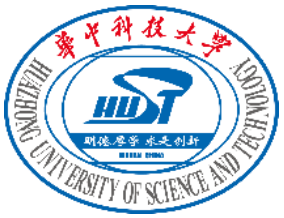

中国研究生创新实践系列大赛

中国光谷·“华为杯”第十九届中国研究生数学建模竞赛

学　　校	杭州电子科技大学
参赛队号	22103360092
队员姓名	1.李闻宇
	2.柯伟杰
	3.邱毅菲

中国研究生创新实践系列大赛

中国光谷·“华为杯”第十九届中国研究生数学建模竞赛

题 目 多约束条件下 PISA 架构芯片资源排布规划

摘 要：

现如今的国际局势之下，每一个国家都很重视高精尖的芯片制造产业，这关系着国家综合国力的发展。传统的交换芯片功能固定，不能满足日益多样化的用户应用需求，可编程的交换芯片弥补了这一不足之处，PISA 架构因此也成为了如今主流的可编程交换芯片架构。其中，PISA 架构芯片资源排布问题的解决受到了各界更大的关注。根据题目所述，本文将最小化基本块占用的流水线总级数作为最终优化目标，查阅文献和相关资料，建立线性规划模型，在 Python 的语言环境下采用数学规划优化器 Gurobi，研究了芯片资源限制等多重约束条件下的基本块排布问题即 PISA 架构芯片资源排布问题。

对于问题一，优化资源排布的目标是尽可能少的占用流水线级数，通过考虑流水线各级的资源约束、依赖约束两方面建立线性优化模型。构建依赖约束需要对数据进行预处理，根据所给的附件分析计算得出大小为 607 阶的邻接矩阵，在此基础上采用广度优先搜索的方法得到节点之间的通路矩阵，然后根据附录中数据依赖和控制依赖的定义在 Matlab 中求出**数据依赖矩阵**和**控制依赖矩阵**，最终求得共有 **11663** 个控制依赖、**607** 个读后写依赖、**3055** 个写后读和写后写依赖。在模型中，最小化基本块占用的流水线总级数作为目标函数，将基本块是否在流水级上选取为 0-1 决策变量，构建资源约束和依赖约束，考虑到所有基本块占用的总级数应该大于或等于每一个基本块所在的级数，从而不断优化流水线总级数，由此将得到的总级数最小化，最终建立 0-1 线性规划模型。建立模型后，利用 **Gurobi** 求解器进行求解，最终得到在多重资源约束条件下，所占用流水线级数为从第 0 级到第 50 级，共占用 **51** 级，求解器共计算了 **30958** 个变量和 **17383** 条约束，求解时间为 **3596.97** 秒。

对于问题二，在问题一的基础上，资源排布优化问题引入了执行流程关于共享资源的概念，同一流水线级的不同执行流程可以共享 HASH 资源和 ALU 资源，折叠的两个流水线级的执行路径可以共享 HASH 资源。在问题二中，我们引入了模拟退火算法，经过分析，第一问中得到的解为第二问的一个可行解，将此结果作为算法中的初始解，在新的资源约束、数据依赖和控制依赖的约束下进行迭代，最终得到了一个较优解，基本块所占用流水线的级数为从第 0 级到第 39 级，共占用 **40** 级，其中第 28-32 级流水线上未排布基本块。

关键词：PISA 架构芯片资源排布；0-1 线性规划模型；Gurobi；模拟退火算法

目录

1. 问题重述

1.1 问题背景

随着科学技术的不断进步和全球产业的数字化发展，无论是人们使用的手机、电脑等电子设备还是企业的数据信息存储等需求都离不开芯片的支撑，更重要的是，芯片的质量和产业化水平如今更代表着一个国家的经济实力和智能智造水平。芯片是电子产业的根基，在日益激烈的国际竞争局面下，芯片的制造技术是每一个国家都必争的高精尖技术。

传统的交换芯片功能固定，网络报文处理等功能均固定在芯片中完成，一定程度上网络的处理速度和性能得到了提升，但是随着互联网的进一步发展和普及，用户的需求更加多样化，上层设计针对底层网络也提出了更多要求，此时，传统固定功能的交换芯片不能很好地满足人们日益增长的应用需求，可编程的交换芯片保证转发性能保持高水平的同时，弥补了传统芯片无法实现的可编程功能，可以灵活地在芯片上添加需要的新功能。可编程交换芯片增强了网络的处理性能，提高了芯片的研发效率。

现有的软件定义网络 SDN（Software-Defined Networking）多使用 OpenFlow 技术，这种技术中使得网络编程受限于规定的网络协议，当版本更新时，OpenFlow 内部支持的协议版本也需要同步更新来保证后续的正常使用，这在一定程度上限制了芯片的可编程性[1]。因此，在可编程交换芯片的应用趋势逐渐扩大的同时，在数据平面完全可编程的 P4 语言弥补了 OpenFlow 的不足之处，实现了编程与协议无关，其中协议无关交换架构 PISA（Protocol Independent Switch Architecture）支持 P4 的可编程功能，也是现如今可编程交换芯片的主流架构之一。

PISA 架构包括了报文解析、报文处理和报文重组三个部分，如图 1-1 所示，关注报文处理过程中的 PISA 架构芯片资源排布问题。在多重的资源约束条件下，求解这个问题显得尤为困难。但是，处理好 PISA 架构芯片资源排布问题，提高芯片的资源利用率，让芯片能够支持运行更多的网络服务，对芯片产业的发展起着举足轻重的作用。

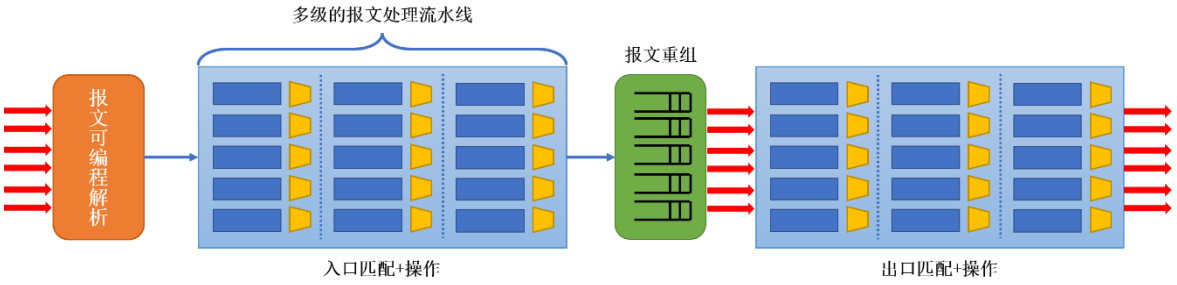

图 1-1 协议无关交换（PISA）架构图

1.2 问题提出

在计算机网络中，基本块是对源程序按照一定规则进行划分而成的程序片段，流水线是报文处理单元的集合，不同级的流水线就是指流水线中不同的处理单元。PISA 架构的三个部分中本题只考虑多级的报文处理流水线（Pipeline Pocket Process），该部分的主要作用是修改报文内容，对于不同的芯片来说，流水线的级别也会有所不同。

在 PISA 架构中，用户运用 P4 程序语言，经过编译器形成 P4 程序，在此过程中，编译器按照规则将源程序划分成一系列片段，即程序基本块，这些基本块会占用一定

的芯片资源，然后基本块进入流水线各级作业中进行相应的处理过程，这个过程就是将芯片资源分布到流水线各级上，即确定芯片资源在流水线上所处级别。上述基本块的排布问题就是 PISA 架构芯片资源排布问题，在多重资源条件的限制下，求解资源排布问题，提高芯片资源利用率对芯片底层执行和编译器设计至关重要。

PISA 架构芯片资源排布问题就是将程序划分而成的基本块，在多重资源限制的条件下，合理地分布到流水线各级当中，最终使得基本块排布满足资源约束且占用的流水线级数少。而基本块的资源约束主要来源于两个方面，一方面是来自于基本块的读写操作，使得基本块之间存在数据和控制依赖影响着流水线的级数长短；另一方面，芯片本身的资源限制，流水线各级对于芯片中 TCAM、HASH、ALU、QUALIFY 四类资源空间都有着严格的限制，基本块的排布不能违背这些规则。

