# 智能环境下基于雾计算的推理节点优化分配研究

汪成亮,黄心田
(重庆大学计算机学院,重庆 400044)

**摘 要:** 智能环境传统的规则推理机制中,网关内布置的推理机从各种传感器中获取推理所需数据并与规则库相匹配,承担整个推理工作. 本文利用 Rete 算法将规则构建为推理网络,并结合雾计算的概念将 Rete 推理节点分配至环境内配置的智能节点中协同推理以减轻网关负载,由此推理节点的分配成为关键,分配不合理将导致资源利用不平衡及响应延迟. 本文利用活动影响下规则触发的规律设计了活动聚类算法 CoA(Clustering of Activities)对活动聚类后分别建立其推理网络,计算出智能节点之间的最短路径后将结果代入针对其层次延迟性而设计的分配算法 AAoRN(Allocation Algorithm of Rete Inference Nodes),从而将推理节点最优分配至各个智能节点. 理论分析和实验结果表明,本文机制在有效利用智能节点资源的同时降低了大致 55% 的延迟.

**关键词:** 智能环境; 规则推理; Rete 算法; 雾计算; 活动模式
**中图分类号:** TP182 **文献标识码:** A **文章编号:** 0372-2112 (2020)01-0035-09
**电子学报 URL:** http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2020.01.005

# Study on Optimal Allocation of Inference Nodes for Fog Computing in Smart Environment

WANG Cheng-liang, HUANG Xin-tian
(*College of Computer Science, Chongqing University, Chongqing* 400044, *China*)

**Abstract:** Under traditional rule-based reasoning in smart environment, the inference engine deployed in gateway collects data from various sensors to match the rules, which undertakes whole reasoning work. In current research, we use Rete algorithm to construct an inference network by rules and then allocate the Rete inference nodes to smart nodes for collaborative reasoning based on fog computing, therefore, the allocation mechanism becomes crucial. In this paper, we utilize the regularity of rules triggered under the influence of activities to design an algorithm CoA(Clustering of Activity), which clusters activities and respectively constructs the inference networks, subsequently, we calculate the shortest path between smart nodes and substitute the results into AAoRN(Allocation Algorithm of Rete Inference Nodes), which is proposed to overcome hierarchical delay for optimally allocating each inference node. Theoretical analysis and experimental results show that the proposed mechanism efficiently utilizes the resources and has reduced the delay by about 55%.

**Key words:** smart environment; rule-based reasoning; Rete algorithm; fog computing; activity pattern

## 1 引言

近年来随着人们对便捷生活环境的向往,智能环境由此诞生. 由于传感器的活动识别具有数据获取自由、抗干扰的优点[1],智能环境中可布置各种传感器以获取周围环境因素及行为模式的数据,再将其用于分析识别人类活动[2],其基于规则推理系统的实现如图 1 所示. 图 1 左将相关知识转化为机器可理解的产生式规则以构建规则库[3],其结构类似于“IF-THEN”,图 1 右中传感器从周围环境获取数据并处理后存储在事实库中,从而推理机匹配两者推断出结论[4],随后更新事实库,并对用户响应或输出到相关控制设备,实现了智能环境的目的.

作为一种高效的模式匹配算法,Rete[5]在推理机中将产生式规则编译成推理网络,从而数据在网络中逐层匹配后过滤,到达终端节点则触发相应规则. 图 2 是

收稿日期:2018-01-16;修回日期:2019-04-21;责任编辑:孙瑶
基金项目:国家自然科学基金(No. 61672115);重庆市社会事业与民生保障科技创新专项(No. cstc2017shmsA30003)

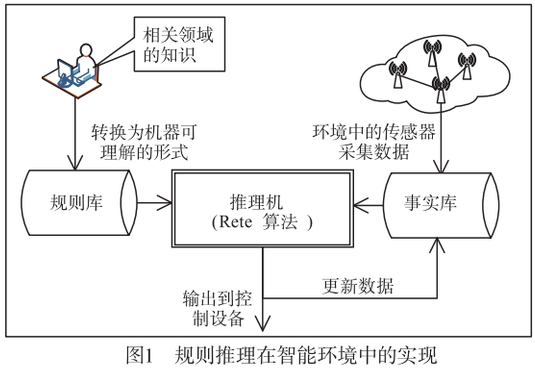
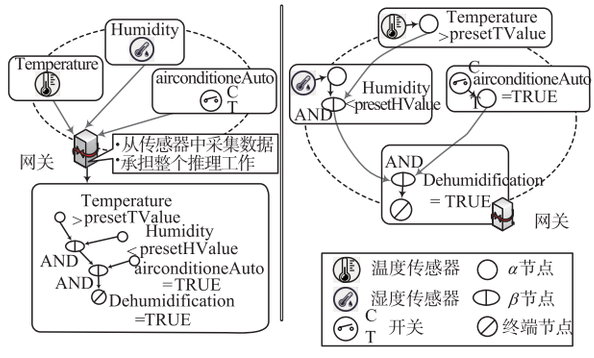



图1 规则推理在智能环境中的实现

由规则“IF Temperature > presetTValue AND Humidity < presetHValue AND airconditioneAuto = TRUE THEN Dehumidification = TRUE”构建的推理网络的两种分配机制. 传统机制如图 2($a$),整个推理网络被建立在网关中,数据量过大时会导致巨大的数据传输量和响应延迟. 图 2($b$)利用雾计算[6]的概念将网关的推理工作部分迁移至智能节点(部署于环境中的嵌入有各类传感器、通讯模块、控制模块的节点),推理节点被分配至智能节点中进行协作推理从而提前过滤大量数据以减轻网关负载,这便是智能环境下的分布式推理[7].




