[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
H04L 29/06 (2006.01)

[12] 发明专利说明书

专利号 ZL 03801496.3

[45] 授权公告日 2009 年 10 月 28 日

[11] 授权公告号 CN 100556024C

[22] 申请日 2003.1.17 [21] 申请号 03801496.3

[30] 优先权

[32] 2002.1.24 [33] US [31] 10/057,516

[86] 国际申请 PCT/GB2003/000167 2003.1.17

[87] 国际公布 WO2003/063447 英 2003.7.31

[85] 进入国家阶段日期 2004.5.10

[73] 专利权人 国际商业机器公司

地址 美国纽约

[72] 发明人 乔尔·沃尔夫 菲利普·于

[56] 参考文献

US6282569B1 2001.8.28

CN1254886A 2000.5.31

US6092178A 2000.7.18

审查员 李普昕

[74] 专利代理机构 中国国际贸易促进委员会专利商标事务所

代理人 付建军

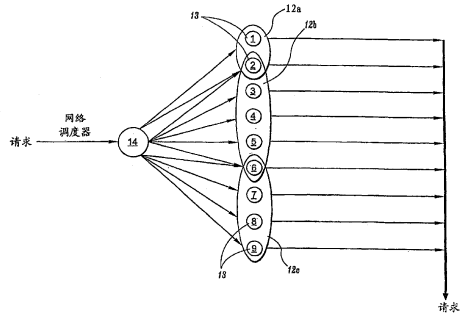
权利要求书 4 页 说明书 25 页 附图 12 页

[54] 发明名称

用于网络场通信控制的方法和网络场

[57] 摘要

公开了一种用于控制具有多个网站和服务器的网络场的方法，该方法包括：将从所述网站接收的客户请求分类成多个类别，所述类别包括可共享客户请求和不可共享客户请求；以使所述服务器中的任一个都可以处理从不同所述网站接收的可共享客户请求的方式传送所述可共享客户请求；以及将所述不可共享客户请求从特定所述网站仅传送到所述特定网站所被分配的特定服务器。



1. 一种用于控制具有多个网站和服务器并且还具有静态过程的网络场的方法，所述方法包括:

将由所述网站接收的客户请求分类成多个类别，所述类别包括可共享客户请求和不可共享客户请求;

以使所述服务器中的任一个都可以处理从所述网站中的不同网站接收的可共享客户请求的方式传送所述可共享客户请求；以及

将所述不可共享客户请求从所述网站中的特定网站仅传送到所述特定网站所被分配的特定服务器，

其中，传送所述不可共享客户请求的步骤包括所述静态过程根据对来自每个所述网站的可共享和不可共享客户请求的预测需求，将所述网站中的特定网站分配给特定所述服务器。

2. 如权利要求 1 所述的方法，还包括目标过程，所述目标过程包括：在给定所述网站到所述服务器的分配和当前客户请求负载的情况下，为每个所述客户请求，从每个所述客户请求所要分配的所述服务器中确定最优服务器，从而最小化任何给定时刻的平均客户响应时间。

3. 如权利要求 2 所述的方法，其中，所述目标过程通过在以下约束条件下:

$$\sum (x_{i,j} + y_{i,j}) \in \{0,\ldots,L_j\},$$

$$\sum_{j=1}^{N} x_{i,j} = c_i,$$

如果$a_{i,j}$=0则$x_{i,j}$=0,

$$\sum_{j=1}^{N} y_{i,j} = d_i,$$

以及

如果$b_{i,j}$=0则$y_{i,j}$=0

最小化函数

$$\sum_{j=1}^{N} R_j \left(\sum_{i=1}^{M} (x_{i,j} + y_{i,j}) \right)$$

来实现，其中，M 是网站数，N 是服务器数，R_j 是作为服务器 j 上的客户到达速率的函数的预期响应时间，$x_{i,j}$ 是表示可能由服务器 j 处理的网站 i 的可共享请求的假设数的决策变量，$y_{i,j}$ 是表示可能由服务器 j 处理的网站 i 的不可共享请求的假设数的决策变量，L_j 是服务器 j 的最大容许负载，c_i 是来自网站 i 的可共享客户请求的当前数，d_i 是来自网站 i 的不可共享请求的当前数，$a_{i,j}$ 是表示来自网站 i 的可共享请求是否可以传送到服务器 j 的索引，并且 $b_{i,j}$ 是表示来自网站 i 的不可共享请求是否可以传送到服务器 j 的索引。

4. 如权利要求 3 所述的方法，还包括:

创建和维护一个有向图，所述有向图包括一个哑节点和多个服务器节点，每个所述服务器节点对应于所述服务器之一;

指定所述服务器节点中表达式 $R_j\left(\sum_{i=1}^{M}(x_{i,j}+y_{i,j}+1)\right)-R_j\left(\sum_{i=1}^{M}(x_{i,j}+y_{i,j})\right)$ 最小的一个服务器节点为获胜节点; 以及

选择从所述哑节点到所述获胜节点的最短有向路径。

5. 如权利要求 2 至 4 中的任一项所述的方法，还包括动态过程，所述动态过程包括:

检查下一个客户请求;

调用所述目标过程从而确定哪个服务器是当前处理所述下一个客户请求的最优服务器; 以及

将所述下一个客户请求调度到所述最优服务器。

6. 如权利要求 5 所述的方法，还包括:

将所述客户请求接收到队列中; 并且

其中，所述动态过程还包括:

监视所述队列中的所述客户请求;

监视当前正在由所述服务器处理的客户请求;

对于每个第 j 服务器通过设置 $\dot{R}_j(z)=R_j\left(z+\sum(\ddot{c}_{i,j}+\ddot{d}_{i,j})\right)$ 来定义函数 $\dot{R}_j(z)$;

对于每个第 j 服务器通过设置 $\dot{L}_j=L_j-\sum_{i=1}^{M}(\ddot{c}_{i,j}+\ddot{d}_{i,j})$ 来定义修正容许负

载限值$\dot{L}_j$；以及

利用所述$\dot{R}_j(z)$函数和修正容许负载限值$\dot{L}_j$来调用所述目标过程，以在以下约束条件下：

$$\sum_{i=1}^{M}(\dot{x}_{i,j}+\dot{y}_{i,j})\in\{0,...,\dot{L}_j\},$$

$$\sum_{j=1}^{N}\dot{x}_{i,j}=\dot{c}_i,$$

如果$a_{i,j}$=0则$\dot{x}_{i,j}$=0,

$$\sum_{j=1}^{N}\dot{y}_{i,j}=\dot{d}_i,$$

以及

如果$b_{i,j}$=0则$\dot{y}_{i,j}$=0

最小化函数

$$\sum_{j=1}^{N}\dot{R}_j\left(\sum_{i=1}^{M}(\dot{x}_{i,j}+\dot{y}_{i,j})\right),$$

其中，$\dot{x}_{i,j}$是表示可能由服务器 j 处理的网站 i 的队列中可共享请求的假设数的决策变量，$\dot{y}_{i,j}$是表示可能由服务器 j 处理的网站 i 的不可共享请求的假设数的决策变量，$\dot{c}_i$是来自网站 i 的队列中可共享客户请求的当前数，$\dot{d}_i$是来自网站 i 的队列中不可共享请求的当前数，$\ddot{c}_{i,j}$是当前正在服务器 j 中处理的来自网站 i 的可共享客户请求的当前数，并且$\ddot{d}_{i,j}$是当前正在服务器 j 中处理的来自网站 i 的不可共享请求的当前数。

7. 一种网络场，具有多个网站和服务器，并且还包括：

用于从客户接收客户请求的部件；

用于处理所述客户请求以产生响应的部件；

用于将所述响应发送到所述客户的部件；

用于将所述客户请求分类成可共享客户请求和不可共享客户请求的部件；

用于以使所述服务器中的任一个都可以处理从所述网站中的不同网站接收的可共享客户请求的方式传送所述可共享客户请求的部件；以及

静态过程，用于将所述不可共享客户请求从所述网站中的特定网站仅传送到所述特定网站根据对来自每个所述网站的可共享和不可共享客

户请求的预测需求所被分配的特定服务器。

8. 如权利要求 7 所述的网络场，还包括：

网络调度器，包括用于执行目标过程、静态过程和动态过程的部件；

所述目标过程包括在给定所述网站到所述服务器的分配和当前客户请求负载的情况下，为每个所述客户请求，从每个所述客户请求所要分配的所述服务器中确定最优服务器，从而最小化任何给定时刻的平均客户响应时间，其中所述共享客户请求可以分配给任何所述服务器，并且其中所述不可共享客户请求根据所述不可共享客户请求所发起的哪个所述网站而仅可以分配给特定服务器；

所述静态过程包括将特定所述网站分配给特定所述服务器；并且

所述动态过程包括：

检查下一个客户请求；

调用所述目标过程从而确定哪个服务器是当前处理所述下一个客户请求的最优服务器；以及

将所述下一个客户请求调度到所述最优服务器。

用于网络场通信控制的方法和网络场

技术领域

本发明涉及网络场(web farm)中的负载均衡。

背景技术

网络场或联合体现在变得非常流行。网络场背后的关键概念是大量不同网站共享集中(pooled)资源这一观念。具体地说，它们典型地共享公共前端调度器来执行负载控制和分布客户请求。它们还共享多个网站服务器；并且它们共享后端带宽以将请求结果返回给客户。硬件和软件典型地由单个服务提供商或内容主(content host)拥有、操作和维护。目前提供或计划提供该服务的公司的例子包括 AT&T、Exodus、Intel、IBM、Qwest、Verio 和 Worldcom。

网络场的主要性能挑战是有效地均衡服务器上的负载从而最小化在系统上实现的平均响应时间。服务器的利用过度会导致客户请求的过度延迟。另一方面，服务器的利用不足是浪费的。最小化平均响应时间将具有最大化网络场所能实现的吞吐量的双重效应。这意味着前端调度器中的负载控制过程可以变得更宽容而拒绝较少的请求。

网络场的主要动机之一是由于集中资源而产生的效率。每个网站分配给由一个或多个服务器构成的集合。然而，现有技术假设网站分配在服务器之间进行分区：也就是，没有服务器处理来自多个网站的通信。

创建包含多个服务器的集合背后的基本原理是可行性：客户请求在各个站点之间的分布将典型地是高度非均匀的。一些站点将很有可能比其他站点受欢迎得多，并且热门网站的受欢迎度一般将大得足以将它分配给单个服务器将使该服务器过载。因此，为了实现可接受性能，将有必要将一些网站分配到多个服务器。

实现分区情况的负载均衡算法是公知的，但是该服务器分区导致较

小的灵活性，并且因此导致不是最优均衡的网站负载均衡。

发明内容

公开了一种用于控制具有多个网站和服务器的网络场的方法。该方法包括将从所述网站接收的客户请求分类成多个类别，所述类别包括可共享客户请求和不可共享客户请求，以使所述服务器中的任一个都可以处理从不同所述网站接收的可共享客户请求的方式传送所述可共享客户请求，以及将所述不可共享客户请求从特定所述网站仅传送到所述特定网站所被分配的特定服务器。

该方法的另一方面包括目标过程(Goal procedure)，所述目标过程包括：在给定所述网站到所述服务器的分配和当前客户请求负载的情况下，为每个所述客户请求，从每个所述客户请求所要分配的所述服务器中确定最优服务器，从而最小化任何给定时刻的平均客户响应时间。

在该方法的另一方面，所述目标过程通过在以下约束条件

$\sum(x_{i,j}+y_{i,j})\in\{0,\ldots,L_j\}$, $\sum_{j=1}^{N}x_{i,j}=c_i$, 如果$a_{i,j}$=0则$x_{i,j}$=0, $\sum_{j=1}^{N}y_{i,j}=d_i$, 以及如果$b_{i,j}$=0则$y_{i,j}$=0

下最小化函数$\sum_{j=1}^{N}R_j\left(\sum_{i=1}^{M}(x_{i,j}+y_{i,j})\right)$来实现，其中，**M** 是网站数，**N** 是服务器数，$\mathbf{R_j}$是作为服务器 **j** 上的客户到达速率的函数的预期响应时间，$\mathbf{x_{i,j}}$是表示可能由服务器 **j** 处理的网站 **i** 的可共享请求的假设数的决策变量(decision variable)，$\mathbf{y_{i,j}}$是表示可能由服务器 **j** 处理的网站 **i** 的不可共享请求的假设数的决策变量，$\mathbf{L_j}$是服务器 **j** 的最大容许负载，$\mathbf{c_i}$是来自网站 **i** 的可共享客户请求的当前数，$\mathbf{d_i}$是来自网站 **i** 的不可共享请求的当前数，$\mathbf{a_{i,j}}$是表示来自网站 **i** 的可共享请求是否可以传送到服务器 **j** 的索引(index)，并且 $\mathbf{b_{i,j}}$ 是表示来自网站 **i** 的不可共享请求是否可以传送到服务器 **j** 的索引。

该方法的另一方面包括创建和维护一个有向图，所述有向图包括一个哑节点和多个服务器节点，每个所述服务器节点对应于所述服务器之一，指定所述服务器节点中表达式$R_j\left(\sum_{i=1}^{M}(x_{i,j}+y_{i,j}+1)\right)-R_j\left(\sum_{i=1}^{M}(x_{i,j}+y_{i,j})\right)$最小的一个服务器节点为获胜节点，以及选择从所述哑节点到所述获胜节点

的最短有向路径。

该方法的另一方面包括静态过程，所述静态过程包括将特定所述网站分配给特定所述服务器。

在该方法的另一方面，所述静态过程根据对来自每个所述网站的可共享和不可共享客户请求的预测需求，将所述网站分配给特定服务器。

该方法的另一方面包括动态过程，所述动态过程包括检查下一个客户请求，调用所述目标过程从而确定哪个服务器是当前处理所述下一个客户请求的最优服务器，并且将所述下一个客户请求调度到所述最优服务器。

该方法的另一方面包括将所述客户请求接收到队列中，并且其中，所述动态过程还包括监视所述队列中的所述客户请求，监视当前正在由所述服务器处理的客户请求，对于每个第 j 服务器通过设置 $\dot{R}_j(z) = R_j\left(z + \sum(\ddot{c}_{i,j} + \ddot{d}_{i,j})\right)$ 来定义函数 $\dot{R}_j(z)$，对于每个第 j 服务器通过设置 $\dot{L}_j = L_j - \sum_{i=1}^{M}(\ddot{c}_{i,j} + \ddot{d}_{i,j})$ 来定义修正容许负载限值 $\dot{L}_j$，并且利用所述 $\dot{R}_j(z)$ 函数和修正容许负载限值 $\dot{L}_j$ 来调用所述目标过程，以在以下约束条件：

$\sum_{i=1}^{M}(\dot{x}_{i,j} + \dot{y}_{i,j}) \in \{0,...,\dot{L}_j\}, \sum_{j=1}^{N}\dot{x}_{i,j} = \dot{c}_i$，如果$a_{i,j}$=0则$\dot{x}_{i,j}$=0，$\sum_{j=1}^{N}\dot{y}_{i,j} = \dot{d}_i$，以及如果$b_{i,j}$=0则$\dot{y}_{i,j}$=0

