协同进化在遗传算法中的应用述评

李 碧[1,2] 林土胜[1]
(华南理工大学电子与信息学院 广州 510641)[1] (广东外语外贸大学信息学院 广州 510420)[2]

摘 要 生态系统中协同进化的含义是几个生存能力相关联的种群的同时进化,在遗传算法中应用协同进化的实质是改变了个体适应度的计算方法:经典遗传算法中个体的适应度由它的染色体所决定,协同进化中个体的适应度却是由个体在协同关系中的表现决定。根据个体之间的适应度关联方式的不同,协同进化在遗传算法中应用可以分为两种:竞争协同进化算法、合作协同进化算法。竞争协同进化算法中的个体适应度由个体在竞争中的表现决定;合作协同进化算法中的个体适应度决定于个体在合作中的表现。对这两种方法的实质以及主要思想进行了述评。

关键词 遗传算法,协同进化,竞争,合作,适应度
中图法分类号 TP301 **文献标识码** A

Comments on Coevolution in Genetic Algorithms

LI Bi[1,2] LIN Tu-sheng[1]
(School of Electronic and Information Engineering, South China University of Technology, Guangzhou 510641, China)[1]
(School of Informatics, Guangdong University of Foreign Studies, Guangzhou 510420, China)[2]

Abstract Ecological coevolution refers to the simultaneous evolution of two or more species with inter-species assessment. The essence to apply coevolution in genetic algorithms is to change the way to evaluate individuals. Standard genetic algorithms evaluate individuals by their chromosomes, independent of other individuals in the evolutionary system. Coevolutionary algorithms evaluate individuals by their performance relative to others. According the way to evaluate individuals, the applications of coevolution in genetic algorithms are generally divided into two categories: Competitive Coevolution Algorithm (Comp-CEA) and Cooperative Coevolutionary Algorithm (Coop-CEA). Comp-CEA assesses individuals by their competitive performance in relation to evaluators. Coop-CEA assesses individuals by their cooperative performance relative to cooperators. This paper commented on the main ideas of Comp-CEA and Coop-CEA and the relevant key skills.

Keywords Genetic algorithm, Coevolution, Competitive, Cooperative, Fitness

遗传算法是密歇根大学 Holland 教授借鉴生物进化中的"生存竞争"和"优胜劣汰"现象提出的全局优化算法[1]。它将遗传操作应用于一群对搜索空间编码的染色体中,在每一代,遗传算法同时作用于整个搜索空间的不同区域,通过优胜劣汰,逐渐去掉解空间中期望值较低的部分,保留高期望值部分,从而能以较大的概率找到最优解。由于遗传算法具有不少传统优化算法所不具有的优点,自从20世纪70年代被提出来后已经得到了广泛的研究和应用[2-4]。但是与自然界中的生态系统相比,遗传算法的自适应能力有限。为了提高遗传算法的性能,特别是为了克服早熟现象,遗传算法吸收了理论生态学中的协同进化理论,取得了不少有效的成果[5-7]。本文概括和分析了生态进化中的协同进化理论及其在遗传算法中的主要应用方法。

1 生态进化中的协同进化理论

生态学中协同进化(coevolution)的概念由 Ehrlich 和 Raven 于1964年首次提出来[8],但是他们没有给出明确的定义。此后,不同的学者对协同进化给出了不同的定义,Roughgarden 认为协同进化的含义是:个体的适应度不仅决定于其染色体构成,还决定于个体所处种群的密度、以及个体与它发生相互关系的其它个体[9]。Jazen 给出了一个更严格的定义:一个物种的某一特性由于回应另一物种的某一特性而变化,而后者的该特性也同样由于回应前者的特性而变化[10]。这些定义都包含这样的含义:个体之间的生存能力是相互影响的,个体的生存能力不仅由它的染色体决定,还受到与之发生关系的其它个体的影响。

大自然中的生物是以在一定时间或一定空间上相互隔离的种群而存在的,同时各种群之间存在各种协同关系[11]。物种之间的相互关系虽然很复杂,但是对于任何一个物种来说,都只存在3种可能性:受益、受害、中性。因此,通过排列组合,两个物种之间可能的相互关系可以概括为表1所列的9个方面,包括为11种具体的相互关系[12]:互惠、共生、共栖、

