分布式交互应用中服务器放置问题的启发式算法

郑晶晶 张 晶 武继刚
(天津工业大学计算机科学与软件学院 天津 300387)

摘 要 分布式交互应用是允许分散在不同地点的多个参与者能实时进行交互的网络系统,它的交互质量在很大程度上取决于网络延迟,而通过对服务器位置的合理布局可以降低网络延迟。因此,服务器放置是影响分布式交互应用的交互性能的关键因素。针对分布式交互应用中服务器放置问题,提出了模拟退火算法和禁忌搜索算法,并与已有的遗传算法进行了比较。通过实验可以看出,尽管在求得较好解的速度方面,遗传算法占据优势,但在求得解的质量方面,提出的模拟退火算法和禁忌搜索算法均优于遗传算法,在服务器数量相同的条件下,延迟平均降低了 15.5%和 15.2%,更加有效地提高了交互质量。

关键词 分布式交互应用,服务器放置,遗传算法,模拟退火算法,禁忌搜索算法
中图法分类号 TP391 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.7.020

Heuristic Algorithm for Server Placement in Distributed Interactive Applications

ZHENG Jing-jing ZHANG Jing WU Ji-gang
(School of Computer Science & Software Engineering, Tianjin Polytechnic University, Tianjin 300387, China)

Abstract Distributed interactive applications(DIAs) are networked systems that allow multiple participants at different locations to interact with each other in real time. The interaction quality heavily depends on network latencies which can be reduced by reasonable distribution of location where the servers of the DIAs are placed. Thus, the placement of servers is a key factor to the interactivity performance of DIAs. The simulated annealing algorithm and tabu search algorithm were used to solve the server placement problem in this paper. Then these two algorithms were compared with genetic algorithm. The experiment results show that the genetic algorithm is much faster in the speed of obtaining a better solution, but simulated annealing algorithm and tabu search algorithm outperform genetic algorithm in the quality of the solution and obtain average latency reduction of 15.5% and 15.2% respectively for the same number of server, which effectively improve the quality of the interaction.

Keywords Distributed interactive application, Server placement, Genetic algorithm, Simulated annealing algorithm, Tabu search algorithm

1 引言

近年来,分布式交互应用得到了广泛发展,它支持分散在各个不同地点的多个参与者实时进行交互来完成特定任务,例如多人网络在线游戏[1]、分布式交互仿真[2]和协作式数字化学习[3]等等。分布式交互应用具有延迟敏感[4],它的交互性能在很大程度上取决于网络延迟[5]。因此,降低网络延迟、保证交互质量成为了分布式交互应用中一大关键性挑战。

分布式交互应用通常采用用户-服务器框架模式,每个用户被分配给唯一的服务器并通过它与其他用户进行交互[6,7]。用户发布一个操作指令后将指令先传递给为其分配的服务器,再传递给其他用户。从一个用户发布操作指令到其他用户接收到指令的延迟时间被用来评价交互质量,这个延迟时间也称做交互时长[8]。服务器地理位置布局不仅会影响服务器与服务器间的延迟,还会影响到用户与服务器间的延迟,因此服务器放置位置是降低延迟时间、提高用户间交互质量的一个重要因素。

关于服务器放置问题,在不同的应用背景下有许多算法被提出。如在文献[9]中提出了大规模网络游戏中服务器位置选择的分布式算法 ZIZO,目标是在所有用户对都满足给定延迟需求的条件下,找出数量最小的服务器组来为用户提供服务,ZIZO 算法并未尝试降低交互时长而只是保证限定性的交互质量。文献[10]中提出了服务设施选址问题(k-median 问题)的分布式算法,旨在大规模网络中最小化用户与其最近的服务设施之间的距离。当网络中服务器位置已被固定,接下来的问题是如何为用户分配服务器,文献[11,12]证明了此组合优化问题为 NP 问题,并提出了集中式和分布式的贪心算法。文献[13]提出了一种改进的遗传算法(EGA)来解决