下最小化函数 $\sum_{j=1}^{N}\dot{R}_j\left(\sum_{i=1}^{M}(\dot{x}_{i,j} + \dot{y}_{i,j})\right)$，其中，$\dot{x}_{i,j}$ 是表示可能由服务器 j 处理的网站 i 的队列中可共享请求的假设数的决策变量，$\dot{y}_{i,j}$ 是表示可能由服务器 j 处理的网站 i 的不可共享请求的假设数的决策变量，$\dot{c}_i$ 是来自网站 i 的队列中可共享客户请求的当前数，$\dot{d}_i$ 是来自网站 i 的队列中不可共享请求的当前数，$\ddot{c}_i$ 是当前正在服务器之一中处理的来自网站 i 的可共享客户请求的当前数，并且 $\ddot{d}_i$ 是当前正在服务器之一中处理的来自网站 i 的不可共享请求的当前数。

公开了一种可由机器读取的程序存储装置，它有形地实施一个可由机器执行的指令程序以执行用于控制具有多个网站和服务器的网络场的方法步骤，所述方法步骤包括将从所述网站接收的客户请求分类成多个类别，所述类别包括可共享客户请求和不可共享客户请求，以使所述服务器中的任一个都可以处理从不同所述网站接收的可共享客户请求的方

式传送所述可共享客户请求，以及将所述不可共享客户请求从特定所述网站仅传送到所述特定网站所被分配的特定服务器。

该装置的另一方面包括目标过程，所述目标过程包括：在给定所述网站到所述服务器的分配和当前客户请求负载的情况下，为每个所述客户请求，从每个所述客户请求所要分配的所述服务器中确定最优服务器，从而最小化任何给定时刻的平均客户响应时间。

在该装置的另一方面，所述目标过程通过在以下约束条件

$\sum(x_{i,j}+y_{i,j})\in\{0,\ldots,L_j\},\sum_{j=1}^{N}x_{i,j}=c_i$，如果$a_{i,j}$=0则$x_{i,j}$=0，$\sum_{j=1}^{N}y_{i,j}=d_i$，以及如果$b_{i,j}$=0则$y_{i,j}$=0

下最小化函数$\sum_{j=1}^{N}R_j\left(\sum_{i=1}^{M}(x_{i,j}+y_{i,j})\right)$来实现，其中，**M** 是网站数，**N** 是服务器数，R_j是作为服务器 **j** 上的客户到达速率的函数的预期响应时间，$x_{i,j}$是表示可能由服务器 **j** 处理的网站 **i** 的可共享请求的假设数的决策变量，$y_{i,j}$是表示可能由服务器 **j** 处理的网站 **i** 的不可共享请求的假设数的决策变量，L_j是服务器 **j** 的最大容许负载，c_i是来自网站 **i** 的可共享客户请求的当前数，d_i是来自网站 **i** 的不可共享请求的当前数，$a_{i,j}$是表示来自网站 **i** 的可共享请求是否可以传送到服务器 **j** 的索引，并且$b_{i,j}$是表示来自网站 **i** 的不可共享请求是否可以传送到服务器 **j** 的索引。

该装置的另一方面包括创建和维护一个有向图，所述有向图包括一个哑节点和多个服务器节点，每个所述服务器节点对应于所述服务器之一，指定所述服务器节点中表达式$R_j\left(\sum_{i=1}^{M}(x_{i,j}+y_{i,j}+1)\right)-R_j\left(\sum_{i=1}^{M}(x_{i,j}+y_{i,j})\right)$最小的一个服务器节点为获胜节点，并且选择从所述哑节点到所述获胜节点的最短有向路径。

该装置的另一方面包括静态过程，所述静态过程包括将特定所述网站分配给特定所述服务器。

在该装置的另一方面，所述静态过程根据对来自每个所述网站的可共享和不可共享客户请求的预测需求，将所述网站分配给特定服务器。

该装置的另一方面包括动态过程，所述动态过程包括检查下一个客户请求，调用所述目标过程从而确定哪个服务器是当前处理所述下一个客户请求的最优服务器，并且将所述下一个客户请求调度到所述最优服

务器。

该装置的另一方面包括将所述客户请求接收到队列中，并且其中，所述动态过程还包括监视所述队列中的所述客户请求，监视当前正在由所述服务器处理的客户请求，对于每个第 j 服务器通过设置 $\dot{R}_j(z)=R_j\left(z+\sum(\ddot{c}_{i,j}+\ddot{d}_{i,j})\right)$ 来定义函数 $\dot{R}_j(z)$，对于每个第 j 服务器通过设置 $\dot{L}_j=L_j-\sum_{i=1}^{M}(\ddot{c}_{i,j}+\ddot{d}_{i,j})$ 来定义修正容许负载限值 $\dot{L}_j$，并且利用所述 $\dot{R}_j(z)$ 函数和修正容许负载限值 $\dot{L}_j$ 来调用所述目标过程，以在以下约束条件：

$\sum_{i=1}^{M}(\dot{x}_{i,j}+\dot{y}_{i,j})\in\{0,...,\dot{L}_j\}$, $\sum_{j=1}^{N}\dot{x}_{i,j}=\dot{c}_i$, 如果$a_{i,j}$=0则$\dot{x}_{i,j}$=0, $\sum_{j=1}^{N}\dot{y}_{i,j}=\dot{d}_i$, 以及如果$b_{i,j}$=0则$\dot{y}_{i,j}$=0

下最小化函数 $\sum_{j=1}^{N}\dot{R}_j\left(\sum_{i=1}^{M}(\dot{x}_{i,j}+\dot{y}_{i,j})\right)$，其中，$\dot{x}_{i,j}$ 是表示可能由服务器 j 处理的网站 i 的队列中可共享请求的假设数的决策变量，$\dot{y}_{i,j}$ 是表示可能由服务器 j 处理的网站 i 的不可共享请求的假设数的决策变量，$\dot{c}_i$ 是来自网站 i 的队列中可共享客户请求的当前数，$\dot{d}_i$ 是来自网站 i 的队列中不可共享请求的当前数，$\ddot{c}_i$ 是当前正在服务器之一中处理的来自网站 i 的可共享客户请求的当前数，并且 $\ddot{d}_i$ 是当前正在服务器之一中处理的来自网站 i 的不可共享请求的当前数。

公开了一种网络场，包括：用于从客户接收客户请求的部件；用于处理所述客户请求以产生响应的部件；用于将所述响应发送到所述客户的部件；用于将所述客户请求分类成可共享客户请求和不可共享客户请求的部件；包括用于执行目标过程、静态过程和动态过程的部件的网络调度器，所述目标过程包括在给定所述网站到所述服务器的分配和当前客户请求负载的情况下，为每个所述客户请求，从每个所述客户请求所要分配的所述服务器中确定最优服务器，从而最小化任何给定时刻的平均客户响应时间，其中所述共享客户请求可以分配给任何所述服务器，并且其中所述不可共享客户请求根据所述不可共享客户请求所发起的哪个所述网站而仅可以分配给特定服务器，所述静态过程包括将特定所述网站分配给特定所述服务器，并且所述动态过程包括检查下一个客户请求，调用所述目标过程从而确定哪个服务器是当前处理所述下一个客户请求的最优服务器，并且将所述下一个客户请求调度到所述最优服务

器。

附图说明

图 1 示出网络场，其中包括网络调度器和多个服务器。

图 2 示出从哑节点到获胜节点的路径，目标过程的一部分。

图 3 是目标过程的流程图。

图 3b 是目标过程的初始化步骤的流程图。

图 3c 是目标过程的路径簿记(bookkeeping)步骤的流程图。

图 3d 是目标过程的图簿记步骤的流程图。

图 4 是动态过程的流程图。

图 5 是说明动态过程的例子中的网站符号的汇集表(glossary)。

图 6 示出相同例子中动态过程的有向图。

图 7 示出相同例子中静态过程的无向图。

图 8 示出静态过程中的邻域扩展试探法(neighborhood escape heuristic)。

图 9 示出静态过程的流程图。

具体实施方式

本发明通过允许多个站点的服务器分配之间的重叠来提供一种对网络场中的负载均衡问题的解决方案，并且以适合于各种站点拥有者的实用方式这样做。可以使用本发明的内容来代替公开于共同转让的共同未决的美国专利申请 09/374,539，申请日期为 1999 年 8 月 16 日，律师事务所号为 12793(YO999-307)，发明名称为“METHOD AND APPARATUS FOR LOAD BALANCING OF WEB CLUSTER FARMS(用于网络集群场的负载均衡的方法和装置)”，现在为美国专利 6374297 的负载均衡系统。

重叠网站集背后的基本原理是灵活性：网络场必须能够同时处理来自很多不同站点的客户请求，并且该客户需求典型地是突发性的。该突发性由于变化的网站受欢迎度和客户牵涉(custom mix)而可以按星期、

天和小时看到。有证据表示它也以小得多的原子时间单位如分钟甚至是秒发生。在此的要点是平均网站访问速率不足以捕获客户行为的变化性。一个网站的通信可能大幅涨落，并且一个网站可能较忙，同时另一个网站则不那么忙。希望是可以使处理若干站点的服务器改变这些网站之间的工作负载均衡以对这些动态通信作出反应。注意，分区设计不允许该反应能力。

本发明有效地集中服务器资源，并且采用用于前端调度器的智能方法，从而均衡这些服务器上的负载。本发明可推广至足以假设每个网站分配给由一个或多个服务器构成的集合，而且这些集合可以重叠。这意味着如果需要，给定服务器可以处理来自多个站点的客户请求。较不期望的替代方案是在各个站点之间对服务器进行分区，从而使每个服务器仅处理来自一个网站的请求。虽然效果差些，本发明也将工作于该较不优选的特殊情况。

然而，在网络场的情况下重叠服务器集不是没有问题。虽然适当高级服务器容纳适度数目的网站从技术的角度是合理的，但是从政策角度可能不十分合理。例如，即使服务器在多个站点之间进行过逻辑分区，电子商务贸易商也可能反对与其他贸易商共享处理交易请求的服务器。贸易商的反对将典型地基于保密的理由。幸运的是，该贸易商对于共享用于浏览请求的服务器较不敏感。为了使负载均衡过程是实用的并且因此是可实现的，它必须有效地处理该问题。如果必要，这可以通过将每个网站的工作负载分成两个类别来实现，其中一个类别是不可共享的，并且另一个是可共享的。这假定公共请求(由浏览者激发)可以与来自其他站点的请求共享服务器。它们也可以与来自相同网站的私有请求(由交易激发)共享服务器。这一关键细化向本发明提供即使在重叠分配的政策性变化问题的情况下也很好地执行的所需灵活性。

虽然本发明在此是按照两个级别的客户通信来描述的，但是本领域的技术人员应该明白，本发明可推广至具有可变网站到服务器限制的分级结构的通信。

注意，虽然公共和私有类别分别由浏览和交易请求来激发，但是这

些术语实际上是由是否能够共享来定义的。例如，对于接受逻辑分区概念的电子商务站点，交易请求将算作公共通信的一部分，因此根本将不存在任何私有通信。在另一个极端，即使对于浏览者也拒绝允许任何服务器共享的电子商务站点将把浏览请求算作私有通信的一部分，并且根本将不存在任何公共通信。当然，可能存在对于其浏览和交易术语没有任何字面意义的非商务网站。为简单起见，通信将根据可共享性分成公共或私有类别。

本发明利用重叠网站分配的可能性以非常有效地均衡网络场负载。该问题自然地分解成两个子问题，并且对每个子问题的求解都利用相同的基本优化方法以设置适当的目标。因此，对于本发明，存在三个主要组件，它们将分别称作目标过程、静态过程和动态过程。

目标过程是关键公共过程。它使用设计成在给定站点对服务器的分配和当前客户请求负载的情况下最小化任何给定时刻的平均客户响应时间的优化方法。该方法是由 Tantawi 等人提出的用于求解所谓的离散类约束可分离凸资源分配问题的方法的略微特殊情况，参见“**Optimal allocation of multiple class resources in computer systems(计算机系统中多类资源的最优分配)**”, **ACM Sigmetrics Conference(1988 年 5 月), pp. 253-260**。从而该方法将确定最优负载均衡目标。具体地说，输出是要由每个适当服务器处理的每个网站的公共和私有类别的客户请求的最优数目，并且因此通过求和是每服务器的所有客户请求的最优数目。该问题将需要相对频繁地进行求解。幸运的是，该求解方法较快(并且在性质上是增量的)。它可以处理异构服务器，这是重要的，因为随着时间的过去一些老的较慢服务器将不可避免地被新的较快服务器代替。目标过程是下面动态过程和静态过程组件的组成部分。

考虑公共和私有请求的要求，静态过程创建良好、有希望几乎最优的站点到服务器的分配。并且，该过程完成得越好，则目标过程中的平均响应时间就会越好。静态过程组件在执行中迭代性地调用目标过程。该方法可以在初始或增量模式下运行。当配置新的网络场时，初始模式是适当的。增量模式考虑限制分配变化数的约束，因此对于维护高质量

的网站到服务器分配是实用的。采用例如在 R. Garfinkel 等人的 "Integer Programming(整数规划)", John Wiley and Sons, New York, NY(1972)中公开的邻域扩展试探法。增量模式意味着周期性地运行，可能是大约每星期运行一次。重新配置网站到服务器分配显然不是琐细的任务。准确的频率将依赖于网站需求预测的变化性和进行新分配的成本。

动态过程根据静态过程组件的输出和波动的网站客户需求来在网络调度器中执行实时网络服务器路由选择。它根据目标过程来求解受限子问题以计算该理想化路由选择。它不转移先前调度过的请求的分配，这可能改善性能，但是将导致高得惊人的开销。

参照图 1，示出具有三个站点 12 和九个服务器 13 的假设网络场 10 的例子。网络调度器 14 将三个站点 12 的客户请求传送到适当的服务器。第一网站 12a 包括两个服务器(1 和 2)，并且与具有五个服务器(2 到 6)的第二网站 12b 重叠，并且第二网站 12b 又与具有四个服务器(6 到 9)的第三网站 12c 重叠。客户请求的结果通过管道回送给客户，然后客户可以发出另外的请求。

给定图 1 的说明性例子，现在详细描述本发明的三个主要组件将是有用的。

目标过程(目标)组件

首先定义一些符号标记将是有用的。设 M 表示网站的数目，为方便起见将总是以 i 索引。设 N 表示服务器的数目，以 j 索引。设{0, 1}M×N 矩阵 $A=(a_{i,j})$表示将网站 i 的公共请求分配给服务器 j 的潜在可能性。换句话说，如果网站 i 的可共享请求可以由服务器 j 处理，则 $a_{i,j}=1$，否则 $a_{i,j}=0$。类似地，设{0, 1}M×N 矩阵 $B=(b_{i,j})$表示将网站 i 的私有请求分配给服务器 j 的潜在可能性。换句话说，如果网站 i 的不可共享请求可以由服务器 j 处理，则 $b_{i,j}=1$，否则 $b_{i,j}=0$。实际上我们将假设处理网站 i 的私有请求的服务器 j 也可以处理该网站的公共请求。因此，如果 $b_{i,j}=1$，则 $a_{i,j}=1$。