($a$) 集中式推理网络 ($b$) 分布式推理网络

图2 智能环境下的集中式与分布式Rete推理网络

然而,分布式推理机制仍面临以下挑战.

①资源平衡问题. 智能节点设备的存储与内存有限[8],推理节点的不合理分配会导致部分设备过载,资源利用不充分.

②实时性问题. 由于推理节点被分配至不同的智能节点,推理节点之间的传输会带来额外耗时,若分配机制未能结合当前环境的智能节点网络,将会加重系统响应延迟.

因此,本文发现利用以下特点可优化推理节点分配机制.

①由于智能环境下的规则是用于分析识别某种活动[9],活动的影响使规则的触发具有一定规律. 同一活动的推理节点在同一时间段被频繁触发,而其余相关性较差活动的推理节点虽有数据传入,但匹配不会成功,将这样的推理节点分配至同一个智能节点就不会抢占资源. 由于智能节点资源受限,可将该特点运用于分配算法的优化.

②Rete 推理网络中数据是逐层匹配过滤的,而由于分布式推理中的推理节点是分散的,每个输入数据的到达时间不同,更迟的输入到达后才能开始匹配,将这个特点称为层次延迟性.

为结合上述特点将推理节点更优地分配至智能节点,从而在克服资源平衡问题的同时进一步提升推理实时性,本文主要工作如下:

①设计活动聚类算法 CoA,利用活动类型、规则、智能节点三者的模型关系以及上述特点,推算出活动类型之间基于其触发节点的 Jaccard 距离,在分配前对活动聚类后分别建立其推理网络,便于后述分配.

②设计推理节点分配算法 AAoRN,此算法在保证资源平衡的同时克服了层次延迟性,逐层将推理节点分配至智能节点.

③设计仿真实验,在不同的规则数目、不同的仿真环境下比较本文提出的机制与集中式、普通分布式的优劣. 结果表明,本文提出方法使资源利用率提升了 2 到 3 倍,采用 AAoRN 算法相比普通分布式的延迟降低了 40% 左右,将 CoA 和 AAoRN 算法相结合比普通分布式的延迟降低了约 60%.

## 2 相关研究

以传感器为基础的活动识别目前已经有一定的研究成果,如 Non-Alisavath K 等[2]利用上下文感知的多传感器技术来捕获用户的移动,Tomoya K 等[10]曾将分布式 Rete 算法运用到电器能源管理领域,但阐述深度不够. 本人以往的研究已实现将环境上下文分为原子上下文和复合上下文,并提出将子规则模式分配至智能节点的可能性,因此 Rete 推理节点分配机制的优劣就成为了关键因素.

## 3 模型建立

### 3.1 智能节点、规则、活动类型

SNode = $\{S_1, S_2, \cdots, S_n\}$ 表示智能环境中所有智能节点的集合;

Rule = $\{R_1, R_2, \cdots, R_m\}$ 表示智能环境中推理所需的规则集;

Activity = $\{A_1, A_2, \cdots, A_l\}$ 表示该智能环境可感知识别的活动集.

**定义 1** **RSM** = $(a_{ij})_{m\times n}$表示规则和智能节点的关系,其中 $a_{ij} = R_i(S_j)$ 为布尔值,1 代表规则 $R_i$需获取智能节点 $S_j$的数据,0 代表规则 $R_i$与智能节点 $S_j$无关.

**定义 2**　**ARM** = $(b_{ij})_{l\times m}$表示活动类型和规则的关系,其中 $b_{ij}=A_i(R_j)$为布尔值,1 代表规则 $R_j$属于活动 $A_i$,0 代表活动 $A_i$不包含规则 $R_j$.

由于规则中的参数从智能节点中获取数据,同时又隶属于不同的活动,因此在后述算法中可建立智能节点与活动的关系.

### 3.2　智能节点网络与 Rete 网络

#### 3.2.1　智能节点网络模型

环境中的智能节点之间互相传输数据,从而构成智能节点网络.

**定义 3**　$SN=(\mathrm{SNode},E_S)$表示整个智能节点网络,其中 $E_S=\{(S_i,S_j)\mid S_i,S_j\in \mathrm{SNode}\}$是智能节点传输边的集合,$(S_i,S_j)$为 $S_i$与 $S_j$之间的直接传输边.

**定义 4**　$\boldsymbol{A}_{SN}=(a_{ij})_{m\times m}$表示 $SN$ 的邻接矩阵,其中

$$a_{ij}=\begin{cases}S_{ij},\text{if}\quad (S_i,S_j)\in Es;\\ 0,\text{if}\quad i=j;\\ \infty,\text{if}\quad (S_i,S_j)\notin Es\end{cases},$$

$S_{ij}$表示边$(S_i,S_j)$不经过其他智能节点的直接传输代价.

**定义 5**　$\mathrm{cost}(S_{ij})$表示智能节点 $S_i$与 $S_j$之间的最小传输代价,可能会经过其他智能节点,在后述算法用 $\boldsymbol{A}_{SN}$作为输入得到.

#### 3.2.2　Rete 网络模型

Rete 推理网络主要由以下推理节点构成:

alphaNode = $\{\alpha_1,\alpha_2,\cdots,\alpha_a\}$为单输入节点,如"$\alpha$: Temperature > presetTValue",其输入为智能节点获取并处理后的数据,过滤成功后流向下一个节点;

betaNode = $\{\beta_1,\beta_2,\cdots,\beta_b\}$为双输入节点,如"$\beta:\alpha_1$ AND $\alpha_2$",其输入为需要匹配的两个数据集合,匹配成功的数据可流向下一个节点;

TNode = $\{tr_1,tr_2,\cdots,tr_m\}$为终端节点,数据过滤至此将触发该终端节点对应的规则.

