(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 112749830 B
(45)授权公告日 2024.04.26

(21)申请号 202011190697.5

(22)申请日 2020.10.30

(65)同一申请的已公布的文献号
申请公布号 CN 112749830 A

(43)申请公布日 2021.05.04

(30)优先权数据
1912231 2019.10.30 FR
1912777 2019.11.15 FR

(73)专利权人 萨沃伊公司
地址 法国第戎
专利权人 贡比涅技术大学

(72)发明人 布兰丁·瓦彻 斯蒂凡·皮托维茨 安托万·约格莱特 德里坦·内斯

(74)专利代理机构 北京戈程知识产权代理有限公司 11314
专利代理师 程伟 王锦阳

(51)Int.Cl.
G06Q 10/04(2023.01)
G06Q 10/063(2023.01)
G06Q 10/087(2023.01)
B65G 43/10(2006.01)

(56)对比文件
CN 102673816 A,2012.09.19
CN 106470922 A,2017.03.01
CN 107640566 A,2018.01.30
CN 109153517 A,2019.01.04
CN 109313731 A,2019.02.05
CN 110111048 A,2019.08.09
CN 110140139 A,2019.08.16
US 2006245858 A1,2006.11.02
US 2010300048 A1,2010.12.02
US 2015066198 A1,2015.03.05
US 2015291356 A1,2015.10.15
US 4501528 A,1985.02.26

审查员 张绚

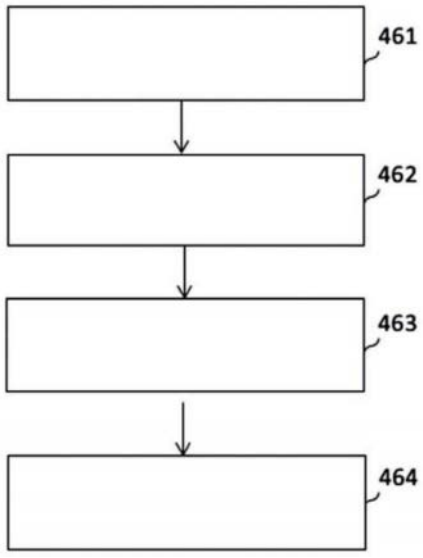
权利要求书4页 说明书34页 附图31页

(54)发明名称

用于在物流仓库内将k个有效负载输入流合并成一个输出流的方法

(57)摘要

一种用于在物流仓库内将分别由k个被称为通道a_i,其中$i\in\{1,\ldots,k\}$的输送机运输的k个有效负载输入流合并成由被称为收集器的另一个输送机运输的一个有效负载输出流的方法。所述k个通道是FIFO类型,沿所述收集器分布,并且在所述收集器的一个移动方向上被编号为a_1到a_k,Δ_i是通道a_i与a_k之间的时间相关距离。由控制系统执行的所述方法包括获得(416)集合L,所述集合L包括分布在所述k个通道上并且必须注入到所述收集器中以形成退出序列的n个有效负载;计算(462)日期t_0,在所述日期处,所述退出序列σ中的第一有效负载σ_1在通道a_k前面经过;根据t_0计算(463)将所述n个有效负载注入到所述收集器中的n个日期;命令(464)所述收集器和所述k个通道按照所述注入日期将所述n个有效负载注入到所述收集器中。



CN 112749830 B

1.一种用于在物流仓库内将分别由k个被称为通道a_i,其中$i\in\{1,...,k\}$的输送机运输的k个有效负载输入流合并成由被称为收集器的另一个输送机运输的一个有效负载输出流的方法:

所述物流仓库使得

-所述k个通道是"先进先出"类型,沿所述收集器分布,并且在所述收集器的一个移动方向上被编号为a_1到a_k,并且

-Δ_i是通道a_i与a_k之间的时间相关距离,以时间单位表示,各自与所述收集器的一个时间跨度相对应,

所述方法由控制系统执行至少一次,给定执行在时刻Tb处执行并且包括:

-获得(416)集合L,所述集合L包括在所述时刻Tb处分布在所述k个通道上并且必须注入到所述收集器中以形成退出序列σ的n个有效负载,所述n个有效负载中的每个有效负载均由所述退出序列σ内的单个序列号标识,所述k个通道中的每个通道均含有必须逐个注入到所述收集器中的h_i个有效负载的有序集合,所述有序集合按序列号的上升次序进行排序;

-计算(462)日期t_0,在所述日期处,所述序列σ中的第一有效负载σ_1在通道a_k前面经过;

-根据以下公式计算(463)将所述集合L的所述n个有效负载中的每个有效负载注入到所述收集器中的日期:$T(u)=t_0+\sigma(u)-1-\Delta_i$,其中:

*$u=a_i(j)\in L$,所述集合L的有效负载并且来自通道a_i的第j位置,$i\in\{1,\cdots,k\}$,$j\in\{1,\cdots,h_i\}$,

*σ(u)是所述退出序列σ中的所述有效负载u的所述序列号;

-命令(464)所述收集器和所述k个通道按照n个注入日期T(u),$\forall u\in L$将所述n个有效负载注入到所述收集器中。

2.根据权利要求1所述的方法,其中$\Delta_{i,i'}$是通道a_i与$a_{i'}$之间的时间相关距离,以时间单位表示,各自与所述收集器的时间跨度相对应,并且所述方法进一步包括以下步骤:

-对于来自通道a_i的每个有效负载$u=a_i(j)\in L$,计算(481)所述有效负载在通道a_i到a_k前面经过的日期,所述有效负载注入到所述收集器中的日期T(u)是已知的;

-对于每个通道a_i,$i\in\{1,...,k\}$,获得(482)各自含有所述有效负载之一在所述通道a_i前面经过的日期的集合V_i;

-对于由沿所述收集器定位并且在所述收集器的所述移动方向上在通道$a_{g(p)}$之前的控制构件控制的干扰流的每个受控干扰有效负载p,其中$g(p)\in\{1,...,k\}$,在所述收集器的所述移动方向上,$a_{g(p)}$也是受控干扰负载在其前面经过的第一通道(483):

a)用日期$\delta_1-\Delta_1$初始化t,其中δ_1表示如果第一受控干扰负载未搁置则此负载到达通道a_k前面的可能日期;

b)计算所述受控干扰负载在通道$a_{g(p)}$到$a_{1(p)}$前面经过的日期$t+\Delta_{1,g(p)}$、$t+\Delta_{1,g(p)+1}$、…、$t+\Delta_{1,1(p)}$,其中在所述收集器的所述移动方向上,$a_{1(p)}$是所述受控干扰负载在其前面经过的最后通道;

c)如果所述日期$t+\Delta_{1,g(p)}$、$t+\Delta_{1,g(p)+1}$、…、$t+\Delta_{1,1(p)}$均不属于相应集合$V_{g(p)}$、$V_{g(p)+1}$、…、$V_{1(p)}$,则用以下公式计算所述受控干扰负载p在通道a_1前面经过的假想日期$start_p$:$start_p=t$;

d)如果所述日期$t+\Delta_{1,g(p)}$、$t+\Delta_{1,g(p)+1}$、…、$t+\Delta_{1,l(p)}$之一属于相应集合$V_{g(p)}$、$V_{g(p)+1}$、…、$Vl_{l(p)}$之一,则将t递增一个单位并且以新的t值重复步骤b)、c)和d);

-命令(484)所述控制构件在每个受控干扰负载p位于通道$a_{g(p)}$之前的日期$start_p+\Delta_{1,g(p)}$处进行注入。

3.一种用于在物流仓库内将分别由k个被称为通道a_i,其中$i\in\{1,...,k\}$的输送机运输的k个有效负载输入流合并成由被称为收集器的另一个输送机运输的一个有效负载输出流的方法,所述物流仓库使得:

-所述k个通道是"先进先出"类型,沿所述收集器分布,并且在所述收集器的一个移动方向上被编号为a_1到a_k,并且

-Δ_i是通道a_i与a_k之间的时间相关距离,以时间单位表示,各自与所述收集器的一个时间跨度相对应,

所述方法由控制系统执行至少一次,给定执行在时刻Tb处完成并且包括:

-获得(471)集合L,所述集合L包括在所述时刻Tb处分布在所述k个通道上并且必须注入到所述收集器中以形成退出序列σ的n个有效负载,所述n个有效负载中的每个有效负载均由所述退出序列σ内的单个序列号标识,所述k个通道中的每个通道均含有必须逐个注入到所述收集器中的h_i个有效负载的按序列号的上升次序的有序集合;

-对于每个通道a_i,$i\in\{1,...,k\}$,获得(472)由于不受控干扰负载或所述收集器的时间跨度的另一个阻碍物而对于将所述通道a_i的所述有效负载之一注入到所述收集器中无效的日期集合U_i;

-在不考虑所述集合U_i,$i\in\{1,...,k\}$的情况下,计算(473)日期t0,在所述日期处,所述退出序列σ中的第一有效负载σ_1在通道a_k前面经过;

-对于所述退出序列σ中的所述第一有效负载σ_1,假设所述第一有效负载σ_1来自所述通道a_i(474):

a)用t_0初始化t;

b)如果日期t、$t-\Delta_{k-1}$、$t-\Delta_{k-2}$、...、$t-\Delta_i$均不属于相应集合U_k、U_{k-1}、U_{k-2}、...、U_i,则用以下公式计算将所述第一有效负载σ_1注入到所述收集器中的日期:$T(u)=t-\Delta_i$,其中$u=\sigma_1=a_i(j)$;

c)如果所述日期t、$t-\Delta_{k-1}$、$t-\Delta_{k-2}$、…、$t-\Delta_i$之一属于相应集合U_k、U_{k-1}、U_{k-2}、…、U_i之一,则将t递增一个单位并且以新的t值重复步骤b)和c);

-对于所述退出序列σ中的每个后续有效负载σ_c,$c\in\{2,...,n\}$,假设所述后续有效负载σ_c来自所述通道ai(475):

a′)将t递增一个单位,其中t用于计算先前有效负载σ_{c-1}的注入日期;

b′)如果所述日期t、$t-\Delta_{k-1}$、$t-\Delta_{k-2}$、…、$t-\Delta_i$均不属于相应集合U_k、U_{k-1}、U_{k-2}、…、U_i,则用以下公式计算将所述后续有效负载σ_c注入到所述收集器中的日期:$T(u)=t-\Delta_i$,其中$u=\sigma_c=a_i(j)$;

c′)如果所述日期t、$t-\Delta_{k-1}$、$t-\Delta_{k-2}$、…、$t-\Delta_i$之一属于相应集合U_k、U_{k-1}、U_{k-2}、…、U_i之一,则将t递增一个单位并且以新的t值重复步骤b′)和c′);

-命令(478)所述收集器和所述k个通道按照n个注入日期T(u),$\forall u\in L$将所述n个有效负载注入到所述收集器中。

4.根据权利要求3所述的方法,其中在命令所述收集器和所述k个通道按照所述n个注入日期T(u),$\forall u \in L$将所述n个有效负载注入到所述收集器中的步骤(478)之前执行以下步骤:

-根据以下公式计算(476)t_{min}:$t_{min} = min\{T(u),\ \forall u \in L\}$;

-如果$t_{min}>0$,则根据以下公式修改(477)所述n个注入日期:$T(u)=T(u)-t_{min}$,$\forall u \in L$。

5.根据权利要求3或4所述的方法,其中$\Delta_{i,i'}$是通道a_i与$a_{i'}$之间的时间相关距离,以时间单位表示,各自与所述收集器的时间跨度相对应,并且所述方法还包括以下步骤:

-对于来自通道ai的每个有效负载$u=a_i(j)\in L$,计算(491)所述有效负载在通道ai到ak前面经过的日期,所述有效负载到所述收集器中的注入日期T(u)是已知的;

-对于每个通道ai,$i\in\{1,\ldots,k\}$,一方面,获得(492)含有所述有效负载之一在所述通道a_i前面经过的每个日期的集合V_i,并且另一方面,获得对于将所述通道a_i的所述有效负载之一注入到所述收集器中无效的日期集合U_i;

-对于由沿所述收集器定位并且在所述收集器的所述移动方向上在通道$a_{g(p)}$之前的控制构件控制的干扰流的每个受控干扰负载p,其中$g(p)\in\{1,\ldots,k\}$,在所述收集器的所述移动方向上,$a_{g(p)}$也是所述受控干扰负载在其前面经过的第一通道(493):

a)用日期$\delta_1-\Delta_1$初始化t,其中δ_1表示如果第一干扰负载未搁置则此负载到达通道a_k前面的可能日期;

b)计算所述受控干扰负载在通道$a_{g(p)}$到$a_{1(p)}$前面经过的日期$t+\Delta_{1,g(p)}$、$t+\Delta_{1,g(p)+1}$、…、$t+\Delta_{1,1(p)}$,其中在所述收集器的所述移动方向上,$a_{1(p)}$是所述受控干扰负载在其前面经过的最后通道;

c)如果所述日期$t+\Delta_{1,g(p)}$、$t+\Delta_{1,g(p)+1}$、…、$t+\Delta_{1,1(p)}$均不属于相应集合$V_{g(p)}$、$V_{g(p)+1}$、…、$V_{1(p)}$,则用以下公式计算所述受控干扰负载p在通道a1前面经过的假想日期$start_p$:$start_p=t$;

d)如果所述日期$t+\Delta_{1,g(p)}$、$t+\Delta_{1,g(p)+1}$、…、$t+\Delta_{1,1(p)}$之一属于相应集合$V_{g(p)}$、$V_{g(p)+1}$、…、$V_{1(p)}$,之一,则将t递增一个单位并且以新的t值重复步骤b)、c)和d);

-命令(494)所述控制构件在每个受控干扰负载p位于通道$a_{g(p)}$之前的日期$start_p+\Delta_{1,g(p)}$处进行注入。

6.根据权利要求1或3所述的方法,其中所述日期t_0用以下公式计算:

$t_0 = \max_{u=a_i(j)\in L'}\{\Delta_i+1-\sigma(u)\}$,其中:

-$u=a_i(j)\in L'$,集合L'的有效负载,所述集合L'包括置于所述通道的第一位置中的有效负载,根据所述收集器的所述移动方向,所述通道被定位成从所述通道a_1一直到含有所述退出序列σ中的所述第一有效负载σ_1的通道a_{i1}。

7.根据权利要求1或3所述的方法,其特征在于,所述方法是迭代执行的,每个新执行均在用以下公式计算的新时刻Tb处进行:$Tb = T(u_{last})-\Delta_1+\Delta_{i^{last}}$,其中最后有效负载$u_{last}=a_{i^{last}}(j^{last})$,所述最后有效负载来自先前时刻Tb处的先前执行的所述退出序列σ的通道$a_{i^{last}}$中的第(j^{last})位置。

8.根据权利要求1或3所述的方法,其特征在于,所述方法是迭代执行的,每个新执行均在被定义为以下时刻的新时刻Tb处进行:在所述时刻处,先前时刻Tb处的先前执行的所述

退出序列σ的负载未定位于所述收集器的一部分中,所述部分从第一通道a_1到含有所述新时刻Tb处的所述新执行的所述退出序列σ的第一负载的通道a_{i_1}。

9.根据权利要求1或3所述的方法,其特征在于,所述方法是迭代执行的,每个新执行均在用以下公式计算的新时刻Tb处进行:$Tb = Max(Tb+1, T(u_{last}) - \Delta_1 + \Delta_{i^{last}})$,其中:

-$u_{last} = a_{i^{last}}(j^{last})$是最后有效负载,所述最后有效负载来自先前时刻Tb处的先前执行的所述退出序列σ的通道$a_{i^{last}}$中的第(j^{last})位置,并且

-(Tb+1)是通过将所述先前时刻Tb递增一个单位而获得。

用于在物流仓库内将k个有效负载输入流合并成一个输出流的方法

技术领域

[0001] 1.技术领域

[0002] 在过去的几年中,内部物流活动已经受到消费者社会行为的急剧变化和新技术发展的极大影响。需求的不断增长迫使物流的主要参与者变得越来越具反应力和竞争力。改善并优化路由、存储和客户订单拣选解法已经变得至关重要。SAVOYE公司是一家客户订单拣选设备的制造商,专门从事物流仓库的自动化和计算机化,并且不断改进并评估其解法。

[0003] 物流仓库的管理包含供应品采购、流和存量的接收和管理。改善客户订单的行进时间通常意指使其输送时间最小化。以下文献指出客户订单的拣选时间(准备时间)的50%专门用于此输送操作:De Koster,R.、Le-Duc,T.和Roodbergen,K.J.,“仓库订单拣选的设计和控制:文献综述(Design and control of warehouse order picking:a literature review)”(2007),《欧洲运筹学杂志(European Journal of Operational Research)》182(2),481-501。因此,首先有必要着眼于仓库所面临的主要问题——即优化所输送流以确保尽可能最高的生产率。将更特别地聚焦于负载(包装、箱、容器等)流的交叉点。目的是使组合来自不同地方的各种输入流(流入量)的最终流(也称为输出流或流出量)的吞吐量最大化。

背景技术

[0004] 2.难题和问题

[0005] 2.1目标

[0006] 在物流仓库中,恰巧将负载的若干个输入流(也称为注入流)组合成单个输出流(或流出量)(也称为退出流)。负载流与一列序列负载相对应,逐一提交。这些流可以通过类似于输送带的输送机运输,所述输送带可以在给定方向上以单行方式移动定位于其上的所有负载。在下文描述中,运输输入流的输送机被称为通道,并且运输退出流的输送机被称为收集器。

[0007] 注入流全部沿收集器分布。构成这些注入流的负载的位置是已知且可识别的。对于每个注入流,可以选择将其第一负载注入到收集器中的时刻。注入流的序列分布使定位于收集器流上游的通道能够受益于比定位于下游的通道更大的注入可能性。的确,只有在更上游的通道留有空间时,后者才能卸下其负载。

[0008] 为了防止注入流之间的负载注入不平衡,最好在将所有负载置于收集器上之前定义负载的次序(此次序称为最终序列或再次称为“退出序列”)。

[0009] 此处的管理思想是将收集器的流推到收集器的最大机械容量。为此,需要限制收集器中的未用空间的数量。提出了一种基于正确日期进行负载注入以构成没有未用空间(或尽可能少的未用空间)的最终流的方法。这些注入日期被定义成符合每个可实现的最终序列,即符合注入流的FIFO(先进先出)序列次序的每个序列。

[0010] 相对于已知的调度难题模型,下文所提出和描述以响应此组难题和问题的解法是

新颖且具有创造性的,例如描述于以下文献中:

[0011] • Runwey Cheng、MitsuoGen和YasuhiroTsujimura,“使用遗传算法进行的作业车间调度难题的教程调查---第一部分陈述(A tutorial survey of Job Shop scheduling problems using genetic algorithms---Part I.Representation)”,足利工业大学(Ashikaga Institute of Technology),足利326,日本;以及

[0012] • Imran Ali Chaudhry和Abid Ali Khan,“一项研究调查:柔性作业车间调度技术回顾(A research survey:review of flexible job shop scheduling techniques)”,《运筹学中的国际交易(International transactions in operation research)》,2015。

[0013] 2.2应用实例

[0014] 简而言之,所研究的系统由通道(输送机集合)所输送的注入流(输入流)和将这些注入流组合成单个最终流(输出流)的收集器(另一输送机)构成。下文所提出和描述的控制系统(或总体控制系统)可以尤其但非专有地应用于“连续非累积式输送机”类型的收集器或分类机类型的收集器(带盘、倾斜盘等)。此控制系统使能够符合任何最终可实现的序列。因此,相对于最终给定序列给出最佳吞吐量的解法。如果不需要最终序列,则能够在完全未占用的收集器上获得最大吞吐量以便组合这些注入流。

[0015] 然而,在某些情况下,阻碍物可能已经在收集器上占据了位置。应理解,术语“阻碍收集器的时间步长(或时间跨度)”尤其但非专有地意指来自不同于注入流的干扰流的干扰负载。阻碍物不一定是干扰负载,但也可能是收集器被损坏、保留的地方等。在此情况下,注入流的管理必须考虑到此类阻碍物。

[0016] 在其中干扰负载来自一个或多个干扰流的特定情况下,可以对两种类型的干扰流加以区分。如果控制系统不能控制构成干扰流的干扰负载到收集器的注入(这些干扰负载在不考虑环境的情况下到达),则所述干扰流被认为是“不受控”的。相反,如果控制系统可以控制构成干扰流的干扰负载到收集器的注入,则所述干扰流被认为是“受控”的。所提出的控制系统支持这些不同类型的干扰流(不受控或受控)并且给出了包含要控制的干扰流的管理在内的最佳解法。

[0017] 具体应用的一个说明性但非详尽性实例是由SAVOYE公司提出的GTP Intelis PTS(货物到人Intelis拣选盘系统)。这是一个全面的自动化客户订单拣选解法,其中基于穿梭的自动化存储和检索(或搬运)系统(称为Intelis PTS系统)3使能够以非常高的速度向拣选站点(也称为GTP站点4)进料。因此,自动化系统(Intelis PTS)的作用是存储并获得负载的进入和退出状态,所述负载包括在GTP站点处高效填充客户订单所需的物品。自动化系统由若干个存储架(也称为PTS过道,然而在本发明的上下文中在输送机的方向上不应被误认为是运输输入流的通道)(请参见下文的解释)组成,各自由能够分别在进入输送机5_1、5_2和退出输送机2_1、2_2上来回移动这些负载的梭子和升降机构成。这些不同的进入和退出输送机全部由收集器1连接,所述收集器向拣选站点4进料,如在图50中可以看到的(例如具有三个拣选站点4_1、4_2和4_3)。

[0018] 为了建立与上文的上下文定义的链接,自动化系统3的退出输送机2_1、2_2形成运输输入有效负载流的输送机(也称为“通道”),这些输入流在收集器1上合并成输出有效负载流。

[0019] 换句话说,借助于收集器来控制进入和退出自动化系统3的负载的来回移动。因

此,在自动化系统3的退出输送机2_1、2_2上有若干种注入负载流(在当前对难题和问题的讨论中也称为输入流),其全部注入到收集器上。

[0020] 另外,当返回自动化系统3的负载流(通过进入输送机5_1、5_2)穿过同一收集器1时,具有干扰流的实例,在此特定情况下称为返回流。这些干扰流非常有趣,因为其在穿过所有注入区之后便消失了,从而为某些注入流而不是其它注入流创造了未用空间。如下文所详细描述的,所提出的解法的一个实施例旨在尽可能地控制这些干扰流以填充这些未用空间而不是让其存在。

[0021] 在应用的此实例中提出的解法描述于下文文件中。

[0022] 2.3当前解法

[0023] 2.3.1现有解法

[0024] -已知解法A,有期望的最终排序:在此情况下,收集器上负载的注入遵循与负载合并之后的预期最终序列相对应的定义明确的序列。如果从收集器退出时所需的此最终序列由例如具有序列号1、2、3等的负载(在下文中称为"负载1"、"负载2"等)形成,则将优先注入负载1,然后仅当负载1在收集器上位于负载2下游时注入负载2,然后仅当负载2在收集器上位于负载3下游时注入负载3,并且依此类推。

[0025] -已知解法B,没有期望的最终排序:一旦其注入区有位置,就将每个注入流的每个第一负载注入到收集器中。

[0026] 2.3.2与这些解法相关的缺点

[0027] -已知解法A:已知解法A的一个缺点是其在负载之间留有(未用)空间,并且这相对于收集器的机械容量减慢了输出流。一旦最终序列中的先前负载属于比后续负载更下游的注入流,最终序列中的两个连续负载之间就会出现未用空间。另外,这些未用空间的长度与用于注入这两个连续负载的通道(输送机)的距离成比例。

[0028] 此已知解法A的若干个步骤如图1所示。在此实例中,考虑期望的最终序列1、2、3、4。为了遵守此序列,不可能同时在收集器上直接注入负载1、2、3和4。在步骤0处,激活安装。负载1和2立即注入到收集器中。因为负载1的注入流位于负载2的注入流的下游,所以进行此操作是可能的。负载1和2位于收集器上,由其所来自的通道(输送机)之间的距离隔开。收集器将在没有变化的情况下向前移动,直到步骤1,其中负载2在含有负载3的流的注入区前面经过。从此刻起,负载3直接注入到负载2后面的收集器中。可以注意到,负载1与负载2之间留出的未用空间保持不变并且永远不会被填充。在同一步骤1处,负载4直接注入到收集器中,在其自身与负载3之间留出空间。步骤2示出了此已知注入方法留出的未用空间。

[0029] -已知解法B:已知解法B的一个缺点是最下游的通道(输送机)将连续地注入其负载,而其它通道则将仅限于填充剩余的自由空间;已知解法B的另一个缺点是其提出了一种控制系统,所述控制系统仅在收集器(输出流的运输机)的速度远大于运输输入流的通道(输送机)的速度(比要注入的输入流大出很多倍)的情况下才能实际工作。在已知解法B中,必须限制通道(输送机)的速度。现在,在本文中,注入流的数量增加的速度快于收集器的机械速度。此已知解法B不再足以并且不能控制输出流的吞吐量的优化。

发明内容

[0030] 3.发明内容

[0031] 在本发明的第一特定实施例中,提出了一种用于在物流仓库内将分别由k个被称为通道a_i,其中$i\in\{1,...,k\}$的输送机运输的k个有效负载输入流合并成由被称为收集器的另一个输送机运输的一个有效负载输出流的方法,所述物流仓库使得:

[0032] • 所述k个通道是"先进先出"类型,沿所述收集器分布,并且在所述收集器的一个移动方向上被编号为a_1到a_k,并且

[0033] • Δ_i是通道a_i与a_k之间的时间相关距离,以时间单位表示,各自与所述收集器的一个时间跨度(也被称为"时间步长")相对应,

[0034] 所述方法由控制系统执行至少一次,给定执行在时刻Tb处执行并且包括:

[0035] -获得集合L,所述集合L包括在所述时刻Tb处分布在所述k个通道上并且必须注入到所述收集器中以形成退出序列σ的n个有效负载,所述n个有效负载中的每个有效负载均由所述退出序列σ内的单个序列号标识,所述k个通道中的每个通道均含有必须逐个注入到所述收集器中的hi个有效负载的有序集合,所述有序集合按序列号的上升次序进行排序;

[0036] -计算日期t_0,在所述日期处,所述序列σ中的第一有效负载σ_1在通道a_k前面经过;

[0037] -根据以下公式计算将所述集合L的所述n个有效负载中的每个有效负载注入到所述收集器中的日期:$T(u)=t_0+\sigma(u)-1-\Delta_i$,其中:

[0038] • $u=a_i(j)\in L$,所述集合L的有效负载并且来自通道a_i的第j位置,$i\in\{1,...,k\}$,$j\in\{1,...,h_i\}$,

[0039] • σ(u)是所述退出序列σ中的所述有效负载u的所述序列号;

[0040] -命令所述收集器和所述k个通道按照n个注入日期T(u),$\forall u\in L$将所述n个有效负载注入到所述收集器中。

[0041] 根据所述第一实施例的一个特定特性,$\Delta_{i,i}$,是通道a_i与a_i之间的时间相关距离,以时间单位表示,各自与所述收集器的时间跨度相对应,并且所述方法还包括以下步骤:

[0042] -对于来自通道a_i的每个有效负载,$u=a_i(j)\in L$,计算所述有效负载在所述通道a_i到a_k前面经过的日期,所述有效负载注入到所述收集器中的所述日期T(u)是已知的;

[0043] -对于每个通道a_i,$i\in\{1,...,k\}$,获得各自含有所述有效负载之一在所述通道a_i前面经过的日期的集合V_i,;

[0044] -对于由沿所述收集器定位并且在所述收集器的移动方向上在通道$a_{g(p)}$之前的控制构件控制的干扰流的每个受控干扰有效负载p,其中$g(p)\in\{1,...,k\}$,在所述收集器的移动方向上,$a_{g(p)}$也是所述受控干扰负载在其前面经过的第一通道:

[0045] a)用日期$\delta_1-\Delta_1$初始化t,其中δ_1表示如果此负载未搁置(即置于等待状态下)则第一受控干扰负载到达通道a_k前面的可能日期;

[0046] b)计算所述受控干扰负载在通道$a_{g(p)}$到$a_{l(p)}$之前经过的日期$t+\Delta_{1,g(p)}$、$t+\Delta_{1,g(p)+1}$、...、$t+\Delta_{1,l(p)}$,其中在所述收集器的移动方向上,$a_{l(p)}$是所述受控干扰在其前面经过的最后通道;