与每个服务器 j 相关联的是作为客户到达速率的函数测量预期响应

时间的函数 R_j。该功能将依赖于服务时间分布，而服务时间分布又将依赖于处理器速度。函数 R_j 一般将具有递增特性和凸特性。对于在有关到达速率模式和服务时间的特定简化假设下计算或估算 R_j，存在如 Lavenberg 提出的经典队列方法，参见"Computer Performance Modeling Handbook(计算机性能建模手册)", Academic Press, New York, NY(1983)，在此将其全文引作参考，但是一般而言，该函数可能需要通过模拟试验或监视来估计。

根据该函数，可以假设在服务器 j 上将存在最大容许负载 L_j，其中，超过该阈值将导致 R_j 的值过大。当然，如果设置 $L_j=\infty$，则该最大负载约束将太松。假设在给定时刻对于网站 i 当前存在 c_i 个公共请求和 d_i 个私有请求。将这些请求进一步分解成服务器 j 上的网站 i 的 $c_{i,j}$ 个公共请求和 $d_{i,j}$ 个私有请求。因此，$c_i=\sum_{j=1}^{N}c_{i,J}$，并且每当 $a_{i,j}=0$ 时，$c_{i,j}=0$(不能从网站未被分配的服务器处理该网站的公共请求)。类似地，每当 $b_{i,j}=0$ 时，$d_{i,j}=0$。设 $C=\sum_{i=1}^{M}c_i$ 表示当前公共请求的总数，并且类似地，$D=\sum_{i=1}^{M}d_i$ 表示当前私有请求的总数。

在给定当前负载和网站到服务器分配的情况下，当在以下约束条件下最小化目标函数

$$\sum_{j=1}^{N}R_j\left(\sum_{i=1}^{M}(x_{i,j}+y_{i,J})\right) \qquad (1)$$

时，服务器负载可以被认为是最优均衡的，其中这些约束条件为：

$$\sum(x_{i,J}+y_{i,j})\in\{0,\ldots,L_j\}, \qquad (2)$$

$$\sum_{j=1}^{N}x_{i,j}=c_i, \qquad (3)$$

如果$a_{i,j}$=0则$x_{i,j}$=0, (4)

$$\sum_{j=1}^{N}y_{i,J}=d_i, \qquad (5)$$

以及

如果$b_{i,j}$=0则$y_{i,j}$=0, (6)

在此，$x_{i,j}$ 是表示可能由服务器 j 处理的网站 i 的公共请求的假设数的决策变量。类似地，$y_{i,j}$ 是表示可能由服务器 j 处理的网站 i 的私有请

求的假设数的决策变量。目标函数测量各个服务器上的预期响应时间之和，它与平均响应时间相差一个倍数常数。从优化问题的角度而言，该常数是无关的。约束条件(2)限制服务器 i 上的容许负载。约束条件(3)确保网站 i 公共请求的总数等于当前这些请求的实际数。约束条件(4)确保对于公共请求考虑网站到服务器分配。约束条件(5)到(6)是私有请求的对应要求。如果在最优解中 $X_j=\sum_{i=1}^{M}x_{i,j}$ 并且 $Y_j=\sum_{i=1}^{M}y_{i,j}$，则注意 X_j+Y_j 表示服务器 j 上的期望负载。我们最终的目标将是确保最优负载 X_j+Y_j 和实际负载 $\sum_{i=1}^{M}(c_{ij}+d_{ij})$ 对于每个服务器 j 总是相互接近。

上述优化问题是所谓的离散类约束可分离凸资源分配问题的特殊情况。这些类对应于各个站点的公共和私有请求。该问题由于约束条件(2)是离散的，由于约束条件(3)和(5)而是资源分配问题，并且由于约束条件(4)和(6)而类约束的。可分离性术语是指目标函数的性质，并且凸特性术语是显然的。如 Tantawi 等人所述，离散类约束资源分配问题可以使用图论优化过程来准确且高效地求解。

参照图 3，示出目标过程的优选实施例的流程图。假设存在可行解，过程共进行 C+D 步。创建例如图 2 中的有向图，并且在整个过程期间都将维护它。图的节点是服务器 1, ..., N 以及标注为“节点 0”的哑节点。对于每个 i，设置 $a_{i,0}=1$ 和 $b_{i,0}=1$，以及 $L_0=0$。我们将创建和修改部分可行解 $\{x_{i,j}\mid i=1,...M,j=0,...,N\}\cup\{y_{i,j}\mid i=1,...M,j=0,...,N\}$。最初，在图 3 中的步骤 100 和 105，对于每个 i 设置该部分可行解，以使得对于所有 j=1, ..., N，$x_{i,0}=c_i$，$y_{i,0}=d_i$，并且 $x_{i,j}=y_{i,j}=0$，因此，所有资源都驻留在哑节点上。在步骤 110 初始化有向图，并且在任何步骤，如果存在至少一个网站 i_1 满足

$$a_{i_1,j_1}=a_{i_1,j_2}=1, \tag{7}$$

$$x_{i_1,j_1}>0, \tag{8}$$

$$\sum_{i=1}^{M}(x_{i,j}+y_{i,j})<L_{j2}, \tag{9}$$

或者，可替换地，存在至少一个网站 i_1 满足

$$b_{i_1,j_1}=b_{i_1,j_2}=1, \tag{10}$$

$$y_{i_1,j_1}>0, \tag{11}$$

以及条件(9)，则有向图具有从节点 $j_1 \in \{0, \ldots, N\}$到节点 $j_2 \in \{0, \ldots, N\}$的有向弧。条件(7)表示节点 j_1 和 j_2 可以处理网站 i_1 的公共请求。条件(8)表示网站 i_1 的公共请求已分配给节点 j_1。条件(9)表示该请求可以转移给节点 j_2，而不超过那个节点上的负载限制。条件(10)、(11)和(9)是私有请求的对应要求。注意，可以存在以节点 0 为起点的有向弧，但是不存在以节点 0 为终点的有向弧。从而接收节点将总是真实服务器，而不是哑节点。

该方法的一般步骤在对于其存在从 0 到 j 的有向路径的所有节点 $j \in \{1, \ldots, N\}$中查找第一差值

$$R_j\left(\sum_{i=1}^{M}(x_{i,j}+y_{i,j}+1)\right)-R_j\left(\sum_{i=1}^{M}(x_{i,j}+y_{i,j})\right) \qquad (12)$$

最小的获胜节点。此所谓的第一差值是连续函数的导数的离散模拟。第一差值是由于 R_j 的凸特性而对于每个 i 在 j 中是非递减的。如果不存在该节点，则过程以不可行解终止。否则，选择从 0 到获胜节点的最短有向路径。针对该路径中的每个有向弧(j_1, j_2)，对于适当的网站 i_1，$x_{i_1,j_1}+y_{i_1,j_1}$的值减 1，并且$x_{i_1,j_2}+y_{i_1,j_2}$的值增 1(由于公共请求的增减或者私有请求的减增)。在所有有向弧上执行该步骤具有从哑节点删除一个单位的负载，并且将一个单位的负载加到获胜节点的效应。对于中间节点的负载没有有效效应(net effect)。因此，哑节点用作资源的集结区域(staging area)，在每个步骤，将其中之一释放到服务器节点中。然后对有向图执行簿记，这可能修改一些有向弧以及潜在断连的某些节点，并且重复该步骤。在(C+D)个步骤之后，过程以对原始离散类约束资源分配问题的最优解终止。可行性由于有向图中各弧上的条件而得到保证。

更具体地参照图 3，目标过程的优选实施例在步骤 100 通过对于所有站点 i 和服务器 j 将变量 $x_{i,j}$和 $y_{i,j}$设为 0 来开始。该过程在执行中将增大和减小这些变量。在步骤 105，将哑节点初始化为节点 0。所有资源初始都放在这里，从而，设置 $x_{i,0}=c_i$ 以及 $y_{i,0}=d_i$。在步骤 110，根据方程(7)-(11)初始化连接 N+1 个节点的有向图，下面将参照图 3b 对此进行更详细的描述。可以从此确定将存在从哑节点 0 到如下节点 j 的有向弧，其中对于节点 j，存在至少一个具有 $a_{i,j}=1$ 或 $b_{i,j}=1$ 的网站 i。在过

程的执行中，将修改该图。在步骤 115，设置 k=1。该变量将在分配资源的时候对资源单位进行计数。在步骤 120，将 m 初始化为无穷大，并且将 j 初始化为 1。在步骤 125，判定是否存在从 0 到节点 j 的有向路径。如果存在，则在步骤 130，对于该服务器 j 估算方程(12)，并且将它与 m 比较。如果小，则在步骤 135 设置 j*=j，并且将 m 设为方程(12)的值。这是 m 的当前最小值。然后，进入步骤 140。如果在步骤 125 没有任何路径，或者如果步骤 130 中的方程(12)估算为 m 或更大，则也进入步骤 140。步骤 140 增大 j。在步骤 145，检查 j 是否为 N 或更小。如果是，则返回到步骤 125。否则，进入步骤 150，其中，将 m 与无穷大进行比较。如果 m 不小于无穷大，则不存在从哑节点到任何节点的路径，因此不存在任何对该问题的可行解。从而，过程在步骤 155 停止。否则，进入步骤 160 并且查找最短该路径。在步骤 165，对该路径执行簿记，增大和减小 $x_{i,j}$ 和 $y_{i,j}$ 的适当值。在这样做当中，一个单位的资源将从哑节点删除，并且一个单元的资源将加到服务器 j*，下面将参照图 3c 对此进行更详细的描述。然后，在步骤 170，对有向图执行簿记，再次确保满足方程(7)-(11)。在该过程中可能增加一些有向弧，并且可能删除一些有向弧，下面将参照图 3d 对此进行更详细的描述。在步骤 175，增大 k。在步骤 180，测试 k 以查看它是否为 C+D 或更小。如果否，则在步骤 185 过程以最终解 $x_{i,j}$ 和 $y_{i,j}$ 完成，否则返回到步骤 120 以继续过程。

参照图 2，示出类约束资源分配过程的有向图中的路径。注意，在此没有示出该图的所有节点和有向弧。负载正在从哑“节点 0”20 转移到获胜(服务器)节点，即其第一差值最小的连接到节点 0 的节点。所示的有向路径旨在是最短有向路径。有向弧编号为 1 至 5。在本图中，某网站的一个新(公共或私有)请求通过有向弧 1 分配，但是向其分配该请求的第一(服务器)节点 21 不改变其活动级别。实际上，起始四个服务器节点 21 到 24 都没有不同的活动级别。第五、获胜服务器节点 25 增加一个单位的活动，而哑节点 20 减少一个单位的活动。

参照图 3b，示出与图 3 的步骤 110 相对应的初始化过程。在步骤

400，初始化 j=1，然后在步骤 405，测试以查看 L_j 是否大于 0。如果是，则在步骤 410 将 i 初始化为 1，并且在步骤 415 测试以查看 a_j 是否大于 0。如果是，则在步骤 420 测试以查看 x_{i0} 是否大于 0。如果是，在步骤 425 创建从哑节点 0 到 j 的有向弧，并且进入步骤 430。如果步骤 415 或步骤 420 失败，则进入步骤 435，其中测试以查看以查看 b_j 是否大于 0。如果是，则在步骤 440 测试以查看 y_{i0} 是否大于 0。如果是，则如前所述，流程进入步骤 425。如果步骤 435 或步骤 440 失败，则控制流至步骤 430，其中，增大 j。在步骤 445，测试以查看 j 是否小于或等于 N。如果否，则控制循环回到步骤 405，否则在步骤 450 退出。

参照图 3c，示出图 3 的步骤 165 的路径簿记过程的流程图。在该流程图中，假设最短路径具有 k 个有向弧，并且标注为从 $j_0=0$ 到 j_1 到 j_2 等等到 $j_k=j$。在步骤 500，初始化 j=0，并且在步骤 505，初始化 i=1。在步骤 510，检查以查看是否 $a_{ij}=a_i(j+1)=1$。如果是，则在步骤 515 检查 x_{ij} 是否大于 0。如果是，则在步骤 520 减小 x_{ij}，并且增大 $x_i(j+1)$。如果步骤 510 或 515 中的任一测试失败，则进入步骤 525，其中，类似地检查以查看是否 $b_{ij}=b_i(j+1)=1$。如果是，则在步骤 530 检查 y_{ij} 是否大于 0。如果是，则在步骤 535 减小 y_{ij}，并且增大 $y_i(j+1)$。如果步骤 525 或 530 中的任一测试失败，则进入步骤 540，其中，增大 i 并且返回到步骤 510(我们知道一个服务器 i 将最终通过具有约束条件的该测试)。现在在步骤 520 或步骤 535 之后，进入步骤 545，其中，增大 j。在步骤 550，测试以查看 j 是否小于 k。如果是，则返回到步骤 505。如果否，则过程在步骤 555 退出。

参照图 3d，示出了图 3 的步骤 170 的图簿记过程的流程图。在步骤 600 开始，将 j_1 初始化成 1。在步骤 605，将 j_2 初始化成 1。在步骤 610，测试以查看关于 L_{j2} 的条件(9)是否为真。如果否，过程在 612 退出，否则在步骤 615 初始化 i=1，然后在步骤 620 测试以查看是否 $a_{j1}=a_{j2}=1$。如果是，则在步骤 625 测试以查看是否 x_{ij1} 大于 0。如果是，则在步骤 630 创建从节点 j_1 到节点 j_2 的有向弧，并且进入步骤 635，其中，增大 j_2。如果步骤 620 或步骤 625 失败，则进入步骤 640，其中测

试以查看是否 $b_{j1}=b_{j2}=1$。如果是，则在步骤 645 测试以查看是否 y_{ij2} 大于 0。如果是，则如前所述进入步骤 630。如果步骤 640 或步骤 645 失败，则进入步骤 635。在步骤 650，测试以查看 j_2 是否小于或等于 N。如果是，在步骤 655 增大 j_1，并且进入步骤 660，其中，测试以查看 j_1 是否小于或等于 N。如果否，则在步骤 665 退出。如果步骤 650 中的测试失败，则进入步骤 670，检查是否 $j_1=j_2$。如果 $j_1=j_2$，则返回到步骤 635；否则返回到步骤 610。如果步骤 660 中的测试失败，则进入步骤 675，其中，还检查是否 $j_1=j_2$。如果 $j_1=j_2$，则返回到步骤 655；否则返回到步骤 605。

动态过程(动态)

