(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 110347180 A
(43)申请公布日 2019.10.18

(21)申请号 201910738940.3

(22)申请日 2019.08.12

(71)申请人 南京邮电大学
地址 210012 江苏省南京市雨花台区软件大道186号

(72)发明人 张迎周 黄秋月 傅建清 陈宏建 肖雁冰

(74)专利代理机构 南京苏科专利代理有限责任公司 32102
代理人 姚姣阳

(51)Int.Cl.
G05D 1/10(2006.01)
G06F 17/15(2006.01)

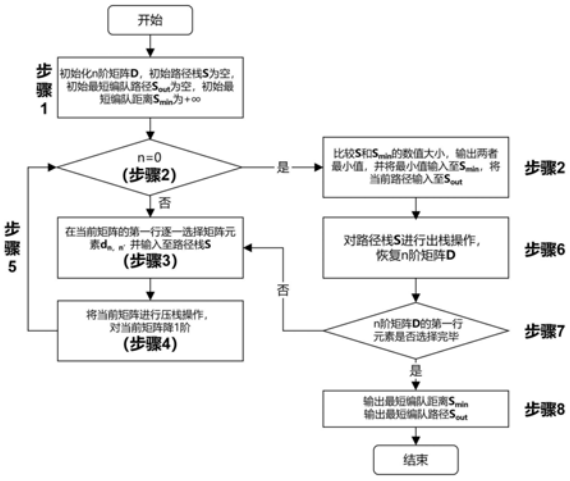
权利要求书2页 说明书4页 附图4页

(54)发明名称

计算无人机集群重新编队的最短编队距离的方法

(57)摘要

本发明提供了一种计算无人机集群重新编队的最短编队距离的方法，主要包括以下步骤：步骤S1、设计递归算法，并对该递归算法进行预处理，以找出递归代码部分；步骤S2、将递归算法中的递归代码部分利用CPS技术转化为尾递归形式；步骤S3、利用Trampoline技术对尾递归形式进行优化，以获得尾递归算法；步骤S4、利用优化后的尾递归算法计算无人机集群重新编队的最短编队距离以及最短编队路径集合。本发明通过递归算法计算无人机集群重新编队后的最短编队路径集合，可以提高无人机在执行任务中的续航时间，减少系统消耗，同时利用尾递归优化递归算法，有效解决了因为递归算法而产生的爆栈问题，保证程序的正常运行。



CN 110347180 A

1.一种计算无人机集群重新编队的最短编队距离的方法,其特征在于,主要包括以下步骤:

步骤S1、设计递归算法,并对该递归算法进行预处理,以找出递归代码部分;

步骤S2、将递归算法中的递归代码部分利用CPS技术转化为尾递归形式;

步骤S3、利用Trampoline技术对尾递归形式进行优化,以获得尾递归算法;

步骤S4、利用优化后的尾递归算法计算无人机集群重新编队的最短编队距离以及最短编队路径集合。

2.根据权利要求1所述的计算无人机集群重新编队的最短编队距离的方法,其特征在于,步骤S2具体包括:

步骤S21:对递归算法中的递归函数添加一个cont参数,标记需要转化的部分;

步骤S22:利用CPS技术进行转化,将递归算法中的递归函数全部转化为尾递归形式。

3.根据权利要求2所述的计算无人机集群重新编队的最短编队距离的方法,其特征在于,步骤S3具体包括:

步骤S31:设置一个对象cont_v＝{cont:…,v:…},用来保存每次得出的结果;

步骤S32:利用Trampoline技术手动强制弹出下一层调用的函数,禁止解释器的压栈行为,将尾递归形式改成循环形式。

4.根据权利要求1所述的计算无人机集群重新编队的最短编队距离的方法,其特征在于,步骤S1中的预处理为:根据无人机集群重新编队前和重新编队后的位置集合,计算出n个无人机的编队距离矩阵,即n阶矩阵D。

5.根据权利要求4所述的计算无人机集群重新编队的最短编队距离的方法,其特征在于:重新编队前无人机集群的位置集合为$W＝\{x_1,x_2...x_n\}$,重新编队后无人机集群的位置集合为$W'＝\{x_1',x_2'...x_n'\}$。

6.根据权利要求5所述的计算无人机集群重新编队的最短编队距离的方法,其特征在于:步骤S4中的最短编队路径集合为$S_{out}＝\{d_{n,n'}|||x_n'-x_n||\}$。

7.根据权利要求6所述的计算无人机集群重新编队的最短编队距离的方法,其特征在于:矩阵元素$d_{n,n'}$为单个无人机重新编队后移动的路径,即$D＝[d_{n,n'}]$。