**定义 6**　RN = (inferenceNode, $E_{\mathrm{Rete}}$)表示整个推理网络,其中 inferenceNode 为上述推理节点,$E_{\mathrm{Rete}}$是推理网络中传输数据边的集合. 利用规则集完成推理网络的建立.

### 3.3　分配评估参数

为评估分配机制的资源平衡与实时性的优化程度,分别设计以下两个评估参数.

#### 3.3.1　资源平衡评估参数

推理节点到智能节点的分配应充分利用智能节点资源,因此将每个智能节点所分配推理节点个数建立一个数据集,用标准差来评判其离散程度从而判断其资源分配的平衡程度.

**定义 7**　$\mathrm{Stdsen}=\sqrt{\dfrac{1}{n-1}\sum_{i=1}^{n}(N(S_i)-\overline{N(S)})^2}$,其中 $N(S_i)$表示智能节点 $S_i$上推理节点个数. 分配完成后,Stdsen 值越小,分配离散程度就越低,资源利用就越高.

图 3 为同一推理网络的不同分配方式,图 3($a$)中 $S_4$资源空闲,$S_2$却被分配了三个推理节点,而图 3($b$)中的每个智能节点都得到充分利用. 由定义 7 可得图 3($a$)的 Stdsen 为 1.09,图 3($b$)的 Stdsen 为 0.45,图 3($b$)的资源利用率更优. 该推理网络只由一条简单规则构建,当规则数目增多时,资源分配也就变得愈加重要.

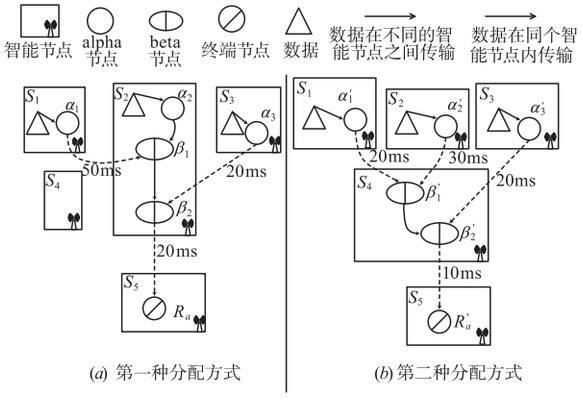



图3　分配评估参数举例

#### 3.3.2　实时性评估参数

实时性是指从数据进入推理网络,到系统推理出规则是否被触发的响应时间. 由于采用分布式推理,传输耗时主要在智能节点之间的传输上,在此同一个智能节点内的传输耗时可忽略不计[8].

**定义 8**　$L(\text{node or rule})$表示数据传输至某个推理节点的耗时. $L(\alpha_i)$表示数据传输至 $\alpha_i$的耗时;$L(\beta_i)$表示数据传输至 $\beta_i$的耗时,需考虑层次延迟性来计算;$L(R_i)$表示数据到达规则 $R_i$对应终端节点的耗时.

**定义 9**　$L(RN)=\sum(L(R_i))$,单个推理网络的实时性可用其所有规则的 $L(R_i)$之和来评估.

仍以图 3 为例,由于 $\alpha$ 节点都在其数据来源的智能节点中,左右都有

$$L(\alpha_1)=L(\alpha_2)=L(\alpha_3)=0 \tag{1}$$

$$L(\alpha_1')=L(\alpha_2')=L(\alpha_3')=0 \tag{2}$$

由于$\beta$ 节点两个输入都到达后才开始匹配,需根据层次延迟性来计算 $L(\beta_i)$的值. $L(\beta_1)$的值与其两个输入推理节点 $\alpha_1$、$\alpha_2$对应的 $L(\alpha_1)$、$L(\alpha_2)$及相关智能节点间的传输耗时有关:

$$L(\beta_1)=\max(L(\alpha_1)+\mathrm{cost}(S_{12}),L(\alpha_2)+0)=50\mathrm{ms} \tag{3}$$

$$L(\beta_1')=\max(L(\alpha_1')+\mathrm{cost}(S_{14}),L(\alpha_2')+\mathrm{cost}(S_{24}))=30\mathrm{ms} \tag{4}$$

$L(\beta_2)$的值与其两个输入推理节点 $\alpha_3$、$\beta_1$对应的 $L(\alpha_3)$、

$L(\beta_1)$及相关智能节点间的传输耗时有关：

$$L(\beta_2)=\max(L(\beta_1)+0,L(\alpha_3)+\text{cost}(S_{32}))=50\text{ms} \tag{5}$$

$$L(\beta_2')=\max(L(\beta_1')+0,L(\alpha_3')+\text{cost}(S_{34}))=30\text{ms} \tag{6}$$

对于终端节点，$L(R_a)$与其输入推理节点$\beta_2$对应的$L(\beta_2)$的值及相关智能节点间的传输耗时有关：

$$L(R_a)=L(\beta_2)+\text{cost}(S_{25})=70\text{ms} \tag{7}$$

$$L(R_a')=L(\beta_2')+\text{cost}(S_{45})=40\text{ms} \tag{8}$$

从而可看出图3(*b*)的实时性更优，因此需利用层次延迟性来获得更优的分配.

