

基于自适应进化算法的软件产品线抽样方法

欧阳丹彤^{1,2}, 袁哲^{1,2}, 张立明^{1,2}

(1. 吉林大学 计算机科学与技术学院, 长春 130012; 2. 吉林大学 符号计算与知识工程教育部重点实验室, 长春 130012)

摘要: 为了进一步提高样本集的多样性, 提出了 DDHNSbS 算法, 其中多样化变异策略 (DMS) 不仅能保留初始种群的多样性, 还能优化传统遗传算子的效果。同时, 为弥补新颖性搜索的局限性, 提出了多样性检测 (DD) 以一定概率对特征的赋值进行检验并翻转。此外, 利用 Halton 数列实现概率感知多样化 (Halton-PaD), 提高初始配置的均匀性。实验在 39 个公共软件产品线 (SPL) 上进行, 结果表明, 与当前性能最优的多样化抽样方法 PaD-NSbS 算法相比, DDHNSbS 在 42.1% 的小型 SPL 上求解性能显著更优, 在其他 SPL 上性能相当。

关键词: 软件产品线; 多样化抽样; 新颖性搜索; 概率感知多样化; 进化算法

中图分类号: TP311 **文献标志码:** A **文章编号:** 1671-5497(2025)09-3007-13

DOI: 10.13229/j.cnki.jdxbgxb.20240344

Sampling method of software product line based on adaptive evolutionary algorithm

OUYANG Dan-tong^{1,2}, YUAN Zhe^{1,2}, ZHANG Li-ming^{1,2}

(1. College of Computer Science and Technology, Jilin University, Changchun 130012, China; 2. Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China)

Abstract: In order to further improve the diversity of the sample set, the DDHNSbS algorithm was proposed, in which the Diversity Mutation Strategy (DMS) not only preserves the diversity of the initial population, but also optimizes the effect of traditional genetic operators. Meanwhile, to make up for the limitations of novelty search, diversity detection (DD) was proposed to check and flip the assigned values of features with a certain probability. So as to further diversify the number of features selected in the sample set, probability-aware diversification via the Halton sequence (Halton-PaD) was achieved, and the generated initial configurations' uniformity was developed thanks to Halton-PaD. Experiments were carried out on 39 public SPLs, and the experimental results show that when DDHNSbS algorithm is compared with PaD-NSbS algorithm, DDHNSbS performs significantly better in 42.1% of the small SPLs, and has the same performance on other SPLs.

Key words: software product lines; diverse sampling; novelty search; probability-aware diversification; evolutionary algorithm

收稿日期: 2024-04-01.

基金项目: 国家自然科学基金项目 (62076108, 61872159); 吉林省教育厅项目 (JKH20211106, JKH20211103KJ).

作者简介: 欧阳丹彤 (1968-), 女, 教授, 博士. 研究方向: 人工智能、模型诊断. E-mail: ouyangdantong@163.com

0 引言

软件产品线(Software product line, SPL)^[1]能降低软件成本,提高可重用性^[2],它根据所选特征及所选特征数量生成多种配置,满足特征间所有约束的配置被称为有效配置。大型真实世界的 SPL 特征数量庞大,面对软件工程任务(例如例如结合多目标进化算法思想^[3,4],根据用户所需获得最优配置^[2]、测试 SPL^[5,6]等)时,考虑到生成配置的不确定性和有限的资源,枚举有效配置非常困难^[7],因此需要高效的抽样方法选取样本反映总体。

目前,抽样方法大致分为 5 种:基于求解器的抽样短时间内通过求解器生成大量样本,但不能保证样本有效性^[8-10];随机抽样随机选择特征生成样本集^[11-13];面向覆盖的抽样通过特定的覆盖标准选取样本^[14-16];均匀抽样在搜索空间中统一采样^[17,18];多样性抽样^[19-26]在 SPL 抽样中占有优势,它不仅保证导出性能预测模型时尽可能保留不同行为的配置^[19],还可以替代覆盖抽样^[20]。目前,已有多种多样性抽样方法被提出并被投入实际应用。DDbS(Diversified distance-based sampling)算法^[19]根据所选特征数量选取样本,它表明该数量越多样化,样本集就越有代表性。多目标 SPL 配置问题^[21-24]也将所选特征数量作为优化目标之一,因此提高样本有关该数量的多样性至关重要。NSbS(Novelty search-based sampling)算法^[25]在此基础上提出行为空间的概念,结合新颖性搜索在行为空间和配置空间进行探索,生成具有多样性的样本集,同时提出分散值作为实验指标衡量算法性能。PaD-NSbS(Probability-aware diversification-NSbS)算法^[26]是目前最新的方法,它根据 PaD 的特点改进了 SAT(Satisfiability)求解器,在生成初始种群的过程中尽可能减少取消赋值的变量数量,从而在行为空间达到更好的覆盖。

然而,最新的 PaD-NSbS 算法仍有不足,为了保留该算法中初始种群的多样性并进行有效搜索,提高样本集的多样性,本文提出 DDHNSbS 算法,主要贡献如下:

(1)多样化变异策略(Diversity mutation strategy, DMS)保留新颖性分数高的个体并以随机生成的新配置替换分数低的个体,弥补了原算法过

早收敛的不足;

(2)多样性检测(Diversity detection, DD)以一定概率进行并防止父代与子代同时选择或取消选择同一特征,避免原算法局部搜索不足,出现个体早熟的情况;

(3)以 Halton 序列^[27]为基础的概率感知多样性(Halton-PaD),其生成的初始种群均匀性得到进一步提高;

(4)本文选择了不同的遗传算子进行实验,判断出最高效的组合。

1 预备知识

本节将介绍从配置空间采样所需的基础知识,包括从配置空间采样的方法以及行为空间的基本概念。在多样性抽样中,通过配置选择特征的数量描述配置的行为,该数量越多样化,性能预测的准确性就越高^[25]。为了清晰表述各配置的行为并计算抽样方法的性能指标,文献[25]提出了行为空间的概念,并说明如何将配置空间映射到行为空间。

1.1 从配置空间采样

与其他问题相同, SPL 在特征之间存在约束,为直观表述约束和特征的关系,需要对 SPL 进行建模^[28]。特征模型(Feature model, FM)^[29]可以清晰地表示出 SPL 中的特征及其约束等信息,该模型采用树状结构,其具体定义如下。

定义 1 FM 用二元组 $\langle F, C \rangle$ 表示^[25], 其中 $F = \{f_1, f_2, \dots, f_n\}$ 为 n 个特征的集合, f_i 为 SPL 的第 i 个特征, C 为各特征间约束关系的集合。

定义 2 配置 c 由集合 $\{\pm f_1, \pm f_2, \dots, \pm f_n\}$ 表示, 其中 $+f_i$ 为选择 FM 中的第 i 个特征, $-f_i$ 为取消选择的第 i 个特征。若用二进制串表示配置 c , 则选择某特征, 该特征位取值为 1, 否则取 0。其中, 满足所有约束的配置被称为有效配置, 否则被称为无效配置。

定义 3 配置空间 Ψ 是所有有效配置的集合, 它的基数由 $|\Psi|$ 表示。

为完成特定的任务目标, 通常需要对配置空间中的有效配置进行探索。然而大型的软件产品线的特征数十分庞大, 导致 $|\Psi|$ 也随之增长。在枚举配置空间并不现实的情况下, 需要选取配置空间的子集作为样本集, 通过样本集对配置空间进