8.根据权利要求7所述的计算无人机集群重新编队的最短编队距离的方法,其特征在于:设置路径栈S用于存放单个无人机重新编队后移动的路径$d_{n,n'}$。

9.根据权利要求8所述的计算无人机集群重新编队的最短编队距离的方法,其特征在于:最短编队距离S_{min}为路径栈S的值,路径栈S的值为当前路径栈中所有矩阵元素$d_{n,n'}$的和。

10.根据权利要求9所述的计算无人机集群重新编队的最短编队距离的方法,其特征在于,最短编队距离算法为:

步骤1:初始化n阶矩阵D,设置初始路径栈S为空,初始最短编队路径集合S_{out}为空,初始最短编队距离S_{min}为$+\infty$;

步骤2:若当前矩阵维数n为0,则计算当前路径栈S的值,并比较当前路径栈S和已知最短编队距离S_{min}的大小,输出两者最小值,并将最小值输入至最短编队距离S_{min},将当前路径输入至最短编队路径集合S_{out},并进入步骤6;若当前矩阵维数n不为0,则进入步骤3;

步骤3:在当前矩阵的第一行逐一选择矩阵元素$d_{n,n'}$并将矩阵元素$d_{n,n'}$输入路径栈S;

步骤4:将当前矩阵进行压栈操作,删除矩阵元素$d_{n,n'}$所在行和所在列的所有元素,对当前矩阵降1阶;

步骤5:重复步骤2;

步骤6:对路径栈S进行出栈操作,恢复n阶矩阵D;

步骤7:若n阶矩阵D的第一行元素没有选择完,则进入步骤3,否则进入步骤8;

步骤8:输出最短编队距离S_{min}和最短编队路径集合S_{out}。

计算无人机集群重新编队的最短编队距离的方法

技术领域

[0001] 本发明涉及一种计算无人机集群重新编队的最短编队距离的方法，属于无人机编队领域。

背景技术

[0002] 随着近十几年来控制技术、通信技术和计算技术的飞速发展，多无人机的协调控制受到广泛重视，其对于大区域范围的考察和开发工作具有重要价值和经济意义。

[0003] 在实际无人机的编队控制中，重新编队是多智能体在作业过程中不可避免要遇到的问题，这一过程是指智能体从原有队形被打乱到新队形形成的过程。最短编队距离问题是指寻找一种可行的算法，使得所有智能体在重新编队过程中的行程之和最小，本发明采用一种递归思想描述遍历算法。众所周知，递归容易引起爆栈，一旦发生爆栈，计算过程就会报错从而影响整个程序的运行。究其原因，便是函数调用前需要先将参数、运行状态压栈，而递归则会导致函数的多次无返回调用，参数、状态积压在栈上，最终耗尽栈空间。利用CPS变换，把任意递归函数改写成尾调用形式，以continuation链的形式，将递归占用的栈空间转移到堆上，以避免爆栈。对系统编队控制的研究则通常转化为智能体的目标跟随问题，通过选择队形中的固定位置作为跟随目标最终使多智能体形成编队，而实际情况下智能体由于选择队形中参考点的不同，其编队过程中智能体的行程也有较大差别。因此如何选择合适的相对位置，使编队中的智能体行程尽可能缩短，对多智能体系统有重要的意义。

[0004] 有鉴于此，确有必要提出一种计算无人机集群重新编队的最短编队距离的方法，以解决上述问题。

发明内容

[0005] 本发明的目的在于提供一种计算无人机集群重新编队的最短编队距离的方法，以解决递归引发的爆栈问题，从而提高整个系统程序可读性、可维护性和响应性。

[0006] 为实现上述目的，本发明提供了一种计算无人机集群重新编队的最短编队距离的方法，主要包括以下步骤：

[0007] 步骤S1、设计递归算法，并对该递归算法进行预处理，以找出递归代码部分；

[0008] 步骤S2、将递归算法中的递归代码部分利用CPS技术转化为尾递归形式；

[0009] 步骤S3、利用Trampoline技术对尾递归形式进行优化，以获得尾递归算法；

[0010] 步骤S4、利用优化后的尾递归算法计算无人机集群重新编队的最短编队距离以及最短编队路径集合。

[0011] 可选的，步骤S2具体包括：

[0012] 步骤S21：对递归算法中的递归函数添加一个cont参数，标记需要转化的部分；

[0013] 步骤S22：利用CPS技术进行转化，将递归算法中的递归函数全部转化为尾递归形式。

[0014] 可选的，步骤S3具体包括：

[0015] 步骤S31:设置一个对象cont_v＝{cont:…,v:…},用来保存每次得出的结果;

