

对于带约束的多目标搜索问题,最优解往往会分布于边界上;其次,DeIBEA 算法通过替代策略可以加快其收敛速度,而且提高运行效率,因为这样可以降低基于指标进行适应度的复杂计算次数。

```

Alg. DeIBEA().
1  For i from 1 to maxPopulation
2    solution←randomGenerate()
3    evaluate(solution)
4    If evaluateConstraint(solution)==true Then
5      feasibleArchive.add(solution)
6    Else
7      infeasibleArchive.add(solution)
8    End-If
9  End-For
10 For i from 1 to maxGenerationNum
11   For j from 1 to maxPopulation
12     feasibleParent←select(feasibleArchive)
13     infeasibleParent←select(infeasibleArchive)
14     offspring←crossover(feasibleParent,infeasibleParent)
15     offspring←mutation(offspring)
16     If evaluateConstraint(offspring)==true Then
17       If dominateCompare(offspring,feasibleParent)==1 Then
18         feasibleArchive.replace(feasibleParent,offspring)
19       Else If dominateCompare(offspring,feasibleParent)==0 Then
20         feasibleArchive.add(offspring)
21       End-IF
22     Else
23       If dominateCompare(offspring,infeasibleParent)==1 Then
24         infeasibleArchive.replace(infeasibleParent,offspring)
25       Else If dominateCompare(offspring,infeasibleParent)==0 Then
26         infeasibleArchive.add(offspring)
27       End-IF
28     End-IF
29     j++
30   End-For
31   While feasibleArchive.size>maxPopulation/2
32     feasibleArchive.removeWorst()
33   End-While
34   While infeasibleArchive.size>maxPopulation/2
35     infeasibleArchive.removeWorst()
36   End-While
37   i++
38 End-For
39 Return selectBest(feasibleArchive)

```

Fig.4 Pseudo code for DeIBEA algorithm

图 4 DeIBEA 算法伪代码

4 案例研究

本节通过海底油气领域中一个实际的工业案例,对 Zen-Fix 的有效性进行验证.第 4.1 节首先介绍了案例描述,随后分别对实验设计(第 4.2 节)和执行(第 4.3 节)进行了介绍,之后对结果进行分析和讨论(第 4.4 节),最后对结果的有效性风险进行了分析(第 4.5 节).

4.1 案例描述

海底采油控制系统通过海底各种传感器采集数据,并控制海底采油树进行油井生产的管理.海底采油控制系统是一种高度可配置系统,往往存在成百上千的可变点,例如传感器的参数配置、海底采油树上各种阀门的配置以及采油树的部署配置等.可变点之间又存在诸多复杂的依赖约束,如图 1 所示.

本文对 ISO 136286:2006 标准^[19]中海底采油控制系统产品线的主要概念采用 SimPL 建模语言进行描述,之后,又基于该领域的工程手册^[20]对模型进行扩展,得到海底采油控制系统产品线模型.该模型共包含 13 个包、71 个类、111 个可变点(包含 13 个基数可变性、91 个属性可变性、7 个类型可变性)以及 25 条一致性约束(OCL

约束).

4.2 实验设计

4.2.1 研究问题

Zen-Fix 旨在为交互式信息物理系统体系结构产品线配置过程中出现的不一致提供高效且优化的修复方案推荐.Zen-Fix 的有效性首先体现在是否可以自动化地推荐不一致问题的修复方案以及推荐的方案是否较优.此外,由于交互式配置过程对时间性能有较高的要求,因此时间性能优劣也是检验 Zen-Fix 是否可在实际产品配置过程中得到成功应用的一个重要指标.为了验证 Zen-Fix 与 DelBEA 算法的有效性,本文定义了如下两个研究问题:

RQ1:本文提出的 DelBEA 算法是否比原始的 IBEA 算法具有更高的效率,即,花费更少的时间?

RQ2:本文提出的 DelBEA 算法是否比原始的 IBEA 算法具有更高的搜索性能,即,可以产生更优的解?

