

第7届中国基于搜索的软件工程研讨会

Beijing, Nov. 17, 2018

当需求遇到搜索

软件需求工程中的搜索技术

赵海燕

高可信软件技术教育部重点实验室

北京大学

Research Interests

1. Requirements Engineering

- *Collaborative* requirements elicitation and analysis
- Requirements modeling based *collective intelligence*
- *Feature-centric* requirements management and reuse

2. Software Product Lines

- Variability modeling and management
- Feature driven Software products/artifacts derivation

3. Software Adaptation

- Adaptivity Modelling and Management
- Adaptation rule generation
- Goal & rule- based adaptation

需求工程

1. 需求的获取与建模

2. 需求驱动的软件自适应

需求获取建模活动的特点

- **涉众密集**
 - 客户、用户、领域专家、开发人员等众多利益相关者
- **知识密集**
 - 大量问题领域的知识，尤其隐性知识
 - 知识传递
- **沟通密集**
 - 涉众间的交流、协商

基于群体智能的 需求获取与建模

利用 群体的力量 求解 复杂问题 的方式

群体智能

- 生物通过大规模群体协同，避敌、筑巢、觅食



鱼群避敌

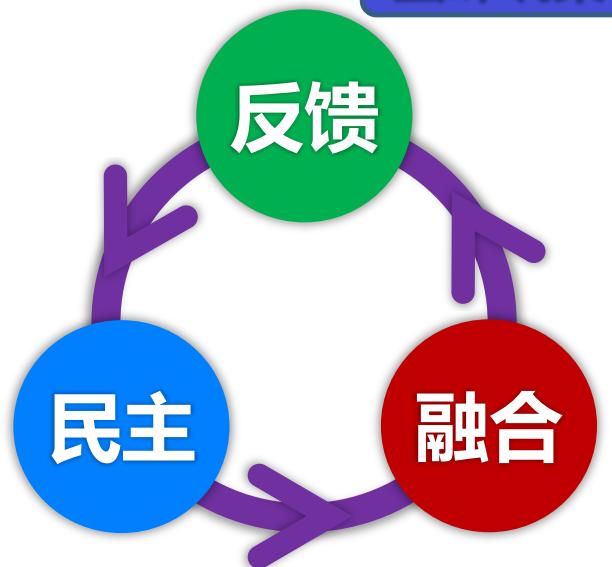


蚂蚁筑巢

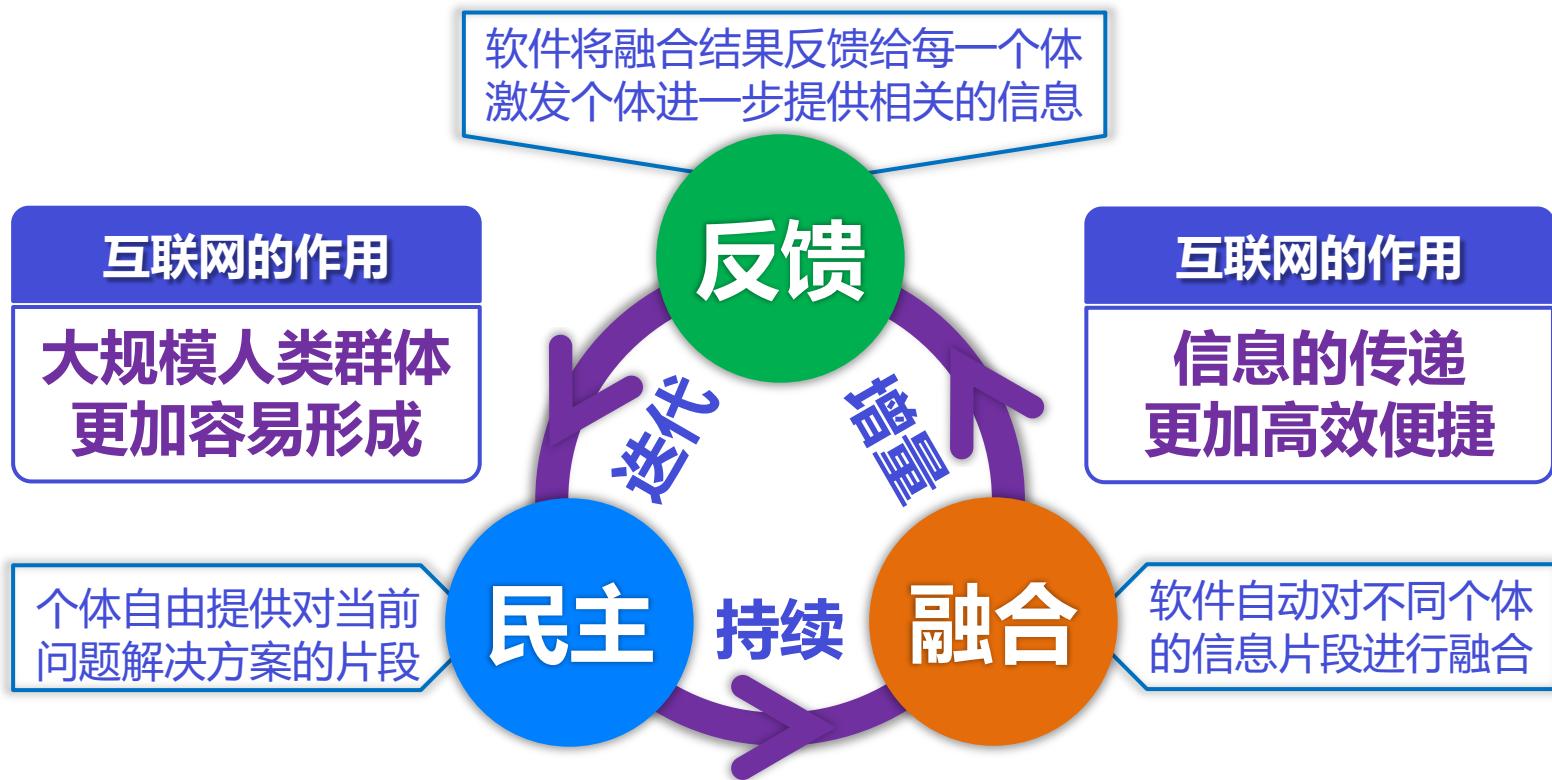


蜜蜂筑巢

- 人类社会通过群体协同完成特定任务



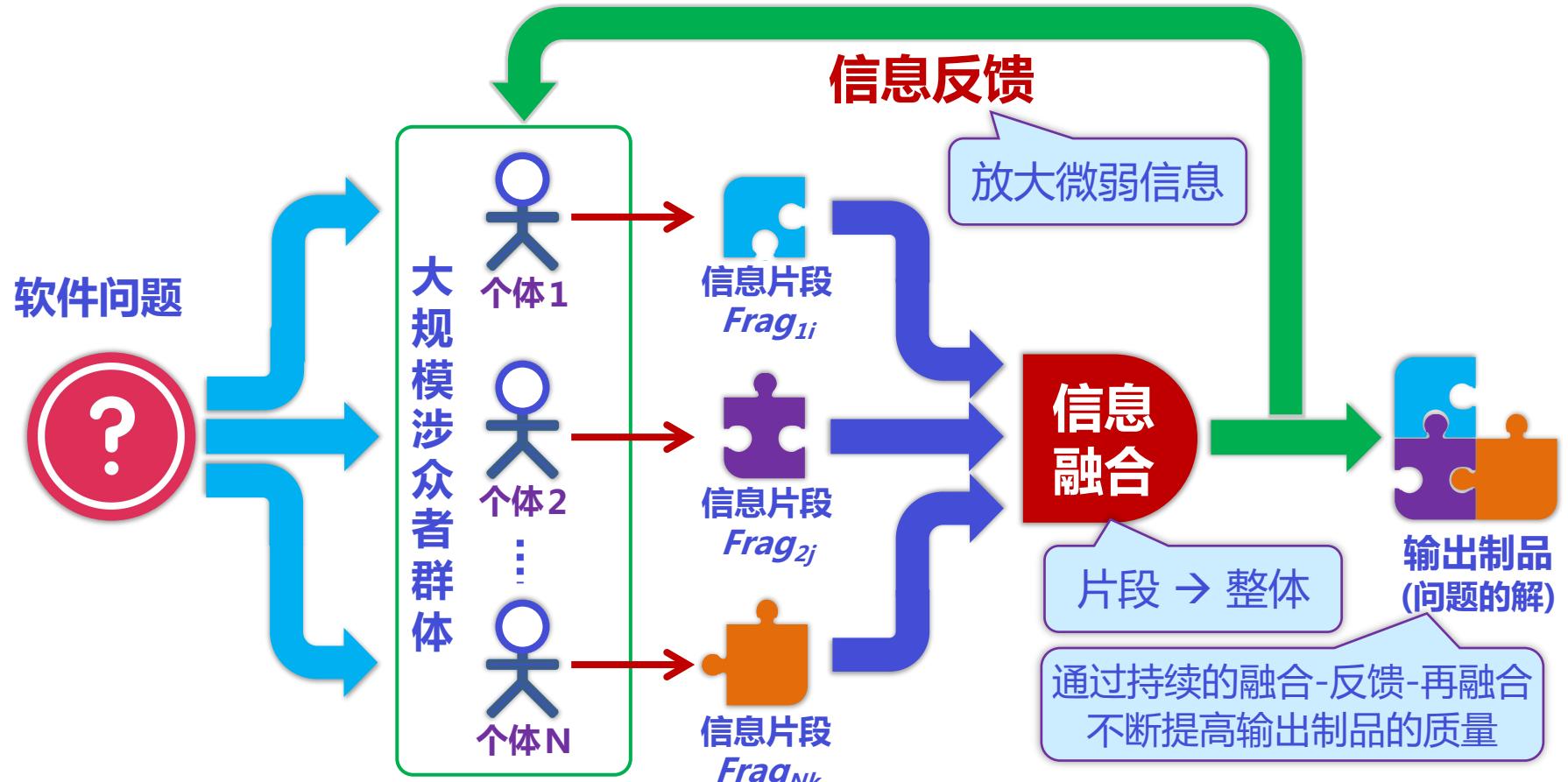
互联网与群体智能



群体协同：大规模对等协同模式激发需求的获取演化

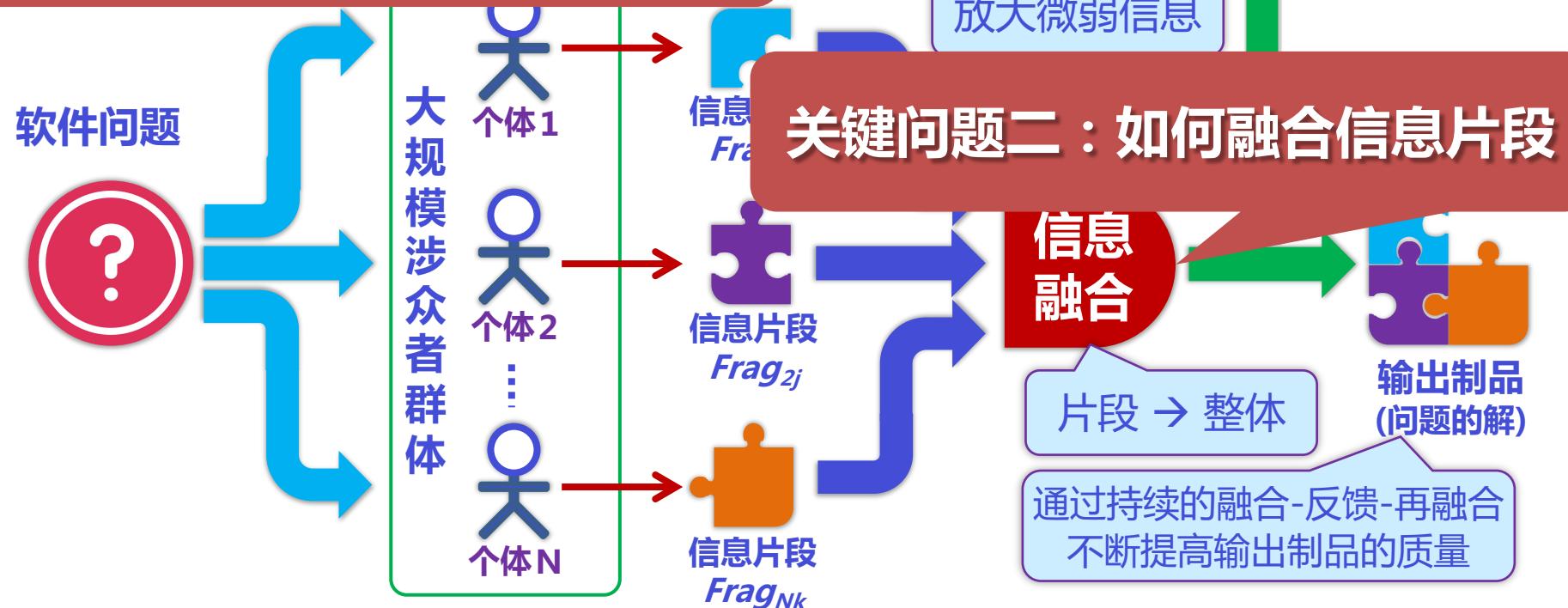
持续演化：需求在网络环境中获得持续反馈和不断改进

基于群体智能的需求建模基本框架



基于群体智能的需求建模基本框架

关键问题三：如何反馈信息片段



$$Frag_{i0} \rightarrow \sum_{i=1}^N Frag_{i0} = S_0$$

民主

→ 信息融合

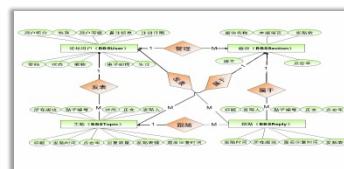
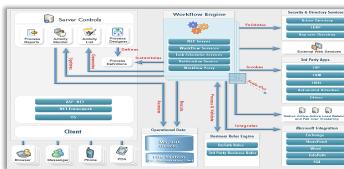
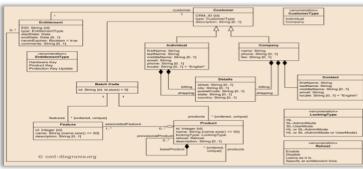
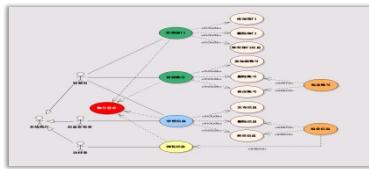
关键问题一：如何表示信息片段

