽可能不用、不能违反特征规则.

3.2 算法的设置

IBEA在具体应用时,需要根据用户偏好设置一个二进制Pareto指标来实例化适应度函数,而文献[8]并未指明其采用的指标类型.目前,IBEA 最常用的两个指标是ε+和 *HD*.因此,本文分别基于这两个指标实现两个具体算法 $MOOF_{\varepsilon+}$和 $MOOF_{HD}$.为对比的公平性,本文也实现了 $IBEA_{\varepsilon+}$和 $IBEA_{HD}$.另外,根据调研[18],在基于搜索的软件工程领域 SBSE 中,应用最多的两个多目标优化算法为 NSGA-II[19]和 SPEA2[20],也作为本文的对比算法.这6种算法均在开源多目标优化算法框架 jMetal[21,22]中得以实现.

6 种多目标优化算法都有一定程度的随机性,为了减少其给实验结果带来的偏差,每种算法对于每个特征模型均独立运行30次.

3.3 多目标优化性能评估指标

对于多目标优化结果的评估,主要从解集的收敛性和多样性两个方面进行,其中,收敛性度量了优化解与Pareto 前沿间的距离,而多样性则度量了优化解在 Pareto 前沿中的分布.为了同时评估解的收敛性与分布性,本文采用超体积 HV[23],SPREAD[19],EPSILON[16]和 IGD[24]这4种常用指标.这4项指标的定义如下.

1) Hypervolume(HV)[23]计算优化解在目标空间所覆盖的体积.令 Pareto 前沿为 *PF*,*r* 为优化解,则 HV 定义如下: $HV(PF,r)=\lambda\left(\bigcup_{s\in PF} space(s,r)\right)$,其中,$space(s,r)=\{v|r\prec v\prec s\}$为优化解与 Pareto 前沿形成的超体积.超体积由所有被 *s* 支配并同时支配 *r* 的所有解 *v* 形成;

2) SPREAD[19]通过度量扩展程度来评估优化解的分布性.令 d_i为连续两个优化解间的欧式距离,$\bar{d}$ 为所有优化解间距离的平均值,d_f 和 d_l 分别为所有优化解距离中的最大和最小值,*N* 为优化解的个数,则 SPREAD 可定义为 $\Delta=\dfrac{d_f+d_l+\sum_{i=1}^{i=N-1}|d_i-\bar{d}|}{d_f+d_l+(N-1)\bar{d}}$.SPREAD 值越大,说明优化解越多样化;

3) *EPSILON*(ε)[16]:不失一般性,假设优化问题中目标值越小越优.解 $\mathbb{Z}^1=(z_1^1,z_2^1,...,z_n^1)$ ε-支配解 $\mathbb{Z}^2=(z_1^2,z_2^2,...,z_n^2)$,当且仅当对于任一给定的$\varepsilon$>0, $\forall 1\leqslant i\leqslant n: z_i^1\leqslant\varepsilon\cdot z_i^2$.指标 EPSILON 可定义为

$$\varepsilon(A,B)=\inf_{\varepsilon\in R}\{\forall z^2\in B,\exists z^1\in A: z^1\succ_\varepsilon z^2\}.$$

越小的 EPSILON,表示优化解与 Pareto 前沿间距离越小;

4) 反转世代距离 Inverted Generational Distance(IGD)[24]:令 Pareto 最优解为 $P=\{P_1,P_2,\ldots,P_{|P|}\}$,优化解集

为 A,则 A 与 P 间的 IGD 距离可定义为 $IGD(A,P)=\frac{\sum_{\tau\in P}d(\tau,A)}{|P|}$, 其中,$d(\tau,A)$为$\tau\in P$ 到 A 的最近欧式距离.IGD 的值越小,意味着 A 与 P 间的距离越短.

