分布式仿真系统实体节点分配问题实时求解算法

王虹森，罗汝斌，刘朝阳

（北京宇航系统工程研究所，北京 100076）

摘　要：分布式仿真系统中，如何使计算节点彼此间通信量尽可能小，是实体节点分配问题研究的内容。针对该问题提出一种基于规则的启发式两阶段实时求解算法。第一阶段构建最小期望事件数量的目标分配模型并进行求解，即根据连通图理论将原问题分解为多个子问题，结合缓存、分治和过滤优选策略，设计递归算法求解子问题，最后得到覆盖所有实体的事件最小集。第二阶段实现分箱算法，将最小集中单个事件关联的实体尽量分配至相同计算节点，最终得到实体节点分配关系。实际应用表明，相比常见的顺序分配策略，该算法能显著减小分布式仿真系统的跨节点网络通信，从而提升仿真效率。该算法还能在秒级耗时生成分配方案，特别适用于包含大量实体的复杂场景分布式仿真。

关键词：分布式仿真；实体节点分配；实时求解

中图分类号：V557　　**文献标志码：**A　　**文章编号：**2096-4080（2024）02-0024-08

Real-Time Solving Algorithm for Entity Node Assignment Problem in Distributed Simulation System

WANG Hongsen, LUO Rubin, LIU Zhaoyang

(Beijing Institute of Astronautical Systems Engineering, Beijing 100076, China)

Abstract: In distributed simulation systems, how to minimize communications between computing nodes is the research topic of entity node assignment. A rule-based heuristic two-stage real-time algorithm for this problem is proposed. In the first stage, an optimization model with minimum expected event number is constructed and solved. Firstly, the original problem is decomposed into multiple subproblems according to the connected graph theory. Secondly, combined with caching, divided and conquer, filter and selection strategies, a recursive algorithm is designed to solve the subproblems. Then, a minimum event set covering all entities is obtained. In the second stage, a binning algorithm is proposed with the goal of trying to allocate entities connected with the same event to the same computing node, ultimately obtaining the entity node assignment solution. Practical applications show that compared to common sequential assignment policy, this algorithm can significantly reduce network communications between computing nodes in distributed simulation systems, thereby improving simulation efficiency. This algorithm can also generate an assignment solution in seconds, making it particularly suitable for distributed simulations of complex scenes containing a large number of entities.

Key words: Distributed simulation; Entity node assignment; Real-time solving

收稿日期：2023-11-01；**修订日期：**2024-02-29

基金项目：中国航天科技集团自主研发项目

作者简介：王虹森（1989—），男，博士，工程师，主要研究方向离散事件仿真、体系仿真。

通信作者简介：罗汝斌（1981—），男，硕士，研究员，主要研究方向为体系仿真。

刘朝阳（1991—），男，硕士，高级工程师，主要研究方向为分布式仿真、软件工程化。

0　引言

基于离散事件的多主体仿真系统广泛应用于灾害救援、物资配送、安防反恐和在轨服务[1]等社会经济生活的多个领域。该类仿真系统仿真引擎的核心是调度离散事件，离散事件不仅驱动系统运行，也完成实体间的交互[2]。随着研究问题规模变大和实体执行逻辑愈发复杂，这类问题对计算性能的要求越来越高。使用单台计算机仿真复杂场景通常需要花费大量时间，而采用分布式仿真技术在多个计算节点开展并行仿真可以显著提升仿真效率。

在分布式仿真系统中，同一计算节点内的事件交互通常采用方法调用，跨计算节点的事件交互则需通过网络通信。网络通信涉及离散事件的封装、发送、传输、接收和解析，加上网络路由不确定性时延和操作系统进程调度等不确定性因素，对分布式仿真效率的影响巨大[3]。一个复杂仿真场景包含成百上千的实体，如何制定实体和计算节点的分配方案，以减少跨节点网络通信是值得研究的问题。特别是在想定方案分析、大样本仿真等业务中，还要求实时给出分配方案。

该问题属于一种扩展的二次多背包问题（Quadratic Multiple Knapsack Problem，QMKP）。QMKP 研究[4]给定多个物品，每个物品 i 的利润为 p_i，权重为 w_i，并给定多个背包，每个背包 K 的容量为 W_K，目标为将物品放入每个背包且放入背包物品的利润值总和最大，该利润除了每个物品的单独利润 p_i，还包括任两个物品间的二次利润（或联合利润）p_{ij}。对应实体节点分配问题，计算节点即是背包，实体即是物品，二次利润为实体间离散事件期望交互次数。该问题属于 NP 完全组合优化决策问题，当实体和节点的数量增多时，计算复杂度呈指数规模增长。