基于图的软件制品的表示与存储

如何对软件制品进行表示，使得能够对其进行有效融合

将软件制品中包含的信息进行尽可能细粒度地分解

用例模型-需求获取 概念模型-需求分析 体系结构-软件设计 ER图-数据存储 程序源代码-编程



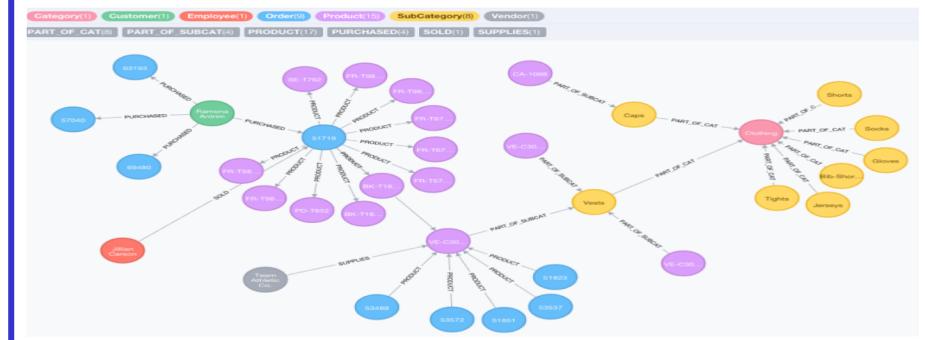
```
String mappingValue = "/api/v1/doc", method = RequestMethod.GET, produces = "application/json", consumes = "application/json");  
@RequestMapping(value = "/{id}", method = RequestMethod.PUT, produces = "application/json", consumes = "application/json");  
@RequestMapping(value = "/{id}/update", method = RequestMethod.PUT, produces = "application/json", consumes = "application/json");  
@RequestMapping(value = "/{id}/delete", method = RequestMethod.DELETE, produces = "application/json", consumes = "application/json");  
  
String suffix = "-"+(String)request.getAttribute("suffix");  
String path = this.serviceContext.getRealPath(suffix);  
String fileType = filename.substring(filename.lastIndexOf("."));  
String file = new String(path+filename.substring(0,filename.lastIndexOf(".")));  
for(String s : suffix){  
    if(s.equals(fileType)){  
        map.put("suffix", s);  
    }  
}  
  
for(Map.Entry entry : map.entrySet()){  
    if(entry.getKey().equals("suffix")){  
        File file2 = new File(file+entry.getValue());  
        try {  
            file2.createNewFile();  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
        file.createNewFile();  
        try {  
            file.write(file2);  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

软件开发中产生的各种不同类型的软件制品

以图 (Graph) 为基本组织方式，对软件开发中各种不同类型的软件制品进行统一化的建模

图结构的简单性为图融合提供了便利

基于图的软件制品统一建模框架



利用图数据库对软件开发产生的海量图数据进行存储与管理

图数据库对外提供的查询与修改机制对图的有效分析与变换提供了便利

基于图数据库的存储与管理

基于熵最小化的大规模图数据的融合

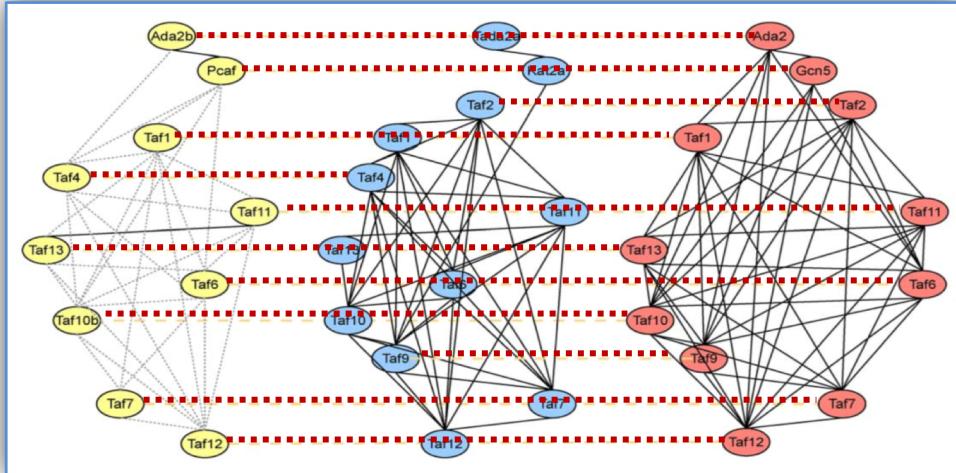
如何将不同个体提供的信息片段融合为更加有序的整体

- **融合**：将不同个体提供的重复信息进行合并
- **重复信息判断准则之一**：两个结点上下文越相似，则其在语义上相同的概率越高