参照图 4，现在描述负载均衡过程的动态过程组件。该方法假设(静态过程组件)站点到服务器分配是已知事物。这相当于确定矩阵 A 和 B。然后它监视服务器上的准确或近似负载以及新客户请求到达网络调度器。动态过程组件的工作是对这些排队请求在网络调度器中进行路由决策，将它们分配给适当的服务器。在这样做当中，它试图在可能的程度内实现由目标过程支配的最优平均响应时间和服务器负载级别。实际上，它对在网络调度器中排对的通信调用目标过程以作出其决策。路由决策可以通过准确或概率策略来实现。一旦请求由静态过程组件分配给服务器，就将在那些服务器上满足它们。换句话说，在优选实施例中，网络调度器的决策被认为是最终的，并且服务器本身上的请求不能根据负载来重定向。

定义一些元符号标记将是有用的。在任何给定时间，一些负载将在网络调度器中排队，并且一些负载将已经调度到服务器。我们希望区分这两种负载以及总体负载。当引用与调度器中的负载有关的变量时，将使用单点，并且当引用与服务器中的负载有关的变量时，将使用双点。如果不采用任何点，则变量将表示组合调度器和服务器负载。

该方法监视服务器 j 上的网站 i 的已调度公共请求的当前数目 $\ddot{c}_{i,j}$ 和已调度私有请求的数目 $\ddot{d}_{i,j}$。动态过程组件不对服务器上的请求完成直接作出反应，但是它确实适当地减小 $\ddot{c}_{i,j}$ 和 $\ddot{d}_{i,j}$ 的值。如果不能容易地获

得该数据，则可以对它进行估算。同时，新请求在网络调度器中到达并且排队等候。设$\dot{c}_i$表示自从最后执行动态过程组件以来网站 i 的新公共请求的数目。类似地，设$\dot{d}_i$表示自从最后执行动态过程组件以来网站 i 的新私有请求的数目。通过合计，设$\dot{C}=\sum_{i=1}^{M}\dot{c}_i$和$\dot{D}=\sum_{i=1}^{M}\dot{d}_i$分别表示调度器中的排队公共和私有请求的总数。

动态过程组件可以设计成在固定时间间隔之后、队列中的项数$\dot{C}+\dot{D}$达到某固定批量阈值之后或者以可能是两个标准的某组合唤醒和执行。

对在网络调度器中排队的请求应用目标过程，同时不管先前已调度的请求。为此，必须需要一些簿记。具体而言，对于每个服务器 j，通过设置

$$\dot{R}_j(z)=R_j\left(z+\sum(\ddot{c}_{i,j}+\ddot{d}_{i,j})\right) \tag{13}$$

来定义新的凹增函数$\dot{R}_j(z)$。

该函数简单地变动原始函数以考虑服务器中不能干扰的负载量。由于相同原因，还通过设置

$$\dot{L}_j=L_j-\sum_{i=1}^{M}(\ddot{c}_{i,j}+\ddot{d}_{i,j}) \tag{14}$$

来定义修正容许负载限值$\dot{L}_j$。

通过这些手续，调度$\dot{C}+\dot{D}$个排队请求的最优方式通过求解下面受限目标过程组件问题来确定：在以下约束条件下

$$\text{最小化}\ \sum_{j=1}^{N}\dot{R}_j\left(\sum_{i=1}^{M}(\dot{x}_{i,j}+\dot{y}_{i,j})\right) \tag{15}$$

这些约束条件为：

$$\sum_{i=1}^{M}(\dot{x}_{i,j}+\dot{y}_{i,j})\in\{0,\ldots,\dot{L}_j\}, \tag{16}$$

$$\sum_{j=1}^{N}\dot{x}_{i,j}=\dot{c}_i, \tag{17}$$

$$\text{如果}a_{i,j}=0\text{则}\dot{x}_{i,j}=0, \tag{18}$$

$$\sum_{j=1}^{N}\dot{y}_{i,j}=\dot{d}_i, \tag{19}$$

以及

$$\text{如果}b_{i,j}=0\text{则}\dot{y}_{i,j}=0, \tag{20}$$

这对应于图 4 的步骤 200。即使在最优解中，服务器 j 上的总负载

$\sum_{i=1}^{M}(\dot{x}_{i,j}+\dot{y}_{i,j}+\dot{c}_{i,j}+\dot{d}_{i,j})$在总体上是次优的。然而，在不能在服务器之间转移先前已分配的负载的附加约束条件下，它是最优的。

而且，注意，在每当新请求到达网络调度器时唤醒该方法从而对于某网站 i_1，$\dot{c}_{i_1,j}=1$或者$\dot{d}_{i_1,j}=1$的特殊情况下，该过程简直是贪婪的。换句话说，如果$\dot{c}_{i_1,j}=1$，从而要分配网站 i_1 的新公共请求，则选择其第一差值

$$R_j\left(\sum_{i=1}^{M}(\ddot{c}_{i,j}+\ddot{d}_{i,j}+1)\right)-R_j\left(\sum_{i=1}^{M}(\ddot{c}_{i,j}+\ddot{d}_{i,j})\right) \qquad (21)$$

最小的满足$a_{i_1,j}=1$且$\sum_{i=1}^{M}(\ddot{c}_{i,j}+\ddot{d}_{i,j})<L_j$的那个服务器 j。话说，如果$\dot{d}_{i_1,j}=1$，从而要增加网站 i_1 的新私有请求，则选择由表达式(21)给出的其第一差值最小的满足$b_{i_1,j}=1$和负载限制约束条件的那个服务器 j。

动态过程可以由图 5 和 6 所示的例子阐述。考虑驻留 8 个站点的 15 服务器配置。有 5 个服务器被分配处理公共网站通信。我们将这些服务器称作群集服务器，因为每个这样的服务器都具有处理来自多个站点的通信的能力。还有 10 个服务器被分配处理私有网站通信。因此我们将这些服务器称作分区服务器。每个这样的服务器可以处理公共和私有通信，但是这些通信仅来自单个网站。考虑图 5，它是本例的网站符号汇集表。每个符号具有唯一于网站的形状。例如，**circularlogic.com** 以圆圈表示。灰符号表示公共通信而黑符号表示私有通信。相关群集和分区服务器基数(cardinalities)也被表示。分别存在分配矩阵 A 和 B 的行总和$\sum_{j=1}^{N}A_{i,j}$和$\sum_{j=1}^{N}B_{i,j}$。例如，**circularlogic.com** 分配有 3 个群集服务器和 2 个分区服务器。

所有这些更详细地见于图 6。内圈包含 5 个群集服务器 C1 到 C5，并且外圈包含 10 个分区服务器 P1 到 P10。例如，注意，**circularlogic.com** 分配给群集服务器 C1、C2 和 C5。在服务器 C1 上，该网站的公共通信必须与 **hexnut.com** 和 **eggcentric.com** 的公共通信共享该服务器。类似地，**circularlogic.com** 分配给分区服务器 P1 和 P2。这两个服务器中的每一个均处理公共和私有通信，但是这些通信仅对应于该网站。

实际上，图 6 传达更多信息。它示出由目标过程使用的种类的略加

修改的有向图。回想图 2，该图示出了在求解类约束资源分配问题中在特定时刻从哑节点到服务器节点的最短路径。作为对比，图 6 示出有向图 H，该图在一个例外的情况下示出该时刻的所有有向弧和节点(例外是消除了哑节点和以其为起点的有向弧)。

从而 H 如下定义：节点对应于服务器。对于不同服务器节点的每个对 j_1 和 j_2，假如存在满足条件(7)到(9)或者条件(10)、(11)和(9)的至少一个网站 i_1，则存在从 j_1 和 j_2 的有向弧。

在图 6 中，不是所有有向弧都出现在两个方向上，因为它们可能不能满足所有相关约束条件。例如，不存在从 P3 到 P4 的有向弧，显然地这是因为在 P3 上不存在任何负载，公共负载(条件 8)或私有负载(条件 11)。不存在从 C3 到 P7 的有向弧，显然地这是因为 P7 以全部容量工作(条件 9)。在 C1 与 P3 之间的任一方向上都不存在弧，显然地这是因为 P3 上没有公共负载(条件 8)并且 C1 以全部容量工作。在 P1 与 C4 之间的任一方向上都不存在弧，这是因为既不满足条件(7)又不满足条件(10)。

再次，有向弧的存在表示减小服务器上的负载、在不超过负载容量的情况下增大另一个服务器上的负载，并且使其他服务器上的负载不受影响的潜在可能性。从哑节点到服务器节点之一的有向路径允许将一单位的负载从集结节点(staging node)转移到那个服务器节点，而使所有其他服务器节点不受影响。在该转移之后，图 H 可以通过簿记方法来修改以考虑约束条件(8)、(9)和(11)的当前状态。(约束条件(7)和(10)的状态保持不变)。

在图 6 中可以看到三种不同类型的有向弧。表示公共或私有负载从一个分区服务器到另一个分区服务器的潜在转移的一种类型可以在 P1 与 P2 之间看到。表示公共负载从分区服务器到群集服务器或者相反的潜在转移的第二类型可以在 P1 与 C1 之间看到。表示公共负载从一个群集服务器到另一个群集服务器的潜在转移的第三类型可以在 C1 与 C2 之间看到。

应该理解，按照具有尽可能多的有向弧，图 H 越密，则目标过程越

有可能可以实现产生较低响应时间的良好质量结果。这是下一步描述的静态过程组件的目标。

动态过程的详细实施例在图 4 的流程图中示出。控制流程以步骤 200 开始，采用目标过程来对于每个网站 i 和服务器 j 求解最优目标$\dot{x}_{i,j}$和$\dot{y}_{i,j}$。在步骤 205，设置 k=1。在步骤 210，对于每个网站 i 和服务器 j，将$\dot{X}_{i,j}$和$\dot{Y}_{i,j}$初始化为 0。这些变量将对相对于目标的调度器进度进行计数。在步骤 215，确定什么网站对应于队列中的请求 k。步骤 220 判定请求是公共的还是私有的。对于这两种可能性的每一种，存在对等的流程，并且我们集中于公共请求情况。这开始于步骤 225，其中，设置 j=1 以及 M 为负无穷大。将试图如下查找在目标和所实现值之间具有最大差值的服务器：在步骤 230，判断 M 是否小于$\dot{x}_{i,j}-\dot{X}_{i,j}$。如果是，在步骤 235 设置 j*=j 以及 M=$\dot{x}_{i,j}-\dot{X}_{i,j}$，并且进入步骤 240。如果步骤 230 中的测试失败，则也分支至步骤 240。在步骤 240，增大 j。在步骤 245，判断 j 是否小于或等于 N。如果是，则返回到步骤 230。如果否，则在步骤 250 得到获胜服务器 j*，增大$\dot{X}_{i,j}$*，并且将请求 k 调度到服务器 j*。然后进入步骤 255，其中，增大 k。在步骤 260，测试 k 是否小于或等于$\dot{C}+\dot{D}$。如果是，则返回到步骤 210 以调度下一请求。如果否，则由于清空了请求队列而停止。如果步骤 220 判定请求是私有请求，则进入步骤 270 到 295。这些步骤除了处理私有请求而不是公共请求之外，完全类似于步骤 225 到 250。最终，控制流程再次返回到步骤 220。

静态过程组件

参照图 7 至 9，现在描述将站点分配给服务器的静态过程组件。目标是优化目标过程的可实现性能，并且因此优化动态过程组件的可实现性能。静态过程组件的关键输入将是每个网站的公共和私有请求的平均需求预测以及容许网站到服务器分配的任何约束条件。静态过程组件的输出将简单地是两个{0, 1}分配矩阵 A=($a_{i,j}$)和 B=($b_{i,j}$)。

还假设网站到服务器分配的约束条件是已知事物。这些约束条件可能与服务器本身的物理容量、实现可接受高缓存命中率的目标、服务器

上的操作系统、某些站点可能协商过的固定服务器分配等等有关。这些约束条件可能相当复杂或相当基本。例如，缓存约束条件可能根据分析命中率模型、网站数据的局部性和缓存大小而可能相对复杂。它将具有确保较小数目的站点分配给每个服务器的效果。另一方面，操作系统约束条件可能简单地是哪些站点不适于某些服务器的列表。要点是静态过程组件在性质上是试探性的(heuristic)，并且可以简单地通过在考虑任何分配变化之前检查约束条件来处理它们。

静态过程组件具有两种可能模式。初始模式(图 9)用来从零开始配置新系统，在这种模式下，尚未有节点分配给服务器。增量模式则周期性地用来根据修正网站需求预测来调整现有网站到服务器分配。为了确保这些调整的实现是实用的，考虑限制所允许的容许网站到服务器分配变化数的约束条件。可以看出，两种模式都将采用本质上相同的方法。在初始模式下，可能必须采用响应时间函数的分析模型，而在增量模式下，作为代替可能使用更准确的测量响应时间。这大概将更好地捕获缓存和其他真实世界现象的效应。

增量静态过程组件不应执行得太频繁，因为进行分配变化存在较高的成本。每星期大约一次可能是合理的。准确的频率将依赖于该成本与在给定更新网站预测的情况下性能改进的潜在可能性之间的相对折衷。大概，静态过程的运行可以通过检测某种负载失衡条件来触发。

两个模式下的主要目标都是实现如下定义的无向图 G 的高连接性：节点对应于服务器。对于不同节点的每个对 j_1 和 j_2，假如存在对于其 $a_{i_1,j_1}=a_{i_1,j_2}=1$ 的至少一个网站 i_1，或者对于其 $b_{i_1,j_1}=b_{i_1,j_2}=1$ 的至少一个网站 i_1，则在 j_1 与 j_2 之间存在一条弧。这些条件相似于有向图 H 的定义中的条件(7)和(10)。由于在任何均衡良好的网络场中一般都将满足条件(8)、(9)和(11)，因此认为 G 用作 H 的有效替代物。还注意，条件(7)和(10)在本质上是静态的，而条件(8)、(9)和(11)在本质上是动态的。这是为什么 G 更适于静态过程组件的原因。

图 7 示出上述例子的图 G。注意该图类似于图 6 所示的有向图 H。然而，在服务器 C1 与 P3 之间，在图 7 中存在一条弧，而图 6 在任何

方向上都没有对等的有向弧。并且图 6 中的若干其他有向弧如从服务器 P4 到 P3 的有向弧没有相反方向的有向弧与之匹配。注意，在图 7 中，两个群集服务器之间的弧以实线表示，两个分区服务器之间的弧以点虚线表示，而群集服务器与分区服务器之间的弧以短划虚线表示。

将要试图优化的目标函数是在本发明中视作黑盒的目标过程组件本身的目标函数。在此的基本原理是该量度标准即有效平均响应时间显然比其他两个图论替代物(直径或平均距离)更直接适合，并且在计算上也没有显著更高的计算成本。

所采用的元标记法是代字号(~)将引用静态过程问题公式表述。

将通过如下创建将要描述(在某种意义上是优选的)站点到服务器分配的空间的新矩阵 $\tilde{A}$ 来进行：$\tilde{A}=(\tilde{a}_{i,j})$ 将再次为大小 M×N 的{0, 1}矩阵。在每个网站都存在公共和私有通信的典型条件下，将以下面方式将 $\tilde{A}$ 转换成 $A_{\tilde{A}}$ 和 $B_{\tilde{A}}$。

$$a_{i,j}=\tilde{a}_{i,j} \tag{22}$$

$$b_{i,j}=\begin{cases}1, & \text{如果}\tilde{a}_{i,j}=1,\text{并且对于所有}i'\neq i,\tilde{a}_{i',j}=0\\ 0, & \text{其他情况}\end{cases} \tag{23}$$