[0016] 步骤S32:利用Trampoline技术手动强制弹出下一层调用的函数,禁止解释器的压栈行为,将尾递归形式改成循环形式。

[0017] 可选的,步骤S1中的预处理为:根据无人机集群重新编队前和重新编队后的位置集合,计算出n个无人机的编队距离矩阵,即n阶矩阵D。

[0018] 可选的,重新编队前无人机集群的位置集合为$W＝\{x_1,x_2...x_n\}$,重新编队后无人机集群的位置集合为$W'＝\{x_1',x_2'...x_n'\}$。

[0019] 可选的,步骤S4中的最短编队路径集合为$S_{out}＝\{d_{n,n'}|||x_n'-x_n||\}$。

[0020] 可选的,矩阵元素$d_{n,n'}$为单个无人机重新编队后移动的路径,即$D＝[d_{n,n'}]$。

[0021] 可选的,设置路径栈S用于存放单个无人机重新编队后移动的路径$d_{n,n'}$。

[0022] 可选的,最短编队距离S_{min}为路径栈S的值,路径栈S的值为当前路径栈中所有矩阵元素$d_{n,n'}$的和。

[0023] 可选的,最短编队距离算法为:

[0024] 步骤1:初始化n阶矩阵D,设置初始路径栈S为空,初始最短编队路径集合S_{out}为空,初始最短编队距离S_{min}为$+\infty$;

[0025] 步骤2:若当前矩阵维数n为0,则计算当前路径栈S的值,并比较当前路径栈S和已知最短编队距离S_{min}的大小,输出两者最小值,并将最小值输入至最短编队距离S_{min},将当前路径输入至最短编队路径集合S_{out},并进入步骤6;若当前矩阵维数n不为0,则进入步骤3;

[0026] 步骤3:在当前矩阵的第一行逐一选择矩阵元素$d_{n,n'}$并将矩阵元素$d_{n,n'}$输入路径栈S;

[0027] 步骤4:将当前矩阵进行压栈操作,删除矩阵元素$d_{n,n'}$所在行和所在列的所有元素,对当前矩阵降1阶;

[0028] 步骤5:重复步骤2;

[0029] 步骤6:对路径栈S进行出栈操作,恢复n阶矩阵D;

[0030] 步骤7:若n阶矩阵D的第一行元素没有选择完,则进入步骤3,否则进入步骤8;

[0031] 步骤8:输出最短编队距离S_{min}和最短编队路径集合S_{out}。

[0032] 本发明的有益效果是:本发明通过递归算法计算无人机集群重新编队后的最短编队路径集合,可以提高无人机在执行任务中的续航时间,减少系统消耗,同时利用尾递归优化递归算法,有效解决了因为递归算法而产生的爆栈问题,保证程序的正常运行。

附图说明

[0033] 图1是本发明计算无人机集群重新编队的最短编队距离的算法流程图。

[0034] 图2是图1所示算法中矩阵元素压栈、矩阵降阶的过程示意图。

[0035] 图3是图1所示算法中,在压栈中出栈并选择次阶矩阵中不同元素完成新的全排列的过程示意图。

[0036] 图4是图1所示算法中矩阵元素出栈、恢复n阶矩阵D的过程示意图。

具体实施方式

[0037] 为了使本发明的目的、技术方案和优点更加清楚,下面结合附图和具体实施例对

本发明进行详细描述。

[0038] 本发明揭示了一种计算无人机集群重新编队的最短编队距离的方法，主要包括以下步骤：

[0039] 步骤S1、设计递归算法，并对该递归算法进行预处理，以找出递归代码部分；

[0040] 步骤S2、将递归算法中的递归代码部分利用CPS技术转化为尾递归形式；

[0041] 步骤S3、利用Trampoline技术对尾递归形式进行优化，以获得尾递归算法；

[0042] 步骤S4、利用优化后的尾递归算法计算无人机集群重新编队的最短编队距离以及最短编队路径集合。

[0043] 以下将对步骤S1-步骤S4做详细说明。

[0044] 其中，步骤S1具体包括：

[0045] 步骤S11(预处理)：假设有n(n>1000)架无人机组成三角编队队形，编队飞行过程中遭遇障碍物，需要重新调整队形，则先记录下重新编队前n架无人机的位置集合$W=\{x_1, x_2...x_n\}$，及重新编队后的位置集合$W'=\{x_1', x_2'...x_n'\}$，继而计算出无人机编队所有路径集合为$S_{out}=\{d_{n,n'}|||x_n'-x_n||\}$，矩阵元素$d_{n,n'}$为单个无人机重新编队后移动的路径，由此得出无人机的编队距离n阶矩阵$D=[d_{n,n'}]$；