文献［4］最早在背包问题的基础上开展了 QMKP 研究，并提出 3 种元启发式求解算法。此后，学者们往往也采用元启发式算法在合理的时间内为 QMKP 求得近似解。代表性算法包括改进的遗传算法[5-6]、蜂群算法[7]、贪婪算法[8]和迭代响应搜索[9]等。目前，QMKP 相关研究不多，尚未有文献提出某种精确型算法。当决策变量规模变大时，元启发式算法可能需要花费很长时间。在实际工程应用中，为满足实时性要求，分布式仿真系统实体节点分配常常采用随机分配或顺序分配策略[10]。本研究针分布式仿真，结合实体和节点分配约束特点，提出一种基于规则的启发式两阶段实时求解算法，显著提升了复杂场景的分布式仿真推演效率。

1　模型建立

结合大型分布式仿真系统实体节点分配应用背景，设定实体节点分配问题如下。

1）所给定实体为一个仿真想定所包含的多个实体。任两个实体间可能存在事件交互关系，该交互关系是仿真模型构建阶段确定的，是已知的。

2）所给定节点为多个分布式计算节点。为充分利用处理器性能，单个节点分配实体数不超过节点处理器线程数，即节点实体容量等于处理器线程数。此时，不同的分配方案不太影响节点上实体计算速度，主要影响网络通信量。

3）最终分配方案需将所有实体分配到节点，使得单个节点内实体事件交互次数尽可能多，跨节点的实体事件交互次数尽可能少。在启动仿真之前无法预知每类事件的实际交互次数，所以设每类事件的产生和响应概率相等，则分配目标等价于使得单个节点内实体尽量处理相同类别事件，跨节点实体尽量处理不同类别事件。

该问题相比经典的 QMKP 放松了部分约束条件，为达到实时性要求，设计基于规则的启发式两阶段求解算法。

第一阶段，将原问题提取并抽象为一类静态目标分配问题[11]，为事件分配实体，根据事件和实体的处理关联关系构建最小期望成本[12]最优化模型，求解覆盖所有实体的事件最小集。最优化模型为

$$
\begin{aligned}
&\min \sum_{i=1}^{m} f(x_{ij}) \\
&f(x_{ij})=\begin{cases}1, & \sum_{j=1}^{n} x_{ij} \geqslant 1 \\ 0, & \sum_{j=1}^{n} x_{ij}=0\end{cases} \\
&\text{st.}\quad \boldsymbol{A}_{\text{matrix}}=[a_{ij}]_{m\times n},\ a_{ij}=0,1 \\
&i=1,2,\cdots,m \quad j=1,2,\cdots,n \\
&\sum_{j=1}^{n} a_{ij}x_{ij} \leqslant b, \quad i=1,2,\cdots,m \\
&\sum_{i=1}^{m} a_{ij}x_{ij} \geqslant 1, \quad j=1,2,\cdots,n \\
&x_{ij}=0,1
\end{aligned}
\tag{1}
$$

其中，事件总数为 m，实体总数为 n，节点实体容量为 b，决策变量为 x_{ij}。$x_{ij}=0$ 表示事件 i 处理关系不覆盖实体 j，$x_{ij}=1$ 表示事件 i 处理关系覆盖实体 j。$\boldsymbol{A}_{\text{matrix}}$ 为已知的事件和实体处理关联矩

阵，其中元素 $a_{i,j}=0$ 表示实体 j 不会处理事件 i，$a_{i,j}=1$ 表示实体 j 会处理事件 i。目标函数实际是最小化事件数量，最优解中单个事件所关联实体是尽可能多的。一般而言，该问题最优解有多个。

第二阶段是实现分箱算法，将最小集中单个事件关联的实体尽量分配至同一节点，最终得到实体节点分配关系。

2 模型求解

2.1 总体计算流程

第一阶段的目标分配问题也属于 NP 完全组合优化决策问题[13]，所以为同时满足大型应用规模和秒级计算速度的要求，提出一种基于规则的启发式算法。总体计算流程如图 1 所示，共涉及 5 个关键算法。在第一阶段，采用分治策略进行求解。首先根据连通图理论将原问题分解为多个子问题，然后对每个子问题单独求解后合并得到原问题的解。在求解子问题的过程中会构建和求解孙问题，再优选孙问题的解。在第二阶段，采用分箱算法求解实体节点分配方案。

