MASTER THESIS XIDIAN UNIVERSITY

西安电子科技大学

硕士学位论文

带释放时间的可分任务调度优化模型与算法研究

作者姓名 蔡坤

指导教师姓名、职称 王宇平 教授

申请学位类别 工学硕士

中国知网 https://

中国知网 https://www.cnki.net

学校代码 10701 学 号 1303121669
分 类 号 TP30 密 级 公开

西安电子科技大学

硕士学位论文

带释放时间的可分任务调度优化模型与算法研究

作者姓名：蔡坤

一级学科：计算机科学与技术

二级学科：计算机软件与理论

学位类别：工学硕士

指导教师姓名、职称：王宇平 教授

学 院：计算机学院

提交日期：2015 年 11 月

中国知网 https://www.cnki.net

Research on Release-Time Aware Divisible-Load Scheduling Models and Algorithms

A thesis submitted to

XIDIAN UNIVERSITY

in partial fulfillment of the requirements

for the degree of Master

in Computer Software and Theory

By

Cai Kun

Supervisor: Wang Yuping Professor

November 2015

中国知网 https://www.cnki.net

摘要

随着信息科学技术和计算机科学的飞速发展，系统对存储、计算速度和带宽的要求也在不断的增加，单一的计算节点已经无法满足很多大规模计算密集型应用的需求，并行与分布式平台应运而生。任务调度问题是并行与分布式平台研究和应用必须解决的一个关键问题。高效的任务调度算法能够根据并行与分布式平台资源和任务的不同情况，在满足一定约束的情况下，达到任务完成时间最短的目标。但是由于分布式平台需要考虑诸多的因素，情况也比单一节点复杂，如何设计出高效的任务调度算法成为当前的一个研究热点。

可分任务理论的提出给了设计高效的任务调度算法一种很简单直观且有效的思路，它在模型的简单性和精确性之间取得了很好的折衷，因而成为并行与分布式平台下任务调度相关问题的一个研究热点。本文主要侧重并行与分布式系统下，可分任务调度模型和算法的研究，主要工作可概括如下：

1、并行与分布式系统下的可分任务调度问题已经被证明是 NP-hard 问题，而遗传算法作为一种启发式随机算法，已经被广泛用于 NP-hard 问题的求解并且有着不错的效果。本文针对带释放时间的可分任务调度问题，建立了新的任务调度模型，然后设计了新的遗传算法对模型进行求解，实验结果表明了模型的有效性和算法的高效性。

2、已有的可分任务调度算法大多假设处理机在任务分配开始时刻全部处于空闲状态，然而在实际的并行与分布式系统中，新的任务到来时，很多处理机可能还处于忙碌状态。每台处理机从忙碌状态转到空闲状态的等待时间一般是不同的，即处理机具有不同的释放时间。本文针对同构系统下带释放时间的可分任务调度问题，详细分析了三种不同约束条件下的任务调度过程，进而提出了一种新的带释放时间的可分任务调度模型，并设计了高效的全局优化遗传算法对其进行求解。实验结果表明所提出的算法能够根据各处理机的释放时间合理分配任务，避免了任务的等待时间，从而减少了任务的完成时间。

3、实际的并行与分布式系统多由异构的处理机构成，而处理机的调度顺序对于任务的完成时间至关重要。本文针对异构系统下带释放时间的可分任务调度问题，详细分析了三种不同约束条件下的任务调度过程，进而提出了一种新的带释放时间并同时考虑处理机最优调度顺序的可分任务调度模型。为了求解该模型，我们设计了新的全局优化遗传算法，并设计新的编码解码和遗传算子，同时在算法中引入了局部搜索策略以加快其收敛速度。实验结果表明，本文所提算法与已有算法相比能获得更短的任务完成时间。

中国知网 https://www.cnki.net

关 键 词：可分任务调度， 遗传算法， 并行与分布式系统， 释放时间， 混合时序约束

中国知网 https://www.cnki.net

ABSTRACT

With the rapid development of computer science and technology, the requirements for memory, computational speed and bandwidth are constantly increasing. A single computing node has been unable to meet the needs of many application scenarios, which led to the development of parallel and distributed computing. One of the key issues in parallel and distributed systems is finding an efficient task-scheduling strategy so that the processing time of the entire workload could be minimized. However, designing an efficient task scheduling strategy under parallel and distributed systems is much more complex than under a single processor, because there are lots of factors need to be considered. Due to its complexity and importance, the designing efficient task-scheduling algorithms has become a hot topic in recent years.

Divisible load theory (DLT) gives a simple, intuitive and effective idea of designing efficient task-scheduling algorithm, and gets a very good compromise between the simplicity and accuracy of scheduling models. Thus designing efficient task-scheduling algorithms becomes a hot topic in the research domain of parallel and distributed systems. This paper mainly focuses on studying divisible-load scheduling models and algorithms in parallel and distributed systems and the main works can be summarized as follows:

1. Divisible load scheduling(DLS) in parallel and distributed system has been proven to be an NP-hard problem, and genetic algorithms(GAs), as a kind of random heuristic algorithm, have been widely used in solving a variety of NP-hard problems and achieved a great success. To solve the divisible-load scheduling problem with release times of processors, we build new models and design genetic algorithms in this paper. Experimental results show the effectiveness of the proposed models and the efficiency of the proposed algorithms.

2. Most existing divisible-load scheduling models assume that all processors are idle at the beginning of workload assignment. However, in the real parallel and distributed systems, many processors may be still in the busy state when a new workload arrived. Processors may have different waiting time from the busy state to the idle, that is, processors have different release time. For the release-time aware divisible-load scheduling problems under

中国知网 https://www.cnki.net

homogeneous distributed systems, first a detailed analysis of divisible-load scheduling under three different time constraints is made, and then a novel release-time aware divisible-load scheduling model with hybrid time constraints is proposed. To solve the proposed model, we design a new genetic algorithm. Experimental results show that the proposed algorithm can divide and distribute the workload to processors according to their release times, which avoids the waiting time of the workload computation, thus reducing the processing time of the entire workload.

3. The realistic parallel and distributed systems are often composed by heterogeneous processors and the distribution sequence of processors plays a significant role in the processing time. For the release-time aware divisible-load scheduling under heterogeneous distributed systems, first a detailed analysis of divisible-load scheduling under three different time constraints in heterogeneous distributed systems is made. Then a new release-time aware divisible load scheduling model taking into account the optimal distribution sequence is proposed. To solve the model, an effective global optimal genetic algorithm is designed with new encoding scheme and genetic operators. Meanwhile, a new local search strategy is put forward in order to accelerate the convergence speed of the proposed algorithm. Experimental results show that compared with the existing scheduling algorithms, the proposed algorithm can obtain shorter processing time.

Keywords: Divisible load scheduling, Genetic algorithm, Parallel and distributed system, Release-time, Hybrid time constraints

中国知网 https://www.cnki.net

插图索引

中国知网 https://www.cnki.net

表格索引

中国知网　https://www.cnki.net

符号对照表

符号	符号名称
N	系统中从处理机数目
P_0	主处理机
P_i	从处理机
l_i	主处理机与从处理机 P_i 之间的链路
r_i	释放时间
W_{total}	总任务大小
α_i	从处理机 P_i 分配到的任务大小
E	同构系统通信启动开销
F	同构系统计算启动开销
z	同构系统链路传输单位大小任务所花费的时间
w	同构系统处理机计算单位任务所需的时间
s_i	从处理机 P_i 的开始时刻
I	编码向量
C	混合时序约束条件
z_i	异构系统链路 l_i 传输单位大小任务所花费时间
w_i	异构系统处理机 P_i 计算单位任务所需的时间
e_i	异构系统链路 l_i 通信启动开销
f_i	异构系统处理机 P_i 计算启动开销

中国知网 https://www.cnki.net

缩略语对照表

缩略语	英文全称	中文对照
DLT	Divisible Load Theory	可分任务理论
GA	Genetic Algorithm	遗传算法
GP	Genetic Programming	遗传编程
ExA	Exhaustive Algorithm	穷举算法