4.2.2 实验设计

如第 1.1 节所述,信息物理系统产品线模型共包含 4 类可变点,即:基数可变性、类型可变性、属性可变性以及拓扑可变性.基于不同的可变点类型,具有两种不同的配置形式,其中,类型可变性、拓扑可变性以及属性可变性中的枚举类型都是通过从预定义的值域列表中进行选择(选择型)的,而基数可变性以及其他的属性可变性需要用户(或者配置工具)为参数进行赋值(参数型).

为了在实际海底采油控制系统产品线配置中评估对比不同的算法在解决一致性修复问题中的优劣,本文模拟一个完整的产品配置过程,并在配置过程中控制不一致问题的产生,保证配置过程的可重复性,即在特定的配置步骤可以反复产生相同的一致性修复问题.本文实验设计包含以下几个阶段.

- 1) 定义配置过程:针对某个特定产品的海底采油系统的配置数据,模拟一个完整的配置过程,该过程包含一系列配置步骤,每个配置步骤是一个三元组,即(配置步骤,可变点实例,配置数据),相应于对一个可变点实例赋值.如, $(10, root::XmasTree[2]::sensors, 9)$ 表示在配置步骤 10 将可变点实例 $root::XmasTree[2]::sensors$ 的值配置为 9;
- 2) 定义不一致性的植入位置(配置步骤):对于一条一致性约束 CR_j ,与其相关的可变点实例集合可以通过每个可变点实例相应的配置步骤的集合表示,记作 $\{CS_1, \dots, CS_n\}$,其中, i 表示相应的配置步骤.本文将相关配置步骤的最大值 n 作为 CR_j 的不一致性植入位置.如图 5 所示,对于第 1 个一致性约束 CR_1 ,其相关的配置步骤为 $\{CS_1, CS_6\}$ (图中灰色单元格所示),则第 6 个配置步骤 CS_6 为 CR_1 的不一致性植入位置(后文简称植入位置,图中红色对勾所示);
- 3) 定义边界值:针对每条一致性约束中关联的可变点实例的类型,基于边界值等价划分标准为每个可变点实例定义边界值.比如对于参数型的可变点,如基数可变性“1...*”,对其进行边界值等价划分可以得到两个分区:一个是 ≤ 1 ,另一个是 ≥ 1 .对于第 1 个分区,考虑到基数可变性只能取非负整数,该分区内只有一个边界值为 0;对于第 2 个分区,采用 1,5,10,20 作为边界值.对于选择型的可变点,即类型、拓扑以及枚举型的属性可变性,所有的选择项都定义为其边界值;
- 4) 定义配置数据:为了更全面、更系统地验证 Zen-Fix 对不一致问题的修复能力,对于每一个植入位置,本文基于步骤 3)定义的边界值考虑了相关已配置可变点实例配置数据的所有可能组合.鉴于一致性约束之间的相互关联性,对于每一个一致性约束 CR_j 的植入位置,不仅需要考虑与 CR_j 相关的已配置可变点实例的配置数据,因为这样可能会不可控地违背其他的一致性约束,因此还需要考虑与 CR_j 相关的所有一致性约束,即 $Closure(CR_j)$ (定义见第 2.1 节).如图 5 所示,对于约束 CR_1 ,其植入位置为 CS_6 ,如图中红色对勾所示.在配置步骤 6,需要考虑 $Closure(CR_1)$ 中的所有约束,即 $\{CR_1, CR_2, CR_3, CR_5\}$ (配置步骤第 6 列所有对勾所对应的一致性约束集合).本文的配置数据定义为与 $Closure(CR_j)$ 相关的已配置可变点实例边界值的所有组合,即,需要考虑与配置步骤 $\{CR_1, CR_2, CR_3, CR_5\}$ 相应的 4 个可变点实例边界值的所有组合.

在每个植入位置,会遍历所有已配置的相关可变点实例的边界值组合进行一致性约束的检验,每个会产生

不一致的组合即构成一个不一致修复问题.通过以上步骤,本文共获得 10 189 个不一致修复问题.基于这 10 189 个修复问题,本文采用 IBEA 和 DeIBEA 各运行 30 次以降低搜索算法中随机带来的影响.对于运行次数的选取在文献[21]中已有充分讨论,本文选择运行 30 次是综合考虑整体运行时间和统计分析有效性而决定的,也是基于搜索的软件工程中经常采用的实验设置^[21].