4 实验结果与分析

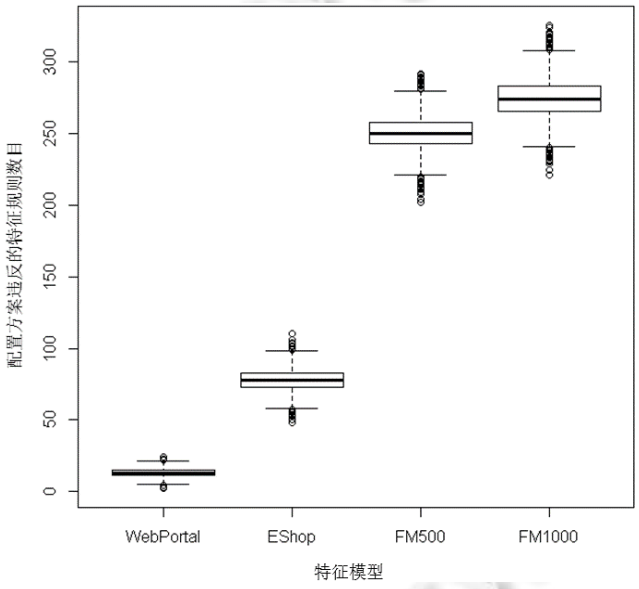
本节首先说明修正算子的必要性,然后从产生有效解的时间效率以及非功能需求优化性能两方面对 6 种算法进行比较.本文采用 4 种最常用的优化性能评价指标 HV[23],SPREAD[19],EPSILON[16]和 IGD[24].由于这 4 项指标的运算需基于优化解及最优解的所有目标值,而实验中无法获得特征模型的最优解,故本文用 6 种算法得到的特征模型的所有优化解作为 Pareto 前沿,这也是针对此问题的最常用的解决方法[22].

4.1 修正算子的必要性

改善种群的基因库,可以促使更多优秀子代的产生.由此可推,每次遗传过程开始之初,向基因库中添加有效解,能够引导更多有效解的产生.

有效解的来源有 3 个:(1) 初始种群中随机产生;(2) 遗传过程中产生;(3) 在优化过程中修正无效解.

首先分析随机产生的配置方案对特征规则的违反程度,以评估第 1 种有效解来源的可能性.具体方法为:针对每个特征模型,随机产生 25K 个配置方案.计算每个配置方案违反的特征规则数目,其分布情况见图 2 中箱形图及具体参数.观察箱形图的下边缘可以看出,4 个特征模型对应的配置方案所违反的规则数目的最小值均大于 0.即:在 4 个特征模型的共 100K 个初始方案中,无有效解产生.从图 2 所示右侧表格中得知:即使对于特征规模最小、复杂度最低的 WebPortal 模型,其配置方案最少也违背了 2 条规则.随着模型规模的扩大,复杂度的提高,配置方案违背的规则数目急剧增大.结合表 1 中 4 个特征模型的规则总数,容易计算得知:对于这 4 个特征模型,其初始配置方案平均违反的特征比例在 23%~43%之间(25K 个配置方案违反的特征数目平均值/规则总数).



	违反的特征规则数目的最小值	违反的特征规则数目的中位数	违反的特征规则数目的平均值
WebPortal	2	13	13.01
EShop	48	78	78.13
FM500	202	250	250.11
FM1000	221	274	274.31

Fig.2 The number of violated rules in the initial configurations shown in box plot and the relevant numbers

图 2 初始化配置方案违反的特征规则数目——箱形图及相关数据说明

随机的交叉变异算子虽然能在一定程度上产生有效解,但产生复杂特征模型的有效解则相当困难,具体数字见表 2.因此,本文设计修正算子来增加种群中有效解的个数,以期提高优化过程中产生的配置方案的质量.

Table 2 The comparisons of the valid solutions generated by the six algorithms

表 2 6 种算法产生有效解的时间对比

特征模型	算法	遗传代数	总运行时间(秒)	有效解个数	平均耗时(秒)
WebPortal	NSGA-II	25K	131.45	239	0.55
	SPEA2	25K	531.72	9	59.08
	$IBEA_{HD}$	25K	286.00	2 523	0.11
	$IBEA_{\varepsilon+}$	25K	412.82	19	21.73
	$MOOF_{HD}$	25K	2 415.48	3 000	0.80
	$MOOF_{\varepsilon+}$	25K	**390.00**	**3 000**	**0.13**
EShop	NSGA-II	500K	5 997.48	127	47.22
	SPEA2	500K	10 852.177	27	401.93
	$IBEA_{HD}$	25K	384.303	27	14.22
	$IBEA_{\varepsilon+}$	500K	10 877.69	80	135.97
	$MOOF_{HD}$	25K	5 457.69	3 000	1.82
	$MOOF_{\varepsilon+}$	25K	**414.31**	**2 980**	**0.14**
FM500	NSGA-II	10M	–	–	–
	SPEA2	10M	–	–	–
	$IBEA_{HD}$	500K	9 321.16	814	11.45
	$IBEA_{\varepsilon+}$	10M	–	–	–
	$MOOF_{HD}$	25K	8 010.11	3 000	2.67
	$MOOF_{\varepsilon+}$	25K	**597.21**	2 520	**0.24**
FM1000	NSGA-II	10M	–	–	–
	SPEA2	10M	–	–	–
	$IBEA_{HD}$	10M	–	–	–
	$IBEA_{\varepsilon+}$	10M	–	–	–
	$MOOF_{HD}$	25K	14 233.49	3 000	4.74
	$MOOF_{\varepsilon+}$	25K	**815.12**	**2 998**	**0.27**