综上所述，PISA 架构芯片资源排布需要综合考虑数据依赖、控制依赖和流水线资源限制的因素，结合相关的文献资料，通过合理的假设，建立适当的数学模型，解决以下问题：

（1）流水线每级对芯片的 TCAM、HASH、ALU、QUALIFY 四类资源都有不同的约束，其中 TCAM 资源偶数级数不能超过 5，同时流水线 32 级内有折叠级数的限制要求，第 0 级与第 16 级，第 1 级与第 17 级……第 15 级与第 31 级为折叠级数，折叠的两级 TCAM 和 HASH 资源总和有约束，一个基本块也只能占用一个流水线级数。在考虑基本块数据依赖和控制依赖对流水线级数影响的情况下，综合考虑流水线中芯片的资源约束，给出资源排布算法，以流水线被占用的级数尽可能短为最终优化目标，输出基本块排布结果。

（2）在问题一的资源约束条件下，引入了基本块之间执行流程的新概念，不同执行流程上基本块可以共享 HASH 和 ALU 资源，修改问题一中的与 HASH、ALU 资源相关的约束条件，同时考虑折叠级数，给出资源排布算法，以流水线被占用的级数尽可能短为最终优化目标，输出基本块排布结果。

2. 问题分析

2.1 问题一的分析

本题所给数据数量庞大，两个问题均需考虑到基本块之间的数据依赖、控制依赖以及芯片的资源约束条件限制，综合考虑的情况下，最终目的是在多重限制下解决 PISA 架构芯片资源排布问题，输出基本块的排布结果，以基本块占用的流水线级数尽可能短为优化目标。

问题一中需要考虑数据依赖、控制依赖和资源约束条件，不需要考虑执行流程的问题，也就不存在共享资源的问题。根据题目阐述数据依赖的定义可知，如果两个基本块之间不存在连接通路，则两者之间必然不存在数据依赖；如果两个基本块之间存在通路，且针对同一个变量进行了读写操作，那么两者之间存在数据依赖。同时根据控制依赖的定义描述，若基本块 a 不存在连接基本块 b 的通路或者 a 的所有邻接基本块均连接至基本块 b，那么两个基本块之间不存在控制依赖；若某一基本块存在多条出路，只有部分出路经过下游一基本块，则两个基本块之间存在控制依赖。

本题建立线性规划模型，将基本块是否在流水级上选取为 0-1 决策变量，用流水线每一级的各类资源限制、数据依赖和控制依赖等进行约束，而后将流水线总级数最小值作为优化的目标函数。

同时该题数据量巨大且需要综合考虑数据和控制两种依赖，所以首先要对题目所给的多条数据进行预处理操作，根据题目所给的 attachment2 和 attachment3 两个表格建立两种依赖的有向关系网络，确定所有基本块之间的数据、控制依赖关系，得出数据依赖和控制依赖的两个初始矩阵，而后依据该矩阵分析后续模型中两种依赖的约束条件，最后通过模型，在 Python 的语言环境下利用数学规划优化器 Gurobi 进行问题优化求解。

问题一思路分析如图 2-1 所示：

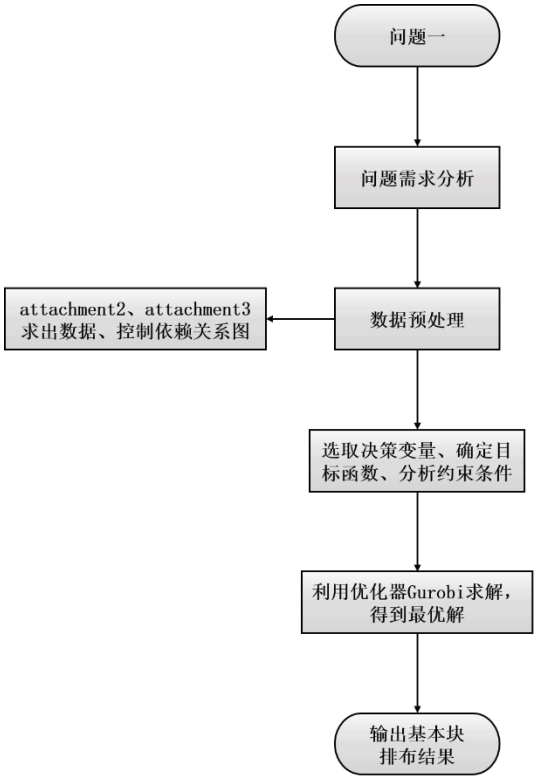

图 2-1 问题一思路分析

2.2 问题二的分析

经过数据预处理，得到的数据依赖和控制依赖矩阵是解决问题的前提，然后在问题一的基础上，问题二引入了执行流程的概念，由一个基本块出发可以到达另一个基本块，则这两个基本块处于同一条执行流程，反之两者不在同一条执行流程中。由于执行流程条件的介入，基本块的资源约束发生变动，不同的执行流程上基本块可以共享 HASH 和 ALU 资源，反之则不能共享资源。

本题依然使用 0-1 线性规划模型，将基本块是否在流水级上、基本块是否在路径上两者分别选取为 0-1 决策变量，流水线每级的 TCAM、QUALIFY 资源约束条件和问题一保持一致，由于引入执行流程关于共享资源的概念，更改 HASH 和 ALU 资源限制条件，同时将数据依赖和控制依赖等加入模型的约束条件，最小化流水线总级数作为优化目标函数。

综合考虑以上条件和数学模型，选取模拟退火算法，将问题一 Gurobi 求解得到的解作为本题的一个可行解，将其当作算法中的初始解进行迭代计算，最终可得问题二的较优解集，输出基本块的最终排布结果。

问题二思路分析如图 2-2 所示：

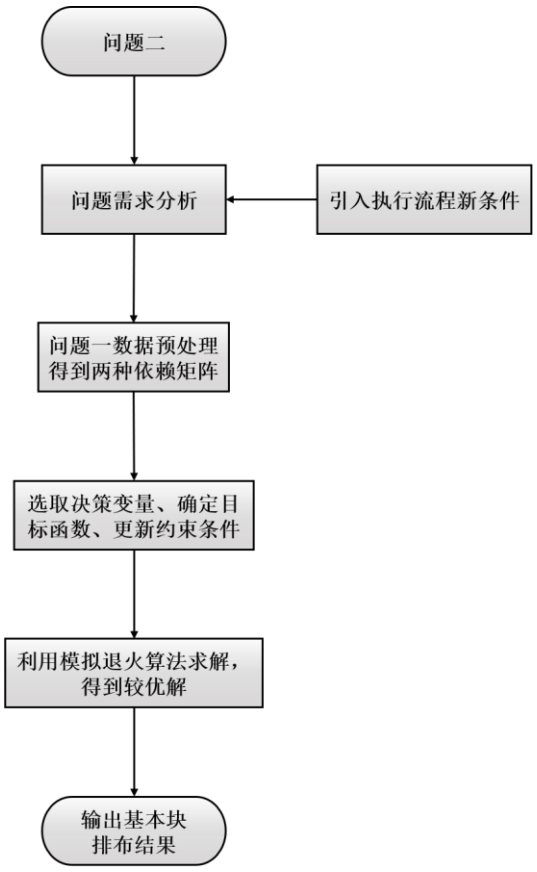

图 2-2 问题二思路分析

3. 模型假设

1. 假设基本块被抽象为节点，抽象后基本块中具体的执行指令被屏蔽，只保留读写信息；
2. 假设流水线级数 32 级之后的级数，不考虑折叠级数的资源限制。

4. 符号说明

符号	说明
I	基本块数目
J	流水线级数
u	初始设定流水线级数
x_{ij}	基本块 i 是否置于流水线第 j 级中，如果是，则 x_{ij} 为 1，否则为 0
Z_i	第 i 个基本块所在的级数
T_i	第 i 个基本块对 TCAM 资源的占用
H_i	第 i 个基本块对 HASH 资源的占用
A_i	第 i 个基本块对 ALU 资源的占用
Q_i	第 i 个基本块对 QUALIFY 资源的占用
S	数据依赖矩阵
s_{mn}	第 m 个基本块对第 n 个基本块是否有数据依赖
C	控制依赖矩阵
c_{mn}	第 m 个基本块对第 n 个基本块是否存在控制依赖
P	路径矩阵
P_{in}	第 i 个基本块是否在第 n 条路径上，如果在则为 1，否则为 0