		配置步骤						
		CS ₁	CS ₂	CS ₃	CS ₄	CS ₅	CS ₆	CS ₇
一致性约束	CR ₁			✓		✓	✓	✓
	CR ₂			✓		✓	✓	✓
	CR ₃			✓		✓	✓	✓
	CR ₄				✓			
	CR ₅					✓	✓	✓

Fig.5 Strategies to generate nonconformities

图 5 不一致性问题生成策略

4.2.3 度量指标

针对第 1 个研究问题,本文定义了一个度量指标,即,每个算法修复不一致问题 i 所需要的平均时间 $Time_i$:

$$Time_i = \sum_{j=1}^{30} Time_{ij} / 30,$$

其中, $Time_{ij}$ 表示第 j 次修复不一致问题 i 所花费的时间,其中, $j \leq 30$ 而且 $i \leq 10189$.

如第 3.1 节所示,对于一致性修复所构成的多目标搜索问题,本文共定义了 4 个启发式优化目标,即: F_{NUM} , F_{IMPACT} , F_{CC} , F_{INFER} . 为了比较不同算法的搜索能力,基于这 4 个优化目标,本文为第 2 个研究问题定义了以下 7 个度量指标:

- 1) $F'_{NUM_{ij}}$: 表示每种算法第 j 次修复不一致问题 i 时所得到的多个解中的 F_{NUM} 最小值;
 - 2) $F'_{IMPACT_{ij}}$: 表示每种算法第 j 次修复不一致问题 i 时所得到的多个解中的最小的 F_{IMPACT} 值;
 - 3) $F'_{CC_{ij}}$: 表示每种算法第 j 次修复不一致问题 i 时所得到的多个解中最小的 F_{CC} 值;
 - 4) $F'_{INFER_{ij}}$: 表示每种算法第 j 次修复不一致问题 i 时所得到的多个解中 F_{INFER} 的最小值;
- 这 4 个指标能够对比不同算法在单个优化方向上的搜索能力.
- 5) $F'_{OVERALL_{ij}}$: 表示每种算法第 j 次修复不一致问题 i 时所得到的多个解中 4 个优化目标平均值的最小值, 该指标度量了算法在各个方向上的平均搜索能力;
 - 6) HV_{ij} : 表示每种算法第 j 次修复不一致问题 i 时所得解集的超体积(hypervolume), 其中, 超体积指标 HV ^[22] 表示算法所获得的 Pareto 前端在目标域中覆盖的体积, HV 越大, 表明 Pareto 解集在真实前端覆盖的范围越大, 算法具有较好的扩展性和分布性;
 - 7) $EPSILON$ ^[22] 指标用来表示算法的收敛性能, 该指标越小, 说明算法的收敛速度越快, $EPSILON_{ij}$ 表示算法第 j 次修复不一致问题 i 时的收敛性.

4.2.4 统计分析

为了对比分析两种不同算法 DeIBEA 与 IBEA, 本文采用 Wilcoxon 符号秩检验^[23]和 Vargha 与 Delaney 统计^[24]来对度量指标进行分析. 通过 Wilcoxon 符号秩检验来获得以 0.05 为显著性水平的 p 值, 代表两种算法差异的显著性. Vargha 与 Delaney 统计($\hat{A}12$)是一种非参数效应量(effect size)度量方法. 在本文中, $\hat{A}12$ 用于表示对于第 4.2.3 节中定义的某一个度量指标: $\hat{A}12$ 小于 0.5, 表示算法 IBEA 比 DeIBEA 具有更高的概率获得较大的指标值; $\hat{A}12$ 等于 0.5, 表示两种算法是相当的; $\hat{A}12$ 大于 0.5, 表示算法 DeIBEA 比 IBEA 具有更高的概率获得较大的指标值.