[0047] c)如果日期$t+\Delta_{1,g(p)}$、$t+\Delta_{1,g(p)+1}$、...、$t+\Delta_{1,l(p)}$均不属于相应集合$V_{g(p)}$、$V_{g(p)+1}$、...、$V_{l(p)}$,则用以下公式计算所述受控干扰负载p在通道a_1前面经过的假想日期$start_p$:$start_p=t$;

[0048] d)如果日期$t+\Delta_{1,g(p)}$、$t+\Delta_{1,g(p)+1}$、...、$t+\Delta_{1,l(p)}$之一属于相应集合$V_{g(p)}$、$V_{g(p)+1}$、...、$V_{l(p)}$之一,则将t递增一个单位并且以新的t值重复步骤b)、c)和d);

[0049] -命令所述控制构件在每个受控干扰负载p位于所述通道$a_{g(p)}$前面的日期$start_p+\Delta_{1,g(p)}$处进行注入。

[0050] 在本发明的第二特定实施例中,提出了一种用于在物流仓库内将分别由k个被称为通道a_i,其中$i\in\{1,...,k\}$的输送机运输的k个有效负载输入流合并成由被称为收集器的另一个输送机运输的一个有效负载输出流的方法,所述物流仓库使得:

[0051] •所述k个通道是"先进先出"类型,沿所述收集器分布,并且在所述收集器的一个移动方向上被编号为a_1到a_k,并且

[0052] • Δ_i是通道a_i与a_k之间的时间相关距离,以时间单位表示,各自与所述收集器的一个时间跨度相对应,

[0053] 所述方法由控制系统执行至少一次,给定执行在时刻Tb处完成并且包括:

[0054] -获得集合L,所述集合L包括在所述时刻Tb处分布在所述k个通道上并且必须注入到所述收集器中以形成退出序列σ的n个有效负载,所述n个有效负载中的每个有效负载均由所述退出序列σ内的单个序列号标识,所述k个通道中的每个通道均含有必须逐个注入到所述收集器中的h_i个有效负载的按序列号的上升次序的有序集合;

[0055] -对于每个通道a_i,$i\in\{1,...,k\}$,获得由于不受控干扰负载或所述收集器的时间跨度的另一个阻碍物而对于将通道a_i的所述有效负载之一注入到所述收集器中无效的日期集合U_i;

[0056] -在不考虑集合U_i,$i\in\{1,...,k\}$的情况下,计算日期t_0,在所述日期处,所述退出序列σ中的第一有效负载σ_1在通道a_k前面经过;

[0057] -对于所述退出序列σ中的所述第一有效负载σ_1,假设所述第一有效负载σ_1来自所述通道a_i:

[0058] a)用t_0初始化t;

[0059] b)如果日期t、$t-\Delta_{k-1}$、$t-\Delta_{k-2}$、...、$t-\Delta_i$均不属于相应集合U_k、U_{k-1}、U_{k-2}、...、U_i,则用以下公式计算将所述第一有效负载σ_1注入到所述收集器中的日期:$T(u)=t-\Delta_i$,其中$u=\sigma_1=a_i(j)$;

[0060] c)如果日期t、$t-\Delta_{k-1}$、$t-\Delta_{k-2}$、...、$t-\Delta_i$之一属于相应集合U_k、U_{k-1}、U_{k-2}、...、U_i之一,则将t递增一个单位并且以新的t值重复步骤b)和c);

[0061] -对于所述退出序列σ中的每个后续有效负载σ_c,$c\in\{2,...,n\}$,假设所述后续有效负载σ_c来自所述通道a_i:

[0062] a')将t递增一个单位t用于计算先前有效负载σ_{c-1}的注入日期;

[0063] b')如果日期t、$t-\Delta_{k-1}$、$t-\Delta_{k-2}$、...、$t-\Delta_i$均不属于相应集合U_k、U_{k-1}、U_{k-2}、...、U_i,则用以下公式计算将所述后续有效负载σ_c注入到所述收集器中的日期:$T(u)=t-\Delta_i$,其中$u=\sigma_c=a_i(j)$;

[0064] c')如果日期t、$t-\Delta_{k-1}$、$t-\Delta_{k-2}$、...、$t-\Delta_i$之一属于相应集合U_k、U_{k-1}、U_{k-2}、...、U_i之一,则将t递增一个单位并且以新的t值重复步骤b')和c');

[0065] -命令所述收集器和所述k个通道按照n个注入日期T(u),$\forall u\in L$将所述n个有效负载注入到所述收集器中。

[0066] 根据所述第二实施例的一个特定特性,在命令所述收集器和所述k个通道按照所述n个注入日期Y(u),$\forall u\in L$将所述n个有效负载注入到所述收集器中的步骤之前执行以下

步骤：

[0067] -根据以下公式计算t_{min}：$t_{min} = min\{T(u), \forall u \in L\}$；

[0068] -如果$t_{min}>0$，则根据以下公式修改所述n个注入日期：$T(u) = T(u) - t_{min}, \ \forall u \in L$。

[0069] 根据所述第二实施例的一个特定特性，$\Delta_{i,i'}$是通道a_i与$a_{i'}$之间的时间相关距离，以时间单位表示，各自与所述收集器的时间跨度相对应，并且所述方法还包括以下步骤：

[0070] -对于来自通道a_i的每个有效负载$u=a_i(j)\in L$，计算所述有效负载在通道a_i到a_k前面经过的日期，所述有效负载到所述收集器中的注入日期T(u)是已知的；

[0071] -对于每个通道a_i，$i\in\{1,\ldots,k\}$，一方面，获得含有所述有效负载之一在通道a_i前面经过的每个日期的集合V_i，并且另一方面，获得对于将通道a_i的所述有效负载之一注入到所述收集器中无效的日期集合U_i；

[0072] -对于由沿所述收集器定位并且在所述收集器的移动方向上在通道$a_{g(p)}$之前的控制构件控制的干扰流的每个受控干扰负载p，其中$g(p)\in\{1,\ldots,k\}$，在所述收集器的移动方向上，$a_{g(p)}$也是所述受控干扰负载在其前面经过的第一通道：

[0073] a)用日期$\delta_1-\Delta_1$初始化t，其中δ_1表示如果第一干扰负载未搁置则此负载到达通道a_k前面的可能日期；

[0074] b)计算所述受控干扰负载在通道$a_{g(p)}$到$a_{l(p)}$前面经过的日期$t+\Delta_{1,g(p)}$、$t+\Delta_{1,g(p)+1}$、...、$t+\Delta_{1,l(p)}$，其中在所述收集器的移动方向上，$a_{l(p)}$是所述受控干扰负载在其前面经过的最后通道；

[0075] c)如果日期$t+\Delta_{1,g(p)}$、$t+\Delta_{1,g(p)+1}$、...、$t+\Delta_{1,l(p)}$均不属于相应集合$V_{g(p)}$、$V_{g(p)+1}$、...、$V_{l(p)}$，则用以下公式计算所述受控干扰负载p在通道a_1前面经过的假想日期$start_p$：$start_p=t$；

[0076] d)如果日期$t+\Delta_{1,g(p)}$、$t+\Delta_{1,g(p)+1}$、...、$t+\Delta_{1,l(p)}$之一属于相应集合$V_{g(p)}$、$V_{g(p)+1}$、...、$V_{l(p)}$之一，则将t递增一个单位并且以新的t值重复步骤b)、c)和d)；

[0077] -命令所述控制构件在每个受控干扰负载p位于通道$a_{g(p)}$前面的日期$start_p+\Delta_{1,g(p)}$处进行注入。

[0078] 根据本发明的所述第一实施例或所述第二实施例的一个特定特性，所述日期t_0用以下公式计算：$t_0 = max_{u=a_i(j)\in L'}\{\Delta_i + 1 - \sigma(u)\}$，其中：

[0079] • $u=a_i(j)\in L'$，集合L'的有效负载，所述集合L'包括置于所述通道的第一位置中的有效负载，在所述收集器的移动方向上，所述通道被定位成从通道a_1一直到含有所述退出序列σ中的所述第一有效负载σ_1的通道a_{i1}。

[0080] 根据本发明的所述第一实施例或所述第二实施例的一个特定特性，所述方法是迭代执行的，每个新执行均在用以下公式计算的新时刻Tb处进行：$Tb = T(u_{last}) - \Delta_1 + \Delta_{i^{last}}$，其中$u_{last} = a_{i^{last}}(j^{last})$，最后有效负载，所述最后有效负载来自先前时刻Tb处的先前执行的所述退出序列σ的通道$a_{i^{last}}$中的第(j^{last})位置。

[0081] 根据本发明的所述第一实施例或所述第二实施例的一个特定特性，所述方法是迭代执行的，每个新执行均在被定义为以下时刻的新时刻Tb处进行：在所述时刻处，先前时刻Tb处的先前执行的所述退出序列σ的负载未定位于所述收集器的一部分中，所述部分从第

一通道a_1延伸到所述通道a_{i_1}，此通道a_{i_1}含有所述新时刻Tb处的所述新执行的所述退出序列σ的所述第一负载。

[0082] 根据本发明的所述第一实施例或所述第二实施例的一个特定特性，所述方法是迭代执行的，每个新执行均在用以下公式计算的新时刻Tb处进行：$Tb = Max(Tb + 1, T(u_{last}) - \Delta_1 + \Delta_{i^{last}})$，其中：

[0083] • $u_{last} = a_{i^{last}}(j^{last})$是最后有效负载，所述最后有效负载来自先前时刻Tb处的先前执行的所述退出序列σ的通道$a_{i^{last}}$中的第(j^{last})位置，并且

[0084] • (Tb+1)是通过将先前时刻Tb递增一个单位而获得。

[0085] 本发明的另一实施例提出了一种计算机程序产品，所述计算机程序产品包括当在计算机上执行所述程序时用于(在其不同实施例中的任一个实施例中)实施上述方法的程序代码指令。

[0086] 本发明的另一实施例提出了一种存储计算机程序的计算机可读且非暂时性存储介质，所述存储介质包括可由计算机执行以(在其不同实施例中的任一个实施例中)实施上述方法的指令集。

[0087] 本发明的另一个实施例提出了一种控制系统(装置)，所述控制系统包括用于在其不同实施例中的任一个实施例中实施如上文所描述的方法中执行的步骤的构件。

附图说明

[0088] 4.附图列表

[0089] 在以下描述中，通过指示性而非详尽性实例给出，对附图进行参考，其中：

[0090] [图1]示出了用于以给定序列次序插入负载的现有方法；

[0091] [图2]是两个通道(输送机)、一个收集器和一个负载(箱)的图像；

[0092] [图3]是两个通道(输送机)、一个收集器和一个负载(箱)的图；

[0093] [图4]是通道中正在研究的系统的图；

[0094] [图5]是具有按其序列号的负载标识的系统的图；

[0095] [图6]示出了离开点和注入点；

[0096] [图7]示出了相对于一般配置没有损失的一致点；

[0097] [图8]示出了将收集器切分成各个跨度(位置或槽)；

[0098] [图9]示出了收集器切分中预期的期望退出序列；

[0099] [图10]示出了与收集器跨度相关联的相对于负载注入的持续时间的时间单位；

[0100] [图11]示出了注入流之间的作为整数值的距离；

[0101] [图12]示出了注入流之间的无需为整数值的距离；

[0102] [图13]是在每个跨度处连续放置了注入点的收集器的图；

[0103] [图14]示出了具有根据收集器的跨度连续放置的四通道的收集器的实例；

[0104] [图15]是与作业车间和检索要通过公式计算的数据相关的甘特图；

[0105] [图16]示出了具有以任何未指定方式放置的四个通道的收集器的实例；

[0106] [图17]是与作业车间相关的甘特图，其中各通道之间有任何未指定距离；

[0107] [图18]示出了收集器和两个具有负载标识符的通道的实例；

[0108] [图19]示出了收集器和两个具有序列次序编码标识符的通道的实例;

[0109] [图20]示出了收集器和两个具有序列编码标识符的通道的实例;

[0110] [图21]示出了在时刻0处的收集器的图像;

[0111] [图22]示出了在时刻1处的收集器的图像;

[0112] [图23]示出了在时刻2处的收集器的图像;

[0113] [图24]示出了在时刻3处的收集器的图像;

[0114] [图25]示出了与三个通道相关联的FIFO列表的实例;

[0115] [图26]示出了收集器上预期的可实现序列;

[0116] [图27]示出了附接到注入点的流的第一负载,从而形成FIFO列表的一部分;

[0117] [图28]示出了未附接到注入点的流的第一负载,从而未形成FIFO列表的一部分;

[0118] [图29]示出了形成FIFO列表的一部分的第一负载后面的连续保持负载;

[0119] [图30]示出了未形成FIFO列表的一部分的第一负载后面的连续保持负载;

[0120] [图31]示出了工作批次的实例;

[0121] [图32]示出了批次1的注入和处理日期的计算;

[0122] [图33]示出了正在进行中的批次的处理;

[0123] [图34]示出了批次1的处理结束和批次2的形成;

[0124] [图35]示出了批次2的处理结束和其应用;

[0125] [图36]是在不受控干扰流的一般情况下的算法图表;

[0126] [图37]示出了干扰流的实例:灰色负载返回到PTS;

[0127] [图38]示出了具有四个具有不受控干扰流的未指定通道的收集器的实例;

[0128] [图39]是形成空间的不受控干扰流的甘特图;

[0129] [图40]是算法4的图表;

[0130] [图41]示出了具有四个具有受控干扰流的以任何未指定方式放置的通道的收集器的实例;

[0131] [图42]是示出了要用于干扰负载的空间的注入的甘特图;

[0132] [图43]是当选择干扰负载放置时的甘特图;

[0133] [图44]示出了根据本发明的一个特定实施例的控制系统的结构;

[0134] [图45]是算法5的图表;

[0135] [图46]是根据基于算法2的第一特定实施例的将k个输入流合并成一个输出流的方法的流程图;

[0136] [图47]是根据基于算法3(图36所示)的第二特定实施例的将k个输入流合并成一个输出流的方法的流程图;

[0137] [图48]是根据基于算法4(图40所示)的第三特定实施例的将k个输入流合并成一个输出流的方法的流程图;

[0138] [图49]是根据基于算法5(图45所示)的第四特定实施例的将k个输入流合并成一个输出流的方法的流程图;

[0139] [图50]示出了应用的实例。

具体实施方式

[0140] 5.控制系统参数

[0141] 如上文进一步所提及的,正在研究的系统由排出系统(称为收集器的输送机)、若干个其它输送机(称为通道)和负载形成。所述系统是动态的,并且负载由通道运输,注入到收集器中,并且然后由收集器运输。在第一阶段,将考虑此系统在给定时刻冻结存在于其中的每个负载的位置的状态。将定义此系统的冻结子系统以便能够计算某些负载的未来注入日期。这与静态问题的解析相对应。第7节将研究动态控制和操纵此系统的方式。

[0142] 图2是展示了两个通道(输送机)2_1、2_2、一个收集器1和一个负载(箱或盒)6的图像。图3是图2的图像中出现的元件的模型化视图(并且保持相同的附图标记)。

[0143] 5.1符号

[0144] 下表是所述符号的汇总。

[0145] [表1]

[0146]

k	系统的注入流（输入流）的数量
$a_i,$ $i \in \{1,\dots,k\}$	k 个通道（输送机），每个通道在给定时刻运输有序集合的负载
$h_i,$ $i \in \{1,\dots,k\}$	存在于通道（输送机）上的负载的数量 a_i
$n = \sum_{i=1\dots k} h_i$	要注入到收集器中的负载总数
L	要注入到收集器中的负载集合

[0147]

$u = a_i(j),$ $i \in \{1,\dots,k\},$ $j \in \{1,\dots,h_i\}$	来自通道a_i中的j^{th}位置的负载 $u \in L$
σ	期望的退出序列，对L的所有负载进行排序
$\sigma(u),$ $u \in L$	负载u在退出序列中的位置
σ_c	序列σ的c^{th}负载，属于L 设置 u = $a_i(j)$ to σ_c表示负载σ_c的u,i,j的计算
L'	置于定位于通道a_1与含有序列σ（即σ_1）的第一负载的通道之间的第一位置中的负载集合
$T_D(u),$ $u \in L$	负载 u的离开日期(与离开点一致)，即负载必须离开其通道(输送机）的时刻
$T(u),$ $u \in L$	负载 u 注入到收集器中的日期，与负载实际到达收集器的注入点一致
$Pt_i,$ $i \in \{1,\dots,k\}$	通道a_i的负载行进离开点与注入点之间的距离所需的时间
跨度	与分成时间跨度(也称为时间位置或时槽)的收集器的细分(或切分）相对应的时间单位
日期0	可以开始将负载注入到收集器中的日期（采取此收集器上第一未占用位置形式的日期）
日期t_0	期望的退出序列的第一负载经过通道a_k前面的日期
$\Delta_i,$ $i \in \{1,\dots,k\}$	通道a_i与通道a_k之间的“跨度”距离
$\Delta_{i,j},$ $i \in \{1,\dots,k-2\},$ $j \in \{1,\dots,k-1\}$	通道a_i与通道a_j之间的“跨度”距离

[0148] 5.1.1输送系统的通道上的符号

[0149] 对于给定时刻的系统配置,每个通道将包括有序集合的负载。k是系统中的通道数量,在收集器的移动方向上,每个通道被编号为a_1到a_k。这些通道分别具有由必须逐一注入到收集器中的先进先出(FIFO)列表表示的负载数量h_i。

[0150] 图4给出了实例。收集器1的移动方向是从右到左,此方向由箭头指示。每个通道均进行编号并且包括有序负载,所述有序负载将必须从下到上逐一注入到收集器中。因此,在此实例中,具有k=3,其中h_1=2,a_1的内容={id11,id12},h_2=3,a_2的内容={id21,id22,id23}并且h_3=4,a_3的内容={id31,id32,id33,id34}。

[0151] 5.1.2要注入的负载的符号

[0152] 总共有来自不同注入流的n个负载(箱、筐、容器等)要注入到收集器中。L表示这些负载的集合,并且u=a_i(j),负载u属于L,其来自通道a_i中的j^{th}位置。

[0153] 另外,σ是退出序列,其中一旦将这些负载全部注入到收集器中后就必须对其进行排序。函数σ(u)按此序列给出负载u的位置。因此,每个负载可以由1与n之间的单个“序列”号来标识,其与在期望退出序列(σ)中的位置相对应。这就是下文如何识别负载的方式。可以注意到,在此情况下,很容易验证序列是否可实现:对于每个通道而言,有一个有序列表的负载就足够了,所述负载由递增次序但不必是连续的序列号标识。

[0154] 现在指定集合L′,所述集合专门将最终退出序列中等待的第一负载分组在一起,放置于通道a_1、a_2、…、a_{i_1}的第一位置处,其中i_1是含有此序列σ的第一负载的通道号。

此集合将所有负载合在一起,所述负载可能是要注入到收集器中的序列的第一负载。实际上,放置于包括序列的第一负载的通道上游的通道中的第一位置中的负载可能首先被注入,而同时在收集器的最终流中定位于第一负载后面。然而,包括第一负载的通道下游的通道中的第一位置中的负载将永远不会在此第一负载进入收集器之前被注入。

[0155] 图5示出了在采用可实现序列σ={id21,id31,id22,id11,id23,id32,id33,id34,id12}方面与图4等效的图。在此实例中,n=9,因此,每个负载均具有使其能够被识别的唯一序列号,同时立即迅速地了解其在序列σ.中的期望位置。图5使得可能看到所选择序列是可实现的,实际上a_1的内容={4,9},a_2的内容={1,3,5},并且a_3的内容={2,6,7,8}..另外,还有特定集合$L_{i_1} = \{id11, id21\} = \{4,1\}$。

[0156] 5.1.3离开日期和注入日期

[0157] 可以回想起每个通道与给定时刻处的注入流相对应,表示为附接到收集器的FIFO列。然而,在仓库中,可能会发生的是,等待注入的负载可能会与收集器相距一定距离。离开点表示此负载等待命令离开其通道(输送机)的位置(由附接到FIFO列上的点展示),并且注入点表示在被真正注入此收集器的过程中,首先要被负载接触的收集器的空间,(由图6中附接到收集器的圆点展示)。

[0158] 在通道中的系统图中,这两个点(圆点)重合并且在图7中用黑色表示。实际上,这些点通常是不同的,并且将负载流i从一个点传递到另一个点所需的持续时间称为Pt_i(参见图6)。当负载必须离开其通道(输送机)时,所述负载的离开日期T_D通过以下简单公式与其注入收集器的日期T相关(与注入点重合):$T_D=T-Pt_i$。因此,在下文一般描述方面没有损失的情况下,我们将仅呈现注入日期的计算。

[0159] 5.2数据结构

[0160] 5.2.1细分排出系统(收集器):时间单位(时间跨度)

[0161] 排出系统(或收集器)将被认为是具有时间跨度(也称为槽或位置)的系统(如分拣器)。

[0162] 图8展示了收集器的时间跨度(位置或槽)的细分。

[0163] 因此,用有序输出流获得最大吞吐量的目的相当于填充收集器的跨度,如在图9中可以看到的。

[0164] 将考虑与此时间跨度相对应的时间单位,所述时间单位根据收集器的移动速度细分收集器。时间单位(称为“时间跨度”)与收集器的一个点精确地移动对应于一个位置的物理距离所需的持续时间相对应。针对此位置所定义的距离与负载的大小加上安全距离相对应。此安全距离必须根据本领域技术人员的需要进行调整并且以符合以下条件(展示于图10中)。

[0165] 时间单位必须强制性地大于从负载的一部分接触收集器到将整个负载准确地放置于收集器上(即在其流中)的将负载注入到收集器中所需的时间。

[0166] 5.2.2距注入系统的距离

[0167] 如图11所展示的,Δ_i被定义为与通道a_i与最后通道a_k之间的收集器(排出系统)的时间跨度数相对应,如在附图中可以看到的,其中k=3。如图12所展示的,此数字可能不是整数,$\Delta_k=0$。

[0168] 两个通道a_i与a_j之间的时间距离表示为$\Delta_{i,j}$,第二指数是相对于其所定位的通道。

[0169] 5.2.3相关联作业车间模型

[0170] 下表是作业车间模型的符号汇总

[0171] [表2]

[0172]

$\{J_u, u \in \{1,\dots,n\}\}$	与作业车间模型相关联的n个作业的集合。 序列σ的u^{th}负载表示为$\sigma_u = a_i(j)$，占据通道a_i的j^{th}位置并且与作业J_u相关联
$\{M_1,\dots,M_k\}$	作业车间模型的k个机器的集合。 通道a_i与机器M_i相关联。
$\{o_{u,i}, o_{u,\{i+1\}},\dots,o_{u,k}\}$ $u \in \{1,\dots,n\}$, $\sigma_u = a_i(j)$	与作业J_u相关联的$k-i$连续操作列表，已知操作$o_{u,i}$必须在机器M_i上进行。

[0173] 对于每个退出序列σ,将系统建模为具有n个作业和k个机器的单位作业作业车间模型。序列σ_u(此序列的第u个负载)的每个负载均与作业J_u相关联,并且每个通道a_i均与机器M_i相关联。与通道a_i中含有的负载相关联的每个作业J_u必须遵循k-i+1单元操作的有序列表$\{o_{u,i}, o_{u,\{i+1\}},\dots,o_{u,k}\}$。表示为$o_{u,i}$的操作是要在机器$M_i$上具体执行的作业$J_u$的单位作业。作业一次只能分配给一台机器,并且机器一次只能执行一项操作。

[0174] 将尝试按照其相应的操作列表给出的序列次序在机器上调度这些作业。目的是对其进行调度以使所有这些作业的执行总持续时间最小化。

[0175] 6.针对仅处理注入流的收集器,确定每个负载的注入日期

[0176] 在此部分中,将考虑仅处理注入流的收集器。收集器完全是空的并且准备检索注入流。不过这些注入流的合并必须被控制以精确地注入其负载,以便具有流体的、连续的以及与期望的退出序列相对应的最终流。

[0177] 6.1针对连续分布良好的通道的解法

[0178] 为了简化问题,将假设注入流在收集器上连续分布的方式为使得收集器上的负载需要时间间隔以从一个注入区传递到下一个注入区,如图13中可以看到的。此简化情况与$\Delta_{i,i+1}=1$,相对应,针对所有$i \in \{1,\dots,k-1\}$。

[0179] 6.1.1将系统建模为作业车间调度

[0180] 在此特定情况下,将系统的建模解释为作业车间调度的难题。

[0181] n个负载中的每个负载均与工作J_u,$u \in \{1,\dots,n\}$相关联,其中根据负载的序列号进行编号。因此,如果σ_u与序列σ中的第u个负载相对应,则其相关联的作业为J_u。这些作业的操作必须由k个计算机的集合$\{M_1,\dots,M_k\}$处理。操作一次只能分配给一台机器。每台机器均与一次只能执行一个操作的通道(即注入流)相关联。可以回想起,这些通道是从上游到下游(即根据收集器的输送方向)进行编号$\{a_1,\dots,a_k\}$的。

[0182] 因此,从通道a_i注入到收集器中的负载σ_u将经过每个通道a_j的前面,其中j处于$\{i, i+1,\dots,k\}$中。此机制由以下事实表示:与通道a_i的负载相关联的每个作业均由要在不等待的情况下连续处理的k-i+1单元操作$\{o_{u,i}, o_{u,\{i+1\}},\dots,o_{u,k}\}$构成。每个操作均必须在特定机器上进行调度。因此,如果(由机器M_i处理的)操作$o_{u,i}$在时刻t开始,则(由机器$M_{\{i+1\}}$处理的)操作$o_{u,\{i+1\}}$在时刻t+1等开始,并且最终(由机器M_k处理的)操作$o_{u,k}$在时刻t+k-i处开始。

[0183] 如果此作业的第一操作$o_{u,i}$的开始时刻与注入收集器的日期相对应,则后续操作

(c.-à-d.{$o_{u,\{i+1\}},..,o_{u,k}$})的开始时刻表示负载位于后续通道前面的时刻。

[0184] 可以注意到,每个作业均由在最后一台机器M_k上的至少一个操作构成(因为每个负载均经过至少最后通道a_k的前面)。在机器M_k上进行调度的操作序列恰好与负载将经过最后的通道前面的次序相对应,并且其离开日期与这些负载将经过此最后的注入通道前面的时刻相对应。这就是为什么如果负载u在给定退出序列中在负载v之前,则u的最后操作必须在M_k上的v的最后操作之前开始。

[0185] 在收集器上没有空间的目的等于在机器M_k的操作之间没有空闲时间。

[0186] 6.1.2算法的原理使能够获得注入日期

[0187] 上文提出的调度问题可以通过以下算法解决。

[0188] 首先,假设序列的第一负载来自通道a_i,,与此作业相关联的操作{$o_{1,i}$,$o_{1,\{i+1\}}$,...,$o_{1,k}$}在时刻t、t+1、...t+k-i开始。

[0189] 然后,将首先通过在机器M_k上执行要处理的操作来调度与所述序列的下一负载相对应的作业的操作。此操作$o_{u,k}$将在先前在M_k上编程的操作之后立即进行调度(没有空闲时间)。假设此操作在时刻t开始,则在时刻t-1对机器$M_{\{k-1\}}$上的操作(如果存在的话)进行调度,并且依此类推,直到对所有操作进行编程。

[0190] 此程序将迭代地应用于给定序列的以下作业中的每个作业。

[0191] 直接从每个作业的第一操作开始推断出注入到收集器中的实际日期。所使用的时间单位是上文定义的时间单位(收集器的“时间跨度”)。

[0192] 6.1.3通过公式计算注入日期

[0193] 现在已经描述了解决调度问题的算法如何工作,将推导用于计算注入日期的公式。

[0194] 下文的公式中所使用的符号是上文进一步讨论的汇总表的符号。

[0195] 将最早的一个或多个负载注入到收集器中的最早日期确定为时间起点。因此,在本系统中,将在时刻0将要接收此负载的空闲位置放置在此负载的通道前面。因此,将计算与序列的第一负载经过通道a_k.前面的最早日期相对应的日期$t_0 = \max_{u=a_i(j)\in L',1\le i\le k,1\le j\le h_i}\{k+1-i-\sigma(u)\}$。

[0196] 证明:0是开始注入负载的日期。t_0是序列的第一负载(即σ_1)经过通道a_k前面的日期。注意到注入到收集器中的第一负载必定是定位于从通道a_1到σ_1的通道的通道中的通道项部的负载。$L' \subset L$是此负载的子集。