行探索。

定义4 样本集是一个集合 $S = \{s_1, s_2, \dots, s_n\}$, 其中 s_i 为一个配置空间中的第 i 个有效配置, 它是样本集的样本之一。

1.2 行为空间

如前所述, 配置的行为可以通过其所选的特征数量进行表述, 所选特征数量的多样性将提高最终样本集的代表性。将配置空间 Ψ 映射到行为空间之后, 算法会对均匀覆盖的配置抽样, 使样本集多样性提高。

定义5 若以每个有效配置所选择的特征数量为基准表示配置的行为, 则配置空间 Ψ 会被映射到由正整数组成的空间, 即行为空间^[19, 25]。行为空间用 B 表示, 它的基数由 $|B|$ 表示。

强制特征是指有效配置中一定选择的特征, 而无效特征是指有效配置中一定取消选择的特征。假设共有 n 个特征, 则 B 集合中的最大值和最小值可以由强制特征的数量 $|core|$ 以及无效特征的数量 $|dead|$ 确定。因为每个有效配置必须选择强制特征, 但不能选择无效特征, 所以 B 中的最小值 $\min(B) \geq |core|$, 最大值 $\max(B) \leq n - |dead|$ 。显然, B 中最多有 $n - |dead| - |core| + 1$ 个整数, 可以理解为配置空间 Ψ 被分为 $|B|$ 个子空间, 算法可以对这些子空间中不同分布的配置进行抽样。 $|B|$ 远小于 $|\Psi|$, 因此行为空间更好理解。然而由于配置的数量繁多, 很难准确地了解行为空间, 可以通过集合 B' 近似表示行为空间 B 。 $B' = \{|core|, |core| + 1, \dots, n - |dead|\}$, 它包括 $|core|$ 至 $n - |dead|$ 间的全部整数, 考虑到 B' 中的整数可能无法与 Ψ 中有效配置选择的特征数量形成映射, 因此 B 是 B' 的子集。 B' 还可以用来计算性能指标。

2 DDHNSbS 算法及其优化策略

本节将介绍本文提出的 DDHNSbS 算法及其 3 种优化策略, 分别是 DMS、DD 以及 Halton-PaD。

2.1 DDHNSbS 算法

DDHNSbS 包括 3 个优化策略, 它的算法流程如算法 1 所示。根据输入的由 FM 生成的命题公式以及用户所需的样本集大小, 算法不断在配置空间中探索新的个体, 从而提高最终样本集的

多样性, 在达到终止条件后将最终的样本集存储在档案 A 中并输出。

算法 1 DDHNSbS 算法

输入: 命题公式 φ , 样本大小 N ;

输出: 档案 A 。

① $A \leftarrow \text{InitializeArchiveWithHalton}(N)$; /* φ 通过 Halton-PaD 生成初始种群 S_{init} 并存储 */

② $D \leftarrow \text{InitializeDistanceMatrix}(N+1)$; /* 初始化距离矩阵 */

③ calculate the novelty score $\rho(x)$ for each $x \in A$;

④ while runtime $\leq$ maxtime & gap ≤ 10

⑤ $\{c_1, c_2\} \leftarrow S_{\text{init}}$ after traditional genetic operators with DMS; /* 用 DMS 结合遗传算子生成新子代 */

⑥ for $i \in \{1, 2\}$

⑦ if $p < \theta$

⑧ DD; /* 进行多样性检测 */

⑨ end if

⑩ end for

⑪ apply ProbSAT solver to repair c_1', c_2' ; /* ProbSAT 求解器修复子代 c_1', c_2' */

⑫ updateArchive(A); /* 新颖性搜索更新样本集 */

⑬ end while

⑭ return A

算法先通过随机抽样以及 rSAT4J 求解器生成初始种群, 再进行后续操作。有效的样本集需要均匀而广泛地抽取, PaD-NSbS 算法中使用 PaD 生成样本, 目的是使特征数量尽可能在 $[0, n]$ 之间分布; 而 Halton-PaD 不仅可以满足该条件, 还能保证取值更加均匀。因此, DDHNSbS 算法中, 首先按照所给样本数量以及所给命题公式 φ , 通过 Halton-PaD 以及 rSAT4J 求解器随机生成 N 个配置, 作为初始种群 S_{init} 并将其存储在档案 A 中(第 1 行)。针对档案 A 中的各有效配置, 算法运用距离公式计算任意 2 个不同配置之间的距离并存入距离矩阵(第 2 行)。其次, 算法依次求出 S_{init} 中各个体的新颖性分数, 为新颖性搜索做准备(第 3 行)。进入探索过程后, PaD-NSbS 以及 NSbS 算法直接使用传统遗传算子执行进化算法; 在迭代期间, 以新颖性分数为标准, 算法保留分数高的个体并淘汰分数低的个体, 然而这样会减弱 PaD 的效果, 影响种群的多样性。为同时保留 PaD 和遗传算子的效果, DDHNSbS 算法使用传统遗传算子与 DMS 相结合的方式产生新的子代(第 5 行)。产生新的子代后, 为了在现有算法的基础上进一步提高样本的多样性, 新的子代需要进行 DD, 而是否进行取决于概率 θ , 若随机

产生的概率 p 小于 θ , 则需进行(第 8 行), 并得到多样化后的子代; 否则不对子代进行修改, 用 c_1' 、 c_2' 表示此时的子代。因为 c_1' 、 c_2' 不能保证其有效性, 所以算法使用 ProbSAT 求解器对无效解进行修复(第 9 行)。最后, 根据新颖性分数更新档案(第 10 行), 直到达到终止条件, 返回档案 A 作为多样化的样本集输出。

算法中计算距离的公式分为 2 部分, 分别在行为空间和配置空间衡量配置的相似性, 从而保证所探索的个体不仅在行为空间具有独特性, 在配置空间也能尽可能保持配置间的不同。新颖性分数的计算依赖于各配置间的距离, 配置相距越远, 说明配置分布越稀疏。处于稀疏区域的配置有助于提高样本集的多样性, 因此该类型的个体会被赋予较高的新颖性分数, 在新颖性搜索时就会更趋于探索分数高的个体^[25]。

算法 DDHNSbS 中的优化策略 DMS、DD 以及 Halton-PaD 将在后续介绍。

2.2 DMS 方法

虽然原算法中交叉、变异等传统的遗传算子可以对样本集进行广泛搜索从而保证其多样性, 但通过概率感知生成的初始解的随机效果会全部被遗传算子取代, 因此初始种群的多样性受到一定影响。为了同时保留两者的效果并且扩大搜索范围, 本文提出了 DMS 方法。

DMS 的具体流程如下: 当从父代选择出 2 个个体后, 同时使用 DMS 以及传统的遗传算子生成新的子代。DMS 首先以种群的新颖性分数平均值作为标准值, 在搜索过程中, 新颖性分数低于标准值的个体将被随机生成的新个体取代; 相对地, 新颖性分数高于标准值的个体会进行遗传操作并进入下一代。基于此, 种群不仅能淘汰适应度低的个体, 还能确保种群的多样性。

算法 2 DMS 方法

输入: 被选择的父代 p_1, p_2 ;