到稿日期:2014-07-05 返修日期:2014-10-23 本文受教育部高校博士点基金课题(20131201110002),国家自然科学基金天元青年基金(11326211)资助。

郑晶晶(1990-),女,硕士生,主要研究方向为组合优化算法设计、并行分布计算,E-mail:ziying280@tom.com;张 晶(1975-),女,硕士,副教授,主要研究方向为组合优化算法设计等;武继刚(1963-),男,博士,教授,主要研究方向为理论计算机科学、并行分布计算等。

分布式交互应用中服务器放置问题，通过合理放置来降低全局的交互时长、提高交互质量。

本文在文献[13]的系统模型上，针对于服务器位置选择提出了求解此问题的模拟退火算法和禁忌搜索算法，并与文献[13]提出的遗传算法进行比较。实验结果表明，本文提出的模拟退火算法和禁忌搜索算法求得的近似解都优于遗传算法。

2 问题描述及计算模型

给定一个分布式交互应用的所在网络用一个图 $G=(V,E)$ 来表示，V 代表图中所有节点，$E\subseteq V\times V$ 代表连接节点之间的边，边的权重代表信息传输延迟。若信息从节点 u 传输到节点 v 经过多条边，u 到 v 的总传输延迟为经过的所有边的权重之和，记为 $d(u,v)$。若 $u=v$，则 $d(u,v)=0$。假定 $C\subseteq V$ 代表应用中所有用户的集合，当服务器位置被固定之后，每一个用户被分配给唯一的一个服务器来发送操作指令和接收状态更新。对于每一个用户 $u\in C$，将分配给它的服务器记为 S_u。从用户 u 发出一个操作指令到另一个用户 v 接收状态更新需要经过 3 段路径：从用户 u 至它所属服务器 S_u，然后从 S_u 至用户 v 所分配的服务器 S_v，最后从 S_v 至用户 v。因此从用户 u 到用户 v 的交互路径长度可以用公式表示为 $d(u,v)=d(u,S_u)+d(S_u,S_v)+d(S_v,v)$，它也被用来代表用户 u 和用户 v 之间的交互时长。如果用户 u 和用户 v 被分配给同一个服务器，则 $d(u,v)=d(u,S_u)+d(S_v,v)$。用户 u 发出指令到它本身接收到状态更新所用的交互时长为 $d(u,u)=2d(u,S_u)$。假设单位时间内每个用户发出一个操作指令，网络中的其他所有用户都需接收到状态更新，以此来保证用户之间的完全交互，那么分布式交互应用中全局交互质量可以用完全交互的平均交互时长 M 来进行衡量，公式为：

$$M=\frac{1}{|C|^2}\sum_{u\in C}\sum_{v\in C}(d(u,S_u)+d(S_u,S_v)+d(S_v,v))$$

其中，$|C|$ 代表用户的总数量。M 的值越小，说明此应用的交互性能越好。因为 $\frac{1}{|C|^2}$ 是常量，降低 M 的值就等价于降低所有用户之间完全交互的总交互时长 D，公式为：

$$D=\sum_{u\in C}\sum_{v\in C}(d(u,S_u)+d(S_u,S_v)+d(S_v,v))$$

综上所述，分布式交互应用中的服务器放置问题可以描述为：给定一个网络 $G=(V,E)$，用户集合 $C\subseteq V$，服务器放置的候选位置集合 $S\subseteq V$，且每个候选位置至多只能放置一个服务器，服务器放置位置集合 $S'\subseteq V$，集合 S' 含有 k 个元素，即放置 k 个服务器，目标是找到集合 $S'\subseteq S$ 放置 k 个服务器使得值 M 最小。本文提出了使用模拟退火算法和禁忌搜索算法分别求解 k 个服务器的放置位置。