[0046] 步骤S12：设计一种最短编队算法，最短编队距离指在有限时间内无人机在队形变化过程中的行程之和最小，记为S_{min}；

[0047] 步骤S13：假设会有n!种全排列，设计一种递归思想的遍历算法，假设从n阶矩阵D的第一行逐行遍历，算法的基本思想是如果已知前n-1步的决策过程，第n步决策只需在第n行中逐一选择不同列的元素，并通过比较得出当前最优解集；

[0048] 步骤S14：设计决策过程，初始化一个最短路径S_{out}为空，设置一个初始路径栈S，用来存放单个无人机重新编队后移动的路径$d_{n,n'}$，路径栈S的值为当前路径栈中所有矩阵元素$d_{n,n'}$的和，初始化最短路径S_{min}的值，可以设为$+\infty$。

[0049] 其中，步骤S2具体包括：

[0050] 步骤S21：对递归算法中的递归函数添加一个cont参数，标记需要转化的部分，为后续CPS转化做准备；

[0051] 步骤S22：利用CPS技术进行转化，将递归算法中的递归函数全部转化为尾递归形式。

[0052] 其中，步骤S3具体包括：

[0053] 步骤S31：首先设置一个对象$cont_v=\{cont:\cdots, v:\cdots\}$，用来保存每次得出的结果，当递归计算过程中，超过一定范围的参数压栈，会引发爆栈行为；

[0054] 步骤S32：接下来处理压栈问题，利用Trampoline技术手动强制弹出下一层调用的函数，禁止解释器的压栈行为，本质上就是将尾递归形式改成循环形式。

[0055] 假设决策者有路径栈S，每一步决策过程由当前编队距离矩阵的第一行开始逐个选取元素，并将元素进行入栈操作。随后将矩阵进行降阶处理，删除入栈元素所在的行和列并开始下一步决策。递归结束的条件是当前矩阵为空，表示路径栈S完成1次全排。在计算1次全排结果后，将路径栈S中的元素进行出栈操作。此时将相应的矩阵进行升阶处理，恢复出栈元素所在的行列，开始下一次全排过程。

[0056] 如图2所示，通过依次将d_{11}、d_{22}、d_{33}、...、d_{nn}压栈，每次压栈后删除相应矩阵元素所

在行和所在列的所有矩阵元素，直到当前矩阵维数n为0时，可以计算$S=d_{11}+d_{22}+d_{33}+...+d_{nn}$。

[0057] 如图3所示，在压栈过程中对路径栈S中的元素进行出栈操作，同时恢复矩阵中出栈元素所在的行和列。例如，当d22出栈后，恢复的二阶矩阵中的第一行元素d23没有选取，故对d23进行压栈操作，直到当前矩阵维数n为0时，计算$S=d_{11}+d_{23}+d_{32}+d_{44}+...+d_{nn}$。

[0058] 如图4所示，如果当前矩阵维数n为0，则将矩阵元素出栈、恢复n阶矩阵D，为下一轮迭代做准备。

[0059] 由此可见，可将无人机集群重新编队后的最短编队距离通过以下算法计算获得，具体如图1所示：

[0060] 步骤1：初始化n阶矩阵D，设置初始路径栈S为空，初始最短编队路径集合S_{out}为空，初始最短编队距离S_{min}为$+\infty$；

[0061] 步骤2：若当前矩阵维数n为0，则计算当前路径栈S的值，并比较当前路径栈S和已知最短编队距离S_{min}的大小，输出两者最小值，并将最小值输入至最短编队距离S_{min}，将当前路径输入至最短编队路径集合S_{out}，并进入步骤6；若当前矩阵维数n不为0，则进入步骤3；

[0062] 步骤3：在当前矩阵的第一行逐一选择矩阵元素$d_{n,n'}$并将矩阵元素$d_{n,n'}$输入路径栈S；

[0063] 步骤4：将当前矩阵进行压栈操作，删除矩阵元素$d_{n,n'}$所在行和所在列的所有元素，对当前矩阵降1阶；

[0064] 步骤5：重复步骤2；

[0065] 步骤6：对路径栈S进行出栈操作，恢复n阶矩阵D；

[0066] 步骤7：若n阶矩阵D的第一行元素没有选择完，则进入骤3，否则进入步骤8；

[0067] 步骤8：输出最短编队距离S_{min}和最短编队路径集合S_{out}。

[0068] 综上所述，本发明通过递归算法来计算无人机集群重新编队后的最短编队路径集合，可以提高无人机在执行任务中的续航时间，减少系统消耗；同时，利用尾递归优化递归算法，有效解决了因为递归算法而产生的爆栈问题，保证程序的正常运行。在一些环境相对复杂且对系统功耗要求较高的应用场合，能够解决系统重编队过程中的最短编队距离问题，有较为重要的实际意义。