这作出以下隐式假设：对于其行总和 $\sum_{i=1}^{M}\tilde{a}_{i,j}=1$ 的任何服务器 j 是具有一个网站的公共和私有通信的分区服务器，并且所有其他服务器是具有多个公共网站通信的服务器。注意，不是分配矩阵 A 和 B 的所有可能对都能被认为是从某矩阵 $\tilde{A}$ 转换出的。然而，不能从某矩阵 $\tilde{A}$ 转换出的矩阵 A 和 B 对是较不可取的。静态过程组件将不选择它们。

必须进行微小的调整来处理特殊情况。如果对于网站 i 只存在私有通信，则将要求每当 $\tilde{a}_{i,j}=1$ 时，对于 $i'\neq i$，$\tilde{a}_{i',j}=0$。这使得 j 是网站 i 的分区服务器。设置 $a_{i,j}=0$ 和 $b_{i,j}=1$。如果对于网站 i 只存在公共通信，则即使 $\sum_{i=1}^{M}\tilde{a}_{i,j}=1$，分配给那个网站的任何服务器 j 也都可以正式认为是公开的。从而设置 $a_{i,j}=\tilde{a}_{i,j}$ 和 $b_{i,j}=0$。

还将需要修改每个服务器 j 的响应时间函数 R_j，从而处理那个服务器上的负载超过其负载限值 L_j 的情况。该不可行性至少在静态过程组件过程的早期阶段可能发生，并且需要通过以大的目标函数成本强制可

行性来处理它们。从而考虑非常大的数 Q，并且设对于 $k \le L_j$，$\tilde{R}_j(k) = R_j(k)$，但是 $\tilde{R}_j(L_j+1) = \tilde{R}_j(L_j) + Q$，$\tilde{R}_j(L_j+2) = \tilde{R}_j(L_j+1) + 2Q$ 等等。对于 k>0，一般递归式定义

$$\tilde{R}_j(L_j+k) = \tilde{R}_j(L_j+k-1) + kQ \qquad (24)$$

注意，$\tilde{R}_j$ 保持凸特性和递增特性。还将最大容许负载重新定义为 $\tilde{L}_j = \infty$。

对于网站 i，设 $\tilde{c}_i$ 和 $\tilde{d}_i$ 分别表示预测公共和私有负载。将基于对于每个网站都存在公共和私有通信的情况来阐述，但是在其他情况下，本领域的技术人员可以很快地确定适当的修改。

在不损失一般性的情况下，假设站点按照增大的预测总负载 $\tilde{c}_i + \tilde{d}_i$ 来重索引。还将假设服务器按照增高的性能来重索引。例如，这可以通过它们的原始最大容许负载 L_j 来确定。

初始分配方法的第一步骤是在可能的程度内顺序匹配站点与服务器。可能存在从某些服务器限制某些站点的可行性约束条件，并且必须考虑这些约束条件。从而，如图 9 的步骤 305 到 335 所示依次从网站 i=1 到 M 循环，在步骤 325，设置 $\tilde{a}_{i,j_i} = 1$，其中，j 是可行但当前未分配服务器的索引，并且在初始化时，对于所有其他 j，设置 $\tilde{a}_{i,j} = 0$。有可能但大概又不太可能的是，对于所有站点都将不存在可能的可行分配。这可能由于约束条件而发生，并且它可能由于 M>N 而发生。在这种情况下，初始分配方法已失败。否则，此刻存在 M 个分区服务器，并且其余 N-M 个服务器空闲。

评估该初始解的目标函数，该函数恰好为

$$\sum_{i=1}^{M} \tilde{R}_{j_i}(\tilde{c}_i + \tilde{d}_i) \qquad (25)$$

初始解现在将通过实现所谓的邻域扩展试探法来修改。该试探法采用度量 Δ 来测量两个可能的站点到服务器分配之间的距离，马上将对此进行描述。简要地，邻域扩展试探法是试图避免陷入局部极小值同时实现较低计算成本的迭代改进过程。假定可行解搜索空间中目前的预定义度量 Δ 以及如上所述的现有初始可行解，该方法分阶段进行。在每个阶段的开头，存在可以在该阶段内修改的所谓当前解。在第一阶段的开

头，当前解是初始解。每个阶段连续搜索当前解周围距离 1、2 等的邻域(当前解周围距离 k 的邻域是与当前解的距离小于或等于 k 的所有可行分配集。当前解周围距离 k 的环是与当前解的距离等于 k 的所有可行分配集，它是距离 k 的邻域减去距离 k-1 的邻域)。如果在当前解周围距离 1 的邻域内可以发现目标函数改进，则该试探法选择最佳该改进，将它重新标注为当前解，并且通过开始下一阶段来迭代过程。如果在距离 1 的邻域内不能发现改进，则该试探法改为考虑距离 2 的环。再次存在两种可能性：如果在此存在改进，则选择最佳该改进作为当前解，并且过程开始下一阶段。如果否，则过程考虑距离 3 的环等等，直到固定距离限值，比方说 K。如果在第 K 环内仍未发现任何目标函数改进，则该过程以等于当前解的最终解终止。各环的检查发生于图 9 的步骤 360、380 和 385。在步骤 370，检查改进。

图 8 示出该过程。初始阶段开头的当前解位于第一邻域集即距离 1 的中心 400。该试探法顺序搜索第一邻域和随后环，直到发现改进。在本图中，改进仅发生于第 K 环即距离 K。然后，该试探法将当前解重新设置为在由新中心 410 表示的在该搜索中所发现的最佳改进，并且开始第二阶段。本图示出该解的邻域。此时在第三环内发现改进。此时开始第三阶段，但是在本图中未示出。如果在第一阶段第 k 环都不能产生改进，则初始解将为最终解。

现在定义距离量度 Δ。设 $\tilde{A}^1=(\tilde{a}^1_{i,j})$ 和 $\tilde{A}^2=(\tilde{a}^2_{i,j})$ 是两个该分配，并且定义

$$\Delta(\tilde{A}^1,\tilde{A}^2)=\sum_{i=1}^{M}\sum_{j=1}^{N}\left|\tilde{a}^1_{i,j}-\tilde{a}^2_{i,j}\right| \qquad (26)$$

在此的意图是简单的：如果例如通过将单个网站加到服务器之一或者从中将其减去来将分配 $\tilde{A}^1$ 修改成新分配 $\tilde{A}^2$，则获得 $\Delta(\tilde{A}^1,\tilde{A}^2)=1$。将网站从旧服务器移到新服务器(先前未分配有该网站)在图 8 中具有距离 2。从服务器减去网站同时将不同网站加到那个服务器也具有距离 2。交换不同服务器上的不同站点具有距离 4 等等。

对于给定解 $\tilde{A}$ 及其转换分配矩阵 $A_{\tilde{A}}$ 和 $B_{\tilde{A}}$，则改进测试涉及求解在以下约束条件下最小化

$$\sum_{j=1}^{N}\tilde{R}_j\left(\sum_{i=1}^{M}(\tilde{x}_{i,j}+\tilde{y}_{i,j})\right) \quad (27)$$

的目标过程组件问题，其中这些约束条件为：

$$\sum_{j=1}^{N}\tilde{x}_{i,j}=\tilde{c}_i \quad (28)$$

如果$a_{i,j}$=0则$\tilde{x}_{i,j}$=0 (29)

$$\sum_{j=1}^{N}\tilde{y}_{i,j}=\tilde{d}_i \quad (30)$$

以及

如果$b_{i,j}$=0则$\tilde{y}_{i,j}$=0 (31)

当然，涉及$\tilde{L}_j=\infty$的约束条件是无意义的。但是在给定超过限值 L_j 的负载的成本的情况下，将预期在邻域扩展试探法中在非常少的阶段之后获得对于每个服务器 j 满足

$$\sum_{i=1}^{M}(\tilde{x}_{i,j}+\tilde{y}_{i,j})\in\{0,...,L_j\} \quad (32)$$

的解。从而从所有角度都可行的分配应该可以迅速地实现。实际上，该过程的前几个阶段将典型地将高量站点分配给当前未用的服务器，然后该方法将开始将分区服务器变成群集服务器，典型地首先改变较高性能服务器的状态。

增量过程将周期性地运行，可能是每星期一次，以在存在变化预测等的情况下保持良好的网站到服务器分配。幸运的是，它使用实质上相同的方法。

当然，初始解是来自最后一次静态过程组件运行的当前解，并且通过上述邻域扩展法来修改它。仅有的改变是采用如果初始解和当前解之间的距离超过某用户定义的阈值 T 则停止的附加停止标准。这样的原因是实现分配改变实在是昂贵的，并且不希望轻率地允许静态过程组件运行的连续实例之间的不同解。

参照图 9，示出静态过程的详细实施例。在步骤 300 开始，将网站 i 初始化为 1。在步骤 305，将服务器 j 初始化为 1。在步骤 310，判定服务器 j 是否适于网站 i。如果是，则进入步骤 320，其中，判定服务器 j 是否是当前未分配的。如果它是当前未分配的，则在步骤 325 分配它，

并且在步骤 330 增大网站 i。在步骤 335，判定 i 是否小于或等于站点的总数 M。如果是，则返回到步骤 305。如果步骤 310 或 320 中的测试失败，则进入步骤 340，其中，增大服务器 j。在步骤 345，判定 j 是否小于或等于服务器的总数 N。如果是，则返回到步骤 310。如果否，则在步骤 350 以非可行解停止。现在返回到在步骤 335 站点已被用尽的情况，进入步骤 355，开始该过程的邻域扩展组件。估计由方程(25)给出的该解的值 R。在步骤 360，设置 k=1。在步骤 365，搜索第 k 环以估计最佳可能解 R_k。在步骤 370，与最佳当前解 R 进行比较来评估 R_k。如果新值更好，则在步骤 375 使其为最佳解，并且返回到步骤 360。如果新值不更好，则在步骤 380 增大 k。在步骤 385，判定所有环 k 是否已被用尽。如果它们尚未用尽，则返回到步骤 365。否则在步骤 390 以最终分配解停止。

本发明可以容易地配置成例如通过在可由能够执行指令的机器如计算机读取的存储介质上有形地实施指令程序来自动化。

还应该理解，术语“网站”无需将本发明限定于在万维网上与超文本标记语言(HTML)文档一起使用，而是应解释为包括能够接收客户请求、输出结果并且与例如局域网等上的服务器通信的任何用户网络接口。

应该理解，除非明确表示，否则在此所公开的所有物理量不应解释为准确地等于所公开的量，而是大约等于所公开的量。此外，仅仅缺少诸如“大约”等的限定词不应解释为明确表示任何这样公开的物理量是准确量，而不管是否对于在此公开的任何其他物理量使用了这样的限定词。