到稿日期:2008-05-23 本文受国家自然科学基金项目(60673191),广东外语外贸大学创新项目(GW2006-TB-012),广东外语外贸大学校级青年项目(GW08Q02)资助。
李 碧(1973-),男,讲师,博士研究生,主要研究方向为进化计算、生物特征识别,E-mail:libi@mail.gdufs.edu.cn;林土胜(1945-),男,教授,博士生导师,主要研究方向为电路与系统、信号与信息处理、生物特征识别。

寄生、类寄生、植食、捕食、竞争、抗生、互惠和中性。表1中的+，－，0分别表示受益、受害、中性；例如“+，－”表示在相互作用中A受益、B受害。

表1 物种间关系一览表

		物种B受到的影响		
		+	0	−
物种A受到的影响	+	+，+ 互惠共生	+，0 共栖(偏利)	+，− 植食 捕食
	0	0，+ 共栖(偏利)	0，0 中性	0，− 偏害 竞争
	−	−，+ 寄生 类寄生	−，0 偏害 抗生	−，− 互抗

2 遗传算法同生态进化的差别

个体的适应度描述了个体的生存能力。在经典遗传算法中，个体适应度由个体的染色体决定，而与周围其它的个体没有关系。在不同时代，或者不同的群体中，具有相同染色体的个体，其适应度是相同的。但是，自然界的生物之间存在各种协同关系，不仅有个体进化，还有群体进化，并且个体之间、群体之间的进化是相互影响的[13]。所以，经典遗传算法中个体适应度的评价方法与生物进化中的实际情形不相符。

协同进化的含义是几个适应度相关联的种群的同时进化[5]（图1是两个种群协同进化的示意图），在遗传算法中应用协同进化的实质是改进个体生存能力的评价方法：个体的适应度不仅与自己的染色体有关，还受与之发生相互关系的其它个体的影响。根据所发生的相互关系的不同特点，协同进化在遗传算法中的应用分为如下两种：“竞争协同进化算法”[7,15,16,18]（Comp-CEA：Competitive Coevolution Algorithm）和“合作协同进化算法”[6,17,19,21]（Coop-CEA：Cooperative Coevolutionary Algorithm）。

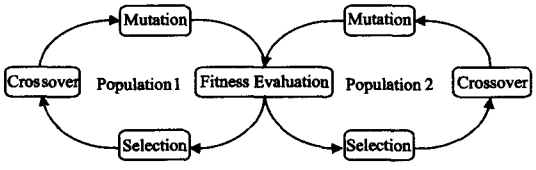

图1 两个种群的协同进化示意图

3 竞争协同进化算法（Comp-CEA）

竞争协同进化算法是对生态进化中捕食现象的模拟：捕食者和被捕食者之间相互产生有害的影响，捕食者和被捕食者的任何一方的进步都会威胁另一方的生存能力。捕食者和被捕食者的生存能力都不完全由本身决定，还受到对方的影响。捕食者为了捕获被捕食者的生存压力会刺激捕食者的进化，被捕食者为了逃脱被捕食的生存压力也会刺激被捕食者逐渐进化。捕食者和被捕食者相互刺激对方而协同进化。自然界中的猎豹和羚羊就是一个很典型的例子，生存的压力迫使猎豹和羚羊在长期的进化过程中，变成了能够高速奔跑的动物。

3.1 Comp-CEA中适应度的计算

在Comp-CEA中，个体的适应度叫做竞争适应度（CF：Competitive Fitness）[16]，竞争适应度决定于个体与临时竞争对手竞争中的表现。具有相同染色体的个体，可能由于临时竞争对手的不同，而得到不同的竞争适应度，所以Comp-CEA中的竞争适应度是一种相对适应度。相对应地，经典遗传算法中完全由染色体决定的适应度叫做绝对适应度。

被计算竞争适应度的个体叫做“学习者”（Learner），临时竞争对手叫做“评价者”（Evaluator）。令L为学习者的集合，E为评价者的集合。

计算机竞争适应度的原始方法是“简单竞争适应度”（Simple Competitive Fitness）[23]，其思想就是统计每个学习者所打败评估者的次数，如式(1)所示：

$$\forall i\in L\Rightarrow CF_i=\sum_{\substack{j\in E\\ i\ \text{defeat}\ j}}1 \tag{1}$$

