

在拉了整整一周电线后，我又患上了无可救药的流水线强迫症。

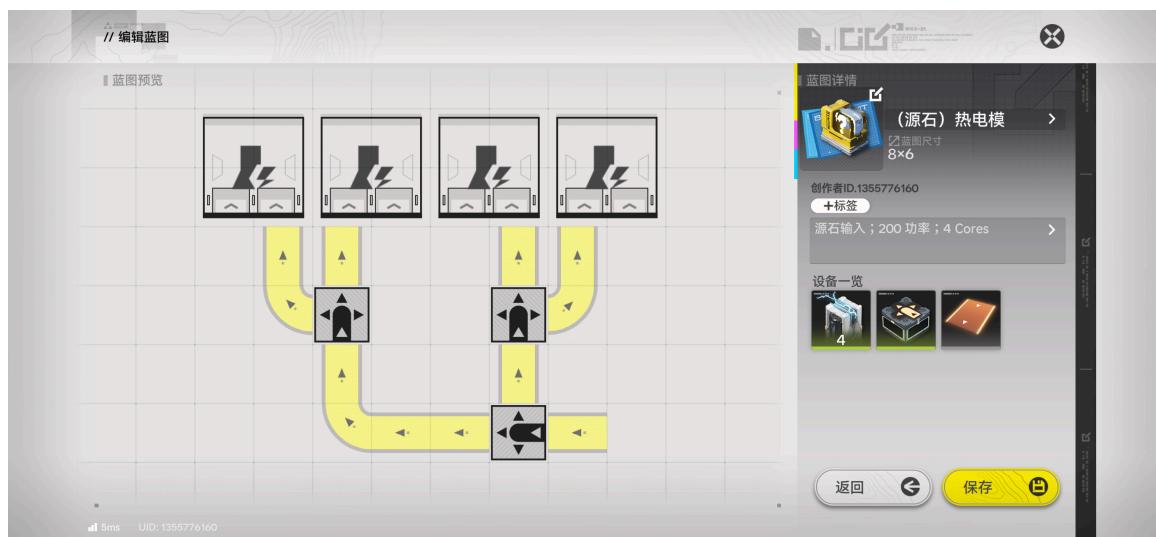
## 1. 引言

流水线配平是一个很实际的需求（尤其对我这种强迫症来说）。为配平的流水线会造成阻塞、爆仓或机器空闲。

这篇文章将系统地讨论 AIC 中传送带速率定制相关的方法。具体而言，我们要解决的核心问题是：**对于某个给定速率的输入，如何得到某种倍率的输出速率？**

我们先来看几个例子。

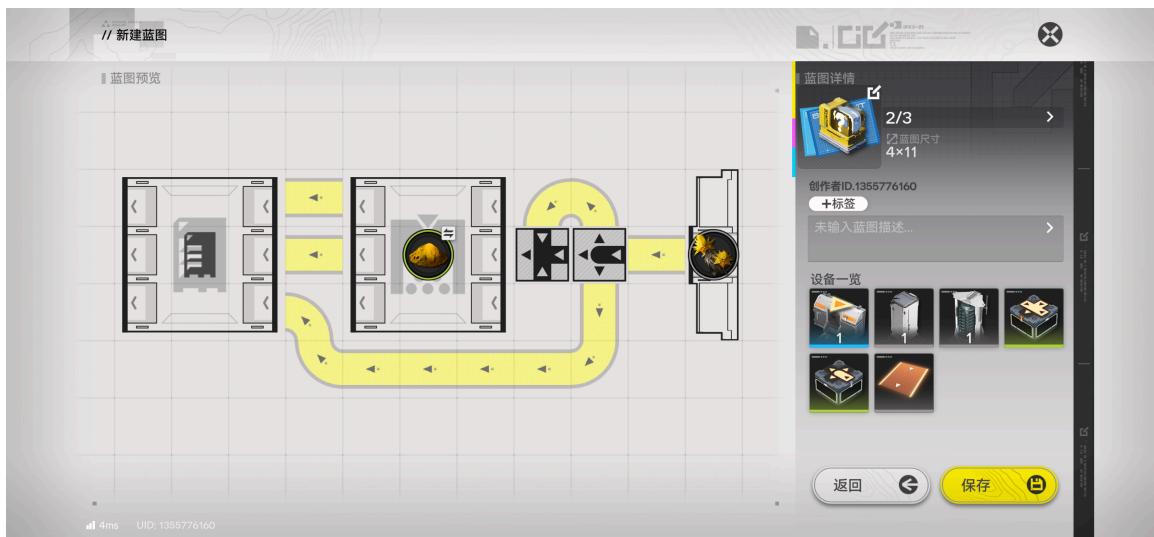
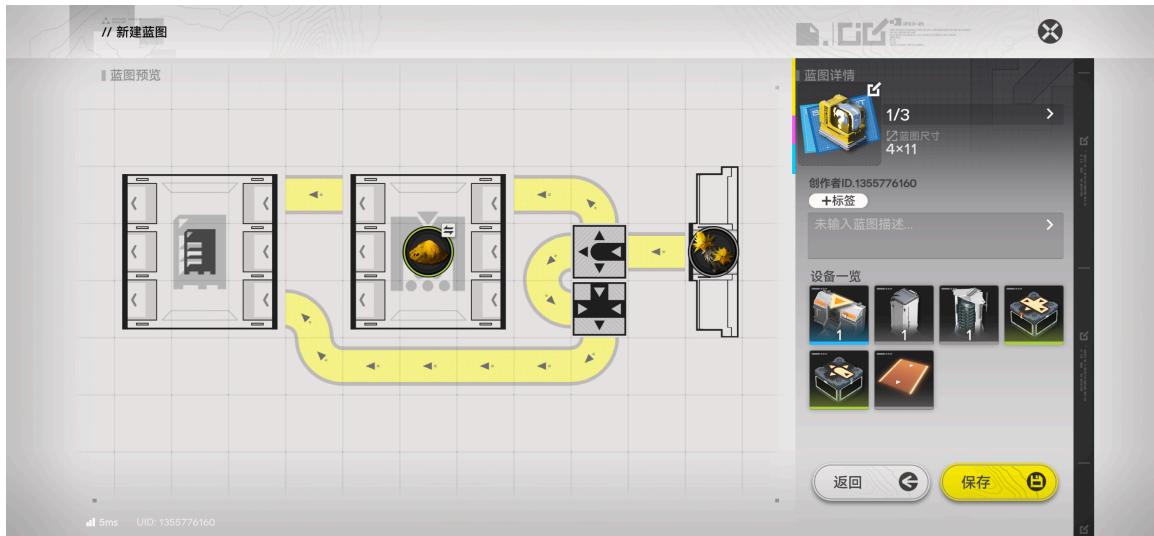
**热能池（源矿供电）** 当以源矿作为输入的时候，热能池消耗一个源矿需要 8 s。这意味着我们需要以  $1/8$  items/s 的速率向它供货。幸运的是，这相当好实现。鉴于游戏中默认的传送带速率是  $v = 0.5$  items/s， $1/8$  正好是它的  $1/4$ ，于是我们只需要进行三次分流，就可以得到目标速率了，具体而言<sup>1</sup>：