[0197] 因此,对于任何负载u=a_i(j)∈L′,要知道,如果要在日期x处注入,其将在日期x+k-i处经过通道a_k前面。其中x≥0,因为无法在日期0之前注入负载。现在,根据t_0的定义,负载u将在日期t_0+σ(u)-1=x+k-i处经过通道a_k前面。即,t_0+σ(u)-1-k+i=x≥0。

[0198] 据此,针对所有得出u=a_i(j)∈L′,t_0≥k+1-i-σ(u)。搜索最早的日期,使能够验证所有这些约束,因此:

[0199] $$t_0 = \max_{u=a_i(j)\in L'}\{k+1-i-\sigma(u)\}$$

[0200] 下文的算法1给出了注入日期T(u),即每个负载u进入收集器的日期。所使用的时间单位是上文进一步定义的“时间跨度”。如果流在收集器上的每个连续位置处注入,则算法1给出用于计算这些注入日期的公式(请参见图14)。

[0201] 算法1:在没有干扰流的情况下在一种特定情况下计算注入日期

$$\textbf{Require}: \sigma, k, L', a_i(j), h_i, \forall i \in \{1, \dots, k\}, \forall j \in \{1, \dots, h_i\}$$

$$t_0 = \max_{u=a_i(j)\in L'} \{k + 1 - i - \sigma(u)\}$$

$$\textbf{for all } i \in \{1, \dots, k\} \textbf{do}$$

$$\quad \textbf{for all } j \in \{1, \dots, h_i\} \textbf{do}$$

[0202] 设置$u = a_i(j)$

$$T(u) = t_0 + \sigma(u) - 1 - (k - i)$$

$$\quad \textbf{end for}$$

$$\textbf{end for}$$

$$\textbf{return } T(u), \forall u \in L$$

[0203] 6.1.4实例

[0204] 在调度中,甘特图是用于及时查看构成作业的操作的非常有用的工具。此工具将使能够以图形方式表示每台机器的工作进度并且以可视方式示出了提供给问题的解法。另外,可以看到通过使能够获得注入日期的算法获得的结果与相关联公式之间的联系。

[0205] 如下进行系统配置(参见图14):k=4包括两个或三个负载的通道。每个负载借助于退出序列σ=(1,2,3,4,5,6,7,8,9).由唯一的序列号标识。这是可实现序列。

[0206] 在此简化情况下,我们可以在甘特图中表示作业车间解法,使得每一行对应于一台机器。每个负载连续经过机器的事实由每个负载序列号出现在表中从左到右的连续框中的事实表示(参见图15)。另外,每个负载的首次出现均指示将时间标度锁定在上文进一步定义的时刻0时由公式给出的注入日期。可以回想起,日期0与第一可能的负载注入的时刻相对应。甚至可以根据其定义放置日期t_0并且通过计算找到所述日期。回想一下,t_0是序列(σ_1)的第一负载经过通道k=4前面的日期,即当机器M_4执行作业1的操作时的编号1。

[0207] 在机器M_4上找到了没有未用空间的期望序列,并且通过给出此负载的标识的序列号首次出现的框(从左到右)的读数来给出每个负载的注入日期。

[0208] 6.2具有任何未指定分布的通道的解法

[0209] 现在已经看到了简化情况,考虑仅处理注入流的收集器的一般情况。因此,流不再在收集器的每个时间跨度处连续注入,而是以任何未指定方式分布。通道之间的距离是已知的。特别感兴趣的是通道a_i的注入点与最后通道a_k之间的行进时间。将表示为$\Delta_i = \Delta_{i,k}$(如上文进一步定义的)。

[0210] 6.2.1通过公式计算注入日期

[0211] 下文的算法2考虑了通道之间任何未指定的时间距离的概念,并且因此对专用于注入流的收集器的一般情况进行响应。

[0212] 再次将日期0视为第一可能注入日期。这次,在日期x处注入的通道i的负载u在日期$x+\Delta_i$处到达通道a_k前面。因此,通过以下公式获得序列的第一负载经过通道a_k前面的最早日期:$t_0 = \max_{u=a_i(j)\in L'}\{\Delta_i + 1 - \sigma(u)\}$.。

[0213] 算法2:在没有干扰流的情况下在一般情况下计算注入日期

$$\textbf{Require}: \sigma, k, L', a_i(j), \Delta_i, h_i, \forall i \in \{1, \dots, k\}, \forall j \in \{1, \dots, h_i\}$$

$$t_0 = \max_{u=a_i(j)\in L'} \{\Delta_i + 1 - \sigma(u)\}$$

$$\textbf{for all } i \in \{1, \dots, k\} \textbf{do}$$

$$\quad \textbf{for all } j \in \{1, \dots, h_i\} \textbf{do}$$

[0214] 设置 $u = a_i(j)$

$$T(u) = t_0 + \sigma(u) - 1 - \Delta_i$$

$$\quad \textbf{end for}$$

$$\textbf{end for}$$

$$\textbf{return } T(u), \forall u \in L$$

[0215] 图46呈现了根据基于算法2的第一特定实施例的用于将k个输入流合并成一个输出流的方法的流程图。所述方法由控制系统执行至少一次。给定执行在时刻Tb处执行,并且包括以下步骤:

[0216] -获得(步骤461)集合L,所述集合L包括在所述时刻Tb处分布在所述k个通道上并且必须注入到所述收集器中以形成退出序列σ的n个有效负载,所述n个有效负载中的每个有效负载均由所述退出序列σ内的单个序列号标识,所述k个通道中的每个通道均含有必须逐个注入到所述收集器中的h_i个有效负载的有序集合,所述有序集合按序列号的上升次序进行排序;

[0217] -计算(步骤462)日期t_0,在所述日期处,所述退出序列σ中的第一有效负载σ_1在通道a_k前面经过;

[0218] -根据以下公式计算(步骤463)将所述集合L的所述n个有效负载中的每个有效负载注入所述收集器上的日期:$T(u)=t_0+\sigma(u)-1-\Delta_i$,其中:

[0219] • $u=a_i(j)\in L$,所述集合L的有效负载并且来自通道a_i的第j位置,$i\in\{1,\dots,k\}$,$j\in\{1,\dots,h_i\}$,

[0220] • σ(u)是所述退出序列σ中的所述有效负载u的所述序列号;

[0221] -命令(步骤464)所述收集器和所述k个通道以与n个注入日期T(u),$\forall u \in L$一致地将所述n个有效负载注入到所述收集器中。

[0222] 6.2.2实例

[0223] 这次以类似于第6.1.4段的实例为例留出通道。

[0224] 如下进行系统配置(参见图16):k=4包括两个或三个负载的通道。每个负载根据退出序列σ=(1,2,3,4,5,6,7,8,9)由唯一的序列号标识。这是可实现序列。已知从每个通道到最后通道的距离:$\Delta_1=12$,$\Delta_2=7$,$\Delta_3=4$并且$\Delta_4=0$。

[0225] 图17以甘特图的形式给出了解法的表示。这次,两台连续机器之间的行间隔必须与两个对应的连续通道之间的"时间跨度"一样多。这就是为什么此解法由具有13(=Δ_1+1)行的表给出的原因。通过计算,得到$t_0=\max_{u\in\{1,2\}}\{\Delta_i+1-\sigma(u)\}=\max(7,11)=11$,实际上可以通过定义在图上的时刻11找到。

[0226] 此处,为了便于理解,给出了通道间距离为整数值的实例,但是采用多个"时间跨度"的非整数值是可能的。

[0227] 6.3一个特定实施例的描述

[0228] 6.3.1要分组在一起的元件列表

[0229] 每个注入流(输入流)由含有FIFO次序的负载列表的通道表示。通道数量可以从2到k变化,并且通道列表i含有h_i负载(h_i可以为零)。总共有要合并到收集器中的n个负载。举例来说(参见图18):$k=2, h_1=h_2=5$并且$n=10$。

[0230] 另外,与一般情况相同,将用$\Delta_1=1$并且$\Delta_2=0$.采用此一般情况的解析。

[0231] 6.3.2创建默认序列

[0232] 当收集器上的负载的最终序列不重要时,以任何未指定方式定义设置负载次序的可实现序列就已足够。

[0233] 可以设计不同的方式来定义“默认”退出序列:

[0234] -轮流从每个通道移除一个负载,$\sigma=(id_5,id_{10},id_4,id_9,id_3,id_8,\ldots,id_6)$

[0235] -根据负载到达注入流的情况来将其移除,

[0236] -从随机绘制的每个通道上移除第一负载,

[0237] -等。

[0238] 6.3.3由无序最小化计算给出的序列

[0239] 图19展示了由无序最小化计算产生的序列($\sigma'=(8,4,5,2,7,6,1,9,3,10)$)的实例。所指示的数字不是最终序列内的序列号,而是相对于要拣选的客户订单的序列订单号。

[0240] 此序列σ'可以被视为用于计算注入日期的算法的给定退出序列。此序列是强制性可实现的并且使无序最小化。对于来自收集器的输出流,获得了最大吞吐量和最小无序化。

[0241] 由于始终可以为所有负载分配从1到n的单个序列号,所述单个序列号与其在最终序列σ中的位置相对应,因此将显示带有这些序列号的负载以对其进行标识并完全表征(参见图20)。图19和20是等效的。退出序列$\sigma=(1,2,3,4,5,6,7,8,9,10)$确实使得有可能看到每个注入流上负载的上升次序(序列的可行性证明)。

[0242] 保留此配置以继续实例的研究。

[0243] 6.3.4 t_0的创建

[0244] 在本发明实例中,L′将编号为1和5的负载汇总在一起。

[0245] 搜索$\{\Delta_2+1-\sigma(1),\Delta_1+1-\sigma(5)\}$之间的最大值,并且获得$t_0=0$。

[0246] 6.3.5注入日期的计算(通过公式)

[0247] $T(1)=t_0+\sigma(1)-1-\Delta_2=0$

[0248] $T(2)=t_0+\sigma(2)-1-\Delta_2=1$

[0249] $T(3)=t_0+\sigma(3)-1-\Delta_2=2$

[0250] $T(4)=t_0+\sigma(4)-1-\Delta_2=3$

[0251] $T(5)=t_0+\sigma(5)-1-\Delta_1=3$

[0252] $T(6)=t_0+\sigma(6)-1-\Delta_1=4$

[0253] $T(7)=t_0+\sigma(7)-1-\Delta_1=5$

[0254] $T(8)=t_0+\sigma(8)-1-\Delta_1=6$

[0255] $T(9)=t_0+\sigma(9)-1-\Delta_1=7$

[0256] $T(10)=t_0+\sigma(10)-1-\Delta_2=9$

[0257] 6.3.6图像中的结果说明

[0258] 注入日期T(u)以时间单位(即收集器的时间跨度)给出,这在视觉上相当于看到收集器在每个时间单位上移动一个跨度。如果在每个时间单位查看系统的图像,则每个负载u均会在日期T(u)(其注入日期)处首次出现在收集器上。

[0259] 收集器在时刻0处的配置(参见图21)与负载的第一实际注入相对应,在此情况下为负载1(因为$t_1=0$),即使并非总是如此。必须指出的是,实际上所述序列的第一负载并不是要强制性地注入到收集器中的第一负载。

[0260] 如图22所示,负载2必须在时刻1处注入,这意指在移动收集器一步之后所述负载便在收集器上。

[0261] 如图23所示,负载3在时刻2处注入。

[0262] 如图24所示,负载4和5在日期3处注入。

[0263] 并且依此类推。

[0264] 7.批次的进展

[0265] 注入日期的计算是在系统的给定状态下完成的。随着系统的适时开发,第一种可能性在于,每当新的负载到达系统时就重新进行此计算,因为其尚未拥有确定注入日期。然而,代替同样定期启动由于各种原因(具体地是计算资源的消耗)的计算算法是不利的,分批次进行是优选的。本节将解释这种分批操作。

[0266] 7.1工作批次

[0267] 希望收集器上的流(输出流)在整个工作日内连续且最佳。已经看到了如何在给定的固定时刻定义存在于通道中的负载的最佳注入日期(以达到目标)。现在,对于在任何给定时刻注入流到达的所有负载,均需要实现此目标。建议在获得收集器的流体和连续填充所需的时刻处调用注入日期计算算法。

[0268] 为此,在本节中,定义了在固定时刻填充用于系统配置的每个通道的FIFO列表的方法。然后,定义批次是什么,并且然后最终解释两个批次的这种连续处理,而不会在收集器上留出任何未用空间。

[0269] 7.1.1填充通道的FIFO列表

[0270] 在每个通道中,均有负载列表,其中每个负载均具有唯一序列号。所述负载的这些序列号是强制性数字,所述数字按通道上升,但不一定是连续的(由于要给出可实现序列的定义)。

[0271] 图25展示了与三个通道相关联的FIFO列表的实例。

[0272] 图26展示了收集器上等待的最终可实现序列。

[0273] 可以回想起,每个通道与表示为附接到收集器的FIFO列的注入流相对应,其中其注入点由图中的黑点定义。

[0274] 针对系统固定配置的通道中考虑的负载的FIFO列表根据以下条件设置了在此通道的注入点等待的负载的次序:

[0275] • 在分析期间考虑的负载在注入点等待(下图中用黑点表示)(图27:附接到注入点的流的第一负载并且因此形成FIFO列表的一部分;图28:未附接到注入点的流的第一负载并且因此未形成FIFO列表的一部分);

[0276] • 分析期间考虑的负载“连续”保持在形成FIFO列表的一部分的第一负载后面(图29:形成FIFO列表的一部分的第一负载后面的连续保持的第二负载;图30:未形成FIFO列表

的一部分的第一负载后面的连续保持负载)。

[0277] 7.1.2批次的定义

[0278] 工作批次与所有通道的FIFO列表的负载集合(其是n个负载的集合L)相对应。然而,这些负载在给定时刻与系统的静态照片相对应,从而符合先前段落的规则。此时存在于系统中的不符合FIFO列表填充规则的负载随后将分配给另一批次。

[0279] 一旦批次的负载得以分配,就可以计算此批次的注入日期。这些负载的注入在时间流逝时完成,从而符合预先计算的注入日期。尽管新负载在FIFO列表的填充规则下或即使新负载出现在系统中也符合条件,但是系统会在未再次调用注入日期计算算法的情况下随着时间的推移不断发展。

[0280] 计算所有这些注入日期以成功地将批次的每个负载置于输送机上的保留位置,从而符合期望最终序列的次序,同时使未用空间最小化。使p为由所述序列的次序给出的此批次的最后负载指定保留的最后位置。在此地方p之后,收集器空闲。这是此最后位置p(仍可以是空闲的)经过将定义新批次的第一注入流(第一通道)以下的时间。

[0281] 后续批次将以同一方式创建,但是会在此固定时刻在系统的新静态照片上创建(作为第一未占用位置,将收集器的跨度附接到此最后位置p)。

[0282] 必须注意,在一个特定实施方案中,如果在完全执行先前批次之前定义新批次,则可以针对此新批次(以下批次)考虑先前批次的退出序列的最后负载。在此情况下,如果将退出序列中位置i处的负载放回到下一批次中,则所有位置大于i)的负载也将放回下一批次中。然而,只有在期望的最终序列已经改变的情况下,才值得再次占据负载。在此情况下,考虑的最后位置p将是下一批次中未被占据的最后负载的位置。对于所占据的负载,先前批次中已经定义的注入日期将被淘汰并且由针对下一批次计算的注入日期代替。

[0283] 在图31展示的实例中,第一批次将由以下负载构成:1、2、3、4、5、6、7、8、9。以下批次可能是:xx、yy或9、xx、yy、...或再次7、8、9、xx、yy、...(但不是8、xx、yy或7、9、xx、yy)。

[0284] 7.2批次的动态处理

[0285] 每个批次的处理均是持续且连续的(批次1、批次2等)。批次的处理是对构成此批次的负载的所有注入日期的静态计算。这些批次的链接和创建为在仓库中将输入流合并成一个输出流的问题(这是动态问题)提供了全面的解法。

[0286] 7.2.1第一批次的注入日期的计算

[0287] 图32展示了退出序列σ=(1,2,3,4,5,6,7,8,9,10)的第一批次的实例,其中负载2、6、7和8(其注入日期分别表示为时间4、时间8、时间9和时间10)位于通道a_3上;负载1、3和5(其注入日期分别表示为时间0、时间2和时间4)位于通道a_2上并且负载4和9(其注入日期分别表示为时间0和时间5)位于通道a_1上。

[0288] 7.2.2批次的注入时间

[0289] 只要等待最高序列号的收集器的时间跨度(在此情况下为n为9,在图32和33中用圆圈包围)没有到达最上游注入通道(即通道a_1)的前面,就不对下一批次进行处理。随着时间的流逝,继续处理第一批次的负载,这些负载在预先计算的注入日期被注入,如图33的上部和下部可以看到的。

[0290] 7.2.3下一批次的形成和其注入的未来日期的静态计算

[0291] 当第一批次的序列的最后负载的最后标记的跨度在通道a_1之前到达时,其是用于

创建下一批次的信号。根据上文进一步解释的规则,此批次由每个通道的每个FIFO列表的所有有效负载构成。

[0292] 图34展示了下一批次的形成,所述下一批次由出现在通道上并且被编号为1到6(新批次期望的新退出序列的序列号)的六个新负载构成。

[0293] 考虑到收集器的未占用部分(在为先前批次中编号为9的负载保留的位置的右侧),用于计算静态注入日期的算法将应用于此新批次。注入日期是根据时间标度来计算的,所述时间标度的起点与新批次的负载的第一注入相对应。

[0294] 如图34中可以看到的,将在"时间5"处获取的静态系统的配置(作为输入数据)给予所述算法。此配置定义如下:k=3包括已经出现的六个新负载(属于新批次)的通道。每个新负载根据此新批次的新退出序列由唯一的序列号标识:$\sigma=(1,2,3,4,5,6)$。这是可实现序列。已知相对于最后通道的时间距离:$\Delta_1=6$,$\Delta_2=3$并且$\Delta_3=0$。

[0295] 通过同时注入两个编号为1和4的新负载(分别进入通道a_2和a_1)来固定日期0。由于此时间起点,所述算法为此新批次的六个新负载u中的每个新负载给出注入日期T(u)。

[0296] $L'=\{1,4\}$并且$t_0=\max\{\Delta_1+1-\sigma(4),\Delta_2+1-\sigma(1)\}=\max\{3,3\}=3$

[0297] $T(1)=t_0+\sigma(1)-1-\Delta_2=0$

[0298] $T(2)=t_0+\sigma(2)-1-\Delta_2=1$

[0299] $T(3)=t_0+\sigma(3)-1-\Delta_3=5$

[0300] $T(4)=t_0+\sigma(4)-1-\Delta_1=0$

[0301] $T(5)=t_0+\sigma(5)-1-\Delta_2=4$

[0302] $T(6)=t_0+\sigma(6)-1-\Delta_1=2$

[0303] 7.2.4批次的注入日期与流逝的实时时间之间的链接

[0304] 每个批次的每个负载均将根据相对于静态系统明确指定的时间起点来给出注入日期。这些注入日期必须置于动态配置中的正确时刻以与动态解法相一致。

[0305] 所有批次共有的时间标度使将所有负载的注入链接到动态系统中成为可能。如上文进一步定义的,此时间标度的时间单位是收集器的"时间跨度"。"时间"是源自此共有时间标度的时间,其中时间0表示系统中操作的开始日期。

[0306] 由算法给出的注入日期必须在此共有时间标度内正确地重新定位。为此,有必要在动态系统中找到批次的日期0处所描述的情况的时间相关对应关系。这意指有必要获得批次的日期0与真正启用此批次的第一注入的共有标度的时间之间的对应关系。然后,这次有必要准确地转换此批次的所有注入日期。

[0307] 为此,有必要知道已经启动算法调用(也称为"Tb")以及最早要注入收集器上的批次的一个或多个第一负载(或其中的至少一个负载)的时间,这将被称为σ_*。批次的日期0与注入此负载的动作相对应,$\sigma_*=a_i(j)$,并且其实际上是由动态系统在调用之后的确切时间跨度$\sigma(\sigma_*)+\Delta_{1,*}$内完成的。

[0308] 返回本发明实例,有必要为由算法计算的每个注入日期添加新批次(对于所有$tu\in L,T(u)$),在此情况下实际日期(算法调用日期)为时间5(即Tb=5)加上注入到新序列中的新批次的第一注入负载的位置(在此情况下为负载1,因此需要添加$\sigma(1)=1$)加上通道a_1与含有此第一注入负载的通道之间的距离(在此情况下$\Delta_{1,2}=3$)。要注意,新批次的负载4也首先(与新批次的负载1同时)注入到收集器中,可以选择并且添加$\sigma(4)+\Delta_{1,1}=4$;这实际

上相当于添加$\sigma(1)+\Delta_{1,2}=1+3=4$。

[0309] 对于动态应用,对注入时间执行如下的线性变换:

[0310] $T(1)+5+4=0+5+4=9$

[0311] $T(2)+5+4=1+5+4=10$

[0312] $T(3)+5+4=5+5+4=14$

[0313] $T(4)+5+4=0+5+4=9$

[0314] $T(5)+5+4=4+5+4=13$

[0315] $T(6)+5+4=2+5+4=11$

[0316] 在此情况下,在a_k前面经过的新序列的第一负载的最早日期将准确地为$\mathrm{Tb}+\sigma(u_*)+\Delta_1-\Delta_{i_*}+t_0$,其中$u_*=a_{i_*}(1)$,新序列的第一负载注入到收集器中(不一定是新序列的第一负载)。现在,已知t_0正是由此负载u_*获得的最大值,从而得出$t_0=\max_{u=a_i(j)\in L'}\{\Delta_i+1-\sigma(u)\}=\Delta_{i_*}+1-\sigma(u_*)$。因此,已知新序列的第一负载将在共有时间标度在$\mathrm{Tb}+\sigma(u_*)+\Delta_1-\Delta_{i_*}+t_0=\mathrm{Tb}+\sigma(u_*)+\Delta_1-\Delta_{i_*}+\Delta_{i_*}+1-\sigma(u_*)=Tb+1+\Delta_1$处在$a_k$前面经过。

[0317] 以共有时间标度给出注入时间的等效方法是在时刻$Tb=T(u_{last})-\Delta_1+\Delta_{i^{last}}$(其中$u_{last}=a_{i^{last}}(j^{last})$是先前时刻的序列的最后负载)处精确地获得系统知识时按照定义给出算法$t_0=\mathrm{Tb}+1+\Delta_1$。

[0318] 在上文的实例中,所述时刻$\mathrm{Tb}=5$恰好与条件相对应,并且因此可以调用所述算法,其中$t_0=\mathrm{Tb}+1+\Delta_1=5+1+6=12$(而不是3),并且可以直接拥有以共有时间标度表示的注入时间。

[0319] 最终,系统可以如图35所示的进行表示。在共有时间标度中:

[0320] • 先前批次的负载6、7和8的注入时间分别表示为时间8、时间9和时间10;并且

[0321] • 新批次的负载1、2、3、4、5和6的注入时间分别表示为时间9、时间10、时间14、时间9、时间13和时间11。

[0322] 8.针对处理一个或多个干扰流的收集器确定负载的注入日期

[0323] 假设现在(干扰负载的)至少一个干扰流已经存在于收集器中。这些干扰负载不是注入流(输入流)的一部分。因此,其未按注入流的期望退出序列进行排序。这些干扰负载会干扰注入流中含有的有效负载的引入,因为其占据了收集器上的位置。有必要考虑这些不时阻止有效负载注入的阻碍物。另外,如果这些干扰负载从注入通道之间的收集器中消失,则可能在来自收集器的退出流中创建未用空间。目标始终是遵循退出序列,以使收集器的吞吐量最大化,并且因此使这些未用空间最小化。

[0324] 为了实现此目标,将使用无效的日期列表以通过上文进一步看到的作业车间建模来加以利用。

[0325] 下表给出了后续部分所需的新符号(干扰流的符号汇总)。

[0326] 下表给出了干扰流的新符号。

[0327] [表3]

[0328]

F	干扰流
n'	干扰负载总数
$l(p)$或 l_p, $p \in F$	干扰负载p在其前面经过的最后通道的编号
$g(p)$或 g_p, $p \in F$	干扰负载p在其前面经过的第一通道的编号
U_i, $i \in \{1,\ldots,k\}$	通道a_i注入的有效负载的注入禁止日期列表
σ'	干扰负载的有序列表
σ'_p	干扰流的p^{th}干扰负载

[0329]

σ'(u,v)	干扰负载u与干扰流中的干扰负载v之间的持续时间，以给定时刻的“跨度”数表示
t_r	干扰流的第一干扰负载到达通道 a_1前面的日期，在此特定情况下此负载此通道前面有效地经过（与δ_p) 有关
δ_p	在日期0处干扰负载p与最后通道 a_k之间的距离
$start_p$	在通道a_1之前“假设地”注入干扰负载p 的日期。（假设如果 $g_p > 1$，因为在此情况下，负载p将不会在通道a_1)之前经过

[0330] 8.1具有不受控干扰流的系统的建模

[0331] 当其在注入通道前面经过时,干扰流的干扰负载会阻止将有效负载注入到收集器中。因此,有必要计算对于每个通道的有效负载注入无效的无效日期。现在,考虑到不受控干扰流(根据定义,假设对其没有任何控制),计算有效负载的新注入日期。注意到,此后退出流中可能会存在未占用空间,但仍将寻求这些未占用空间的最佳最小化。

[0332] 8.1.1无效日期列表的创建

[0333] 可以考虑包括保留在收集器上的干扰负载的不受控干扰流,从而在所有通道前面经过并且形成收集器的最终流的一部分(与注入流的有效负载混合)。还可以考虑包括存在于收集器上但在注入通道之间消失的干扰负载的不受控干扰流,从而在某些通道而不是所有通道前面创建未占用位置。还可以考虑这些类型的不受控干扰流的混合情况以及在通道之间到达和分布的干扰负载。一切都可以设想,并且甚至可以考虑在收集器的一个或多个跨度上的未指定阻碍物(阻碍物不是强制性干扰负载,但也能够是收集器被损坏、保留的地方等)。

[0334] 在任何情况下,所述想法是考虑干扰负载和/或其它阻碍物以计算对于通道的有效负载的注入无效的日期。无效日期通过计算每个阻碍物(即干扰流的每个干扰负载或其它任何阻碍物)在此注入通道前面经过的时刻来获得。U_i表示无效日期的集合,其中因为干扰负载和/或收集器的时间跨度的任何未指定阻碍物,所以通道a_i无法注入有效负载。

[0335] 8.1.2针对连续分布良好的通道的解法的想法

[0336] 可以通过遵循与第6.2.1段的步骤几乎相同的步骤来解决此作业车间调度问题(针对每台机器具有无效日期列表)。

[0337] 这是在通道通过输送机跨度分布良好的特定情况下的方法。

[0338] 首先,假设序列的第一负载来自通道a_i,与此作业相关联的操作$\{o_{1,i}, o_{1,\{i+1\}}, \ldots, o_{1,k}\}$在从日期t开始的日期t、t+1、...t+k-i开始,从而使在每台机器$\{M_i, M_{\{i+1\}}, \ldots, M_k\}$上分别拥有空闲(即无阻碍物)日期t、t+1、...、t+k-i成为可能。

[0339] 然后,将首先通过在机器M_k上执行要处理的操作来调度与所述序列的下一负载相

对应的作业的操作。此操作$o_{u,k}$将在先前在M_k上编程的操作之后立即(没有空闲时间)进行调度。假设此操作在新的时刻t开始(与上文针对与第一负载相关的计算而进一步定义的时刻t不同),则在时刻t-1处对机器$M_{\{k-1\}}$上的操作(如果存在的话)进行调度,并且依此类推,直到对所有操作进行编程。如果由于阻碍物而无法在指示日期处执行操作,则将开始调度(整个)作业以转换要在时间单位(即位置、跨度或槽)的机器M_k上处理的操作的执行时刻t。