注: –表示未能产生有效解

4.2 运行时间对比

特征选择的前提是对特征间依赖和约束关系的遵守,否则,即使质量属性优化最好的方案也是无效的.因此,下文的评估只针对各种算法产生的有效解进行.本节对比各种算法产生有效解的效率.由于 IBEA 等算法往往在最后环节进行 Pareto 排序,然后删除被支配解,在此过程中,有效解也可能被淘汰,所以在优化过程中,比较第 1 个有效解所需时耗无意义.另一方面,为了降低算法的随机性对选择结果的影响,本节的比较基于算法对于一个特征模型产生单个有效解的平均时间,即,30 次独立运行的时间总和除以有效解的总数.

6 种算法的遗传代数初始值均为 25K(jMetal 中默认值).若某种算法在当前遗传代数下无有效解产生,则将该参数依次设为 500K,1M,2M,5M,10M 进行尝试.若在 10M 的情况下仍无有效解产生,则实验结束.

从表 2 中可以得出如下结论.

1) 对于规模较小的 WebPortal 和 EShop 模型,6 种算法都能产生有效解.而随着特征模型规模的扩大,复杂度的增加,本文算法 $MOOF_{HD}$ 和 $MOOF_{\varepsilon+}$ 的优势更加明显.在 4 种对比算法中,对于模型 FM500,只有 $IBEA_{HD}$ 产生了有效解.而对于规模更大的 FM1000 模型,4 种对比算法均未产生有效解.而 $MOOF_{HD}$ 和 $MOOF_{\varepsilon+}$ 均可在遗传代数为 25K 时得到有效解,说明了修正算子的有效性,也说明了 MOOFs 较好的通用性;
2) 从“平均耗时”列可知:对于规模最小的 WebPortal 模型,$IBEA_{HD}$ 产生一个有效解的平时耗时最短.而对于其他规模更大的模型,$MOOF_{\varepsilon+}$ 表现最好,$MOOF_{HD}$ 次之.对于规模最大的 FM1000 模型,$MOOF_{\varepsilon+}$ 产生一个有效解的平均耗时不超过 0.3s,$MOOF_{HD}$ 也平均在 5s 之内产生一个有效解.

因此相对于 4 种对比算法,基于 HD 和ε+实例化的 MOOFs 在面向含有约束空间多属性优化特征选择时,无论从效率还是通用性上都有更好表现.

4.3 算法优化性能评估

本节评估 6 种算法对于多个非功能属性进行优化的表现.由于这 6 种算法均具有一定的随机性,为使评估

结果更加可信,应该采用某种统计检验方法[25].由于非参数 Mann-Whitney 的 U-检验对于数据分布要求较低,本文采用此方法评估 6 种算法产生的优化解集的收敛性和多样性.

由表 2 分析可知,$MOOF_{\varepsilon+}$产生一个有效解的平均用时最短.本节将 NSGA-II,SPEA2,$IBEA_{\varepsilon+}$,$IBEA_{HD}$ 和 $MOOF_{HD}$ 这 5 种算法对于每个模型产生的有效解分别与 $MOOF_{\varepsilon+}$的解进行对比.每一次对比均假设 $MOOF_{\varepsilon+}$表现不劣于对比算法,因此,假设检验为单边检验过程.假设检验的置信水平为 0.95,显著性水平α为 0.05.比较基于 P 值进行.P 值表示对原假设的支持程度.当 P 值不小于 0.05 时,在显著性水平α下接受原假设,即 $MOOF_{\varepsilon+}$表现较优.如前文所述,由于违反特征间约束或者特定功能需求的特征选择方案在实践中并无意义,本文在对比优化结果时首先去除了不正确方案,仅关注各种算法产生的有效解的质量属性.故表 3 中只列出了能够产生有效解的各种算法的对比结果,其中,黑体标识的单元格表明 $MOOF_{\varepsilon+}$表现较优的情况."对比统计"行与列分别总结了 $MOOF_{\varepsilon+}$优于及劣于对比算法的次数.