“简单竞争适应度”容易导致“转圈”（cycling）、“脱节”（disengagement）等弊端的出现[16]。计算竞争适应度的常用方法是“共享竞争适应度”（Competitive Fitness Sharing）[5]，方法如下：

$$\forall j\in E\Rightarrow N_j=\sum_{\substack{k\in L\\ k\ \text{defeat}\ j}}1 \tag{2}$$

$$\forall i\in L\Rightarrow CF_i=\sum_{\substack{j\in E\\ i\ \text{defeat}\ j}}\frac{1}{N_j} \tag{3}$$

从式(2)和式(3)可以分析出，一个学习者能够击败的评价者越多，学习者的适应度可能越大。所以，在生存的压力下，由于评价者的激励，学习者的总体适应度水平会逐渐升高。从式(2)和式(3)还可以分析出，一个学习者如果能够击败一个很少有其它学习者能够击败的评价者，那么这次击败对这个学习者的适应度能够贡献一个较大的增量，因为总值为1的适应度被平均分配给较少的学习者。反之，一个能被学习者击败的评价者同时也被很多其它的学习者击败，那么这次击败对这个学习者的适应度只能够贡献一个较小的增量。

一个能被学习者能够击败的评价者，越是能被很少的其它学习者击败，那么这个不平常的学习者对寻找最优解越是有价值；所以在这种情况下，要增大该学习者的竞争适应度，从而提高它的生存能力。反之，一个学习者的所击败的个体越被更多的其它学习者击败，这个平常的学习者对寻找最优解的价值就越小；所以在这种情况下，就要减小该学习者的竞争适应度，从而减小它的生存能力。这样就可以鼓励新颖的优良积木块的出现，从而提高学习者染色体的多样性。

评价者的选取方式有多种：随机配对（Random Paring）[18]，指的是为每一个学习者随机指定一个临时对手；锦标赛淘汰制（Single Elimination Tournament）[18]，指的是首先所有个体随机配对，负者出局，胜者进入下一轮随机配对；随机选定多个临时对手（K-Random Opponent）[14]；穷举式（Round Robin）[18]，指的是所有的评价者都是任何一个学习者的临时竞争对手；共享取样（Shared Sampling）[5]，指的是选取的临时对手在上一代要有较大的竞争适应度。其中随机配对的计算量最小，性能最差；穷举式的计算量最大，性能最好。综合权衡计算量的大小以及是否对学习者形成全方位的有效刺激，一般选取共享取样[5]。

3.2 Comp-CEA的流程

Comp-CEA中的个体轮流担任学习者和评价者的角色，轮流给对方施压，轮流刺激对方总体适应度水平的提高，轮流刺激对方新颖积木块的出现，从而以一种“军备竞赛”（Arms Race）的方式寻找最优解[5]。

在 Comp-CEA 通常存在两个群体或多个群体，两个群体的 Comp-CEA 的伪代码如下：

```
Begin;
Initialize Pop1,Pop2;
Let pop1 serve as learner;
Select the set of evaluators form pop2;
While (not termination){
    Evaluation (learner);
    Selection (learner);
    Crossover (learner);
    Mutation (learner);
    Interchange the individuals in pop1 and pop2;
    Select the set of evaluators from pop2}
End
```

3.3 Comp-CEA 的特点

Comp-CEA 的优点：

1) Comp-CEA 不仅模拟了生态进化中的“适者生存”、“优胜劣汰”现象，还模拟了生态系统中普遍存在的协同进化现象。

2) Comp-CEA 不仅能够通过生存的压力(由选择算子来实现)提高群体总的适应度；而且能够鼓励新颖积木块的出现，从而维护群体的多样性，这样有助于解决经典遗传算法中长期存在的过早收敛问题。

3) 当由染色体决定的绝对适应度无法直接有效地定义时，Comp-CEA 的优势就尤为明显。因为，在 Comp-CEA 中，竞争适应度是由个体在竞争中的表现来决定。

但是，Comp-CEA 也有自己特有的缺点，常见的弊端有：“转圈”(cycling)、“脱节”(disengagement)等。

1) “转圈”：为在 Comp-CEA 中，个体进化的目的是打败对手，而不是在保留以前进化的基础上进一步进化，这样可能导致以前进化中的优良积木块的丢失，从而可能出现“形成优良积木块”，“丢失优良积木块”这样的一个“转圈”[16]的弊端。避免“转圈”的常见方法是“名人堂”(fame hall)[5]。“名人堂”保存进化过程中的优秀个体，并且选取部分优秀个体作为评价者。