[0340] 此程序将迭代地应用于给定序列的以下作业中的每个作业。

[0341] 直接从每个作业的第一操作开始推断出有效负载注入收集器上的实际日期。所使用的时间单位是上文进一步定义的时间单位。

[0342] 注意:此方法并不总是给出没有未用空间的流。干扰流和/或阻碍物不受控,并且因此在不确保极大性的情况下使吞吐量最大化。

[0343] 8.1.3不受控干扰流在一般情况下的解法的想法

[0344] 基于以上段落,这是一般情况下用来计算包括输入流(注入流)和(至少)一个不受控干扰流的若干个流的合并期间的实际注入日期的方法。

[0345] 考虑可以从时刻0处向前对作业进行调度开始。必须牢记的是,在执行机器Mk的操作时必须遵循特定的序列次序,同时最小化此机器的空闲时间。

[0346] 首先,t_0(≥0)表示最早的日期,与序列的第一负载相关联的作业可以在所述日期处分配给机器M_k(如果其没有干扰流)。在不考虑干扰流的情况下,按期望序列给出的次序对负载进行调度,并且然后遵循此次序修改注入日期。实际上,可以注意到,负载σ_c的调度仅对负载$\sigma_{c'}$有影响,使得c′>c。

[0347] 对于与负载u相关联的作业,首先对其要在机器M_k上处理的操作进行调度。如果其是负载σ_1,则此操作$o_{u,k}$将在t_0处进行调度,否则将在先前在M_k上编程的操作之后立即(没有空闲时间)进行调度。假设在M_k上的此操作在时刻t处开始,则在时刻$t-\Delta_{k-1}$处对机器$M_{\{k-1\}}$上的操作(如果存在的话)进行调度。然后在时刻$t-\Delta_{k-2}$处对机器$M_{\{k-2\}}$上的操作(如果存在的话)进行调度,并且依此类推,直到对所有操作进行编程。如果在阻碍物期间所指示的这些日期处无法执行操作,则将重新开始调度(整个)作业以转换要通过时间单位(即通过位置、跨度或槽)在机器M_k上处理的操作的执行时刻t。

[0348] 此程序将迭代地在给定序列的以下作业中的每个作业上应用。

[0349] 如有必要,需要将日期0和日期t_0转换成与其定义相对应的日期。由于日期0的转换必须与收集器上的第一可能的负载注入相对应,因此其对将依次转换的所有调度(注入日期)均有影响。

[0350] 直接从每个作业的第一操作开始推断出有效负载注入到收集器中的实际日期。所使用的时间单位是上文进一步定义的时间单位。

[0351] 注意:不受控干扰流的存在并不总是能够使流在收集器上没有未用空间。用此方法,在不保证极大性的情况下使收集器的吞吐量最大化。

[0352] 8.1.4一般情况下的算法

[0353] 考虑到干扰流采取禁止通过通道注入的日期的形式,下文的算法3计算有效负载的注入日期。

[0354] 日期0与最早注入到收集器中的第一负载相对应。日期t_0始终是序列的第一负载在通道a_k前面经过的日期。如果在注入序列的第一负载期间不存在与干扰流的冲突,则通

用公式$i = 1 \ldots k: t_0 = \max_{u=a_i(j)\in L', j\le h_i}\{\Delta_i + 1 - \sigma(u)\}$仍然有效。

[0355] 考虑到干扰流并且在必要时校正日期t_0,算法3给出每个负载的注入日期。通过对集合U_i,i=1...k的初步计算来对干扰流进行考虑,从而给出此干扰流的每个干扰负载在通道a_i前面经过的日期。因此,这些日期是禁止的注入日期(也称为对有效负载注入无效的日期)。

[0356] 在此算法中,使用简化符号“$u=\sigma_c=a_i(j)\in L$”来定义与序列σ的第c负载相对应的负载u,所述负载也由其在通道a_i中的位置j定义。

[0357] 算法3:用不受控干扰流计算注入日期

$\boldsymbol{Require}: \sigma, \Delta_i, U_i, i \in \{1, \ldots, k\}$//给出值$U_i$，计算要视情况而定

$t_0 = \max_{u=a_i(j)\in L'}\{\Delta_i + 1 - \sigma(u)\}$//$t_0$的理论计算

$last = t_0$//σ_1在通道a_k前面经过的理论日期

$\boldsymbol{for}\ c = 1$ 到 $n\ \boldsymbol{do}$

设置 $u = a_i(j) = \sigma_c$

$\boldsymbol{do}\{$ //对负载σ_c进行调度以避免所有阻碍物

[0358]

$check = true$

$\boldsymbol{for\ all}\ z = k$ 到 $z = i, z - -\ \boldsymbol{do}$

$\boldsymbol{if}(last - \Delta_z \in U_z)\boldsymbol{then}\ check = false;$

$\boldsymbol{end\ for}$

$\boldsymbol{if}(check = false)\boldsymbol{then}\{$

$last = last + 1;$

$\boldsymbol{if}(c = 1)\boldsymbol{then} t_0 = t_0 + 1;$//如果干扰则$t_0$被修改

$\}\boldsymbol{end\ if}$

$\}\ \boldsymbol{while}\ (check = false);$

$T(u) = last - \Delta_i$//负载σ_c注入的最终日期

$last = last + 1$//σ_{c+1}在通道a_k前面经过的理论日期

$\boldsymbol{end\ for}$

//日期0的可能转换

将t_{min}设置为$min\{T(u), \forall u \in L\}$

[0359] $\boldsymbol{if}(t_{min} != 0)\boldsymbol{then}$

$$t_0 = t_0 - t_{min}$$

$\boldsymbol{for}\ u \in L\ \boldsymbol{do}$

$$T(u) = T(u) - t_{min}$$

$\boldsymbol{end\ for}$

$\boldsymbol{end}$

$\boldsymbol{return} t_0, T(u), \forall u \in L$

[0360] 图36是以上算法3的图。

[0361] 图47呈现了根据基于算法3(图36所示)的第二特定实施例的用于将k个输入流合并成一个输出流的方法的流程图。所述方法由控制系统执行至少一次。给定执行在时刻Tb处执行,并且包括以下步骤:

[0362] -获得(步骤471)集合L,所述集合L包括在所述时刻Tb处分布在所述k个通道上并且必须注入到所述收集器中以形成退出序列σ的n个有效负载,所述n个有效负载中的每个有效负载均由所述退出序列σ内的单个序列号标识,所述k个通道中的每个通道均含有必须逐个注入到所述收集器中的h_i个有效负载的根据序列号的上升次序的有序集合;

[0363] -对于每个通道a_i,i∈{1,...,k},获得(步骤472)由于不受控干扰负载或所述收集器的时间跨度的另一个阻碍物而对于将通道ai的所述有效负载之一注入到所述收集器中无效的日期集合U_i;

[0364] -在不考虑集合U_i,i∈{1,...,k}的情况下,计算(步骤473)日期t_0,在所述日期处,所述退出序列σ中的第一有效负载σ_1在通道a_k前面经过;

[0365] -对于所述退出序列σ中的所述第一有效负载σ_1,假设所述第一有效负载σ_1来自所述通道a_i(步骤474):

[0366] a)用t_0初始化t;

[0367] b)如果日期t、$t-\Delta_{k-1}$、$t-\Delta_{k-2}$、...、$t-\Delta_i$均不属于相应集合U_k、U_{k-1}、U_{k-2}、...、U_i,则用以下公式计算将所述第一有效负载σ_1注入到所述收集器中的日期:$T(u)=t-\Delta_i$,其中$u=\sigma_1=a_i(j)$;

[0368] c)如果日期t、$t-\Delta_{k-1}$、$t-\Delta_{k-2}$、...、$t-\Delta_i$之一属于相应集合U_k、U_{k-1}、U_{k-2}、...、U_i之一,则将t递增一个单位并且以新的t值重复步骤b)和c);

[0369] -对于所述退出序列σ中的每个后续有效负载σ_c,c∈{2,...,n},假设所述后续σ_c来自所述通道a_i(步骤475):

[0370] a')将t递增一个单位,其中t用于计算先前有效负载σ_{c-1}的注入日期;

[0371] b')如果日期t、$t-\Delta_{k-1}$、$t-\Delta_{k-2}$、...、$t-\Delta_i$均不属于相应所述集合U_k、U_{k-1}、U_{k-2}、...、U_i,则用以下公式计算将所述后续有效负载σ_c注入所述收集器上的日期:$T(u)=t-\Delta_i$,其中$u=\sigma_c=a_i(j)$;

[0372] c')如果日期t、$t-\Delta_{k-1}$、$t-\Delta_{k-2}$、...、$t-\Delta_i$之一属于相应集合U_k、U_{k-1}、U_{k-2}、...、U_i之一,则将t递增一个单位并且以新的t值重复步骤b')和c');

[0373] -根据以下公式计算(476)de t_{min}:$t_{min}=min\{T(u),\forall u\in L\}$;

[0374] -如果$t_{min}>0$,则根据以下公式修改(477)所述n个注入日期:$T(u)=T(u)-t_{min},\ \forall u\in L$;

[0375] -命令(步骤478)所述收集器和所述k个通道按照所述n个注入日期T(u),$\forall u\in L$将所述n个有效负载注入到所述收集器中。

[0376] 8.1.5一个特定实施例的描述

[0377] 看看已经存在于收集器上的不受控干扰流,由于最终干扰负载将在最后注入通道之前全部消失,因此没有按期望序列考虑在内。这是例如在图50(PTS Intelis GTP自动化系统)的上下文中的情况,其中管理来自自动化系统(PTS)的有效负载注入到收集器中并且管理到达此收集器的负载的返回以返回到自动化系统(PTS)中(在此可能的应用的上下文中,参见第2.2段)。

[0378] 图37展示了此具有返回到自动化系统(PTS)的返回流(构成不受控干扰流)的干扰负载(用叉标记)的配置。

[0379] 根据此配置,得到两个观察结果:

[0380] -返回流的干扰负载(并且因此干扰流的干扰负载)将不会在最后通道前面经过;以及

[0381] -在通道a_1之前离开收集器(朝着自动化系统(PTS))的干扰负载没有产生问题并且因此将不进行研究。所有剩余负载将必然在通道a_1前面经过。

[0382] 8.1.5.1计算由干扰负载的流得出的无效日期列表

[0383] 利用已经存在于收集器中的n'个干扰负载的流F并且退出注入通道之间的收集器。此干扰流(被称为返回流)与必须返回到自动系统(PTS)中的存量的负载相对应。这些干扰负载确实必须在一个或多个注入通道前面经过。对于此返回流(干扰流)的每个干扰负载p,g_p(并且分别为l_p)表示干扰负载p经过其前面的第一通道(并且分别为最后通道)的数量。将看到使能够根据已知数据计算由返回流得出的无效注入日期(U_i)的两种方法。

[0384] 在此特定实施例中,对于特定于自动化系统(PTS)的返回流的每个负载p,$g_p=1$并且$l_p<k$。然而,将用符号g_p和l_p给出一般解法。

[0385] -要么,在上文所提及的返回流的具体情况下,已知返回流(干扰流)的第一干扰负载到达编号为a_1的通道(或再次称为机器M_1)前面的日期(表示为t_r)。还已知此返回流的干扰负载在收集器上分布的次序(称此为序列σ')以及每个干扰负载与此干扰流的另一干扰负载之间的持续时间。使σ'_p给出返回流的p^{th}干扰负载,并且σ'(u,v)以"跨度"数量给出干

扰负载u与干扰流中的干扰负载v之间的持续时间。

[0386] 然后,时刻t_r、$t_r+\sigma'(\sigma'_1,\sigma'_2)$、...、$t_r+\sigma'(\sigma'_1,\sigma'_{n'})$对于机器$M_1$和形式$U_1$而言拥塞。

[0387] • 对于机器M_i,i=2,...,k,拥塞日期是$U_i=\{t_r+\sigma'(\sigma'_1,\sigma'_j)+\Delta_{i,1}$,如果$l(\sigma'_j)\geq i,\ \forall j=1,\ldots,n'\}$。

[0388] • 另一种看待事情的方式是复制干扰负载u的机器M_i的"拥塞"(或无效)日期U_i,其为l(u)≥i+1,并且添加到所有这些时刻$\Delta_{i,i+1}$以获得机器M_{i+1}的拥塞日期U_{i+1}的列表。

[0389] -或在更一般的框架中,已知每个干扰负载p相对于将要表示为δ_p的日期0的最后通道a_k的距离。用这些数据如下计算出值U_i:

$$For\ all\ i\in\{1,\ldots,k\}\ do\ U_i=\emptyset;$$

$$For\ all\ p\in F\ do$$

$$For\ all\ i\in\{g_p,\ldots,l_p\}\ do$$

[0390] $$U_i=U_i\cup\{\delta_p-\Delta_i\};$$

$$end\ for$$

$$end\ for$$

$$return\ U_i,\forall i=1\ldots k$$

[0391] 8.1.5.2实例

[0392] 返回到第6.2.2段的实例的配置。

[0393] 采用包括两个或三个有效负载的k=4通道。每个有效负载根据退出序列σ=(1,2,3,4,5,6,7,8,9)。由唯一的序列号标识。这是可实现序列。已知从每个通道到最后通道的距离:$\Delta_1=12$,$\Delta_2=7$,$\Delta_3=4$并且$\Delta_4=0$。

[0394] 如图38所展示的,假设这次收集器的三个时间跨度受到干扰负载A、B和C的阻碍。已知这些干扰负载会在某些通道前面经过并且此后从收集器中消失。负载A和B在通道a_1和a_2前面经过。最后,负载C在所有通道a_1、a_2和a_3前面经过。因此,得出$l_A=2$,$l_B=2$并且$l_C=3$,以及$g_A=g_B=g_C=1$。

[0395] 当所述批次中的第一有效负载将注入到收集器的第一未占用位置中时示出了日期0处的系统状态。在所述图中,收集器的未占用位置是空闲的,并且又指示先前批次的有效负载所占据的位置,并且字母表示干扰流的干扰负载。日期0处的干扰流的位置使能够推断出以下信息片段:$\delta_A=13$,$\delta_B=14$并且$\delta_C=18$.

[0396] 干扰负载B迫使有效负载4及时挪动一个位置并且将在收集器的最终流中留出未用空间。然后,干扰流C迫使有效负载7挪动又另一位置,从而稍后在收集器的最终流中留出未用空间。

[0397] 图39是展示了在此情况下提出的解法的甘特图。可以看到,如果要控制(而不是不受控)干扰流,则第一未用空间可以由通道a_3或a_4填充,并且第二未用空间可以仅由通道a_4填充。因此,可以避免未用空间,并且来自收集器的最终流(输出流)的吞吐量将增加。

[0398] 8.2具有受控干扰流的系统的建模

[0399] 已经看到,当干扰流不受控时,其将在收集器的最终流中产生未用空间。因此,控

制干扰流以避免这些未用空间是值得的。因此,可以使注入流的负载优先于干扰流的负载,并且能够使干扰流进入由不同通道的注入留出的未占用或自由空间。

[0400] 8.2.1问题的制定

[0401] 现在假设可以控制干扰流。这相当于实现了对干扰流的干扰负载的注入的控制,并且必须决定在通道a_l之前注入干扰负载p的日期(表示为$start_p$)以免干扰输出流(由注入流的合并形成的输出流),即在退出流中不创建任何未用空间。

[0402] 8.2.2求解算法的想法

[0403] 将首先计算来自通道的有效负载的注入日期,如同不存在干扰流一样(参见第6节)。正要给出这些注入日期作为对于在通道(a_i)前面经过的干扰负载不可用的日期(U_i)。然后,考虑到无效日期列表,将计算干扰流的负载注入。为此,按干扰负载的到达次序,最早对其每次经过通道前面进行调度,即进入到相应机器中的这些日期落到自由日期之后。

[0404] 8.2.3通用算法

[0405] 下文详细描述的算法4给出了有效负载的注入日期以及干扰流的控制。要注入的负载构成收集器的最终流。在不考虑干扰流的情况下决定注入。然后,考虑到这次注入的有效负载与上述算法3中的不受控流具有同一作用,将确定何时允许干扰流的负载通过。因此,可以使用与上文相同的技术来计算集合,

[0406] V_i,i=1...k从而给出注入流的每个有效负载在通道a_i前面经过的日期。因此,这些日期是禁止干扰负载注入的日期。

[0407] 可以回想一下,$\sigma'_{p,p'}$给出了日期0处的干扰负载p与p′之间的距离,并且t_r给出了干扰流的第1负载在无需等待的情况下到达通道号a_1前面的日期。

[0408] 在此算法中,使用简化符号“$u=\sigma_c=a_i(j)\in L$”来定义与序列σ的第c负载相对应的负载u,所述负载也由其在通道a_i中的位置j定义。

[0409] 算法4:计算受控干扰流的负载的注入日期

$$\textbf{Require}: \sigma, \Delta_i, \Delta_{1i}, i \in \{1, \dots, k\}$$

$t_0 = \max_{u=a_i(j)\in L'}\{\Delta_i + 1 - \sigma(u)\}$//$t_0$的理论计算

$\textbf{for all}\ i \in \{1, \dots, k\}\textbf{do}$

$\quad\textbf{for all}\ j \in \{1, \dots, h_i\}\textbf{do}$

设置 $u = a_i(j)$

$T(u) = t_0 + \sigma(u) - 1 - \Delta_i$

$\quad\quad\textbf{for all}\ z \in \{i, \dots, k\}\ \textbf{do}$

$\quad\quad\quad V_z = V_z \cup \{T(u) + \Delta_i - \Delta_z\}$

[0410] $\quad\quad\textbf{end for}$

$\quad\textbf{end for}$

$\textbf{end for}$

设置 $t = \delta_1 - \Delta_1$//*第一干扰负载到达*a_1前面

$\textbf{for all}\ p = 1$ 到 $f\ \textbf{do}$//*按其在*σ_f中的位置标识负载

$\quad\textbf{do}\{$

$\quad check = true$

$\quad\quad\textbf{for}\ i = g_p$ 到 $l_p\textbf{do}$

$\quad\quad\quad\textbf{if}(t + \Delta_1 - \Delta_i \in V_i)\textbf{then}\ check = false; t++; break;$

$\quad\quad\textbf{end for}$

$\quad\}\textbf{while}\ (check = false);$

[0411] $\quad start_p = t$

$\quad t = Max\ (t + 1,\ \delta_{p+1} - \Delta_1)$//*可能等待* $p + 1$ *的到达*

$\textbf{end for}$

$\textbf{return}\ T(u), start_p, \forall u \in L, \forall p \in F$

[0412] 图40是以上算法4的图。

[0413] 图48呈现了根据基于算法4(图40所示)的第三特定实施例的用于将k个输入流合并成一个输出流的方法的流程图。所述方法由驱动系统执行至少一次。给定执行在时刻Tb处执行,并且包括以下步骤:

[0414] -对于来自通道ai的每个有效负载u=a_i(j)∈L,计算(步骤481)所述有效负载在所述通道a_i到a_k前面经过的日期,所述有效负载注入到所述收集器中的所述日期T(u)是已知的;

[0415] -对于每个通道a_i,i∈{1,...,k},获得(步骤482)各自含有所述有效负载之一在所述通道a_i前面经过的日期的集合V_i;

[0416] -对于借助于沿所述收集器定位并且在所述收集器的移动方向上具有通道$a_{g(p)}$的

控制构件控制的干扰流的每个受控干扰有效负载p,其中g(p)∈{1,...,k},在所述收集器的移动方向上,$a_{g(p)}$也是所述受控干扰负载在其之前经过的第一通道(步骤483):

[0417] a)用日期$\delta_1-\Delta_1$初始化t,其中δ_1表示如果第一受控干扰负载未在通道a_k之前搁置则可能到达的日期;

[0418] b)计算所述受控干扰负载在通道$a_{g(p)}$到$a_{l(p)}$前面经过的日期$t+\Delta_{1,g(p)}$、$t+\Delta_{1,g(p)+1}$、...、$t+\Delta_{1,l(p)}$,其中在所述收集器的移动方向上,$a_{l(p)}$是所述受控干扰负载在其之前经过的最后通道;

[0419] c)如果日期$t+\Delta_{1,g(p)}$、$t+\Delta_{1,g(p)+1}$、...、$t+\Delta_{1,l(p)}$均不属于相应集合$V_{g(p)}$、$V_{g(p)+1}$、...、$V_{l(p)}$,则用以下公式计算所述受控干扰负载p在通道a_1前面经过的假想日期$start_p$:$start_p=t$;

[0420] d)如果日期$t+\Delta_{1,g(p)}$、$t+\Delta_{1,g(p)+1}$、...、$t+\Delta_{1,l(p)}$之一属于相应集合$V_{g(p)}$、$V_{g(p)+1}$、...、$V_{l(p)}$之一,则将t递增一个单位并且以新的t值重复步骤b)、c)和d);

[0421] -命令(步骤484)所述控制构件在每个受控干扰负载p位于通道$a_{g(p)}$之前的日期$start_p+\Delta_{1,g(p)}$处进行注入。

[0422] 8.2.4一个特定实施例的描述

[0423] 以所有先前案例的共同实例为例以能够对最终解法进行比较。

[0424] 使k=4包括两个或三个负载的通道。每个负载根据退出序列σ=(1,2,3,4,5,6,7,8,9).由单个序列号标识。这是可实现序列。已知从每个通道到最后通道的距离:$\Delta_1=12$,$\Delta_2=7$,$\Delta_3=4$并且$\Delta_4=0$。干扰流由负载A、B、C构成,使得$l_A=2$,$l_B=2$,$l_C=3$,$g_A=g_B=g_C=1$,$\delta_A=13$,$\delta_B=14$并且$\delta_C=18$。

[0425] 这次,干扰流是受控的。在图41中可以看到此干扰流的控制点,由通道a_1之前的竖直矩形表示。

[0426] 图42是呈甘特图形式的在未考虑干扰流(“纯注入解法”)的情况下的解法的图示。

[0427] 考虑干扰流的算法解法相当于将干扰负载滑入到灰色方框中,而纯注入解法是留空的。负载A一到达干扰流的控制点就被允许经过,而负载B保持一个时间单位。此等待使得在即将注入负载5之前使干扰负载B进入注入的自由空间。序列的负载C和负载8也是这种情况。

[0428] 图43是呈甘特图形式的为控制干扰流(即通过选择干扰负载的放置)而提出的算法的图示。负载B和C在收集器的最终流中不再形成未用空间。其被放置在战略性地点,使得通过时间兼容负载的注入来填充最初留出的未用空间。通过干扰流的控制,表明收集器上的未用空间是可避免的。

[0429] 8.3用不受控干扰流和受控干扰流对系统建模

[0430] 如果同时存在至少一种不受控干扰流和至少一种受控干扰流,将继续进行。

[0431] 将不受控干扰负载与受控干扰负载区分开来,如下:

[0432] -使S为s个在没有控制的情况下被允许强制进入收集器的通道的注入区的不受控干扰负载的集合,各自由唯一的序列号标识,所述序列号与其在到达次序中的位置(表示为σ'_s)相对应。

[0433] 使δ'_p指定在日期Tb处的每个干扰负载p∈S与最后通道a_k之间的距离,并且使g'_p和l'_p分别指定干扰负载p必须经过其前面的第一通道和最后通道的编号。

[0434] 最后,集合U_i,i=1...k,给出了有效负载无法在通道a_i前面经过的日期。

[0435] -使F指定f个在控制的情况下被允许进入收集器的通道的注入区的干扰负载的集合,各自由唯一的序列号标识,所述序列号与其在到达次序中的位置(表示为σ_f)相对应。

[0436] 使δ_p指定在日期Tb处的从每个干扰负载$p\in F$到最后通道a_k的距离,并且使g_p和l_p分别表示干扰负载p必须经过其前面的第一通道和最后通道的编号。

[0437] 最后,集合V_i,i=1...k,给出了这些受控干扰负载无法在通道a_i前面经过的日期。

[0438] 考虑到不受控干扰流,算法5在下文提供了有效负载的注入日期以及干扰流的控制的详细视图。

[0439] 算法5:在考虑到不受控干扰流的情况下计算受控干扰流的有效负载和负载的注入日期:

$\boldsymbol{Require}: F, S, \sigma, \Delta_i, \Delta_{1i}, a_i(j), \forall i \in \{1, \ldots, k\}, \forall j \in \{1, \ldots, h_i\}$

$t_0 = \max_{u=a_i(j) \in L'} \{\Delta_i + 1 - \sigma(u)\}$//$t_0$的理论计算

//计算值U_i:

for all $p = 1$ 到 s **do**

for all $i \in \{g'_p, \ldots, l'_p\}$ **do**

$U_i = U_i \cup \{\delta'_p - \Delta_i\}$//在前面的负载$a_i$

end for

end for

//计算值$T(u)$

$last = t_0$//σ_1在通道a_k前面经过的理论日期

[0440] **for** $u = 1$ 到 n **do**

设置 $u = a_i(j)$

do {//σ的u^{th}负载的调度

$check = true$

$for\ all\ z = k$ à $z = i, z - -\ do$

$if\ (last - \Delta_z \in U_z)\ then\ check = false;$

$end\ for$

$if\ (check = false)\ then$ {

$last = last + 1$;

$if\ (u = 1)\ then\ t_0 = t_0 + 1;$

} **end if**

} **while** $(check = false)$;

$T(u) = last - \Delta_i$//负载u注入的最终日期

$last = last + 1$//a_k之前的后续位置

end for

//日期 0 的可能转换

将t_{min}设置为$min\{T(u), \forall u \in L\}$

if $(t_{min} != 0)$**then**

$t_0 = t_0 - t_{min}$

for $u \in L$ **do**

$T(u) = T(u) - t_{min}$

end for

end if

//计算值V_z:

for $u = 1$ **到** n **do**

设置 $u = a_i(j)$

for all $z \in \{i, \dots, k\}$ **do**

$V_z = V_z \cup \{T(u) + \Delta_i - \Delta_z\}$

end for

[0441]

end for

for all $z \in \{1, \dots, k\}$ **do**

$V_z = V_z \cup U_z$

end for

//计算值$start_p$:

设置 $t = \delta_1 - \Delta_1$ //第一干扰负载到达前面 a_1

for all $p = 1$ 到 f **do**//按其在σ_f中的位置标识负载

do {

$check = true$

for $i = g_p$ 到 l_p**do**

if$(t + \Delta_1 - \Delta_i \in V_i)$**then** $check = false; t++; break;$

end for

} **while** $(check = false)$;

$start_p = t$

$t = Max\ (t + 1,\ \delta_{p+1} - \Delta_1)$//可能等待 $p + 1$ 的到达

end for

[0442] **return** $t_0, T(u), \forall u \in L, start_p, \forall p \in F$

[0443] 图45是以上算法5的图。

[0444] 图49是根据基于算法5(图45所示)的第四特定实施例的用于将k个输入流合并成一个输出流的方法的流程图。所述方法由控制系统执行至少一次。给定执行在时刻Tb处执行,并且包括以下步骤:

[0445] -对于来自通道ai的每个有效负载$u=a_i(j)\in L$,计算(步骤491)所述有效负载在通道a_i到a_k前面经过的日期,所述有效负载到所述收集器中的注入日期T(u)是已知的;

[0446] -对于每个通道a_i,$i\in\{1,\ldots,k\}$,一方面,获得(步骤492)含有所述有效负载之一在通道a_i前面经过的每个日期的集合V_i,并且另一方面,获得对于将通道a_i的所述有效负载之一注入到所述收集器中无效的日期集合U_i;

[0447] -对于通过沿所述收集器定位并且在所述收集器的移动方向上在通道$a_{g(p)}$之前的控制构件控制的干扰流的每个受控干扰有效负载p,其中$g(p)\in\{1,\ldots,k\}$,在所述收集器的移动方向上,$a_{g(p)}$也是所述受控干扰负载在其前面经过的第一通道(步骤493):

[0448] a)用日期$\delta_1-\Delta_1$初始化t,其中δ_1表示如果第一受控干扰负载在通道a_k前面未搁置则到达的可能日期;

[0449] b)计算所述受控干扰负载在通道$a_{g(p)}$到$a_{l(p)}$前面经过的日期$t+\Delta_{1,g(p)}$、$t+\Delta_{1,g(p)+1}$、...、$t+\Delta_{1,l(p)}$,其中在所述收集器的移动方向上,$a_{l(p)}$是所述受控干扰负载在其之前经过的最后通道;

[0450] c)如果日期$t+\Delta_{1,g(p)}$、$t+\Delta_{1,g(p)+1}$、...、$t+\Delta_{1,l(p)}$均不属于相应集合$V_{g(p)}$、$V_{g(p)+1}$、...、$V_{l(p)}$,则用以下公式计算所述受控干扰负载p在通道a_1前面经过的假想日期$start_p$:$start_p=t$;