此外，由于智能环境下的规则具有活动类型的标签，如图4(*a*)(*b*)是相同的推理网络，若一个智能节点最多能分配一个推理节点，图4(*a*)不考虑活动类型，$S_4$中已存在一个推理节点，$\beta_2$被分配至另一个智能节点$S_5$中，而图4(*b*)考虑活动类型，若$\beta_2$与$\beta_1$是完全不相关的两个活动下的规则构造的推理节点，它们一般不会同时发生匹配工作而抢占资源，因此可存在于同一个智能节点中.

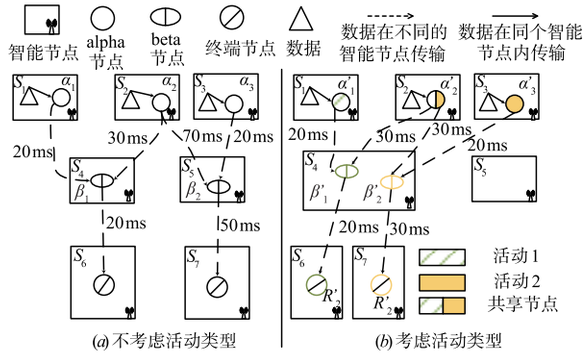



图4 活动类型对分配的影响

图4(*a*)(*b*)的$L(R_1)$值相同，主要是比较$L(R_2)$的值，按上述方法计算可得，

$$L(R_2)=70+50=120\text{ms} \tag{9}$$

$$L(R_2')=20+30=50\text{ms} \tag{10}$$

由式(9)与式(10)可得，图4(*b*)考虑活动类型的影响，其实时性更优.

因此考虑活动类型，允许一个智能节点存在不会同时发生匹配而抢占资源的推理节点，对实时性也有所提高，可对活动聚类后分别建立其推理网络，综上，可用以下参数来评估多个推理网络的实时性.

**定义10** $L(\text{total})=\sum(L(RN_i))$，对活动类型聚类后分别建立其推理网络，用所有推理网络的$L(RN_i)$之和来评估实时性的优化程度. 分配完成后$L(\text{total})$的值越小，实时性的优化程度越高.

## 4 优化分配算法

在智能环境下进行推理节点到智能节点的优化分配主要包括以下三个步骤：

①根据智能环境中智能节点网络的布置，计算出智能节点两两之间的最小传输代价并保存其路径信息；

②根据环境中活动类型、规则、智能节点的模型，推算出活动类型之间的距离，聚类后分别构造其Rete推理网络，具体实现见算法1、2；

③利用步骤①的结果，对步骤②中每个推理网络，利用算法3、4将其各类推理节点分配至智能节点.

### 4.1 智能节点最短路径计算

读取智能节点的配置文件，将节点之间的直接传输代价存储在矩阵$\boldsymbol{A}_{SN}$中，采用Floyd算法计算节点之间的最小传输代价，并保存最短路径相关信息，便于后述分配.

**定义11** $\text{next}(S_i,S_j)$表示从智能节点$S_i$到$S_j$的最短路径中$S_i$的下一个智能节点，将在后述算法中用到.

### 4.2 活动的聚类

若以下三点已知：智能环境下具体的规则集；识别感知每个活动需要哪些规则；规则中每个参数需要从哪个智能节点获取数据. 则可根据算法1完成定义1、2中相关矩阵的建立.

**算法1 CoA(1),ARM与RSM的建立**

```
Input: Rule set, the relationship between activities and rules, the relationship between parameters and smart nodes;
Output: ARM, RSM;
Initialization: ARM←0, RSM←0;
1: readActivityAndRule( ) //Read the activity label of each rule and store it in the corresponding properties of each rule.
2: for i←0 to sizeof(Rule) - 1 do
3:     j←Rule[i].ItsActivity
4:     ARM[i][j]←1
5: readRule( )
6: readSNodeAndPara( )
7: for i←0 to sizeof(Rule) - 1 do
8:     for j←0 to Rule[i].RelevantPara.size( ) - 1 do
9:         m←Rule[i].RelevantPara[j]
10:        n←para[m].ItsSNode
11:        RSM[i][n]←1
12: return ARM, RSM
```

算法1～4行根据规则的活动类型标签建立规则与活动的关系矩阵**ARM**.

算法第5行读取规则集，保存每条规则的前后件参数等信息，第6行读取每个智能节点可获取的参数，从而用于算法7～11行以参数为桥梁构建规则与智能节

点的关系矩阵 **RSM**.

以上述 **ARM** 和 **RSM** 为输入,完成算法 2.

**算法 2　CoA(2),ASM 的建立;聚类**

```
Input:ARM,RSM,Rule set;
Output:ASM,established inference network;
Initialization:ASM←0,JaccardM←0;
1:for i←0 to sizeof(Activity) - 1 do
2:      for j←0 to sizeof(SNode) - 1 do
3:          sum←0
4:          for k←0 to sizeof(Rule) - 1 do
5:              sum←sum + ARM[i][k] RSM[k][j]
6:          if sum! =0 then
7:              ASM[i][j]←1
8:for i←0 to sizeof(Activity) - 1 do
9:      for j←i to sizeof(Activity) - 1 do
10:         JaccardM[i][j]←Jaccard(ASM[i],ASM[j])
11:group←cluster(JaccardM)
12:for i←0 to sizeof(group) - 1 do
13:     Rete(g[i])// respectively build the Rete inference network
14:return established inference network
```

**定义 12**　**ASM** = $(c_{ij})_{l\times n}$表示活动和智能节点的关系,其中 $c_{ij}=A_i(S_j)$为布尔值,0 表示活动 $A_i$与智能节点 $S_j$无关;1 表示活动 $A_i$需要从智能节点 $S_j$中获取数据. $A_i(S_j)$的计算如下:

$$A_i(S_j)=\begin{cases}0, & \text{if } \sum_{k=0}^{m}A_i(R_k)R_k(S_j)=0\\ 1, & \text{else}\end{cases}$$

算法 1 ~ 7 行计算矩阵 **ASM**,8 ~ 10 行用下列公式计算两个活动类型之间的 Jaccard 距离: $J(A_a,A_b)=1-\frac{|A_a\cap A_b|}{|A_a|+|A_b|-|A_a\cap A_b|}$,其中 $A_a$对应矩阵 **ASM** 中就是第 $a$ 行数据. 由两两活动之间的距离便可得到整个活动的 Jaccard 距离矩阵,从而完成算法第 11 行的聚类.