注:其它符号将在文中具体说明

5. 问题一模型的建立与求解

5.1 数据预处理

本题首先需要对题目所提供的冗杂数据进行预处理，理清各个基本块之间的对应关系，绘制邻接矩阵，然后明确每个基本块的数据依赖和控制依赖的实际情况，经过预处理操作后的数据情况更利于后续建立模型，进行求解运算，输出最终基本块排布情况。

5.1.1 邻接矩阵

根据 attachment3 中各个基本块在流程图中的邻接基本块信息，可以求出该有向图的邻接矩阵，该矩阵为一个二维数组，用来存放各个顶点间的关系，然后运用 Matlab 对该有向图进行绘制，得到的有向图如图 5-1 所示：

图 5-1 有向图

统计得出基本块之间连通情况如图 5-2 所示，共有 122847 对节点存在通路：

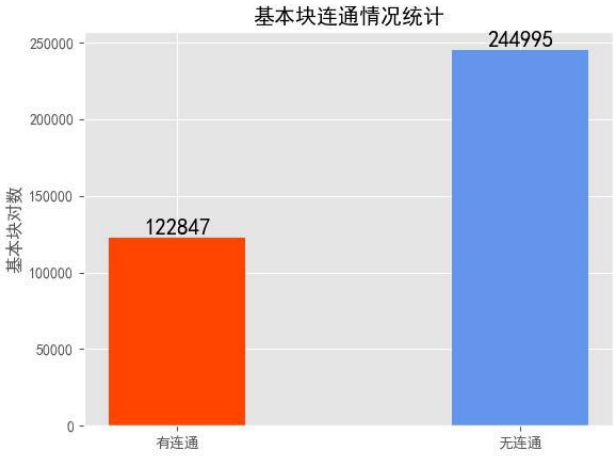

图 5-2 基本块连通情况统计

5.1.2 数据依赖矩阵

数据依赖定义为：语句或代码块间数据流造成的一种约束，具体包括三种形式：写后读、读后写、写后写。

两个基本块之间存在数据依赖的前提是两个基本块之间有通路，故先采用广度优先搜索对两两基本块之间是否存在通路进行计算[2]，将基本块抽象为节点，如果两个节点之间不存在通路，则这两个节点之间不存在数据依赖，如果两个节点之间存在通路并且都对同一个变量进行了读或者写的操作，那么可以称这两个节点之间存在数据依赖。

求解数据依赖矩阵的算法流程图如图 5-3 所示：

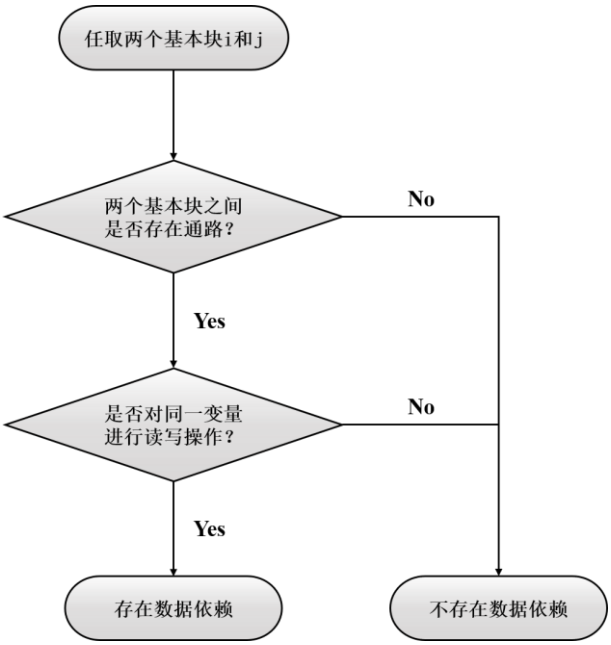

图 5-3 数据依赖计算流程图

数据预处理将数据依赖情况处理成矩阵展示，其中基本块有写后读依赖或写后写依赖矩阵元素数值显示为 1，有读后写依赖数值为-1，没有数据依赖数值为 0。

根据数据依赖矩阵，可得所有基本块数据依赖情况如图 5-4 所示，其中有 3055 对基本

块存在写后读或者写后写依赖，有 607 对基本块存在读后写依赖。

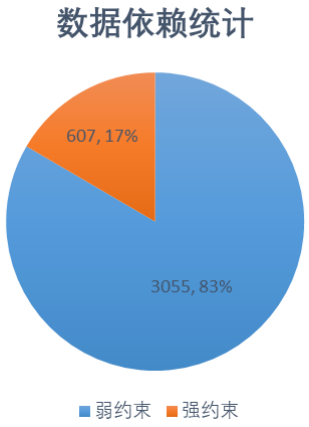

图 5-4 基本块数据依赖情况

基本块数据依赖强约束（写后读、写后写依赖）部分情况如表 5-1 所示，弱约束（读后写数据依赖）部分情况如表 5-2 所示。所有基本块完整数据依赖参见附件。

基本块编号	产生写后读、写后写依赖的基本块编号														
0															
1															
2															
3															
4	586	588													
5	57	69	76	77	90										
6															
7	10	89													
8	10	23	98												
9															
10	23	57	89	98											
11															
12	31	175	178	181	184	187	201	211	304	307	309	341	414	440	……
13															
14	31	140	158	160	162	164	187	214	304	307	309	531	538	554	563
15	10	28	80	83	88	128	135	143	146	149	152	155	172	175	……
16	35	333	366	369											
17															
18															
19	31	39	41	43	45	54	56	66	69	77	175	178	181	184	……
20															

表 5-1 强约束（写后读、写后写数据依赖）部分情况

基本块编号	产生读后写依赖的基本块编号												
⋮													
12	5	7	10	89	99	443	458	463					
13													
14	7	8	10	89	98	443							
⋮													
19	103	107	111	115	119	175	178	181	184	250	253	256	……
20													
21	403	438	449	453	604								
⋮													
54	5	56	175	178	181	184	604	606					
55													
56	5	175	178	181	184	606							

表 5-2 弱约束（读后写数据依赖）部分情况

5.1.3 控制依赖矩阵

控制依赖定义为：当从某个基本块出发的路径，只有部分路径通过下游某个基本块时，这两个基本块构成控制依赖。将基本块抽象为节点，综上所述只有一条出向边的节点不会和其它任何节点构成控制依赖。

判断节点 a 和节点 b 之间是否存在控制依赖首先判断该节点的出度是否大于 1。若大于 1，则找出节点 a 的所有邻接节点。如果节点 a 的所有邻接节点均和节点 b 之间存在通路或均无法到达节点 b，那么则称节点 a 和节点 b 之间不存在控制依赖；若只有部分邻接节点和目标节点 b 之间存在通路，那么称节点 a 和节点 b 之间存在控制依赖。