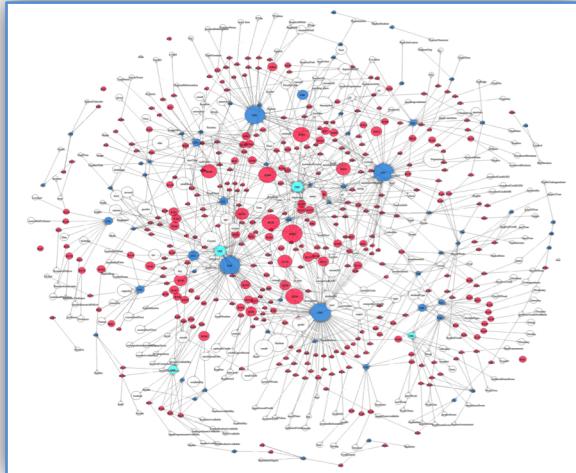
$$\mathcal{H}_s(X) = \sum_{x \in \mathcal{X}} p(x) \log \frac{1}{\sum_{x' \in \mathcal{X}} p(x') s(x, x')}$$

借鉴热力学与信息论的思路
采用**熵对图的有序程度**进行度量

Step 0: 随机生成一组融合方案



Step 1: 度量融合方案的有序程度



Step 2: 对融合方案进行优化选择

采用**演化式优化算法**
寻找足够好的融合方案



上下文感知的实时信息推荐

	元素 1	元素 2	元素 3	元素 4	元素 5
用户A					
用户B					
用户C					
用户D					
用户E					

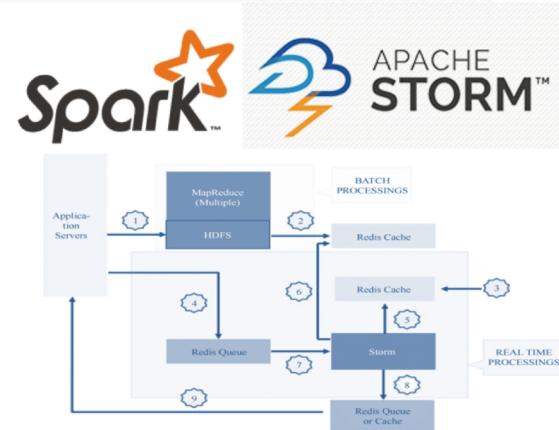
用户对元素的选择关系



协同式过滤 输入-输出 示意

- 向用户B推荐元素2
 - 向用户E推荐元素5
 - ...
 - ...
- 向用户推荐相关元素

- ❖ 针对图数据特点，对现有协同过滤方法进行优化，形成面向大规模图数据、基于实时协同过滤的信息推荐方法
- 协同式过滤的成熟研究成果和成功实践应用为该技术的研究提供了良好的技术积累
- ❖ 基于Spark、Storm等成熟的开源实时大数据处理平台，进行实时信息推荐系统的开发、实验验证、以及实际应用
 - 基于成熟的开源框架进行系统开发与应用，保证了研究成果的可用性

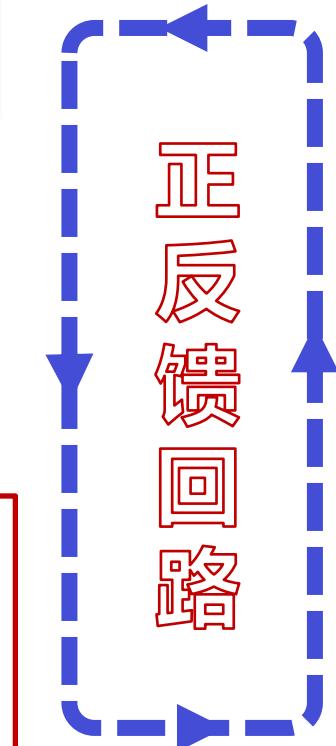
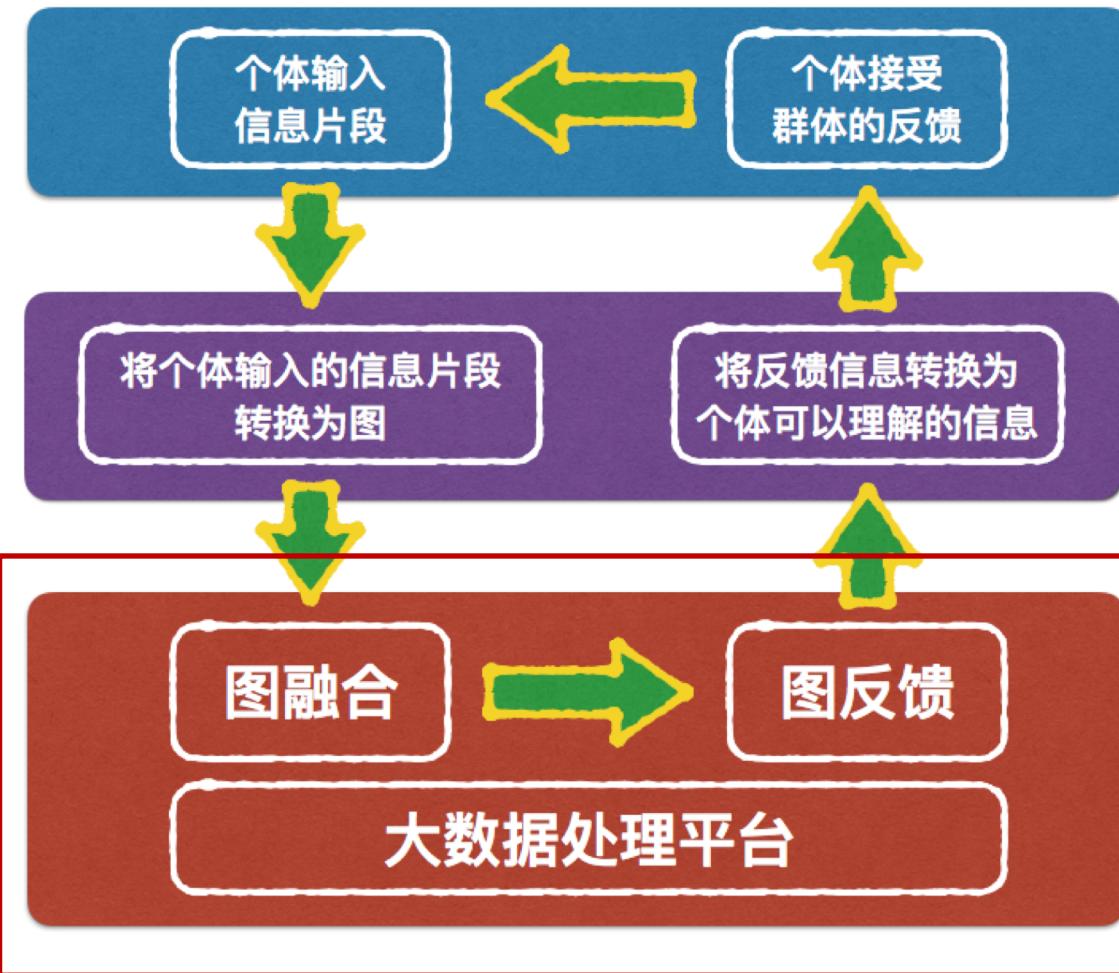


基于Storm的实时推荐系统框架

如何根据个体当前的信息为其推荐有价值的相关信息

基于群体智能求解问题的支持机制

《前端层》

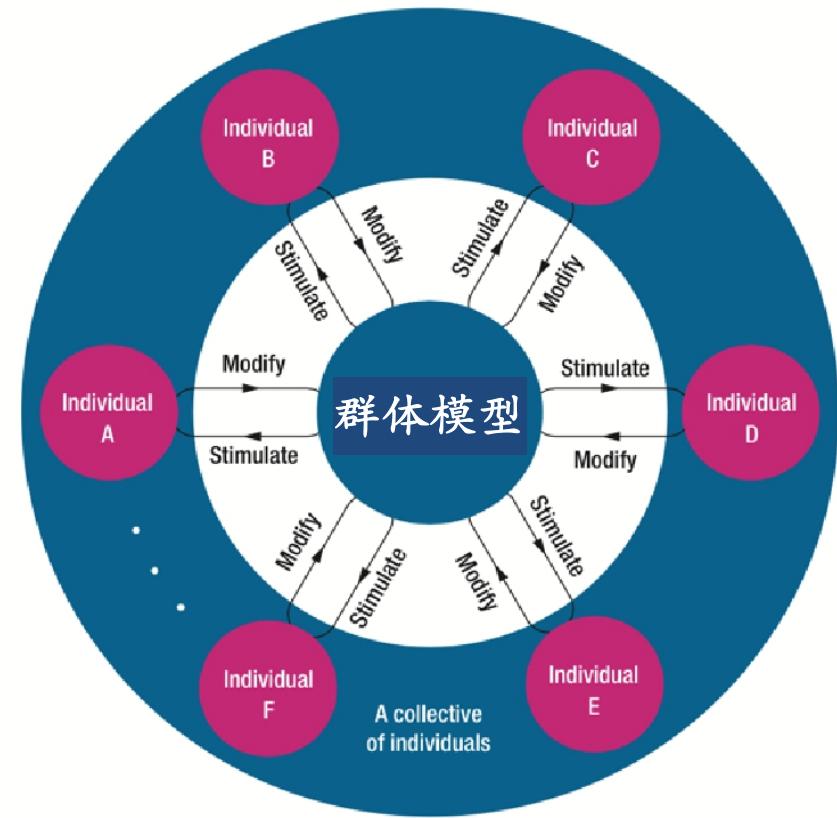


为各类软件问题的群智化求解提供共性技术支撑平台

基于熵最小化的大规模图数据的融合

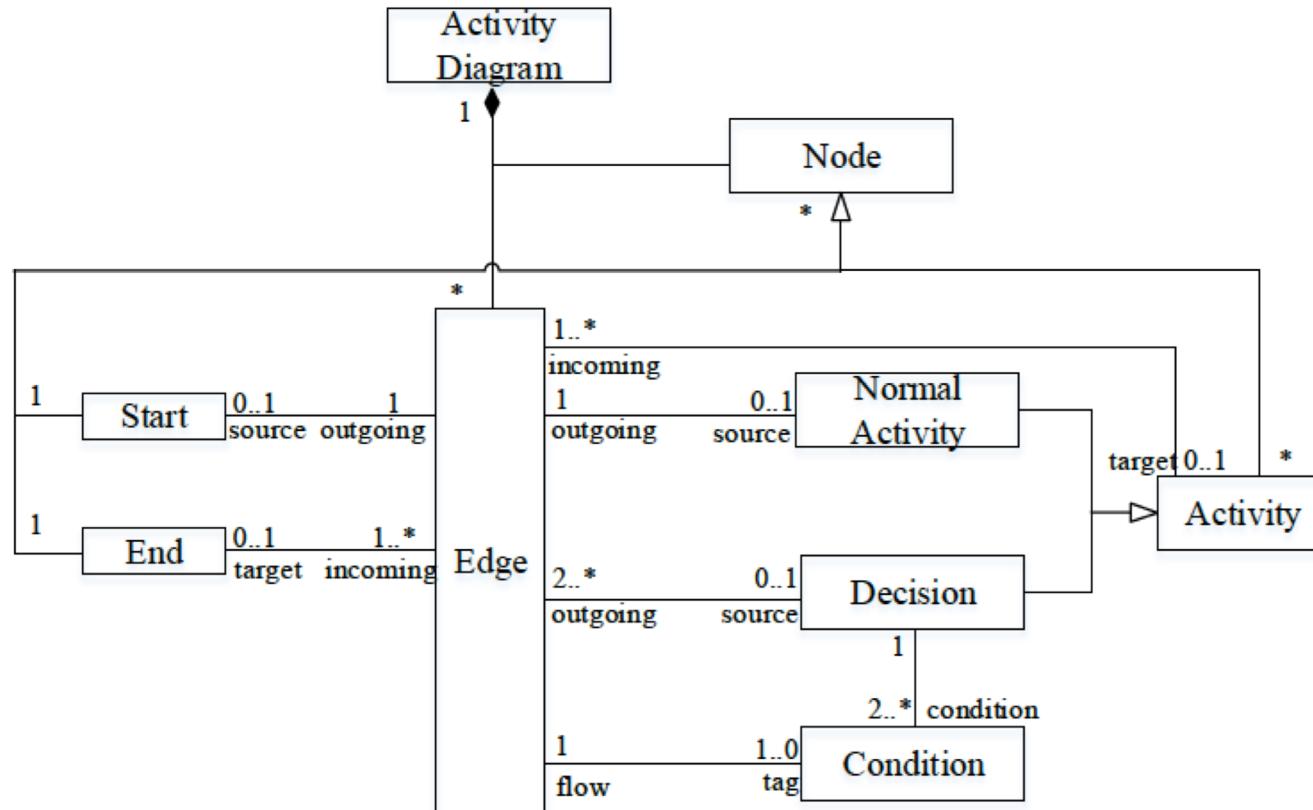
核心层：

1. 融合形成的群体模型
 - 反映群体的共识
 - 弱化群体中的个体偏见
2. 表达个性化需求的个体模型
 - 反映群体的多样性

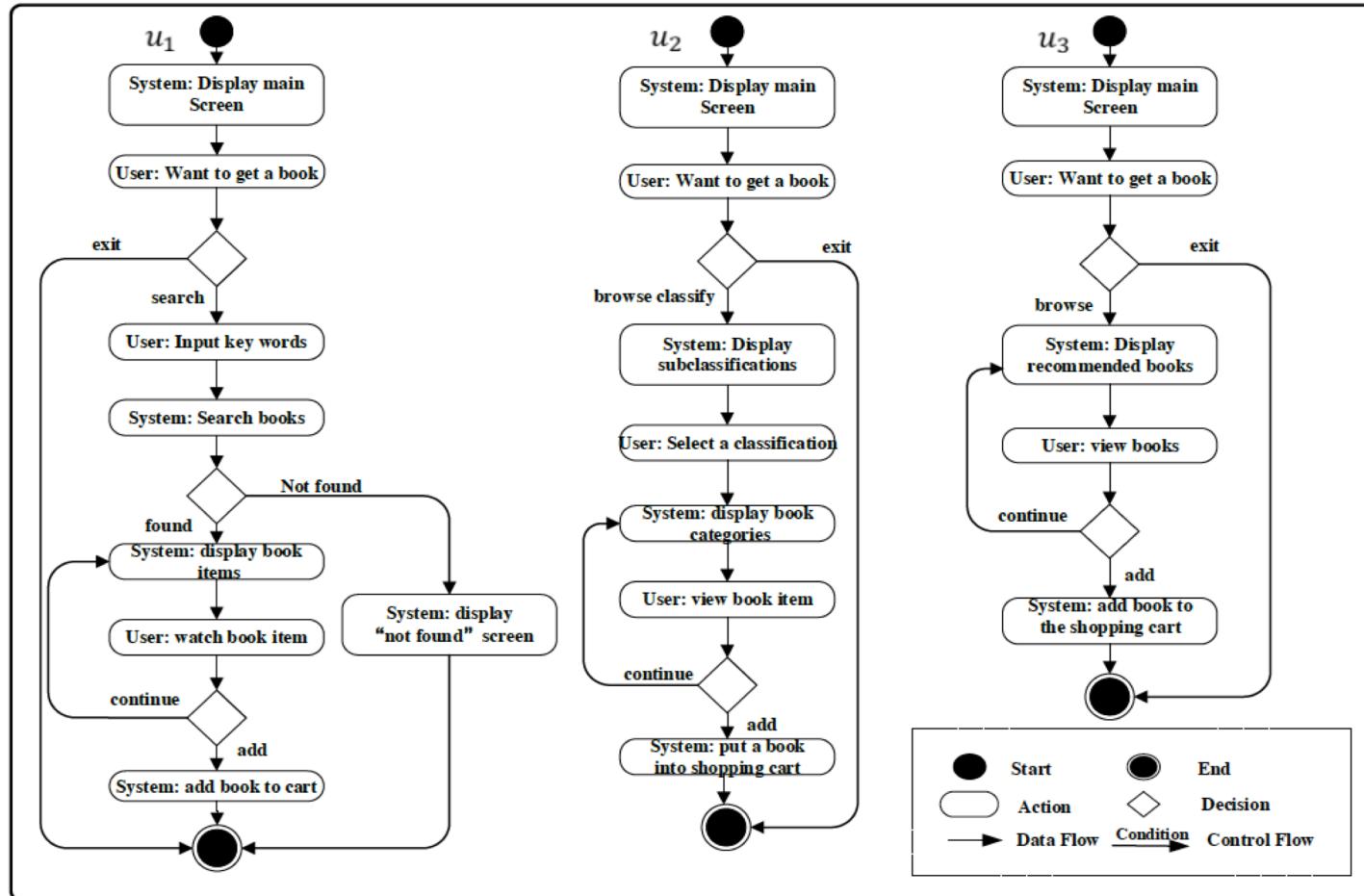


基于熵最小化的大规模图数据的融合

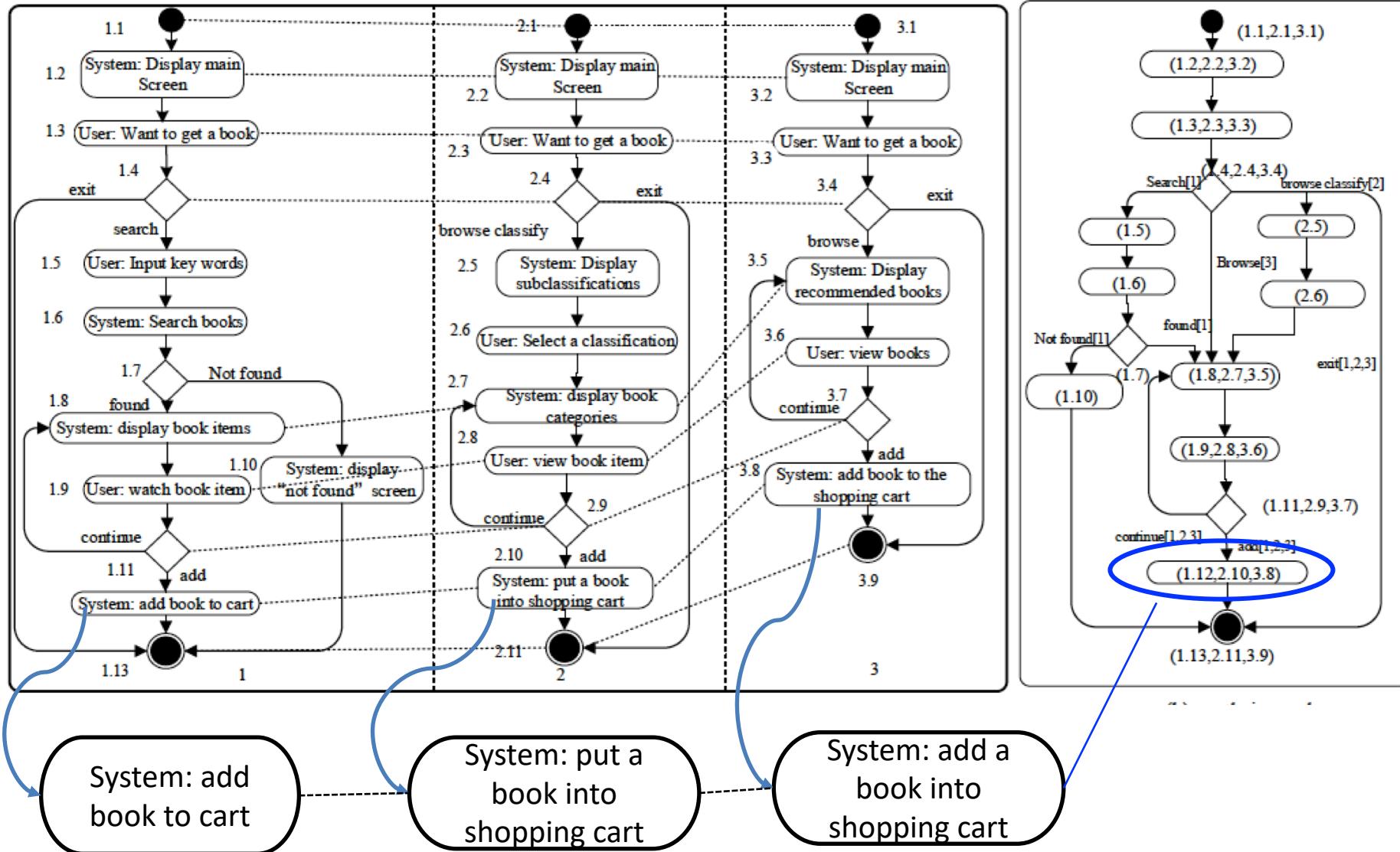
以活动图为例



基于熵最小化的大规模图数据的融合

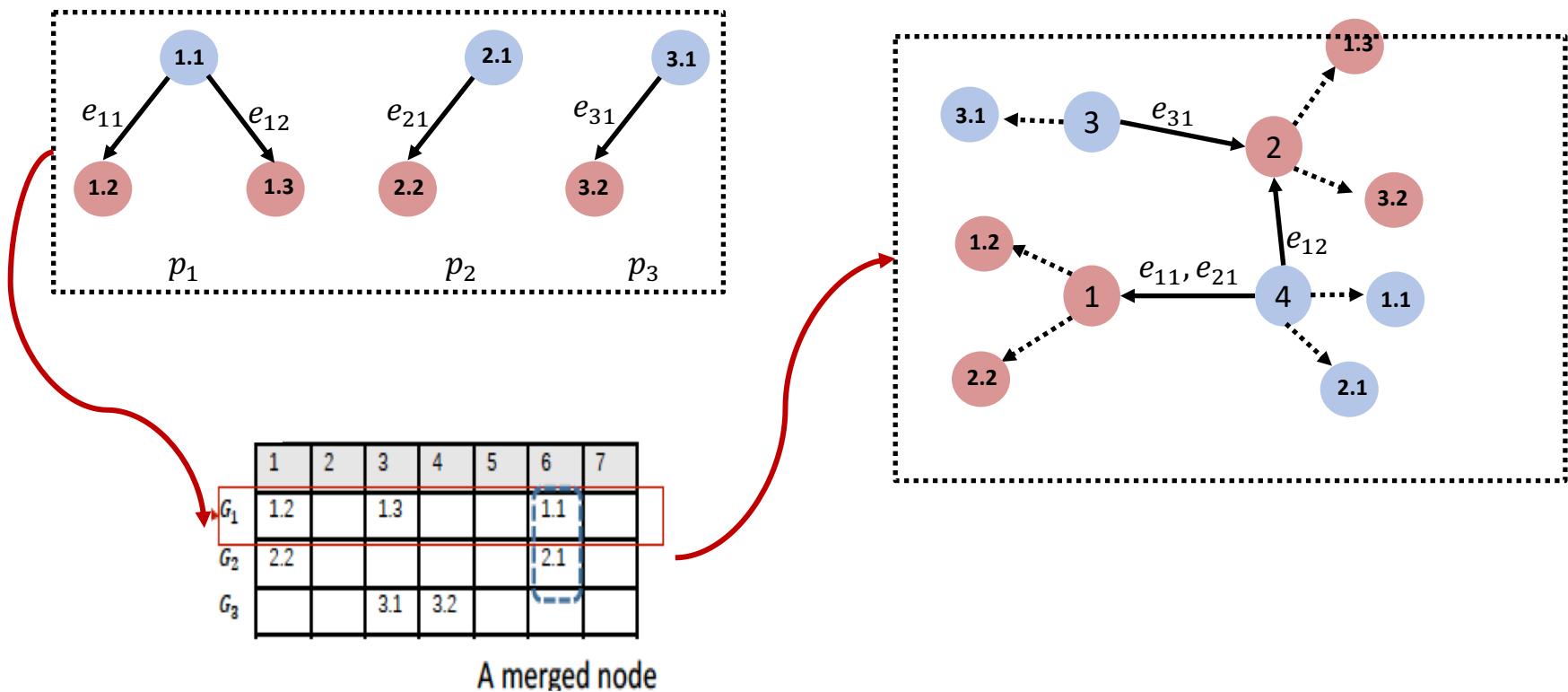


在线购书的活动图片段



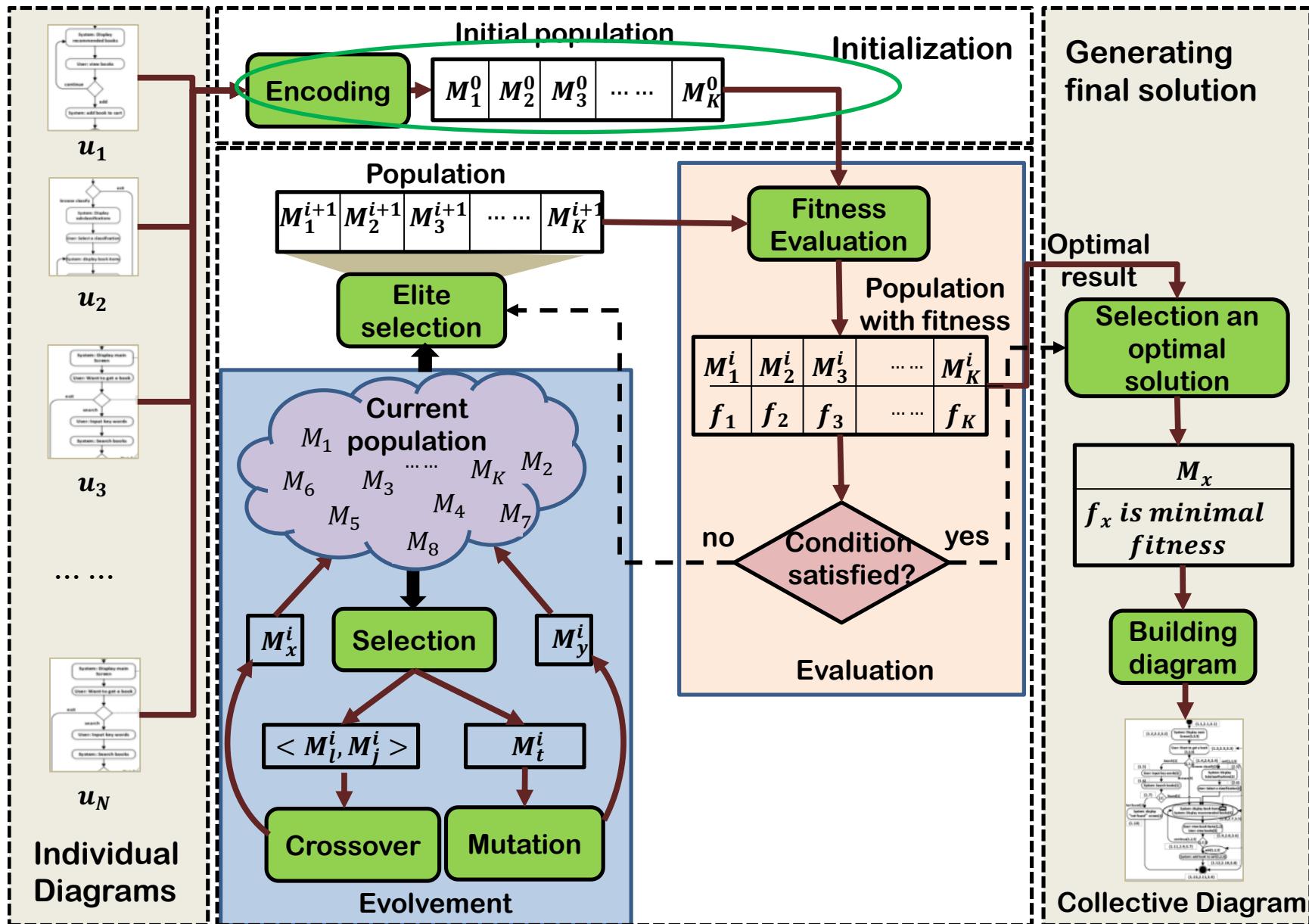
将个体图中高度相似的结点融合为一，且保留个体的相应信息

识别个体图的可融合结点

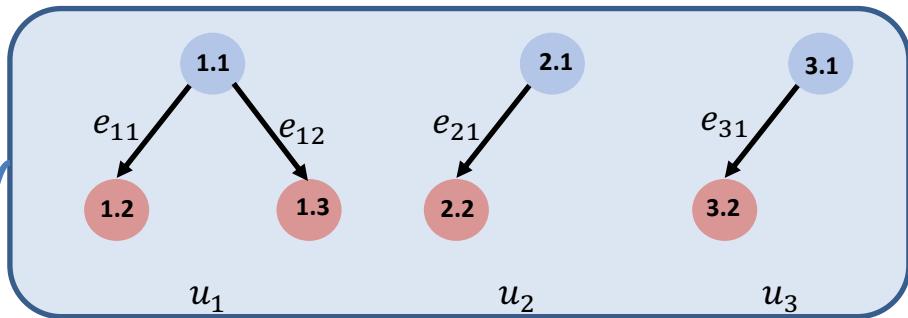


共具有 c 个结点的 N 个图，搜索复杂度为达 $O(c!^N)$

基于启发式搜索的活动图融合



编码与初始种群

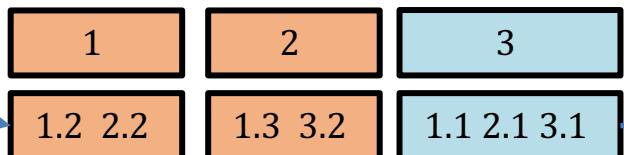


随机选择个体图中的结点，满足以下的约束：

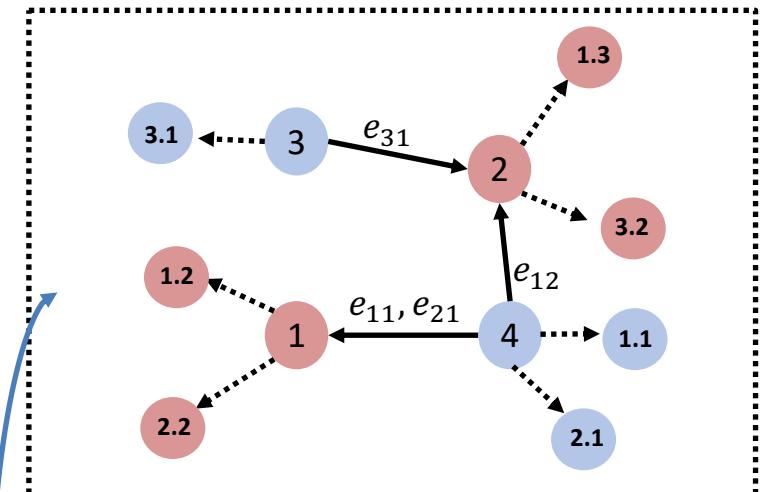
- (1) 具有相同类型
- (2) 同一基因 (box) 的组成元素 (可能等价/相似结点) 来自不同的个体图
- (3) 任意个体图的任意结点须出现某个基因中



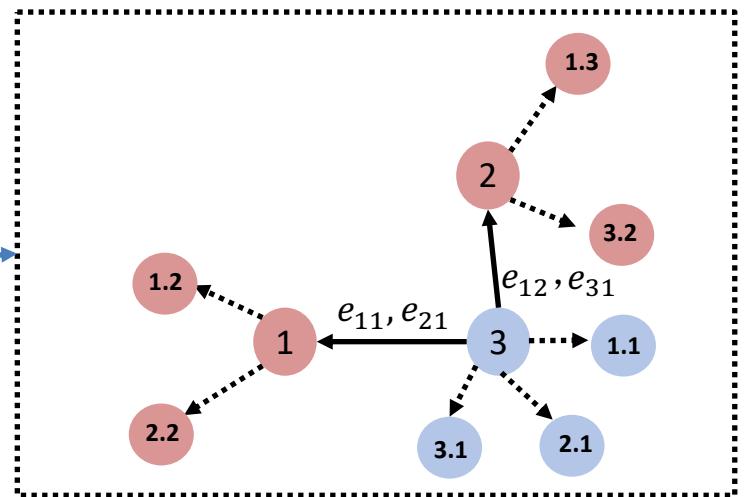
The encoding representation of individual I_1



The encoding representation of individual I_2



The labeled Graph(LG) of individual I_1