**定义 13**　Group = $\{g_1,g_2,\cdots,g_x\}$表示聚类之后的集合.

算法 12 ~ 13 行在完成聚类后,对每一类活动的规则分别采用 Rete 算法建立其推理网络.

### 4.3　推理节点的分配

$\alpha$ 节点与终端节点的分配算法见算法 3.

**算法 3　AAoRN(1),$\alpha$ 节点与终端节点的分配**

```
Input:established inference network;
Output:allocated alpha inference nodes,allocated terminal nodes;
1:for i←0 to sizeof(alphaNode) - 1 do
2:      alphaNode[i].Lat←0
3:      alphaNode[i].ItsSNode←alphaNode[i].getPara().ItsSNode
4:for i←0 to sizeof(Rule) - 1 do
5:      terminalNode[i].ItsSNode←rule[i].outputSNode
6:return allocated alpha inference nodes,terminal nodes
```

对于 $\alpha$ 节点,为提前过滤数据以减少数据传输量,只需将其分配至提供其数据来源的智能节点,而终端节点输出的结论会传输至相关控制设备,将其分配至对应输出智能节点即可.

$\beta$ 节点的分配算法见算法 4.

**算法 4　AAoRN(2),$\beta$ 节点的分配**

```
Input:established inference network,Path;
Output:allocated beta inference nodes;
Initialization:topRN←1. 1 * sizeof(inferenceNode)/sizeof(SNode);
1:for i←0 to sizeof(betaNode) - 1 do
2:      leftrn←betaNode[i].antNode[0]
3:      rightrn←betaNode[i].antNode[1]
4:      trn←betaNode[i].getTerminalNode()
5:      StrNode←trn.ItsSNode
6:      SleftNode←leftrn.ItsSNode
7:      SrightNode←rightrn.ItsSNode
8:      SNodeArray←getSNode ()
9:      costArray←getCost(SNodeArray)
10:     indexMinheap.insert(costArray)
11:     tmpSNIndex←indexMinheap.getMinIndex();
12:     fullcount←0
13:     for j←1 to 6 do
14:         minSNIndex←indexMinheap.extractMinIndex()
15:         if SNode[minSNIndex].ItsRNCount <topRN then break
16:         else fullcount + +
17:     if fullcount = =6 then
18:         minSNIndex ← tmpIndex
19:     betaNode[i].ItsSNode← minSNIndex
20:     SNode[minSNIndex].ItsRNCount + +
21:return beta inference nodes
```

从顶至下对每个 $\beta$ 节点,算法第 2 ~ 4 行获取其两个输入推理节点以及终端推理节点,算法 5 ~ 7 行保存这三个推理节点所在智能节点,分别称其为 $S_{\text{left}}$、$S_{\text{right}}$和 $S_{\text{tr}}$,在算法 3 中这三个节点已确定. 由于智能节点超过两跳的传输代价很高,每个 $\beta$ 节点有以下六个适合分配的智能节点位置.

算法第 8 行按照表 1 获取当前 $\beta$ 推理节点的后四个智能节点位置,第 9 行根据层次延迟性,分别保存从 $S_{\text{left}}$和 $S_{\text{right}}$到上述六个智能节点的较大值,即该 $\beta$ 推理节点被分配到这六个智能节点的 $L(\beta_i)$,第 10 行将这六个值插入最小索引堆中.

**定义 14**　topRN = $a\times\frac{\text{推理节点个数}}{\text{智能节点个数}}$,根据推理节点数和智能节点数对 topRN 取值,可调整 $a$ 的值以获得

更合理的分配.

算法第 11 行保存拥有最小 $L(\beta_i)$ 的智能节点,12 行中的 fullcount 表示被分配推理节点数目超过 topRN 的智能节点个数,清空 fullcount. 13 ~ 16 行取出最小索引堆顶对应的智能节点,如果该智能节点所分配的推理节点数目已超过 topRN,则重复上述步骤,17 ~ 18 行判断上述六个智能节点被分配推理节点的数目是否都超过 topRN,如果是则该 $\beta$ 节点仍被分配至算法 11 行所保存的智能节点,目的是为了在无法保证资源平衡的情况下就优先保证其更优的实时性. 19 ~ 20 行更新被分配推理节点的相关参数,按上述方法继续逐层分配推理网络剩余的推理节点.

**表 1 $\beta$ 节点可分配的六个智能节点**

| 智能节点 | 位置 |
|---|---|
| $S_{\text{left}}$ | 其左输入推理节点所在智能节点 |
| $S_{\text{right}}$ | 其右输入推理节点所在智能节点 |
| $S_{\text{leftnext}}$ | $S_{\text{leftnext}} = \text{next}(S_{\text{left}}, S_{\text{tr}})$ |
| $S_{\text{left2next}}$ | $S_{\text{left2next}} = \text{next}(S_{\text{leftnext}}, S_{\text{tr}})$ |
| $S_{\text{rightnext}}$ | $S_{\text{rightnext}} = \text{next}(S_{\text{right}}, S_{\text{tr}})$ |
| $S_{\text{right2next}}$ | $S_{\text{right2next}} = \text{next}(S_{\text{rightnext}}, S_{\text{tr}})$ |

将上述过程举例阐述如下.