**砂叶粉末** 砂叶制备砂叶粉末有着  $1 : 3$  的配方量比，而下游对砂叶粉末的需求一般不是 3 的整倍数，此时这也是一个类似的简单的分流问题，如图所示，我们可以轻松得到  $1/6$  ips 和  $1/3$  ips 两种生产速率，以 0.5 ips 为精度控制产出速率，如图所示<sup>2</sup>。

<sup>1</sup>实际上能继续压缩，但这只是个示意图，就不必在意了。

<sup>2</sup>图中的  $1/3$  和  $2/3$  是原输出和现输出的配方速率之比



**热能池 (电池供电)** 当以任意种类的电池作为输入的时候，热能池消耗一个电池需要 40 s. 这意味着我们需要以  $1/40 \text{ items/s}$  的速率向它供货<sup>3</sup>. 这看起来就不简单了. 事实上，这是做不到的，我们将在 小节 3.2 进行讨论.

## 2. 基本量与基本单位

虽然刚刚我们已经见识过了，但在这里还是要重新地声明一下我们使用的单位和记号：

**配方量** 配方量是指一台机器完成一次生产时所需/所产出的某种物品的数量，用符号  $n$  表示，无量纲. 下文中，配方量总是针对某一个特定的原材料或产物而言的. 例如，生产中容谷地电池时，封装机对于通用铁质零件的配方量是  $n = 10$ ，对于源矿粉末的配方量  $n = 15$ ，对于中容谷地电池的配方量是  $n = 1$ .

**配方速率** 配方速率指的是单台机器对某种物品的理论处理速率，由配方量  $n$  和生产周期时长  $t_c$  决定，使用  $v_r$  表示. 下文中，配方速率总是针对一个特定的原材料或产物而言的. 在本文中，我们将使用  $\text{items/second}$  (以下简称 ips) 作为速率的单位. 例如对于热能池，当消耗源矿的时候：

<sup>3</sup>好吧，其实按照游戏的设计，封装及产出电池的速率正好是  $1/10 \text{ ips}$ ，而只要进行  $1/4$  分流后就能得到目标速率了. 不过，如果我们就此停止就不会有这篇文章了，所以还是让我们继续吧 🎉🎉🎉

$$v_r = \frac{n}{t_c} = \frac{1}{8} \text{ ips}$$

**生产周期时长** 每台机器完成一次生产都会花费一个确定的时间，这个时间称作生产周期时长，用  $t_c$  表示。例如对于热能池，当消耗源矿的时候， $t_c = 8 \text{ s}$ ；当消耗任意种类的电池的时候， $t_c = 40 \text{ s}$ 。