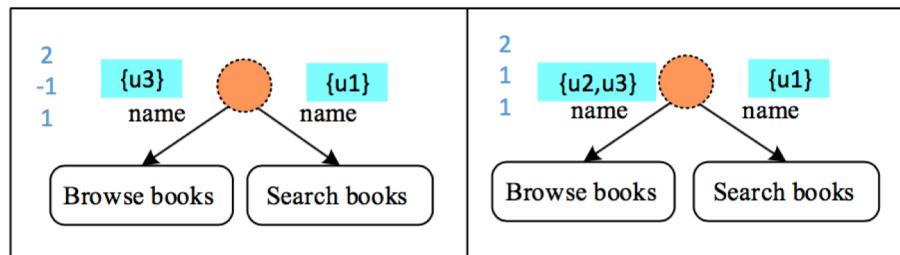


The labeled Graph(LG) of individual I_2

适应度函数

借鉴热力学与信息论的思路，定义广义熵作为fitness function 衡量结点的分歧程度

$$H(v_i) = - \sum_{k=1}^n (p(x_k) * \log(\sum_{t=1}^n p(x_t) s(x_k, x_t)))$$



其中，

- x_k : 可能分歧的结点
- $p(x_k)$: x_k 的概率 (与个体数目相关)
- $s(x_k, x_t)$: x_t 和 x_k 的相似度，综合考虑结点的语义和结构相似度

X_k	Browse books	Search books	概率分布
$v_1: p(x_k)$	1/2	1/2	结点相似度
$v_2: p(x_k)$	2/3	1/3	

$$s(browse books, search books) = 1/3$$

种群中一个LG的熵值即为所含所有结点的熵值之和

$$f(M) = |e| \cdot \sum_{i=1}^{N_V} (E(v_i) + \Delta - \Delta^{\frac{|p_{v_i}|}{|N|}})$$

$$\begin{aligned} H(v_i) &= -p(x_k) * \log(\sum_{t=1}^n p(x_t) * s(x_k, x_t)) \\ &= -\frac{1}{2} * \log(\frac{1}{2} + \frac{1}{2} * \frac{1}{3}) - \frac{1}{2} * \log(\frac{1}{2} + \frac{1}{2} * \frac{1}{3}) \\ &= 0.176 \end{aligned}$$

$$\begin{aligned} H(v_i) &= -\sum_{k=1}^2 p(x_k) * \log(\sum_{t=1}^n p(x_t) * s(x_k, x_t)) \\ &= -\frac{2}{3} * \log(\frac{2}{3} + \frac{1}{3} * \frac{1}{3}) - \frac{1}{3} * \log(\frac{1}{3} + \frac{2}{3} * \frac{1}{3}) \\ &= 0.158 \end{aligned}$$

融合效果的初步实验

- ATM
 - WithdrawnFund 5份
 - ValidatePIN 4份

WithdrawFund			ValidatePIN		
ID	#node	#edge	ID	#node	#edge
AD_1	17	16	AD_1	18	17
AD_2	16	15	AD_2	9	8
AD_3	10	9	AD_3	14	13
AD_4	12	11	AD_4	16	15
AD_5	14	13			

- mobile shopping 3份

	#nodes	#edges	#scenarios
u_1	60	73	13
u_2	62	74	14
u_3	65	83	16

融合效果的评估

Precision: the ratio of correct synthesized nodes *correctSN* (the number of correct synthesized nodes in SAD) to the total synthesized nodes SN (the number of all of synthesized nodes); measuring the quality of merge

$$precision = \frac{|correctSN|}{|SN|}$$

$$recall = \frac{|correctSN|}{|expectedSN|}$$