[0069] 以上实施例仅用以说明本发明的技术方案而非限制，尽管参照较佳实施例对本发明进行了详细说明，本领域的普通技术人员应当理解，可以对本发明的技术方案进行修改或者等同替换，而不脱离本发明技术方案的精神和范围。

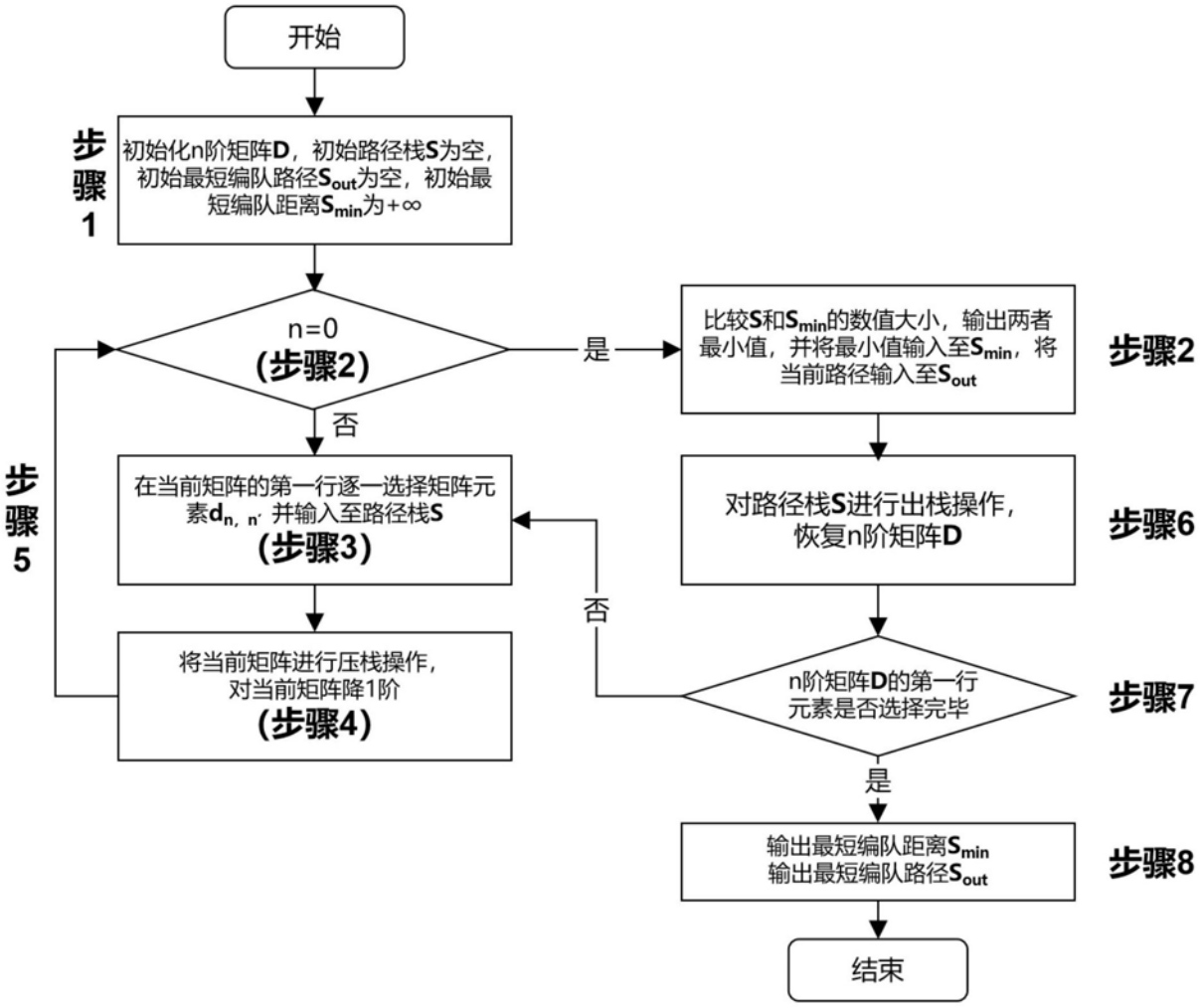

开始
步骤1
初始化n阶矩阵D，初始路径栈S为空，初始最短编队路径S_{out}为空，初始最短编队距离S_{min}为+∞
n=0
(步骤2)
是
比较S和S_{min}的数值大小，输出两者最小值，并将最小值输入至S_{min}，将当前路径输入至S_{out}
步骤2
否
步骤5
在当前矩阵的第一行逐一选择矩阵元素$d_{n,n'}$并输入至路径栈S
(步骤3)
将当前矩阵进行压栈操作，对当前矩阵降1阶
(步骤4)
对路径栈S进行出栈操作，恢复n阶矩阵D
步骤6
否
n阶矩阵D的第一行元素是否选择完毕
步骤7
是
输出最短编队距离S_{min}
输出最短编队路径S_{out}
步骤8
结束