中国知网 https://www.cnki.net

目录

中国知网 https://www.cnki.net

中国知网 https://www.cnki.net

第一章 绪论

随着信息科学技术和计算机科学的飞速发展，数据的规模也在以惊人的速度不断增长，单一的服务器已经远远不能满足信息科学疯狂的增长速度。更强，速度更快，性能更高的计算机系统受到更多的关注，故而工作站集群、无线传感网、网格等基于网络并行化和分布式的系统应运而生。网络中任务对于资源的需求复杂多样，有的是计算密集型任务，对 CPU 的要求高；有的是 I/O 密集型任务，对网络带宽需求大。针对这些不同的任务，如何高效的调度它们，使得在充分利用网络平台的计算资源和网络带宽等资源的基础上，达到任务完成时间最短的目标是并行与分布式平台的一个及其关键的问题[1]。在并行与分布式系统中，任务调度是一项十分困难而且复杂但是又是极其重要的工作，它与系统拥有的资源、处理的应用模型以及调度的目标密切相关。鉴于此，我们希望建立一个相对精确的数学模型，来解决我们的问题。然而，由于任务调度问题往往过于复杂，建立精确的任务调度模型是不现实甚至几乎是不可能的，因此在实际应用中的任务调度，我们通常需要对调度的任务和资源进行一些合理而必要的假设。即使对于简化以后的任务模型，任务调度问题也已经被证明是 NP-hard 问题[2]。作为并行与分布式平台研究中的重要研究方向之一，任务调度问题受到了国内外学者广泛关注和研究。本文基于可分任务理论，建立了并行与分布式系统的考虑启动开销和释放时间的可分任务调度模型。

1.1 选题背景和意义

可分任务的研究开始于 1988 年，在分析传感器网络数据中的测试数据时，通过对并行与分布式系统带来的好处与相应的任务传输时不可忽略的通信开销进行详细分析后，Robertazzi 等人提出了可分任务理论(Divisible Load Theory，可简记为 DLT)[3]。原来属于 NP-hard 问题的传统任务调度问题，经过可分任务理论的解析，可以得到一种简单且易求解的解决方案，因而近年来得到了广泛的关注。可分任务理论主要有两个基本的假设：第一，任务传输到各个处理机上的所需的传输时间与要传输任务的大小成正比；第二，任务在处理机上的计算时间与分配在该处理机上任务的大小成正比；第三，一个大任务可以被任意切分成若干个且大小任意的任务，切分后的任务可以分配到任意一台处理机进行处理，相互独立，互不影响[4-6]。

可分任务理论给出了一种简明清晰且易于优化解决问题的思路，其在模型的简单性和精确性之间达到了很好的平衡。因为这些优点，可分任务调度模型与算法的研究已经成为最近这些年来任务调度领域的一个研究热点[7-8]。可分任务调度被广泛的应

中国知网 https://www.cnki.net

用于一些大型实验数据的处理、DNA 测序分析、图像处理、数据密集型的科学研究计算、信号处理、数据挖掘以及计算生物信息学等等。

1.2 国内外研究现状

并行与分布式平台的拓扑结构多种多样，有星型网络，总线型网络，阻塞型网络和非阻塞型网络。任务调度模型的建立与网络拓扑结构息息相关，可分任务理论（DLT）能够有效的描述各种不同拓扑结构的并行与分布式平台，能很大程度上简化模型的建立。

文献[4,7-8]详细分析探讨了可分任务理论中的一些基本概念，介绍了可分任务理论在大规模数据处理、图像处理、视频处理以及网络处理等等领域内的应用。基于同构线性网络，Mani，Ghose 等人[9]通过详细的数学分析，建立了以最短完成时间为目标的，综合讨论总线型网络拓扑结构下带前后端和不带前后端的，任务位于中间处理机和边缘处理机等情况下的紧式(Closed-form)最优解。文献[10]和文献[11]详细分析了同构树形网络和总线型网络拓扑结构下，带前后端和不带前后端，且以最短完成时间为目标的，考虑通信开销的可分任务调度模型，最后分析得到模型的渐近最优解。实际的并行与分布式平台大多数是异构的，即系统中的每个处理机有着不同的计算速度，网络传输速率，通信启动开销以及计算启动开销。针对异构系统树形网络拓扑结构，以最短完成时间为目标的情况，Kim[12]等人详细分析了处理机的调度顺序对最短完成时间的影响，并给出了求解最优调度顺序的算法，同时证明了异构树形网络拓扑结构下，处理机的最优调度顺序与传输链路速度之间有很大的关系。Bharadwaj，Ghose 等人[13]详细分析了异构系统星型网络拓扑结构下，以最短完成时间为目标的可分任务调度模型，得到了任务完成时间的紧式最优解，同时还证明了当处理机的调度顺序按照链路传输速度递减时，模型能够达到最短完成时间。启动开销的存在对任务调度模型有着较大的影响，当前的一些研究已经将传输启动开销和计算启动开销考虑在内了。文献[14]基于总线型网络拓扑结构，以最短完成时间为目标，在考虑了传输启动开销和计算启动开销的情况下，深入分析了调度顺序的影响因素，并且证明了当处理机调度按照计算速度递减时，能够得到最短的任务完成时间。文献[15]考虑了存在计算启动开销且处理机配备了前后端的情况下，分析了总线型网络的最优任务调度方案，并且第一次证明了当所有处理机同时完成时，此时能得到最短完成时间这一重要规律。文献[16]基于总线型网络拓扑结构，在考虑了计算启动开销和通信启动开销的情况下，得到了最短完成时间的一个紧式表达式，并且证明了参与计算处理机数目和处理机顺序对最短完成时间的重要影响。

当前大多数工作都是基于阻塞式通信模型，即对于每台处理机，只有当它接收完

中国知网 https://www.cnki.net

给它分配的任务之后，才能进行任务的处理工作，然后在实际的系统中，可能存在一些实时流数据处理，在这种场景下，阻塞式的通信模型不再有效，这种处理机一边接收数据和任务一边执行处理操作称为非阻塞式通信模型。文献[17]详细分析了同构并行与分布式系统下，总线型和树形网络拓扑结构的基于非阻塞式通信模型的可分任务调度算法。文献[18]分析了异构并行与分布式系统下的非阻塞通信模型，通过详细的分析得到了任务最短完成时间紧式最优解表达式，并且证明了非阻塞式通信模型优于阻塞式通信模型的地方。文献[19]在此基础上，考虑了处理机的启动开销，在详细的讨论和分析之后证明最优处理机调度顺序与链路传输速度从大到小得到的顺序相同。

文献[20]讨论了同构并行与分布式系统下，多趟任务调度模型的建立和求解，通过详细的数学分析，最终得到优化处理机的选择方法和相应的趟数。文献[21]讨论了异构并行与分布式平台下的总线型网络拓扑结构以最短完成时间为目标的多重任务调度问题，通过引入一些条件来建立模型，最后获得了不错的结果。尚明生[22] 研究了异构并行与分布式平台，总线型拓扑网络结构的情况下负载的优化调度问题，通过详细分析和讨论处理机选择、任务分配顺序和各个处理机分配任务数量，得出结论表明：当按照处理机计算速度递减的顺序调度处理机时，能够使得任务完成时间最短，即得到最优的调度方案。Ghose[23]等人在信息和参数缺失的情况下，提出探针技术来估计参数值，并提出一种自适应算法来划分任务，最终得到了很好的结果。文献[24]分析了同构和异构等并行与分布式系统，星型网络拓扑结构的多重可分任务调度问题，分析表明在一些特殊情况下，求解需要多项式的时间复杂度，在其他情况下不能在多项式时间得到解决方案。文献[25]详细分析了异构并行与分布式平台下的静态和稳态调度技术，并分析了它们的局限性。更多关于稳态可分任务调度的分析在文献[26-28]被引入，文献[26]介绍了稳态可分任务调度的应用场景，并详细分析了其优点和局限；文献[27]和文献[28]分析了异构并行与分布式系统下以最大吞吐量为目标的稳态可分任务调度模型。[29]考虑了总线型网络中，释放时间和处理机内存是有限的这两个现实场景中常遇到的因素，提出了基于上面约束条件下的多趟调度算法，并设计出了一个高效的算法进行模型的推导和求解。但是该算法具有模型复杂，难于推导的缺点，并且该算法没有考虑处理机存在网络启动开销和计算启动开销的情况，而且文中也强调了网络启动开销和计算启动开销存在的情况下，算法会受到很大的影响。文献[30]提出了总线型网络条件下的多趟调度算法，该算法考虑了很多现实场景中的因素，例如任务初始化时间和任意的处理机释放时间，并设计出了一个很高效的算法，但是该算法具有模型复杂，难于推导的缺点，并且该算法没有考虑处理机存在网络启动开销和计算启动开销的情况，我们通过证明，已经确定了当存在网络启动开销和计算启动开销的情况下，算法是得不到正确解的。文献[31]基于异构并行与分布式系统星型网络拓扑结构，并且假定处理机存储是有限的，研究了大小独立任务和可分任务两种情