Recall: the ratio of the correct synthesized nodes to the expected synthesized nodes *expectedSN* (the number of expected synthesized nodes in an expected SAD); measuring the coverage.

case	E_{M_1}	E_{M_2}	M_1		M_2	
			<i>precision</i>	<i>recall</i>	<i>precision</i>	<i>recall</i>
<i>adding book to cart</i> (motivation example)	1382.13	1496.05	20/20	20/20	18/22	18/20
<i>ATM</i>	644.89	979.94	19/19	19/19	16/22	16/19
<i>Online shopping</i>	17281.97	17932.76	76/76	76/76	72/80	72/76

A few nodes in M2 are not synthesized. We call these nodes as "missed synthesized nodes". In these cases, no node is incorrectly synthesized into other nodes, so that there no "wrong synthesized node"

基于搜索的融合的稳定性



The stability of the merging algorithm measures whether it get a better solution (with decreased entropy value) after each generation's execution.

基于搜索的融合的规模可扩展性

#graphs	#nodes	#synthesizedNode	#generation	entropy	time(m)
5	110	22	129	0.0	0.29
10	220	22	330	0.0	1.64
15	330	22	614	0.0	8.34
20	440	22	1048	0.0	28.18

The running time is related with the number of Personal ADs and the number of their nodes

The base activity diagram from online shopping system has 22 nodes.
different sizes

Experiments of different sizes in the number (5, 10, 15 and 20) of PADs are conducted to evaluate the scalability of GA

需求驱动的 软件自适应

软件系统



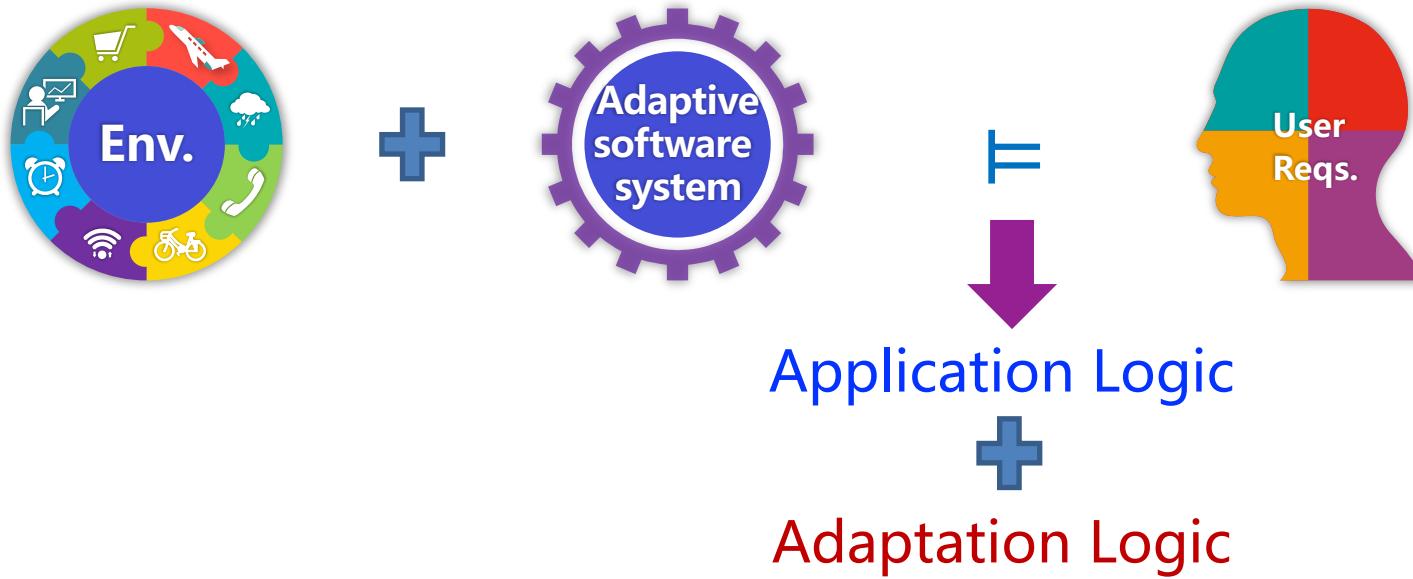
软件系统



Changes are inevitable, for either environment or user requirements

- Environment are dynamic and uncertain
- User requirements are volatile

自适应软件系统



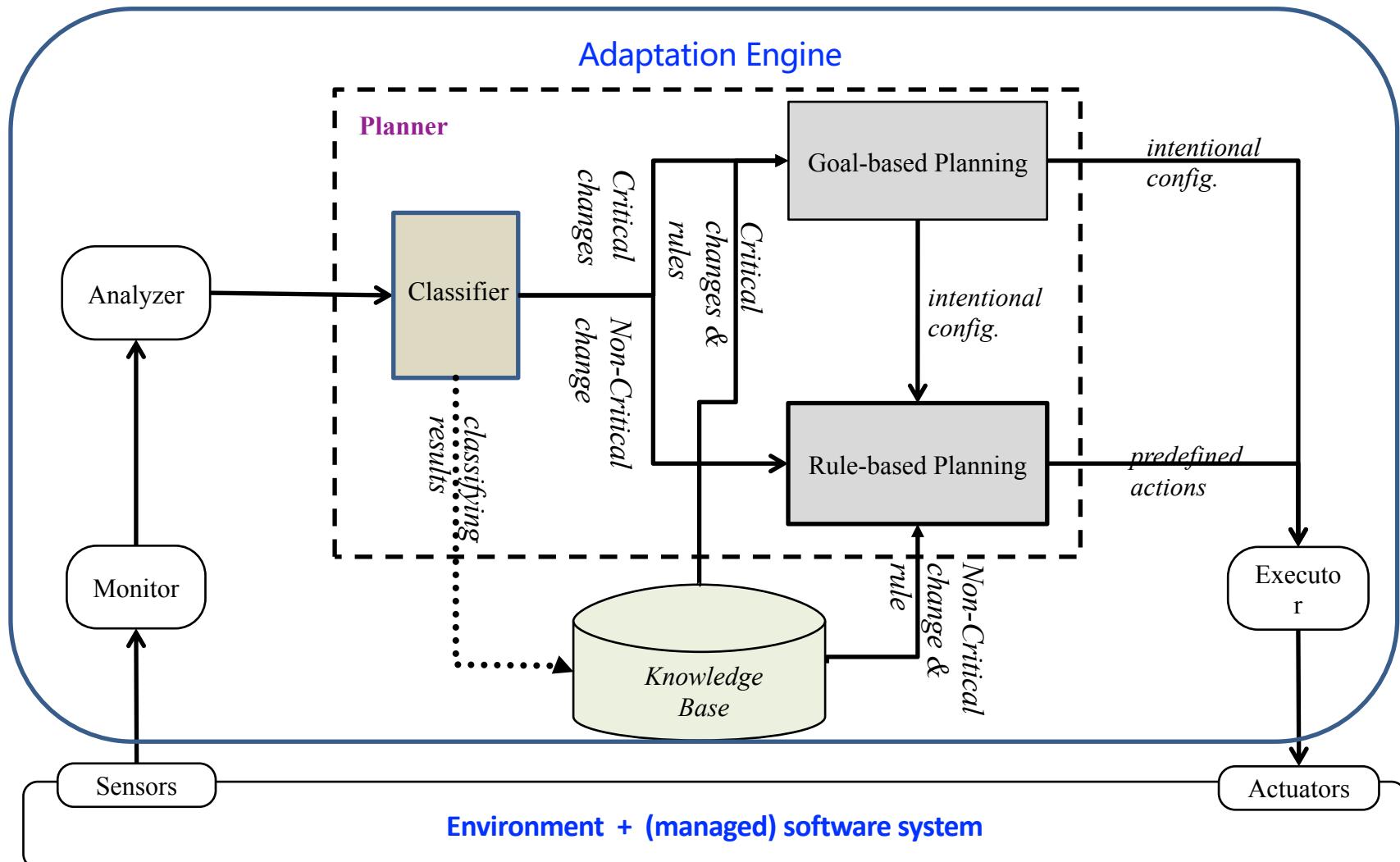
capable of *modifying* its properties and behavior (and beyond) at runtime
in response to both the *environment dynamics* and the *changes of user needs*.

自适应软件系统

- Assume the correctness with respect to the control system's functional requirements, and focus on problem of continuous satisfaction of the non-functional requirements