Table 3 The U-test results on HV, SPREAD, IGD and EPSILON of the selections made by the six algorithms

表 3 6 种算法关于 HV,SPREAD,IGD,EPSILON 的 U-检验对比结果

特征模型	算法	遗传代数	HV (*p*-value)	SPREAD (*p*-value)	IGD (*p*-value)	EPSILON (*p*-value)	对比统计
WebPortal	NSGAII	25K	**1.0**	**1.0**	**1.0**	**1.0**	**4**
	SPEA2	25K	**1.0**	**1.0**	**1.0**	**1.0**	**4**
	$IBEA_{HD}$	25K	<2.2e–16	**1.0**	<2.2e–16	**1.0**	**2↑2↓**
	$IBEA_{\varepsilon+}$	25K	**1.0**	**1.0**	**1.0**	1.36e–06	3↑1↓
	$MOOF_{HD}$	25K	5.5e–06	**1.0**	1.06–04	**0.12**	**2↑2↓**
EShop	NSGAII	500K	**1**	**1**	**1**	**1.0**	**4**
	SPEA2	500K	**1**	**1**	**1**	**1.0**	**4**
	$IBEA_{HD}$	25K	**0.98**	**0.99**	**1**	**0.87**	**4**
	$IBEA_{\varepsilon+}$	500K	**1**	**1**	**1**	**1.0**	**4**
	$MOOF_{HD}$	25K	0.03	**1**	3.0e-03	**1.0**	**2↑2↓**
FM500	$IBEA_{HD}$	500K	6.1e–09	**1**	**0.62**	4.02e–06	2↑2↓
	$MOOF_{HD}$	25K	**0.74**	**1**	**0.20**	**1.0**	**4**
FM1000	$MOOF_{HD}$	25K	0.01	**0.33**	**0.92**	**1.0**	**3↑1↓**
对比统计	$MOOF_{\varepsilon+}$	–	**8↑5↓**	**13↑**	**10↑3↓**	**11↑2↓**	**42↑10↓**

从表 3 中可以看出:

1) 与 NSGA-II 和 SPEA2 比较,$MOOF_{\varepsilon+}$在 4 种指标上表现均更优,即:$MOOF_{\varepsilon+}$产生的优化解更接近 Pareto 前沿,并且这些优化的特征配置方案具有更好的多样性.而与 $IBEA_{\varepsilon+}$和 $IBEA_{HD}$ 在 3 个模型运行结果的 20 次比较中发现,$MOOF_{\varepsilon+}$有 15 次优 5 次劣.因此可以得出结论:整体上讲,6 种算法中,$MOOF_{\varepsilon+}$最优;
2) 结合表 2 可知,IBEA 产生一个有效解的时间小于 NSGA-II 和 SPEA2.因此,IBEA 在面向多个非功能属性的优化特征选择问题上确实优于 NSGA-II 和 SPEA2,此结论与文献[8]一致;
3) 与 $MOOF_{HD}$ 的 16 次比较中,$MOOF_{\varepsilon+}$优 11 次,劣 5 次.且从表 2 中获知,$MOOF_{\varepsilon+}$以更快的速度产生有效解,因此对于 4 个特征模型来说,$MOOF_{\varepsilon+}$更优.

综合来看,在与 NSGA-II,SPEA2,$IBEA_{\varepsilon+}$,$IBEA_{HD}$ 以及 $MOOF_{HD}$ 这 5 种算法针对 4 个特征模型优化解在 4 项质量指标上的 52 次比较中,$MOOF_{\varepsilon+}$有 42 次优于对比算法,仅 10 次劣于对比算法.因此可以说,$MOOF_{\varepsilon+}$在面向多个非功能属性优化的特征选择问题上不仅能更快地产生有效解,且其解有更好的收敛性及多样性.

另外需要说明的是,MOOFs 能够将 4 种特征模型中所有不满足 *COST* 数值约束的特征选择方案排除.

4.4 有效性分析

实验部分采用多个随机生成的非功能属性来验证 MOOFs,特别是 $MOOF_{\varepsilon+}$,在特征优化选择方面(即:在满足特征约束及功能需求的前提下,权衡多个非功能属性来实现特征优化选择)的有效性.真实项目中的特征属性在极大概率上与本文假设的特征属性值的分布规律不一致,可能会造成与本文实验结果稍有不同.但由于本文将 4 个模型完全相同的属性值应用于 6 种算法,可以排除随机产生的特征属性值对实验结果公平性的影响.