[0451] d)如果日期$t+\Delta_{1,g(p)}$、$t+\Delta_{1,g(p)+1}$、...、$t+\Delta_{1,l(p)}$之一属于相应集合$V_{g(p)}$、$V_{g(p)+1}$、...、$V_{l(p)}$之一,则将t递增一个单位并且以新的t值重复步骤b)、c)和d);

[0452] -命令(步骤494)所述控制构件在每个受控干扰负载p位于通道$a_{g(p)}$之前的日期$start_p+\Delta_{1,g(p)}$处进行注入。

[0453] 9.结论

[0454] 由于行进时间对于获得尽可能最高的生产率而言至关重要,因此在物流中能够组合若干个输入流非常重要。使用每个系统的所有机械容量并且因此使这种收集器的吞吐量最大化对于避免例如客户订单拣选中的时间浪费而言至关重要。

[0455] 已经通过以同步方式最早注入负载而获得最佳吞吐量。当收集器专用于输入流时,尤其获得了所述收集器的最大容量。除了建模和数学解析之外,还提取了使能够直接计算每个负载的注入日期的公式。一旦负载全部位于收集器上,就给出这些结果以符合所述负载的最终退出序列,从而可以组合速度的优化和分拣操作的优化。

[0456] 讨论了收集器上若干种类型的流的管理,使用术语“干扰流”来表示与注入流不同的流。能够使用在不受控时考虑此干扰流的算法来计算注入日期。然而,在不保证极大性的情况下吞吐量是最佳的。实际上,这种不受控干扰流使得不可能恢复由干扰流留出的空间,所述干扰流会从注入通道之间的收集器中流出。

[0457] 还提出了一种控制干扰流的解法。已经给出了另一算法以使受控干扰流通过的方式为使得一旦所有注入通道已经通过,就优先考虑收集器的最大吞吐量。当受控干扰流的所有负载在最后注入通道之前消失时,对此受控干扰流的控制使注入流能够一次达到收集

器上的最大吞吐量。如果没有,则剩余的干扰流将包含在收集器的最终流中并且在没有空间的情况下插入到序列中。

[0458] 还已经给出了一种算法以管理至少一个不受控流(作为非强制干扰负载的阻碍物)并且同时管理至少一个受控流(干扰负载)。

[0459] 10.控制系统的实例

[0460] 所提出的解法是一种在物流仓库内将分别由k个被称为通道a_i(其中$i\in\{1,\ldots,k\}$)的输送机运输的k个有效负载输入流合并成由被称为收集器的另一个输送机运输的一个有效负载输出流的方法。

[0461] 合并方法由控制系统执行。这是例如中央仓库控制系统或WCS。

[0462] 图44呈现了根据本发明的一个特定实施例的控制系统90的结构。此控制系统包括有效存储器92(例如RAM)、处理单元91,所述处理单元例如配备有处理器并且由存储在只读存储器93(例如ROM或硬盘驱动器)中的计算机程序930控制。

[0463] 在初始化时,计算机程序的代码指令例如被加载到有效存储器92中,并且然后由处理单元91的处理器执行以实施本发明的合并方法(根据上文所描述的不同实施例中的任一个实施例)。处理单元91输入与输入流有关的信息片段94。处理单元91的处理器处理信息94并且在出口处生成用于控制(命令)系统中包含的不同元件(尤其是通道、收集器、控制构件等)的指令或命令95。

[0464] 在若干种可能的方式中,此图44仅展示了在其实施例中的任一个实施例中执行本发明的技术的一种特定方式。实际上,在执行包括指令序列的程序的可重编程计算机器(例如PC计算机、DSP处理器、微控制器等)上或在专用计算机器(例如一组逻辑门,如FPGA或ASIC或任何其它硬件模块)上同样很好地获得控制系统。

[0465] 如果控制系统是用可重新编程计算机器来执行的,则对应的程序(即指令序列)可以存储在可拆卸的存储介质中(如例如软盘、CD-ROM或DVD-ROM)或不可拆卸的存储介质中,此存储介质可由计算机或处理器部分或完全读取。