2) “脱节”：“脱节”现象指的是种群内的适应度多样性趋向于零，那么选择的梯度就会消失，无法形成一种你追我赶的“军备竞赛”[17,20]。克服脱节的常见手段有“资源共享”(Resource Sharing)和“减小毒性”(Reducing Virulence)[16]。“资源共享”通过维护种群内部的染色体多样性来避免“脱节”的发生。“减小毒性”则是直接通过选择评价者来保证学习者中的适应度值存在一定的方差。

3.4 Comp-CEA 的应用

可以在单种群中应用 Comp-CEA，也可以在多种群中应用 Comp-CEA。基于单种群的 Comp-CEA 是通过同一种群内个体之间的竞争来实现的，任何一个个体都是进化中的学习者，任何一个个体都可以作为其它个体的评价者。基于单种群的 Comp-CEA 已经成功地应用到了智能游戏（如 Tic-Tac-Toe)[23]，函数优化[18]。Comp-CEA 通常包含两个或者多个种群，通过模拟生态进化中的捕食关系来实现协同进化，不同的种群轮流担任学习者和评价者的抉择。基于多种群的 Comp-CEA 已经成功地应用到了智能游戏(如：3-D Tic-Tac-Toe，Nim)[5]、函数优化[7]、多目标优化[14]、最小路径分类网[16]、模式分类[22]、细胞自动机规则训练[24]、非线性控制器的设计[25]、人工神经网络[26]等领域。

4 合作协同进化算法(Coop-CEA)

Coop-CEA 是采用“分而治之”、“分工合作”的思想，把一个问题分解成几个部分分别求解[17,19,21]。Coop-CEA 一般应用于可以自然地进行分解问题的情形下[4]，Coop-CEA 已经成功地应用到了函数优化、生产调度、人工神经网络设计，以及计算机视觉等领域[16,21]。

4.1 Coop-CEA 的流程

传统的遗传算法中只有一个种群在进化，种群中的每一个个体表示一个完整的解，在进化的过程中每个个体的适应度值逐渐提高。而 Coop-CEA 却包含处于合作关系的多个种群在同时进化，种群中的每一个个体只表示解的一个部分。Coop-CEA 的每一个子种群求一个部分解，把多个子种群的最终解按顺序连接起来就是 Coop-CEA 的解。伪代码如下所示：

```
Begin;
Set 子种群的个数 n;
For (i=1; i<=n; i++)
    Initialize (pop[i]);
For (i=1; i<=n and not termination; i++){
    For (j=1; j<=n and j<>i; j++)
        pop[i] Cooperate with pop[j];
    Select (pop[i]);
    Crossover (pop[i]);
    Mutate (pop[i]);}
Solution=NULL;
For (i=1; i<=n; i++)
    Solution=combine (Solution,solution[i]);
End;
```

从形式上看，Coop-CEA 把传统遗传算法中的种群人为地从纵向分为多个子种群，每个子群体对应一个子任务(例如多个参数中的一个)。所以在应用 Coop-CEA 时，首要的工作是进行任务的分解。如果任务是确定 3 个参数的取值，那么就可以把整个种群分为 3 个子群体，如图 2 所示。Coop-CEA 的物理实现可以采用图 3 所示的粗粒度并行模型，每一个处理器对应一个子群体。

	子群体1	子群体2	子群体3
个体1	101	000	001
个体2	001	001	100
个体3	101	000	110
个体4	001	001	110
个体5	101	000	101
个体6	001	001	110
个体7	101	000	110
个体8	001	001	110
个体9	101	000	110

图 2 纵向划分

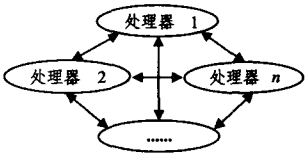

图 3 粗粒度并行模型

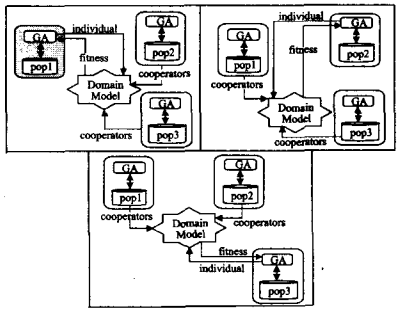

图 4 Coop-CEA 模型

图 4 是一个包含 3 个子群体的 Coop-CEA 拓扑模型[19]。在图 4 中，子群体中的个体相对独立地进化(选择、交叉、变异)，通过共享的“领域模型”(Domain Model)发生合作关系。领域模型的具体实现方法是与所解决的问题直接相关的，它是子群体发生合作的场所，也是计算所有个体适应度的场所。