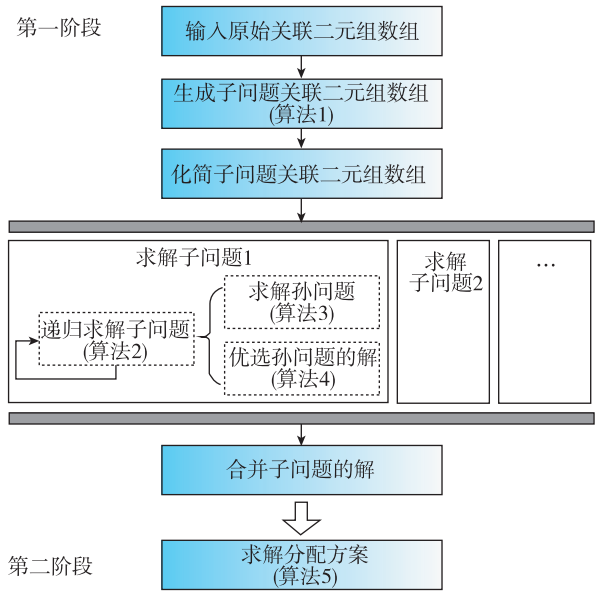

图 1 总体计算流程

Fig. 1 General calculation process

后续算法使用关联二元组数组表示关联矩阵。如果关联矩阵中 $a_{i,j}=1$，则二元组数组存在元素 (i, j)，两者一一对应并可相互转换。例如有关联矩阵 $[(1, 0), (0, 1)]$，则对应二元组数组为 $[(0, 0), (1, 1)]$。

2.2 生成子问题关联二元组数组

在第一阶段目标分配问题中，事件实体的关联关系实际上构成二部图（Bipartite Graph）。二部图中顶点集可分为两个不相交子集，仅不同子集中的顶点间存在边。考虑到大规模场景的事件实体关联矩阵一般是稀疏的，那么将原始二部图分解为多个连通子图，从而构建子问题。分解示意如图 2 所示，事件 1～3 和实体 1～2 构成连通子图，事件 4～5 和实体 3～4 构成连通子图，两个连通子图间无边连接。显然各连通子图的可行解无交叉，所以将连通子图对应的子问题最优解合并后，得到的解一定是原问题的最优解。

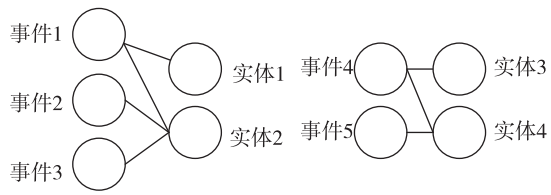

图 2 连通子图划分示意

Fig. 2 Connected subgraph division sample

后续算法关键符号及含义如表 1 所示。

表 1 关键符号及含义

Tab. 1 Key symbols and meanings

符号	含义
{}	set 集合
[]	vector 数组，通过索引号 i 取值为 vector [i]
<，>	map 键值对，键为 map. key，值为 map. value
(,)	tuple 二元组，前后值为 tuple. first，tuple. second
size（x）	求 x 的元素数量，x 为集合、数组或键值对
INT _ MAX	极大正整数

实体事件关联二元组数组分解算法如下所示。

算法 1：实体事件关联二元组数组分解算法

输入：事件实体关联二元组数组 $\boldsymbol{A}$，事件总数 m，实体总数 n
输出：多个事件实体子关联二元组数组 $\boldsymbol{S}_l$，$l=1, 2, \cdots$

```
初始化每个事件关联实体集合 T_i = {}，i = 0，1，…，m − 1
循环迭代 k = 0，1，…，size(A) − 1
    向 T_{A[k].first} 添加 T_{A[k].second}
初始化实体连通集合的数组 L = []，元素为{}
循环迭代 i = 0，1，…，m − 1
    初始化合并索引数组 M = []
    循环迭代 k = 0，1，…，size(L) − 1
        如果 T_i 与 L[k] 有交集
            向 M 添加 k
```

```
10    如果 M 为空
11      向 L 中添加 T_i
12    否则
13      向 L[M[0]] 中添加 T_i
14      循环迭代 k = size(M)－1，…，2，1
15        向 L[M[0]] 中添加 L[M[k]]，并从 L 删除 L[M[k]]
16  初始化二元组数组 S_l = []，元素为()，l = 0，1，…，size(L)－1
17  循环迭代 k = 0，1，…，size(A)－1
18    循环迭代 l = 0，1，…，size(L)－1
19      如果 L[l] 包含 A[k].second
20        向 S_l 中添加 A [k]
```

2.3　化简子问题关联二元组数组

为松弛单个节点的分配实体不能超过数目 b 的约束，对关联二元组数组进行转换：如果某个事件 i 的关联实体总数为 d_i，且 d_i 大于 b，则去除事件 i，新建 $C_{d_i}^{b}$ 个虚拟事件，每个虚拟事件的关联实体为从 d_i 个实体中挑选 b 个不同元素形成的组合。

然后，精简子问题关联二元组数组。如果一个事件 i 的关联实体是另一个事件 j 关联实体的子集或者相同，则仅保留事件 j，删除事件 i。显然，如果最优解包含事件 i，则将事件 i 替换为 j，目标函数值不变，仍然是最优解。