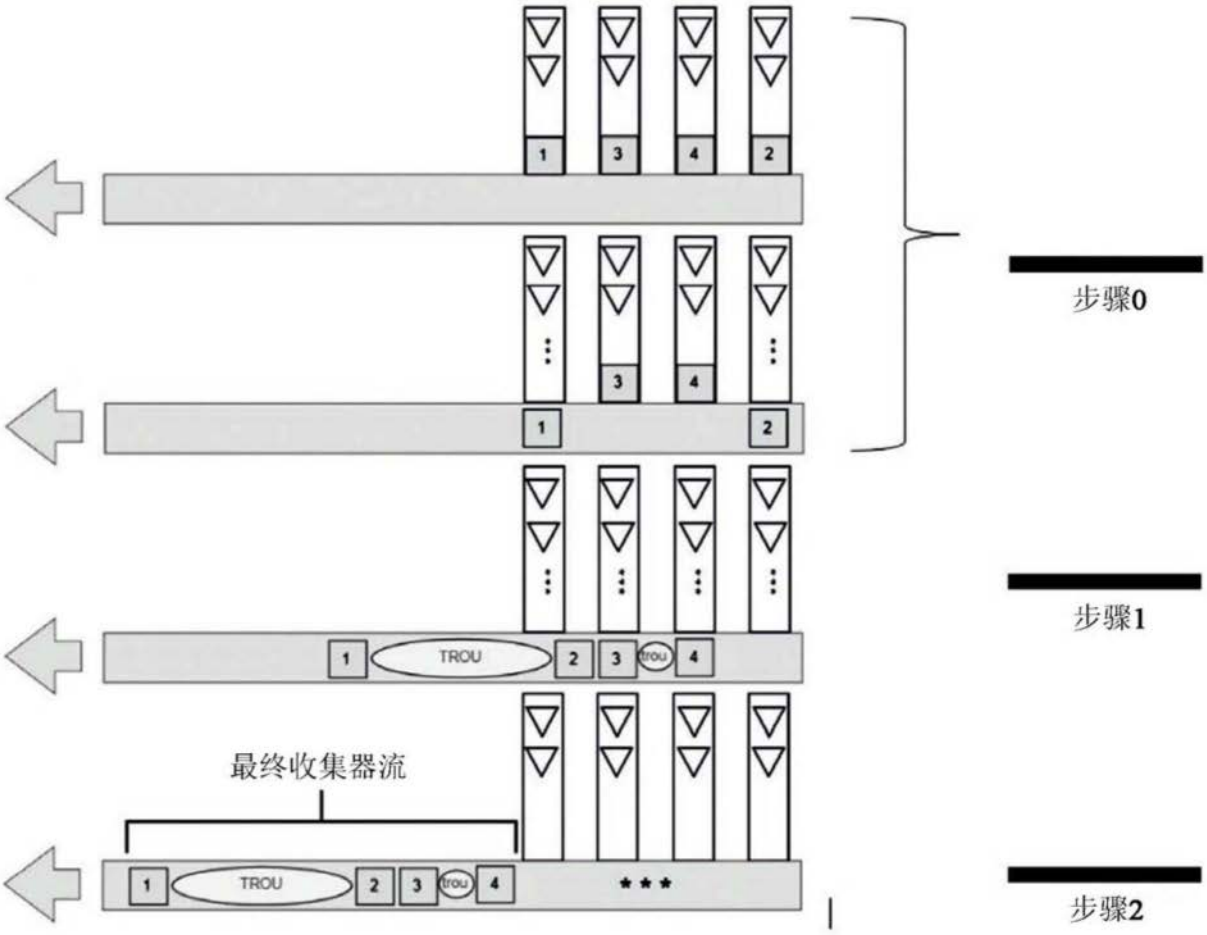


图1

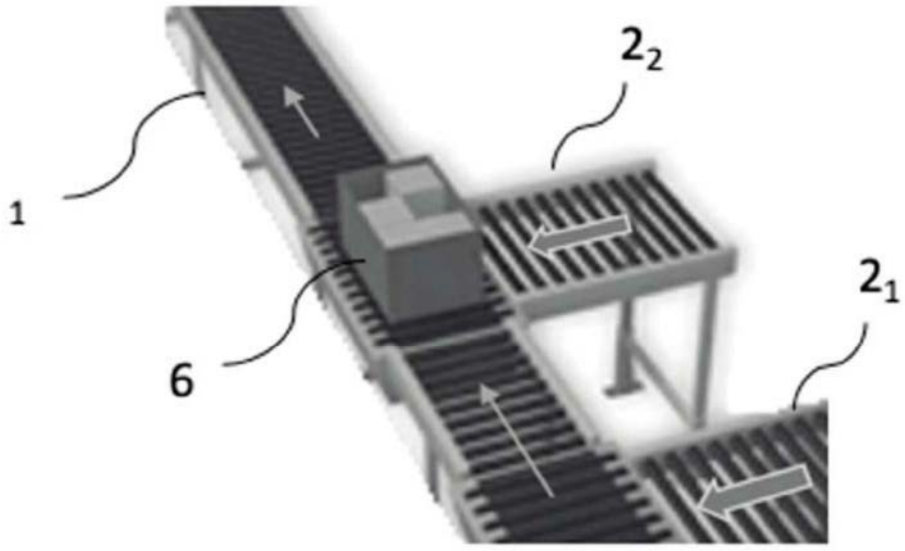


图2

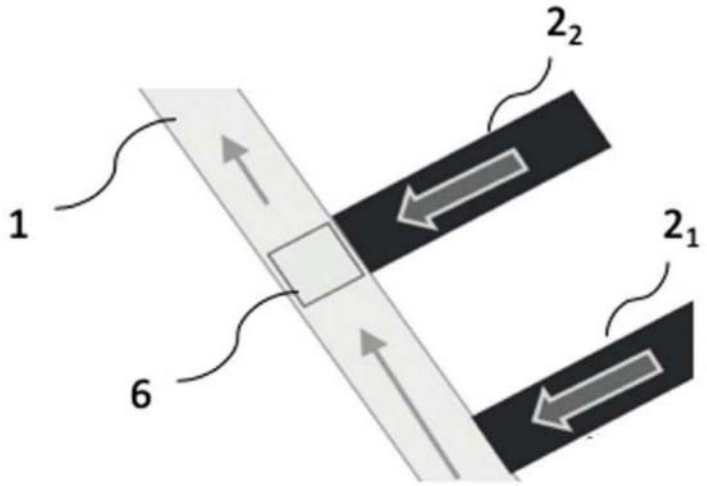

2_2
1
2_1
6

图3

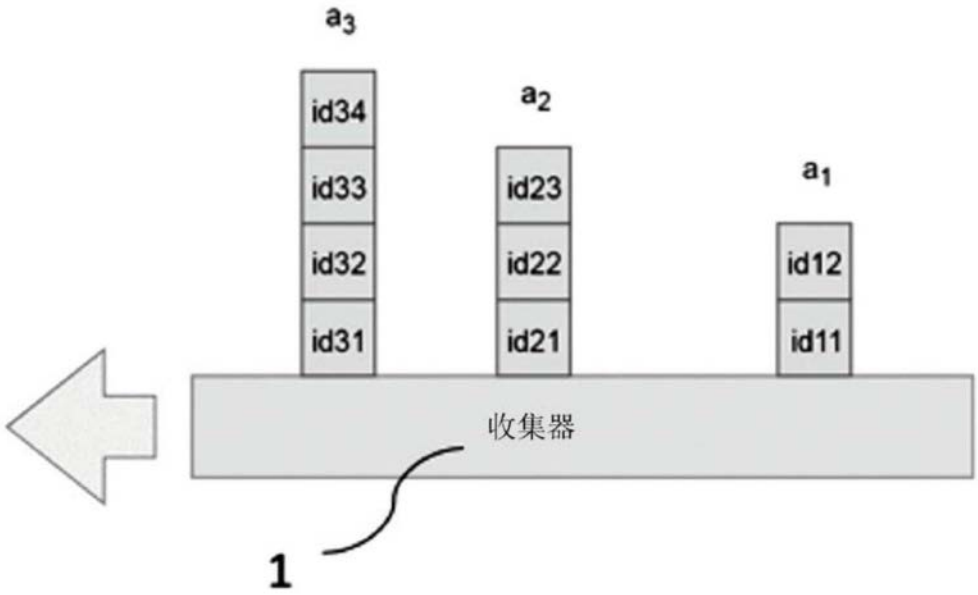

a_3
a_2
a_1
id34
id33
id32
id31
id23
id22
id21
id12
id11
收集器
1

图4

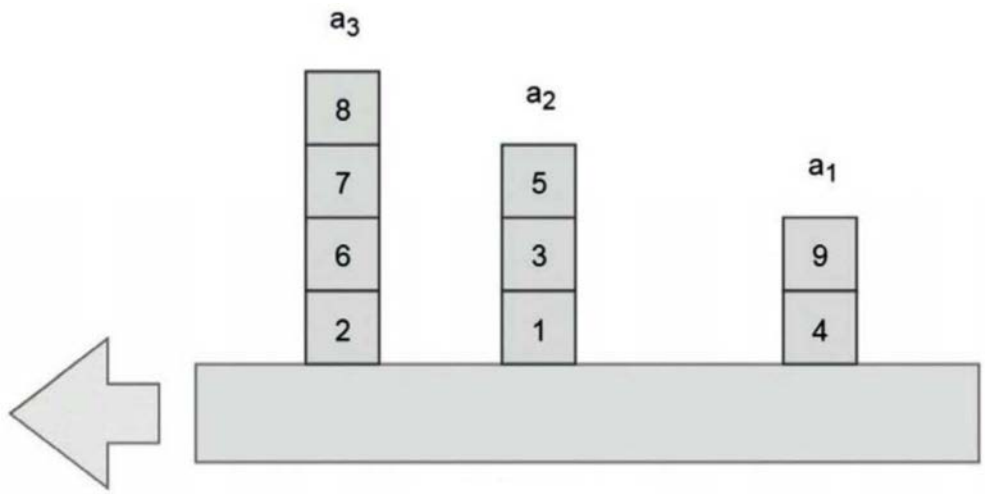


图5

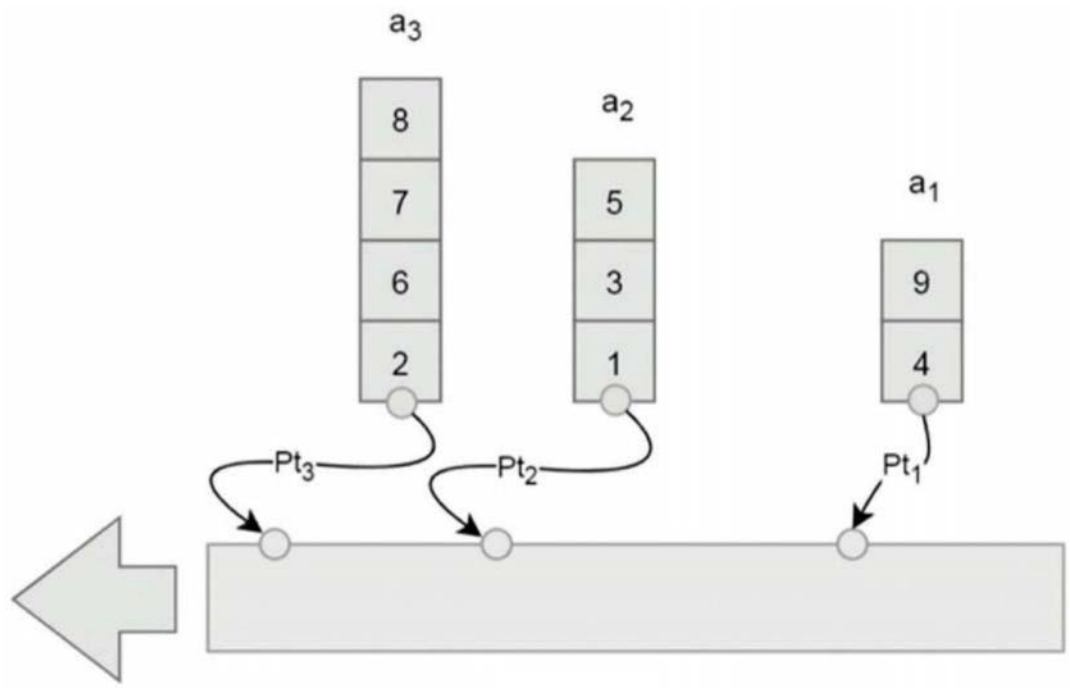


图6

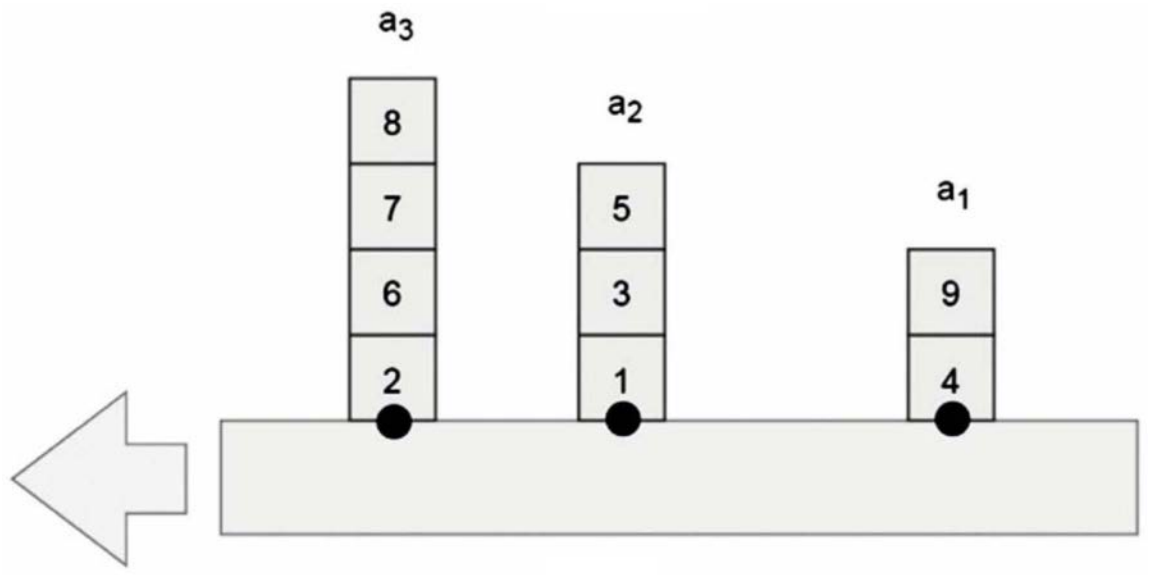


图7

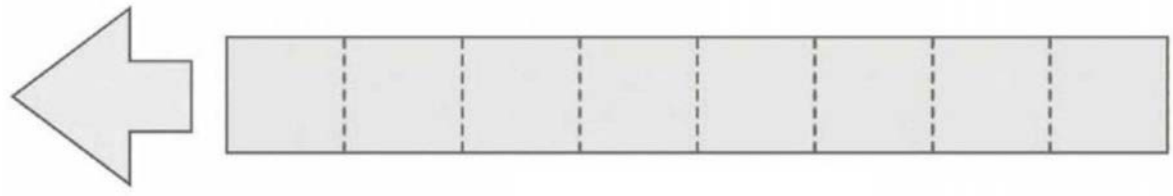


图8

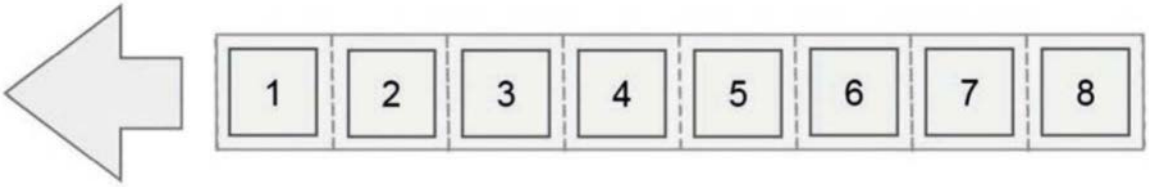


图9

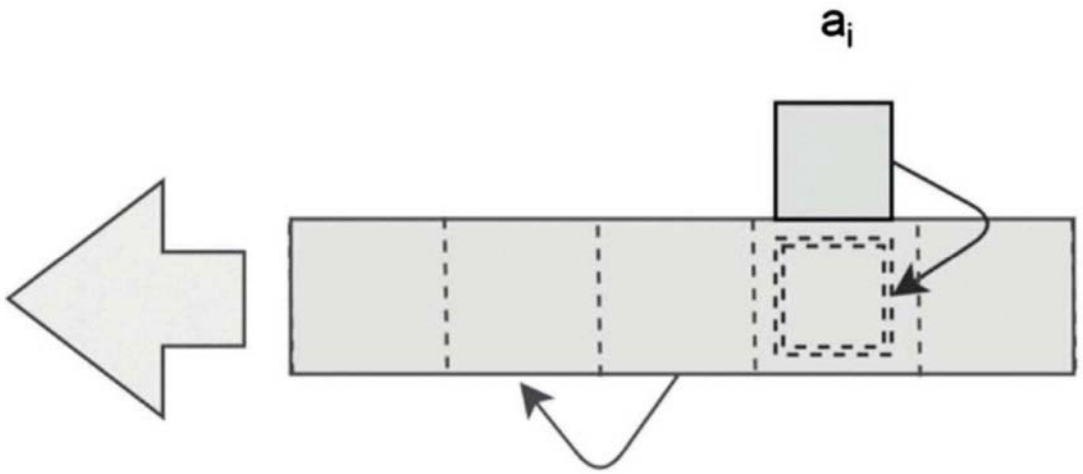


图10

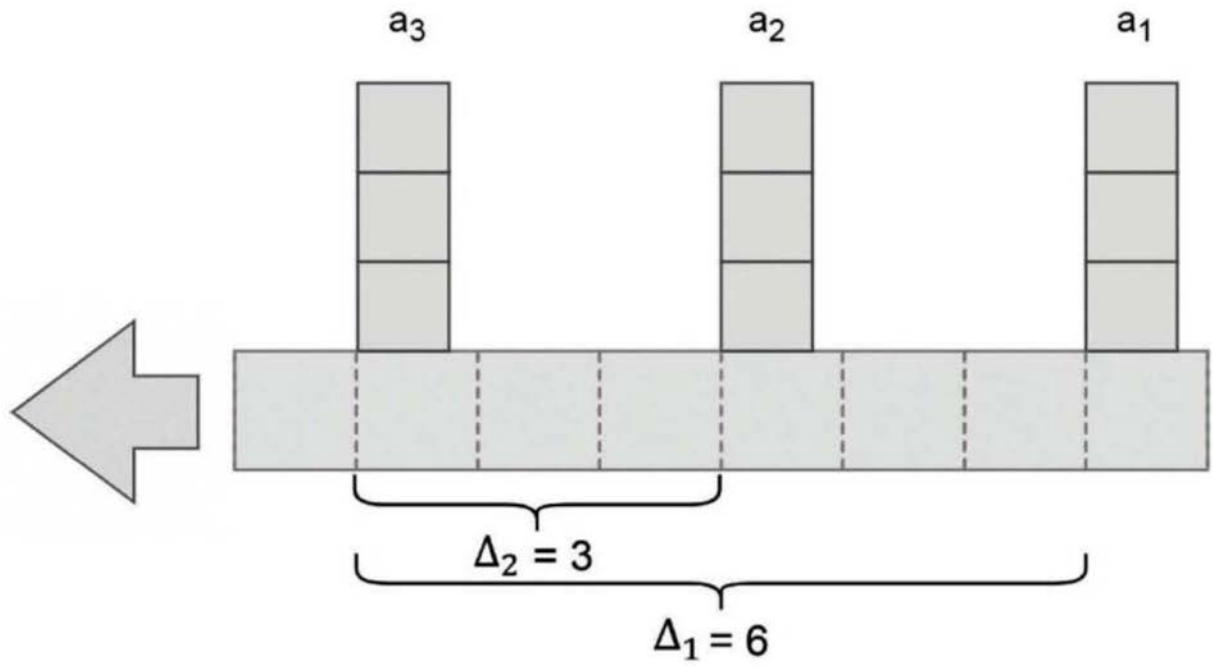


图11

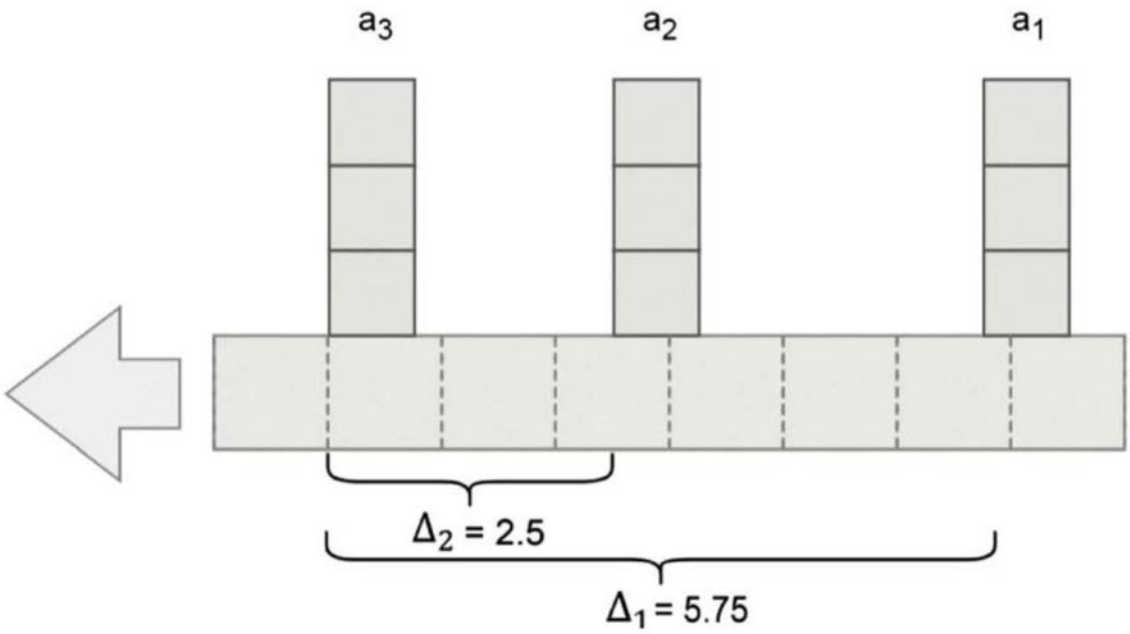


图12

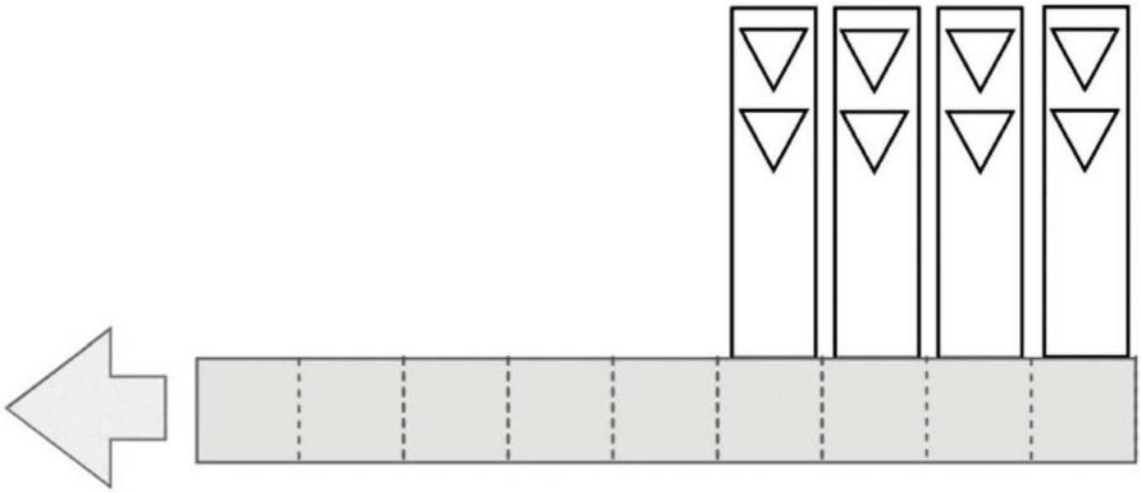


图13

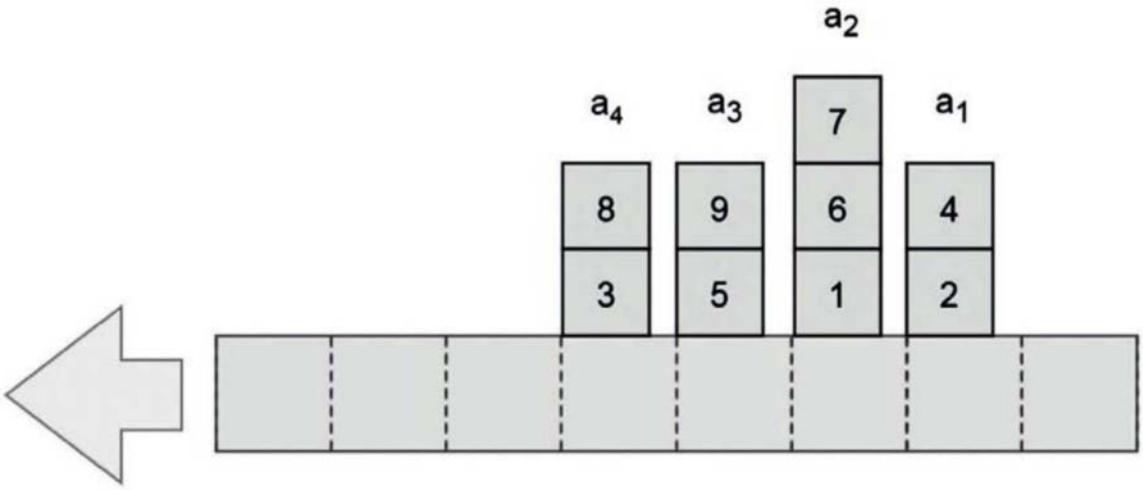

a2
a4
a3
a1
7
8
9
6
4
3
5
1
2

图14

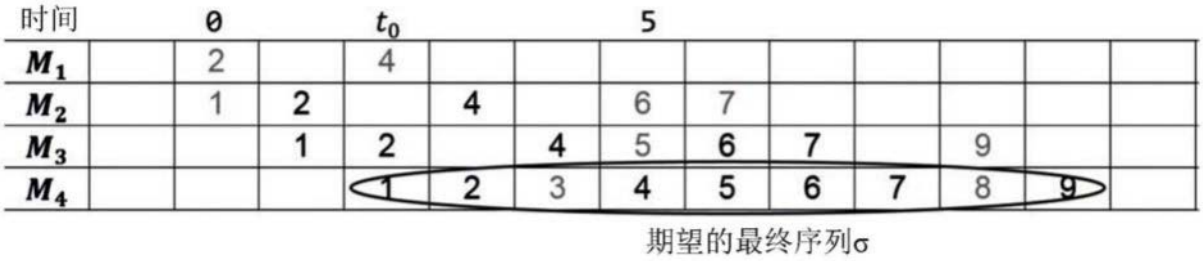

时间
0
t_0
5
M_1
M_2
M_3
M_4
期望的最终序列σ

图15

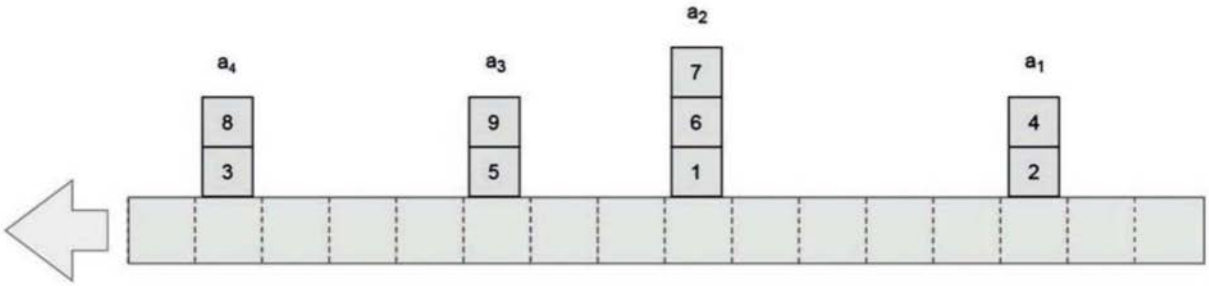

a2
a4
a3
a1
7
8
9
6
4
3
5
1
2

图16

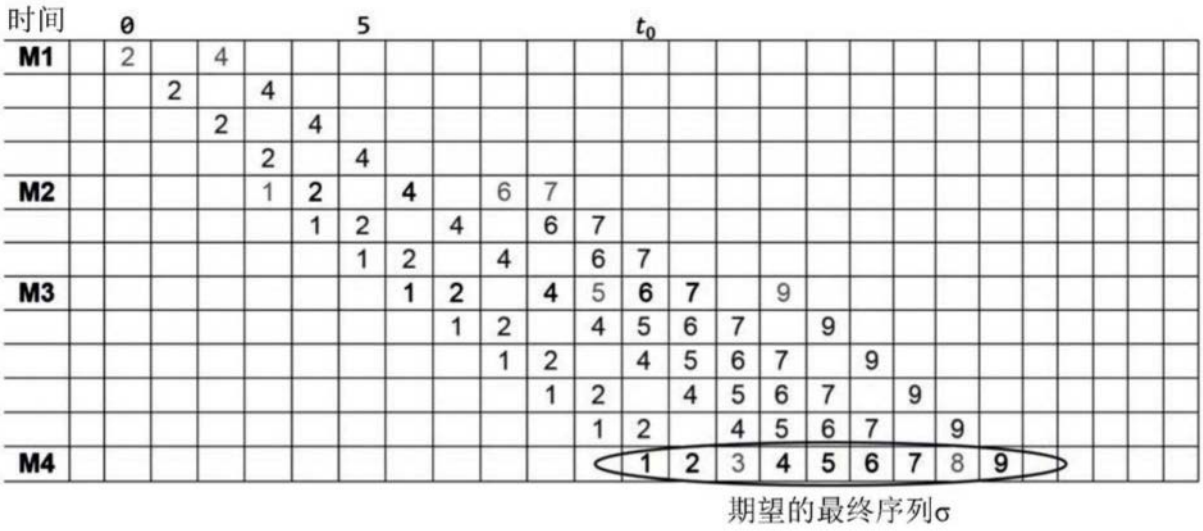

时间
0
5
t_0
M1
M2
M3
M4
期望的最终序列σ

图17

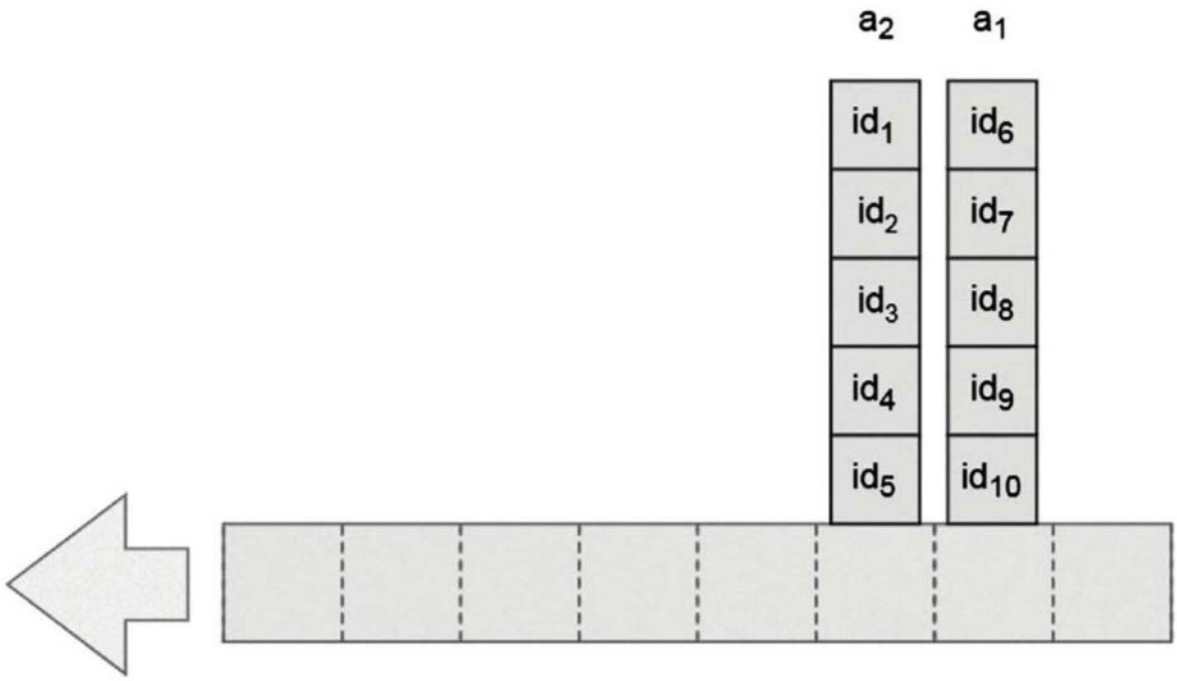

a_2
a_1
id_1
id_2
id_3
id_4
id_5
id_6
id_7
id_8
id_9
id_{10}

图18

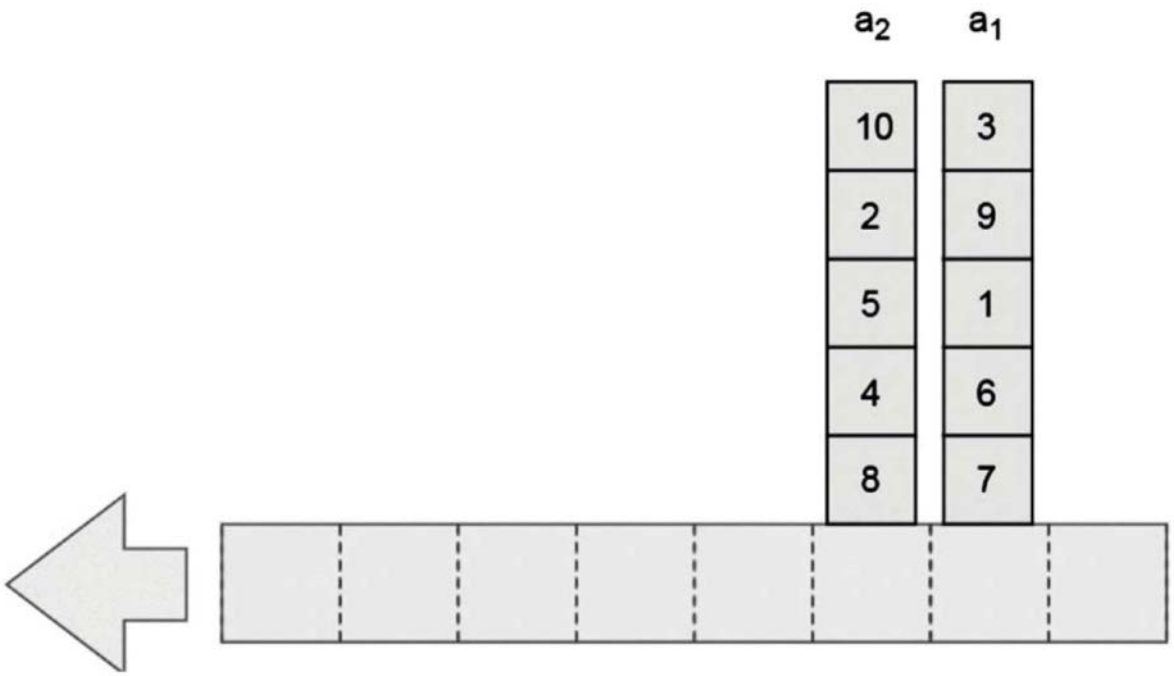


图19

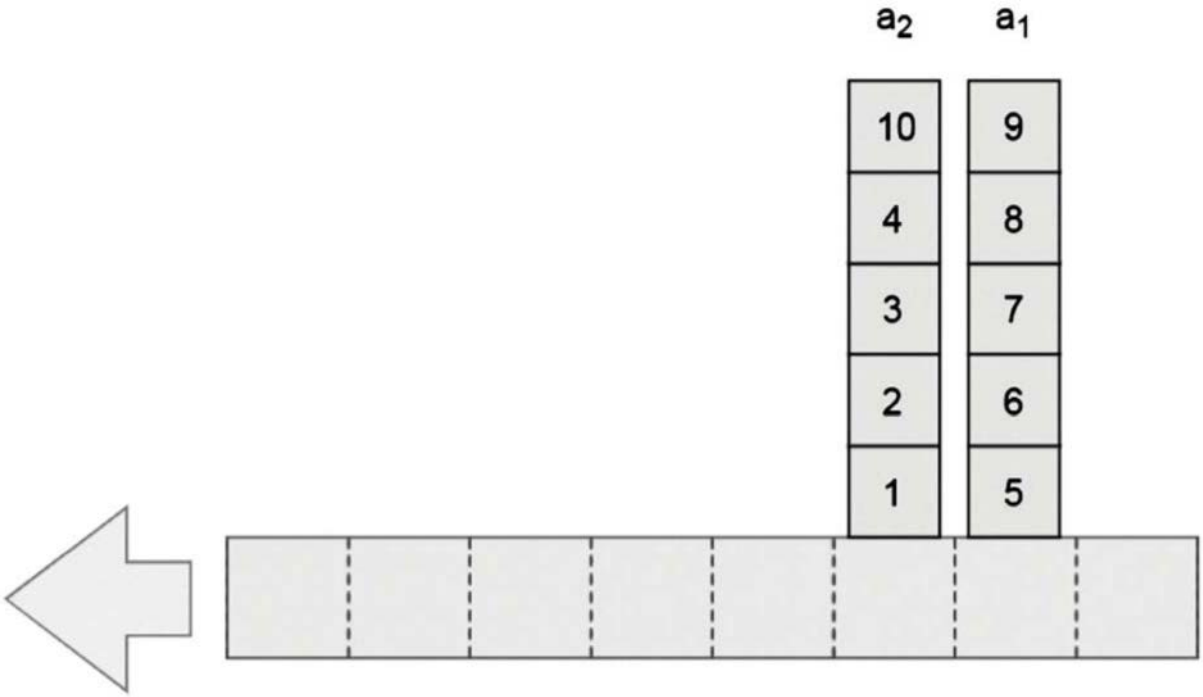


图20

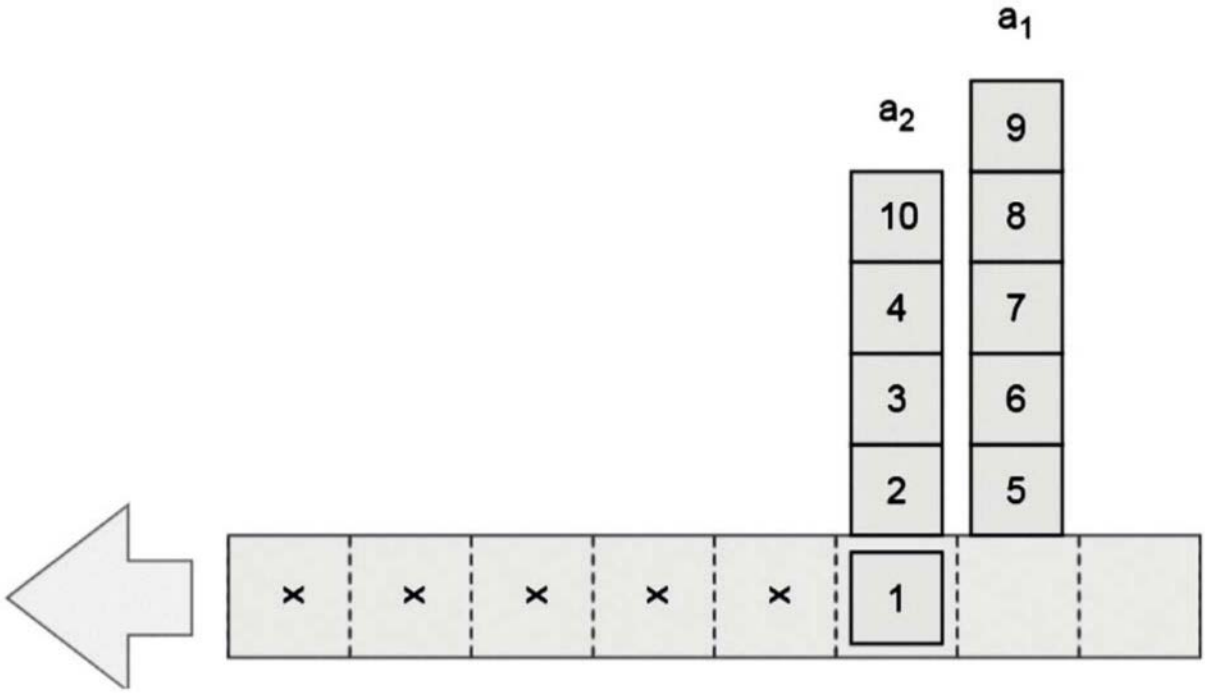


图21

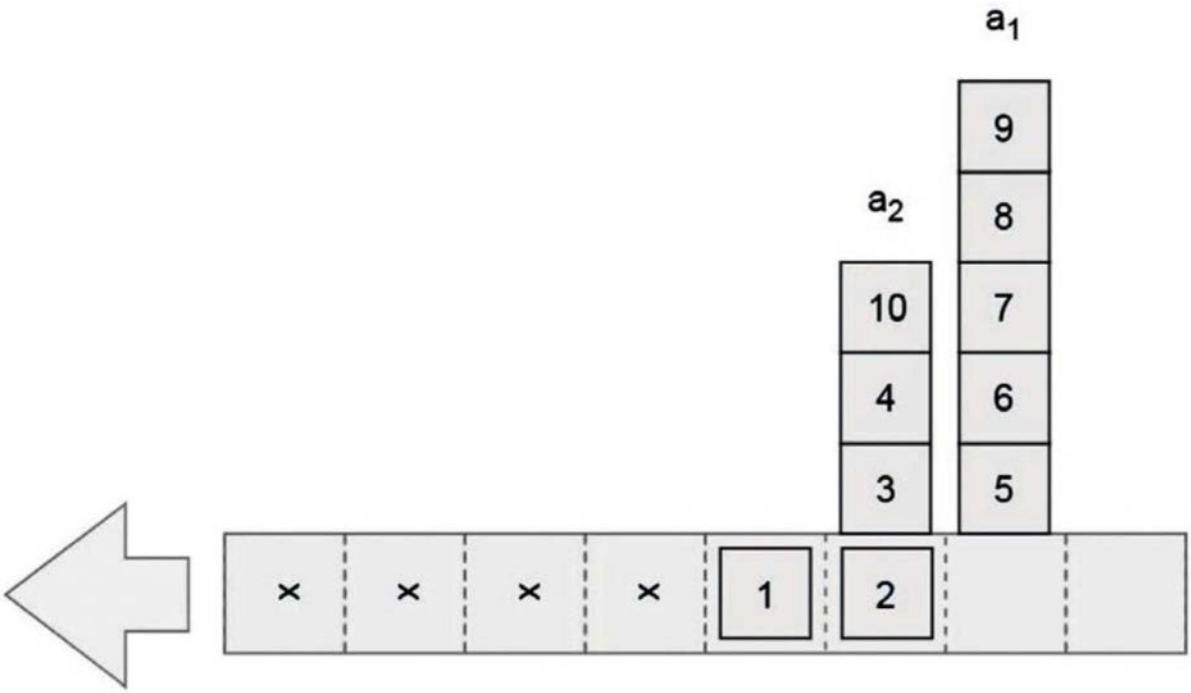


图22

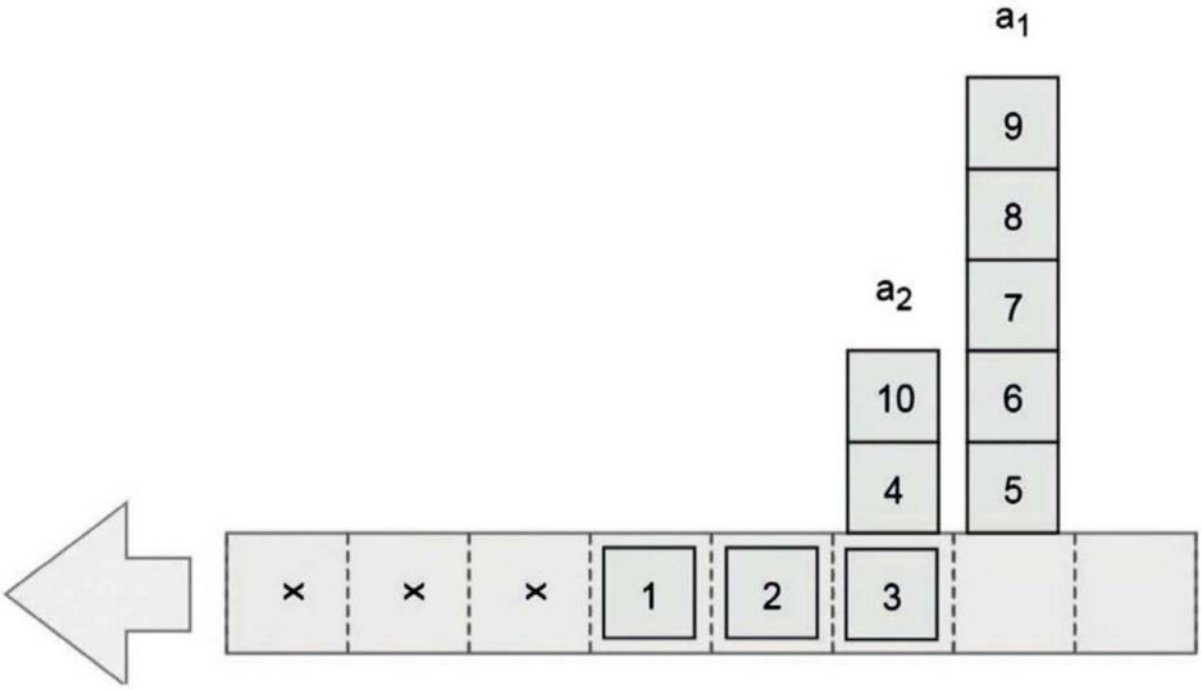


图23

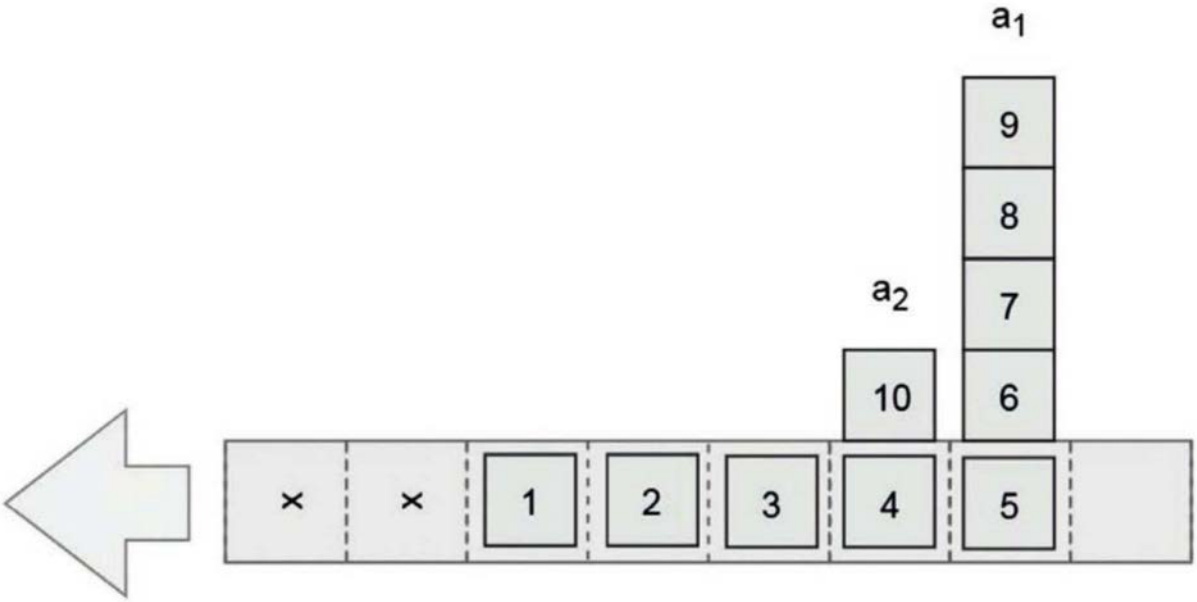


图24

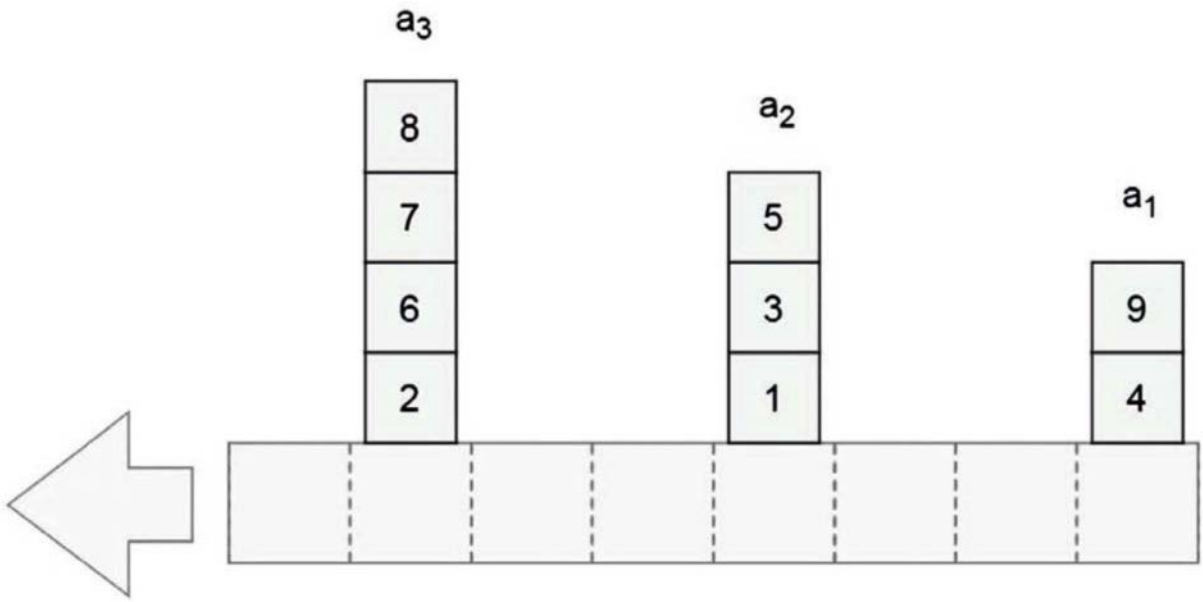