输出: 新生成的子代 c_1, c_2 。

① $\rho_{\text{better}} \leftarrow \max(\rho_1, \rho_2)$; /* 确定被选择父代中较高的新颖性分数 */

② $\text{avg_score} \leftarrow$ find the average novelty score of all samples;

③ if $\rho_{\text{better}} < \text{avg_score}$

④ $\{c_1, c_2\} \leftarrow$ randomly generated solutions;

⑤ else

⑥ $\{c_1, c_2\} \leftarrow p_1, p_2$ after crossover;

⑦ for $i \in \{1, 2\}$

⑧ $c_i \leftarrow c_i$ after mutation;

⑨ end for

⑩ end if

ρ_1, ρ_2 分别是被选择的 2 个父代 p_1, p_2 的新颖性分数, DMS 先比较 2 个个体的新颖性分数, 选择较高的新颖性分数, 即 ρ_{better} (第 1 行); 再结合标准值 avg_score 进行判断(第 3 行), 如果父代的新颖性分数 ρ_{better} 低于平均新颖性分数 avg_score , 即父代的新颖性分数均低于平均分数, 则被选择的 2 个父代的适应度较低, 因此放弃父代, 调用 Halton-PaD 随机生成新的解, 从而扩大搜索范围(第 4 行), 并且跳过后续的遗传操作, 以保证其多样化效果; 否则, 将继续进行交叉、变异等操作(第 6~8 行)生成子代, 保留适应度较高的个体。最终算法返回新的解, 继续新颖性搜索。

2.3 DD 方法

原算法中每次只生成 2 个子代, 然后以新颖性分数作为标准, 筛去分数较差的配置。但这种方法存在一定缺陷, 在局部搜索中可能存在搜索不足的情况, 从而陷入局部最优, 在规定的时间内难以对配置空间进行有效探索, 导致样本集的多样性受到影响。为此, 本文提出了 DD 方法, 它根据概率选择性进行, 进一步提高种群的多样性。

算法 3 DD 方法

输入: 新生成的子代 c_1, c_2 ;

输出: 新生成的多样化子代 c'_{d1}, c'_{d2} 。

① for each features in SPL

② $f_{\text{fix}} \leftarrow$ find the first feature with fixed value in all samples;

③ judge whether f_{fix} is allowed to flip;

④ if f_{fix} is allowed to flip;

⑤ $\{c'_{d1}, c'_{d2}\} \leftarrow c_1, c_2$ with flipped f_{fix} ;

⑥ break;

⑦ end if

⑧ end for

DD 首先在档案的各配置中和经 DMS 后得到的 2 个子代中进行比较, 若某个特征 f_{fix} 赋值始终相同, 且其并非强制特征或无效特征, 则将子代中该特征 f_{fix} 赋值反转(第 5 行)。DD 是否进行取决于概率 θ , θ 的取值随着解决的 SPL 不同而变化。为了更直观地阐述, 给出 DD 实例, 见图 1。

```
00011100001000...1
10101100100000...0
00010100001001...1
00011110011100...0
```

图 1 DD 实例

Fig. 1 Example of DD

图1中的二进制串表示档案中的各个配置,对比发现第2个特征的取值在各配置中取值均为“0”;假设该特征不是强制特征或无效特征,则算法此时将子代的该位翻转并赋值为“1”。通过这种方法得到的样本集会更多样化。

2.4 Halton-PaD方法

抽样时需要样本集均匀地覆盖行为空间,而最终的样本集是在初始种群的基础上进行新颖性搜索得到,因此初始种群的选择尤为关键,即需要初始种群分布均匀,从而尽可能获取更多有关行为空间分布的信息^[30]。Halton序列是标准的低差异度序列^[27],它可以生成无穷个样本,并保证这些样本分布均匀^[31]。Halton-PaD以Halton序列为基础实现,它根据作为概率的Halton序列值决定是否选择某特征,和PaD都能保证概率在 $[0, 1]$ 间取值。

对于从Halton序列得到的概率 p_h ,Halton-PaD生成的初始种群中,所选特征数量的数学期望为 $\sum_{i=1}^n [p_h \times 1 + (1 - p_h) \times 0]$,则所选特征数量的平均值为 $p_h n$,故而所选特征数量的范围是 $[0 \times n, 1 \times n]$,可以保证初始种群在行为空间均匀分布。

因此,HaltonPaD不仅能达到PaD的效果,还可以提高初始种群的均匀性。

3 实验设置

3.1 所用的FM

实验共选用39个FM作为数据集,FM的选取基于先前的研究^[5,20,31-34],除了SPLOT-Generated-FM-5 000为人工生成^[20],其余都可以代表真实世界的SPL或高度可配置系统。FM中特征数量、子句数量不等,可以根据特征数量将其分为3类:小型的FM,即特征数量 $<1\ 000$;中型的FM,即特征数量在 $1\ 000 \sim 6\ 000$;大型的FM,即特征数量 $>6\ 000$ 。

表1具体给出了各FM的信息,其中 $|F|$ 表示FM中的特征数量, $|C|$ 表示特征间约束数量, $|core|$ 和 $|dead|$ 分别表示强制特征与无效特征的数量。将39个FM按表中顺序分别编号为1~39,其余表格使用编号分别对应相应的FM。

表1 实验所用的FM

Table 1 FMs used in the experiment

FM	$ F $	$ C $	$ core $	$ dead $
fiasco_17_10	234	1 178	7	6
lrzip	20	63	2	0
BerkeleyDBC	18	19	2	0
HiPAcc	31	104	1	0
Jhipster	45	104	7	0
E-shop	290	426	30	0
Printers	172	310	50	0
SmartHomev2.2	60	82	7	0
WebPortal	43	68	4	0
busybox_1_28_0	998	962	12	0
Polly	40	100	9	0
7z	44	210	5	0
JavaGC	39	105	7	0
VP9	42	104	9	0
axtls_2_1_4	94	190	4	0
uClibc-ng_1_0_29	269	1 403	14	0
toybox_0_7_5	316	106	8	15
toybox	544	1 020	4	365
axTLS	684	2 155	3	381
fiasco	1 638	5 228	49	964
SPLOT-FM-5000	5 000	9 419	75	0
uCLinux	1 850	2 468	7	1 237
ref4955	1 218	3 099	0	50
adderII	1 276	3 206	0	37
ecos-icse11	1 244	3 146	0	35
m5272c3	1 323	3 297	0	43
pati	1 248	3 266	0	47
olpce2294	1 274	3 881	0	42
integrator_arm9	1 267	50 606	0	44
at91sam7sek	1 319	3 963	0	74
se77x9	1 319	49 937	0	58
phycore229x	1 360	4 026	0	89
freebsd-icse11	1 396	62 163	3	38
busybox-1.18.0	6 796	17 836	12	3 939
uCLinux-config	11 254	31 637	15	6 012
buildroot	14 910	45 603	100	6 895
freetz	31 012	102 705	117	14 445
2.6.28.6-icse11	6 888	343 944	58	102
embtoolkit	23 516	180 511	839	6 561

3.2 绩效指标

Xiang等^[25]提出了衡量样本集多样性的指标分散值(Spread),该值用于衡量样本集在行为空间的分布是否均匀,表示为:

$$\text{Spread}(S, B') = \frac{1}{|B'|} \sum_{i=1}^{|B'|} d_{\min}(b'_i, S) \quad (1)$$