仍用规则“IF Temperature > presetTValue AND Humidity < presetHValue AND airconditioneAuto = TRUE THEN Dehumidification = TRUE”举例,其规则前件可替换为“$\alpha_1$ AND $\alpha_2$ AND $\alpha_3$”,上述步骤用图 5 作简单演示.

如图 5($a$),参数 temperature、humidity 和 airconditioneAuto 分别从智能节点 $S_1$、$S_2$ 和 $S_3$ 中获取数据,根据算法 3 将 $\alpha_1$、$\alpha_2$ 和 $\alpha_3$ 分别分配至 $S_1$、$S_2$ 和 $S_3$,而参数 Dehumidification 需要被传输至嵌入有湿度控制设备的智能节点 $S_8$,将终端节点分配至 $S_8$.

图 5($b$)中 $\beta_{(1,2)}$ 的输入推理节点为 $\alpha_1$、$\alpha_2$,其 $S_{\text{left}}$、$S_{\text{right}}$ 和 $S_{\text{tr}}$ 分别为 $S_1$、$S_2$ 和 $S_8$,根据步骤①得到对应最短路径,代入算法 4 若得出 $S_5$ 为满足最小 $L(\beta_{(1,2)})$ 的智能节点且其被分配推理节点未超过 topRN,则将 $\beta_{(1,2)}$ 分配至 $S_5$ 中.

图 5($c$)中 $\beta_{(1,2,3)}$ 的输入推理节点为 $\beta_{(1,2)}$ 和 $\alpha_3$,其 $S_{\text{left}}$、$S_{\text{right}}$ 和 $S_{\text{tr}}$ 分别为 $S_5$、$S_3$ 和 $S_8$,根据步骤①得到对应最短路径,代入算法 4 若得出 $S_6$ 为满足最小 $L(\beta_{(1,2,3)})$ 的智能节点且其被分配推理节点未超过 topRN,则将 $\beta_{(1,2,3)}$ 分配至 $S_6$,完成该规则的分配并更新相关参数. 继续按该方法分配剩余规则.

## 5 实验验证

本文采用 Smart Environment Simulator Tool[11],在仿真环境内布置智能节点网络,从而模拟分配过程:首先采用 AAoRN 分配算法,观察 3.3 节所述评估参数是否得到改善;再在分配前加入 CoA 算法,观察评估实时性的参数是否得到进一步优化.

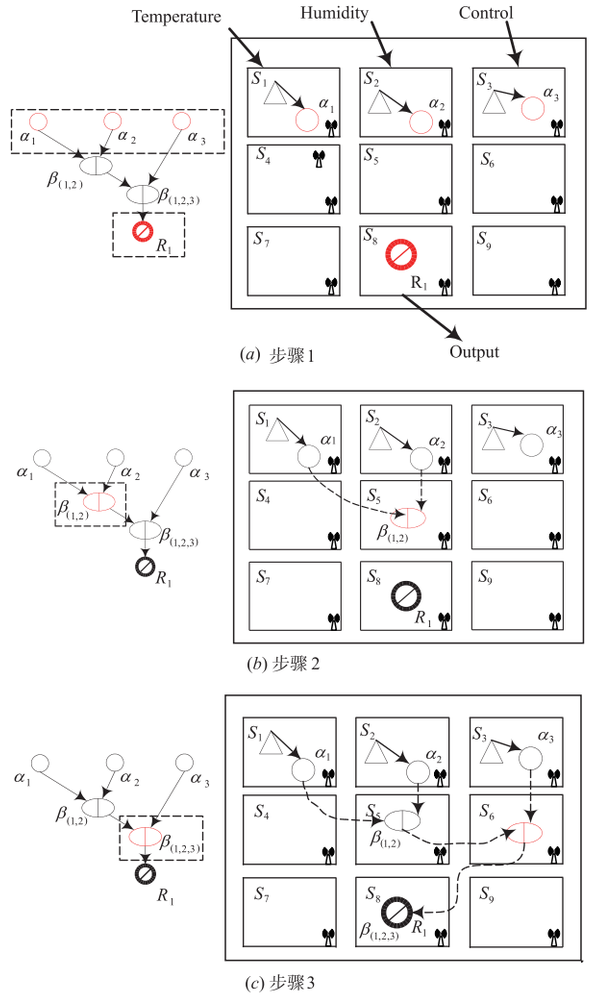



图5 单条规则构建的推理网络分配过程

### 5.1 AAoRN 分配算法的优化程度分析

不同的智能环境布置着不同的智能节点网络,规则集也不同,为验证 AAoRN 分配算法的有效性,将本文分配机制与集中式分布、普通分布式机制三者进行对比,设计以下两组数据.

(1)在智能节点网络相同的仿真环境下,设置规则数目逐渐增长的六组实验,实验编号及对应规则数如表 2 所示. 上述三种分配机制在六组实验中得到的 $L(\text{total})$ 如图 6 所示,对于集中式分布不用对其进行资源平衡的评估,AAoRN 与普通分布式机制在六组实验中得到的 Stdsen 如表 3 所示.

由图 6 可得,随着规则的增长推理网络变得更加复杂,集中式分布的延迟急剧增长,实时性很差;普通分布

式的实时性得到明显改善;采用 AAoRN 算法在普通分布式的基础上进一步改善了实时性.