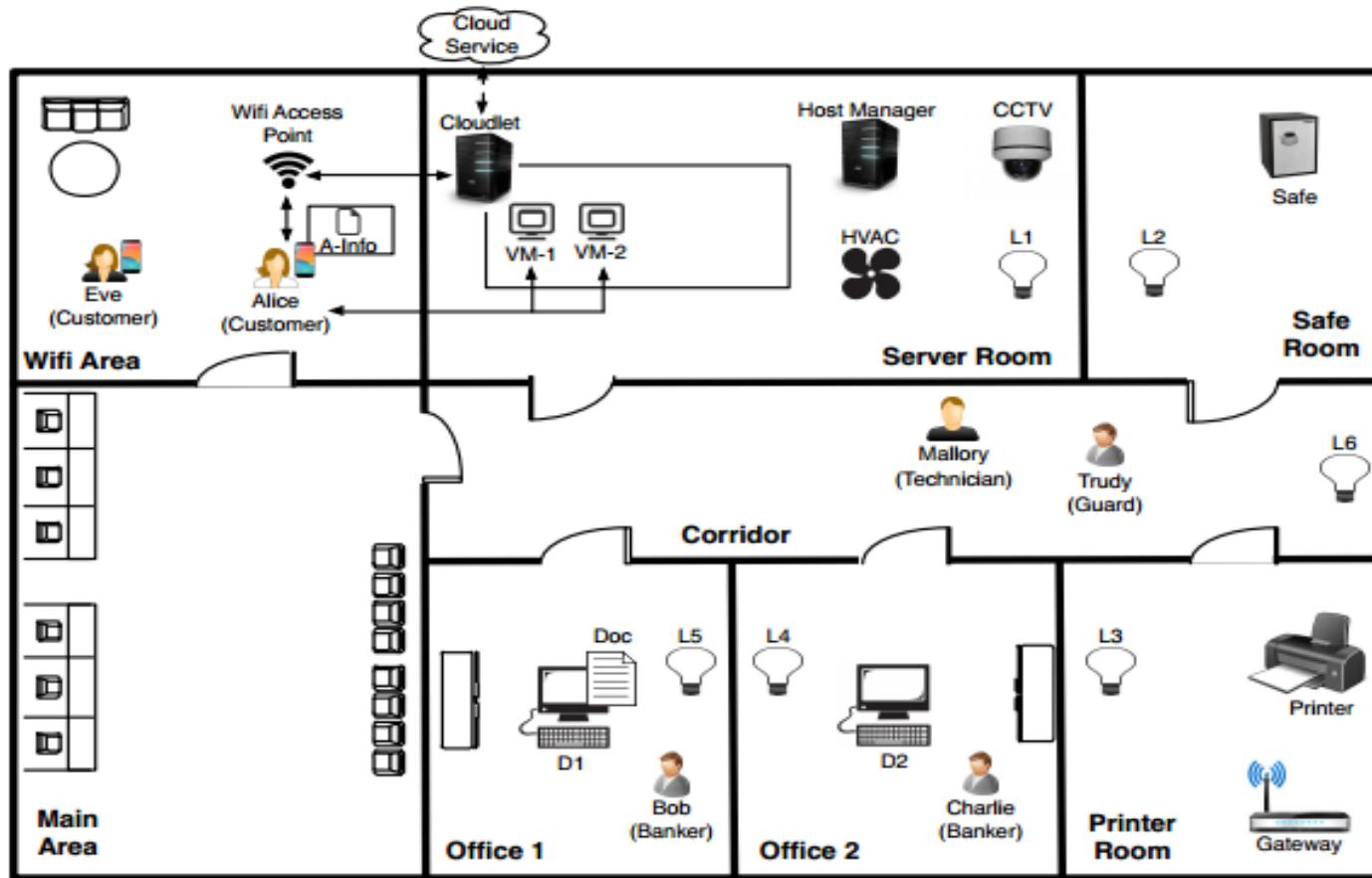


自适应软件系统

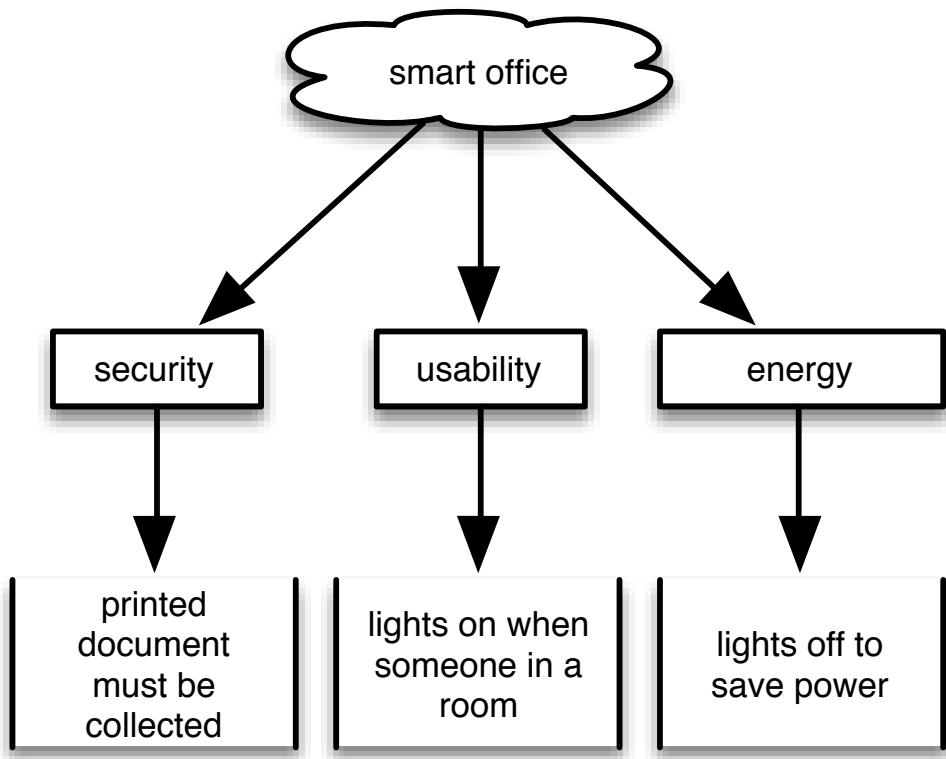


Adaptation Engine: a control loop with feedback from the system itself and the environment

Cyber-Physical Space: Smart office



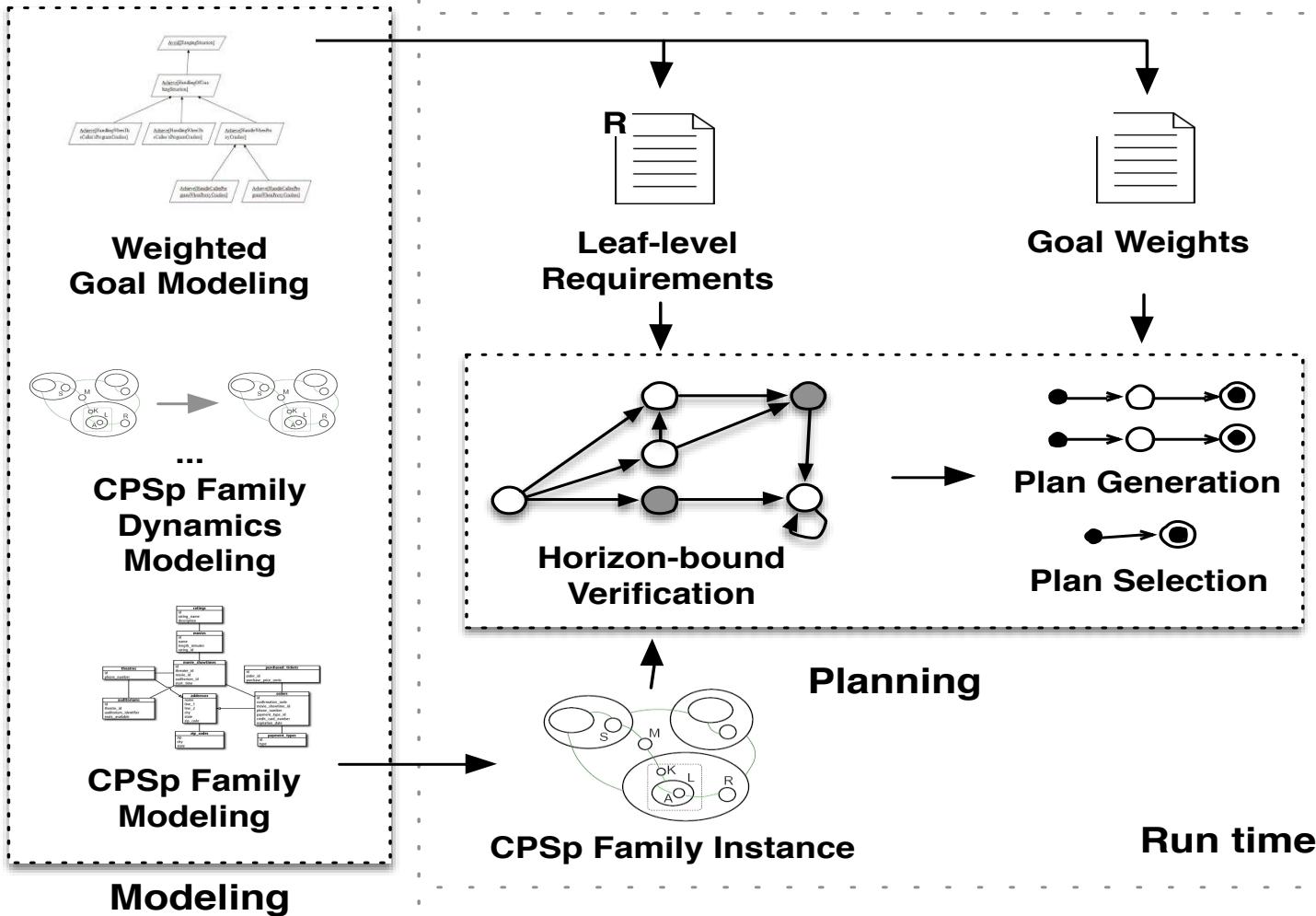
Cyber-Physical Space: Smart office



- 安全：访客不能接触到资料
- 节能：没人时关灯
- 可用性：清洁，房间亮度舒适

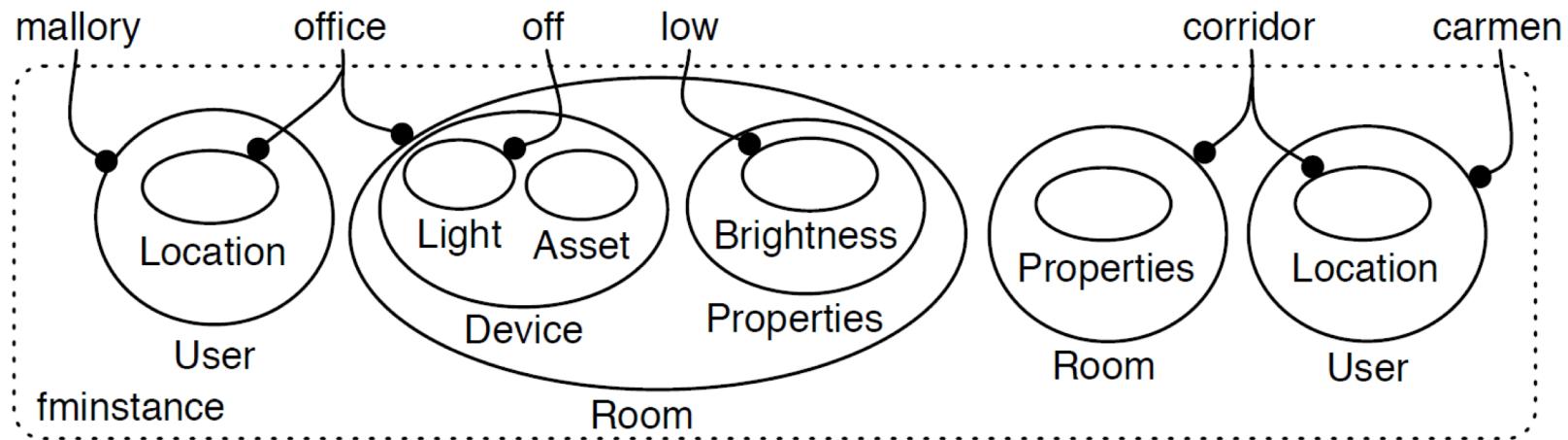
Cyber-Physical Space: Smart office

Design time



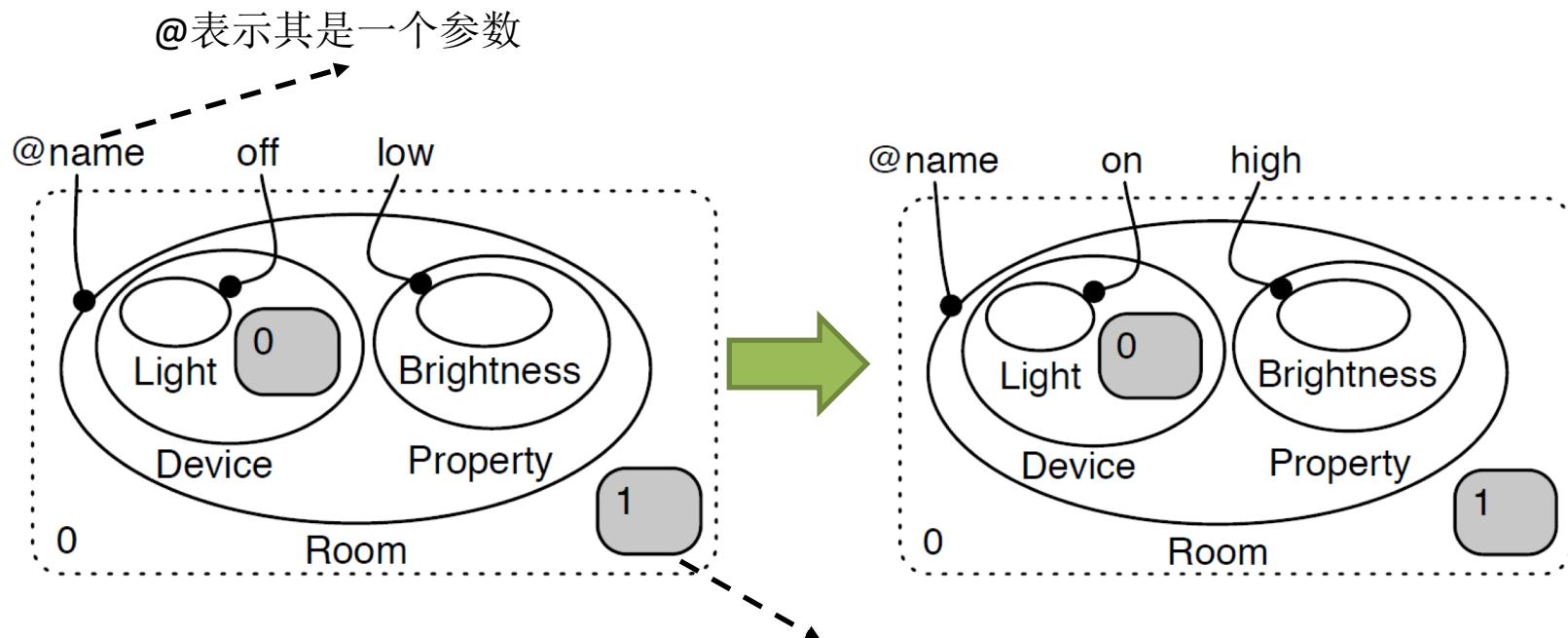
Cyber-Physical Space: Smart office

Bigraph刻画Office@Space 的状态



Cyber-Physical Space: Smart office

Reaction Rules刻画自适应规则 (ECA)



灰色部分表示规则不关心的部分，即可以匹配任何的pattern

当规则中的参数确定并应用到具体的状态中称为操作

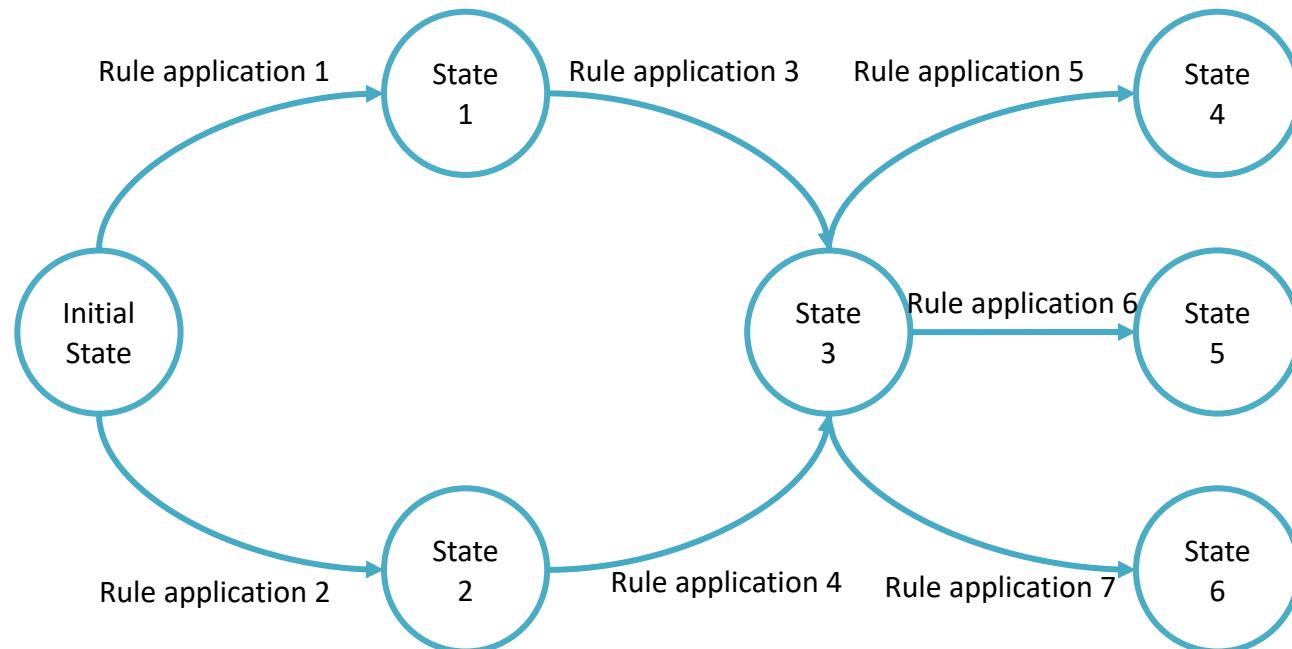
Cyber-Physical Space: Smart office

自适应系统的规划问题

- 给定
 - 目标模型 G
 - 规则集合 \mathfrak{R}
 - 初始状态 s_0
- 求一个操作序列 (a_0, a_1, \dots, a_n) , 使得 $G(s_n)$ 最大, 其中 $s_{i+1} = a_i(s_i)$

Cyber-Physical Space: Smart office

- 传统的精确搜索需要遍历和存储整个状态空间

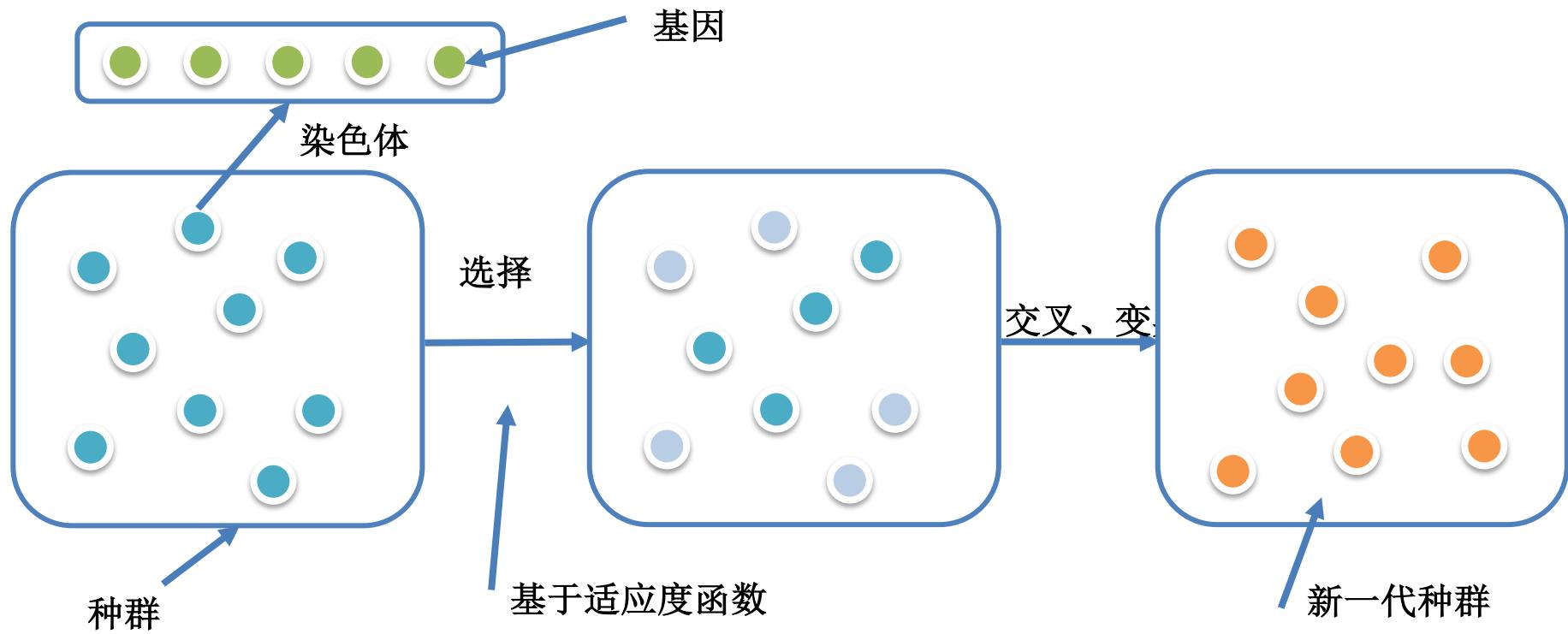


Cyber-Physical Space: Smart office

- 时间约束
 - 实时性：由于环境不断变化，自适应系统需要在限定时间内对环境做出响应
- 时间开销
 - 规划开销 —— 规划算法所花费的时间：每次evaluate一个操作都需变换，此部分是算法最费时间的步骤；
 - 执行开销 —— 与规划的结果有关：规划结果所产生的操作序列越长，需要执行的操作就越多，耗时也越多
- 规划约束 —— 算法搜索的操作的次数
- 执行约束 —— 算法产生结果的长度

基于多目标遗传算法的规划方法

- 无需存储中间状态，节省空间资源
- 启发式，可以减少时间开销
- 搜索时间可以控制，不随搜索空间爆炸，且在可控的时间内可以产生比较高质的解

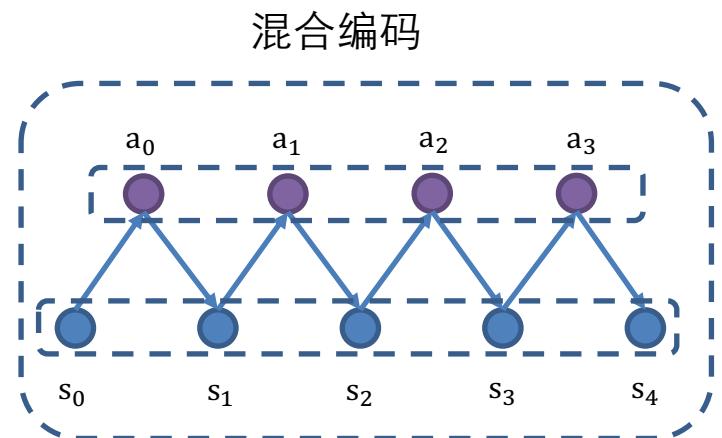


编码

- 基于状态的编码
 - 系统的状态实例作为基因
 - $g_i = s_{i+1}$
- 基于操作的编码
 - 操作作为基因
 - $g_i = a_i$
- 混合编码
 - 操作和状态共同作为基因
 - $g_i = (a_i, s_{i+1})$

基于操作的编码

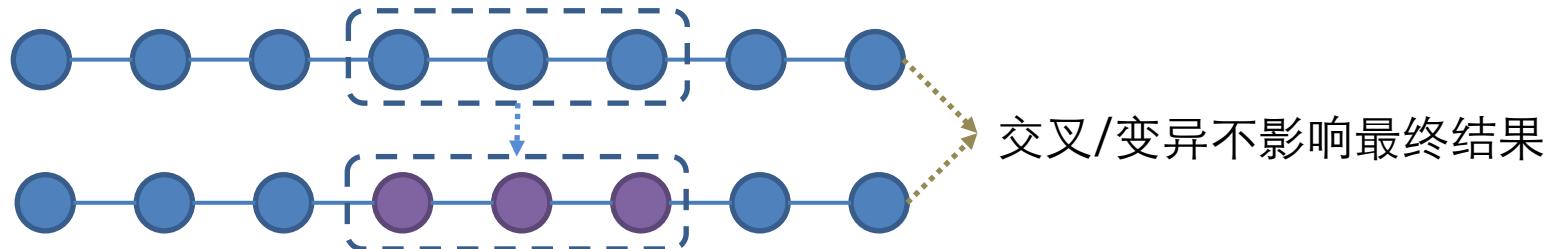
基于状态的编码



编码

- 由于目标的满意度只取决于最终的状态，中间状态的交叉和变异不会改变结果

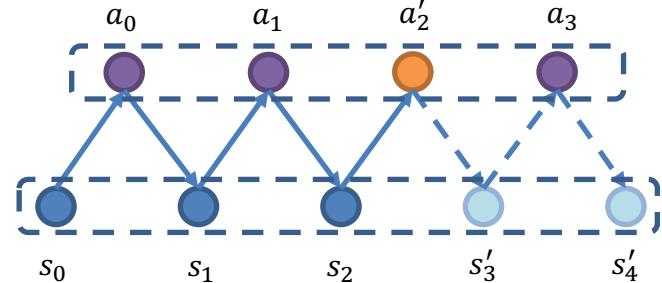
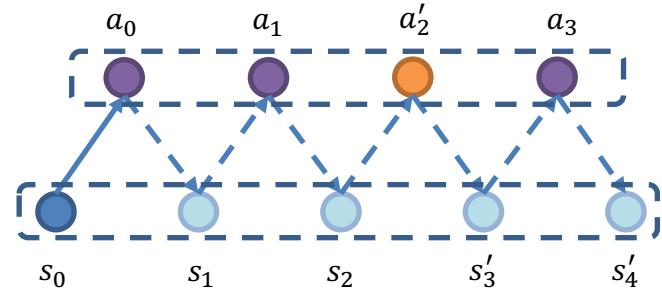