图1

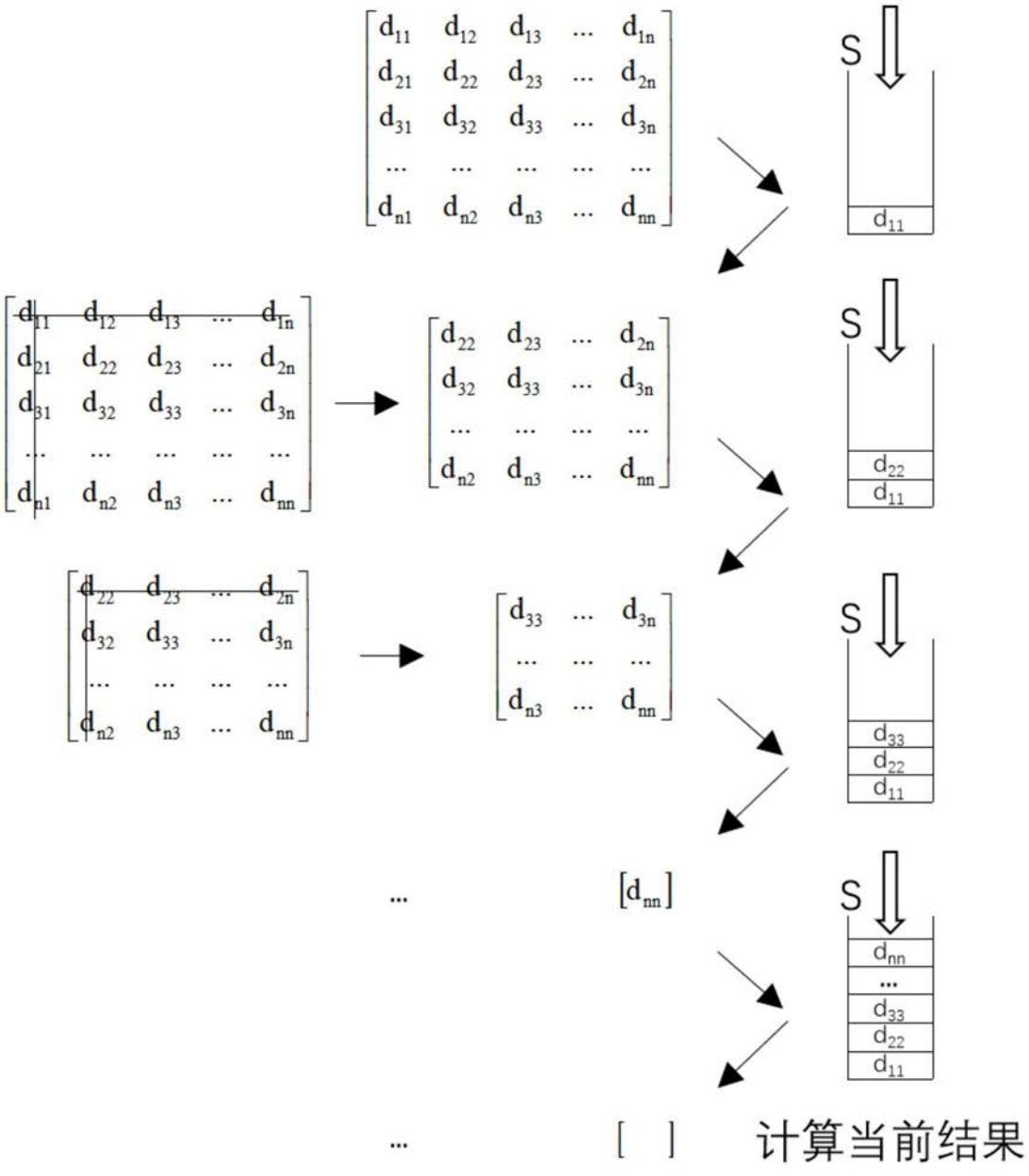


图2

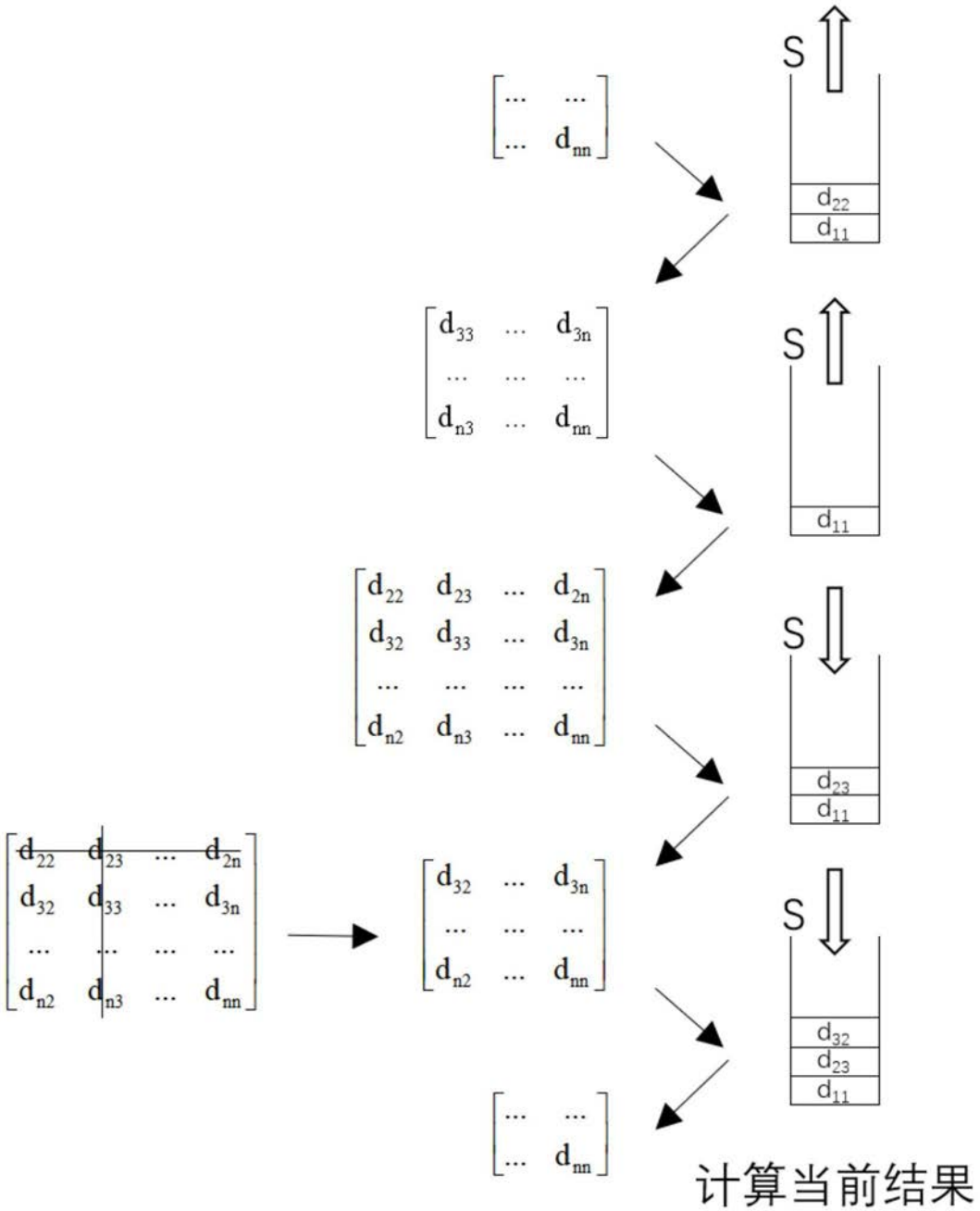


图3

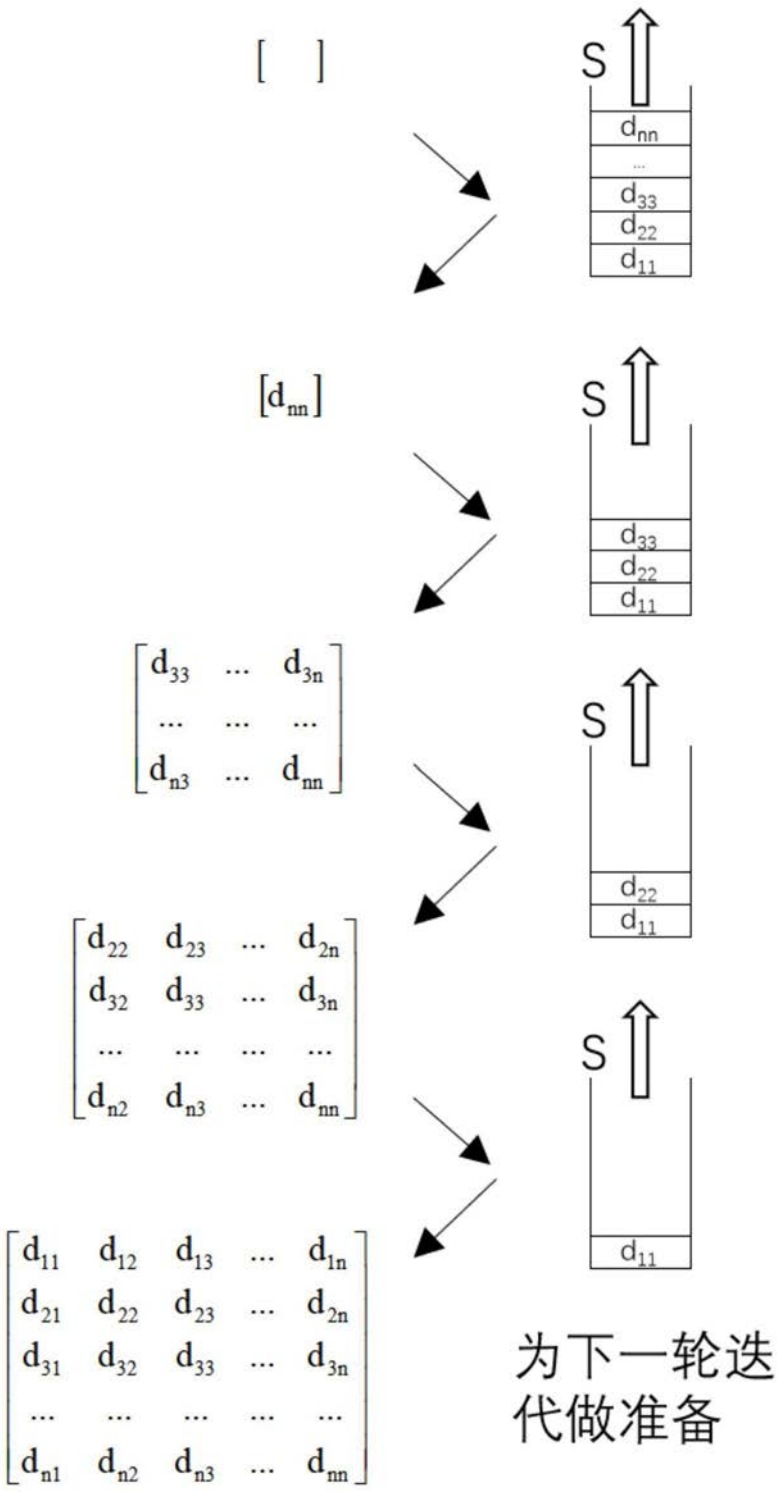

[ ]
S
d_{nn}
...
d_{33}
d_{22}
d_{11}
$[d_{nn}]$
S
d_{33}
d_{22}
d_{11}
$\begin{bmatrix} d_{33} & \dots & d_{3n} \\ \dots & \dots & \dots \\ d_{n3} & \dots & d_{nn} \end{bmatrix}$
S
d_{22}
d_{11}
$\begin{bmatrix} d_{22} & d_{23} & \dots & d_{2n} \\ d_{32} & d_{33} & \dots & d_{3n} \\ \dots & \dots & \dots & \dots \\ d_{n2} & d_{n3} & \dots & d_{nn} \end{bmatrix}$
S
d_{11}
$\begin{bmatrix} d_{11} & d_{12} & d_{13} & \dots & d_{1n} \\ d_{21} & d_{22} & d_{23} & \dots & d_{2n} \\ d_{31} & d_{32} & d_{33} & \dots & d_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ d_{n1} & d_{n2} & d_{n3} & \dots & d_{nn} \end{bmatrix}$
为下一轮迭代做准备

图4