算法流程图如图 5-5 所示：

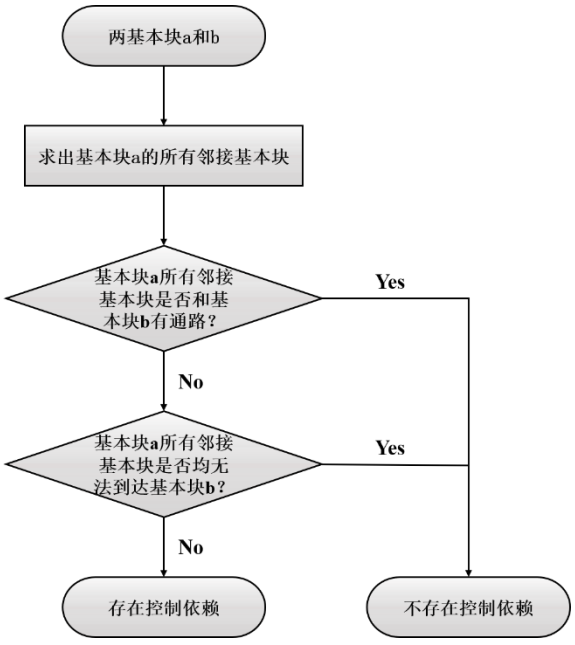

图 5-5 控制依赖计算流程图

根据控制依赖矩阵，可得所有基本块控制依赖情况如图 5-6 所示，一共有 11663 对基本块存在控制依赖。

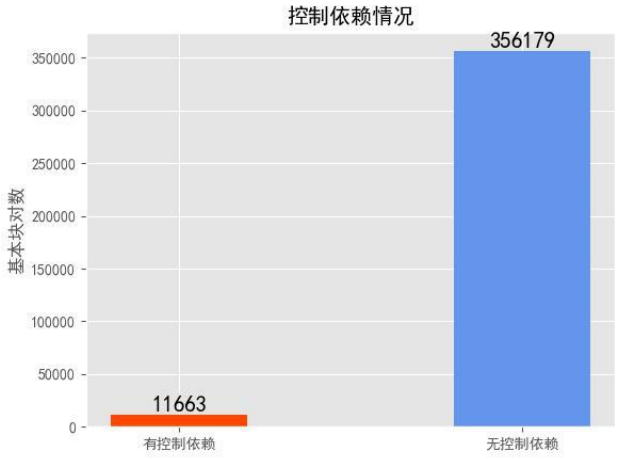

图 5-6 基本块控制依赖情况

基本块控制依赖部分情况如表 5-3 所示。所有基本块完整数据依赖参见附件。

基本块编号	产生控制依赖的基本块编号												
⋮													
⋮													
74	102	103	104	105	106	107	108	109	110	111	112	113	……
75	344	345	346	347	348	349	350	351	352	353	354	355	
76	77												
77													
78	23	24	25	26	57	79	80	81	82	83	84	85	……
79	80												
80													
81													
82	83	84											
83	84												
84													
85													
86	91	92	93	94	95								
101	23	24	25	26	57	82	83	84	85	86	87	88	……
102	103	104	105										
103	104	105											

表 5-3 基本块控制依赖部分情况

5.2 模型建立

由分析可得，本题选择建立 0-1 线性规划模型来求解最终结果，明确基本块所处流水线级数，从而根据流水线总级数最小化来建立目标函数，进一步通过资源限制、数据依赖和控制依赖的约束条件对已建立的目标函数进行规划求解。

设基本块的数目为 I；流水线级数为 J；设 u 为初始设定流水线级数，其值足够大；x_{ij} 表示是否将基本块 i 放置于流水线第 j 级中，如果放置，则 x_{ij} 为 1，否则为 0；Z_i 表示第 i

个基本块所在的级数 $Z_i=\sum_{j=0}^{u} j\times x_{ij}$；第 i 个基本块对资源 TCAM、HASH、ALU、QUALIFY 的占用分别为 T_i、H_i、A_i、Q_i。

由数据预处理求得的数据依赖矩阵为 S，s_{mn} 表示第 m 个基本块对第 n 个基本块是否有数据依赖，有写后读依赖或写后写依赖其值为 1，有读后写依赖其值为-1，没有数据依赖其值为 0；控制依赖矩阵为 C，c_{mn} 表示第 m 个基本块对第 n 个基本块是否存在控制依赖，存在控制依赖其值为 1，不存在控制依赖其值为 0。

1. 决策变量

$$\begin{cases} x_{ij}=1, \text{ 基本块 } i \text{ 在流水线 } j \text{ 上} \\ x_{ij}=0, \text{ 基本块 } i \text{ 不在流水线 } j \text{ 上} \end{cases} \tag{5-1}$$

2. 目标函数

考虑到规划目标为尽可能少的占用流水线级数 J，设定目标函数为：

$$\min J \tag{5-2}$$

3. 约束条件为

(1) Z_i 为基本块 i 所在流水线的级数，流水线的总级数应大于或等于每个基本块所在级数，因此有：

$$Z_i=\sum_{j=0}^{u} j\times x_{ij}\ ,\ i=0,1,...,I-1 \tag{5-3}$$

$$J\geq Z_i,\ \ i=0,1,...,I-1 \tag{5-4}$$

(2) 所有基本块均有对应的使用 TCAM 资源（T_i）、HASH 资源（H_i）、ALU 资源（A_i）、QUALIFY 资源（Q_i）的数目，在每级流水线中占用这四种资源的数目均不得超过每级流水线的资源上限，则有：

$$\begin{cases} \sum_{i=0}^{I-1} x_{ij}\times T_i\leq 1\ ,\ j=0,1,...,u \\ \sum_{i=0}^{I-1} x_{ij}\times H_i\leq 2\ ,\ j=0,1,...,u \\ \sum_{i=0}^{I-1} x_{ij}\times A_i\leq 56\ ,\ j=0,1,...,u \\ \sum_{i=0}^{I-1} x_{ij}\times Q_i\leq 64\ ,\ j=0,1,...,u \end{cases} \tag{5-5}$$

(3) 在 32 级流水线以下级数相差 16 的流水线的两级记为折叠的两级，折叠的两级流水线 TCAM 资源加起来最大为 1，HASH 资源加起来最大为 3，排布过程中基本块占用资源数目不得超过该要求的资源上限，则有：

$$\begin{cases} \sum_{i=0}^{I-1} x_{ij}\times T_i+x_{i,j+16}\times T_i\leq 1\ ,\ j=0,1,...,15 \\ \sum_{i=0}^{I-1} x_{ij}\times H_i+x_{i,j+16}\times H_i\leq 3\ ,\ j=0,1,...,15 \end{cases} \tag{5-6}$$

(4) 每个基本块只能排布到一级流水线，可得：

$$\sum_{j=0}^{u} x_{ij} = 1 \ , i = 0,1,...,I-1 \tag{5-7}$$

(5) 基本块占有 TCAM（T_i）资源的偶数级数量不超过 5，由此得：

$$\frac{1}{2} \times \sum_{j=0}^{u} (1+(-1)^j) \sum_{i=0}^{I-1} x_{ij} \times T_i \le 5 \tag{5-8}$$

(6) 数据依赖约束，若基本块 m 和基本块 n 之间具有写后读或写后写依赖，则基本块 m 的流水线级数 Z_m 应小于基本块 n 的流水线级数 Z_n；若具有读后写依赖，则基本块 m 的流水线级数 Z_m 应不大于基本块 n 的流水线级数 Z_n，由此可得：

$$\begin{cases} Z_m - Z_n < 0 \ , \ s_{mn} = 1 \ , m = 0,1,...,I-1 \ , n = 0,1,...,I-1 \\ Z_m - Z_n \le 0 \ , \ s_{mn} = -1 \ , m = 0,1,...,I-1 \ , \ n = 0,1,...,I-1 \end{cases} \tag{5-9}$$

(7) 控制依赖约束，若基本块 m 和基本块 n 之间具有控制依赖，则基本块 m 的流水线级数 Z_m 应不大于基本块 n 的流水线级数 Z_n；由此可得：

$$Z_m - Z_n \le 0 \ , \ \mathrm{c}_{mn} = 1 \ , \ m = 0,1,...,I-1 \ , \ n = 0,1,...,I-1 \tag{5-10}$$