图25



图26

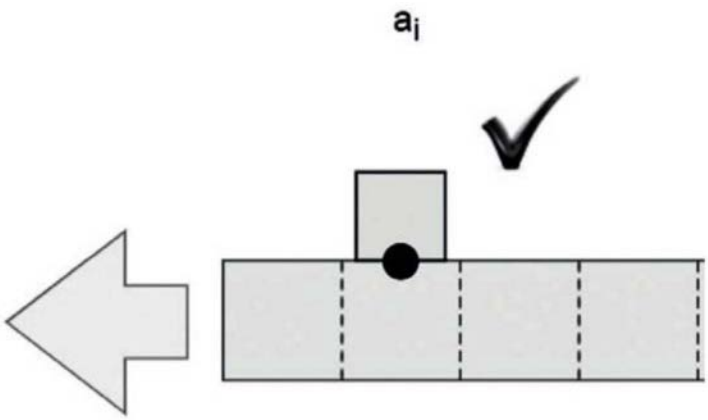


图27

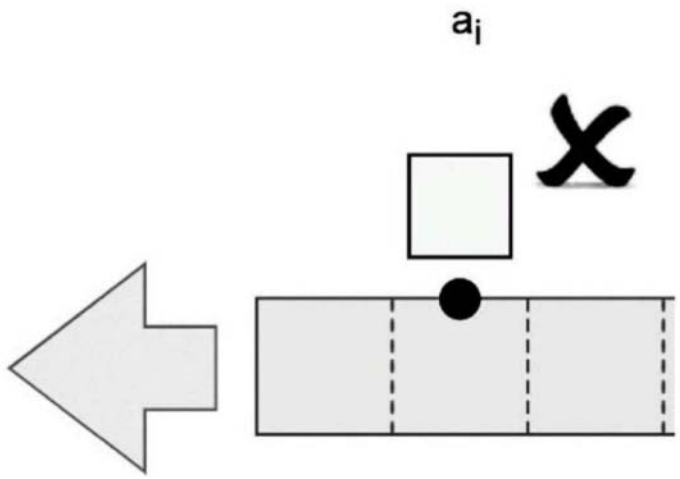


图28

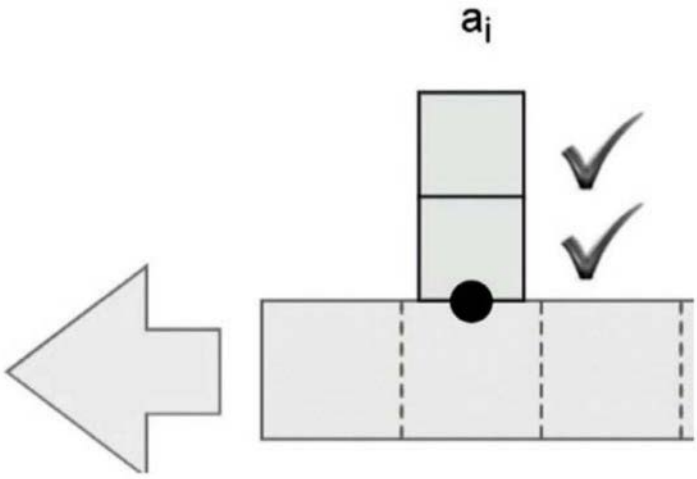


图29

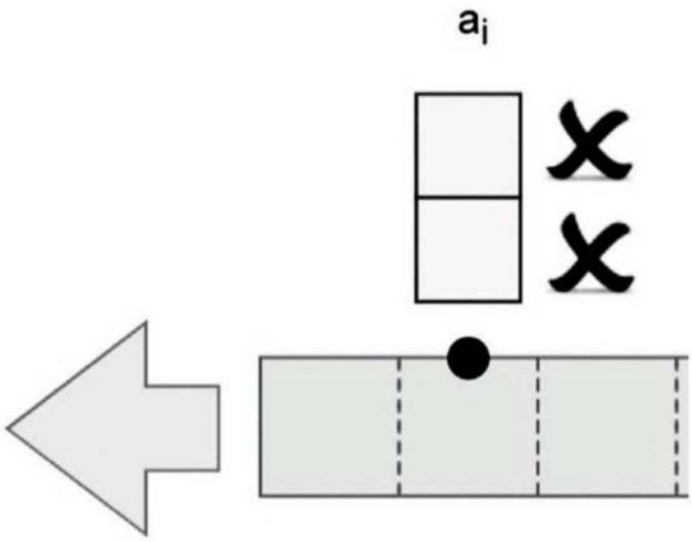


图30

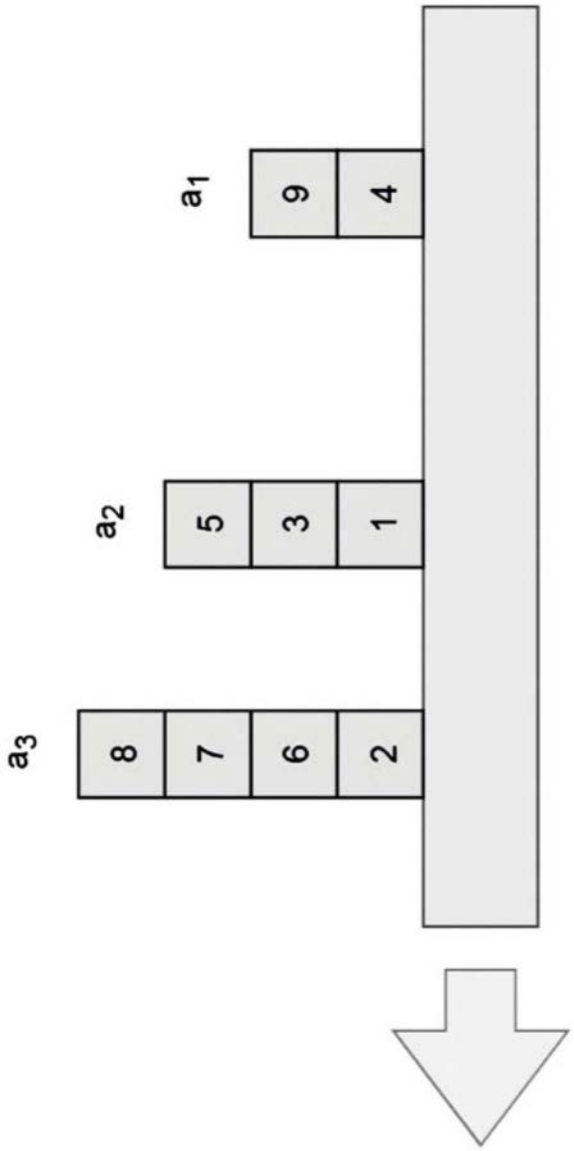


图31

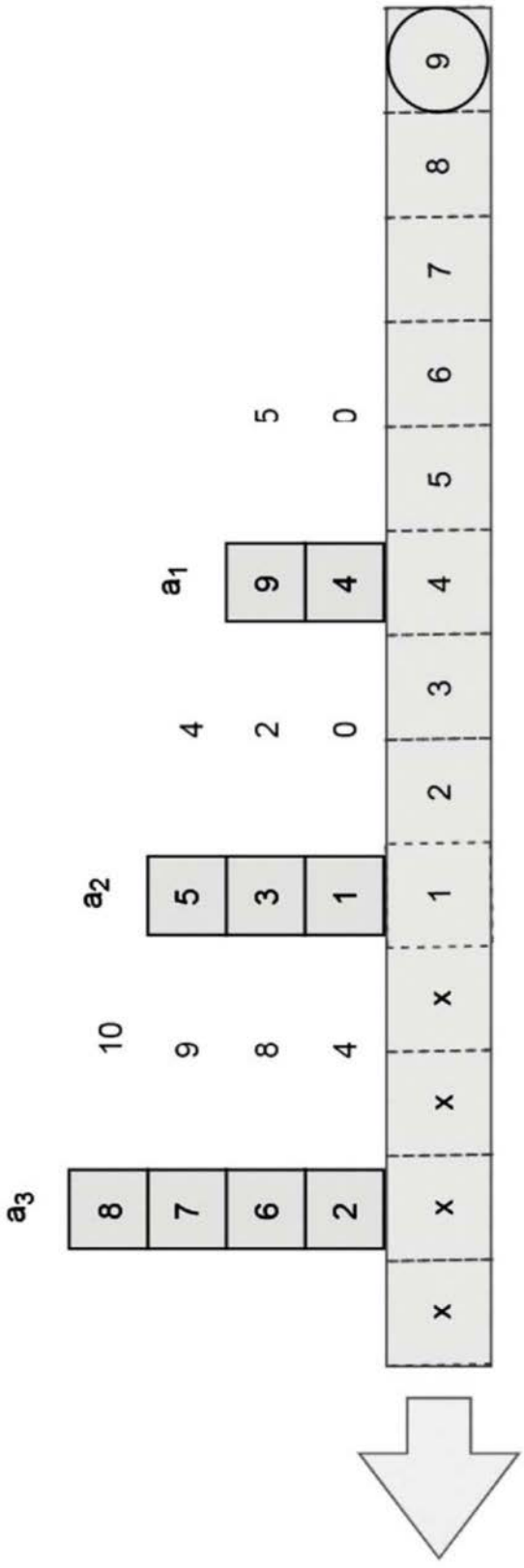


图32

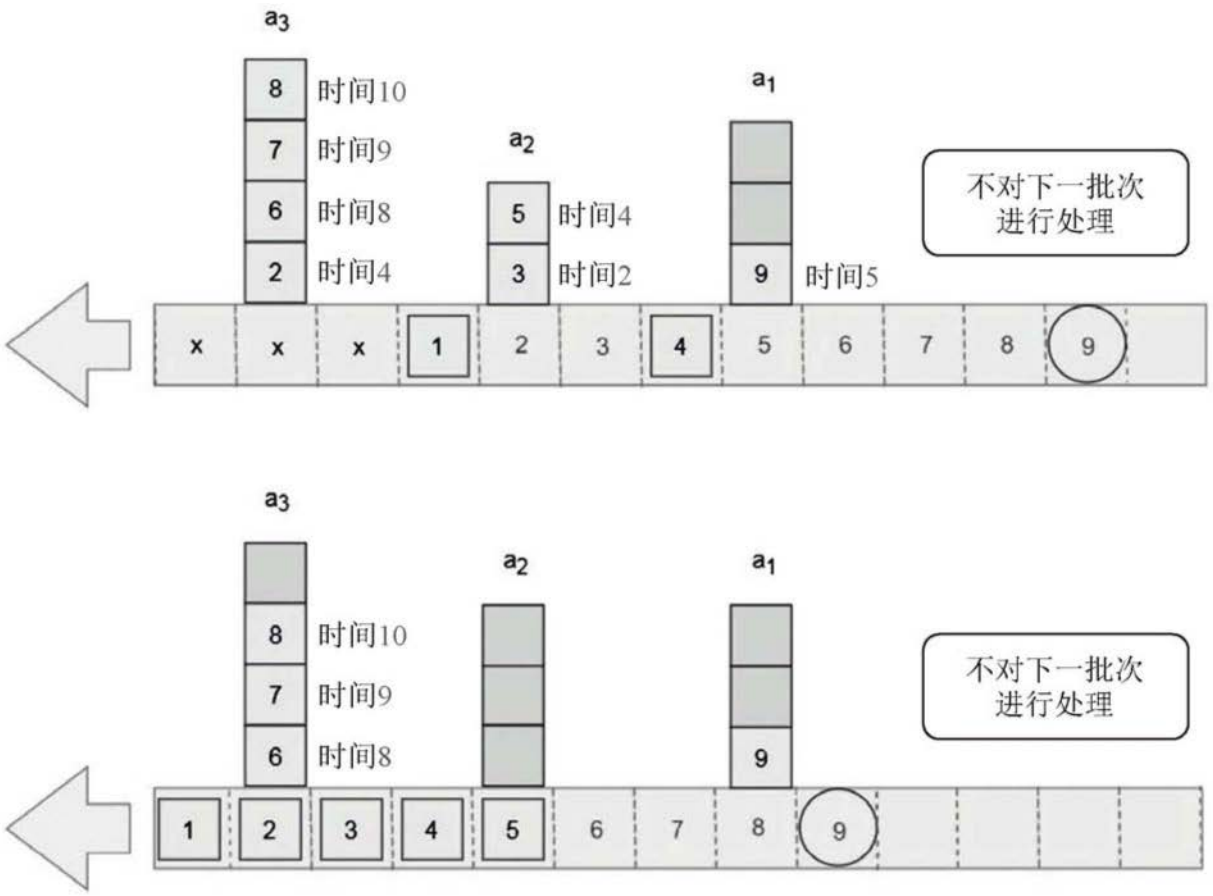

a3
8 时间10
7 时间9
6 时间8
2 时间4
a2
5 时间4
3 时间2
a1
9 时间5
不对下一批次
进行处理
x x x 1 2 3 4 5 6 7 8 9
a3
8 时间10
7 时间9
6 时间8
a2
a1
9
不对下一批次
进行处理
1 2 3 4 5 6 7 8 9

图33

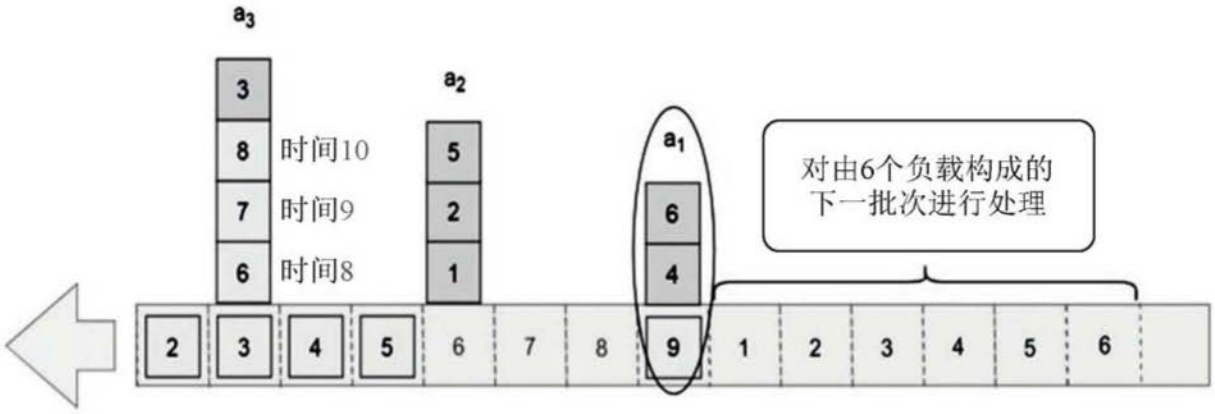

a3
3
8 时间10
7 时间9
6 时间8
a2
5
2
1
a1
6
4
对由6个负载构成的
下一批次进行处理
2 3 4 5 6 7 8 9 1 2 3 4 5 6

图34

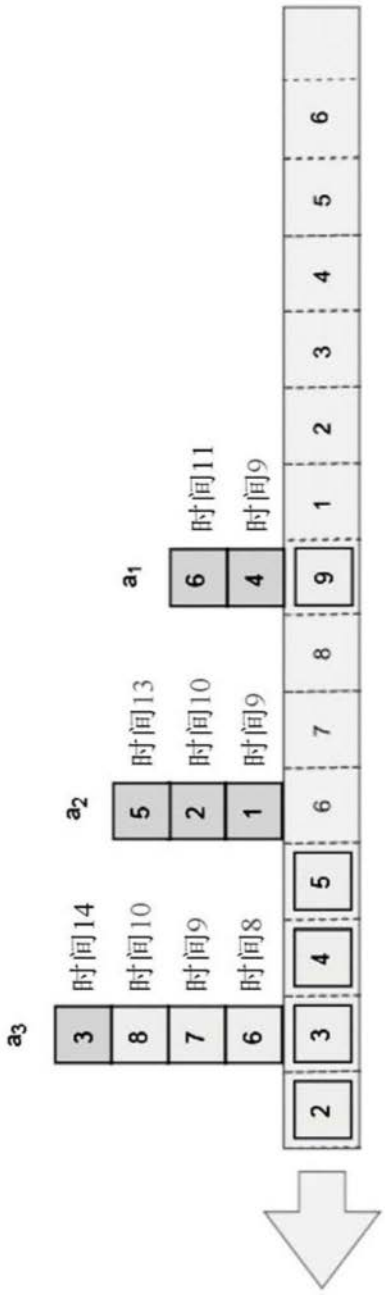


图35

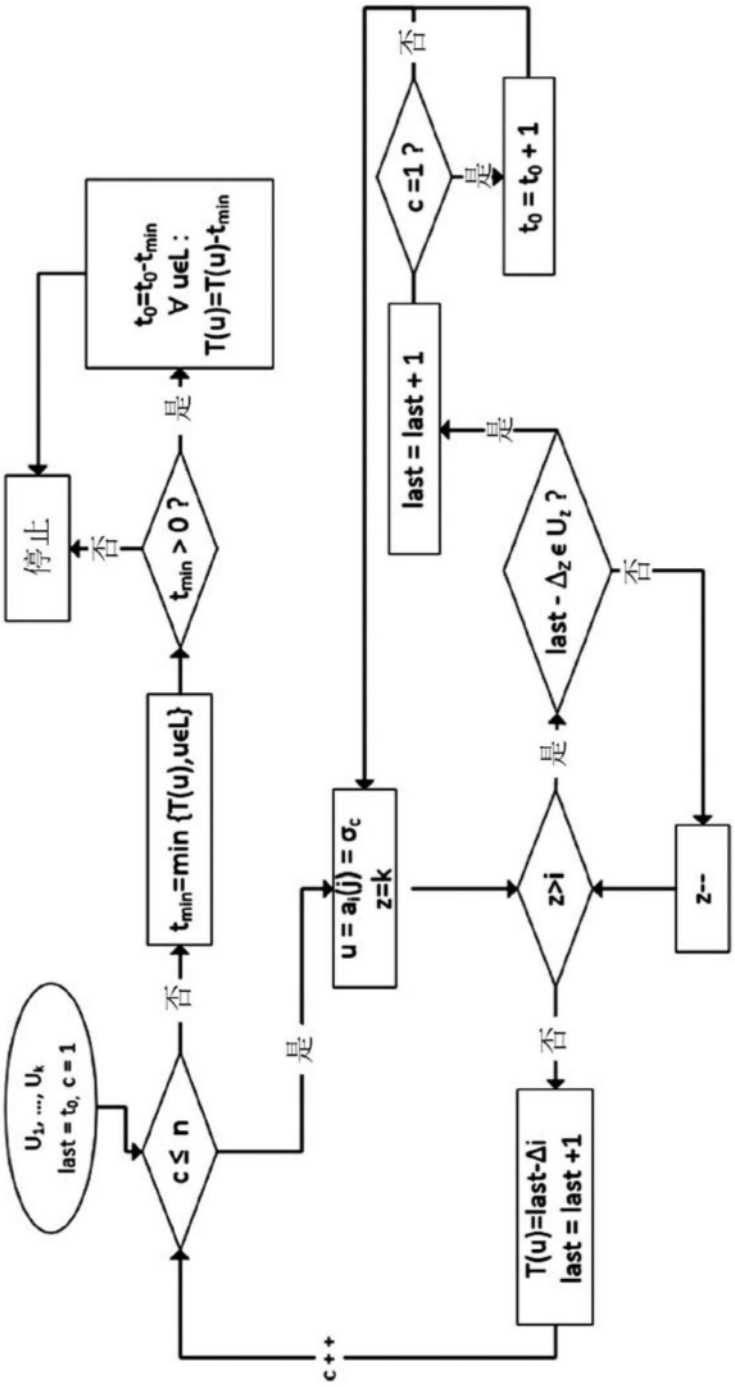

$U_1, \ldots, U_k$
last = t_0, c = 1
c ≤ n
否
是
$t_{min}=min\{T(u),u\in L\}$
$t_{min} > 0$?
否
停止
是
$t_0=t_0-t_{min}$
∀ u∈L :
$T(u)=T(u)-t_{min}$
u = $a_i(j) = \sigma_c$
z=k
c=1 ?
否
是
$t_0 = t_0 + 1$
last = last + 1
last - $\Delta_z \in U_z$?
是
否
z>i
是
否
z--
T(u)=last-Δi
last = last +1
c++

图36

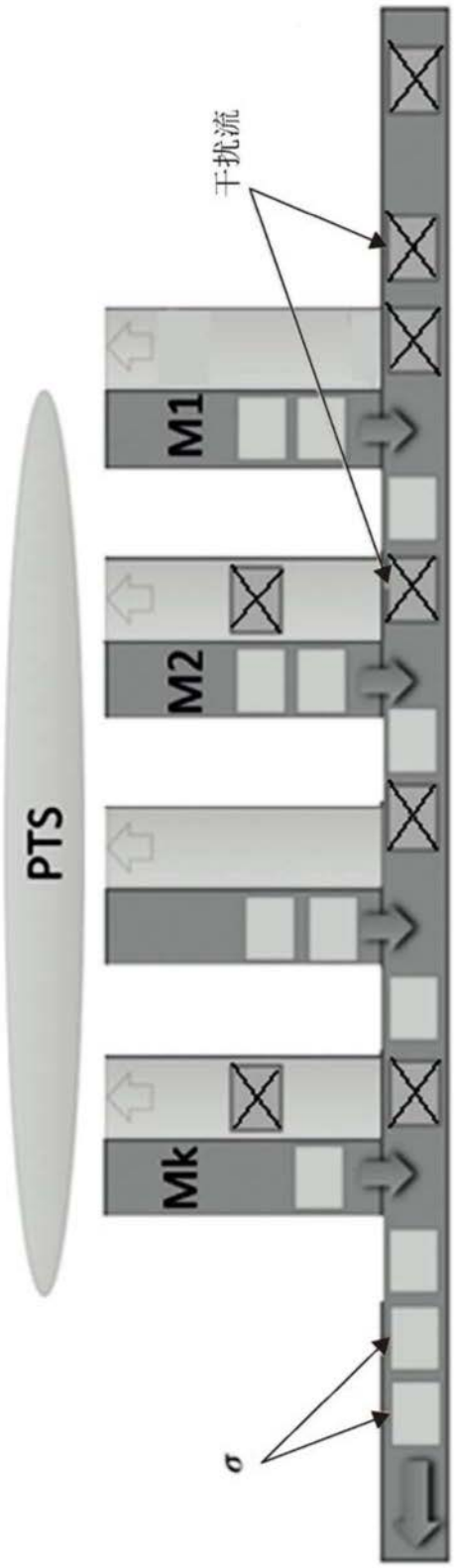


图37

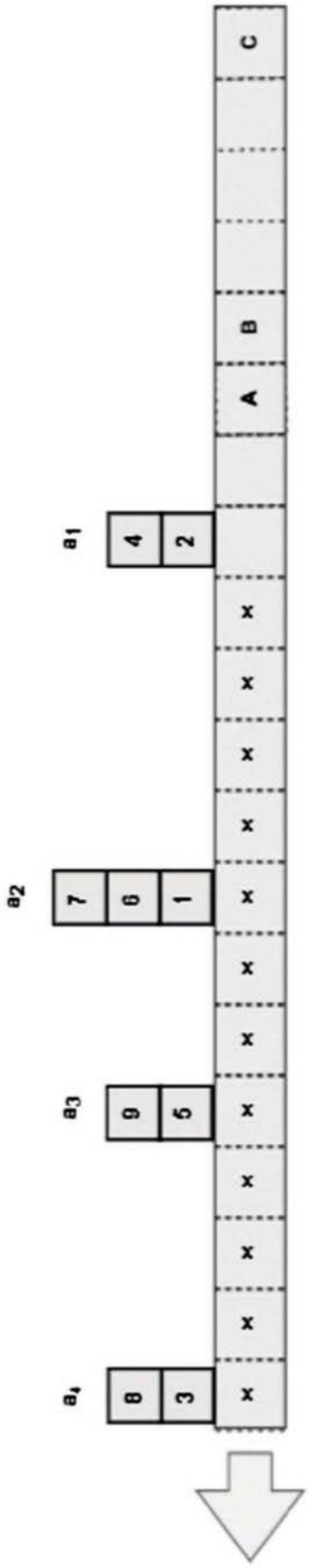


图38

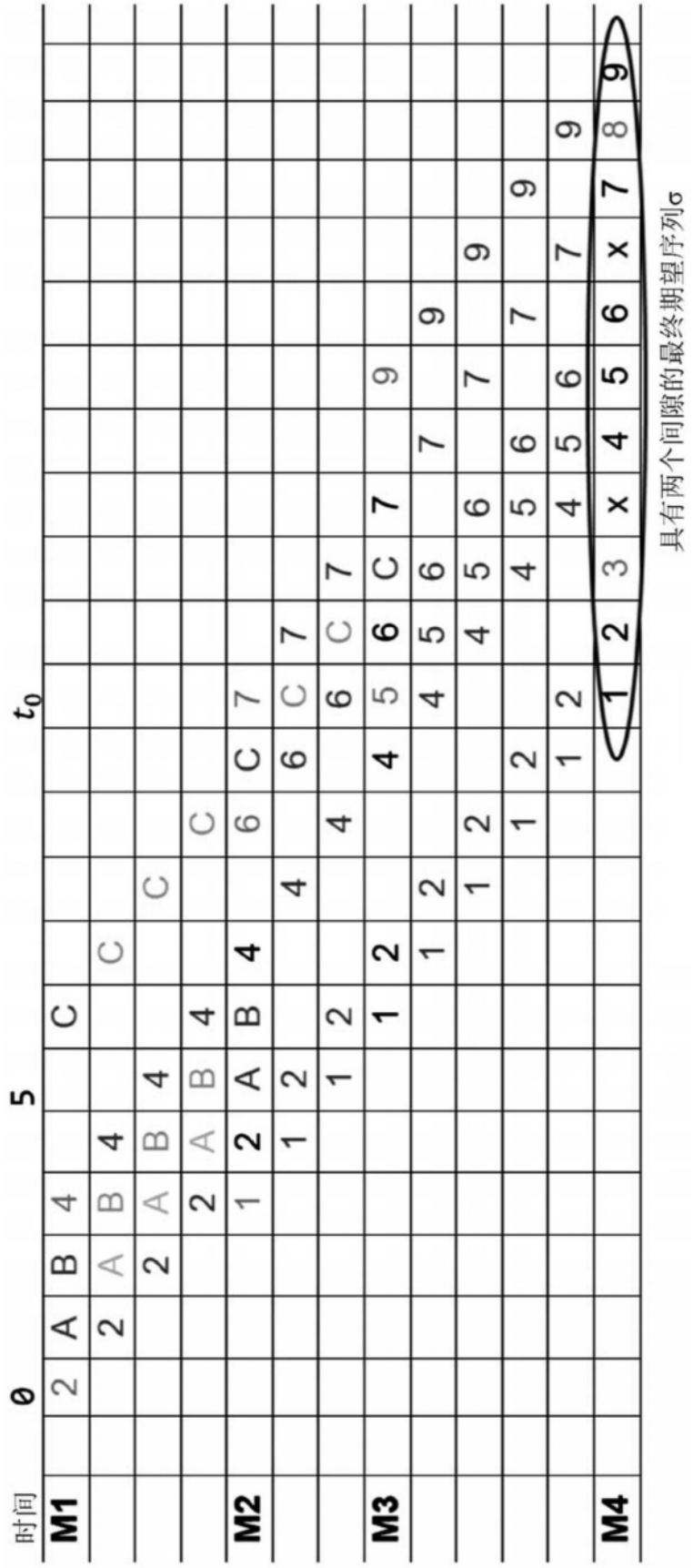


图39

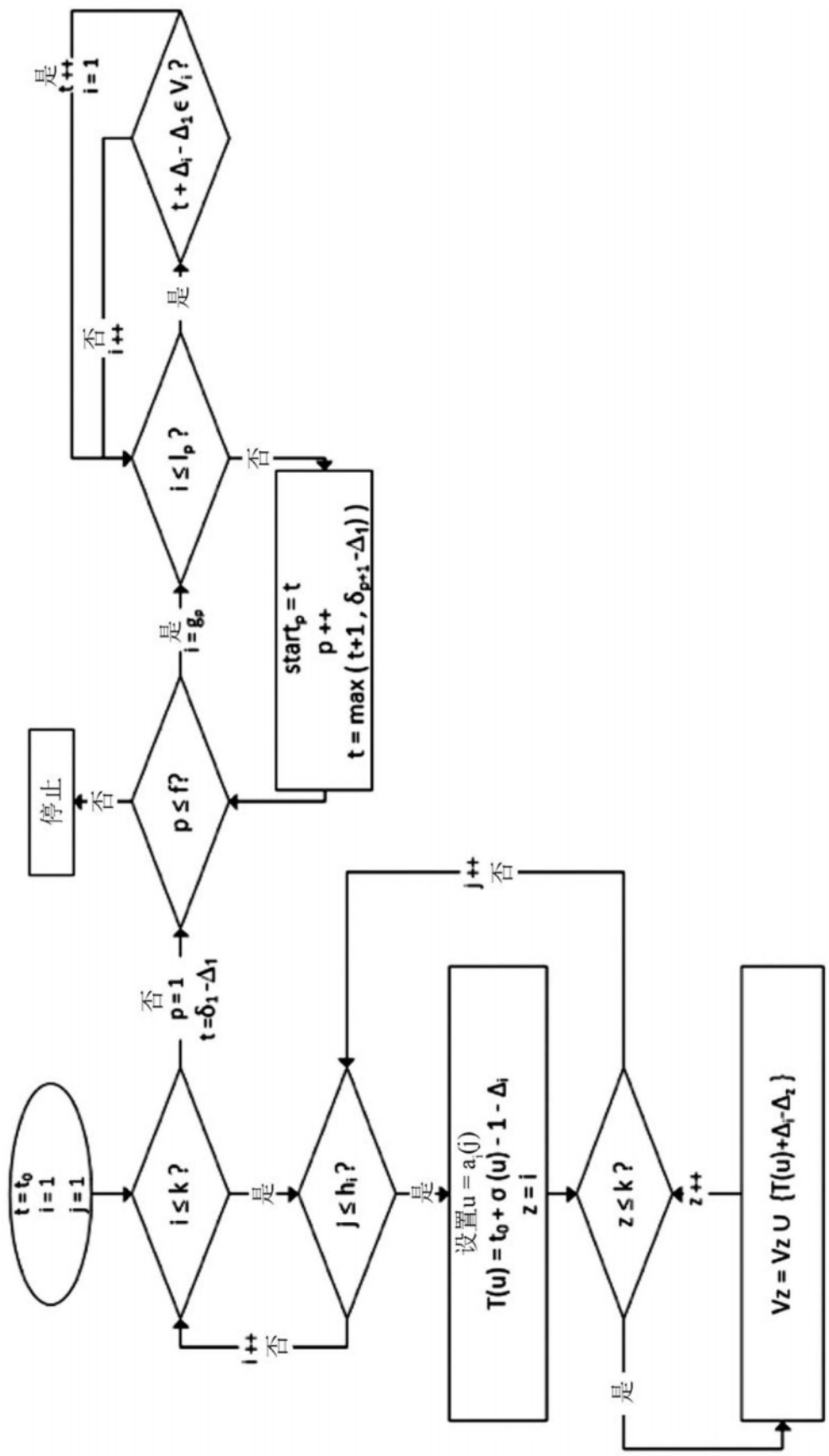

t=t0
i=1
j=1
i≤k?
是
否
p=1
t=δ1-Δ1
j≤hi?
是
否
i++
设置u=a(j)
T(u)=t0+σ(u)-1-Δi
z=i
z≤k?
是
否
z++
Vz=Vz∪{T(u)+Δz-Δt}
j++
否
p≤f?
是
否
停止
i=gp
ip≤lp?
是
否
i++
t+Δi-Δ1∈Vi?
是
否
t++
i=1
startp=t
p++
t=max(t+1,δp+1-Δ1))