3 模拟退火算法

模拟退火(Simulated Annealing，SA)算法是一种启发式的随机寻优算法，它源于固体退火原理，从一个给定的初始高温开始，利用具有概率突跳特性的 Metropolis 抽样准则在解空间中进行随机搜索，伴随温度的不断下降重复抽样过程，最终得到问题的全局最优解[14]。

在 SA 算法初始化阶段，首先采用自然数编码表示法对状态进行表示。使用 $L[1:n]$ 表示问题的一个状态(即一个解)，其中 n 为候选位置总数，$L[i]=1$ 代表候选位置 i 被选中放置服务器资源，$L[i]=0$ 表示未选中，状态 $L[1:n]$ 满足 n 个候选位置中有且仅有 k 个位置被选中放置服务器。然后，采用 k-median 问题[9]的贪心算法寻找一个初始近似解。该贪心算法思想是先令服务器放置位置集合 S′为空，每一轮求出所有用户离自身最近的服务器位置的距离的总和，从候选位置集合 S 中选取一个增益最大的位置放入集合 S'，直到集合 S' 有 k 个位置。算法的形式化表述如下：

```
Algorithm 1 Greedyinitial  /* Finding an approximate solution* /
1. for i:=1 to n do
2.    L[i]:=0;
3. for i:=0 to C do
4.    Nearest[i]=0;
5. for i:=0 to k do //循环 k 轮选出 k 个位置
6.   total_distance←∞;
7.   for all s∈S∩L[s]=0 do
8.     distance=0;
9.     for i:=0 to C do
10.        if d(c,Nearest[c])>d(c,s) then
11.          distance+=d(c,s);
12.        else
13.          distance+=d(c,Nearest[c]);
14.     if total_distance>distance then
15.        Total_distance=distance,s*=s;
16.     L[s*]=1;//此轮选中位置 s*
17.   for i:=0 to C do
18.     if d(c,Nearest[c])>d(c,s*) then
19.        Nearest[i]=s*;//在选中位置集合中为每个用户计算离
           其最近的服务器
```

假设候选位置个数为 n，用户数为 C，服务器个数为 k，初始近似解生成算法的时间复杂度为 $O(k\cdot n\cdot C)$。

SA 算法目标函数计算涉及到为所有用户分配服务器的问题，状态 $L[1:n]$ 的目标函数则为相应解的总交互时长函数 $D(L[1],L[2],\cdots,L[n])$。为了计算解的目标函数值，为用户分配服务器最简单的算法就是最近分配法，即将每一个用户单独分配给离其位置最近的服务器，另一种算法为文献[13]使用的贪心分配算法。

在对 SA 算法的邻域进行构造时，修改解的状态的操作为对选中位置集合 S' 中的元素与未选中位置集合 $S-S'$ 中的元素进行互换，以此来得到当前解的邻域解。每次执行上述操作后，都需要重新为所有用户分配服务器，计算目标函数值。SA 算法采用了一种特殊的 Metropolis 准则的邻域移动方法，它以一定的概率来决定当前解是否移向新解。若新解的目标函数值小于历史最优解的目标函数值，则进行无条件移动；否则，依据一定的概率进行有条件移动。

SA 算法外循环过程通过降温函数来控制温度的下降。当温度很高时，当前邻域中几乎所有的解都会被接受，SA 进行广域搜索；当温度变低时，当前邻域中越来越多的解将被拒绝，SA 进行局域搜索。若温度下降过快则可能导致过早陷入局部最优，因此选择合理的降温函数能够帮助提高 SA 算法的性能，在此降温函数采用 $t=t*T$，其中降温系数 $T\in(0.95,0.99)$。内循环次数设定为一个常数 N，在同一温度下，内循环迭代设相同次数。