4. 综上所述，得到最终的线性规划模型为：

$$\min J$$

$$s.t \begin{cases} Z_i = \sum_{j=0}^{u} j \times x_{ij} \ , i = 0,1,...,I-1 \\ J \ge Z_i \ , \ i = 0,1,...,I-1 \\ \begin{cases} x_{ij} = 1, \text{ 基本块 } i \text{ 在流水线 } j \text{ 上} \\ x_{ij} = 0, \text{ 基本块 } i \text{ 不在流水线 } j \text{ 上} \end{cases} \\ \begin{cases} \sum_{i=0}^{I-1} x_{ij} \times T_i \le 1 \ , \ j = 0,1,...,u \\ \sum_{i=0}^{I-1} x_{ij} \times H_i \le 2 \ , \ j = 0,1,...,u \\ \sum_{i=0}^{I-1} x_{ij} \times A_i \le 56 \ , \ j = 0,1,...,u \\ \sum_{i=0}^{I-1} x_{ij} \times Q_i \le 64 \ , \ j = 0,1,...,u \end{cases} \\ \begin{cases} \sum_{i=0}^{I-1} x_{ij} \times T_i + x_{i,j+16} \times T_i \le 1 \ , \ j = 0,1,...15 \\ \sum_{i=0}^{I-1} x_{ij} \times H_i + x_{i,j+16} \times H_i \le 3 \ , \ j = 0,1,...15 \end{cases} \\ \frac{1}{2} \times \sum_{j=0}^{u} (1+(-1)^j) \sum_{i=0}^{I-1} x_{ij} \times T_i \le 5 \end{cases}$$

$$s.t\begin{cases}\sum_{j=0}^{u} x_{ij}=1\ ,\ i=0,1,...,I-1\\ \begin{cases}Z_m-Z_n<0\ ,\ s_{mn}=1\ ,m=0,1,...,I-1\ ,n=0,1,...,I-1\\ Z_m-Z_n\le 0\ ,\ s_{mn}=-1\ ,m=0,1,...,I-1\ ,n=0,1,...,I-1\end{cases}\\ Z_m-Z_n\le 0\ ,\ c_{mn}=1\ ,m=0,1,...,I-1\ ,n=0,1,...,I-1\end{cases}$$

5.3 模型求解

Gurobi 采用最新的优化技术，充分利用了多核处理器的优势，经过 Gurobi 的处理，得到的结果确定而非随机，对于分析求解数学规划优化问题有着巨大优势，可以求解线性规划、整数规划等数学问题。本文建立了线性规划模型对芯片资源排布问题进行分析，结合数学模型，在 Python 的语言环境下利用大规模数学规划优化器 Gurobi 对该问进行求解[3]。

本题利用 Gurobi 优化器进行求解，展示其界面展示的求解结果如图 5-7 所示，得到在多重资源约束条件下，求得所有基本块占用流水线总级数为 51 级，求解器的求解时间为 3596.97 秒。

```
Explored 198755 nodes (7785362 simplex iterations) in 3596.97 seconds
Thread count was 16 (of 16 available processors)

Solution count 10: 51 52 54 ... 74
```

图 5-7 Gurobi 求解结果界面

得到最小化所有基本块占用的流水线级数的结果，同时输出基本块的部分排布结果如表 5-4 所示，所有基本块的排布情况参见附件。

流水线级数 \ 基本块编号	问题一输出基本块的部分排布结果													
第 0 级	13	14	15	17	18	20	21	22	37	38	58	59	132	……
第 1 级	12	19	135	136	137	145	148	151	152	154	157	159	161	……
第 2 级	139	144	162	374	382	383	384	385	386	387	388			
第 3 级	146	158	160	375										
第 4 级	27	147	149	155	156	361	380	505	509	512	515	518	525	……
第 5 级	150	391	508	513	516	521	529	534	542	554	557			
第 6 级	16	153	370	510	519	522	538	555	565					
第 7 级	362	526	544	546	559	560	581							
第 8 级	506	507	523	537	549	562	563	583						
第 9 级	211	212	392	535	543	553	564	585	595	596				
第 10 级	28	141	213	214	511	514	517	520	524	527	541	570	571	……
第 11 级	3	32	140	216	218	219	221	226	227	597	598	599	600	
第 12 级	54	142	143	215	217	222	223	224	225	313	573	578	579	……
第 13 级	56	167	230	302	311	314	324	330						
第 14 级	33	51	187	201	220	303	306	326	331	397				
第 15 级	36	40	42	168	189	193	196	200	305	308	309	310	315	……

表 5-4 问题一输出基本块的部分排布结果

5.4 灵敏性分析

在实际芯片资源排布问题中，各级资源的约束值可能会发生变化，包括每级流水线的 ALU 资源上限、QUALIFY 资源上限、偶数级流水线的 TCAM 资源上限等。将这些值上下波动一到两个单位，利用 Matlab 软件绘制这些参数分别与最小流水线级数之间的示意图，如图 5-8 所示：

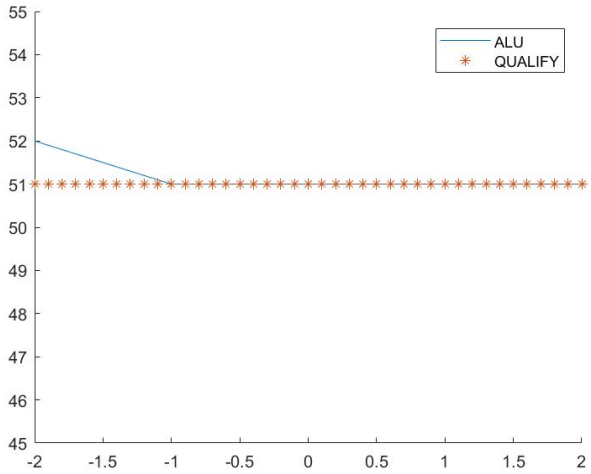

图 5-8 灵敏性分析

图 5-8 中，问题一的资源排布模型对每级流水线中 ALU 资源上限、QUALIFY 资源上限敏感度较低，其值的变化对结果影响不大，A 的值改变了 3.6%，J 值仅改变了 1.9%，从而可以说明模型的稳定性较好，在实际进行芯片资源排布时不用过多注意 ALU 资源和 QUALIFY 资源的上限值。

6. 问题二模型的建立与求解

6.1 模型建立

由分析可得，本题仍然利用线性规划模型来求解最终结果，明确基本块所处流水线级数，从而根据流水线总级数最小化来建立目标函数，引入执行流程这一新约束条件，进一步通过资源限制、数据依赖和控制依赖的约束条件对已建立的目标函数进行规划求解。

设基本块的数目为I；流水线级数为J；设u为初始设定流水线级数，其值足够大；x_{ij}表示是否将基本块i放置于流水线第j级中，如果放置，则x_{ij}为1，否则为0；Z_i表示第i个基本块所在的级数$Z_i=\sum_{j=0}^{u} j\times x_{ij}$；第$i$个基本块对资源TCAM、HASH、ALU、QUALIFY的占用分别为T_i、H_i、A_i、Q_i。设定E为有向图中所有路径的集合，集合中共有N条路径，第n条路径为E_n。求得的路径矩阵为P，其中p_{in}表示第i个基本块是否在第n条路径上，如果在此路径上，则p_{in}数值为1，否则为0。

由数据预处理求得的数据依赖矩阵为S，s_{mn}表示第m个基本块对第n个基本块是否有数据依赖，有写后读依赖或写后写依赖其值为1，有读后写依赖其值为-1，没有数据依赖其值为0；控制依赖矩阵为C，c_{mn}表示第m个基本块对第n个基本块是否存在控制依赖，存在控制依赖其值为1，不存在控制依赖其值为0。

1. 决策变量

$$x_{ij}=\begin{cases}1\text{，基本块 }i\text{ 在流水线 }j\text{ 上}\\0\text{，基本块 }i\text{ 不在流水线 }j\text{ 上}\end{cases} \tag{6-1}$$