4.3 执行

本文选用 jMetal 中的 IBEA 算法与本文设计的 DeIBEA 算法进行对比. 通过模拟真实案例下的整个配置过

程,Zen-Fix 分别采用 DeIBEA 与 IBEA 两种算法对整个配置过程中植入的 10 189 个不一致问题进行修复,针对两种算法对整个配置过程分别模拟 30 次,并记录 Zen-Fix 在修复每一个问题时所花费的时间.IBEA 算法采用 jMetal 的默认配置,DeIBEA 算法采用与 IBEA 类似的设置,具体的算法设置参数见表 1.

Table 1 Parameterization for IBEA and DeIBEA

表 1 算法 IBEA 与 DeIBEA 参数设置表

	IBEA	DeIBEA
Population size	100	100
Archive size	100	50(infeasible)+50(feasible)
Generation	100	100
Selection of patterns	binary tournament+binary tournament	binary tournament+binary tournament
Mutation	polynomial, mutation rate=1/n	polynomial, mutation rate=1/n
Crossover	simulated binary, crossover rate=0.9	Differential evolution, crossover rate=0.9

n is the number of variables

4.4 结果和分析

为了回答以上两个研究问题,本文对第 4.2.3 节中定义的度量指标进行 Wilcoxon 秩检验和 Vargha 与 Delaney 统计,结果见表 2.以时间 Time 为例,对于问题 i ,DeIBEA 算法和 IBEA 算法各自运行 30 次分别得到 30 个数据 $Time_{ij}(j \leq 30)$.对两种算法的两组 $Time_{ij}$ 值进行 Wilcoxon 秩检验和 Vargha 与 Delaney 统计,根据 $\hat{A}12$ 与 p 值的大小,将问题分为 3 大类、6 小类.在此需要说明的是:在这些度量指标中, HV 的值越大越好;对于其他度量指标而言,值越小越好.表 2 已经对这种不同进行了处理,其中,DeIBEA>IBEA 对应的列表示对于每一个度量指标,DeIBEA 与 IBEA 相比性能较优的情况,每个单元格斜线后面的数据表示 DeIBEA 比 IBEA 性能较优的问题个数(以 Time 为例,表示 $\hat{A}12 < 0.5$),而相应单元格中斜线前的数据表示 DeIBEA 显著比 IBEA 较优的问题个数($\hat{A}12 < 0.5$ 且 $p < 0.05$).DeIBEA<IBEA 列类似,表示 DeIBEA 与 IBEA 相比性能较差的情况.对于 DeIBEA=IBEA 这一列来说,斜线前的数据表示 DeIBEA 与 IBEA 相比没有显著性差异的问题个数($p \geq 0.05$),而斜线后面的数据记录了 DeIBEA 与 IBEA 表现相同的问题个数($\hat{A}12 = 0.5$).

Table 2 Results of the Wilcoxon signed-rank test and the Vargha and Delaney statistics

表 2 Wilcoxon 秩检验和 Vargha 与 Delaney 分析结果 F'_{IMPACT}

Indicator	DeIBEA>IBEA	DeIBEA<IBEA	DeIBEA=IBEA
Time	9244/10170	0/14	945/5
F'_{NUM}	2552/5228	764/3960	6873/1001
F'_{IMPACT}	2518/5228	765/3960	6906/1001
F'_{CC}	0/651	7/4899	10181/4639
F'_{INFER}	9081/10175	0/0	1108/14
$F'_{OVERALL}$	2548/5263	762/3959	6879/967
HV	1858/9635	13/299	8318/255
$EPSILON$	5660/9574	8/455	4521/160