算法的随机性可能会导致特征选择结果的偶然性.为了减少随机性的影响,每种算法对每个模型的优化选

择均独立运行 30 次,每个模型产生 3 000 个方案.基于平均值分析的运行时间对比及统计检验分析的优化解集质量对比,进一步减少了随机性对实验结果公平性的影响.

6 种随机算法的初始种群均为随机产生,则初始种群的质量可能会影响优化过程.实际上,我们对 6 种算法产生的 4 个模型的初始种群对本文的 5 个优化目标值分别做了 U-检验统计分析,结果显示,6 种算法随机产生的初始化选择方案并无明显差距.由于篇幅所限,这里不再列出分析过程及结果.

5 相关工作及分析

当前已有一些关于产品线中特征选择的工作,为了对这些方法进行系统的比较,基于产品线特征选择领域的需求与挑战,本文设计了一些比较准则:(1) 能够优化无数值约束的多个非功能需求;(2) 能够处理有数值约束的非功能需求;(3) 能够处理明确的功能需求;(4) 确保符合特征间的依赖和约束关系;(5) 配置过程自动化;(6) 能够表达用户的偏好;(7) 时间效率;(8) 优化过程尽可能不依赖用户.

我们首先在第 5.1 节对各项工作进行简述,然后在第 5.2 节基于比较准则对各工作进行多角度评价.

5.1 相关工作

将特征选择问题转化为约束满足性问题 CSP,是目前最常见的做法,如文献[5,7,9].Benavides 等人[5]首先将特征选择问题转化为约束满足性问题;然后基于商用的 CSP 解算器,使用约束规划技术对约束满足性问题进行推理.Siegmund 等人[7]提出一种集非功能属性的定义、度量及优化于一体的综合性特征选择方法 SPL Conqueror.SPL Conqueror 提供了图形化的界面供用户定义合适的参数,将多个冲突的非功能属性集成为一个优化目标.White 等人[9]将产品线配置视为多步骤过程并转化为约束满足性问题,然后提出一种自动化方法 MUSCLE 选择特征,考虑连续两步配置时的非功能属性满足情况.

为支持产品推演过程中对非功能属性的考虑,Siegmund 等人[6]将特征模型与其实现单元集成到一个统一的模型中,将非功能属性表示为实现单元的属性.在半自动化的产品推演过程中,用户定义的非功能属性的数值约束有助于排除配置空间中不满足约束的方案,但不考虑无数值约束的质量属性的优化.

Sayyad 等人[8]首先将多目标优化算法应用于面向多个非功能属性的特征选择问题中,通过比较包括 NSGA-II,SPEA2,IBEA 等在内的多个广泛应用的多目标优化算法,展现了 IBEA 算法在特征选择问题中的突出表现,强调 IBEA 考虑用户优化偏好的重要性.

Soltani 等人[26,27]将特征选择问题转化为规划问题.基于用户对每个非功能属性的偏好,为每个特征定义一个综合的非功能需求效应值,然后将特征模型以及用户对于各种非功能需求的约束转化为层次化任务网络模型 HTN.基于一种开源的规划系统 SHOP2 来实现自动配置.在选择的过程中,通过选择最大的产品效应值实现针对非功能需求的优化配置.

White 等人[28]将有资源约束的特征选择问题转化为多维多选择背包问题,并针对此背包问题提出了优化算法 FCF.

Guo 等人[10]提出一种基于遗传算法的特征选择方法 GAFES,该方法主要针对资源约束问题,通过罚函数的方法控制特征选择方法对多个资源约束的满足情况.该工作也提出了一种特征配置方案的修改算子,但该算子只针对基本的特征模型,没有考虑特征之间的基数关系以及额外的 3-CNF 跨层次约束描述.

5.2 对比分析

表 4 是对已有的特征选择方法的分析结果.

- 功能需求处理

Siegmund 等人[6]提供工具 SAD 支持用户选择特征,包括功能需求对应的特征.类似地,SPL Conquer[7]提供与用户的交互界面,供用户选择功能需求特征.Soltani 等人[26]将特定的功能需求特征描述为规划问题必须满足的初始条件.本文用特征选择过程中必须遵守的特征规则来描述功能需求.