4.2 Coop-CEA 中适应度的计算

传统遗传算法中的个体代表一个完整的解，个体的适应度表示解的性能。而在 Coop-CEA 中，子群体中的任何一个个体只是完整解的一个部分，个体的适应度表现为与其它各子群体中的个体的合作能力。为了求得某子群体中的一个个体(individual)的适应度，该子群体先把个体发送至“领域模型”，“领域模型”同时从其它的每一个子群体接收若干“合作者”(cooperators)来同那个个体合作，一起组成若干完整的解。领域模型把求得的适应度(fitness)中的最佳值作为所求个体的适应度发送给子群体[19]。

可以从其它各子群体随机选取合作者，也可以选取其它子群体中最佳的个体来同它合作。合作者的个数可多可少，最“贪婪”的做法是把其它子群体中所有的个体都选作合作者，“贪婪”的做法可以全面地衡量一个个体的合作能力，但是计算量很大。

另一种计算个体适应度的方法是根据个体的模板 (schemata)来计算。具体方法是：把子群体中的个体看成完整解模板中的确定部分，而其它的部分是不确定的，该个体的适应度为模板中的部分个体的平均适应度[17]，这种方法的实质是随机产生若干合作者。例如，图 2 的子群体 1 中个体 1 是“101”，其模板是“101＊＊＊＊＊＊”。根据模板随机生成 2 个个体：“101000101”，“101110001”，把这两个个体的平均值作为个体“101”的适应度。

在 Coop-CEA 中，为了计算子群体中个体的适应度，所选取的合作者的个数越少，计算个体的适应度的偶然性就越大；选取的合作者的个数越多，计算量就越大。“减小偶然性”和“减小计算量”是一对无法兼得的目标。减小计算量就得增加偶然性，减小偶然性就得增加计算量。所以，在 Coop-CEA 中，如何有效地计算个体的适应度是一个值得深入探讨的问题。同时，取样的方法也有待探讨。

Coop-CEA 中的同一个个体，可能由于合作者的不同而得到不同的适应度值，所以 Coop-CEA 中的适应度也是一种相对适应度。Coop-CEA 中的任何一个个体生存能力的体现，离不开其它的合作者。任何一个子群体的进化，对其它所有子群体都会产生正的影响。所以，Coop-CEA 是对生态进化中互惠共生的模拟。

结束语 本文概述了生态进化中的协同进化理论，对协同进化理论在遗传算法中的应用进行了概括和分析。经典遗传算法完全根据个体的染色体来计算个体的适应度，在遗传算法中应用协同进化的实质就是采用个体之间的协同关系来计算个体的适应度。根据个体之间的协同关系是竞争类型还是合作类型，协同进化算法可以分为竞争协同进化算法和合作协同进化算法。协同进化同遗传算法一样，不是一个具体的算法，而是一种来源于生物进化的优化思想，如何有效地应用到实际应用领域、如何更有效地模拟自然界的协同进化，以及协同进化算法的数学基础，还有待进一步的研究。

参 考 文 献

[1] Holland J H. Adaptation in Nature and Artificial System [M]. MIT Press, 1992

[2] Goldberg D E. Genetic Algorithms in Search, Optimization, and Machine Learning [M]. USA: Addison-Wesley, 1989

[3] Zheng Wei-Shi, Lai Jiang-Huang, Yuen P C. GA-Fisher: A New LDA-Based Face Recognition Algorithm with Selection of Principal Components [J]. IEEE Transactions on System, Man, and Cybernetics-Part B: Cybernetics, 2005, 35(5): 1065-1078

[4] 吕跃进，刘南星，陈磊. 一种基于并行遗传算法的粗糙集属性约简 [J]. 计算机科学，2008，35(3)：219-221

[5] Rosen C, et al. New methods for competitive coevolution [J]. Evolutionary Computation, 1997, 5(1): 1-29

[6] Tan K C, et al. A distributed cooperative coevolutinary algorithm for multiobjective optimization [J]. IEEE Transactions on evolutionary computation, 2006, 10(5): 527-549

[7] Li B, Lin Tu-Sheng, Liao Liang, et al. Genetic Algorithm Based on Multipopulation Competitive Coevolution [C]. IEEE Congress on Evolutionary Computation (CEC 2008), 2008: 225-228