2. 目标函数

考虑到规划目标为尽可能少的占用流水线级数J，设定目标函数为：

$$\min J \tag{6-2}$$

3. 约束条件为

(1) 规定Z_i为基本块i所在流水线的级数，流水线的总级数J应大于或等于每个基本块所在级数，因此有：

$$Z_i=\sum_{j=0}^{u} j\times x_{ij}\ ,\ i=0,1,...,I-1 \tag{6-3}$$

$$J\geq Z_i,\ \ i=0,1,...,I-1 \tag{6-4}$$

(2) 所有基本块均有对应的使用TCAM资源（T_i）、HASH资源（H_i）、ALU资源（A_i）、QUALIFY资源的数目（Q_i），在每级流水线中占用这四种资源的数目均不得超过每级流水线的资源上限，则有：

$$\begin{cases}\sum_{i=0}^{I-1} x_{ij}\times T_i\leq 1\text{，}\ j=0,1,...,u\\ \sum_{i=0}^{I-1} x_{ij}\times Q_i\leq 64\text{，}\ j=0,1,...,u\end{cases} \tag{6-5}$$

(3) 在每级流水线中一条执行流程上占用的HASH资源（H_i）和ALU资源（A_i）数目不得超过规定上限值，用P_{in}来表示基本块i是否在路径n上，故有：

$$\begin{cases}\sum_{i=0}^{I-1} x_{ij} \times p_{in} \times H_i \le 2,\ j=0,1,...,u,\ \ n=1,2,...,N \\ \sum_{i=0}^{I-1} x_{ij} \times p_{in} \times A_i \le 56,\ j=0,1,...,u,\ \ n=1,2,...,N \end{cases} \tag{6-6}$$

(4) 在 32 级流水线以下级数相差 16 的流水线的两级记为折叠的两级，折叠的两级流水线所占的 TCAM 资源总和不得超过资源上限，则有：

$$\sum_{i=0}^{I-1} x_{ij} \times T_i + x_{i,j+16} \times T_i \le 1\ ,\ j=0,1,...15 \tag{6-7}$$

(5) 折叠级数前一级流水线一条执行流程上的基本块所占用的 HASH 资源与后一级流水线一条执行流程上的基本块所占用的 HASH 资源之和不得超过规定上限，故有：

$$\sum_{i=0}^{I-1} x_{ij} \times p_{in} \times H_i + \sum_{i=0}^{I-1} x_{i,j+16} \times p_{in} \times H_i \le 3,\ j=0,1,...,15,\ \ n=1,2,...,N \tag{6-8}$$

(6) 每个基本块只能排布到一级流水线，可得：

$$\sum_{j=0}^{u} x_{ij} = 1\ \ ,i=0,1,...,I-1 \tag{6-9}$$

(7) 基本块占有 TCAM（T_i）资源的偶数级数量不超过 5，由此得：

$$\frac{1}{2} \times \sum_{j=0}^{u} (1+(-1)^j) \sum_{i=0}^{I-1} x_{ij} \times T_i \le 5 \tag{6-10}$$

(8) 数据依赖约束，若基本块 m 和基本块 n 之间具有写后读或写后写依赖，则基本块 m 的流水线级数 Z_m 应小于基本块 n 的流水线级数 Z_n；若具有读后写依赖，则基本块 m 的流水线级数 Z_m 应不大于基本块 n 的流水线级数 Z_n，由此可得：

$$\begin{cases} Z_m - Z_n < 0\ ,\ \ s_{mn}=1\ ,m=0,1,...,I-1\ ,n=0,1,...,I-1 \\ Z_m - Z_n \le 0\ ,\ \ s_{mn}=-1\ ,m=0,1,...,I-1\ ,\ n=0,1,...,I-1 \end{cases} \tag{6-11}$$

(9) 控制依赖约束，若基本块 m 和基本块 n 之间具有控制依赖，则基本块 m 的流水线级数 Z_m 应不大于基本块 n 的流水线级数 Z_n；由此可得：

$$Z_m - Z_n \le 0\ ,\ \ \mathrm{c}_{mn}=1\ ,\ m=0,1,...,I-1\ ,\ n=0,1,...,I-1 \tag{6-12}$$

4. 综上所述，得到最终的线性规划模型为：

$$\min J$$

$$s.t \begin{cases} Z_i = \sum_{j=0}^{u} j \times x_{ij}\ \ ,i=0,1,...,I-1 \\ J \ge Z_i,\ \ i=0,1,...,I-1 \\ \begin{cases} \sum_{i=0}^{I-1} x_{ij} \times T_i \le 1\ ,\ j=0,1,...,u \\ \sum_{i=0}^{I-1} x_{ij} \times Q_i \le 64\ ,\ j=0,1,...,u \end{cases} \end{cases}$$

$$
s.t\begin{cases}
\begin{cases} x_{ij}=1\text{，基本块 } i \text{ 在流水线 } j \text{ 上} \\ x_{ij}=0\text{，基本块 } i \text{ 不在流水线 } j \text{ 上} \end{cases} \\
\begin{cases} \sum_{i=0}^{I-1} x_{ij}\times p_{in}\times H_i \le 2,\ j=0,1,...,u,\ n=1,2,...,N \\ \sum_{i=0}^{I-1} x_{ij}\times p_{in}\times A_i \le 56,\ j=0,1,...,u,\ n=1,2,...,N \end{cases} \\
\sum_{i=0}^{I-1} x_{ij}\times T_i + x_{i,j+16}\times T_i \le 1\text{，}\ j=0,1,...15 \\
\sum_{i=0}^{I-1} x_{ij}\times p_{in}\times H_i + \sum_{i=0}^{I-1} x_{i,j+16}\times p_{in}\times H_i \le 3\text{，}\ j=0,1,...,15,\ n=1,2,...,N \\
\sum_{j=0}^{u} x_{ij}=1\ \text{，}\ i=0,1,...,I-1 \\
\frac{1}{2}\times\sum_{j=0}^{u}(1+(-1)^j)\sum_{i=0}^{I-1} x_{ij}\times T_i \le 5 \\
\begin{cases} Z_m-Z_n<0\ \text{，}\ s_{mn}=1\text{，}m=0,1,...,I-1\text{，}n=0,1,...,I-1 \\ Z_m-Z_n\le 0\ \text{，}\ s_{mn}=-1\text{，}m=0,1,...,I-1\text{，}n=0,1,...,I-1 \end{cases} \\
Z_m-Z_n\le 0\ \text{，}\ c_{mn}=1\text{，}m=0,1,...,I-1\text{，}n=0,1,...,I-1
\end{cases}
$$

6.2 模型求解

问题要求的基本项流水线排布，无法通过穷举法来求出解，因此我们选用模拟退火算法来解决问题。模拟退火算法的优点就是在解决此类问题有非常广泛的适用性，它可以跳出局部最优解，具有强大的搜索能力，并且稳定性更好，非常适合本问题的求解[4]。

(1) 决策变量设置

首先设置 607 行 100 列的决策变量 A_{ij}，A_{ij} 为 0-1 变量，当 A_{ij}=1 时，表示基本块 i 位于流水线的第 j 级上，反之，则表示基本块 i 不在流水线的第 j 级上。本文将 Gurobi 求出的较优的可行解作为初始解 X_i，这样可以大大地提高求解速度。决策变量设定如图 6-1 所示。

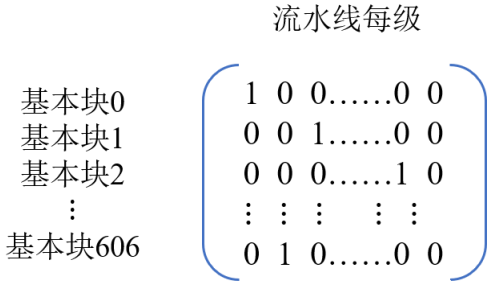

图 6-1 决策变量设定示意图

(2) 确定目标函数

因为题目要求所有基本块在流水线中占用的级数要尽可能少，先设定一个较大的数 J，令每个基本块所在的级数 Z_i 要小于等于 J，再将 J 最小化，即求 $\min J$，这样就