图40

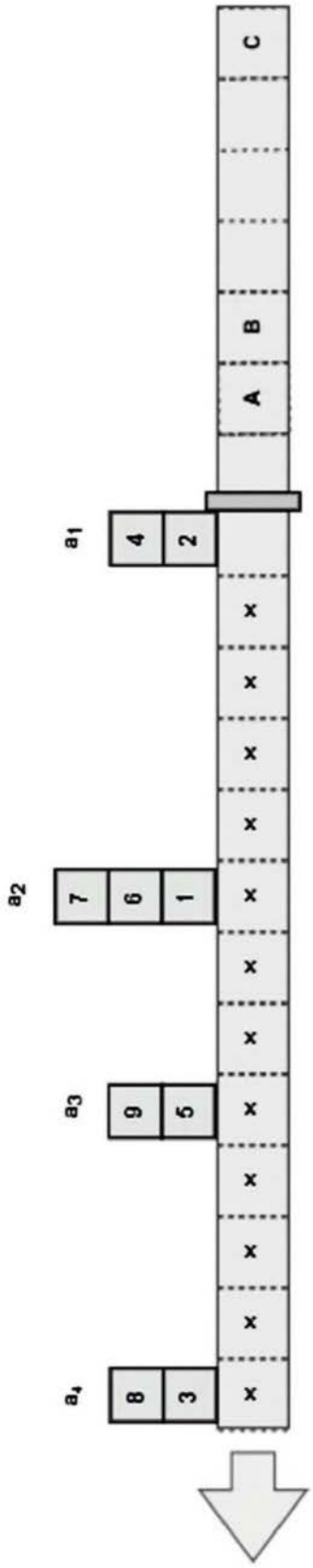


图41

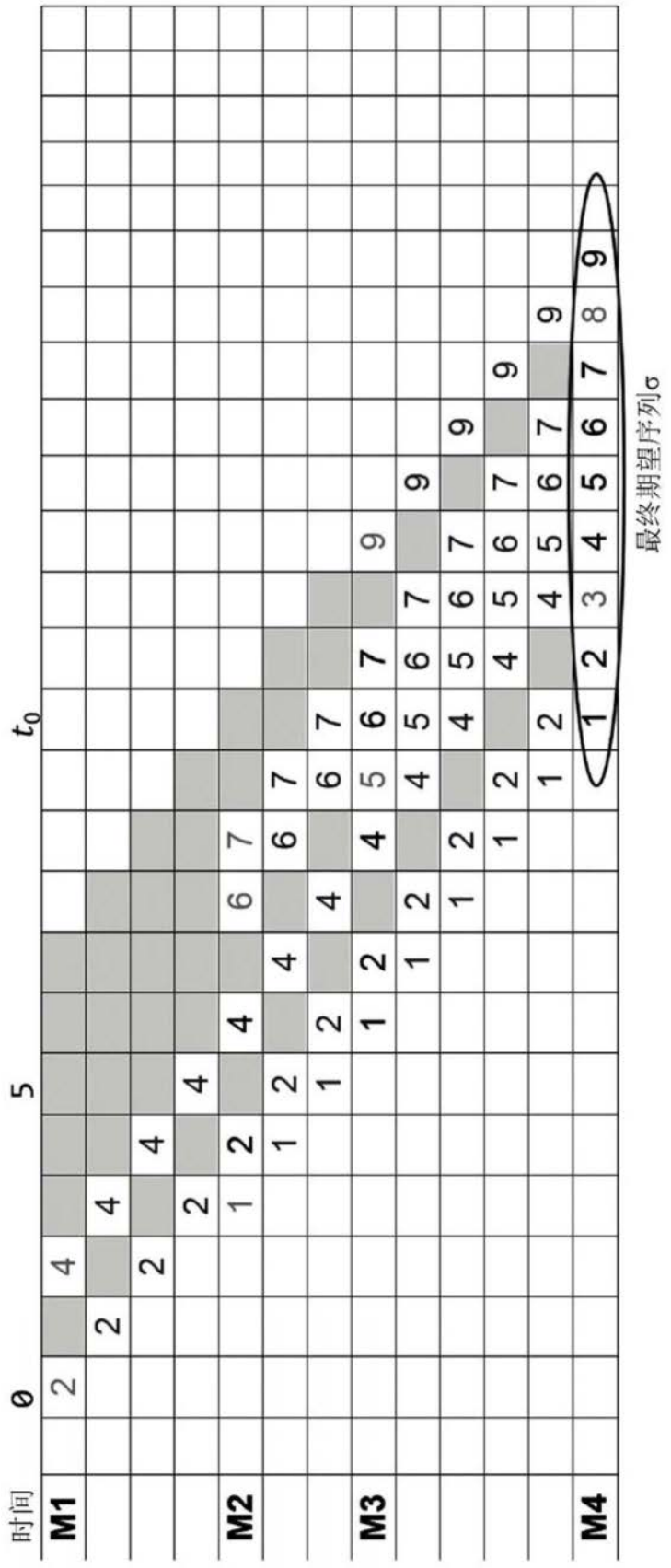


图42

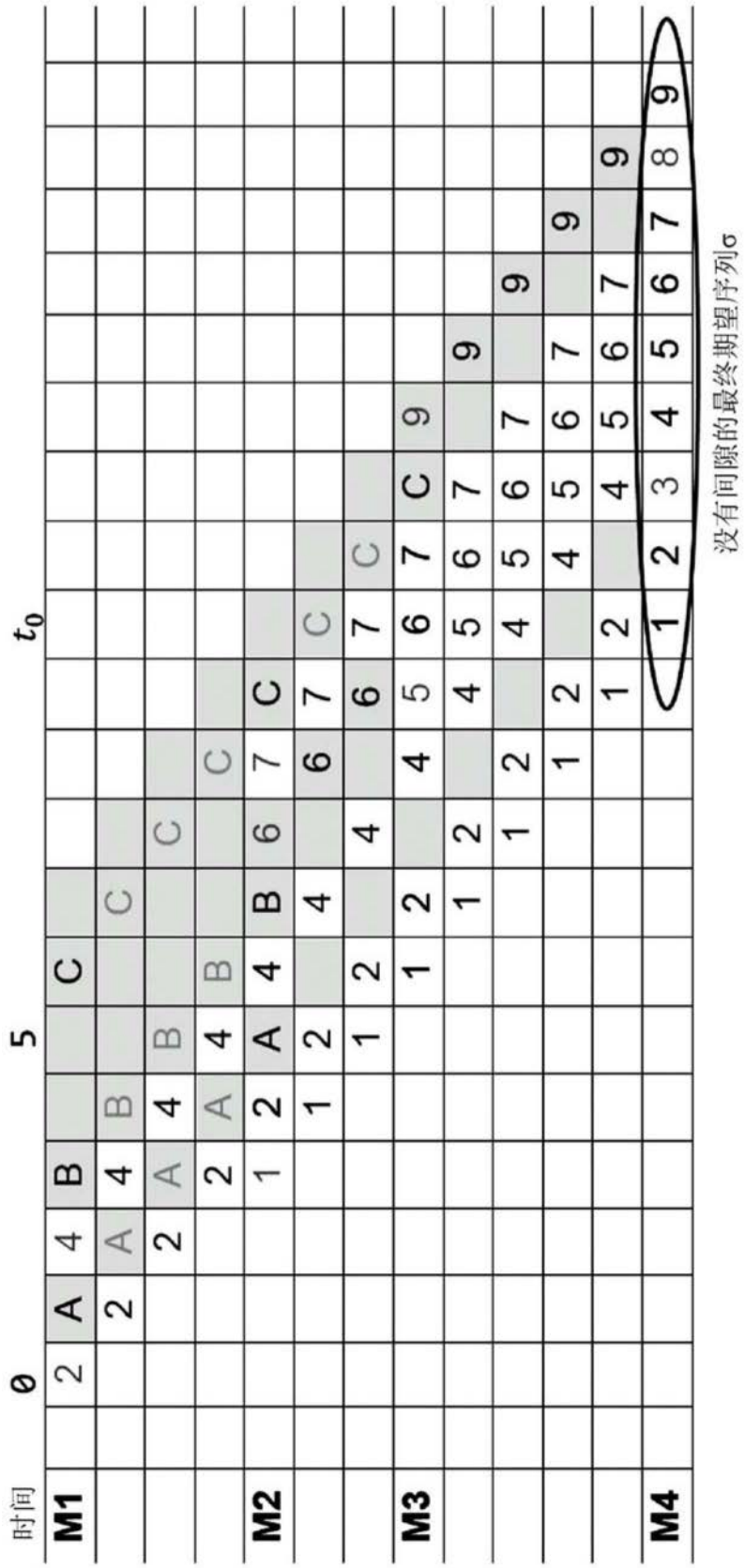


图43

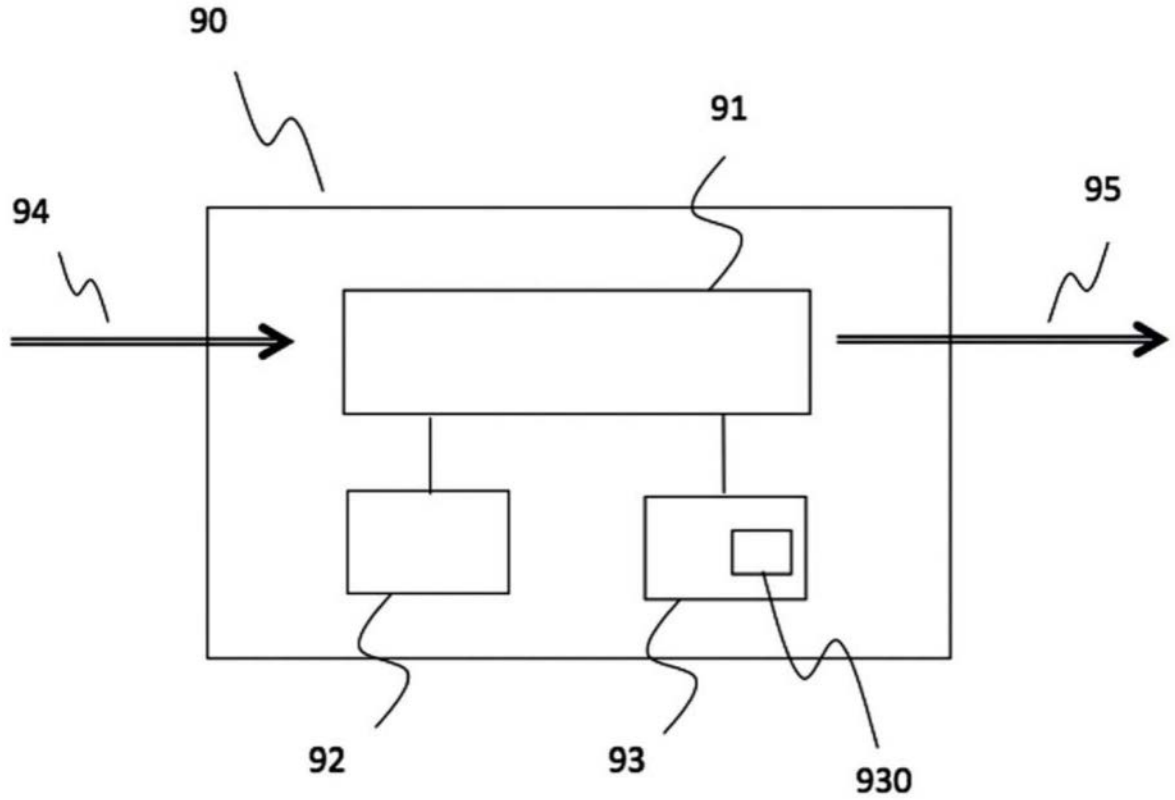


图44

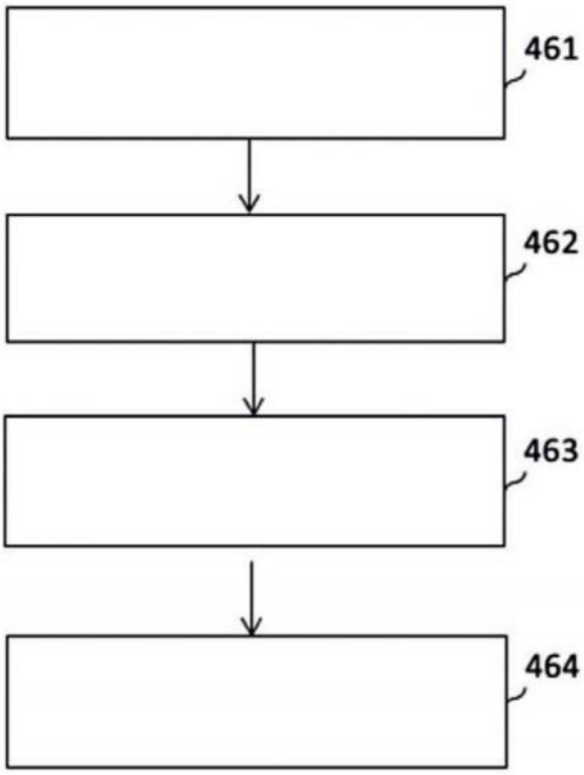


图46

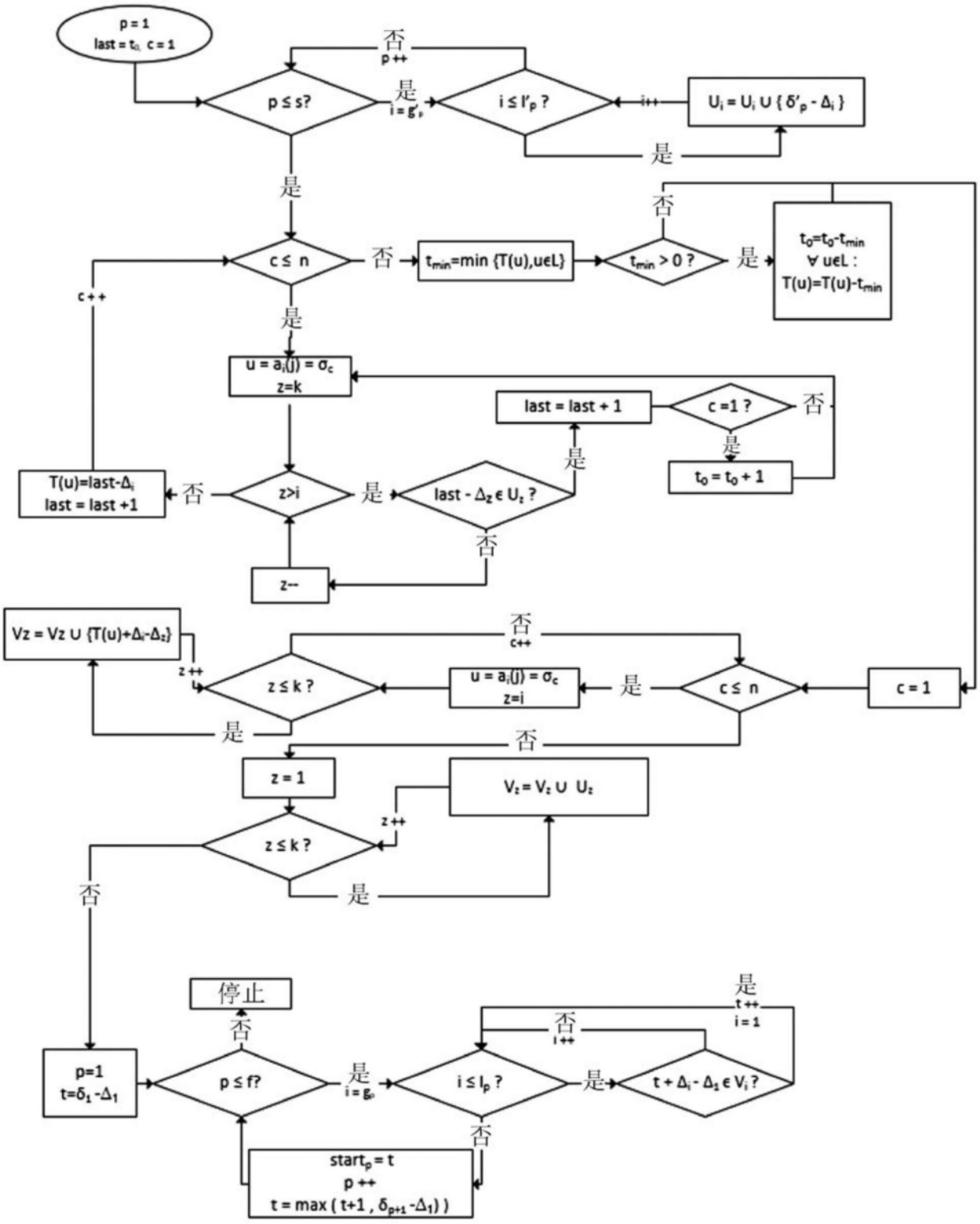

p = 1
last = t0, c = 1
否
p++
p ≤ s?
是
i = g'p
i ≤ l'p ?
i++
Ui = Ui ∪ { δ'p - Δi }
是
是
否
c ≤ n
否
tmin=min {T(u),uϵL}
tmin > 0 ?
是
t0=t0-tmin
∀ uϵL :
T(u)=T(u)-tmin
c++
是
u = ai(j) = σc
z=k
last = last + 1
c =1 ?
否
是
t0 = t0 + 1
T(u)=last-Δi
last = last +1
否
z>i
是
last - Δz ϵ Uz ?
是
否
z--
Vz = Vz ∪ {T(u)+Δi-Δz}
否
c++
z++
z ≤ k ?
u = ai(j) = σc
z=i
是
c ≤ n
是
c = 1
是
否
z = 1
Vz = Vz ∪ Uz
z++
z ≤ k ?
否
是
停止
否
是
t++
i = 1
否
i++
p=1
t=δ1 -Δ1
p ≤ f?
是
i = gp
i ≤ lp ?
是
t + Δi - Δz ϵ Vi ?
否
startp = t
p ++
t = max (t+1 , δp+1 -Δ1))

图45

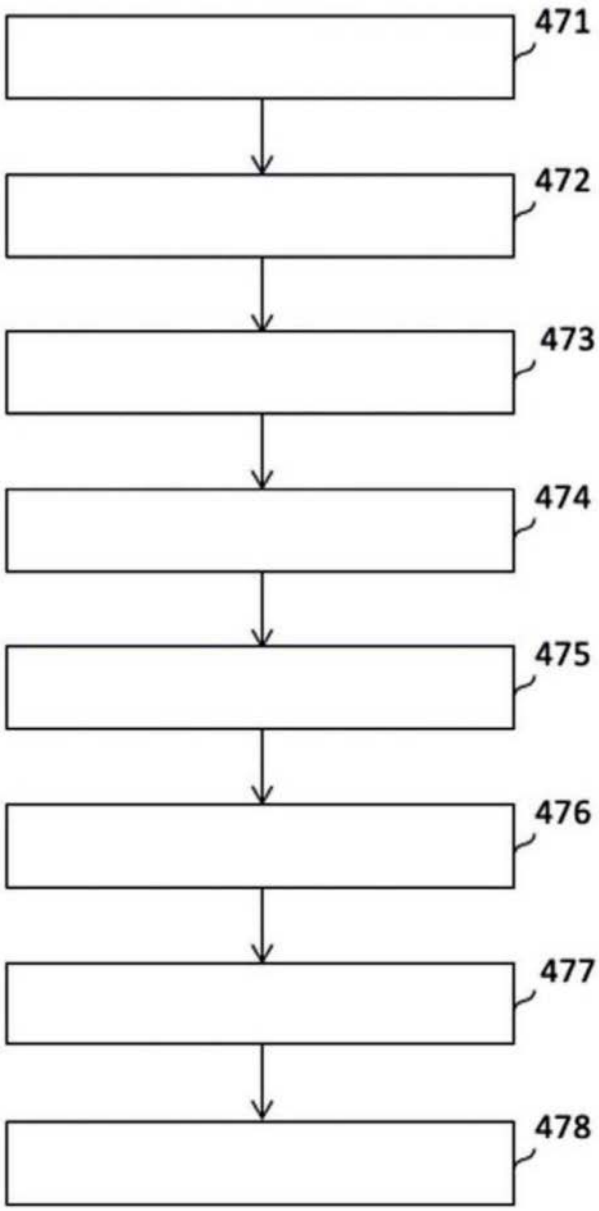


图47

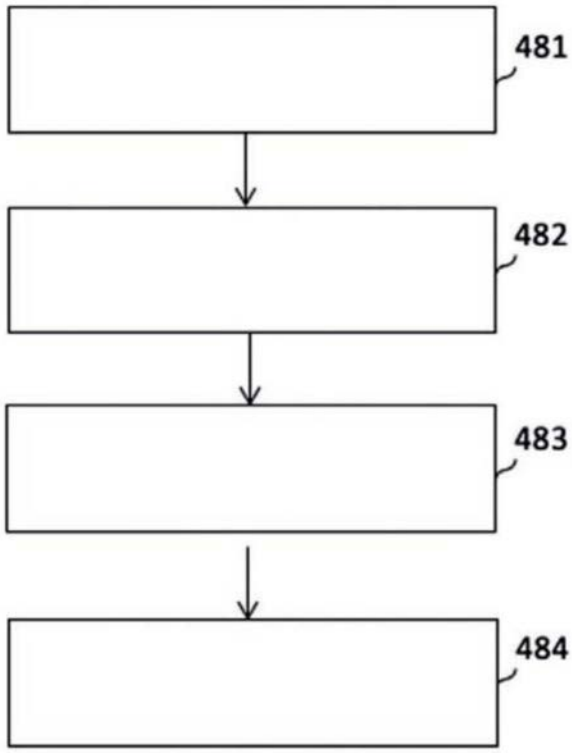


图48

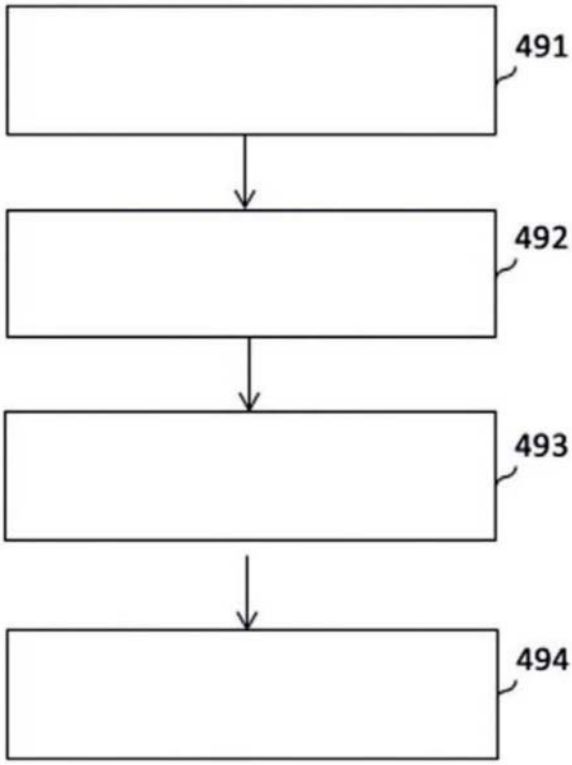


图49

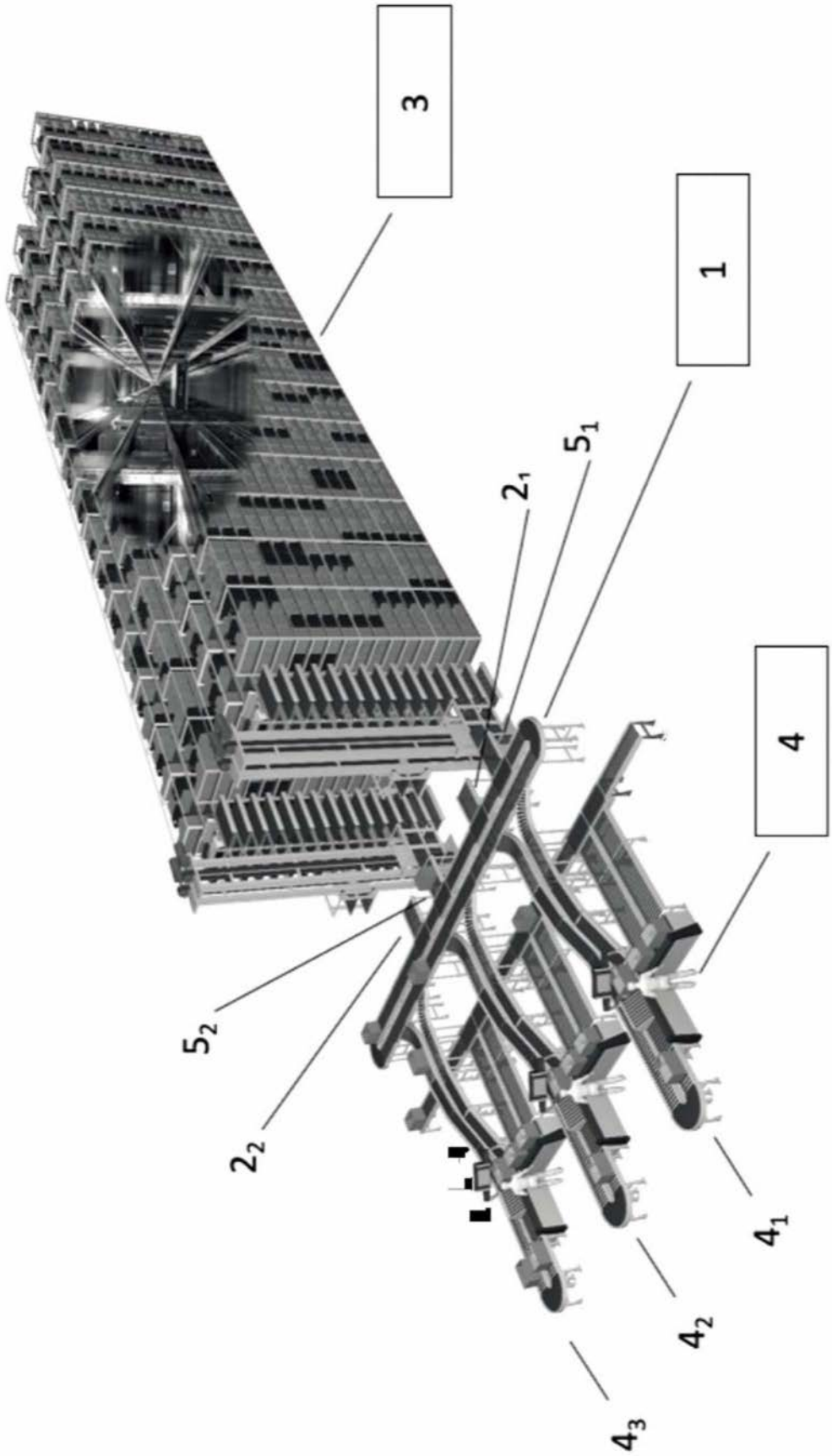


图50