[8] Ehrlich P R, Raven P H. Butterflies and plants: a study in coevolution [J]. Evolution, 1964, 18: 586-608

[9] Roughgarden J. Resource partitioning among competing species—a coevolutionary approach [J]. Theoretical Population Biology, 1976, 9(3): 388-424

[10] Janzen D H. When is coevolution [J]. Evolution, 1980, 34: 611-612

[11] 王德利，高莹. 竞争进化与协同进化[J]. 生态学杂志，2005，24(10)：1182-1186

[12] 尚玉书. 普通生态学[M]. 北京：北京大学出版社，2002

[13] 曹先彬，罗文坚，王煦法. 基于生态种群竞争模型的协同进化[J]. 软件学报，2001，12(4)：556-562

[14] Tan T G, et al. Competitive Coevolution with K-Random Opponents for Pareto Multiobjective Optimization [A]// Third International Conference on Natural Computation (ICNC 2007) [C]. 2007(4): 63-67

[15] Hillis W D. Co-evolving parasites improve simulated evolution as an optimization procedure [J]. Physica D, 1990, 42(1-3): 228-234

[16] Cartlidge J, Bullock S. Combating Coevolutionary Disengagement by Reducing Parasite Virulence [J]. Evolutionary Computation, 2004, 12(2): 193-222

[17] Vrajitoru D. Parallel Genetic Algorithms Based on Coevolution [A]// Genetic and Evolutionary Computation Conference [C]. 2001: 450-457

[18] Panait L, Luke S. A comparative study of two competitive fitness functions [A]// Genetic and Evolutionary Computation Conference [C]. 2002: 503-511

[19] Potter M A, De Jong K A. Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents [J]. Evolutionary Computation, 2000, 8(1): 1-29

[20] De Jong E D, Pollack J B. Ideal Evaluation from Coevolution [J]. Evolutionary Computation, 2004, 12(2): 159-192

[21] Krawiec K, Bhanu B. Visual Learning by Evolutionary and Coevolutionary Feature Synthesis [J]. IEEE Transactions on Evolutionary Computation, 2007, 11(5): 635-650

(下转第 63 页)

开发方法(AOSD, Aspect-Oriented Software Development),利用传统的面向对象技术建立领域模型及对象模型,实现为类(class)模块;通过用例(use case)捕捉业务功能需求,实现为方面(aspect)模块。每一个横切(crosscutting)方面可以跨越多个类,但其组织及变更又限制在单独的方面模块内。这就有效地将结构化方法的主要关注维度(功能需求)和面向对象方法的主要关注维度(对象实体)各自分离,成为各自独立的模块,并通过编织(weave)机制使模块合成为一个整体;在实现复杂功能的同时,使软件系统模块结构更清晰,并具有方便的可扩展性,为递增式演化开发提供了一种切实可行的程序设计方法论。

这些创新思想涉及专门编程技术和整体开发方法,被认为是继结构化开发范型和面向对象的开发范型之后,最重要的软件开发范型。AspectJ, JMangler, Spring, Tapestry, JBoss等工具和框架为Java应用提供了面向方面的实现支持。其他语言支持AOP特性的工具也在研究发展中。可以预见,随着对关注点分离在软件工程中的方法论意义的更深入更全面的认识,面向方面的开发方法将在软件开发实践中得到更广泛的应用。

架构作为近十几年业界研究和应用的热点之一,其定义还有待统一。Perry等在文献[21]给出的定义与我国学者徐家福的定义比较接近:软件架构={结构元(Elements),结构型(Forms),结构理(Rationale)},即软件架构是由若干具特定结构形按结构理组合起来的结构元的集合。结构元包括处理元、数据元和连接元。

结束语 关注点分离作为一种普适的处理复杂问题的系统思维方法和原则,在计算思维和软件工程中有着重要的方法论意义。关注点分离原则给出了处理复杂性的基本思路,在软件开发实践中呈现为具体设计原则和设计方法。本文对软件和计算本质的归纳,对关注点分离原则在软件工程中的具体体现的分析,只是冰山一角。

在实践中,关注点分离原则必须和具体问题具体分析策略相结合,以期获得恰如其分的分离视角以及简明优雅的合成策略。正如书法家刘炳森所说,世间的大学问全在于分寸和火候的把握。对应用领域的问题,须通过深入细致的调查研究,把握问题精微细致的"关节"。在"关节"处解"牛",才能体现融科学与艺术为一体的工程之精妙。