式中: $B' = \{|core|, |core| + 1, \dots, n - |dead|\}$, 如 1.2 节所述, 它可以近似为行为空间; b_i' 为 B' 中的第 i 个整数; $S = \{s_1, s_2, \dots, s_N\}$ 为样本集; $d_{\min}(b_i', S)$ 为 b_i' 和 S 的最小距离。

$$d_{\min}(b_i', S) = \min_{j=1}^N |b_i' - T(s_j)| \quad (2)$$

式中: $T(s_j)$ 为 S 中有效配置 s_j 所选择的特征数量。

通过式(2)可以得到结论: 分散值越小的配置所在的区域就越稀疏, 而新颖性搜索会更倾向于稀疏区域的配置, 对该区域的探索会促进种群的多样性, 因此分散值越小的样本集往往在行为空间具有更好的覆盖。

为了更好比较不同的采样器, Xiang 等^[26]还提出了关于分散值的改善率(Improvement rate, IR)。假设采样器 s_1, s_2 所选择的样本集分别是 A_1, A_2 , 则 IR 表达式为:

$$IR(s_1, s_2) = \frac{\text{Spread}(A_1, B) - \text{Spread}(A_2, B)}{\text{Spread}(A_1, B)} \times 100\% \quad (3)$$

如果采样器 s_1 的性能优于 s_2 , 则 $IR(s_1, s_2) < 0$, 即样本集 A_1 比 A_2 具有更好的行为空间覆盖, 因此是更优的样本集。

3.3 实验细节

为了消除随机性的偏差, 参考文献[25, 26]的研究成果, 所有采样器都将在各 FM 上独立运行 30 次; 在每次独立运行中, 若采样器迭代超过 10 次, 且在此期间, 其样本集的新颖性分数变化率不超过 0.1%, 则说明采样器已经相对稳定, 此时停止运行, 返回样本集。如果采样时间超过 1 h, 算法也会终止; 同时, 所有采样器的种群大小设置为 100, 交叉概率设置为 1.0, 变异概率设置为 0.1。本文后续将用性能指标衡量各样本集的优劣, 以达到判定采样器性能的目的。DDHNSbS 采样器与 NSbS 采样器相似, 也同时运用 2 种求解器。其中, 将 ProbSAT 求解器的参数 max_flips 赋值为 30 000, 采样器返回的样本集中有效配置数设置为 100。

所有实验都在安装了 ubuntuLTS 的台式机上运行, 硬件配置如下: Intel(R)Core(TM)i5-3470 CPU@3.20 GHz 和 7.7 GB RAM。实验涉及的所有算法都由 java 实现。

4 实验结果及分析

为了判断改进后是否显著提高采样器的性能, 对每个 FM 都执行显著性水平为 0.05 的 Mann-Whitney U 测试^[35], 用“•”“‡”和“°”分别表示显著优于、近似相同、显著差于。为评估 DMS、DD 以及 Halton-PaD 对算法性能的贡献, 针对不同优化策略分别进行实验, 通过各实验结果比较分析优化策略的效果。

4.1 DDHNSbS 与 NSbS、PaD-NSbS 进行比较

在本小节中, 将 DDHNSbS 采样器与 NSbS 采样器和 PaD-NSbS 采样器进行比较, 以分散值作为判定标准, 将最优结果用粗体表示, 其余表格保持一致。实验结果如表 2 所示, 表中用 DDHN 代表 DDHNSbS, PaDN 代表 PaD-NSbS。

由表 2 可知, 在所有 FM 中, DDHNSbS 的性能要显著优于或近似于 NSbS、PaD-NSbS。其中, 在中型的 FM 中, DDHNSbS 在 85.7% 的 FM 上对比 NSbS 有显著提高, 而在小型和大型的 FM 中, DDHNSbS 也能保证性能与 NSbS 相当; 对比最新的算法 PaD-NSbS, DDHNSbS 在 42.1% 的小型 FM 中表现显著优于 PaD-NSbS, 在中型的 FM 中也有显著改进的例子, 如 freebsd-icse11。DDHNSbS 在所有 FM 中, 能保证显著优于或近似于 PaD-NSbS。图 2 为算法分散值最佳的数量, 可以直观地表示实验结果, 其中横坐标表示 3 种算法, 纵坐标表示针对每个 FM, 某算法计算得到的分散值最佳的数量, 包括并列最佳的情况。从图 2 可得, 在 39 个 FM 中, DDHNSbS 计算得到的分散值为最佳的数量超过 NSbS 以及 PaD-NSbS, 它在 32 个 FM 中表现最佳, PaD-NSbS 在 14 个 FM 中表现最佳。

为比较 DDHNSbS 与最新采样器 PaD-NSbS 的性能, 图 3 根据 Mann-Whitney U 测试结果, 总结了分散值显著提高的 FM 中, 各规模 FM 所占的百分比。如图 3 所示, 在分散值显著提高的 FM 中, 有 72.8% 的 FM 为小型, 27.2% 为中型, DDHNSbS 与 PaD-NSbS 在大型 FM 上的性能相当, 这说明 DDHNSbS 在小型和中型 FM 中更具优势。根据实验结果可得, 当 FM 的特征数量较少时, DDHNSbS 通过 Halton-PaD 对比 PaD 生成的初始种群会更均匀地分布在行为空间, DMS 在此基础上保留新颖性分数高的个体进行遗传操作生成子代, 使最优解更为集中, 同时淘汰新颖性分数低的个体, 并

表 2 DDHNSbS 与 NSbS、PaD-NSbS 的比较

Table 2 Comparison of DDHNSbS, NSbS and PaD-NSbS

FM 编号	DDHN	NSbS			PaDN	FM 编号	DDHN	NSbS			PaDN
1	63.006	63.148	‡	65.220	‡	21	357.001	362.475	‡	376.201	•
2	2.053	2.053	‡	2.053	‡	22	2.125	18.621	•	9.992	•
3	1.824	1.824	‡	1.824	‡	23	5.635	7.794	•	5.466	‡
4	4.002	4.045	‡	4.355	•	24	5.932	8.152	•	5.868	‡
5	4.179	4.179	‡	4.195	‡	25	5.893	7.643	•	5.893	‡
6	1.128	4.175	•	1.202	•	26	6.624	8.902	•	6.581	‡
7	5.841	5.888	‡	5.891	‡	27	5.682	7.480	•	5.606	‡
8	0.051	0.331	•	0.064	•	28	5.962	7.542	•	5.964	‡
9	0.268	0.399	•	0.289	•	29	6.142	7.468	•	6.099	‡
10	4.098	6.743	•	4.623	•	30	6.093	8.701	•	6.044	‡
11	6.813	6.813	‡	6.813	‡	31	6.225	9.028	•	6.098	‡
12	12.650	12.650	‡	12.650	‡	32	6.143	8.877	•	6.144	‡
13	7.636	7.636	‡	7.636	‡	33	18.858	20.218	•	21.663	•
14	8.500	8.500	‡	8.500	‡	34	279.247	284.050	‡	284.028	‡
15	3.691	3.703	‡	4.007	•	35	399.488	401.478	‡	401.233	‡
16	21.432	21.724	‡	23.859	•	36	312.876	322.055	‡	321.407	‡
17	1.078	3.150	•	1.299	•	37	1 349.705	1 366.916	‡	1 370.377	‡
18	8.267	8.355	‡	8.448	‡	38	45.597	47.574	‡	47.405	‡
19	28.318	28.332	‡	28.591	‡	39	2 169.514	2 175.095	‡	2 175.915	‡
20	92.307	93.106	‡	94.124	‡						