以上两步对关联二元组数组的处理算法比较简单，此处不再赘述。由此得到单个子问题的模型为

$$\begin{aligned}&\min\sum_{i=1}^{p} f(x_{ij})\\ &f(x_{ij})=\begin{cases}1, & \sum_{j=1}^{q} x_{ij}\geqslant 1\\ 0, & \sum_{j=1}^{q} x_{ij}=0\end{cases}\\ &\text{st.}\quad \boldsymbol{S}_{\text{matrix}}=[s_{ij}]_{p\times q},s_{ij}=0,1\\ &i=1,2,\cdots,p\quad j=1,2,\cdots,q\\ &\sum_{i=1}^{p} s_{ij}x_{ij}\geqslant 1,\quad j=1,2,\cdots,q\\ &x_{ij}=0,1\end{aligned}\tag{2}$$

其中，事件总数为 p，实体总数为 q。一般而言，$p\geqslant m$，$q\leqslant n$。

2.4　求解子问题

对于该子问题，目标函数为非线性函数，无法采用经典的匈牙利算法求解。结合缓存、分治和过滤优选策略，设计快速递归迭代算法，如下所示。

算法 2：递归求解算法

输入： 事件实体关联二元组数组 $\boldsymbol{S}$，已分配事件数组 $\boldsymbol{X}=[]$，事件总数 p，实体总数 q

输出： 已分配事件数组 $\boldsymbol{X}$

全局变量： 缓存键值对 $\boldsymbol{H}=<,>$，$\boldsymbol{H}$.key 为关联二元组数组，$\boldsymbol{H}$.value 为分配事件数组最优解

预置参数： θ 孙问题关联二元组数组规模阈值

```
1   如果 S 为空
2     返回
3   如果 H 的键包含当前输入的 S
4     X 赋值对应 H.value，并返回
5   初始化实体所能关联事件数组 W_j = []，j = 0，1，…，q－1
6   循环迭代 k = 0，1，…，size(S)－1
7     向 W_{S[k].second} 添加 S[k].first
8   初始化事件最少数量 w_min = INT_MAX
9   循环迭代 j = 0，1，…，q－1
10    如果 size(W_j) 小于 w_min
11      w_min = size(W_j)
12  初始化孙问题的关联二元组数组 G = []，元素为()
13  循环迭代 j = 0，1，…，q－1
14    如果 size(W_j) 等于 w_min
15      循环迭代 k = 0，1，…，w_min－1
16        向 G 中添加(W_j[k]，j)
17      如果 size(G) 大于 θ
18        跳出循环
19  求解 G 关联孙问题，得到其所有可行解 X_G(见算法 3)
20  对 X_G 进行过滤优选，形成较优解 X_P(见算法 4)
21  初始化事件最少数量 w_min = INT_MAX
22  循环迭代 i = 0，1，…，size(X_P)－1
23    复制 X_P[i] 为 X_I
24    如果 size(X_I) 大于等于 w_min
25      继续循环
26    初始化 X_I 能关联的实体集合 V = {}
27    循环迭代 k = 0，1，…，size(S)－1
28      如果 X_I 包含 S[k].first
29        向 V 中添加 S[k].second
30    初始化递归迭代的关联二元组数组 S_R
31    循环迭代 k = 0，1，…，size(S)－1
32      如果 X_I 不包含 S[k].first 并且 V 不包含 S[k].second
33        向 S_R 中添加 S[k]
34    将 S_R，X_R = [] 作为输入，递归调用算法 2，输出 X_R
35    向 X_I 中添加 X_R 所有元素
36    如果 size(X_I) 小于 w_min
37      w_min = size(X_I)，X = X_I
38  向 H 中添加键值对 <S，X>
```

算法 2 中，缓存策略体现在第 3、4 和 38 步。在组合遍历迭代时，某个关联二元组数组对应的最优解可能在前序迭代中已经算出，将其缓存为键值对，直接查询以加速计算。分治策略体现在第 12～19 步，构建和求解决策变量规模较小的孙问题。过滤优选策略体现在第 20 步，优选孙问题可行解。所构建孙问题与子问题模型(2) 相同，

但决策变量更少。孙问题的关联二元组数组 $\boldsymbol{G}$ 是子问题 $\boldsymbol{S}$ 的子集。$\boldsymbol{G}$ 的特征是其完全包含了 $\boldsymbol{G}$ 中实体相关的事件实体关联关系，即差集 $\boldsymbol{S}$-$\boldsymbol{G}$ 不会包含 $\boldsymbol{G}$ 中的实体。在迭代遍历孙问题所有可行解的基础上可以求得子问题的最优解基于以下结论。