**传送带速率** 传送带速率指的是物品在传送带上实际流动的速率，用  $v_c$  或  $v$  表示，单位同为 ips。在游戏中，一级传送带的标准速率上限为  $v_{\max} = 0.5 \text{ ips}$ ，这是我们进行所有速率定制操作的基准。

以上前三者满足一个显而易见的关系：

$$v_r t_c = n$$

### 3. 速率定制

所谓速率定制，就是对于任意给定的  $v_t$ ，我们希望能使用可接受数量的分流器和汇流器得到一条以这样的速度输出的传送带。

#### 3.1. 基本原件的作用

##### 3.1.1. 分流器

分流器有一个输入口，三个输出口，其中输出口可以空接。根据实验和观察，一个分流器的工作方式是：以某个固定的顺序轮询输出口，并尝试输出，且这种轮询顺序与最初连接输出口的顺序相关（具体的关系我还没有得到）。这其实是一个有限状态机。

如果将时间拉长，以统计的角度看，分流器会将输入均分到每个输出上，这意味着**均分传送带速率**<sup>4</sup>。分流器只存在以下三种连接方式，即(1)只连接一个输出口(2)只连接两个输出口(3)接满三个输出口。假设输入的传送带速率是  $v_{\text{in}}$ ，则在三种情况中，(1)是平凡的，(2)和(3)则分别得到了两条  $v_{\text{out}} = v_{\text{in}}/2$  和三条  $v_{\text{out}} = v_{\text{in}}/3$  的传送带分支。

这实际上是某种乘法，而我们拥有的只有  $1/2$  和  $1/3$  这两个因子。

##### 3.1.2. 汇流器

汇流器有三个输入口，一个输出口，其中输入口可以空接。根据实验和观察，我发现一个分流器的工作方式是：这依然一个有限状态机。

这样的机制意味着**传送带速率之和**，即假设第  $i$  个输入传送带的传送带速率是  $v_i$ ，则有

$$v_{\text{out}} = \max \left( v_{\max}, \sum_i v_i \right)$$

#### 3.2. 数学原理

结合分流器和汇流器，我们拥有了两种基本运算：

**乘法** 可以将一条传送带的速率乘以  $1/2$  或  $1/3$ （通过分流器）。

**加法** 可以将多条传送带的速率求和（通过汇流器），但结果不能超过  $0.5 \text{ ips}$ 。

我们的目标是从基础速率  $v_0 = 0.5 \text{ ips}$  出发，通过有限次上述运算，得到一个尽可能接近目标速率  $v_t$  的输出速率。

---

<sup>4</sup>然而实际上，由于这是一个有限状态机，它轮询的顺序、相位不能忽视，这些因素会导致短期内的波动，我们将在小节 4 讨论这个问题。如果不加任何说明，下文涉及的所有速率都是长时间统计意义上的平均速率。

### 3.2.1. 可表达的速率形式

每次使用分流器进行乘法操作，相当于给当前速率乘上一个  $1/2$  或  $1/3$  的因子。如果我们用  $m$  次  $1/2$  乘法， $n$  次  $1/3$  乘法，那么从  $v_0$  可以得到一个形如

$$\left(\frac{1}{2}\right)^m \left(\frac{1}{3}\right)^n v_0 \quad (m, n \in \mathbb{N})$$

的中间速率。我们称这样一个由  $m$  和  $n$  确定的项为一个基。

汇流器允许我们将多个这样的基的速率相加。因此，通过组合有限个基并进行求和，我们能得到的所有速率  $v_{\text{out}}$  都具有以下形式：

$$v_{\text{out}} = v_0 \sum_{m,n \in \mathbb{N}} c_{m,n} \left(\frac{1}{2}\right)^m \left(\frac{1}{3}\right)^n$$

其中， $c_{m,n} \in \mathbb{N}$ ，每个分支由一对非负整数  $(m, n)$  描述。

为了简化讨论，我们令  $f = v_{\text{out}}/v_0$ ，即归一化到基础速率的比例系数。那么：

$$f = \sum_{m,n \in \mathbb{N}} c_{m,n} \left(\frac{1}{2}\right)^m \left(\frac{1}{3}\right)^n$$

$f$  的取值范围在  $[0, 1]$  之间。

因此，所有由有限个  $\left(\frac{1}{2}\right)^m \left(\frac{1}{3}\right)^n$  相加得到的有理分数都可以被表达，这样的分数的共性是其分母必然是  $2^m 3^n$  的形式。显然，开头提到的  $1/40$  不在这个范围内，因为其含有  $1/5$  这个不能被  $2, 3$  整除的因子。

### 3.2.2. 精度与层数限制

首先，我们规定  $c_{m,n}$  的取值。观察我们刚刚写出的和式，这看起来像二进制和三进制的某种混合进制。再者，由于  $c_{m,n}$  是每一项的系数，如果它太大，会造成麻烦的约分或者需要复制多份相同传送带的情况，因此  $c_{m,n}$  尽可能小些是最好的。