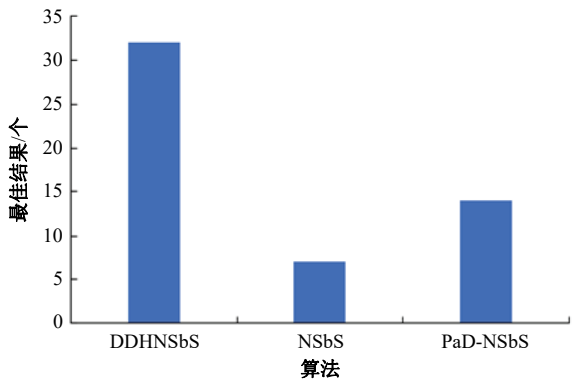


图 2 算法分散值最佳的数量

Fig. 2 Number of the best spread of each algorithm

随机生成新个体提高多样性;DD 根据特定的概率检测并翻转特征值,避免局部最优,使生成的样本集均匀广泛地覆盖行为空间,从而保证其改进的显著性。然而随着 FM 中特征数量的增加,PaD-NSbS 所选取配置的多样性也随之提高^[26],这就导致 DDHNSbS 的优势不再明显,DDHNSbS 和 PaD-NSbS 在大型 FM 中性能相当。

为了分析 DDHNSbS 性能好的原因,本文选择小型 FM 作为具体的例子生成配置。图 4 为各采样器在 WebPortal 上生成的配置所选择特征的数量。从选择特征的数量增加趋势可以看出,

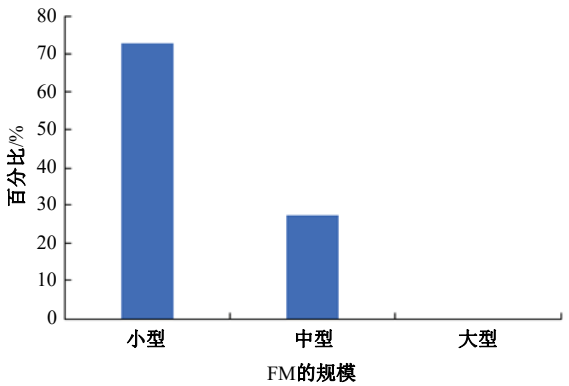


图 3 显著提高的 FM 中各规模 FM 所占百分比

Fig. 3 Percentage of FM of each scale in the significantly increased FM

DDHNSbS 的增加趋势更加规律,因此它所选取的配置在行为空间中分布更为均匀。

为进一步详细说明,需要观察各采样器在 WebPortal 上选取的连续配置增量,见图 5。其中,只有 DDHNSbS 的增量始终保持为 1,PaD-NSbS 为 1 或 2,NSbS 的表现则相对较差,它的增量除了 1 和 2 还存在 4。除了增量的稳定性,DDHNSbS 和 PaD-NSbS 还能保证选取大小近似的子集;相比之下,NSbS 选取平均分区的能力弱于前两者。

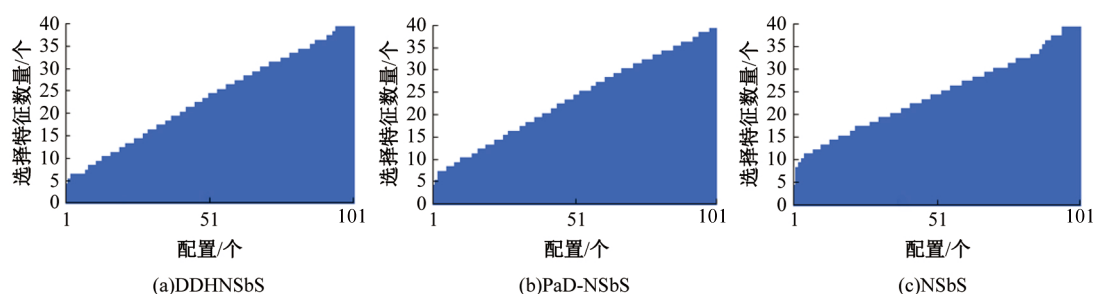


图 4 对于 WebPortal 的每个采样器生成配置所选择特征的数量

Fig. 4 Number of features selected for each sampler generation configuration of the WebPortal

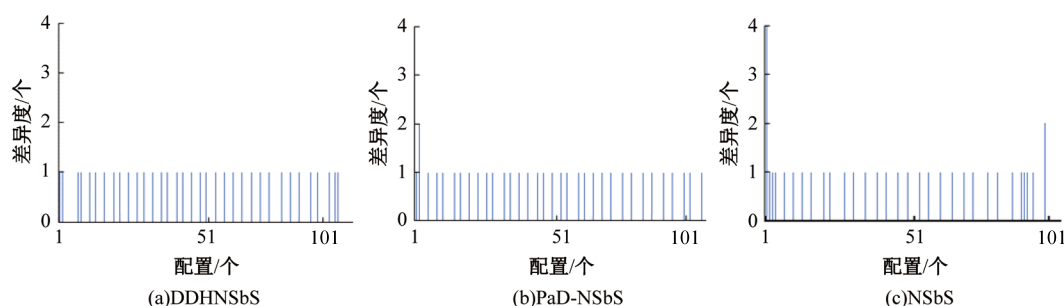


图 5 各采样器在 WebPortal 上生成的连续配置的增量

Fig. 5 Increment of the continuous configuration generated by each sampler on WebPortal

上述结果表明, DDHNSbS 采样时能够在行为空间中选取分布均匀且广泛的配置集。

4.2 DMS 的影响

将不采用 DMS 的 DDHNSbS 采样器称为 M1, 从而对比 DDHNSbS 采样器和 M1, 2 种算法在所有 FM 上执行显著性水平为 0.05 的 Mann-Whitney U 测试, 结果如下所示: DDHNSbS 对比 M1 分别在 24、15、0 个 FM 上性能近似相同、显著更优、显著更差, 可见对比 M1, 使用 DMS 策略的 DDHNSbS 在 15 个 FM 上有显著提高; 而在近似相同的 24 个 FM 中, DDHNSbS 有 22 个分散值小于或等于 M1。从 FM 的特征数量分析, 在 71.4% 的中型 FM 中, DDHNSbS 比 M1 有显著改进, 在小型的 FM 中也有显著改进的例子。总之, M1 算法的性能明显差于或近似于 DDHNSbS, 因此 DMS 策略能提高算法的多样性, 且如果 FM 为中型或小型, 改进效果会更好。

为了进一步探究 DMS 策略对算法的影响, 本文以数据集中的 2 个 FM 为例, 并分别绘制了经过 2 种采样器抽样后得到的样本集在行为空间的分布, 结果如图 6 所示。通过对比可知, DDHNSbS 在边界处也能进行采样, 因此在行为空间的分布很均匀; 而 M1 没有 DMS 的辅助, 在边界处生成的配置很少, 因此在边界处分布较为稀疏。