- 非功能需求处理

除 Sayyad 等人[8]所提方法以外的所有方法均可处理有约束的非功能需求,其中,Siegmund 等人[6]在每一步选择中都对非功能需求的满足情况做出判断,及时删除无效的配置;文献[5,7,9]基于约束满足性问题 CSP 的处理方法判断对数值约束的满足与否;Soltani 等人[26]将非功能需求的约束表达为规划过程中必须满足的初始条件;文献[11,27]专门针对有资源约束条件下的特征选择问题.本文在遗传过程结束后的 Pareto 排序阶段,判断方案对非功能需求约束的满足情况,过滤不满足约束的配置方案.

只有 Sayyad 等人[8]和本文的方法能够处理没有约束的多个非功能需求,通过将这些非功能需求转化为特征选择过程中的多优化目标,在遗传过程中权衡选择较优的 Pareto 集合.然而,Sayyad 等人[8]并未提及对于有数值约束的质量属性的处理.

- 对特征模型的满足性

除 White 等人[5]所提方法以外的所有方法,在特征选择过程中均考虑了完整的特征约束.文献[5]没有对跨层次约束做出处理.

- 自动化配置过程

所有方法都提供了算法或工具支持特征选择过程,但 Siegmund 等人[6]仅根据用户的当前选择过滤可能的特征,支持半自动特征选择.

- 用户偏好的处理

文献[7,9,28]将多个非功能需求叠加为一个综合目标,叠加因子由用户偏好确定.但实际上,合适的叠加因子的确定往往非常困难.Sayyad 等人[8]和本文的方法考虑了用户优化过程中对优化指标的偏好.文献[26]考虑了不同用户对不同非功能需求之间的偏好.

- 时间效率

基于 CSP 的方法,如文献[5,7,9,26],往往需要昂贵的时间开销[26,28].文献[10,11]和本文都提出了多项式时间复杂度的方法.

Table 4 The analysis of the existing feature selection approaches

表 4 已有特征选择方法对比分析

评价指标 / 方法	功能需求	多个无约束的非功能需求	有约束的非功能需求	特征模型满足性	自动化	用户偏好	时间效率	对用户的依赖
SAD(Siegmund 等人)[6]	+	–	+	+	+/–	–	–	–
SPL conqueror(Siegmund 等人)[7]	+	–	+	+	+	+/–	+/–	–
MEOAs(Sayyad 等人)[8]	–	+	–	+	+	+	–	+
HTN(Soltani 等人)[26,27]	+	–	+	+	+	+	+/–	–
CSP(Benavides 等人)[5]	+	–	+	–	+	–	+/–	+
MUSCLE(Whilte 等人)[9]	–	–	+	+	+	+/–	+/–	–
FCF(White 等人)[28]	–	–	+	+	+	+/–	+	–
GAFES(Guo 等人)[10]	–	–	+	+	+	+/–	+	–
我们的方法	+	+	+	+	+	+	+	+

注: +满足, –不满足,+/ –部分满足

6 结束语

本文针对产品线工程产品配置过程中的特征之间的约束、特定的特征需求以及多个非功能需求这 3 个挑战,提出一种基于遗传算法的产品线特征优化选择算法框架 MOOFs,并基于两个常用指标 HD 和ε+实例化得到两种算法 $MOOF_{HD}$ 和 $MOOF_{\varepsilon+}$.通过实验证明了这两种算法,特别是 $MOOF_{\varepsilon+}$在产生有效解的效率、有效解集的收敛性及多样性方面,均优于当前广泛应用的多目标优化算法,甚至 IBEA.

下一步工作中,本文将致力于以下 3 件事情.

(1) MOOFs 并没有对种群初始化算子、交叉算子、变异算子等做出约束,而直接基于 Sayyad 等人[8]的实验结论,采用 jMetal 中 IBEA 默认的遗传算子,并与修正算子结合得到两种具体算法,显示了这两种算法在特征选择中的突出表现.下一步工作中,我们将采用多目标优化算法来寻找针对不同特征模型的

最优算子组合方案,以确定 MOOFs 中最佳的算子组合;

(2) 由于在特征之间复杂的约束和依赖关系以及一些功能或非功能条件的约束下,不一定存在有效配置方案,且不同功能、非功能需求约束下有效解的个数也有很大差异,可能会影响 MOOFs 的有效性.下一步工作中,我们将对此问题进行研究;

(3) 目前并无公用的真实的特征属性及赋值,本文随机产生子特征的属性值并通过累加的方式获取父特征及产品所需的属性值.为了获取真实项目中特征的属性值,需要建立特征与代码间、特征与需求间的追踪关系,并分析用户的具体偏好,这也是本文的下一个工作目标;

(4) 虽然大多数特征是布尔型的,然而还有一些需要配置具体值,如设置整数或字符串变量[29],计划在下一步工作中研究并行基因模型及相关算法以同时对这两种类型的特征进行优化选择.