使用以上规则构造的模拟退火算法在本文中记作 SA，该算法的形式化描述如下：

```
Algorithm 2 SA   /* Simulated annealing to refine the initial solution */
Input:Meridian Dataset
Output:Location state:L[1:n]
       Assignments:S[1:k]
       Total interaction time:D
1. Begin
2.   for i:=0 to m do
3.     S[i]:=0;
4.   Greedyinitial();//使用贪心 k-median 算法产生初始解
5.   bestmintime=assignment(L[1:n])
6.   t:=T0;//设置初始温度
7.   repeat
8.     for m:=0 to N do //内循环次数 N
9.       neighborhood();//产生邻域可行解
10.      dif1=assignment(L[1:n])-bestmintime;
11.      if (dif1 < 0) //此解为历史最优解,接受
12.        premin=assignment(L[1:n]);//更新当前最优解
13.        bestmintime=premin//更新历史最优解
14.      else
15.        dif2=assignment(L[1:n])-premin;
16.      if (dif2 < 0) //此解为临域最优解,接受
17.        premin=assignment(L[1:n]);
18.      else //以一定的概率决定是否接受此解
19.        if (exp((-dif2)/ t) > p) //接受此解,p 为(0,1) 之间的
           随机数
20.          premin=assignment(L[1:n]);
21.        else //拒绝此解
22.          premin=premin;
23.   end for;
24.   t=t * T;//降温
25.   until(t≤TF);//外循环终止条件
26.   D=bestmintime;
27.   output L[1:n],S[1:k],D;
28. End
```

其中,$T0$ 代表初始温度,TF 代表终止温度,N 代表内循环的次数,T 代表降温系数,k 是内循环次数的计数变量,t 是当前温度值的变量。

4 禁忌搜索算法

禁忌搜索(Tabu Search,TS)算法是一种全局逐步寻优算法。其求解的过程是从一个初始可行解出发,然后在邻域中搜索较佳解或是移动到较差的区域搜索该区域最佳解,并且记录曾经搜寻的路径,作为下次搜索的依据,以避免陷入局部最优解中。通过引入特殊存储结构的禁忌表和相应的禁忌准则来避免迂回搜索,并通过特赦准则来赦免一些被禁忌的优良状态,从而实现全局优化[15]。

使用禁忌搜索算法对分布式应用服务器放置问题进行求解的设计如下。

(1)为了加快禁忌搜索算法的收敛,使用算法 Greedyinitial 产生一组初始解,生成一组候选位置选择的序列 $L[1:n]$,然后根据为用户选择分配服务器的方案计算出当前解的目标函数值,即总交互时长。

(2)在禁忌搜索算法中,每一轮搜索都是基于当前解的邻域。邻域是一种移动操作,移动是从当前解产生新解的途径,从当前解可以进行的所有移动构成邻域。在本文中通过随机选择一个未选中的候选位置 i(即 $L[i]=0$)替代已被选中的候选位置 j(即 $L[j]=1$)来生成当前解的邻域解,即令 $L[i]=1$,$L[j]=0$,将位置选择置换关系 $i \to j$ 作为禁忌表中的禁忌对象。

(3)假设已经产生当前解 L_{cur} 的候选解集,根据特赦准则来评价当前解的候选解,更新当前解与禁忌表。在服务器放置问题中目标函数值为总交互时长,所以评价候选解时,目标函数值越小,表示候选解质量越好。若候选解的最好解的目标值优于当前的最优目标值 L_{best},不管其禁忌属性如何,更新为当前最优解 L_{best} 并更新禁忌表,否则选择候选解中不被禁忌的最好解,更新该解为当前解并更新禁忌表。当所有候选对象均被禁忌,则选择禁忌表中最早进入的对象对其取消禁忌属性。