可以得到最优的级数值。

(3) 新界的产生

a) 上下交换法：任选解中 X_i 的一点 A_{pq}，交换其上下相邻的两点 $A_{(p-1)q}$ 和 $A_{(p+1)q}$ 的值左右交换法；

b) 左右交换法：任选解中 X_i 的一点 A_{pq}，交换其上下相邻的两点 $A_{p(q+1)}$ 和 $A_{p(q+1)}$ 的值。

交替使用上面的两种方法，并且所有的状态转移都是在解空间中进行的。

(4) 目标函数差

$$\Delta f = J_{i+1} - J_i$$

(5) 设定接受准则

因为要求所有基本块占用流水线级数的最小值，那么

$$P = \begin{cases} 1, & \Delta f < 0, \\ \exp(-\Delta f / T), & \Delta f \geq 0. \end{cases}$$

(6) 参数选取

降温系数 α =0.999，初始温度 T =1，终止温度 e =10^{-30}，马尔科夫链的长度 L =20000[5]。

模拟退火算法流程图如图 6-2 所示：

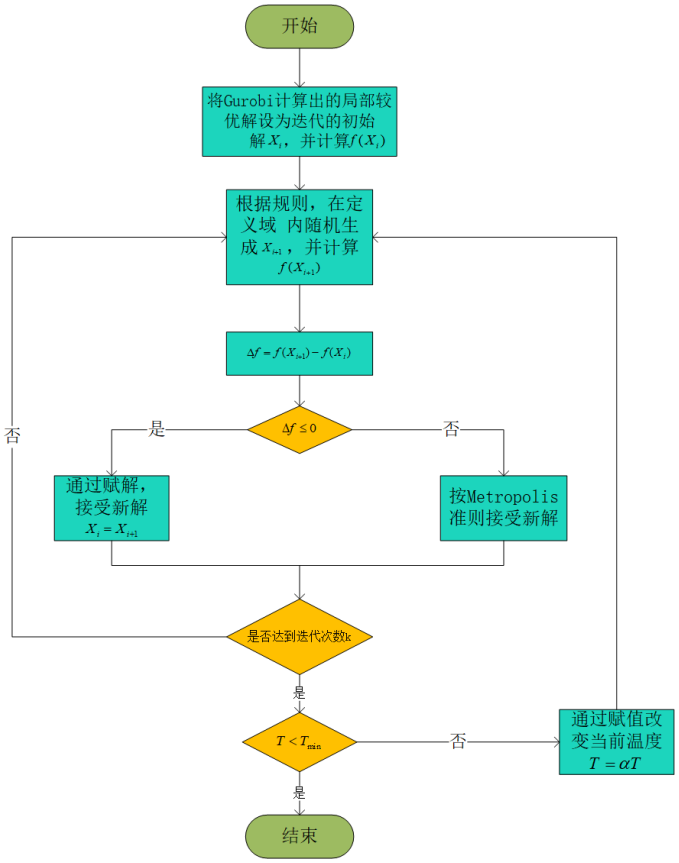

图 6-2 模拟退火算法流程图

6.3 结果展示

根据上述分析，更新线性规划模型，通过模拟退火算法，最终得到最小化所有基本块占用的流水线级数的结果，从第 0 级到第 39 级，共占用 40 级，其中第 28-32 级流水线上未排布基本块。同时输出基本块的部分排布结果如表 6-1 所示，所有基本块的排布情况参见附件。

基本块编号 / 流水线级数	问题二输出基本块的部分排布结果													
第 0 级	13	14	15	16	17	18	19	20	21	22	37	38	58	……
第 1 级	11	12	136	137	144	145	146	147	148	149	150	151	152	……
第 2 级	27	361	362	374	375	382	383	384	385	386	387	388	505	……
第 3 级	28	139	380	534	563	564	581	583						……
第 4 级	553	567	568	569	585	595	596	601						
第 5 级	537	602												
第 6 级	535	546												
第 7 级	53	54	55	141	211	212	213	214	541	549	570	571	572	……
第 8 级	3	32	56	140	216	217	218	219	220	221	226	227	597	……
第 9 级	51	52	142	143	166	188	215	222	223	224	225	229	230	……
第 10 级	39	40	41	42	167	231	296	301	304	323	324			
第 11 级	33	34	36	43	44	45	46	201	232	307	325	326	327	
第 12 级	35	47	48	168	187	189	190	191	192	193	194	195	196	……
第 13 级	31	49	50	198	200	202	205	206	207	316	328	332	333	……
第 14 级	4	60	61	63	203	234	340	342	402	407	469	471	472	……
第 15 级	64	65	235	236	237	238	239	240	241	242	243	244	245	……
第 16 级	62	66	204	208	209	210	228	283	321	343	356	357	358	……
第 17 级	67	366	434	435	436									

表 6-1 问题二输出基本块的部分排布结果

7. 模型的评价与改进

7.1 模型的优点

1. 本文充分理解了数据依赖和控制依赖的定义，并得到了正确的各个基本块的数据和控制关系，在准确表述出资源约束的前提下，建立了合理的流水线级数排布模型。
2. 在建立模型中遇到了不同的困难，一开始确定的目标函数是非线性的，这会导致数据计算的效率大大降低，经过对题目的最小化流水线级数深入分析后，建立了 0-1 线性整数规划模型，大大降低了求解的难度。
3. 由于题目所给数据过于庞大，导致 Lingo 软件无法对文中建立的模型求解，于是本文选择使用 Gurobi 来求解大规模的数学规划，并获得了一个非常好的结果。
4. 由于第二问的求解难度进一步加大，于是采用鲁棒性更强，适应性更好的模拟退火算法，经过分析可知，第一问的解可作为第二问的可行解，将其作为初始解进行迭代，大大提高了求解效率，并求得了一个较优解。

7.2 模型的缺点

问题一约束条件繁多，建立线性规划模型后，利用 Python 调用 Gurobi 优化器求解耗时较多，后续可以进一步思考完善模型并考虑设计其他的启发式算法。

参考文献

[1] 常坤. 协议无关网络编程方法研究及其环境研制[D]. 中国科学技术大学, 2018.

[2] 赵真一, 吴娜, 王晓璇. 基于广度优先搜索的无人飞行器航路自主寻优算法[J]. 第六届中国指挥控制大会论文集 (上册), 2018.

[3] Gurobi Optimization L L C. Gurobi optimizer reference manual[J]. 2018.

[4] 司守奎，孙玺菁. 数学建模算法与应用[M].国防工业出版社:299-303,2011.

[5] 康雯轩. 基于模拟退火算法的共享单车城市配送路径规划[J]. 科技与创新, 2022(000-013).