致谢 在此,我们对北京航空航天大学软件所刘辉老师对本文算法性能优化方面的指导致以衷心的感谢.

References:

[1] Clements P, Northrop L. Software Product Lines: Practices and Patterns. 3rd ed., Boston: Addison-Wesley Professional, 2001.

[2] Clements P, Northrop L, Wrote; Zhang L, Wang L, Trans. Software Product Lines: Practices and Patterns. Beijing: Tsinghua University Press, 2004 (in Chinese).

[3] Bosch J. Design and Use of Software Architectures: Adopting and Evolving a Product-Line Approach. New York: ACM Press/ Addison-Wesley Publishing Co., 2000.

[4] Kang K, Cohen S, Hess J, Novak W, Peterson S. Feature-Oriented domain analysis (FODA) feasibility study. CMU/SEI-90-TR-21. Pittsburgh: Carnegie-Mellon University Pittsburgh Pa Software Engineering Inst, 1990. 1−161.

[5] Benavides D, Ruiz-Cortés A, Trinidad P. Automated reasoning on feature models. In: Pastor O, *et al*, eds. Proc. of the 17th Int'l Conf. on Advanced Information Systems Engineering (CaiSE 2005). Berlin, Heidelberg: Springer-Verlag, 2005. 491−503. [doi: 10.1007/11431855_34]

[6] Siegmund N, Kuhlemann M, Rosenmüller M, Kästner C, Saake G. Integrated product line model for semi-automated product derivation using non-functional properties. In: Heymans P, Yang KC, Metzger A, Pohl K, eds. Proc. of the Int'l Workshop on Variability Modelling of Software-Intensive Systems (VaMoS 2008). 2008. 25−32. http://www.cs.cmu.edu/~ckaestne/pdf/tmp/vamos08.pdf

[7] Siegmund N, Rosenmüller M, Kuhlemann M, Kästner C, Apel S, Saake G. SPL conqueror: Toward optimization of non-functional properties in software product lines. Software Quality Journal, 2012,20(3-4):487−517. [doi: 10.1007/s11219-011-9152-9]

[8] Sayyad AS, Menzies T, Ammar H. On the value of user preferences in search-based software engineering: A case study in software product lines. In: Notkin D, Cheng BHC, Pohk K, eds. Proc. of the 2013 Int'l Conf. on Software Engineering (ICSE 2013). Piscataway: IEEE Press, 2013. 492−501. [doi: 10.1109/ICSE.2013.6606595]

[9] White J, Dougherty B, Schmidt DC. Automated reasoning for multi-step feature model configuration problems. In: Proc. of the 13th Int'l Software Product Line Conf. (SPLC 2009). 2009. 11−20. http://dl.acm.org/citation.cfm?id=1753238

[10] Guo J, White J, Wang G, Li J, Wang Y. A genetic algorithm for optimized feature selection with resource constraints in software product lines. Journal of Systems and Software, 2011,84(12):2208−2221. [doi: 10.1016/j.jss.2011.06.026]

[11] Mendonca M, Branco M, Cowan D. S.P.L.O.T.-Software product lines online tools. In: Proc. of the 24th ACM SIGPLAN Int'l Conf. on Object-Oriented Programming, Systems Languages and Applications (OOPSLA 2009). New York: ACM Press, 2009. 761−762. http://www.splot-research.org/ [doi: 10.1145/1639950.1640002]

[12] Benavides D, Segura S, Ruiz-Cortés A. Automated analysis of feature models 20 years later: A literature review. Information Systems, 2010,35(6):615−636. [doi: 10.1016/j.is.2010.01.001]

[13] Mendonca M, Wasowski A, Czarnecki K. SAT-Based analysis of feature model is easy. In: Proc. of the 13th Int'l Software Product Line Conf. (SPLC 2009). Pittsburgh: Carnegie Mellon University, 2009. 231−240. http://dl.acm.org/citation.cfm?id=1753267