中国知网 https://www.cnki.net

况下任务调度算法的相关问题。文献[31-37]分析了存储受限情况下的可分任务调度模型的建立和求解方法。文献[38]详细分析了同构与异构并行与分布式系统下多趟可分任务调度算法的模型建立和求解方法，并且得到了比以往研究成果更好的解决方案。文献[39]全面分析了各种情况下可分任务调度模型的建立。文献[40]和文献[41]详细讨论了多趟可分任务调度中三个重要的问题：怎样处理启动开销，怎样处理异构平台和如何进行多趟调度，最后作者提出了 UMR 算法，很好的解决了上述三个重要问题。在一些并行与分布式平台中，存储不是无限大的，即存储是受限的，文献[42]给出了一种用暴力穷举方法来求解同构总线型网络的可分任务调度问题，在该文章中，作者详细讨论了处理机开始时间与释放时间之间的种种关系，最后抽象成简洁清晰的数学模型，即抽象成矩阵运算的形式，最后用穷举法列举所有开始时间和释放时间之间的情况，并用 LP 进行求解。该文章所建立的模型简单清晰，易于求解，但是算法存在巨额的时间开销，另外模型也没有考虑到网络启动开销和计算启动开销。文献[43]详细讨论和分析了在线性链路网络条件下，考虑处理机含有相同的释放时间和含有不同的释放时间的两种情况，处理机完成时间最短需要满足的优化条件，并提出了两种贪心算法用以求解问题的近似最优解。该文章对各种情况分析比较到位，但是也有模型复杂的缺点，并且该算法没有考虑处理机存在网络启动开销和计算启动开销的情况。

多趟调度在某种程度上能够提高系统的性能，但是多趟调度本身的解决方案还处于探索阶段，最优趟数的确定和求解仍是一个很大的难点。如何建立简单高效的多趟可分任务调度模型是未来的一个研究方向，本文重点在于单趟任务调度的研究。

近些年来，越来越多的专家和学者投入到可分任务调度理论和应用的研究中，其中并行与分布式系统下的可分任务调度问题应用前景广泛，已成为并行调度领域重要的研究方向之一。

1.3 本文研究内容及组织框架

1.3.1 研究内容与成果

前面两节，我们分析了可分任务理论在信息科学研究和应用方面的诸多成果。可分任务调度以其简单直观精确的优点在并行于分布式平台任务调度领域有着重要的应用。当前已经取得了一些不错的成果，但还有很多有待解决的问题，可分任务调度的模型与算法的研究吸引了众多的学者。本文的研究成果如下：

1、已有的可分任务调度模型和算法大多假设处理机在任务分配开始时刻全部处于空闲状态且没有启动开销存在，然而在实际真实的并行与分布式系统下，当新的任务到来时，处理机可能还处在忙碌的状态，并且每次传输和计算开始前都会有一个启动开销。每个处理机从忙碌状态转到空闲状态的等待时间一般是不同的，也即处理机

中国知网 https://www.cnki.net

可能具有不同的释放时间，启动开销跟处理机有关系，若处理机相同，则启动开销相同，若处理机不同，则会有着不同的启动开销。本文的模型与算法研究中将处理机存在释放时间和启动开销这些因素考虑了进去，同时考虑到遗传算法是一种启发式随机算法，在组合优化问题的求解方面有着很好的表现，因此本文选择遗传算法来求解可分任务调度模型，并取得了很好的结果。

2、研究了同构并行与分布式平台星型网络拓扑结构下的可分任务调度模型建立与算法，详细分析了三种不同时序约束下的可分任务调度，建立了一种新的考虑处理机释放时间和启动开销的混合时序约束可分任务调度模型，并采用遗传算法对模型进行求解。实验证明，同已有的穷举算法相比，本文提出的算法在得到最优解的同时拥有很好的时间性能。

3、实际生活中异构并行与分布式平台大量存在，本文研究和分析了异构并行与分布式平台星型网络拓扑下考虑处理机释放时间和启动开销的可分任务调度问题，建立了异构并行与分布式平台下的混合时序约束可分任务调度优化模型，根据模型的特点，设计了新的编码解码方案和交叉变异等遗传算子，由于问题的搜索空间很大，为了加快算法的收敛速度，本文引入了局部搜索算子来加快算法的收敛。实验结果表明，同已有的一些异构并行与分布式平台下的调度方案相比，本文的方法能够得到更好的调度方案。

4、分析了可分任务调度领域内当前的一些成果、存在的问题和发展趋势，对未来可分任务调度的研究方向作了总结和展望。

1.3.2 内容组织框架

论文共分为五章，安排如下：

第一章 绪论，主要介绍了可分任务调度研究的背景和意义，概述了当前国内外学者的研究现状和研究趋势，最后给出了本文研究的主要内容和成果。

第二章 遗传算法简介，详细介绍了遗传算法的起源与发展，遗传算法的基本算法思想和概念，遗传算法框架和设计流程，最后给出了遗传算法的特点和应用场景。

第三章 带释放时间的同构系统可分任务调度模型及算法，给出了同构并行与分布式平台星型拓扑网络结构下考虑处理机释放时间和启动开销的混合时序约束可分任务调度模型，建立了以完成时间最短为目标的任务调度模型，并引入了遗传算法这一启发式算法对该模型进行求解，最后通过实验证明算法的正确性和高效性。

第四章 带释放时间的异构系统可分任务调度模型及算法，提出了异构并行与分布式平台星型网络拓扑结构下考虑启动开销和处理机释放时间的混合时序约束可分任务调度模型，建立了以完成时间最短为目标的优化模型，并对该任务调度模型设计了新的遗传算法，最后进行了实验仿真验证和理论分析。

中国知网 https://www.cnki.net

第五章 总结和展望，主要对本文所做的研究工作进行了简单的总结，提出了未来可以深入研究和改进的工作的展望和设想。

1.4 本章小结

本章内容主要概述了所研究的并行与分布式平台下可分任务调度问题的选题背景和意义，然后详细介绍了并行与分布式平台下可分任务调度这一领域已有的研究成果和实际应用，接着介绍了本文的一些研究成果，最后给出了本文的组织框架。

中国知网 https://www.cnki.net

第二章 遗传算法

2.1 遗传算法的起源与发展

遗传算法 GA（Genetic Algorithm）是模拟达尔文（Darum）“物竞天择，适者生存”生物进化论的自然选择和遗传学机理的生物进化过程的计算模型。遗传算法通过模拟大自然进化法则来搜索问题的最优解，也是一种随机迭代搜索的智能优化方法。它最初是由美国 Michigan 大学的 J.Holland 教授于 1975 年提出来的[44]。

遗传算法研究起源于 60 年代，人们在对自然和人工自适应系统的研究中提炼出了遗传算法最初的模型。生物学家 Fraser 在他的 1962 年论文中提出了遗传算法最早的模型 [45]，在论文中，他用计算机模拟了带有突变和选择的进化过程，并通过实验证明了选择算子对遗传算法的重要影响。1967 年，Bagley 在他的博士论文中首次使用了遗传算法 GA（Genetic Algorithm）一词，论文中提出的交叉、变异和选择等操作与现今的遗传算法已经十分相似了。他还观察到在遗传算法不断进化过程中，针对不同阶段的不同情况，采用不同的选择概率可以有效防止遗传算法的过早收敛，同时提出了遗传算法参数自动调整的概念。