**表 2　不同规则数目下的实验参数**

| 实验编号 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 规则数目 | 50 | 100 | 200 | 300 | 400 | 500 |

**表 3　不同规则数目下的 Stdsen**

| 实验编号 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| AAoRN 分布式 | 3. 61 | 4. 04 | 8. 75 | 12. 94 | 16. 93 | 20. 04 |
| 普通分布式 | 7. 48 | 11. 59 | 25. 70 | 41. 75 | 43. 48 | 72. 87 |

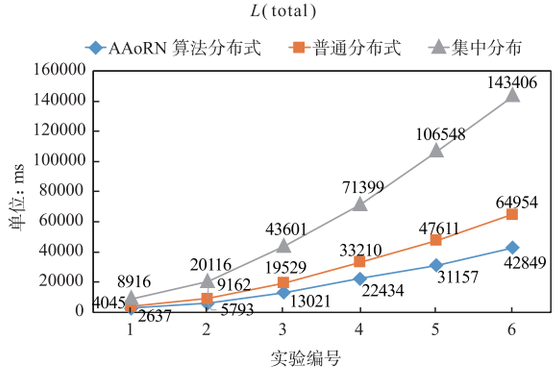



图6　不同规则数目下的$L$(total)

由表 3 可得采用 AAoRN 算法的资源利用明显比普通分布式更优,并且规则量大时优化程度更明显.

(2)同样的 300 条规则,即构造的推理网络相同,设置六组不同的仿真环境,其智能节点数目与传输边数目各不相同,实验编号及其对应参数如表 4 所示. 上述三种分配机制在六组实验中得到的 $L$(total)如图 7 所示, AAoRN 与普通分布式机制在实验中得到的 Stdsen 如表 5 所示.

**表 4　不同仿真环境下的实验参数**

| 实验编号 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 智能节点数目 | 16 | 16 | 36 | 36 | 50 | 50 |
| 直接传输边数目 | 20 | 36 | 36 | 52 | 52 | 70 |

**表 5　不同仿真环境下的 Stdsen**

| 实验编号 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| AAoRN 分布式 | 24. 65 | 18. 96 | 11. 77 | 12. 10 | 8. 14 | 7. 39 |
| 普通分布式 | 53. 36 | 42. 23 | 45. 44 | 29. 86 | 22. 62 | 21. 51 |

由图 7 及表 5 可看出,在规则相同,即推理网络相同的情况下,AAoRN 算法的实时性和资源平衡都得到了明显的提升. 但可观察到智能节点较多时,该分配机制的资源利用率优化不明显,这是由于同样规则集下推理节点个数相同,当智能节点的数目过多时,由于推理工作量并不大,为保证其实时性可牺牲资源平衡. 这也说明在今后研究中,应根据规则集的规模部署相应的智能节点网络.

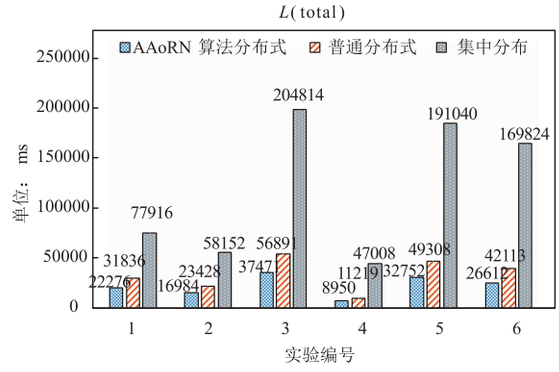



图7　不同仿真环境下的$L$(total)

## 5.2　加入 CoA 算法的优化程度分析

为观察加入 CoA 算法后的实时性是否进一步提升,采用表 6 的实验指标,在仿真环境中布置相应智能节点网络.

**表 6　验证 CoA 算法的实验参数**

| 参数名称 | 智能节点数 | 传输边 | 规则数 | 活动数 |
|---|---|---|---|---|
| 规则数目 | 50 | 100 | 200 | 300 |

对于 15 个活动下的 100 条规则,根据 CoA 算法建立 **ARM**、**RSM** 和 **ASM**,从而得到 Jaccard 距离矩阵后对活动进行聚类,图 8 为活动聚类树.

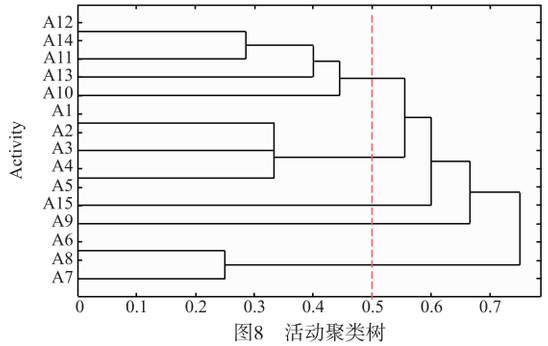



图8　活动聚类树

图 8 中的红色虚线将活动分为五类,这五类的推理节点一般不会同时发生匹配而抢占智能节点资源,每组的活动见表 7.