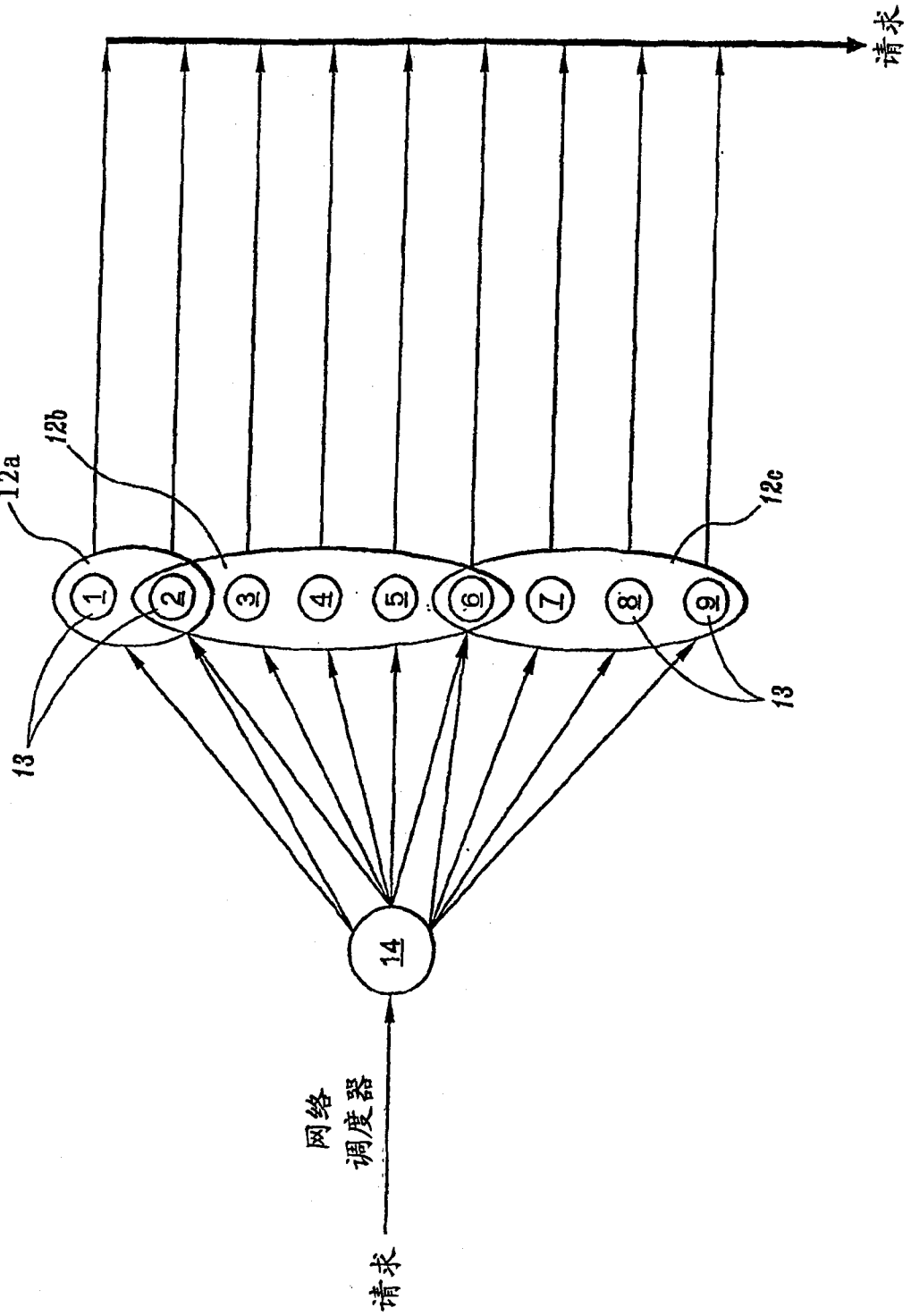


图1

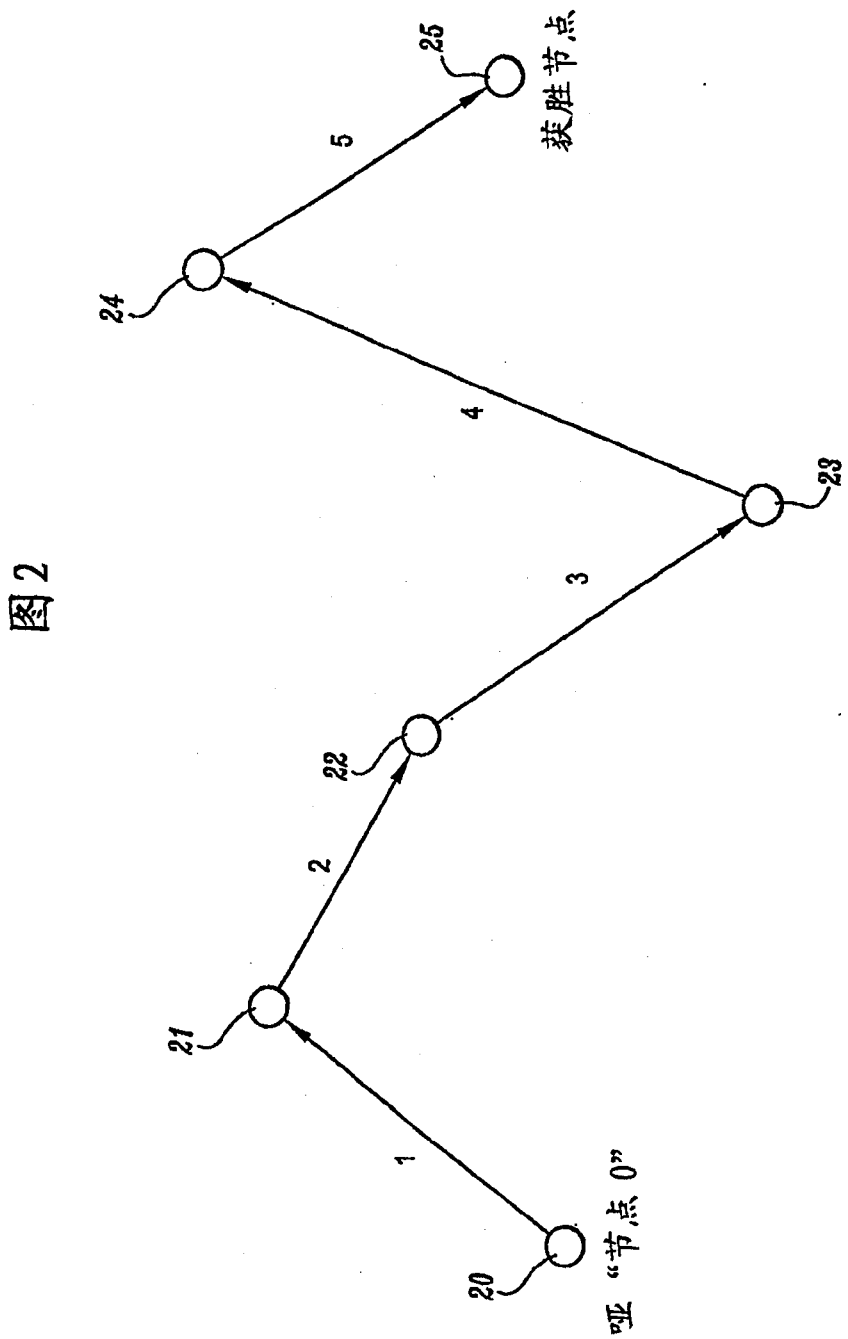


图 2

图 3a

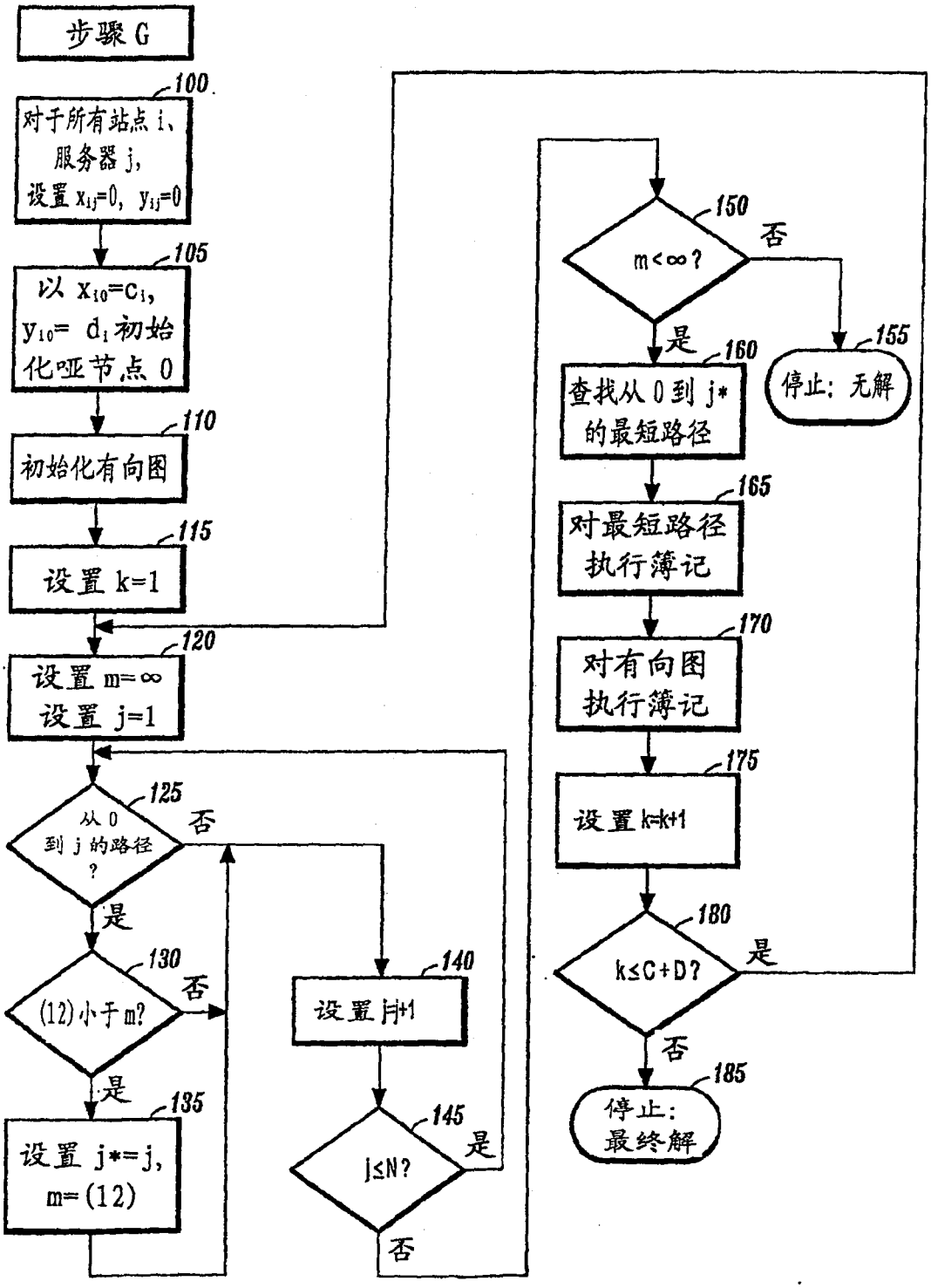

步骤 G
100
对于所有站点 i、服务器 j，设置 $x_{ij}=0$, $y_{ij}=0$
105
以 $x_{i0}=c_i$, $y_{i0}=d_i$ 初始化哑节点 0
110
初始化有向图
115
设置 k=1
120
设置 m=∞ 设置 j=1
125
从 0 到 j 的路径？
否
是
130
(12)小于 m?
否
是
135
设置 j*=j, m=(12)
140
设置 j=j+1
145
j≤N?
是
否
150
m<∞?
否
是
155
停止：无解
160
查找从 0 到 j* 的最短路径
165
对最短路径执行簿记
170
对有向图执行簿记
175
设置 k=k+1
180
k≤C+D?
是
否
185
停止：最终解

图 3b

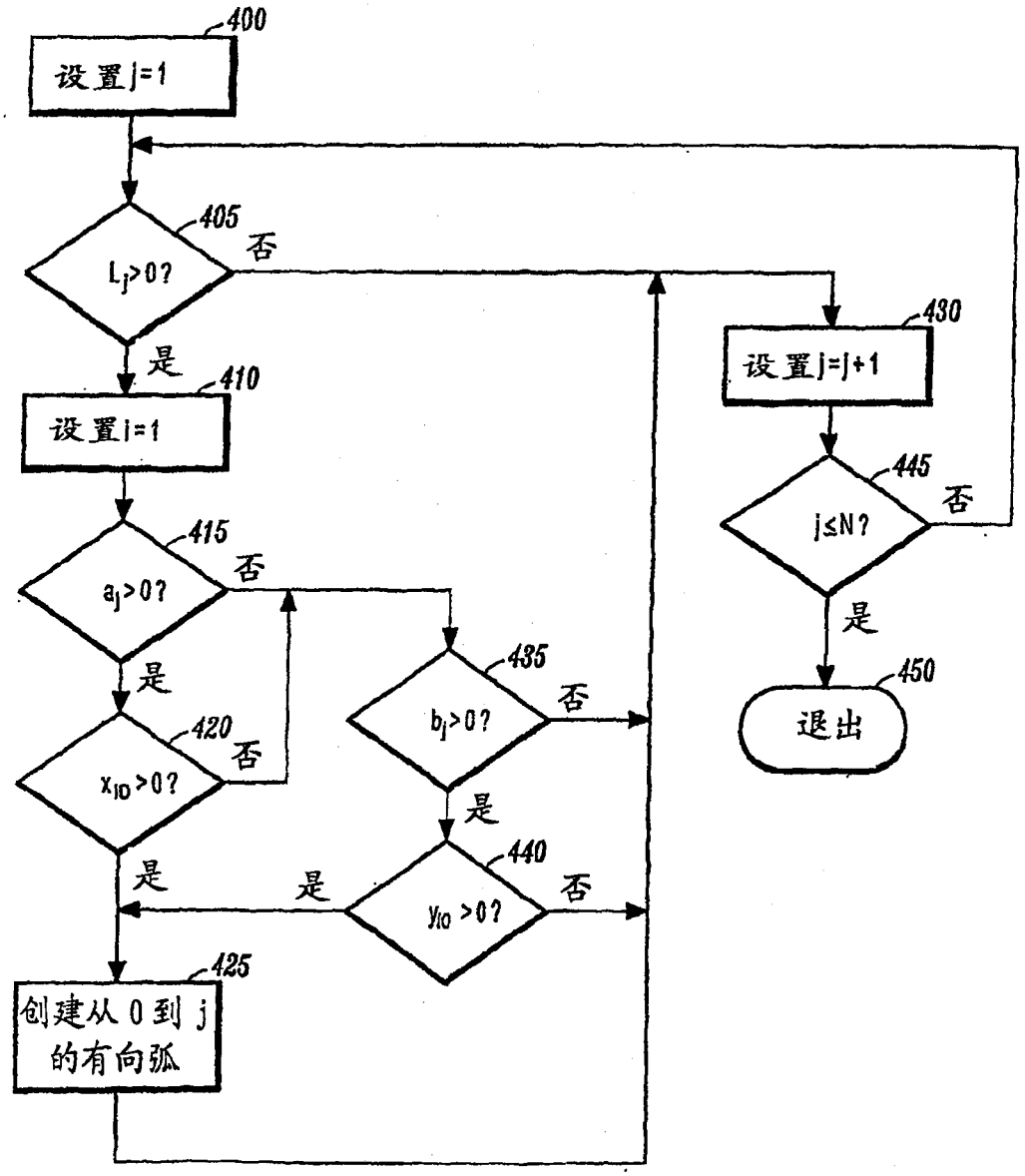

400
设置j=1
405
L_j>0?
否
是
410
设置i=1
415
a_j>0?
否
是
420
x_{i0}>0?
否
是
425
创建从 0 到 j 的有向弧
435
b_j>0?
否
是
440
y_{i0}>0?
否
是
430
设置j=j+1
445
j≤N?
否
是
450
退出

图 3c

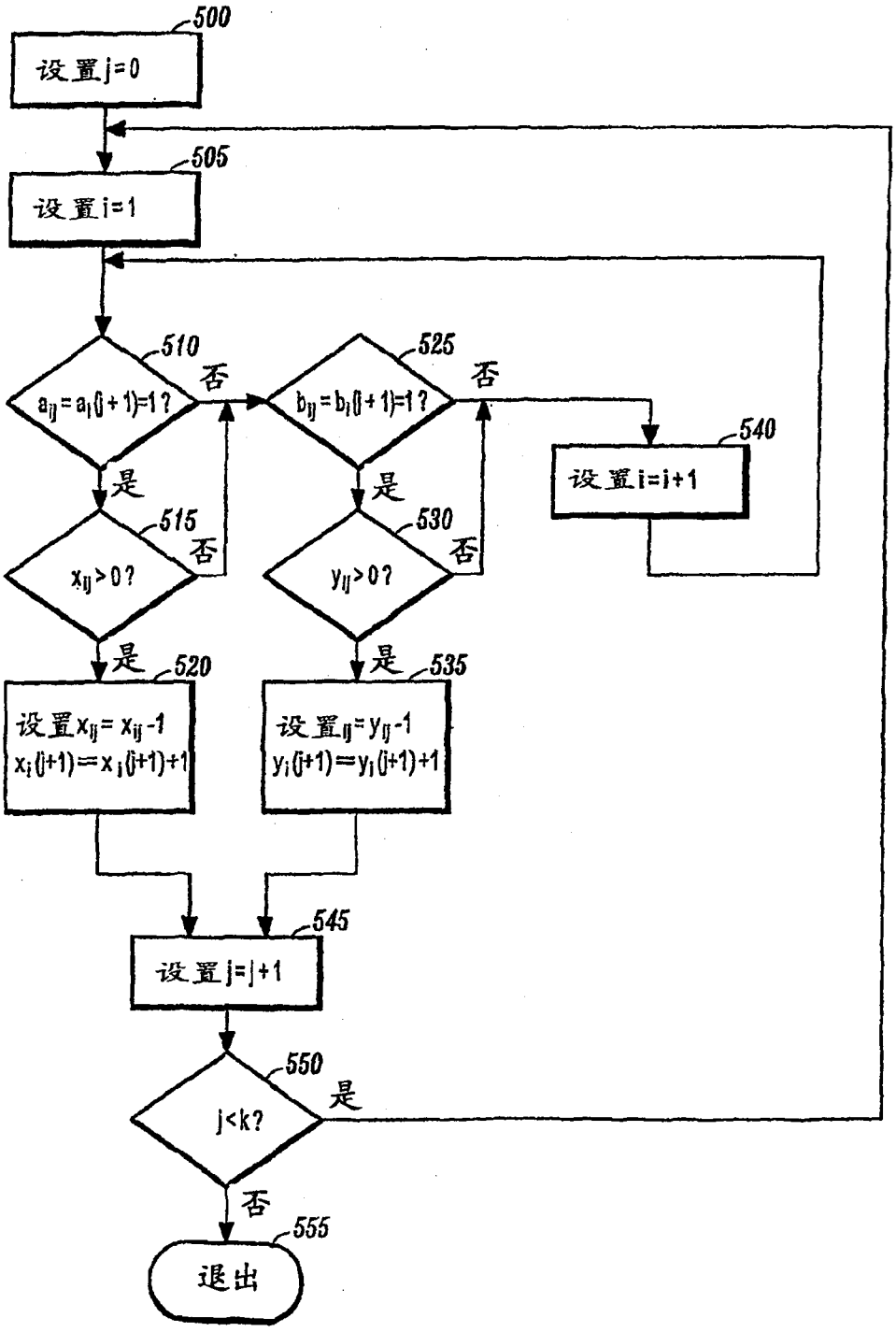

500
设置j=0
505
设置i=1
510
$a_{ij}=a_i(j+1)=1?$
否
525
$b_{ij}=b_i(j+1)=1?$
否
540
设置i=i+1
是
是
515
$x_{ij}>0?$
否
530
$y_{ij}>0?$
否
是
是
520
设置$x_{ij}=x_{ij}-1$
$x_i(j+1)=x_i(j+1)+1$
535
设置$y_{ij}=y_{ij}-1$
$y_i(j+1)=y_i(j+1)+1$
545
设置j=j+1
550
j<k?
是
否
555
退出