进入 70 年代，Halland 教授出版了他的第一本遗传算法相关的专著《Adaption in Nature and Artificial Systems》，使得 GA（Genetic Algorithm）的概念逐渐为人所知。《Adaption in Nature and Artificial Systems》一书详细阐述了遗传算法的基本理论和方法，并提出了奠定遗传算法理论基础的隐形并行性原理和模式定理。由于遗传算法的隐形并行性，遗传算法很适合并行化操作，这一特性也成为必要时刻提升遗传算法运行速度的有效途径。

到了 80 年代，遗传算法已经在诸多领域中得到了很好地应用。Goldberg[46,47]将遗传算法用于管道系统的优化和机器学习问题，通过遗传算法模拟复杂多变管道系统的控制，得出了一套针对管道系统供气问题的解决方案，是遗传算法应用的一个良好范例。Fitzpatrick 等人将遗传算法应用于医学图像变换问题，取得了很好的结果。Axelord 和 Forrest 研究了遗传算法在博弈论经典问题囚徒困境中的应用[48,49]。

80 年代末，遗传算法的发展达到了高潮，很多理论和实验成果不断涌现出来。Goldberg 出版了专著《搜索、优化和机器学习中的遗传算法》[50]，该书全面讲述了遗传算法的原理和应用，奠定了遗传算法的理论基础。1992 年，Koza 提出了遗传编程 GP（Genetic Programming）的概念。遗传编程将程序片段作为进化的对象，即种群中的个体设定为一段程序，然后将遗传算法应用于计算机程序的优化，最终不断进化成为更优的计算机程序。

中国知网 https://www.cnki.net

经过几十年的努力，遗传算法无论在算法设计，理论基础还是在应用研究上都取得了很大的进步，已经成为了计算机科学、运筹学和应用科学等诸多学科所共同关注的热点研究领域。

2.2 遗传算法的基本思想

遗传算法的思想受遗传学理论的启发，主要由编码、解码、交叉、变异和选择等遗传算子组成，虽然子过程众多，但是框架结构比较清晰。遗传算法求解问题的基本框架如图 2.1 所示：

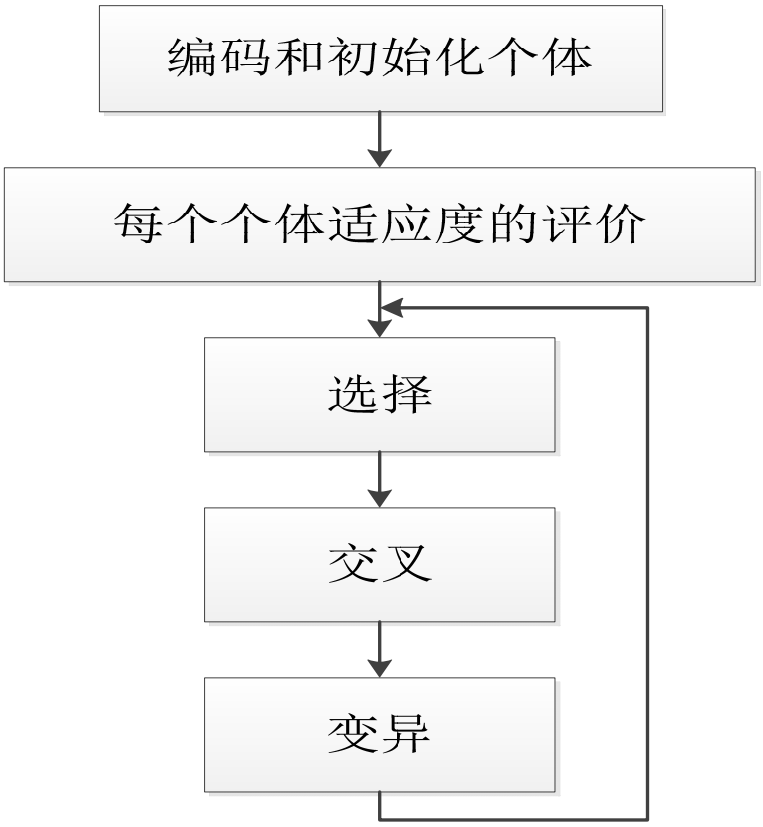

图2.1 遗传算法基本框架

遗传算法的框架的主要步骤如下：

1. 根据问题，设计对应的遗传算子，即明确遗传算法的编码、解码、交叉、变异、选择等子过程需要进行的操作，并确定遗传参数的数值；
2. 初始化总群，随机产生若干个符合要求的个体作为种群，该种群代表遗传空间可能解的集合；
3. 对种群中的每个个体进行适应度的评价，为后面选择过程提供依据；
4. 按一定的概率从种群中选择若干个个体用于下一步的进化，适应度值越接近

中国知网 https://www.cnki.net

最优解有更大的概率被选中。

5. 对种群中的个体进行交叉操作，随机选择每个个体是否参与交叉操作，并将两个参与交叉操作的个体进行基因重组，产生新的子代个体；
6. 对种群中的个体进行变异操作，以一个较小的概率从种群中选择一些个体，对其中的一些基因位进行变异操作；
7. 若达到了遗传算法终止条件，则终止算法，通过对种群中最好的个体进行解码操作即可得到问题的最优解；若没有达到终止条件，则转到第三步，继续运行算法。

2.3 遗传算法的基本概念

为便于对遗传算法的理解，首先，将遗传算法的一些基本概念和术语[51]描述如下：

种群：用来表示问题对应的多个解的集合。

种群规模：种群中个体的数目。种群的规模对算法收敛速度有直接的影响，且在遗传算法进化过程中一般是不变的。

染色体：种群中的元素称为个体或染色体。

父代及子代：在遗传算法进化过程中，选择当前解进行交叉变异等操作以产生新解，当前解称为新解的父代，新解称为当前解的子代。

编码：将实际问题的解空间映射到遗传算法解空间的过程称为编码。常见的编码方法有二进制编码、实数编码和符号编码。

解码：编码的逆过程，用于将遗传算法的解映射到实际问题的解。

基因位：染色体中每一位都称为基因位。

适应度：根据模型的目标函数确定适应度函数，代表了个体适应环境的程度，适应度越接近最优解的个体有更大的概率存活下来。

交叉：遗传算法产生新的个体基因型的主要算子，从两个染色体中各取几个基因片段，组合成两个新的染色体的过程。常见的交叉算子主要有单点交叉、双点交叉、多点交叉、均匀交叉、部分匹配交叉等。

变异：一个个体基因被其等位基因替换，从而产生出新的染色体的过程。大自然中变异出现几率比较小，因而种群中变异的个体一般也比较少，但是对生物的进化和多样性有着重要的意义。常见的变异算子主要有位点变异、对换变异、插入变异等。

选择：从种群中选择一些个体进行下一代的进化，适应度越接近最优解的个体有更大的概率在选择过程被选中，用于遗传算法的下一步进化。常见的选择算子主要有轮盘赌轮选择、排序选择法、联赛选择法等。

中国知网 https://www.cnki.net

精英保留：强制在选择操作前保留最好的几个个体用于下一步的进化。遗传算法引入精英保留的目的是避免种群中最好的一些个体意外丢失，从而达到加快收敛速度的目的。

2.4 遗传算法的特点与应用

2.4.1 遗传算法的特点

作为被广泛使用的随机化搜索算法，遗传算法具有如下优点：

1. 种群搜索特性。传统的搜索算法大多都是单点搜索，从单个个体初始值开始逐步求得最优解，然而传统的搜索算法有容易陷入局部最优解的缺点。遗传算法从含多个个体的种群中进行搜索，种群中不断相互作用，可以有效避免传统优化中容易出现的陷入局部最优解的问题，利于全局择优。

2. 适合先验知识少的情况。遗传算法是一种启发式的随机算法，仅用适应度的数值来评价个体的好坏，求解时会不断趋向于更优的解决方案，需要的先验知识少，具有很强的适用性，使得其应用范围大大扩展。

3. 隐含的并行性。遗传算法种群含有多个个体，在个体交叉时需要两个个体相互作用，其他情况下，个体相互独立，故而可以同时对多个个体进行操作，即运用并行化的思想，这样可以实现对种群的快速进化，为提高遗传算法性能提供了一种思路。