(4)判断是否达到指定的迭代步长,未达到则重复(2)、(3),否则输出服务器放置位置、用户分配服务器方案以及总交互时长的最终结果。

使用禁忌搜索算法解决分布式应用的服务器放置问题的算法流程图,如图 1 所示。

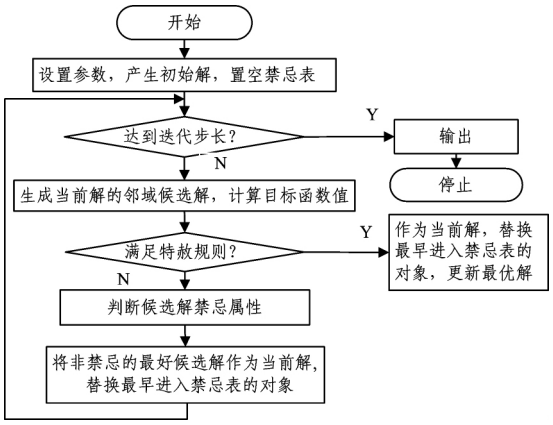

图 1 禁忌搜索算法的算法流程图

使用以上规则构造的遗传算法在本文中记作 TS。

5 实验结果及分析

本文将提出的 SA 算法和 TS 算法与文献[13]提出的改进的遗传算法 EGA 进行性能比较,实验中的数据依然采用文献[13]中使用的 Meridian 数据集,Meridian 数据集采集了现实网络中随机选取的 2500 个节点之间的链路时延值,数据通过三角矩阵方式进行组织,单位为 μs。实验操作同文献[13],去除了数据集中信息不完整的节点后随机选取了其中 1000 个节点进行实验。实验假设网络中任意一个节点上均置一个用户,所有节点都是服务器放置的候选位置。实验中,在得出服务器放置位置后,分别采用两种分配策略为用户进行服务器分配,即最近分配策略和贪心分配策略。

本文中所涉及的 SA 算法、TS 算法和 EGA 算法均使用 C 语言实现,在英特尔酷睿处理器、2.16GHz CPU、2.0GB RAM 和 Windows 7 操作系统上运行。

在 EGA 算法中各参数使用文献[13]中的参数配置,种群大小设为 80,最大迭代次数设为 100,交叉操作中的交叉概率为 0.15。在 SA 算法中各参数的设置是根据多次实验选取的较好值,其中初始温度 $T0$ 取 100,终止温度 TF 取 1,降温系数 T 取 0.95,迭代次数 N 取 100;在 TS 算法中,邻域大小设为 100,迭代次数设为 200,禁忌表的长度设为 20。以下的