图 3d

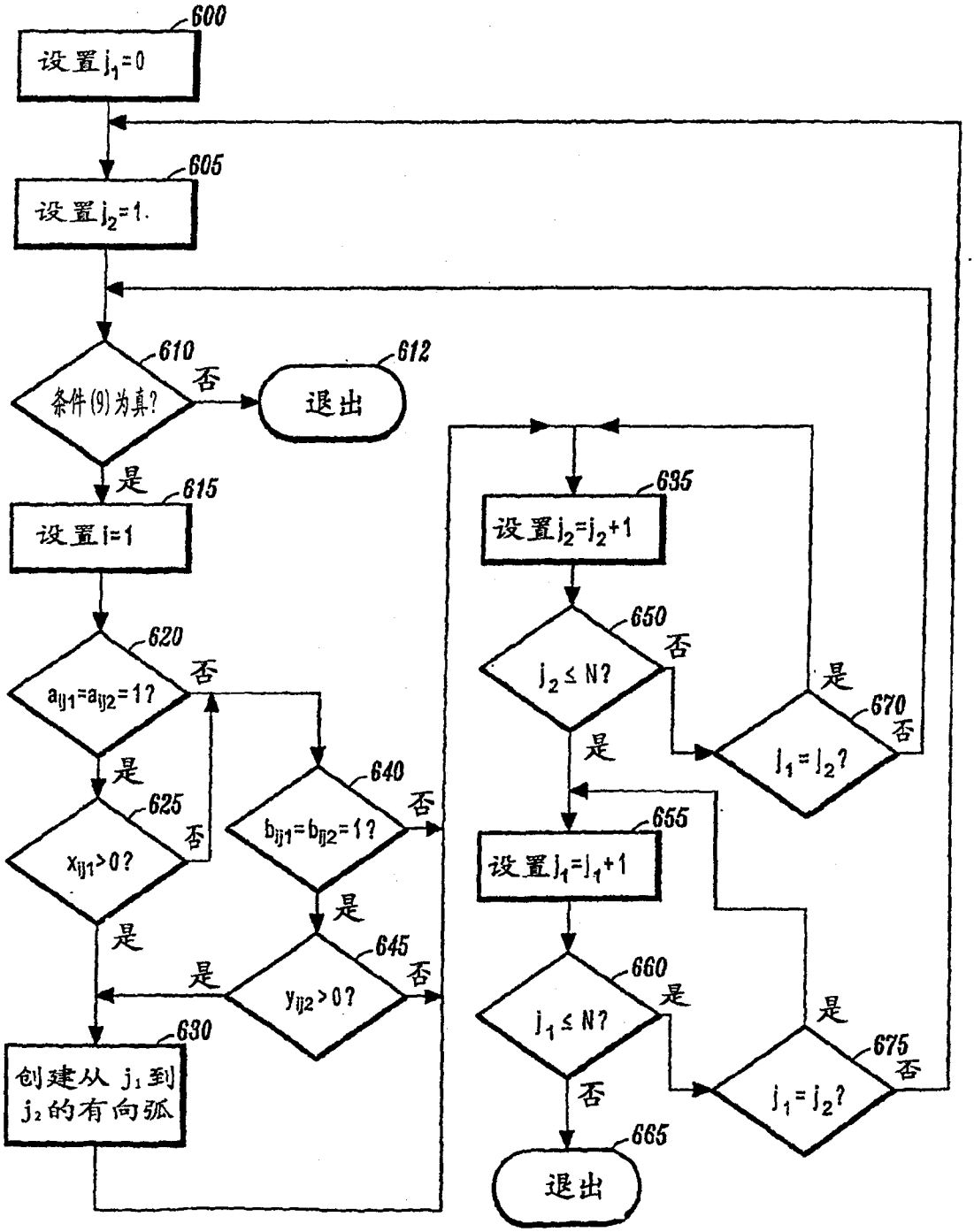

600
设置 $j_1=0$
605
设置 $j_2=1$
610
条件(9)为真?
否
612
退出
是
615
设置 i=1
620
$a_{ij1}=a_{ij2}=1$?
否
是
625
$x_{ij1}>0$?
否
是
640
$b_{ij1}=b_{ij2}=1$?
否
是
645
$y_{ij2}>0$?
否
是
630
创建从 j_1 到 j_2 的有向弧
635
设置 $j_2=j_2+1$
650
$j_2 \le N$?
否
是
670
$j_1=j_2$?
否
是
655
设置 $j_1=j_1+1$
660
$j_1 \le N$?
是
否
675
$j_1=j_2$?
否
是
665
退出

图4

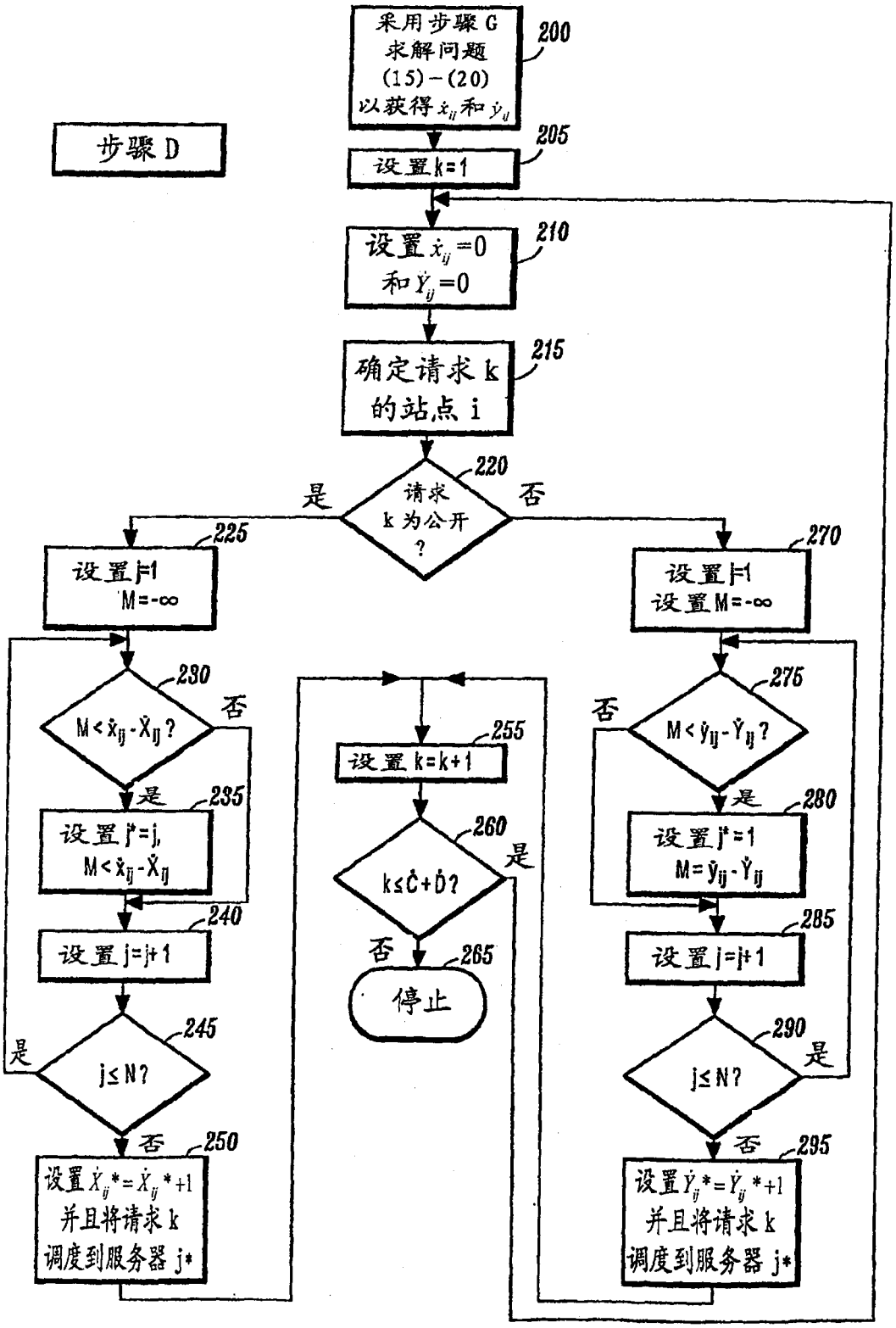
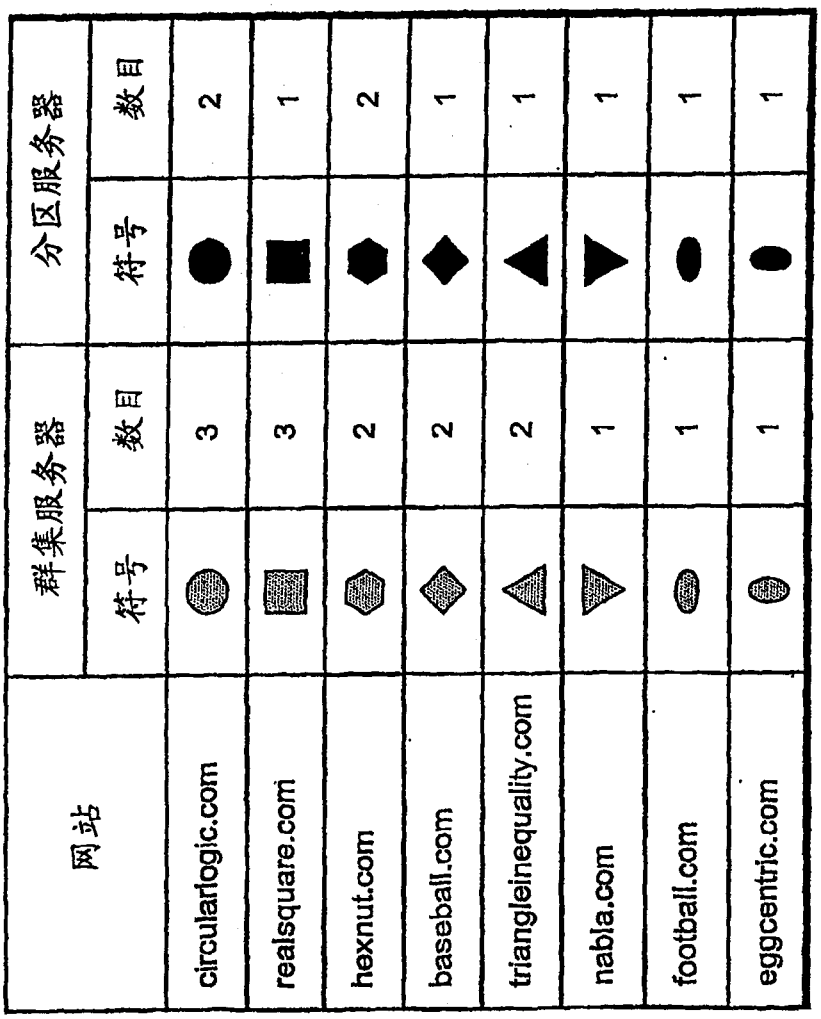

步骤 D
采用步骤 G 求解问题 (15)-(20) 以获得 $\dot{x}_{ij}$ 和 $\dot{y}_{ij}$
200
设置 k=1
205
设置 $\dot{x}_{ij}=0$ 和 $\dot{Y}_{ij}=0$
210
确定请求 k 的站点 i
215
请求 k 为公开 ?
220
是
否
设置 j=1 M=-∞
225
M< $\dot{x}_{ij}$-$\dot{X}_{ij}$?
230
是
否
设置 j*=j, M< $\dot{x}_{ij}$-$\dot{X}_{ij}$
235
设置 j=j+1
240
j≤N?
245
是
否
设置 $\dot{X}_{ij}$*=$\dot{X}_{ij}$*+1 并且将请求 k 调度到服务器 j*
250
设置 k=k+1
255
k≤Ċ+Ḋ?
260
是
否
停止
265
设置 j=1 设置 M=-∞
270
M< $\dot{y}_{ij}$-$\dot{Y}_{ij}$?
275
否
是
设置 j*=1 M= $\dot{y}_{ij}$-$\dot{Y}_{ij}$
280
设置 j=j+1
285
j≤N?
290
是
否
设置 $\dot{Y}_{ij}$*=$\dot{Y}_{ij}$*+1 并且将请求 k 调度到服务器 j*
295

图5



网站	群集服务器		分区服务器	
	符号	数目	符号	数目
circularlogic.com	●	3	●	2
realsquare.com	■	3	■	1
hexnut.com	⬢	2	⬢	2
baseball.com	◆	2	◆	1
triangleinequality.com	▲	2	▲	1
nabla.com	▼	1	▼	1
football.com	⬮	1	⬮	1
eggcentric.com	⬮	1	⬮	1