**表 7　活动聚类结果**

| *Group* | 活动 |
|---|---|
| 1 | $\boldsymbol{A}_1$:Enter the door;$\boldsymbol{A}_2$:Go out;$\boldsymbol{A}_3$:Have a meal;<br>$\boldsymbol{A}_4$:Cook;$\boldsymbol{A}_5$:Use the Fridge |
| 2 | $\boldsymbol{A}_6$:Wash hands;$\boldsymbol{A}_7$:Go to toilet;$\boldsymbol{A}_8$:Take a shower |
| 3 | $\boldsymbol{A}_9$:Play the computer |
| 4 | $\boldsymbol{A}_{10}$:Read Books;$\boldsymbol{A}_{11}$:Watch TV;$\boldsymbol{A}_{12}$:Use air conditioning;<br>$\boldsymbol{A}_{13}$:Draw the curtain;$\boldsymbol{A}_{14}$:Have a rest |
| 5 | $\boldsymbol{A}_{15}$:Go to sleep |

聚类完成后,分别构造其推理网络并按照 AAoRN 算法分配其推理节点,如图 9,其中不同颜色的推理节点是由不同类活动的规则构造的(省略了推理节点之间的传输边).

观察图 10 的 $L(\text{total})$ 可得,在分配前加上 CoA 算法,实时性进一步得到了提升.


图9 推理节点的分配

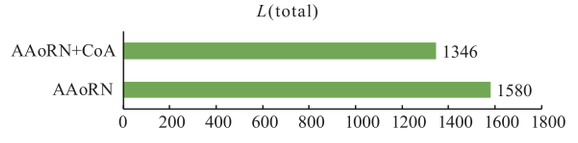



图10 加入CoA算法后的$L(\text{total})$

综上,由于 AAoRN 算法考虑了层次延迟性使每个推理节点都寻找最优的智能节点进行分配,时间延迟比普通分布式减少了大致 40%,资源平衡也提升了约 2 到 3 倍;而利用活动对规则推理的影响,在分配之前加入 CoA 算法进一步提升了实时性,减少了大致 15% 的延迟.

## 6 结束语

智能环境下的分布式规则推理主要存在实时性与资源平衡两大问题,本文利用智能环境下推理的层次延迟性以及规则触发受活动影响的特点,分别提出对应算法. 仿真实验表明,本文分配机制的实时性和资源平衡都明显更优.

由于本文构造推理网络所用的 Rete 算法更新换代迅速,如果能以更优的 Rete 算法得到推理网络,实时性将进一步得到提升,尤其在规则量较大时,Rete 算法就显得尤其重要. 此外,在大规模环境下,智能节点网络合理的布局也很大程度地影响推理的实时性,这都是今后研究需要关注的问题.

## 参考文献

[1] 汪成亮,王小均. 基于三轴传感器的老年人日常活动识别[J]. 电子学报,2017,45(3):570 – 576.
WANG Cheng-liang, WANG Xiao-jun. Daily activity recognition based on triaxial accelerometer of elderly people [J]. Acta Electronica Sinica, 2017, 45(3): 570 – 576. (in Chinese)

[2] Non-Alisavath K, Kanthavong S, Luangxaysana K, et al. Context-awareness application to control multiple sensors for monitoring smart environment[A]. The 14th International Conference on Electrical Engineering / Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)[C]. Phuket: IEEE, 2017. 920 – 923.

[3] 马森,赵文,袁崇义,等. 基于规则推理的语义检索若干关键技术研究[J]. 电子学报,2013,41(5):977 – 981.
MA Sen, ZHAO Wen, YUAN Chong-yi, et al. Research on critical technologies of semantic retrieval based on rule reasoning[J]. Acta Electronica Sinica, 2013, 41(5): 977 – 981. (in Chinese)

[4] Li J, Sato A, Huang R, et al. A rule-based knowledge discovery engine embedded semantic graph knowledge repository for retail business[A]. International Conference on Advanced Cloud and Big Data (CBD)[C]. Chengdu: IEEE, 2016. 81 – 86.

[5] Forgy C L. Rete: A fast algorithm for the many pattern/many object pattern match problem[J]. Artificial Intelligence, 1982, 19(1): 17 – 37.

[6] Medina Javier, Macarena Espinilla, Daniel Zafra, et al. Fuzzy fog computing: A linguistic approach for knowledge inference in wearable devices[A]. International Conference on Ubiquitous Computing and Ambient Intelligence [C]. Cham: Springer, 2017. 473 – 485.

[7] 汪成亮,温鑫. 智能环境下分布式 Rete 算法[J]. 计算机应用,2016,36(7):1893 – 1898.
WANG Cheng-liang, WEN Xin. Distributed Rete algorithm in smart environment[J]. Journal of Computer Applications, 2016, 36(7): 1893 – 1898. (in Chinese)

[8] 刘先省,申石磊,潘泉. 传感器管理及方法综述[J]. 电子学报,2002,30(3):394 – 398.
LIU Xian-sheng, SHEN Shi-lei, PAN Quan. A survey of sensor management and methods[J]. Acta Electronica Sinica, 2002, 30(3): 394 – 398. (in Chinese)

[9] Rashidi P, Cook D J, Holder L B, et al. Discovering activities to recognize and track in a smart environment[J]. IEEE Transactions on Knowledge and Data Engineering, 2011, 23(4): 527 – 539.

[10] Tomoya K, Naotaka F, Tomoki Y, et al. An evaluation and implementation of rule-based home energy management

system using the Rete algorithm[J]. The Scientific World Journal,2014,2014:1 - 8.

[11] Wang C,De D,Song W Z. Trajectory mining from anonymous binary motion sensors in smart environment[J]. Knowledge-Based Systems,2013,37:34.

## 作者简介

**汪成亮(通信作者)**　男,1975 年 5 月出生,四川资阳人. 博士,现为重庆大学计算机学院教授,博士生导师. 主要研究领域为复杂系统智能控制,无线网络及 RFID 研究与应用等.
E-mail:wangcl@ cqu. edu. cn

**黄心田**　女,1994 年 8 月出生,重庆人. 硕士,主要研究方向为物联网、规则推理计算.
E-mail:lpxhxt@ 163. com