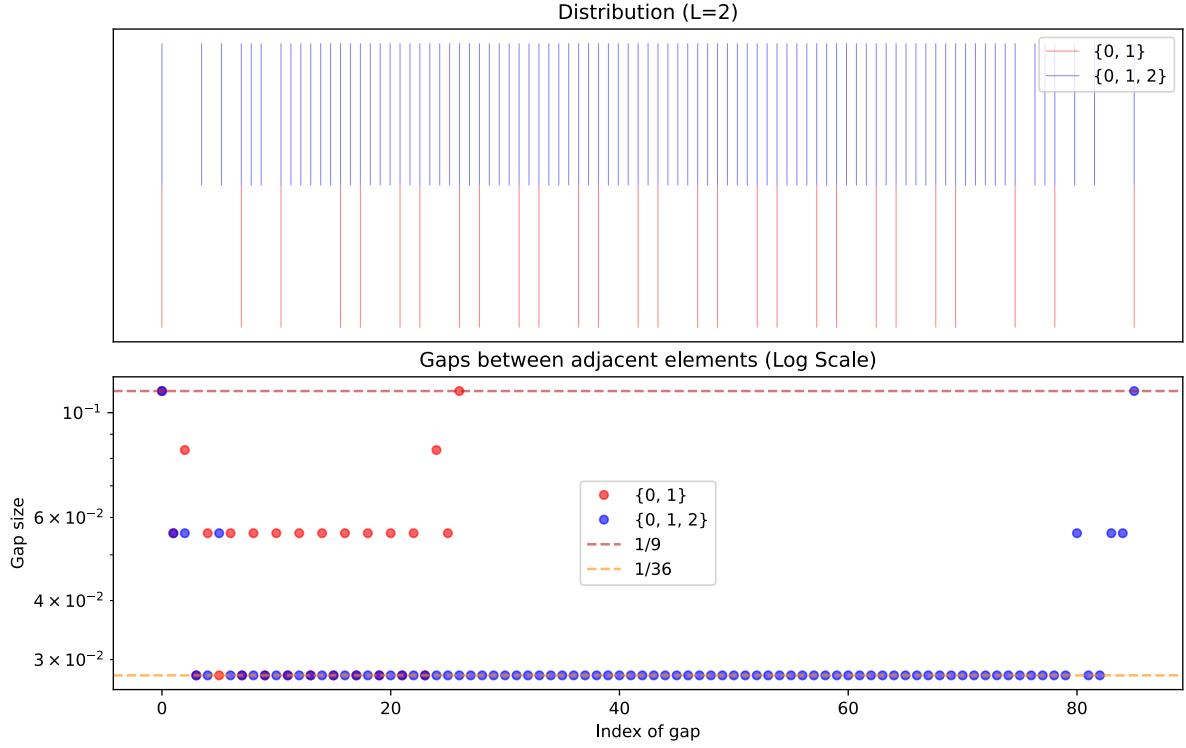
我们可以让  $c_{m,n} \in \{0, 1, 2\}$ 。后者其实并不麻烦，因为我们可以轻松地使用汇流器制造形如  $2/3$  的因子以得到系数 2。而前者的效率是不如后者的，因为只有  $\{0, 1\}$  的三进制是 Cantor 集的缩放，它是千疮百孔的。就算在我们的和式中有  $1/2$  因子帮忙填补空隙使得整个集合依然在一些区间上稠密，在有限个项的情况下，其能表示的有理点的个数只有后者的约  $2/3$  倍。因此，让  $c_{m,n} \in \{0, 1, 2\}$  是更好的选择。

接下来讨论我们需要多少项。每个基  $(m, n)$  需要消耗  $m + n$  个分流器（因为需要进行  $m$  次二分和  $n$  次三分）。在实际布线中，我们通常希望总复杂度可控，即所有分支的  $m_i + n_i$  之和不超过某个上限  $L$ ，即  $0 < m_i + n_i \leq L$ 。

我们可以看看  $L = 2$  的时候的分布情况，并对比前面提及的两种系数取值<sup>5</sup>：

---

<sup>5</sup>由于改变  $c_{m,n}$  的取值范围会改变和式的最大值，这张图中的 Distribution 仅作示意，不表示实际范围。



我们注意到，分布的两边值点非常稀疏，而中间的值点间距均匀且密集（始终保持着 $(1/6)^L$ 的间距）。直观上讲，这是因为在两边，我们需要更高的次数才能达到更小的值，然而这样的高次数是不被允许的；而在中间，可以用来线性组合生成新值的可选项非常多，自然更加细密。这也决定了我们接下来在设计上的原则：对于很小或者很大的目标比率，我们可以使用 $1 - f$ 或者 $2f, 3f$ 得到一个落在合理范围内的 $f'$ ，并设计这个 $f'$ 的分流电路，再通过一些简单的分流操作得到 $f$ 。

在实际上，我们希望我们运行的产线在 24 小时内浮动不超过 1 组，这意味着我们定制的速率的绝对误差需要小于

$$\frac{50}{60 \times 60 \times 24} = \frac{1}{1728} \approx 5.787 \times 10^{-4} \text{ ips}$$

经过简单的计算，不难得到这需要  $L \geq 5$ . 不论是搜索成本还是在游戏中的实现，这都有点过于昂贵了. 对于这个的解决方法，我们放在小节 3.3.2 讨论.