实验结果均是在 30 次随机实验的基础上求得的，图 2 和图 4 的实验结果是使用贪心分配策略完成的，图 3 的实验结果是使用最近分配策略完成的。

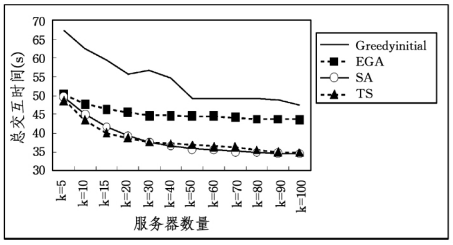

图 2　贪心分配策略下总交互时间与服务器数量之间的关系

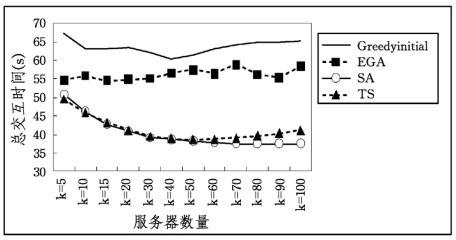

图 3　最近分配策略下总交互时间与服务器数量之间的关系

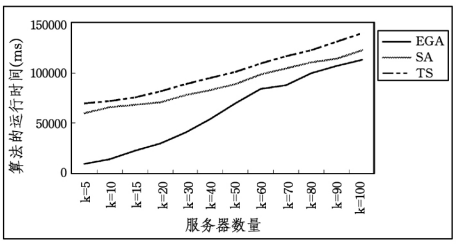

图 4　算法的运行时间与服务器数量之间的关系

图 2 和图 3 是对不同算法求得解的质量进行的比较，横坐标是服务器的数量，纵坐标为用户的总交互时间。从图 2 和图 3 可以看出，无论使用哪一种分配策略，SA 算法和 TS 算法所求得的解质量都明显优于 EGA 算法，并且随着服务器数量的增长，优势在逐渐扩大，解的差异逐渐明显。使用 EGA 算法时，尽管服务器数量增大却不能使得总交互时间减少。在图 2 中可以看到，当服务器数量 $k=100$ 时使用 EGA 算法与当服务器数量 $k=15$ 时使用其他两种算法求得的总交互时长相接近。这说明与 EGA 算法相比，SA 算法和 TS 算法能在节省服务器资源的条件下为分布式应用提供更好的交互性能。在相同的服务器条件下，与 EGA 算法相比，SA 算法和 TS 算法求得的总交互时长平均缩短了 15.5% 和 15.2%。另外可以看出，无论使用哪一种分配策略，使用 SA 算法和 TS 算法得出的解的差异性并不大，但当 k 达到 100 时，SA 算法能产生最小的总交互时长，解的改进幅度达到 20.6%。

图 4 是在使用贪心分配策略下对 EGA 算法、SA 算法和 TS 算法的运行时间进行比较，横坐标是服务器数量的大小，纵坐标为程序的运行时间。图中运行时间越小，表示算法执行得越快。从图 4 可以看出 EGA 算法的运行时间最短，其次是 SA 算法，TS 算法的运行时间最长。在服务器数量很小时，EGA 算法运行时间很短，而当服务器数量逐渐增大时，它的运行时间显著增大，与其他两种算法的运行时间差异逐渐减小。EGA 算法的平均运行时间比 SA 算法、TS 算法的平均运行时间分别少 2.7s、3.9s。

实验结果显示，在解决分布式交互应用中的服务器放置问题上，模拟退火算法和禁忌搜索算法求得解的质量较好，更大程度地缩短了总交互时长。而禁忌搜索算法和模拟退火算法的运行时间相对于遗传算法的运行时间较长。遗传算法由于赋予了一个较好的初始种群，相对于其他两种算法而言降低了运行时间，它能在一个更短的时间内得出较好的放置方案。

结束语　为了降低分布式交互应用中的网络延迟、提高用户之间的交互质量，针对服务器放置问题，本文提出了两种启发式的算法，分别为模拟退火算法和禁忌搜索算法，并与前人提出的改进的遗传算法进行比较。实验结果表明，模拟退火算法和禁忌搜索算法在解决分布式交互应用中的服务器放置问题上有着较明显的优势，求得解的改进幅度平均可达到 15.5%和 15.2%，大大缩短了全局的交互时长，而遗传算法的运行时间则相对较短。故对于服务器放置问题，在可以接受的算法运行时间的范围内，我们建议使用模拟退火或者禁忌搜索算法来求解问题相应的解。

参 考 文 献

[1] Webb S D, Soh S, Lau W. Enhanced mirrored servers for network games[C]// Proceedings of the 6th ACM SIGCOMM Workshop on Network and System Support for Games. 2007: 117-122

[2] Ahmad L, Boukerche A, Hamidi A A, et al. Web-based e-learning in 3D large scale distributed interactive simulations using HLA/RTI[C]// Proceedings of IEEE International Conference on IPDPS. 2008: 1-4

[3] Bouras C, Philopoulos A, Tsiatsos T. e-Learning through distributed virtual environments[J]. Network and Computer Applications, 2001, 24(3): 175-199

[4] Beigbeder T, Coughlan R, Lusher C, et al. The effects of loss and latency on user performance in unreal tournament[C]// Proceedings of the 3rd Workshop on Network and System Support for Games. 2004: 144-151

[5] Delaney D, Ward T, McLoone S. On consistency and network latency in distributed interactive applications [J]. A survey-Part I. Presence: Teleoperators & Virtual Environments, 2006, 15(2): 218-234