4. 随机搜索特性。遗传算法是一种随机搜索算法，通过与概率相关的随机数来指导种群的进化，进化方向复杂多变，不容易陷入局部最优解。

5. 可扩展的能力。遗传算法可以很好地和别的框架组合使用，有效的利用不同框架的优点，从而得到更好的解决方案。

同时，遗传算法也有一些不足之处：

1. 遗传算法存在编码的不规范以及编码存在表示的不准确性。

2. 遗传算法容易出现过早收敛。

3. 遗传算法对其算法的精度、计算复杂度等方面还没有很有效的度量分析方法。

2.4.2 遗传算法的应用

遗传算法的整体搜索策略和搜索方向是根据模型的目标函数制定的，在遗传算子适应度函数中得到体现，在优化时不需要其他的辅助信息，故而遗传算法具有很好的适用性。遗传算法不依赖问题的具体领域，对问题的种类也没有很高的要求，所以在许多学科中得到了广泛的应用，如函数优化、遗传编程、车间调度、组合优化、数据挖掘、机器学习、图像处理、博弈论中的囚徒困境、人工生命等。

中国知网 https://www.cnki.net

2.5 本章小结

本章详细介绍了遗传算法的起源与发展、基本思想、基本要素，阐述了遗传算法的特点及其应用。鉴于遗传算法在解决组合优化等问题上的优势，采用遗传算法作为本文主要的问题求解方法，用来求解和设计并行与分布式系统下的可分任务调度模型和算法。

中国知网 https://www.cnki.net

中国知网 https://www.cnki.net

第三章 带释放时间的同构系统可分任务调度模型与算法

可分任务调度模型在调度模型的精确性和简单性之间取得了很好地平衡，运用可分任务理论建立的模型既清晰直观又容易求解，因而成为目前任务调度领域的研究热点之一。本章将研究同构并行与分布式平台星型网络拓扑结构下的优化调度问题，找出合理的任务调度方案，使得任务的完成时间最短。

3.1 问题描述

随着信息科学技术和计算机科学的飞速发展，大数据时代已经到来，现在已经进入了一个信息大爆炸的时代，系统对存储、计算速度和带宽的要求也在不断的增加，单一的计算节点已经无法满足很多应用场景下的需求，并行与分布式平台应运而生。如何充分利用并行与分布式平台下的资源，合理高效地对大规模计算任务进行调度和处理逐渐成为了极具挑战性的课题。任务调度问题是并行与分布式平台研究和应用必须解决的一个关键问题。高效的任务调度算法能够根据并行与分布式平台资源和任务的不同情况，在满足一定约束的情况下，达到任务完成时间最短等目标。但是由于并行与分布式平台需要考虑的因素很多，情况比单一节点复杂很多，如何设计出高效的任务调度算法成为当前的一个研究热点。并行与分布式平台下可分任务调度的主要目标是寻求最好的任务调度策略以达到任务的完成时间最短的目标。并行与分布式平台结构各异，有同构和异构，其网络拓扑结构可分为总线型网络、星型网络和树型网络，需要考虑的因素很多，如链路传输输入、处理机计算速度、释放时间和启动开销等等，目前已经有很多基于各种不同并行与分布式平台设计的可分任务调度模型和算法，1.2 节介绍了当前的一些成果。

已有的研究大多假定处理机在任务到来时均处于空闲状态且处理机没有任何启动开销，然而在实际的并行与分布式应用场景中这个假设并非总是成立的。新的任务到来时，处理机可能尚未完成前一个任务，从而处于忙碌状态，不能立即参与新任务的接收和计算；当一台处理机传送任务给另一台处理机时，处理机之间不可能立即响应，因此可能存在通信启动开销；当处理机接受完任务后，也需要一个计算启动开销。本文我们将新任务到来后，处理机由忙碌状态转变为空闲状态的时间间隔称为该处理机的释放时间；处理机由准备接收任务到任务到达的时间间隔称为该处理机的通信启动开销；处理机由接收完任务到开始处理任务的时间间隔称为该处理机的计算启动开销。考虑处理机释放时间和启动开销的可分任务调度的研究尚处于起步阶段，然而实际并行与分布式平台很多处理机都有释放时间和启动开销。文献[42]给出了一种穷举法用于求解同构并行与分布式平台下总线型网络拓扑结构的可分任务调度问题，作者

中国知网 https://www.cnki.net

考虑了处理机中存在释放时间的情况，但是没有考虑计算启动开销和传输启动开销，算法模型简单直观，思路清晰，能够得到最好的调度结果，但是时间复杂度太高且不太贴合实际情况。本章基于同构并行与分布是平台星型拓扑网络结构，吸收了已有成果里面一些调度算法模型简单直观的优点，克服了存在的时间复杂度高，不贴合实际的缺点，建立了一种同时考虑释放时间和启动开销的可分任务调度模型，并提出了高效的遗传算法对可分任务调度模型进行了求解，取得了很好的时间性能。

3.2 模型建立

如图 3.1 所示， $N+1$台处理机通过星型网络拓扑结构互连，其中， P_0为主处理机，用于任务的分配，$\left\{P_i \middle| i \in \{1,2,...,N\}\right\}$为从处理机，用于任务的处理。假定同构并行与分布式系统中所有处理机的计算速率相等。从处理机 P_i的释放时间为r_i，其中$i=1,2,...,N$。l_i是连接 P_0和 P_i的通信链路，所有链路的数据传输速率相同。待分配的任务位于主处理机 P_0上，任务大小记为W_{total}。并不是所有的处理机都必须参与计算，假设参与计算的处理机数目为n。P_0只负责将任务划分为n个子任务$\alpha_1,\alpha_2,...,\alpha_n$，并将这些子任务依次调度到从处理机$P_1,P_2,...P_n$上完成并行计算，其中$\sum_{i=1}^{n}\alpha_i=W_{total}$。主处理机 P_0自身并不参与任务的处理，只负责传输任务到从处理机。系统中链路传输速率和各个处理机的计算速率相同，从处理机的调度顺序遵循释放时间递增的顺序，也即$r_1 \le r_2 \le \cdots \le r_N$。考虑到并行与分布式系统上任务大多比较大，计算复杂，处理所需的时间很长，因此从处理机反馈结果给主处理机的时间可以忽略不计。

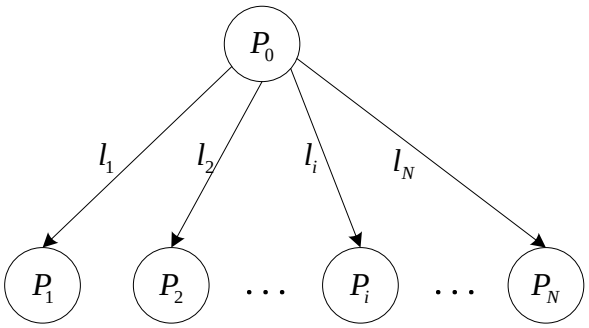

图3.1 同构并行与分布式系统星型网络拓扑结构

根据上面的描述可得，主处理机 P_0传输大小为α_i的子任务给从处理机 P_i所需的时间为$E+z\alpha_i$，其中E为通信启动开销，z为链路传输单位大小任务所花费的时间。处理机 P_i计算子任务 α_i所需时间为$F+w\alpha_i$，其中F为计算启动开销，w为处理机计算单位任务所需的时间。

本章我们记处理机$P_i\left(i=1,2,...,n\right)$开始从主处理机接收任务的时刻为$P_i$的开始时

中国知网 https://www.cnki.net

刻，记为s_i。已知所有处理机的释放时间，下面分三种情况讨论处理机释放时间和开始接收任务时刻可能满足的约束关系。图 3.2、3.3 和 3.4 分别给出了满足约束条件 I、II 和 III 情况下的可分任务调度图。

I. $r_{i+1} \le s_i + z\alpha_i + E, i = 1,2,...,n-1$：该约束左边表示处理机$P_{i+1}$的释放时间，右边表示主处理机$P_0$向从处理机$P_i$传输任务完成时刻，不等式表示从处理机$P_{i+1}$从忙碌状态转为空闲状态的时刻发生在$P_0$给$P_i$传输任务$\alpha_i$的过程中，调度图如 3.2 所示。