源程序

题号	数据预处理	代码语言	Matlab

```
clc,clear;
%%求解通路矩阵 tl
%输入有向图
path=xlsread('C:\Users\lwy03\Desktop\2022 年 D 题\有向图.xlsx',1,'B2:WJ608');
%将有向图中的 0 值变为 inf
for i=1:607
    for j=1:607
        if(path(i,j)==0)
            path(i,j)=inf;
        end
    end
end
a=path;
n=size(a,1);
% 初始化距离矩阵
tl=a;
% 初始化路由矩阵
for i=1:n
    for j=1:n
        r(i,j)=j;
    end
end
r;

% Floyd 算法开始
for k=1:n
    for i=1:n
        for j=1:n
            if tl(i,k)+tl(k,j)<tl(i,j)
                tl(i,j)=tl(i,k)+tl(k,j);
                r(i,j)=r(i,k);
            end
        end
    end
    k;
    tl;
    r;
end
%化简通路矩阵
for i=1:607
    for j=1:607
        if tl(i,j)==inf
            tl(i,j)=0;
        end
    end
```

```
end
for i=1:607
    for j=1:607
        if tl(i,j)~=0
            tl(i,j)=1;
        end
    end
end
%%
%输入读写矩阵
[wr,txt]=xlsread('C:\Users\lwy03\Desktop\2022 年 D 题\attachment2.xlsx');
tw=zeros(607,1000);
tr=zeros(607,1000);%tw 为写矩阵、tr 为读矩阵
for i=1:1214
    for j=2:48
    if(mod(i,2)==1)
        if(isnan(wr(i,j))==0)
            wr(i,j)=wr(i,j)+1;
        tw((i+1)/2,wr(i,j))=1;
        end
    else
        if(isnan(wr(i,j))==0)
            wr(i,j)=wr(i,j)+1;
            tr(i/2,wr(i,j))=1;
        end
    end
end
end

%tl 通路矩阵和 t 读写矩阵
s1=zeros(607);%写写
s2=zeros(607);%写读
s3=zeros(607);%读写

%计数器，计算共产生几次约束
j1=zeros(607);%写写
j2=zeros(607);%写读
j3=zeros(607);%读写

%先判断写后写
for i=1:607
    for j=1:607
        if(tl(i,j)==1)
            for k=1:1000
                if(tw(i,k)==1&&tw(j,k)==1)
                    s1(i,j)=5;
                    j1(i,j)=j1(i,j)+1;
                end
```

```
                end
            end
    end
end

%判断写后读
for i=1:607
    for j=1:607
        if(tl(i,j)==1)
            for k=1:1000
                if(tw(i,k)==1&&tr(j,k)==1)
                    s2(i,j)=3;
                    j2(i,j)=j2(i,j)+1;
                end
            end
        end
    end
end
%判断读后写
for i=1:607
    for j=1:607
        if(tl(i,j)==1)
            for k=1:1000
                if(tr(i,k)==1&&tw(j,k)==1)
                    s3(i,j)=1;
                    j3(i,j)=j3(i,j)+1;
                end
            end
        end
    end
end

%将读后写约束置为-1，其他约束置为 1
s4=s1+s2+s3;
for i=1:607
    for j=1:607
        if(s4(i,j)==1)
            s4(i,j)=-1;
        end
        if(s4(i,j)>1)
            s4(i,j)=1;
        end
    end
end

%找到写写、写读依赖
for i=1:607
    for j=1:607;
        if(s4(i,j)~=1)
```

```
                s4(i,j)=0;
            end
        end
end



%找到基本块
for i=1:607
        for j=1:607
            s3(i,j)=s3(i,j)*j;
            s4(i,j)=s4(i,j)*j;
        end
end


c1=zeros(607);
c2=zeros(607);


for i=1:607
        e3=s3(i,:);
        f3=e3(e3~=0);
        for j=1:length(f3)
            c1(i,j)=f3(j);
        end
end

for i=1:607
        e4=s4(i,:);
        f4=e4(e4~=0);
        for k=1:length(f4)
            c2(i,k)=f4(k);
        end
end

for i=1:607
        for j=1:607
            if(c1(i,j)==0)
                c1(i,j)=nan;
            end
            if(c2(i,j)==0)
                c2(i,j)=nan;
            end
        end
end
%c1 读后写  级数-1
for i=1:607
        for j=1:607
            if(isnan(c1(i,j))~=1)
```

```
                c1(i,j)=c1(i,j)-1;
            end
            if(isnan(c2(i,j))~=1)
                c2(i,j)=c2(i,j)-1;
            end
        end
end
```

题号	问题一	代码语言	Python

```
from gurobipy import *
import xlrd, xlsxwriter, datetime, math, xlwt
import numpy as np
import gurobipy as gp
from gurobipy import GRB

################################################################################读取数据约束
data = xlrd.open_workbook('dd.xlsx')
table = data.sheet_by_index(0)
dd = []
for i in range(table.nrows):    # table.nrows 表示总行数
    line = table.row_values(i)   # 读取每行数据，保存在 line 里面，line 是 list
    dd.append(line)   # 将 line 加入到 resArray 中，resArray 是二维 list
dd = np.array(dd)   # 将 resArray 从二维 list 变成数组

################################################################################读取控制约束
data = xlrd.open_workbook('cd.xlsx')
table = data.sheet_by_index(0)
cd = []
for i in range(table.nrows):    # table.nrows 表示总行数
    line = table.row_values(i)   # 读取每行数据，保存在 line 里面，line 是 list
    cd.append(line)   # 将 line 加入到 resArray 中，resArray 是二维 list
cd = np.array(cd)   # 将 resArray 从二维 list 变成数组

################################################################################读取资源约束
data = xlrd.open_workbook('rd.xlsx')
table = data.sheet_by_index(0)
T = []
H = []
A = []
Q = []
J = []
T = table.col_values(0)
T = np.array(T)
H = table.col_values(1)
H = np.array(H)
A = table.col_values(2)
A = np.array(A)
Q = table.col_values(3)
Q = np.array(Q)
data = xlrd.open_workbook('J.xlsx')
table = data.sheet_by_index(0)
for i in range(table.nrows):
    line = table.row_values(i)
```

```
        J.append(line)
J = np.array(J)

Ip = 607
Jp = 80
print(type(J))
#################################################################################依赖读取完毕


#################################################################################开始模型构建
m = gp.Model('basic_project_distribution_model')
X = m.addVars(Ip, Jp, vtype=GRB.BINARY, name='X')
M = m.addVar(vtype=GRB.INTEGER, name='M')
#Z = m.addVars(1, 607, vtype=GRB.INTEGER, name='Z')

# Limit how many solutions to collect
#m.setParam(GRB.Param.PoolSolutions, 1)

# Limit the search space by setting a gap for the worst possible solution
# that will be accepted
#m.setParam(GRB.Param.PoolGap, 0.1)

# do a systematic search for the k- best solutions
#m.setParam(GRB.Param.PoolSearchMode, 2)




#   ###################################资源约束
for j in range(Jp):
        m.addConstr(gp.quicksum(X[i, j] * T[i] for i in range(Ip)) <= 1, 'tc')
for j in range(Jp):
        m.addConstr(gp.quicksum(X[i, j] * H[i] for i in range(Ip)) <= 2, 'hc')
for j in range(Jp):
        m.addConstr(gp.quicksum(X[i, j] * A[i] for i in range(Ip)) <= 56, 'ac')
for j in range(Jp):
        m.addConstr(gp.quicksum(X[i, j] * Q[i] for i in range(Ip)) <= 64, 'qc')

#   ###################################基本块只能排布到一级
for i in range(Ip):
        m.addConstr(gp.quicksum(X[i, j] for j in range(Jp)) == 1, 'oc')

#   ###################################折叠资源约束
for j in range(15):
        m.addConstr(gp.quicksum(X[i, j] * T[i]+X[i, j+16] * T[i] for i in range(Ip)) <=1, 'zc')

for j in range(15):
        m.addConstr(gp.quicksum(X[i, j] * H[i] + X[i, j+16] * H[i] for i in range(Ip)) <=3, 'ic')

#   ###################################数据依赖约束


for a in range(Ip):
        for b in range(Ip):
```

```
            if dd[a, b] == 1:
                m.addConstr(gp.quicksum((J[a, j] * X[a, j] - J[b, j] * X[b, j]) for j in range(Jp)) <= -1)

for a in range(Ip):
      for b in range(Ip):
            if dd[a, b] == -1:
                m.addConstr(gp.quicksum((J[a, j] * X[a, j] - J[b, j] * X[b, j]) for j in range(Jp)) <= 0)

#   ####################################控制依赖约束
for a in range(Ip):
    for b in range(Ip):
            if cd[a, b] == 1:
                m.addConstr(gp.quicksum((J[a, j] * X[a, j] - J[b, j] * X[b, j]) for j in range(Jp)) <= 0)


#   ####################################偶数级约束

m.addConstr(gp.quicksum((1+(-1)**j) * gp.quicksum(X[i, j] * T[i] for i in range(Ip))for j in range(Jp)) <=
10)

for i in range(Ip):
      m.addConstr(gp.quicksum(X[i, j] * J[i, j] for j in range(Jp)) <= M)

m.setObjective(M, GRB.MINIMIZE)
#m.setParam('Timelimit', 4200)

m.optimize()

#############################################################################

m.write("mip1.sol")
```