### 3.3. 基本部件与最优结构搜索

在这个章节，我们将把上文提到的和式中需要使用到的运算变成游戏内的传送带模块，并把整个任务变成一个树结构的搜索任务.

#### 3.3.1. 基本部件

**二分流器 (Splitter(2))** 二分流器接受一个输入，给出两个输出，且  $v_{\text{out}} = 0.5v_{\text{in}}$ . 它实现了  $\times 1/2$  的操作.

**三分流器 (Splitter(3))** 三分流器接受一个输入，给出两个输出，且  $v_{\text{out}} = v_{\text{in}}/3$ . 它实现了  $\times 1/3$  的操作.

除此之外，我们还希望从同一个输入引出只包含  $1/2^m$  和只包含  $1/3^n$  的分支. 因为我们不希望使用两个输入（这意味着对取料口的消耗），但我们又没办法在不引入  $1/2$  或  $1/3$  因子的前提下拆分输入，因此我们也需要能消耗  $1/2$  因子和  $1/3$  因子的工具. 以下两个结构正是这样的工具.

**2/3 因子 (Structure A)** 该结构如图所示，它有一个输入，两个输出，且  $v_{\text{out},1} = 2/3v_{\text{in}}$ ,  $v_{\text{out},2} = 1/3v_{\text{in}}$ . 它引入了 2 这个因子，可以实现  $c_{m,n} = 2$  的情况，也可以消去 1/2 因子. 下面的两张图分别给出了 Structure A 的结构和其消去 1/2 因子的用法.

**3/4 因子 (Structure B)** 该结构如图所示，它有一个输入，两个输出，且  $v_{\text{out},1} = 3/4v_{\text{in}}$ ,  $v_{\text{out},2} = 1/4v_{\text{in}}$ . 它引入了 3 这个因子，可以消去 1/3 因子. 下面的两张图分别给出了 Structure B 的结构和其消去 1/3 因子的用法.

### 3.3.2. 最优传送带拓扑结构

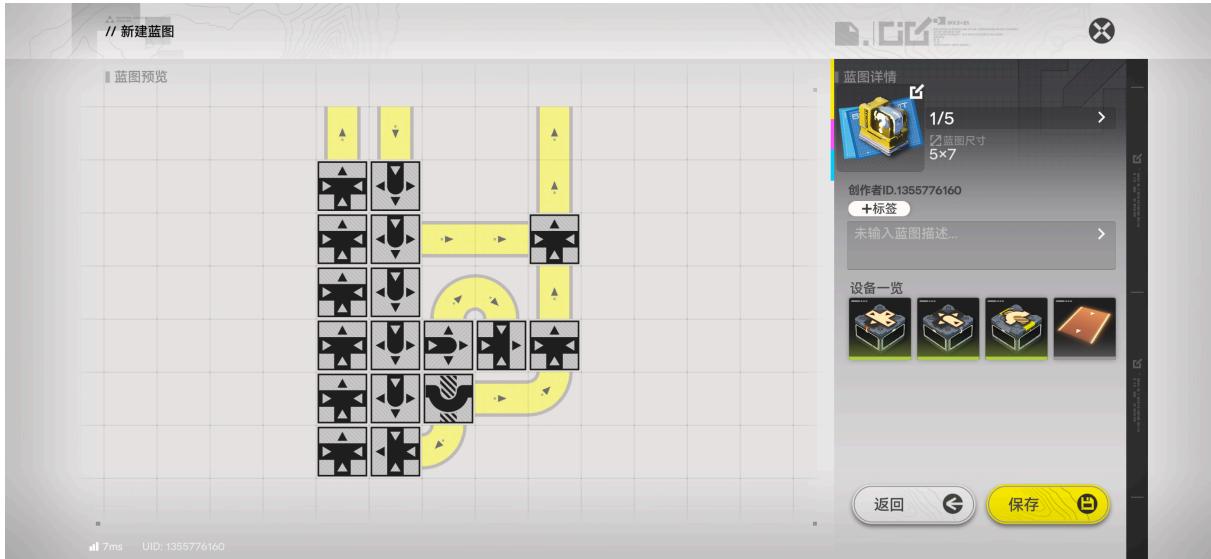
其实我们本可以先搜索最优和式的. 但是，在已知最优多项式的情况下搜索最优传送带拓扑结构的复杂度和直接搜索区别不大. 因此，我们干脆直接搜索传送带拓扑.

在我们最终的设计中，真正发挥作用的实际上只有在小节 3.3.1 中我们给出的几个部件以及汇流器，剩余的普通传送带等都可以简化为一种有向的传送关系. 因此，我们最终要搜索的其实是一个**有向无环图**，它从输入出发，获得两个输出，其中一个输出的速率是  $f \cdot v_{\text{in}}$ ，另一个是  $(1 - f) \cdot v_{\text{in}}$ .