这种结果可以预见, 因为在 M1 算法进行局

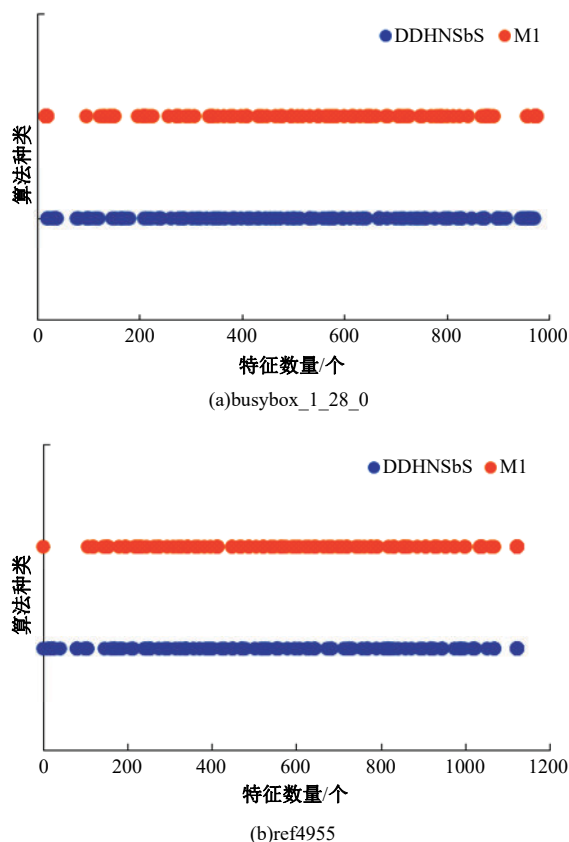


图 6 DDHNSbS 与 M1 在 busybox_1_28_0 以及 ref4955 上选取的配置在行为空间的分布

Fig. 6 Distribution of configurations selected by DDHNSbS and M1 on busybox_1_28_0 and ref4955 in the behavior space

部搜索时,仅对某个子代进行处理,会局限搜索范围,导致算法陷入局部最优的状态。同时,当父代产生新的子代时可能不会产生有利于种群多样化的解,通过使用DMS可以避免该问题,该策略保护适应度高的个体进行遗传操作,淘汰适应度低的个体,并随机生成新个体,从而扩大算法的探索空间,进一步提高多样性。由此可见,DMS对于算法的改进有效。

4.3 DD的影响

用M2表示不采用DD的DDHNSbS采样器,并将其与DDHNSbS采样器进行对比。通过实验比较DDHNSbS与M2可得,DD在39个FM上对多样性的提高并不显著,但DDHNSbS仍能获得较好的结果,具体结果如下:DDHNSbS所得分散值大于、等于、小于M2的数量分别为8、7、24,可见在79.5%的FM中,DDHNSbS对比M2都能取得较好或相等的结果。这说明在执行进化算法时存在特征赋值不变的情况,DD会找到未被翻转的变量并重新赋值,从而在一定程度上促使种群多样化,但并不足以显著提高算法性能。

4.4 Halton-PaD的影响

将采用PaD的DDHNSbS求解器命名为DDHNSbS_P,并将其与DDHNSbS进行对比,MannWhitney U测试结果如下:DDHNSbS对比DDHNSbS_P分别在35、4、0个FM上性能近似相同、显著更优、显著更差。具体地说,DDHNSbS在18个FM上表现较好,其中在4个FM上显著优于DDHNSbS_P;在8个FM上与DDHNSbS_P的分散值相等,且均为小型或中型的FM,不存在显著差于DDHNSbS_P的情况。通过2种算法的对比,Halton-PaD对行为空间的覆盖比PaD更加完整;但在面对特征数量庞大的大型FM时,Halton-PaD与PaD对初始种群的采样都趋向于均匀覆盖整个行为空间。因此,在大型FM中,DDHNSbS与DDHNSbS_P性能相当。总体来看,使用Halton-PaD策略提高初始种群的平均性对算法性能的改进有效。

4.5 不同遗传算子的影响

通过比较不同交叉算子以及不同选择算子,以判断出最高效的遗传算子组合,并应用到DDHNSbS中。将采用统一交叉遗传算子的DDHNSbS采样器称为M3,采用随机选择遗传算子的DDHNSbS采样器称为M4,以便于后续的对比如。

首先从不同的交叉算子开始实验,即UC与单点交叉(Singlepoint crossover, SPC)之间进行比较。UC是指子代按位交错选择2个父代的值生成新的子代;SPC则需要确定交叉点,再将第1个父代的首位至交叉点位置的值与第2个子代相应位置的值交换,从而获得新子代。交叉算子的比较结果见表6。

表6 DDHNSbS与M3的比较

Table 6 Comparison of DDHNSbS and M3

FM编号	DDHNSbS	M3		IR/%
1	63.006	63.177	‡	-0.27
2	2.053	2.053	‡	0.00
3	1.824	1.824	‡	0.00
4	4.002	4.025	‡	-0.57
5	4.179	4.179	‡	0.00
6	1.128	1.142	‡	-1.24
7	5.841	5.854	‡	-0.22
8	0.051	0.053	‡	-3.92
9	0.268	0.264	‡	1.49
10	4.098	4.212	‡	-2.78
11	6.813	6.813	‡	0.00
12	12.650	12.650	‡	0.00
13	7.636	7.636	‡	0.00
14	8.500	8.500	‡	0.00
15	3.691	3.693	‡	-0.05
16	21.432	21.568	‡	-0.63
17	1.078	1.116	‡	-3.53
18	8.267	8.394	‡	-1.54
19	28.318	28.311	‡	0.02
20	92.307	92.300	‡	0.01
21	357.001	357.762	‡	-0.21
22	2.125	2.380	•	-12.00
23	5.635	6.236	•	-10.67
24	5.932	6.660	•	-12.27
25	5.893	6.558	•	-11.28
26	6.624	7.165	•	-8.17
27	5.682	6.261	•	-10.19
28	5.962	6.728	•	-12.85
29	6.142	6.762	•	-10.09
30	6.093	6.689	•	-9.78
31	6.225	6.933	•	-11.37
32	6.143	6.809	•	-10.82
33	18.858	18.697	‡	0.85
34	279.247	276.312	‡	1.05
35	399.488	399.740	‡	-0.06
36	312.876	317.214	‡	-1.39
37	1 349.705	1 351.619	‡	-0.14
38	45.597	46.134	‡	-1.18
39	2 169.514	2 172.433	‡	-0.13

由表 6 可知,UC 在 39 个 FM 上的表现要显著优于或近似于 SPC。图 7 为 DDHNSbS 对比 M3 的改善率,可以直观发现,在抽样中型的 FM 时,UC 的性能普遍显著优于 SPC。经计算可得,IR 的平均值为 -3.43% ,DDHNSbS 采样器相比 M3,多样性有所提高,因此选择 UC 作为算法的遗传算子。DDHNSbS 对比 M4 的改善率见图 8。