[6] Briceno L D, Siegel H J, Maciejewski A A, et al. Robust resource allocation in a massive multiplayer online gaming environment [C]// Proceedings of the 4th International Conference on Foundations of Digital Games. 2009: 232-239

[7] Cronin E, Filstrup B, Kurc A R. A distributed mutiplayer game server system[R]. Technical report, University of Michigan, 2001

[8] Jay C, Glencross M, Hubbold R. Modeling the effects of delayed haptic and visual feedback in a collaborative virtual environment [J]. ACM Transaction on Computer-Human Interaction, 2007, 14(2): 1-31

[9] Lee K-W, Ko B-J, Seraphin Calo. Adaptive server selection for large scale interactive online games[J]. Computer Networks, 2005, 49(1): 84-102

[10] Laoutaris N, Smaragdakis G, Oikonomou K. Distributed placement of service facilities in largescale networks[C]// Proceedings of IEEE /nternational Conference on INFOCOM. 2007: 2144-2152

(下转第 121 页)

性属性进行验证时，验证结果往往与通信双方每方只有一个 agent 时的结果并不一致。

基于 Promela 模型 Spin 检测安全性和活性时，还可以采取其它一些方面应对状态爆炸问题。如：利用工具 Spin 选择 safety state properties 对 Promela 模型进行安全性检测，由于可采用穷举算法对状态空间进行可达性分析，因此对于大规模的系统，可可采用 Bit State Hashing[13] 等方法有选择地搜索部分状态空间。一般地，如果安全性质不满足，则状态数不多；而安全性质满足时，状态数量多，状态爆炸的问题显现出来。因此，在检测时，使用 Promela 中的 atomic 和 d_step，尽量使用同步 channel、Spin 里的 partial order reduction 和 slicing algorithm 等。最重要的是建立的模型既要能反映原协议的时序逻辑，又要足够的抽象，没有冗余计算和冗余数据。利用工具 Spin 对 Promela 模型进行活性检测时，可以直接选择 liveness 对含有标号为 progress 或 accept 的模型进行活性验证。

在模拟多个 agent 运行时，对图 2 中的时序进行修正，使得两个主体间的通信能够完成协议所规定的要求，则在 Promela 建模时，要加入对通道的限制如 xr 和 xs，此时运行的模拟如图 6 所示。从图 6 中可以看出，一组(Alice0，Bob3)和另一组(Alice1，Bob2)之间 3 个消息的时序是正确的。

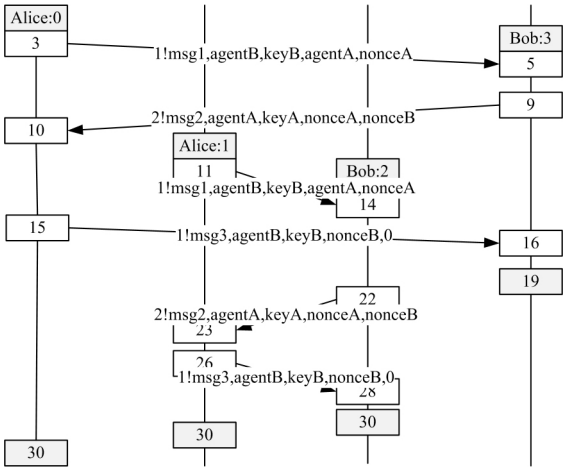

图 6　修正后的模拟运行图

结束语　本文分析了通信主体数量增加而引发的状态爆炸问题，提出了组合式的抽象验证方法。根据待验证 LTL 性质，通过对各个通信主体建立抽象模型，再检验组合抽象模型，对比了组合抽象前后的状态数量的变化。最后通过实例说明，组合抽象模型检测可以使状态数量明显减少，有利于缓解状态爆炸问题。如果既考虑到模型又考虑到了验证性质，那将比单一应用某种技术能更好地节省内存空间，这些都是进一步研究的方向。