RQ1:如表 2 第 1 行所示:对于产生的 10 189 个不一致修复推荐问题,有 10 170 个问题 DeIBEA 算法消耗的时间小于 IBEA,其中,对 9 244 个问题的修复 DeIBEA 所用的时间显著小于 IBEA 算法;仅仅有 14 个问题 IBEA 消耗时间小于 DeIBEA,但是它们之间的差异并不显著;共有 945 个不一致问题在修复的时候,DeIBEA 与 IBEA 的时间性能没有显著性差异,其中有 5 个问题 DeIBEA 与 IBEA 所用时间完全一样.此外,本文还对每种算法对问题 i 的平均时间 $Time_i$ 进行了统计.如表 3 所示:对于所有的 10 189 个问题,DeIBEA 运行时的最大平均时间是 4 668ms,最小平均时间是 614ms,平均平均时间是 3 263ms;而 IBEA 分别为 6 371,1 529 和 4 664ms.在交互式产品线配置过程中,系统的响应时间是决定其实用性的重要因素.因此,修复不一致问题所需要的时间是决定 Zen-Fix 成功与否的关键因素.通过以上分析结果可知,本文提出的改进算法 DeIBEA 较原来的 IBEA 算法具有更高的时间性能,可以更大程度地提高 Zen-Fix 进行不一致修复的效率.

Table 3 Time performance for DeIBEA and IBEA (ms)**表 3** 算法 DeIBEA 与 IBEA 时间性能(毫秒)

$Time_i$	Max	Min	Average
DeIBEA	4 688	614	3 263
IBEA	6 371	1 529	4 664

RQ2:为了对比两种算法的搜索性能,本文对另外 7 个度量指标进行 Wilcoxon 秩检验和 Vargha 与 Delaney 统计,统计分析的方法与上文对时间指标 Time 的方法类似,分析结果见表 3.对于前两个度量指标 F'_{NUM} 与 F'_{IMPACT} ,分析结果比较类似.对于 F'_{NUM} (F'_{IMPACT}) 来说,10 189 个问题中:5 228(5 228)个问题的搜索结果是 DeIBEA 优于 IBEA,其中,2 552(2 518)个问题下,DeIBEA 显著占优;3 960(3 960)个问题的搜索结果是 IBEA 优于 DeIBEA,但是只有 764(765)个问题是 IBEA 显著占优的.对两种算法来说,没有显著性差异的问题个数为 6 873(6 906),其中,1 001(1 001)个问题两种算法的搜索结果完全相同.以上数据可以说明:针对前两个优化目标,即最小化修复的可变点实例与最小化修复对未配置可变点的影响,DeIBEA 算法较之 IBEA 算法在更多的问题上具有更好的搜索能力,可以搜索到更优的解.

对于第 3 个度量指标 F'_{CC} ,有 10 181 个一致性修复问题,两种算法的搜索结果没有显著性差异.这可能是由于,在此优化方向上,解空间中各个解本身对约束验证代价的影响的差异性较小造成的.对于第 4 个度量指标 F'_{INFER} ,10 189 个问题中有 10 175 个问题,DeIBEA 的搜索结果比 IBEA 要更优,对于其中 9 081 个问题,DeIBEA 具有显著性的优势.在第 4 个优化方向上,DeIBEA 远远优于 IBEA 的搜索性能.

对于第 5 个度量指标 $F'_{OVERALL}$,体现了搜索算法在各个方向上的平均搜索性能.由表 3 可以看出:有将近一半(5 263)的问题,DeIBEA 优于 IBEA 算法,即,可以得到更小的平均适应度值.其中,在 2 548 个问题上,DeIBEA 显著优于 IBEA 算法.而 IBEA 算法只在 762 个问题上显著优于 DeIBEA.总体上来说,针对平均搜索性能,DeIBEA 稍微优于 IBEA 算法.

对于第 6 个度量指标超体积 HV,有 9 635 个问题 DeIBEA 优于 IBEA.其中,1 858 个问题中,DeIBEA 显著占优;而 IBEA 仅仅在 13 个问题上显著优于 DeIBEA,远远小于其显著劣于 DeIBEA 的问题个数(1 858).分析结果说明,DeIBEA 算法较之 IBEA 算法具有更好的扩展性和分布性.

对于评估收敛速度的度量指标 EPSILON,对于 9 574 个问题,DeIBEA 的收敛速度快于 IBEA,其中,有 5 660 个问题 DeIBEA 的收敛速度显著快于 IBEA;而 IBEA 的收敛速度只在 455 个问题上优于 DeIBEA,其中只有 8 个问题是显著占优.通过以上分析很容易得出结论:DeIBEA 的收敛速度与 IBEA 相比具有明显的优势.