$$\text{eval}(s_0, s_1, \dots, s_k, \dots, s_n) = \text{eval}(s_0, s_1, \dots, s'_k, \dots, s_n)$$



- 交叉/变异的操作对象应该是Action，通过交叉或变异产生的新个体可能为非法个体：
 - 设原个体为
 - $a_0, a_1, \dots, a_{i-1}, a_i, \dots, a_n$
 - 交叉/变异后得到的个体为
 - $a_0, a_1, \dots, a_{i-1}, a'_i, \dots, a'_n$
 - i 为发生交叉/变异的位置
 - 需要验证从 a'_i 到 a'_n 的所有操作是否可行

编码

- 基于操作的编码没有记录执行操作 a_0, a_1, \dots, a_{i-1} 后的状态 s_i ，因此需要重新执行整个操作序列，即计算
 - $a'_n(a'_{n-1}(\dots(a_0(s_0))\dots))$
- 混合编码则只需要重新执行
 - $a'_n(a'_{n-1}(\dots(a'_i(s_i))\dots))$
- 故采用混合编码的方式

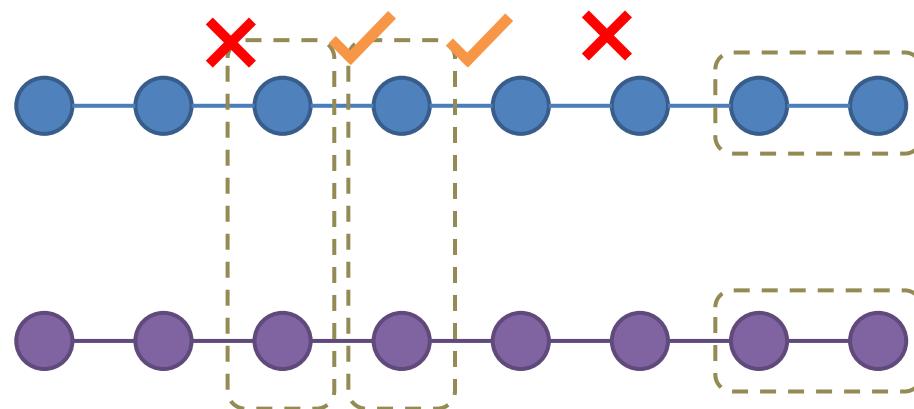


交叉

- 给定两个长度为n的个体编码 A 和 B，其中
$$A_i = (a_i^A, s_{i+1}^A), \quad B_i = (a_i^B, s_{i+1}^B)$$
- 保证交叉产生个体的有效性
 - 对于两个基因 $A_i = (a_i^A, s_{i+1}^A)$, $B_i = (a_i^B, s_{i+1}^B)$, 如果 s_i^A 不满足 a_i^B 的前置条件，或 s_i^B 不满足 a_i^A 的前置条件，则交换这两个基因产生的个体是非法的，即这个交换操作是非法的
- 降低交叉操作的时间开销
 - 减少重试的次数

交叉

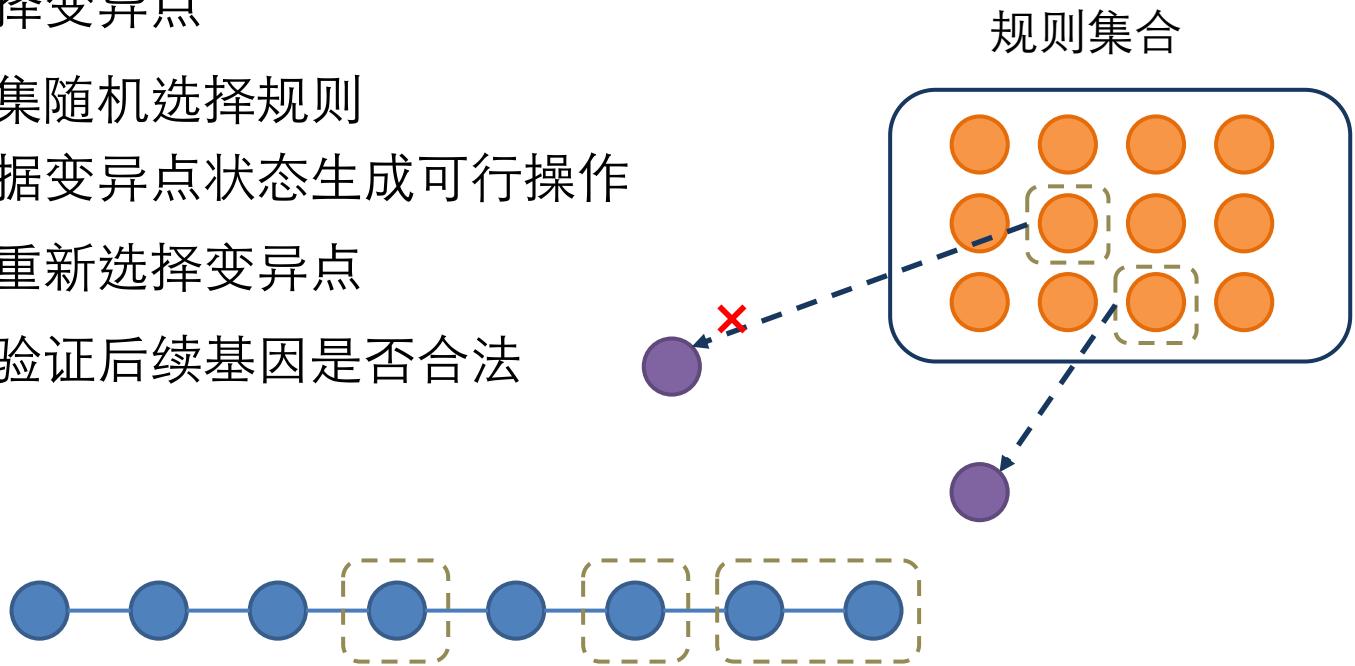
1. 随机选择交叉起点
2. 尝试交换两个基因
3. 非法则重新选择交叉起点
4. 合法则继续交换后续基因



变异

给定一个长度为 n 的个体编码 A , $A_i = (a_i^A, s_{i+1}^A)$, 规则集合 R

1. 随机选择变异点
2. 从规则集随机选择规则
3. 尝试根据变异点状态生成可行操作
4. 失败则重新选择变异点
5. 成功则验证后续基因是否合法



维持种群的基因多样性

选择与适应度函数

朴素适应度函数

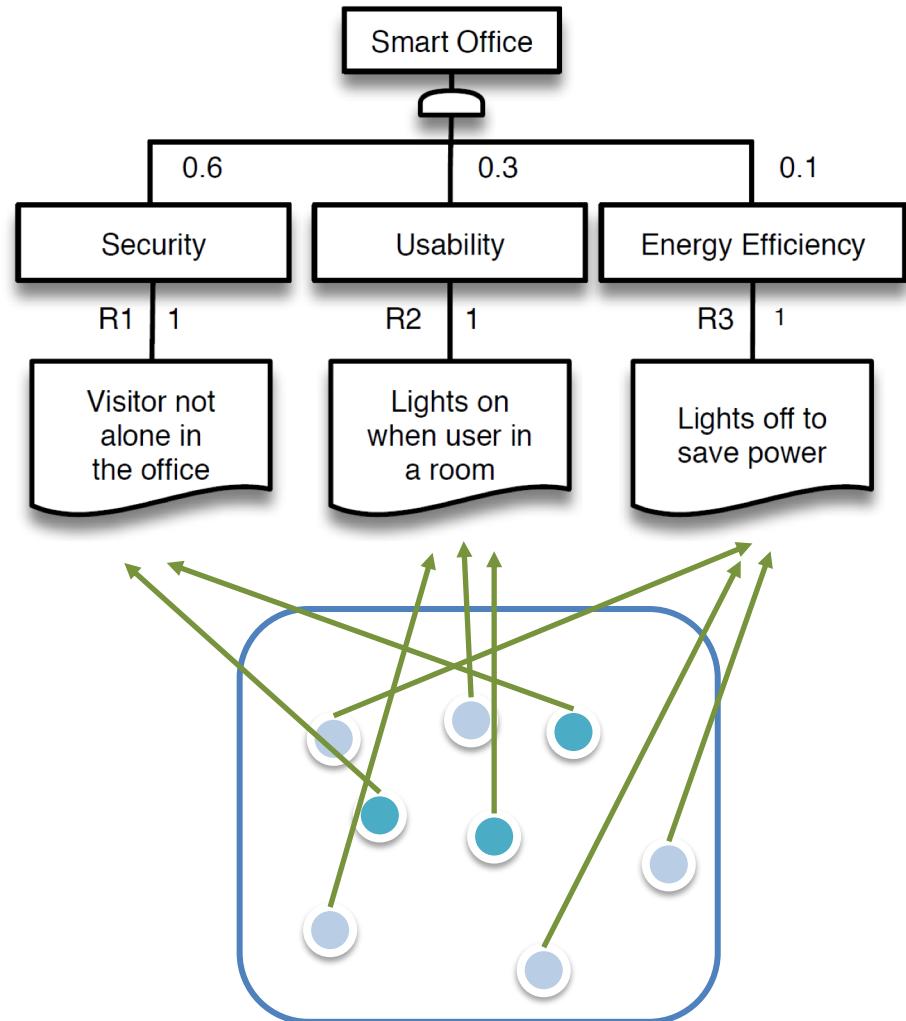
个体对最终目标的满意度

不足与问题：子目标权重的影响

- 根据叶目标对根目标的权重，满足不同的叶子目标对根目标的贡献也不尽相同；同等情况下，满足大权重的叶子目标的个体比满足小权重的叶子目标的个体有更高的适应度
- 在计算时间比较有限的情况下，优先满足大权重目标是有益的，会使算法在较短的时间内得到较优的解；但同时，大权重目标对小权重目标的压制会使得算法快速收敛到局部最优解，使得小权重目标最终也无法满足

适应度函数

- 满足低权重目标的个体在选择中处于劣势
 - E.g., 满足 R3 的个体都被淘汰，导致在下一个种群中很难有满足 R3 的个体。即使之后又通过交叉变异产生了满足 R3 的个体，也同样难以在选择竞争中存活下来



适应度函数

多目标优化问题

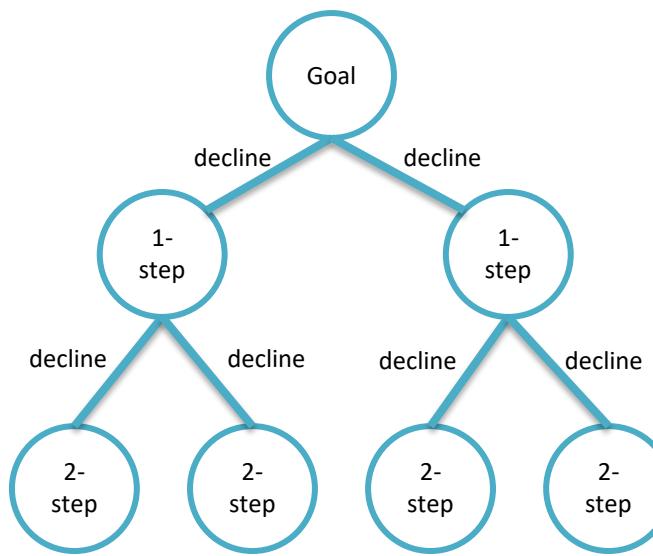
- 采用更低层次的目标的满足度作为适应度函数，缓解对低权重目标的不公平
- 将问题转化为多目标优化问题，利用多目标遗传算法（NSGA-II）的多样性保留策略，使得不同权重的目标都尽可能地被满足
 - 采用的层次不宜过低，也即目标不宜设置过多，否则会导致算法难以收敛，得不到有效解

目标复杂度

- 假设状态空间中总共有 S 个状态，其中能够满足叶子目标 g 的状态个数为 S_g ，则叶子目标 g 的复杂度为 $1 - \frac{S_g}{S}$