文献[15]第一次证明了只有当所有处理机同时完成计算的时候，任务的完成时间最短，否则可以将完成时间长的处理机上的任务拿出一部分分配到完成时间短的处理机上，使得任务的最短完成时间变短，一直这样直到所有处理机上的完成时间相同即可。这一结论广泛的用于可分任务调度模型中，能够大大简化模型并且确保得到最短完成时间。由于所有处理机同时完成计算，可以得到递推公式(3-1)：

$$s_i + E + z\alpha_i + F + w\alpha_i = s_{i+1} + E + z\alpha_{i+1} + F + w\alpha_{i+1}, \quad i = 1,2,...,n-1. \tag{3-1}$$

递推公式(3-1)左边表示处理机P_i的完成时间，即处理机P_i的开始时间s_i，通信启动开销时间E，任务传输时间$z\alpha_i$，即任务传输完成的时刻为$s_i + E + z\alpha_i$，计算启动开销F，处理机计算任务所需的时间$w\alpha_i$之和，即任务处理完成时刻为$s_i + E + z\alpha_i + F + w\alpha_i$；递推公式(3-1)右边表示处理机$P_{i+1}$的完成时间，即处理机$P_{i+1}$的开始时间$s_{i+1}$，通信启动开销时间$E$，任务传输时间$z\alpha_{i+1}$，即任务传输完成的时刻为$s_{i+1} + E + z\alpha_{i+1}$，计算启动开销$F$，处理机计算任务所需的时间$w\alpha_{i+1}$之和，即任务处理完成时刻为$s_{i+1} + E + z\alpha_{i+1} + F + w\alpha_{i+1}$。

由图 3.2 可以看出，处理机P_i的开始时间s_i满足递推公式(3-2)：

$$s_i = s_{i-1} + E + z\alpha_{i-1}, \quad i = 2,3,...,n. \tag{3-2}$$

递推公式(3-2)右表示处理机P_i的开始时间等于处理机P_{i-1}的传输任务结束时刻，由于$s_1 = r_1$，递推可以得到s_i关于r_1的递推公式(3-3)：

$$\begin{cases} s_1 = r_1 \\ s_i = s_{i-1} + z\alpha_{i-1} + E \\ s_n = s_{n-1} + z\alpha_{n-1} + E \end{cases} \Rightarrow s_i = r_1 + (i-1)E + z\left(\sum_{j=1}^{i-1}\alpha_j\right), \quad i = 2,3,...,n \tag{3-3}$$

中国知网 https://www.cnki.net

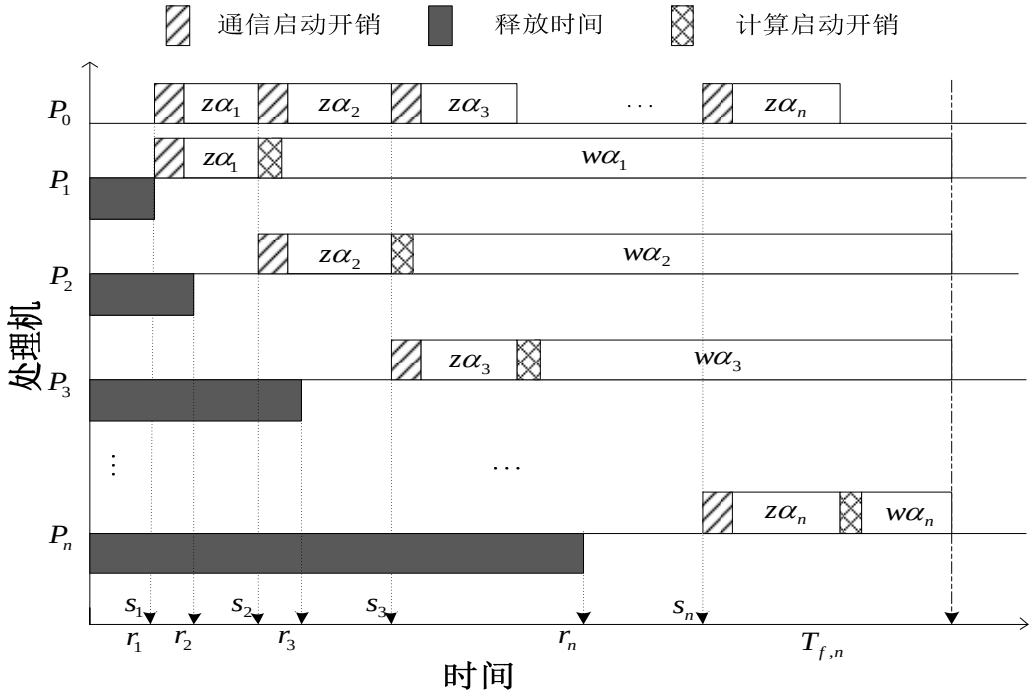

图3.2 一种满足约束条件 I 的同构系统可分任务调度图

记 $\beta_i = \alpha_i + E/\mathrm{z}$，结合递推公式(3-1)和递推公式(3-2)可得：

$$\begin{cases} s_i + E + z\alpha_i + F + w\alpha_i = s_{i-1} + E + z\alpha_{i-1} + F + w\alpha_{i-1} \\ \qquad \Rightarrow\ s_i - s_{i-1} = (\mathrm{z}+\mathrm{w})\ (\alpha_{i-1} - \alpha_i)\ = (\mathrm{z}+\mathrm{w})\ (\beta_{i-1} - \beta_i) \\ s_i = s_{i-1} + E + z\alpha_{i-1} \quad \Rightarrow\ \ s_i - s_{i-1} = E + z\alpha_{i-1} \\ \beta_i = \alpha_i + \dfrac{E}{z} \end{cases} \tag{3-4}$$

$$s_i - s_{i-1} = E + z\alpha_{i-1} = z\beta_i = (\mathrm{z}+\mathrm{w})\ (\beta_{i-1} - \beta_i)$$

由递推公式(3-4)可得出：

$$\beta_i = q\beta_{i-1} = q^{i-1}\beta_1, \quad q = \frac{w}{z+w}, \quad i = 2,3,...,n. \tag{3-5}$$

已知 $\sum_{i=1}^{n}\alpha_i = W_{total}$，则 $\sum_{i=1}^{n}\beta_i = W_{total} + En/z$，代入递推公式(3-5)可得：

$$\beta_1 = \frac{W_{total} + E\dfrac{n}{z}}{\sum_{i=1}^{n} q^{i-1}} = \frac{1-q}{1-q^n}\left(W_{total} + \frac{En}{z}\right), \quad q = \frac{w}{z+w}. \tag{3-6}$$

中国知网 https://www.cnki.net

$$\alpha_1=\beta_1-\frac{E}{z}=\frac{1-q}{1-q^n}W_{total}+\frac{n-nq-1+q^n}{1-q^n}\times\frac{E}{z},\quad q=\frac{w}{z+w}. \tag{3-7}$$

由图 3.2 可以看出任务的完成时间为：

$$\begin{aligned}T&=r_1+(z+w)\alpha_1+E+F=r_1+E+F+(z+w)\\&\times(\frac{1-q}{1-q^n}W_{total}+\frac{n-nq-1+q^n}{1-q^n}\times\frac{E}{z}),\quad q=\frac{w}{z+w}.\end{aligned} \tag{3-8}$$

根据递推公式(3-8)可以得出，在约束条件 I 下，任务的完成时间跟处理机的释放时间有关系，但是只受从处理机 P_1 的释放时间 r_1 影响，而不受其他处理机释放时间的影响。

II. $r_{i+1}>s_i+z\alpha_i+E, i=1,2,...,n-1$：该约束说明主处理机 P_0 给从处理机 P_i 传输完任务 α_i 的时刻 $s_i+z\alpha_i+E$ 早于从处理机 P_{i+1} 的释放时间 r_{i+1}，即 P_0 在给 P_i 传输完任务 α_i 后需要等待一段时间直到 P_{i+1} 结束忙碌状态才能为其传输任务 α_{i+1}，调度图如 3.3 所示。