通过以上实验分析,我们可以得出以下结论:

- 1) 本文提出的 DeIBEA 算法比原始的 IBEA 算法具有更好的效率;
- 2) 本文提出的 DeIBEA 算法比原始的 IBEA 算法具有更好的搜索性能.

4.5 有效性风险

可能的内部有效性风险在于本文对两种算法的参数设置.为了降低这种风险的可能性,本文对传统的 IBEA 算法采用默认的配置,并对 DeIBEA 算法采用相同的参数配置.外部有效性风险主要存在于对实验结果的普适化.本文通过对一个工业案例配置过程进行模拟,生成了 10 189 个不一致性修复问题,因此,本文的实验结果可以在一定程度上进行泛化.但是,为了更进一步地验证本文结果的普适性,需要更大规模的案例以及更多的修复问题.

5 结 论

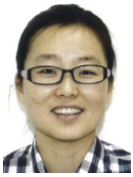
在信息物理系统产品线的交互配置过程中,人工的配置难免引入配置的不一致性,即,配置违背了预定义的约束.而信息物理系统本身的复杂性和约束密集性,为不一致修复带来了很大的挑战.本文将不一致修复问题转化为带约束的多目标搜索问题:首先,面向配置效率定义了 4 个优化目标以启发搜索过程;并基于前期增量验证的工作实现了增量约束(OCL)违背距离求解.通过多目标搜索算法,为用户推荐较优的修复方案.

IBEA 算法在产品线配置(多为特征选择)中已经得到成功运用,但本文工作的不同之处在于:首先,本文面向的是交互式产品线配置,为不一致修复的时间性能带来了很高的需求,IBEA 算法由于其本身运算的复杂度导致时间性能不高;其次,信息物理系统产品线中约束较为复杂,IBEA 算法中采用的占优方式不能很好地利用不可行解,从而会影响算法的搜索性能.基于以上两点,本文通过将差分进化算法引入到 IBEA 算法中,并区分可行解与不可行解,提出了改进的 DeIBEA 算法,通过工业案例的实验,其结果表明,DeIBEA 算法无论从时间性能还是搜索性能上都优于传统的 IBEA 算法.

References:

- [1] Rajkumar RR, Lee I, Sha L, Stankovic J. Cyber-Physical systems: The next computing revolution. In: *Cyber-Physical Systems: The Next Computing Revolution*. 2010. 731–736. <http://dl.acm.org/citation.cfm?id=1837461>
- [2] Cordy M, Schobbens PY, Heymans P, Legay A. Towards an incremental automata-based approach for software product-line model checking. In: *Proc. of the 16th Int'l Software Product Line Conf.*, Vol.22012. 2005. 74–81. [doi: 10.1145/2364412.2364425]
- [3] Hubaux A, Xiong Y, Czarnecki K. A user survey of configuration challenges in Linux and eCos. In: *Proc. of the 6th Int'l Workshop on Variability Modeling of Software-Intensive Systems* 2012. 2012. 149–155. [doi: 10.1145/2110147.2110164]
- [4] Henard C, Papadakis M, Harman M, Le Traon Y. Combining multi-objective search and constraint solving for configuring large software product lines. In: *Proc. of the ICSE 2015*. 2015. <http://dl.acm.org/citation.cfm?id=2818819>
- [5] Zitzler E, Künzli S. Indicator-Based selection in multiobjective search. In: *Indicator-Based Selection in Multiobjective Search*. 2004. 832–842. [doi: 10.1007/978-3-540-30217-9_84]
- [6] Behjati R, Yue T, Briand L, Selic B. SimPL: A product-line modeling methodology for families of integrated control systems. *Information and Software Technology*, 2013,55(3):607–629. [doi: 10.1016/j.infsof.2012.09.006]
- [7] OMG. OCL 2.0 Specification. 2005.
- [8] Sayyad AS, Ingram J, Menzies T, Ammar H. Scalable product line configuration: A straw to break the camel's back. In: *Scalable Product Line Configuration: A Straw to Break the Camel's Back*. 2013. 465–474. [doi: 10.1109/ASE.2013.6693104]
- [9] Storn R, Price K. Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 1997,11(4):341–359. [doi: 10.1023/A:1008202821328]
- [10] OMG. UML 2.2 Superstructure Specification (formal/2009-02-04). 2009.
- [11] Stoiber R, Glinz M. Supporting stepwise, incremental product derivation in product line requirements engineering. *System*, 2010,1:2.
- [12] Hong L, Tao Y, Ali S, Kunming N, Li Z. Zen-CC: An automated and incremental conformance checking solution to support interactive product configuration. In: *Proc. of the IEEE 25th Int'l Symp. on Software Reliability Engineering (ISSRE)*. 2014. 13–22. [doi: 10.1109/ISSRE.2014.13]
- [13] Hong L, Yue T, Ali S, Li Z. Integrating search and constraint solving for nonconformity resolving recommendations of system product line configuration. In: *Proc. of the IEEE Int'l Conf. on Software Testing, Verification and Validation (ICST)*. 2016.
- [14] Ali S, Iqbal MZ, Arcuri A, Briand L. Generating test data from OCL constraints with search techniques. *IEEE Trans. on Software Engineering*, 2013,39(10):1376–1402. [doi: 10.1109/TSE.2013.17]
- [15] Arcuri A. It really does matter how you normalize the branch distance in search-based software testing. *Software Testing, Verification and Reliability*, 2013,23(2):119–147. [doi: 10.1002/stvr.457]
- [16] Nie K, Yue T, Ali S, Zhang L, Fan Z. Constraints: The core of supporting automated product configuration of cyber-physical systems. In: *Constraints: The Core of Supporting Automated Product Configuration of Cyber-Physical Systems*. 2013. [doi: 10.1007/978-3-642-41533-3_23]
- [17] McMinn P. Search-Based software test data generation: A survey. *Software Testing Verification & Reliability*, 2004,14(2): 105–156. [doi: 10.1002/stvr.294]
- [18] Hsieh MN, Chiang TC, Fu LC. A hybrid constraint handling mechanism with differential evolution for constrained multiobjective optimization. In: *A Hybrid Constraint Handling Mechanism with Differential Evolution for Constrained Multiobjective Optimization*. 2011. 1785–1792. [doi: 10.1109/CEC.2011.5949831]

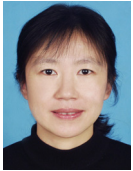
- [19] ISO 13628-6: 2006. Petroleum and natural gas industries-design and operation of subsea production systems. Standard, Part 6: Subsea Production Control Systems. 2006.
- [20] Bai Y, Bai Q. Subsea Engineering Handbook. Gulf Professional Publishing, 2010. https://books.google.com/books/about/Subsea_Engineering_Handbook.html?id=ZkkPos9OKDcC
- [21] 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. Waikiki, 2011. 1–10. [doi: 10.1145/1985793.1985795]
- [22] Durillo JJ, Nebro AJ. jMetal: A Java framework for multi-objective optimization. Advances in Engineering Software, 2011,42(10): 760–771. [doi: 10.1016/j.advengsoft.2011.05.014]
- [23] Wilcoxon F. Individual Comparisons by Ranking Methods. Springer-Verlag, 1992. 196–202. [doi: 10.1007/978-1-4612-4380-9_16]
- [24] Vargha A, Delaney HD. A critique and improvement of the CL common language effect size statistics of McGraw and Wong. Journal of Educational and Behavioral Statistics, 2000,25(2):101–132. [doi: 10.3102/10769986025002101]



路红(1986—),女,河北石家庄人,博士生,主要研究领域为产品线建模与配置,一致性检查与修复.



岳涛(1974—),女,博士,高级研究员,博士生导师,主要研究领域为需求工程,基于模型的产品线工程,基于模型的测试,基于搜索的软件工程.



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体系结构和产品线,需求建模,模型驱动软件工程.