参 考 文 献

[1] 欧阳莹之. 复杂系统理论基础[M]. 田国宝,等译. 上海:上海科技教育出版社,2002:90-91

[2] Wang J M. Computational Thinking. Communications of ACM, 2007,49(3):33-35

[3] Parnas D L. On the criteria to be used in decomposing systems into modules. Comm. ACM,1972,15(12):1053-1058

[4] Dijkstra E W. On the cruelty of really teaching computing science. Comm. ACM,1989,32(12):1398-1414

[5] Ghezzi C, Jazayeri M, Mandrioli D. Fundamentals of Software Engineering (2nd edition). Pearson Education,2003

[6] Baldwin C Y,Clark C Y. Design Rules (vol. 1): The Power of Modularity. Cambridge,Mass: MIT Press,1999

[7] Stanley JR S M, Rouvellou I. 面向方面软件开发的关注点建模// R. E. Filman,et al. ,eds. 莫倩,等译. 面向方面的软件开发[M]. 北京:机械工业出版社,2006:299-315

[8] Tarr P,Ossher H,Sutton JR S M,et al. N度分离:关注点的多维分离// Filman R E,et al. ,eds. 莫倩,等译. 面向方面的软件开发[M]. 北京:机械工业出版社, 2006:21-36

[9] Jacobson I, Ng P W. Aspect-Oriented Software Development with Use Cases. Pearson Education,2005

[10] Simon H A. The Science of the Artificial. Cambridge. Mass: MIT Press,1969

[11] Wieringa R J. Design Methods for Reactive Systems: Yourdon, Statemate, and the UML. San Francisco: Morgan Kaufmann Publishers,2003

[12] Swatout W,Balzer R. On the inevitable intertwining of specification and implementation. CACM,1982,25(7):438-440

[13] 何明昕. 面向问题的系统化程序设计方法及其描述工具[J]. 计算机科学,1995,22(5):54-57

[14] Dreyfus H L. 计算机不能做什么—人工智能的极限[M]. 宁春岩,译. 北京:三联书店,1986

[15] Wirth N. Algorithms + Data Structures = Programs. Prentice Hall,1976

[16] Jackson M A. Principles of Program Design. Academic Press, 1975

[17] Warnier J D. Logical Construction of Programs. NY: Von Nostrand Reinhold,1976

[18] Gries D. The Science of Programming. Springer-Verlag,1981

[19] Stavely A M. 零缺陷程序设计[M]. 夏昕,王尧,译. 北京:机械工业出版社,2003

[20] Mayer B. Objected-oriented Software Construction, Englewood Cliffs NJ: Prentice Hall,1988

[21] Hornstmann C. 面向对象的设计与模式[M]. 张琛恩,译. 北京:电子工业出版社,2004:107-111

[22] Perry D E, Wolf A L. Foundations for the study of software architecture. Software Engineering Notes,1992,17(2):40-52

[23] Keuchten P. The 4+1 view model of architecture. IEEE Software,1995,12(6):42-50

[24] Soni D, Nord R L, Hofmeister C. Software Architecture in industry applications// Proc. of the 17th Inter. Conf. on SE. 1995: 196-207

[25] Clements P, Bachmann F, Bass L, et al. Documenting Software Architectures: Views and Beyond. Boston. MA: Pearson Education,2003

[26] Krafzig D, Banke K, Slama D. Enterprise SOA: Service-Oriented Architecture Best Practice. Pearson Education,2005

(上接第37页)

[22] Kowaliw T, et al. Using Competitive Co - evolution to Evolve Better Pattern Recognisers [J]. International Journal of Computational Intelligence and Applications,2005,5(3):305-320

[23] Angeline P J, Pollack J B. Competitive environments evolve better solutions for complex tasks [A]// Proceedings of the Fifth international Conference on Genetic Algorithms [C]. 1993:264-270

[24] Hugues J, Jordan B P. Coevolutionary Learning : a case study [A] // The Fifteenth International Conference on Machine Learning [C]. 1998:251-259

[25] Claverie J M, et al. Robust nonlinear control design using competitive coevolution// Proceedings of the 2000 Congress on Evolutionary Computation. 2000(1):403-409

[26] Floreano D, Nolfi S. Adaptive Behavior in Competing Co-evolving Species [C] // Fourth European Conference on Artificial Life. 1997:378-387