然而，搜索图结构依然太复杂了. 实际上，我们可以完全不考虑汇流器，把这个图结构简化为一个树结构. 这个树结构以输入作为根节点，连接在小节 3.3.1 中提到的四种节点，并最终在叶子节点得到所有的项. 我们可以规定两种叶子节点：`Output` 和 `Discard`. 其中 `Output` 节点提供我们需要的输出，而 `Discard` 节点的则是“被分流”的不需要的部分. 举例而言，一个树结构可以长这样：

```
Tree:  (0, 0) Splitter(2) [v=1.000000]
      (1, 1) Discard
      (1, 1) Splitter(3) [v=0.500000]
      (2, 2)   Output [v=0.166667]
      (2, 2) Discard
      (2, 2) Splitter(2) [v=0.166667]
      (3, 3)   Splitter(3) [v=0.083333]
      (4, 4)   Structure B [v=0.027778]
      (5, 4)   Output [v=0.018519]
      (5, 5) Discard
      (4, 4)   Splitter(2) [v=0.027778]
      (5, 5)   Output [v=0.013889]
      (5, 5) Discard
      (4, 4) Discard
      (3, 3) Discard
```

在游戏中，它长这样：



接下来，我们讨论搜索的边界。我们注意到在这里，Structure A 拥有  $2^2$  的分母，它可以提供两个  $1/2$  的分母；同时，Structure A 和 Structure B 还可以抵消部分因子。因此如果我们要限定  $L = 5$  之类，我们不能只看树的深度，而是要看分配到的速率的分母包含了哪些因子。

这样一来，我们就可以写一个深度优先搜索对这个树结构进行搜索了。这个搜索是双目标的：我们要同时最小化误差和最小化结构复杂度。经过适当的剪枝，我们可以在十几秒的时间内对  $L = 3$  的树结构进行搜索——此时，状态空间的大小小于  $2^{(3+1)^2} = 65536$ ，依然可以接受的。

然而，当  $L = 4$  时，传统的搜索方法已经无法胜任了，这是因为此时，状态空间的大小来到了  $2^{(4+1)^2} = 33554432$  左右，不论在空间上还是时间上，这都是不可接受的。因此，我们要采用别的搜索方法，放弃最优解，转而使用启发式搜索。在这里，我们使用多目标的遗传算法 NSGA-II (Non-dominated Sorting Genetic Algorithm II, 非支配排序遗传算法 II)，在最小化误差的同时，也最小化使用的传送带部件的个数。实验表明，对于不太大的  $L$  (如  $L \leq 8$ ) 和不太小的目标值，在几百次以内的迭代下，我们总能找到一个足够好的解。

以下是几个解的例子。

**目标值  $f = 0.4$ ;  $L = 5$**

```
Error: 0.000823
Cost: 9.0
Output: 0.399177
Tree:
  (0, 0) Splitter(3) [v=1.000000]
    (1, 1) Discard
    (1, 1) Output [v=0.333333]
    (1, 1) Splitter(3) [v=0.333333]
    (2, 2) Discard
    (2, 2) Splitter(3) [v=0.111111]
    (3, 3) Structure B [v=0.037037]
    (4, 4) Output [v=0.024691]
    (4, 4) Structure B [v=0.012346]
    (5, 5) Discard
    (5, 5) Output [v=0.004115]
  (3, 3) Output [v=0.037037]
```

```

(3, 3)      Discard
(2, 2)      Discard
Error: 0.001235
Cost: 7.0
Output: 0.401235
Tree: (0, 0) Splitter(3) [v=1.000000]
(1, 1) Discard
(1, 1) Output [v=0.333333]
(1, 1) Splitter(3) [v=0.333333]
(2, 2) Discard
(2, 2) Splitter(3) [v=0.111111]
(3, 3) Structure B [v=0.037037]
(4, 4) Output [v=0.024691]
(4, 4) Splitter(2) [v=0.012346]
(5, 5) Output [v=0.006173]
(5, 5) Discard
(3, 3) Output [v=0.037037]
(3, 3) Discard
(2, 2) Discard

```

目标值  $f = 1/e \approx 0.367879$ ;  $L = 5$

```

Error: 0.000176
Cost: 8.0
Output: 0.368056
Tree: (0, 0) Splitter(2) [v=1.000000]
(1, 1) Discard
(1, 1) Splitter(2) [v=0.500000]
(2, 2) Output [v=0.250000]
(2, 2) Splitter(3) [v=0.250000]
(3, 3) Output [v=0.083333]
(3, 3) Discard
(3, 3) Splitter(2) [v=0.083333]
(4, 4) Splitter(2) [v=0.041667]
(5, 5) Discard
(5, 5) Output [v=0.020833]
(4, 4) Structure B [v=0.041667]
(5, 4) Discard
(5, 5) Output [v=0.013889]

Error: 0.002491
Cost: 7.0
Output: 0.370370
Tree: (0, 0) Splitter(2) [v=1.000000]
(1, 1) Discard
(1, 1) Splitter(2) [v=0.500000]
(2, 2) Splitter(3) [v=0.250000]
(3, 3) Output [v=0.083333]
(3, 3) Discard
(3, 3) Splitter(3) [v=0.083333]
(4, 4) Discard
(4, 4) Structure B [v=0.027778]
(5, 4) Discard
(5, 5) Output [v=0.009259]
(4, 4) Output [v=0.027778]
(2, 2) Output [v=0.250000]