$\boldsymbol{G}$ 关联孙问题的所有可行解中至少有一个是 $\boldsymbol{S}$ 关联子问题最优解的子集。若 $\boldsymbol{G}$ 关联孙问题的所有可行解都不是 $\boldsymbol{S}$ 关联子问题最优解的子集，那么 $\boldsymbol{S}$ 关联子问题最优解不是 $\boldsymbol{G}$ 关联孙问题的可行解，由于差集 $\boldsymbol{S}$-$\boldsymbol{G}$ 不会包含 $\boldsymbol{G}$ 中的实体，那么 $\boldsymbol{S}$ 关联子问题最优解不是 $\boldsymbol{S}$ 关联子问题的可行解，因此矛盾，结论可证。

在构建关联二元组数组 $\boldsymbol{G}$ 时，优先选择关联事件数目最小的那些关键实体，以减少外层迭代的循环次数，提升计算效率。例如，有 7 条与 4 个实体相关的关联关系：实体 1 与事件 1，2 关联，实体 2 与事件 2，3 关联，实体 3 与事件 1，3，4 关联。那么实体 1 和实体 2 关联的事件数是 2，实体 3 关联事件数是 3。根据规则，关联事件数最小为 2，则提取实体 1 和实体 2 相关的关联关系构成 $\boldsymbol{G}$。

孙问题的求解算法如下所示。该求解算法与算法 2 相似，主要区别是算法 2 的关键实体是多个，算法 3 的关键实体是 1 个。算法 3 从一个关键实体开始递归迭代，找到孙问题所有的可行解。

算法 3：算法 2 第 19 步关联孙问题求解算法

输入：事件实体关联二元组数组 $\boldsymbol{G}$，所有可行解数组 $\boldsymbol{X}_{\mathrm{G}}=[]$，元素为[]，事件总数 u，实体总数 v

输出：所有可行解数组 $\boldsymbol{X}_{\mathrm{G}}$

全局变量：最小事件数量 $w^{*}=\mathrm{INT\_MAX}$

```
如果 G 为空
  返回
初始化实体所能关联事件数组W_j=[]，j=0，1，…，v-1
循环迭代 k=0，1，…，size(G)-1
  向 W_{G[k].second} 添加 G[k].first
初始化事件最少数量 w_min=INT_MAX，事件集合Y=[]
循环迭代 j=0，1，…，v-1
  如果 size(W_j) 小于 w_min
    w_min=size(W_j)，Y=W_j
 循环迭代 i=0，1，…，size(Y)-1
   复制 X_G 为 X，并向 X 中添加 Y[i]
   初始化关联实体集合 V={}
   循环迭代 k=0，1，…，size(G)-1
     如果 X 包含 G[k].first
       向 V 中添加 G[k].second
   如果 size(V) 等于实体总数 v
     如果 size(X) 小于 w*
       w*=size(X)
     向 X_G 添加 X
     继续循环
初始化递归迭代的关联二元组数组 G_R=[]
循环迭代 k=0，1，…，size(G)-1
   如果 X 不包含 G[k].first，并且V不包含 G[k].second
     向 G_R 中添加 G[k]
将 G_R 和 X 作为输入参数，递归调用算法 3，输出 X
```

孙问题的可行解非常多，全部遍历非常耗时，采用过滤优选是保证实时返回局部最优解的关键。孙问题可行解过滤优选算法如下所示。

算法 4：算法 2 第 20 步孙问题可行解过滤优选算法

输入：事件实体关联二元组数组 $\boldsymbol{G}$，所有可行解数组 $\boldsymbol{X}_{\mathrm{G}}$，事件总数 u，实体总数 v