 - 直观上，目标复杂度衡量了一个目标被满足的难易程度，一个目标复杂度越高，就越难被满足，搜索得到满足该目标的解就越困难
 - 例如，“希望房间A的灯是关着的” 这一目标的复杂度就很低，因为不论系统处于哪种状态，A的灯只有开着和关着两种可能的状态，同时在处于开着的状态下，只需要“关灯”这一操作就可以满足目标
- 对于高复杂度的目标而言，通常需要一连串而非一个操作才能从当前状态到某一满足状态。与此同时，这一连串操作中的单独一个通常并不会导致任何一个目标被满足，而只是单纯的“接近”满足该目标的状态
 - 这类操作称为接近操作；接近操作所到达的状态称为中间状态
 - 执行接近操作不会带来个体适应度的提升，使得这些个体难以在选择中存活下来，最终导致低覆盖率的目标难以被满足

改进的适应度函数：反向奖励传播

- 为发现与奖励这些接近操作，从目标pattern出发，通过反向执行规则，得到一系列奖励pattern，基于与目标pattern 的距离赋予奖励值，得到pattern tree
- 奖励函数
 - $reward(p) = rate^i$, i为 pattern p 在树中所处的层次， $0 \leq rate < 1$ 是衰退率
- 在评估一个状态对某一目标满足程度时，自上向下广度优先遍历pattern tree，若满足其中一个pattern，则将其对应的奖励值作为适应度



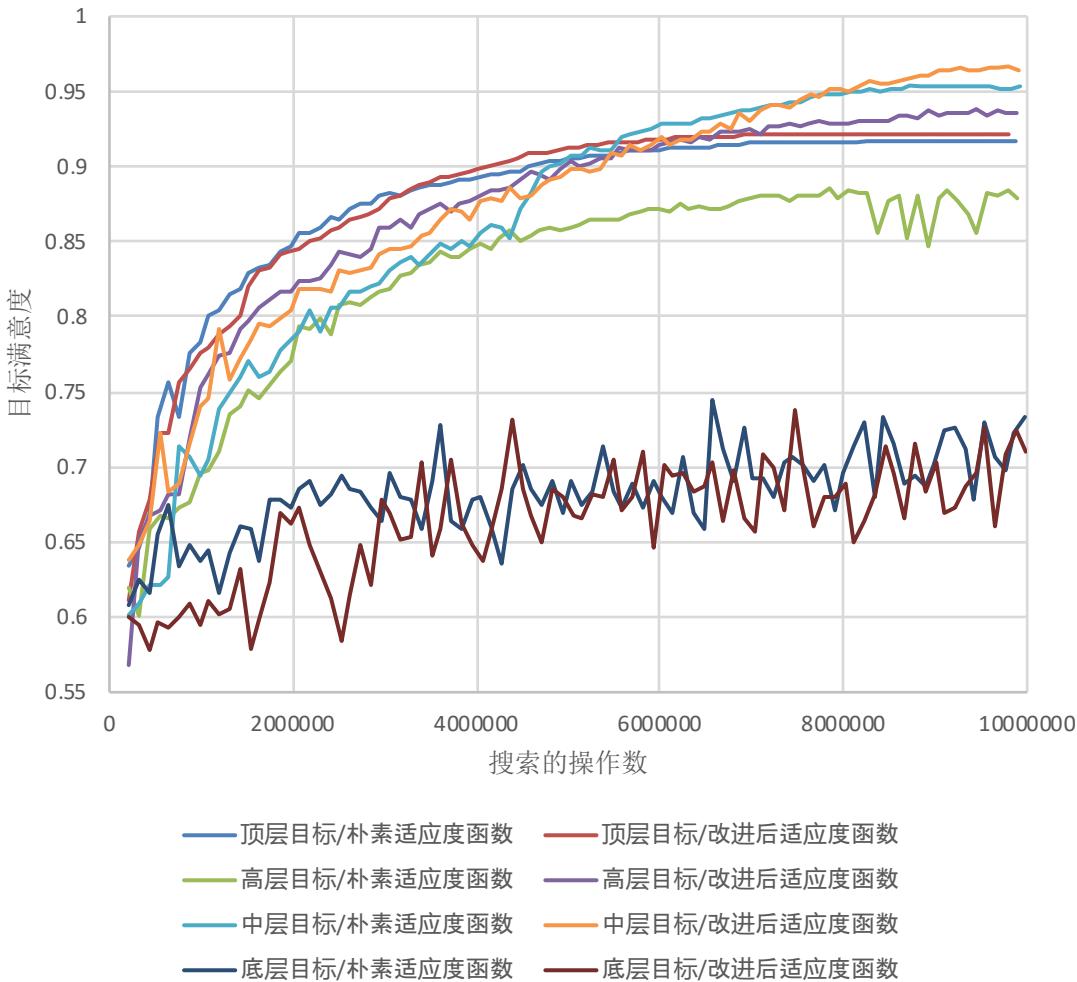
初步实验

- 实验设置
 - 场景
 - 40个房间
 - 692个实体
 - 38 个叶子目标
 - 算法 : NSGA-II
 - 种群大小 : 1000
 - 执行约束 : 100
 - 规划约束 : [0, 10000000)



实验初步结果：时间

遗传算法实验结果



- 采用不同配置运行遗传算法的结果，同一配置运行10次取平均值
- 采用的目标层次越低，算法收敛得越慢，但得到的最终解通常越好
 - 采用中层的目标效果最好
 - 采用顶层的目标收敛最快
 - 采用底层的目标算法难以收敛
- 在采用同层次的目标下，反向奖励传播可以提升算法的效果

实验初步结果：层次&衰退率的影响

- 最优满意度

传播衰退率\目标层次	0	1	2	3
0(无奖励传播)	0.9109	0.8877	0.9595	0.6803
0.1	0.9198	0.9333	0.9688	0.7331
0.4	0.9126	0.917	0.9692	0.6745
0.6	0.8989	0.927	0.9626	0.7398
0.9	0.88	0.9097	0.9676	0.7101
0.95	0.8591	0.9138	0.9658	0.6886

- 最终种群平均满意度（几何平均）

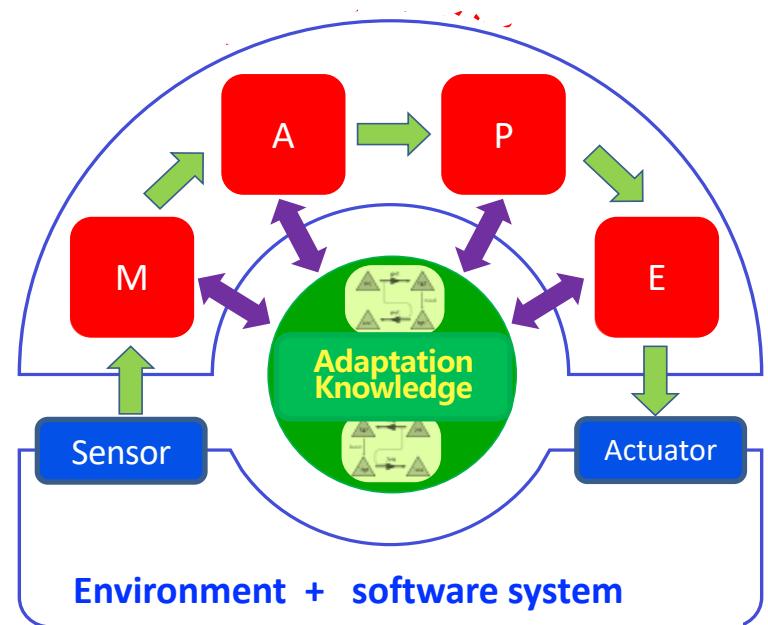
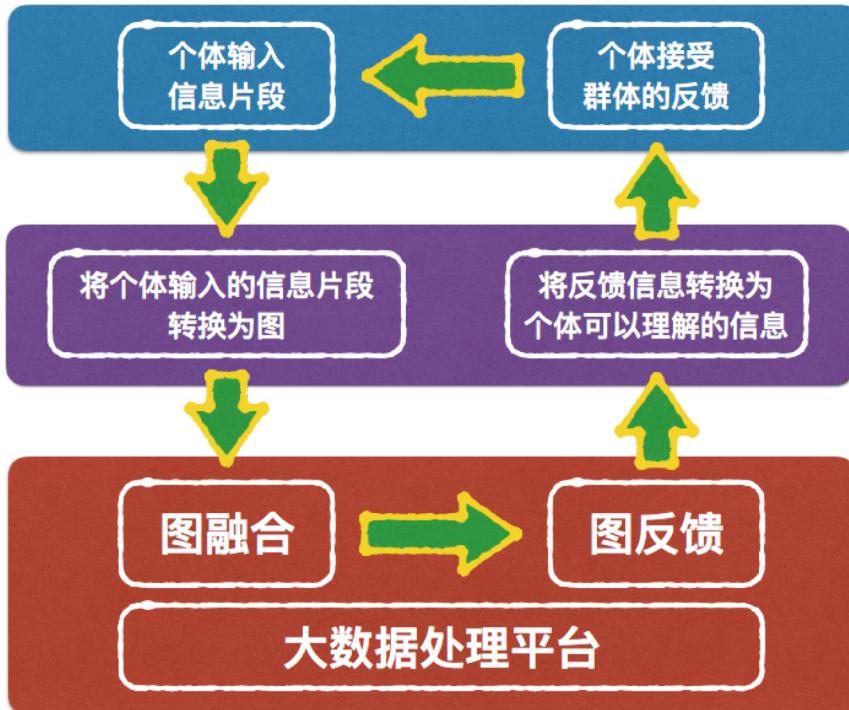
传播衰退率\目标层次	0	1	2	3
0(无奖励传播)	0.8938	0.7591	0.7357	0.2928
0.1	0.9034	0.9058	0.6316	0.2822
0.4	0.9046	0.8979	0.7042	0.2863
0.6	0.8812	0.8988	0.6702	0.2849
0.9	0.8591	0.8775	0.6488	0.283
0.95	0.8424	0.8775	0.6753	0.2832

实验初步结果分析

- 采用低层次的目标会对最优解有比较大的提升
 - 层次不宜过低，否则会使算法无法收敛
 - 在实验的例子中，采用第二层的目标效果最好
- 反向奖励传播总体上对最优解有提升
 - 衰减率不宜过高，否则对效果没有提升甚至可能导致下降，原因是这会导致算法更大概率地停留在中间状态，反之目标被满足的可能性就降低
- 对最终种群的平均满意度来说，两种改进均没有提升，反而有所下降
 - 这是由于改进前的算法更容易收敛，使得最终种群中的个体都维持在一个较优的满意度，但是最优个体不突出

总结

- 搜索技术在需求工程中的初步应用
 - 需求融合
 - 软件自适应系统的规划



Q & A

谢谢！