```

目标值  $f = (\sqrt{5} - 1)/2 \approx 0.618034$ ;  $L = 8$

```

Error: 0.000022
Cost: 6.0
Output: 0.618056
Tree: (0, 0) Splitter(2) [v=1.000000]
      (1, 1)  Splitter(2) [v=0.500000]
      (2, 2)  Discard
      (2, 2)  Splitter(3) [v=0.250000]
      (3, 3)  Output [v=0.083333]
      (3, 3)  Discard
      (3, 3)  Splitter(3) [v=0.083333]
      (4, 4)  Discard
      (4, 4)  Output [v=0.027778]
      (4, 4)  Splitter(2) [v=0.027778]
      (5, 5)  Splitter(2) [v=0.013889]
      (6, 6)  Output [v=0.006944]
      (6, 6)  Discard
      (5, 5)  Discard
      (1, 1)  Output [v=0.500000]

```

综上，我们实现了一个通用且快速的结构搜索算法

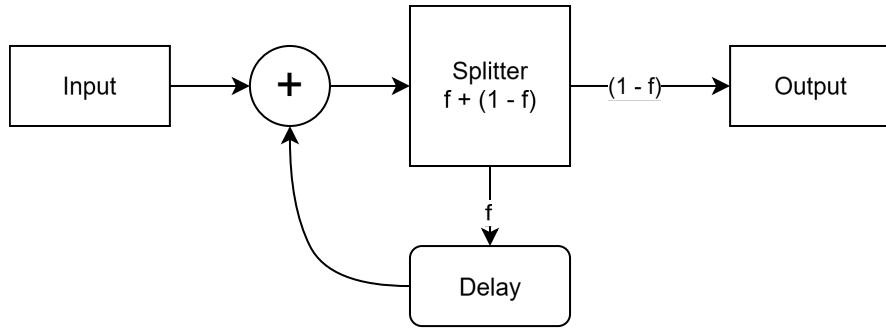
## 4. 环

以上的讨论都是基于无环结构的、线性的。如果我们引入环结构（即引入某种“反馈机制”），事情会发生意想不到的变化。

让我们抽象化地表示一个环结构。

### 4.1. 环的数学

环的结构如下图所示



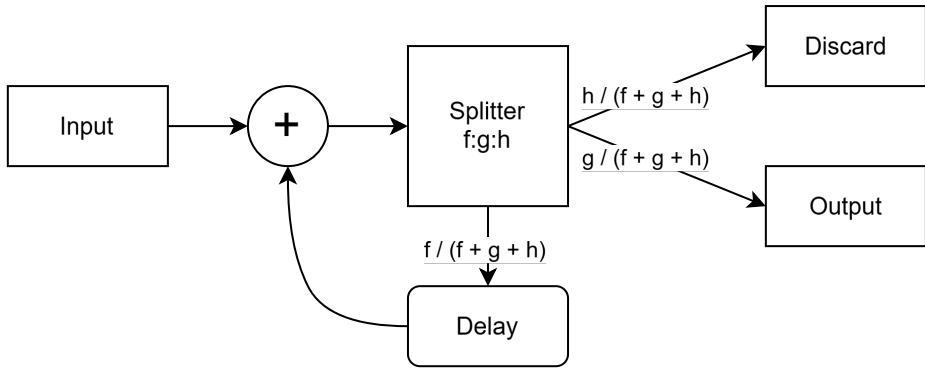
现在，我们从  $t = 0$  开始离散地计时。对于某个  $t \in \mathbb{N}$ ，我们记环刚刚与 Input 汇流时环内的速率为  $v_{\text{cyc}}[t]$ ，输入速率为  $v_{\text{in}}[t]$ ，输出速率为  $v_{\text{out}}[t]$ ，延迟块的延迟时间为  $T \in \mathbb{N}^+$ 。同时，我们假设除了 Delay 会造成延迟以外，其他的操作都是瞬时的。于是，我们可以写出如下的差分方程

$$\begin{cases} v_{\text{cyc}}[t] = v_{\text{in}}[t] + f \cdot v_{\text{cyc}}[t - T] \\ v_{\text{out}}[t] = (1 - f) \cdot v_{\text{cyc}}[t] \end{cases}$$

为了简化问题，我们认为  $v_{\text{in}}[t]$  是个常数。这种情况下，在稳态的时候， $v_{\text{out}} = v_{\text{in}}$ 。不过这之间可能存在  $T$  的相位差。