参 考 文 献

[1] Edmund M C，Oran G，Doron A P. Model Checking [M]. Cambridge：MIT Press，1999

[2] 林惠民，张文辉. 模型检测：理论、方法与应用[J]. 电子学报，2002，30(12A)：1907-1912
Lin Hui-min，Zhang Wen-hui. Model Checking：Theories，Techniques and Applications[J]. Acta Electronica Sinica，2002，30(12A)：1907-1912

[3] Edmund M C，Oran G，David E L. Model checking and abstraction [J]. ACM Transactions on Programming Languages and Systems，1994，16(5)：1512-1542

[4] 刘志锋，孙博，周从华. 概率实时时态认知逻辑模型检测中抽象技术的研究[J]. 电子学报，2013，41(7)：1343-1351
Liu Zhi-feng，Sun Bo，Zhou Cong-hua. Abstraction in Model Checking Probabilistic Real-Time Temporal Logic of Knowledge[J]. Acat Electronica Sinica，2013，41(7)：1343-1351

[5] Grumberg O，Vardi M Y，Sifakis J，et al. 2010 CAV award announcement[J]. Formal Methods in System Design，2012，40(2)：117-120

[6] Mendoza L E，Capel M I，Perez M A. Conceptual framework for business processes compositional verification [J]. Information and software technology，2012，54(2)：149-161

[7] 曾红卫，缪淮扣. 构件组合的抽象精化验证 [J]. 软件学报，2008，19(5)：1149-1159
Zeng Hong-Wei，Miao Huai-Kou. Verification of Component Composition Based on Abstraction Refinement[J]. Journal of Software，2008，19(5)：1149-1159

[8] Carbone R. LTL model-checking for security protocols[J]. AI communications，2011，24(3)：385-396

[9] 高建华，蒋颖. 基于余归纳的最小 Kripke 结构的求解[J]. 软件学报，2014，25(1)：16-26
Gao Jian-hua，Jiang Ying. Coinduction-Based Solution for Minimization of Kripke Structures[J]. Journal of Software，2014，25(1)：16-26

[10] Gerard J H. The SPIN Model Checker：Primer and Reference Manual [M]. Boston：Addison-Wesley，2004

[11] 吕威，黄志球，陈哲，等. ESpin：基于 SPIN 的 Eclipse 模型检测环境[J]. 计算机工程与应用，2013，49(7)：45-51
Lv Wei，Huang Zhi-qiu，Chen Zhe，et al. ESpin：SPIN-based Eclipse model checking environment[J]. Computer Engineering and Applications，2013，49(7)：45-51

[12] Patig S，Stolz M. A pattern-based approach for the verification of business process descriptions [J]. Information and Software Technology，2013，55(1)：58-87

[13] Ikeda，Satoshi，Jibiki，et al. Coverage Estimation in Model Checking with Bitstate Hashing[J]. IEEE Transactions on Software Engineering，2013，39(4)：477-486

(上接第 98 页)

[11] Zhang Lu，Tang Xue-yan. Client assignment for improving interactivity in distributed interactive applications[C]// Proceedings of IEEE international Conference on INFOCOM. 2001：3227-3235

[12] Zhang Lu，Tang Xue-yan. The client assignment problem for continuous distributed interactive applications[C]// Proceedings of IEEE Conference on ICDCS. 2001：203-214

[13] Zhen Han-ying，Tang Xue-yan. An Enhanced Genetic Algorithm for Server Placement in Distributed Interactive Applications[C]// Proceedings of IEEE Conference on Parallel and Distributed Systems. 2012：596-603

[14] Gelfand S B，Mitter S K. Analysis of simulated anneling for optimization[J]. Decision and Control，1985，24(1)

[15] Rolland E，Schilling D A，Current J R. An efficient tabu search procedure for the p-Median problem[J]. European Journal of Operational Research，1997，96(2)：329-342