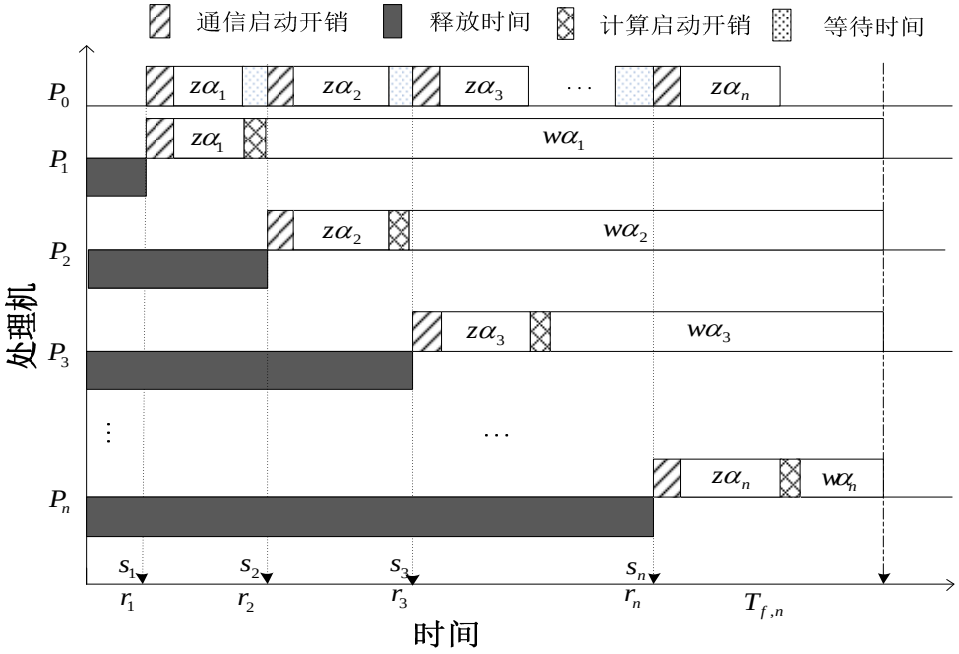

图3.3 一种满足约束条件 II 的同构系统可分任务调度图

同样地，根据所有参与计算处理机同时完成计算可以得到递推公式(3-1)。由图 3.3 可以看出，处理机 $P_i\left(i=1,2,...,n\right)$ 的释放时间 r_i 和开始时间 s_i 是相同的，即 $r_i=s_i$。将递推公式(3-1)中的 s_i 替换为 r_i 可以得到：

中国知网 https://www.cnki.net

$$\begin{cases} r_2 + E + z\alpha_2 + F + w\alpha_2 = r_1 + E + z\alpha_1 + F + w\alpha_1 \\ r_i + E + z\alpha_i + F + w\alpha_i = r_{i-1} + E + z\alpha_{i-1} + F + w\alpha_{i-1} \\ r_n + E + z\alpha_n + F + w\alpha_n = r_{n-1} + E + z\alpha_{n-1} + F + w\alpha_{n-1} \end{cases} \Rightarrow r_i - r_1 = (z+w)(\alpha_i - \alpha_1) \quad (3\text{-}9)$$

化简递推公式(3-9)可得：

$$\alpha_i = \alpha_1 + \frac{r_1 - r_i}{z+w},\ i = 2,3,...,n. \quad (3\text{-}10)$$

已知 $\sum_{i=1}^{n} \alpha_i = W_{total}$，代入递推公式(3-10)可得：

$$\alpha_1 = \frac{1}{n}\left(W_{total} - \frac{(n-1)r_1 - \sum_{i=2}^{n} r_i}{z+w} \right) = \frac{1}{n}\left(W_{total} + \frac{\sum_{i=1}^{n} r_i - nr_1}{z+w} \right) \quad (3\text{-}11)$$

由图 3.3 可以看出任务的完成时间为：

$$T_{f,n} = r_1 + (z+w)\alpha_1 = r_1 + \frac{1}{n}(z+w)W_{total} + \frac{1}{n}\left(\sum_{i=1}^{n} r_i - nr_1 \right) \quad (3\text{-}12)$$

由递推公式(3-12)可以看出，在约束条件 II 下，任务的完成时间和处理机释放时间有关，且受到所有参与计算的处理机的释放时间的影响。

III. 混合时序约束：在实际的并行与分布式平台中，任意两个相邻的从处理机之间可能满足约束条件 I，也可能满足条件 II。假设有 n 台处理机参与计算，所有可能的情况有 2^{n-1} 种。图 3.4 给出了一种可能的混合时序约束的可分任务调度图。

我们将处理机 P_{i-1} 和 P_i 满足约束条件 I 和约束条件 II 的情况分别用 0 和 1 表示，其中 $i = 2,3,...,n$ 。用 $C = (c_2, c_3, ..., c_n)$ 表示一种混合时序约束条件，其中 $c_i \in \{0,1\}$，$i = 2,3,...,n$。若 $c_i = 1$，表明主处理机 P_0 在给 P_{i-1} 传输完数据后紧接着为 P_i 传输数据，中间没有空闲。此时，处理机 P_i 的开始时间为 $s_i = s_{i-1} + z\alpha_{i-1} + E$ 。若 $c_i = 0$，表明主处理机 P_0 在给 P_{i-1} 传输完数据后需要等待 P_i 由忙碌转为空闲状态才能开始传输数据，中间存在空闲时间。此时，处理机 P_i 的开始时间为 $s_i = r_i$。图 3.4 给出了满足混合时序约束条件 C 的可分任务调度图，其中 $C = (1,...,1,0,0,...,0,1,...,1)$。

中国知网 https://www.cnki.net

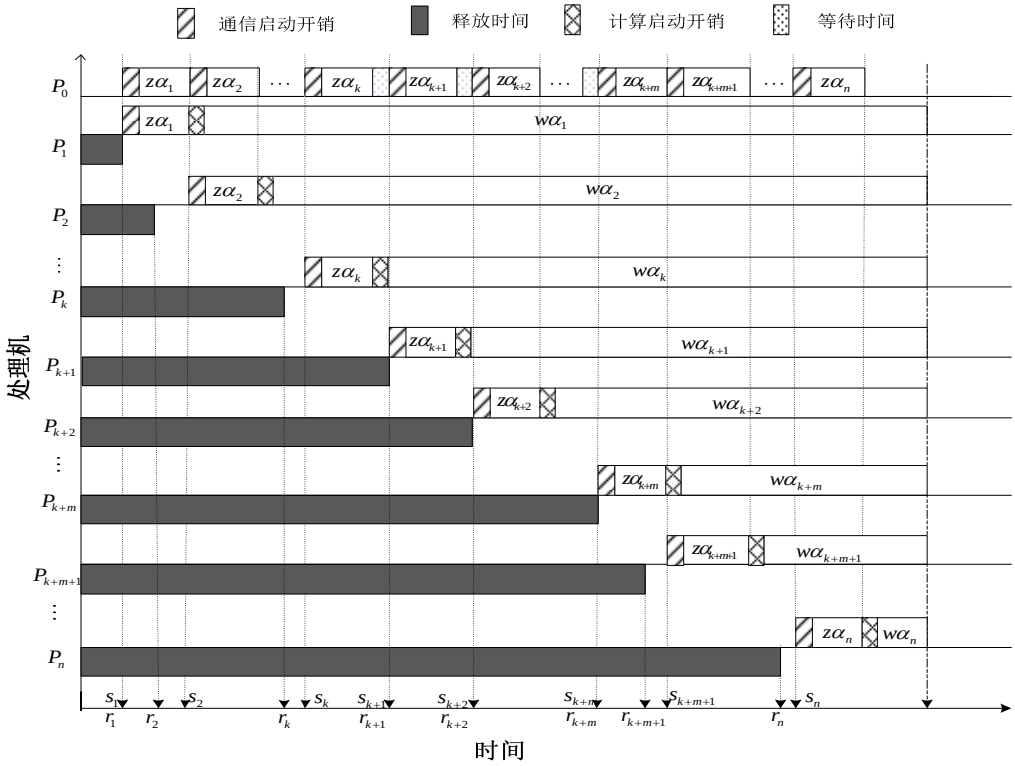

图3.4 一种可能的混合时序约束的同构系统可分任务调度图

根据混合时序约束条件C，可以得到处理机开始时间之间的递推公式(3-13)。由于所有处理机同时完成计算，此时完成时间最短，可以得到递推公式(3-1)。将递推公式(3-13)带入递推公式(3-1)可得β_i如递推公式(3-14)所示。

$$\begin{cases} s_1 = r_1 \\ s_2 = s_1 + z\alpha_1 + E = s_1 + z\beta_1 \\ \vdots \\ s_k = s_{k-1} + z\alpha_{k-1} + E = s_{k-1} + z\beta_{k-1} \\ s_{k+1} = r_{k+1} \\ s_{k+2} = r_{k+2} \\ \vdots \\ s_{k+m} = r_{k+m} \\ s_{k+m+1} = s_{k+m} + z\alpha_{k+m} + E = s_{k+m} + z\beta_{k+m} \\ \vdots \\ s_n = s_{n-1} + z\alpha_{n-1} + E = s_{n-1} + z\beta_{n-1} \end{cases} \tag{3-13}$$