### 4.2. 更好的有理数

有了环，我们可以生成更多的分母，得到更好的有理数[1]。让我们稍稍扩展环结构。



根据刚才的分析，我们知道在稳态的时候

$$v_{\text{out}} + v_{\text{discard}} = v_{\text{in}}$$

而在这里，由于

$$\frac{v_{\text{out}}}{v_{\text{discard}}} = \frac{g}{h}$$

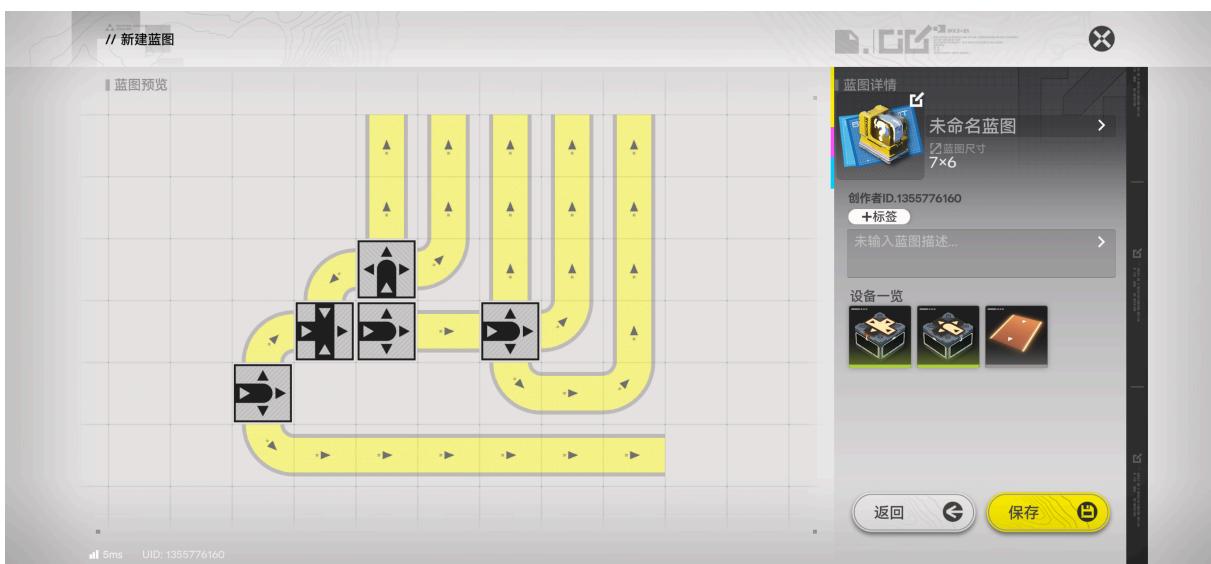
我们可以得到

$$v_{\text{out}} = \frac{g}{g+h} v_{\text{in}}$$

的输出.

现在根据小节 3.2 的讨论，我们知道任何分母是  $2^n 3^m$  的有理数是可达的. 既然如此，对于任何  $f + g + h = 2^n 3^m$ ，我们就可以构造出  $g/(g+h)$  这个值. 实际上，这意味着现在，任何有理数都是可达的.

但这里有一个值得注意的地方. 我们最初写的差分方程其实是不完整的，因为传送带有速率限制，即加法后的结果不能超过  $v_{\text{max}} = 0.5 \text{ ips}$ . 这样一来，这样的分流器还需要另外的设计才能正确运行，因为只要  $v_{\text{in}} = 0.5 \text{ ips}$ ，汇流就一定会导致阻塞. 于是，我们需要在进行这种分流前先进行一次二分，使得  $v_{\text{in}}$  降低到  $0.25 \text{ ips}$ . 以下是一个十分之一分流器.



### 4.3. 低通滤波

接下来我们来讨论汇流器求和后速度达到 0.5 ips 而造成阻塞的情况. 显然, 阻塞发生的条件是

$$v_{\text{in}[t]} + f \cdot v_{\text{cyc}[t-T]} \geq 0.5 \text{ ips}$$

同样, 我们认为  $v_{\text{in}[t]}$  是定值. 这样一来, 在稳态条件下, 只要

$$\frac{v_{\text{in}}}{1-f} \geq 0.5 \text{ ips}$$

阻塞就会发生. 此时

$$v_{\text{out}} = 0.5(1-f) \text{ ips}$$

这样一来, 这个环状结构就充当了一个缓冲区, 它可以吸收过大的输入速度, 并在输入速度减小的时候释放环中物品. 这构成了一个低通滤波器. 举例而言, 假如有一个非均匀输出的 1/5 分流器, 而我希望输出尽可能平滑, 我可以令  $f = 0.75$ , 这样我的输出的最大值会被限制在 0.125 ips, 任何瞬时的达于该速率的输入波动都会被环吸收, 并在平缓时释放.

## 参考文献

- [1] @早色芋 xyIvneu\_(uid: 277683791), 《原来在终末地里还有五等分分流器? 赤石科技 #1》. 2026 年.