输出：优选可行解数组 $\boldsymbol{X}_{\mathrm{P}}$

预置参数：μ_1，μ_2，μ_3，μ_4 分别对应不同长度解的挑选个数

```
如果 X_G 为空
  返回
如果 size(X_G) 等于 1
  X_P=X_G，返回
初始化实体所能关联事件数组 W_j=[]，j=0，1，…，v-1
循环迭代 k=0，1，…，size(G)-1
  向 W_{G[k].second} 添加 G[k].first
初始化每个事件关联实体集合 T_i={}，i=0，1，…，u-1
初始化每个实体能关联事件数 w_j=0，j=0，1，…，v-1
 循环迭代 k=0，1，…，size(G)-1
   向 T_{G[k].first} 添加 G[k].second，W_{G[k].second} 加 1
初始化可行解覆盖的实体集合数组 Z=[]，元素为{}
初始化事件最少数量 w_min=INT_MAX
循环迭代 i=0，1，…，size(X_G)-1
  向 Z 添加空集合{}
  循环迭代 k=0，1，…，size(X_G[i])-1
    向 Z 的最后一个集合元素中添加 T_{X_G[i][k]}
  如果 size(X_G[i]) 小于 w_min
    w_min=size(X_G[i])
初始化过滤控制器 D=<，>，键为事件数量，值为{{}}
初始化每个可行解的得分元组数组 C=[]，元素为()
循环迭代 k=0，1，…，size(X_G)-1
  如果 D.key 为 size(X_G[k]) 的集合中包含 Z[k]
    继续循环
  向 D.key 为 size(X_G[k]) 的集合中添加 Z[k]
  初始化得分 e 为 0.0
  循环遍历 l=0，1，…，size(Z[k])-1
    e 加上 1.0/w_{Z[k][l]}
  向 C 中添加元组(k，e)
对数组 C 照元组的后值进行降序排列，形成数组 F
循环迭代 k=0，1，…，size(F)-1
  如果 μ1 大于 0，并且 size(X_G[F[k].first]) 等于 w_min
    向 X_P 添加 X_G[F[k].first]，μ1 减 1
循环迭代 k=0，1，…，size(F)-1
```

```
35      如果 μ2 大于 0，并且 size(X_G[F[k].first]) 等于 w_min+1
36        向 X_P 添加 X_G[F[k].first]，μ2 减 1
37    循环迭代 k = 0，1，…，size(F)−1
38      如果 μ3 大于 0，并且 size(X_G[F[k].first]) 等于 w_min+2
39        向 X_P 添加 X_G[F[k].first]，μ3 减 1
40    循环迭代 k = 0，1，…，size(F)−1
41      如果 μ4 大于 0，并且 size(X_G[F[k].first]) 大于 w_min+2
42        向 X_P 添加 X_G[F[k].first]，μ4 减 1
```

过滤优选算法核心有两点：过滤和优选。

过滤是指对于拥有相同事件数量且相同实体覆盖范围的多个可行解，只保留一个。体现在算法中第 20～25 步，具体由 $\boldsymbol{D}$ 实现。例如，有 4 个解，事件分别是［［1，2］，［3，4］，［1，5，6］，［3，7，8］］，事件［1，2］覆盖实体［1，2，3，4］，事件［3，4］覆盖实体［1，2，3，4］，事件［1，5，6］覆盖实体［1，2，3，4，5］，事件［3，7，8］覆盖实体［1，2，3，5，6］。那么，先判断拥有 2 个事件的解，［1，2］和［3，4］覆盖实体相同，则保留事件［1，2］，删除事件［3，4］，显然，如果最优解包含［3，4］，则将［3，4］换为［1，2］一定也是最优解。再判断拥有 3 个事件的解，［1，5，6］和［3，7，8］覆盖实体不同，则两者都保留。

优选是指对过滤后的解进行统一打分排序，再挑选得分较高的解参与后续迭代。体现在算法中第 21 和第 26～42 步。打分规则是（第 27～28 步）：该解覆盖的每个实体的关联事件数量倒数之和。例如，有 2 个解，事件［1］和［3，4］，事件［1］覆盖实体［1，2］，事件［3，4］覆盖实体［1，3］，实体 1 在关联二元组数组中涉及 3 个事件，实体 2 涉及 4 个事件，实体 3 涉及 5 个事件。那么，解［1］得分是 1/3＋1/4＝7/12，解［3，4］得分是 1/3＋1/5＝8/15，由于 7/12＞8/15，所以解［1］排在解［3，4］前面。该打分策略采用加法是基于：若解覆盖实体越多，该解越有可能靠近最优解，那么这时加项越多，分值越大。采用倒数是基于：若解覆盖实体数目相同，这时加项数量相同，选择拥有较小关联事件数目的实体更可能靠近最优解，通过采用取倒数的方式，该解得分越高。

算法 4 中有 4 个预置参数 μ_1、μ_2、μ_3 和 μ_4，分别对应不同长度解的挑选个数，不同的预置参数设置影响计算耗时。

2.5　合并子问题的解

对每个子问题分别求解，合并子问题的解就得到原问题的最优解。设最优解事件数组为 $\boldsymbol{X}^*$，数组元素为事件编号，$\boldsymbol{X}^*$ 覆盖的实体集合的数组为 $\boldsymbol{T}^*$，数组元素为实体的集合，与 $\boldsymbol{X}^*$ 顺序对应。

2.6　求解分配方案

在事件最小集的基础上，执行分箱算法如下，将每个实体分配至特定的节点，得到最终分配方案。

算法 5：求解分配方案的分箱算法

输入：最优解事件数组 $\boldsymbol{X}^*$，$\boldsymbol{X}^*$ 覆盖的实体集合的数组 $\boldsymbol{T}^*$，节点总数 t，单个节点实体容量 b