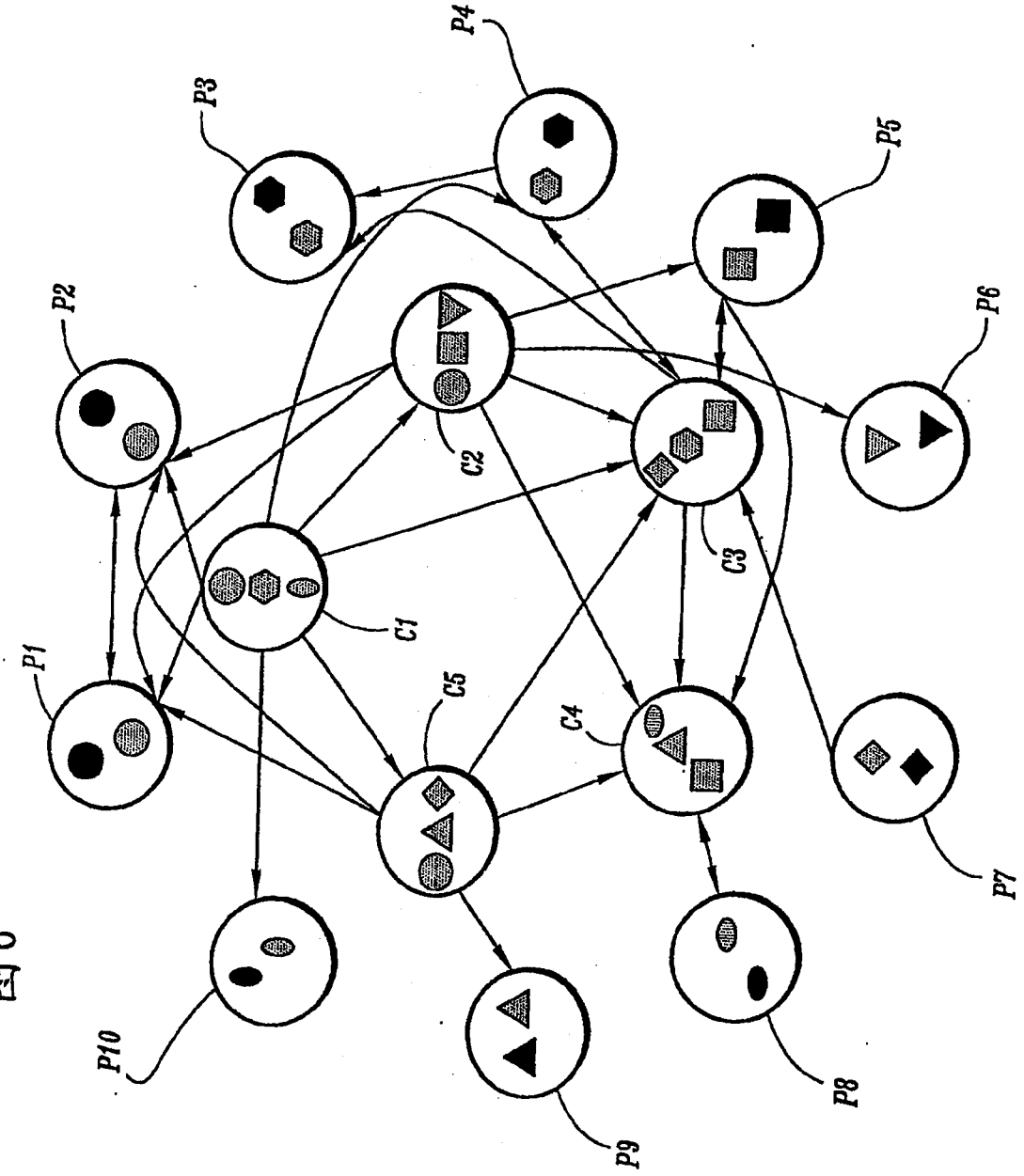


图6

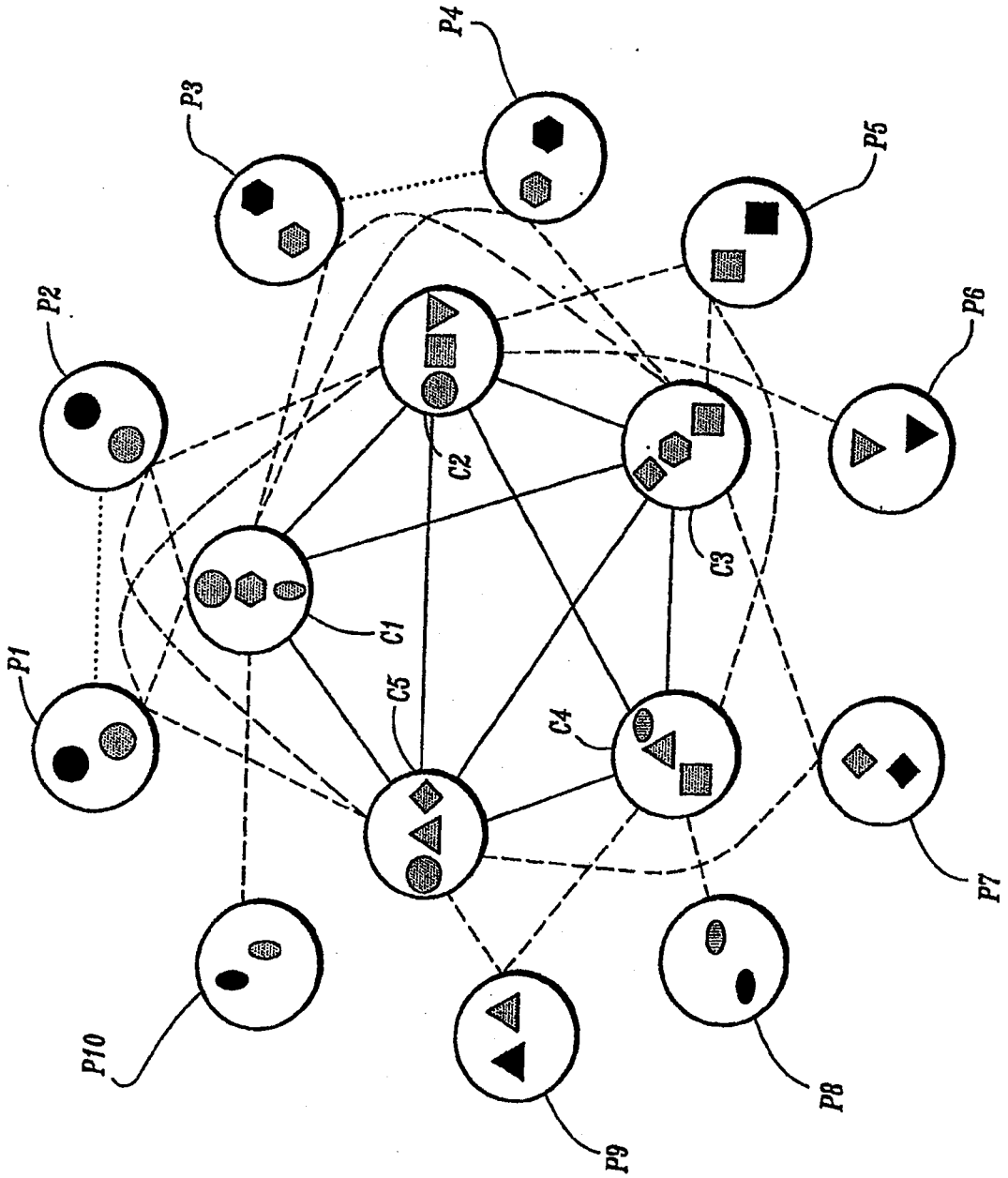


图 7

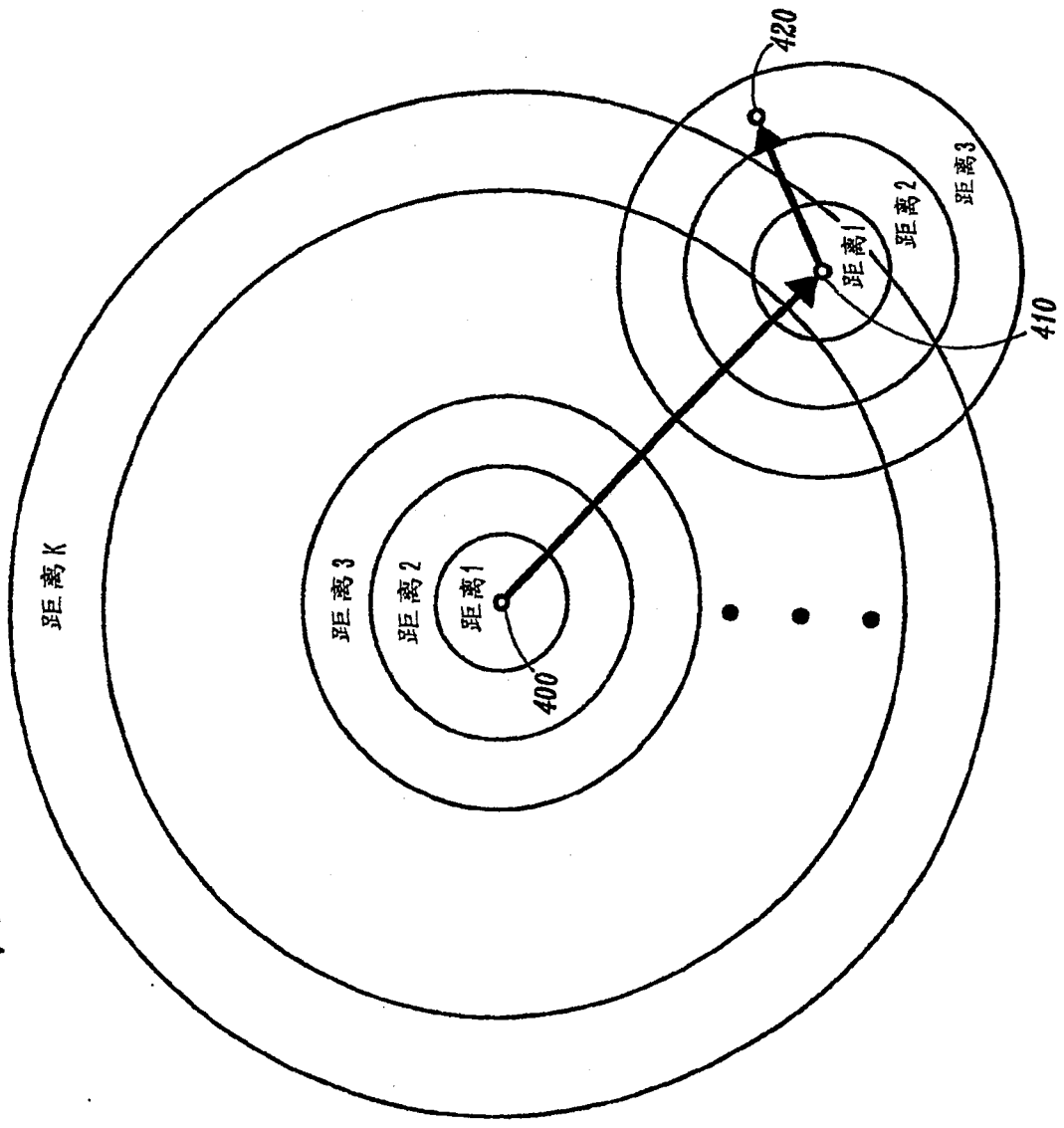
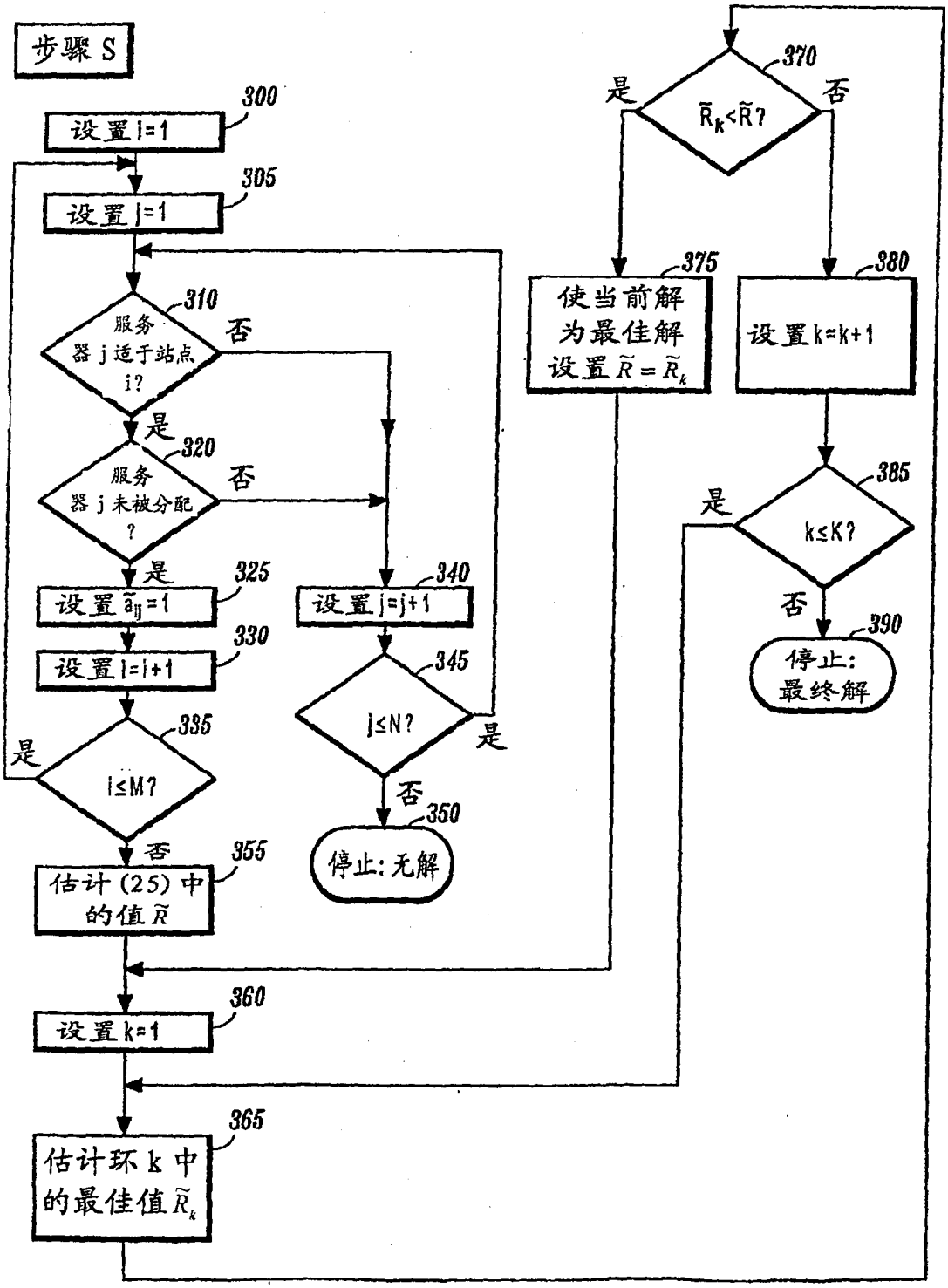


图8

图9


步骤 S
300
设置 i=1
305
设置 j=1
310
服务器 j 适于站点 i?
否
是
320
服务器 j 未被分配?
否
是
325
设置 $\tilde{a}_{ij}=1$
330
设置 i=i+1
335
i≤M?
是
否
355
估计 (25) 中的值 $\tilde{R}$
340
设置 j=j+1
345
j≤N?
是
否
350
停止: 无解
360
设置 k=1
365
估计环 k 中的最佳值 $\tilde{R}_k$
370
$\tilde{R}_k<\tilde{R}$?
是
否
375
使当前解为最佳解 设置 $\tilde{R}=\tilde{R}_k$
380
设置 k=k+1
385
k≤K?
是
否
390
停止: 最终解