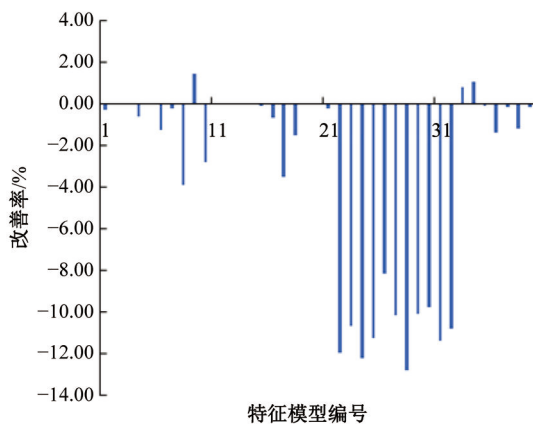


图 7 DDHNSbS 对比 M3 的改善率

Fig. 7 Improvement rate of DDHNSbS compared to M3

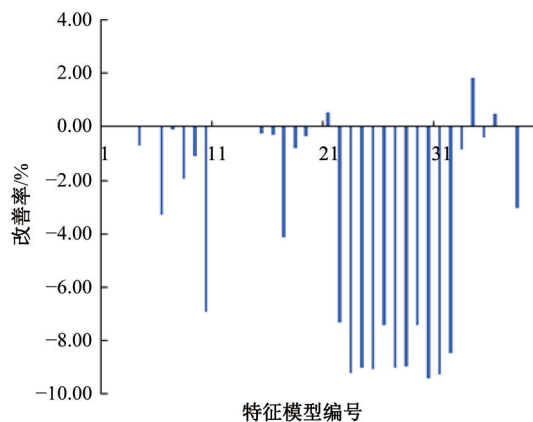


图 8 DDHNSbS 对比 M4 的改善率

Fig. 8 Improvement rate of DDHNSbS compared to M4

接下来比较不同的选择算子对算法的影响,本文采用二进制锦标赛选择(Binary tournament selection, BTS)和随机选择(Random selection, RS)。其中,BTS 在种群中选取指定数量的个体进行竞争,并捕获最优个体;RS 则在种群中随机选择 2 个个体作为父代。选择算子的比较结果见表 7。

由表 7 可得,2 种选择算子在小型和大型 FM 中的表现几乎一致,而 BTS 在中型 FM 中的表现

表 7 DDHNSbS 与 M4 的比较

Table 7 Comparison of DDHNSbS and M4

FM 编号	DDHNSbS	M4		IR/%
1	63.006	62.968	‡	0.06
2	2.053	2.053	‡	0.00
3	1.824	1.824	‡	0.00
4	4.002	4.031	‡	-0.72
5	4.179	4.179	‡	0.00
6	1.128	1.165	‡	-3.28
7	5.841	5.847	‡	-0.10
8	0.051	0.052	‡	-1.96
9	0.268	0.271	‡	-1.12
10	4.098	4.383	•	-6.95
11	6.813	6.813	‡	0.00
12	12.650	12.650	‡	0.00
13	7.636	7.636	‡	0.00
14	8.500	8.500	‡	0.00
15	3.691	3.700	‡	-0.24
16	21.432	21.492	‡	-0.28
17	1.078	1.123	‡	-4.17
18	8.267	8.332	‡	-0.79
19	28.318	28.421	‡	-0.36
20	92.307	92.305	‡	0.00
21	357.001	354.963	‡	0.57
22	2.125	2.281	•	-7.34
23	5.635	6.156	•	-9.25
24	5.932	6.468	•	-9.04
25	5.893	6.429	•	-9.10
26	6.624	7.118	•	-7.46
27	5.682	6.197	•	-9.06
28	5.962	6.498	•	-8.99
29	6.142	6.599	•	-7.44
30	6.093	6.667	•	-9.42
31	6.225	6.802	•	-9.27
32	6.143	6.667	•	-8.51
33	18.858	19.021	‡	-0.86
34	279.247	274.058	‡	1.86
35	399.488	401.03	‡	-0.39
36	312.876	311.247	‡	0.52
37	1 349.705	1 349.448	‡	0.02
38	45.597	46.994	‡	-3.06
39	2 169.514	2 168.406	‡	0.05

要明显优于 RS,该结论在 IR 数值中也有所体现。通过图 8 可得,在小型与大型的 FM 中,IR 的数值基本接近于 0;在中型 FM 中,IR 的绝对值较大。IR 的平均值为 -2.98% ,说明 DDHNSbS 采样器比 M4 性能更优,因此 BTS 的表现要优于或近似

于RS,本文在算法中倾向于选择BTS作为选择算子。

从上述实验和分析可得,最高效的遗传算子组合为UC与BTS。

5 结束语

本文提出了一种自适应多样性采样算法,即DDHNSbS算法。该算法包括3个策略,分别是DMS、DD以及Halton-PaD。从实验结果可得,DDHNSbS在部分FM中的表现显著优于NSbS以及最新算法PaD-NSbS,在其他FM中则性能相当。结合实验数据,在85.7%的中型FM中,DDHNSbS的表现显著优于NSbS;在42.1%的小型FM中,DDHNSbS的表现显著优于PaD-NSbS。对于各优化策略,本文设置了消融实验,分别探究其对整体算法性能的影响。DMS使DDHNSbS的性能在71.4%的中型FM和部分小型FM中显著提高;在79.5%的FM中,运用DD的DDHNSbS表现较好或与M2相同;运用Halton-PaD的算法与运用PaD的算法对比时,有4个FM显著优于后者,不存在显著差于的情况。因此,从实验可得,各策略有效。此外,本文对不同遗传算子进行比较,运用UC的DDHNSbS比运用SPC的DDHNSbS性能更好,IR的平均值为-3.43%;运用BTS的DDHNSbS比运用RS的性能更好,IR的平均值为-2.98%。因此DDHNSbS算法选择UC以及BTS作为遗传算子。

在未来的工作中,除了将分散值作为实验指标,还可以考虑其他度量方式,从不同角度衡量样本集的多样性。此外,可以考虑将该抽样工具投入实际应用中,完成现实目标,如通过样本集中配置的性能预测其余配置的性能。同时,本文所提出的3个优化策略也适用于其他SPL的任务,如SPL测试等。由于DDHNSbS算法只在小型和中型FM中对比PaD-NSbS更有优势,后续将针对大型FM的多样化抽样进行改进,使DDHNSbS算法的优势得以保持。最后,可以尝试优化进化算法,从而更好地结合DDHNSbS进行抽样,如结合卡尔曼滤波预测模型的进化算法。

参考文献:

- [1] Clements P, Northrop L. Software Product Lines [M]. Boston: Addison-Wesley, 2002.
- [2] Hou Y M, Ouyang D T, Tian X L, et al. Evolutionary many-objective satisfiability solver for configuring software product lines[J]. Applied Intelligence: The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies, 2022, 52(9): 10650-10673.
- [3] 孙宝凤, 任欣欣, 郑再思, 等. 考虑工人负荷的多目标流水车间优化调度[J]. 吉林大学学报: 工学版, 2021, 51(3): 900-909.
Sun Bao-feng, Ren Xin-xin, Zheng Zai-si, et al. Multi-objective flow shop optimal scheduling considering worker's load[J]. Journal of Jilin University (Engineering and Technology Edition), 2021, 51(3): 900-909.
- [4] 马永杰, 陈敏. 基于卡尔曼滤波预测策略的动态多目标优化算法[J]. 吉林大学学报: 工学版, 2022, 52(6): 1442-1458.
Ma Yong-jie, Chen Min. Dynamic multi-objective optimization algorithm based on Kalman filter prediction strategy [J]. Journal of Jilin University (Engineering and Technology Edition), 2022, 52(6): 1442-1458.
- [5] Pett T, Krieter S, Thüm T, et al. AutoSMP: an evaluation platform for sampling algorithms[C]//Proceedings of the 25th ACM International Systems and Software Product Line Conference-Volume B, New York, NY, USA, 2021: 41-44.
- [6] Devroey X, Perrouin G, Legay A, et al. Search-based similarity-driven behavioural SPL testing[C]//Proceedings of the 10th International Workshop on Variability Modelling of Software-Intensive Systems, Salvador, Brazil, 2016: 89-96.
- [7] Oh J, Gazzillo P, Batory D, et al. Scalable uniform sampling for real-world software product lines[R]. Austin: The Univ Texas at Austin, 2020.
- [8] Dutra R, Laeuffer K, Bachrach J, et al. Efficient sampling of SAT solutions for testing[C]//Proceedings of the 40th International Conference on Software Engineering, Gothenburg, Sweden, 2018: 549-559.
- [9] Plazar Q, Acher M, Perrouin G, et al. Uniform sampling of sat solutions for configurable systems: Are we there yet? [C]//Proceedings of the 2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST), Xi'an, China, 2019: 240-251.
- [10] Heradio R, Fernandez-Amoros D, Galindo J A, et al. Uniform and scalable SAT-sampling for configurable systems[C]//Proceedings of the 24th ACM Conference on Systems and Software Product Line, Montreal, Quebec, Canada, 2020: 1-11.

- [11] Liebig J, Von Rhein A, Kästner C, et al. Scalable analysis of variable software[C]//Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering, Saint Petersburg, Russia, 2013: 81-91.
- [12] Medeiros F, Kästner C, Ribeiro M, et al. A comparison of 10 sampling algorithms for configurable systems[C]//Proceedings of the 38th International Conference on Software Engineering, Austin, Texas, USA, 2016: 643-654.
- [13] Melo J, Flesborg E, Brabrand C, et al. A quantitative analysis of variability warnings in linux[C]//Proceedings of the 10th International Workshop on Variability Modelling of Software-intensive Systems, Salvador, Brazil, 2016: 3-8.
- [14] Cohen M B, Dwyer M B, Shi J. Interaction testing of highly-configurable systems in the presence of constraints[C]//Proceedings of the 2007 International Symposium on Software Testing and Analysis, London, England, United Kingdom, 2007: 129-139.
- [15] Krieter S, Thüm T, Schulze S, et al. YASA: yet another sampling algorithm[C]//Proceedings of the 14th International Working Conference on Variability Modelling of Software-Intensive Systems, Magdeburg, Germany, 2020: 1-10.
- [16] Garvin B J, Cohen M B, Dwyer M B. Evaluating improvements to a meta-heuristic search for constrained interaction testing[J]. Empirical Software Engineering, 2011, 16(1): 61-102.
- [17] Chakraborty S, Fremont D J, Meel K S, et al. On parallel scalable uniform SAT witness generation[C]//Proceedings of the 21st International Conference on Tools and Algorithms for the Construction and Analysis of Systems, London, UK, 2015: 304-319.
- [18] Ermon S, Gomes C P, Selman B. Uniform solution sampling using a constraint solver as an oracle[J/OL]. [2024-03-15]. <https://arxiv.org/abs/1210.4861>
- [19] Kaltenecker C, Grebhorn A, Siegmund N, et al. Distance-based sampling of software configuration spaces[C]//Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE), Montreal, Quebec, Canada, 2019: 1084-1094.
- [20] Henard C, Papadakis M, Perrouin G, et al. Looking for novelty in search-based software combinatorial exploration: using similarity to generate and prioritize t-wise test configurations for software product lines[J]. IEEE Transactions on Software Engineering, 2014, 40(7): 650-670.
- [21] Henard C, Papadakis M, Harman M, et al. Combining multi-objective search and constraint solving for configuring large software product lines[C]//2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, Firenze, Italy, 2015: 517-528.
- [22] Hierons R M, Li M, Liu X, et al. SIP: optimal product selection from feature models using many-objective evolutionary optimization[J]. ACM Transactions on Software Engineering and Methodology (TOSEM), 2016, 25(2): 1-39.
- [23] Sayyad A S, Ingram J, Menzies T, et al. Scalable product line configuration: a straw to break the camel's back[C]//2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE), Palo Alto, USA, 2013: 465-474.
- [24] Xiang Y, Zhou Y R, Zheng Z B, et al. Configuring software product lines by combining many-objective optimization and SAT solvers[J]. ACM Transactions on Software Engineering and Methodology (TOSEM), 2018, 26(4): 1-46.
- [25] Xiang Y, Huang H, Zhou Y R, et al. Search-based diverse sampling from real-world software product lines[C]//Proc of the 44th International Conference on Software Engineering, Pittsburgh, USA, 2022: 1945-1957.
- [26] Xiang Y, Yang X W, Huang H, et al. Sampling configurations from software product lines via probability-aware diversification and SAT solving[J]. Automated Software Engineering, 2022, 29(2): 1-45.
- [27] 黄美发, 景晖, 钟艳如, 等. 基于拟随机序列的三维模型表面采样方法[J]. 计算机工程, 2008, 34(14): 263-265.
- [28] 周慧思, 欧阳丹彤, 田新亮, 等. 基于模型诊断的一种新编码方法[J]. 计算机研究与发展, 2023, 60(1): 95-102.
- [29] Zhou Hui-si, Ouyang Dan-tong, Tian Xin-liang, et al. A novel encoding for model-based diagnosis[J]. Journal of Computer Research and Development, 2023, 60(1): 95-102.
- [30] Kang K C, Cohen S G, Hess J A, et al. Feature-oriented domain analysis (FODA) feasibility study[R]. Pittsburgh: Carnegie Mellon University, 1990.
- [30] Maaranen H, Miettinen K, Mäkelä M M. Quasi-ran-

- dom initial population for genetic algorithms[J]. Computers & Mathematics with Applications, 2004, 47 (12): 1885-1895.
- [31] Keller A. Quasi-Monte Carlo image synthesis in a nutshell[C]//Monte Carlo and Quasi-Monte Carlo Methods 2012, Sydney, Australia, 2013: 213-249.
- [32] Luo C, Sun B Q, Qiao B, et al. LS-sampling: an effective local search based sampling approach for achieving high t-wise coverage[C]//Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Athens, Greece, 2021: 1081-1092.
- [33] Baranov E, Legay A, Meel K S. Baital: an adaptive weighted sampling approach for improved t-wise coverage[J/OL]. [2024-03-15]. <https://www.cs.toronto.edu/~meel/Papers/fse20blm.pdf>
- [34] Xiang Y, Huang H, Li M Q, et al. Looking for novelty in search-based software product line testing[J]. IEEE Transactions on Software Engineering, 2021, 48(7): 2317-2338.
- [35] Arcuri A, Briand L. A practical guide for using statistical tests to assess randomized algorithms in software engineering[C]//Proceedings of the 33rd International Conference on Software Engineering, Honolulu, Hawaii, USA, 2011: 1-10.