输出：每个节点中实体编号集合的数组 $\boldsymbol{E}=[]$，元素为{}

```
1     为 E 中添加 t 个空集合{}
2     初始化剩余实体数组 R = []
3     循环迭代 size(T*) 大于 0
4       初始化布尔标记变量 f 为假
5       循环迭代 i = 0，1，…，t−1
6         如果 E[i] 为空，或 E[i] 与 T*[0] 并集元素总数为 b
7           将 T*[0] 所有元素添加至 E[i]
8           将 f 设置为真
9           跳出循环
10      如果 f 为假
11        循环迭代 j = 0，1，…，size(T*[0])−1
12          向 R 中添加 T*[0][j]
13      循环迭代 i = 1，2，…，size(T*)−1
14        循环迭代 j = 0，1，…，size(T*[0])−1
15          如果 T*[i] 包含 T*[0][j]
16            从 T*[i] 删除 T*[0][j]
17      从 T* 中移除 T*[0]
18      对 T* 按照元素长度降序排序
19    循环迭代 size(R) 大于 0
20      循环迭代 i = t−1，…，1，0
21        如果 size(E[i]) 小于 b
22          初始化个数变量 n 为 b-size(E[i])
23          如果 n 大于 size(R)
24            n = size(R)
25          循环迭代 j = 0，1，…，n−1
26            向 E[i] 中添加 R[j]
27          循环迭代 j = 0，1，…，n−1
28            在 R 中移除 R[0]
```

3　应用试验

结合实际应用需求在分布式仿真平台上编制仿真想定，共包含 130 个事件和 105 个实体。梳理仿真模型得到共 500 条事件实体处理关联关系。关联 1～15 个实体的事件数分别是［28，22，17，17，19，9，6，2，5，3，0，1，0，0，1］。关联 1～15 个事件的实体数分别是［0，7，26，29，13，11，8，1，4，1，1，1，1，0，1］。

设单个节点实体容量 $b=6$，松弛该约束后，形成了 42 802 条事件实体关联关系，再化简后形成 36 765 条关联关系。

采用 C++语言实现第一阶段算法，并设置 3 组预置参数，如表 2 所示。其中 FAST 组和 MEDI 组得到局部最优解，SLOW 组实际上得到全局最优解。

计算节点 CPU 使用 Intel 酷睿 i7-7700 K，实际运行对比结果如表 3 所示。3 个组的最优解和最优值都相同，说明都找到了全局最优解，而且 FAST 组仅耗时 0.42 s。对最优解的覆盖实体进行检验，最优解能完全覆盖 105 个实体。

表 2 第一阶段 3 个组预置参数设置

Tab. 2 Three preset parameter groups in the first stage

组名	预置参数设置
FAST	$\theta=200$，$\mu_1=1$，$\mu_2=1$，$\mu_3=0$，$\mu_4=0$
MEDI	$\theta=200$，$\mu_1=4$，$\mu_2=2$，$\mu_3=1$，$\mu_4=0$
SLOW	$\theta=200$，$\mu_1=\mu_2=\mu_3=\mu_4=100\ 000$

表 3 第一阶段 3 个组计算结果

Tab. 3 Results of three groups in the first stage

组名	最优解	最优值	耗时
FAST	10，14，18，27，32，42，46，50，54，57，59，63，64，69，75，80，87，92，102，115，121，130	22	0.42 s
MEDI	10，14，18，27，32，42，46，50，54，57，59，63，64，69，75，80，87，92，102，115，121，130	22	7.07 s
SLOW	10，14，18，27，32，42，46，50，54，57，59，63，64，69，75，80，87，92，102，115，121，130	22	1 373.68 s

采用 C++语言实现第二阶段算法，其中计算节点总数 $t=18$，单个节点实体容量 $b=6$。算法共耗时 0.08 s，最终求得每个计算节点分配实体编号为：[{37，71，20，23，24，94}，{3，104，10，44，62，31}，{11，45，55，59，60，93}，{5，38，8，25，26，92}，{33，68，6，105，49，50}，{32，12，13，56，57，58}，{14，15，16，46，47，48}，{64，2，73，43，18，82}，{65，102，42，76，85，86}，{1，101，77，53，87，88}，{99，100，41，89，90，29}，{34，4，69，81，19}，{96，97，40，78，28}，{35，67，7，80，51}，{66，39，79，27，95}，{70，9，75，83，84，61}，{98，103，74，21，91，30}，{36，72，52，54，22，63}]。

接下来对比本算法和顺序分配策略的仿真效率。在相同软硬件环境的分布式仿真平台分别运行所编制的仿真想定，计算节点间采用开源的基于数据分发服务[14]实现通信。仿真开始为想定时间 0 s，结束为想定时间 10 000 s。两种策略想定时间向前推进的真实世界实际耗时变化如图 3 所示。