[14] Bakar NH, Kasirun ZM, Salleh N. Feature extraction approaches from natural language requirements for reuse in software product lines: A systematic literature review. Journal of Systems and Software, 2015,106:132−149. [doi: 10.1016/j.jss.2015.05.006]

[15] Nie KM, Zhang L. A software product line domain requirement model construction method based on model difference and model composition. Chinese Journal of Computers, 2014,37(3):539−550 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2014.00539]

[16] Zitzler E, Kunzli S. Indicator-Based selection in multiobjective search. In: Yao X, Burke EK, *et al*., eds. Proc. of the Parallel Problem Solving from Nature (PPSN VIII). Berlin, Heidelberg: Springer-Verlag, 2004. 832−842. [doi: 10.1007/978-3-540-30217-9_84]

[17] Sayyad AS, Ingram J, Menzies T, Ammar H. Scalable product line configuration: A straw to break the camel's back. In: Proc. of the 28th Int'l Conf. on Automated Software Engineering (ASE 2013). IEEE, 2013. 465−474. [doi: 10.1109/ASE.2013.6693104]

[18] Sayyad AS, Ammar H. Pareto-Optimal search-based software engineering (POSBSE): A literature survey. In: Harrison R, Mernik M, *et al*., eds. Proc. of the 2nd Int'l Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE). IEEE, 2013. 21−27. [doi: 10.1109/RAISE.2013.6615200]

[19] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. on Evolutionary Computation, 2002,6(2):182−197. [doi: 10.1109/4235.996017]

[20] Zitzler E, Laumanns M, Thiele L. SPEA2: Improving the strength Pareto evolutionary algorithm. In: Giannakoglou KC, ed. Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (Eurogen). 2001,3242(103): 95−100.

[21] Durillo JJ, Nebro AJ. jMetal: A framework for multi-objective optimization. Advances in Engineering Software, 2011,42(10): 760−771. [doi: 10.1016/j.advengsoft.2011.05.014]

[22] Durillo JJ, Nebro AJ. jMetal 4.4 User Manual. 2013.

[23] Zitzler E, Thiele L. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. IEEE Trans. on Evolutionary Computation, 1999,3(4):257−271. [doi: 10.1109/4235.797969]

[24] Van Veldhuizen DA, Lamont GB. Multiobjective evolutionary algorithm research: A history and analysis. Technical Report, TR-98-03, Wright-Patterson: Dept. Elec. Comput. Eng., Graduate School of Eng., Air Force Inst. Technol., 1998.

[25] Arcuri A, Briand L. A practical guide for using statistical tests to assess randomized algorithms in software engineering. In: Proc. of the 33rd Int'l Conf. on Software Engineering (ICSE 2011). New York: ACM Press, 2011. 1−10. [doi: 10.1145/1985793.1985795]

[26] Soltani S, Asadi M, Gašević D, Hatala M, Bagheri E. Automated planning for feature model configuration based on functional and non-functional requirements. In: Proc. of the 16th Int'l Software Product Line Conf. (SPLC 2012). New York: ACM Press, 2012. 56−65. [doi: 10.1145/ 2362536.2362548]

[27] Asadi M, Soltani S, Gasevic D, Hatala M, Bagheri E. Toward automated feature model configuration with optimizing non-functional requirements. Information and Software Technology, 2014,56(9):1144−1156. [doi: 10.1016/j.infsof.2014.03.005]

[28] White J, Dougherty B, Schmidt DC. Selecting highly optimal architectural feature sets with filtered cartesian flattening. Journal of Systems and Software, 1999,82(8):1268−1284. [doi: 10.1016/j.jss.2009.02.011]

[29] Berger T, She S, Lotufo R, Wasowski A, Czarnecki K. Variability modeling in the real: A perspective from the operating systems domain. In: Raskin JF, ed. Proc. of the IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE 2010). New York: ACM Press, 2010. 73−82. [doi: 10.1145/1858996.1859010]

附中文参考文献:

[2] Clements P,Northrop L,著;张莉,王雷,译.软件产品线实践与模式.北京:清华大学出版社,2004.

[15] 聂坤明,张莉.基于模型对比和组合的软件产品线领域需求建模.计算机学报,2014,37(3):539−550. [doi: 10.3724/SP.J.1016.2014.00539]

连小利(1985－),女,河北邯郸人,博士生,主要研究领域为软件产品线配置,软件需求,架构设计追踪.

张莉(1968－),女,博士,教授,博士生导师,CCF 高级会员,主要研究领域为软件工程,需求工程.