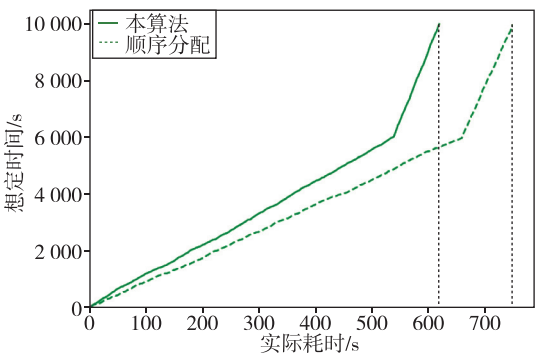

图 3 仿真推演效率对比

Fig. 3 Comparison of simulation deductions

图 3 中，曲线出现拐点是因为在该想定时间部分实体结束任务并退出仿真。拐点前的时间段，实体数目较多，本算法相比顺序分配的推演效率提升约 22%，拐点后的时间段，实体数目较少，本算法推演效率提升约 6%。实体间事件交互次数越多，本算法提升效率越明显。想定推进到 10 000 s 结束时，本算法实际耗时 617 s，顺序分配实际耗时 748 s，仿真推演效率综合提升约 17%。

4 结束语

本研究针对分布式仿真系统的实体节点分配问题，提出了一种基于规则的启发式两阶段实时求解算法。第一阶段将原问题提取并抽象为一类静态目标分配问题，构建了最小期望事件数的最优化模型，得到覆盖所有实体的事件最小集。第二阶段利用事件最小集进行分箱操作，得到最终分配方案。应用试验表明，该算法能显著减小分

布式仿真系统的网络通信，提升仿真推演效率，并能在秒级耗时内生成分配方案，特别适用于包含大量实体的复杂场景分布式仿真。

参考文献

[1] 杨自鹏，胡声超，周佑君，等. 多任务在轨服务模块化智能航天器技术研究[J]. 宇航总体技术，2019，3(4)：15-20.

[2] 郭勇陈，沈洋. 基于离散事件仿真原理的多主体仿真控制方法[J]. 计算机仿真，2015，32(6)：349-355.

[3] 王鹏，刘东玲，杨辉，等. 分布式仿真系统的网络通信问题研究[C]// 第三十三届中国仿真大会，北京，2021.

[4] Hiley A，Julstrom B A. The quadratic multiple knapsack problem and three heuristic approaches to it [C]//Proceedings of the 8th annual conference on Genetic and evolutionary computation. Seattle，Washington，USA. New York：ACM，2006：547-552.

[5] Singh A，Baghel A S. A new grouping genetic algorithm for the quadratic multiple knapsack problem[C]//Proceedings of the 7th European conference on Evolutionary computation in combinatorial optimization. Valencia，Spain. New York：ACM，2007：210-218.

[6] Saraç T，Sipahioglu A. A genetic algorithm for the quadratic multiple knapsack problem[C]//Proceedings of the 2nd international conference on Advances in brain，vision and artificial intelligence. Naples，Italy. New York：ACM，2007：490-498.

[7] Sundar S，Singh A. A swarm intelligence approach to the quadratic multiple knapsack problem[C]//Proceedings of the 17th international conference on Neural information processing：theory and algorithms-Volume Part I. Sydney，Australia. New York：ACM，2010：626-633.

[8] García-Martínez C，Rodriguez F J，Lozano M. Tabu-enhanced iterated greedy algorithm：a case study in the quadratic multiple knapsack problem [J]. European Journal of Operational Research，2014，232(3)：454-463.

[9] Chen Y N，Hao J K. Iterated responsive threshold search for the quadratic multiple knapsack problem [J]. Annals of Operations Research，2015，226(1)：101-131.

[10] 宋卫. 分布式系统中的通信机制及负载均衡[J]. 电子技术与软件工程，2019(14)：25-26.

[11] 杨进帅，李进，王毅. 武器-目标分配问题研究[J]. 火力与指挥控制，2019，44(5)：6-11.

[12] Kwon O，Lee K，Kang D H，et al. A branch-and-price algorithm for a targeting problem[J]. Naval Research Logistics (NRL)，2007，54(7)：732-741.

[13] Lloyd S P，Witsenhausen H S. Weapons allocation is NP-complete[C]// The 18th Summer Computer Simulation Conference，Reno，USA，1986.

[14] 雷媛元，焦璐，王锐，等.基于数据分发服务的通用仿真框架技术[J].计算机应用，2020，40(S1)：146-151.

引用格式：王虹森，罗汝斌，刘朝阳.分布式仿真系统实体节点分配问题实时求解算法[J].宇航总体技术，2024，8(2)：24-31.

Citation：Wang H S，Luo R B，Liu Z Y. Real-time solving algorithm for entity node assignment problem in distributed simulation system [J]. Astronautical Systems Engineering Technology，2024，8(2)：24-31.