中国知网 https://www.cnki.net

$$\begin{cases} \beta_2 = q\beta_1 \\ \vdots \\ \beta_k = q\beta_{k-1} \\ \beta_{k+1} = \beta_1 + (r_1 - r_{k+1})/(z+w) \\ \beta_{k+2} = \beta_1 + (r_1 - r_{k+2})/(z+w) \\ \vdots \\ \beta_{k+m} = \beta_1 + (r_1 - r_{k+m})/(z+w) \\ \beta_{k+m+1} = q\beta_{k+m} \\ \vdots \\ \beta_n = q\beta_{n-1} \end{cases} \tag{3-14}$$

我们将递推公式(3-14)中的$n-1$个等式同$\sum_{i=1}^{n}\beta_i = W_{total} + nE/z$共$n$个等式表示成如下标准形式：

$$A \cdot \beta = b \tag{3-15}$$

其中，β表示的是待求的$n\times 1$维的解变量$(\beta_1, \beta_2, ..., \beta_n)$，通过$\beta$即可求出每个处理机需要处理的任务$\alpha$，$A$是$n\times n$的系数矩阵，$b$是$n\times 1$维的向量，在图 3.4 给出的任务调度图中，$A$和$b$可以分别表示为：

$$A = \begin{bmatrix} -q & 1 & 0 & \cdots & & & & & & & 0 \\ \vdots & \ddots & \ddots & \ddots & \cdots & & & & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \cdots & & & & & \vdots \\ 0 & \cdots & 0 & -q & 1 & 0 & \cdots & & & & 0 \\ -1 & 0 & \cdots & & 0 & 1 & 0 & \cdots & & & 0 \\ -1 & 0 & & \cdots & & 0 & 1 & 0 & \cdots & & 0 \\ \vdots & \vdots & & & \cdots & & \ddots & \ddots & & \cdots & \vdots \\ -1 & 0 & & & & \cdots & & 1 & 0 & \cdots & 0 \\ 0 & 0 & & & & & \cdots & -q & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & & & & & & & \cdots & 0 & -q & 1 & 0 \\ 0 & 0 & & \cdots & 0 & 0 & \cdots & 0 & \cdots & 0 & -q & 1 \\ 1 & 1 & & & \cdots & 1 & 1 & & & \cdots & 1 & 1 \end{bmatrix}$$

$$b = \left(0, \quad \cdots, \quad 0, \quad \frac{r_1 - r_{k+1}}{z+w}, \quad \frac{r_1 - r_{k+2}}{z+w}, \quad \cdots, \quad \frac{r_1 - r_{k+m}}{z+w}, \quad 0, \quad \cdots, \quad 0, \quad W_{total} + \frac{nE}{z} \right)^T$$

给定一种混合时序约束条件$C = (c_2, c_3, ..., c_n)$，就可以求得一个形如递推公式(3-15)的标准式，然后可以采用线性规划方法求出β的解，进而由关系式$\beta = \alpha + E/z$求得任务分配方案α的解。下面我们给出混合时序约束的可分任务调度模型：

$$\min_{n,C}(\mathrm{T}) = \min\left(r_1 + (z+w)\alpha_1 + E + F\right)$$

中国知网 https://www.cnki.net

s.t.

1. $0 < n \le N$，其中 N 为处理机的总数，n 为参与计算的处理机数量。
2. $0 < \alpha_i \le W_{total}$，$\frac{E}{z} < \beta_i = \alpha_i + \frac{E}{z} \le W_{total} + \frac{E}{z}, i = 1,2,...,n$
3. $\sum_{i=1}^{n} \alpha_i = W_{total}$，$\sum_{i=1}^{n} \beta_i = W_{total} + \frac{nE}{z}$
4. $s_i + E + z\alpha_i + F + w\alpha_i = s_{i+1} + E + z\alpha_{i+1} + F + w\alpha_{i+1}$，$i = 1,2,...,n-1$
5. $C = (c_2, c_3, ..., c_n)$，$c_i \in \{0,1\}$，$i = 2,3,...,n$
6. $\begin{cases} r_{i+1} \le s_i + z\alpha_i + E, & if\ (c_i = 1), \\ r_{i+1} > s_i + z\alpha_i + E, & else\ (c_i = 0). \end{cases}$，$i = 2,3,...,n$
7. $s_i = \begin{cases} s_{i-1} + z\alpha_{i-1} + E, & if\ (c_i = 1), \\ r_i, & else\ (c_i = 0). \end{cases}$，$i = 2,3,...,n$

可分任务调度模型的目标是任务的完成时间最短。模型的约束 1 表示并不一定要求所有的处理都参与计算，即参与计算的处理机数目可以小于 N；约束 2 表示每台参与计算的处理机分配的任务量非负，且不能超过总任务量 W_{total}；约束 3 表示所有处理机分配的任务量之和为总任务量 W_{total}，相应的可以得出 β_i 之和与 W_{total} 的关系；约束 4 表示所有处理机必须同时完成任务，即 $s_i + E + z\alpha_i + F + w\alpha_i, i = 1,2,...,n$ 的值均相等；约束 5 限定了混合时序约束条件 C 的取值范围；约束 6 表示了约束条件 C 取两种情况下处理机释放时间 r_i 和前一个处理机传输结束时间的关系；约束 7 给出了约束条件 C 在两种情况下处理机 P_i 开始时间 s_i 的计算方法。

本小节我们对同构并行与分布式平台星型拓扑网络结构下混合时序约束的可分任务调度问题进行了详细的讨论和分析，并建立了以任务完成时间最短为目标的带约束的单目标优化模型，下一节我们将根据上面建立的优化模型，设计遗传算法来求解可分任务调度模型。

3.3 可分任务调度算法

遗传算法作为一种被广泛使用的启发式随机搜索算法，在组合优化问题的求解上有着很好的表现。通过上一小节的分析，最后建立的可分任务调度模型最终可以归结为组合优化问题，因此我们精心设计了编码方案和其对应的选择，交叉，变异等遗传算子。通过实验分析可以看出，遗传算法在解决可分任务调度模型方面也有着很好的表现，不仅可以求出问题的最优解，而且具有良好的时间性能。

3.3.1 编码

编码是连接问题和算法的桥梁。遗传算法在求解之前，首先要选择合适的编码方

中国知网 https://www.cnki.net

式，将问题的所有参变量编码成对应的子串，再将各个子串首尾联接形成一定长度的串，即染色体，一个串表示解空间的一个解。编码方案的设计是遗传算法设计中很重要的一环，编码的好坏直接影响对问题的求解精度和算法收敛速度。

针对上面建立好的任务调度模型，我们采用实数编码方式，将该混合时序约束的可分任务调度问题表示成一个向量$I=(n,C)$，其中n表示参与计算的处理机数目，种群初始化阶段假设所有的处理机均参与计算，即参与计算的处理机数目为N，$C=(c_2,c_3,...,c_n)$表示一种混合时序约束条件，设定$c_i\in\{1,0\},i=2,3...,n$，若$c_i=1$，则表示处理机对P_{i-1}和P_i满足约束条件 I，即$r_{i+1}\le s_i+z\alpha_i+E$；反之若$c_i=0$，则表示处理机对$P_{i-1}$和$P_i$满足约束条件 II，即$r_{i+1}>s_i+z\alpha_i+E$。例如，假设共有 7 台处理机，其中一台处理机是主处理机，用于任务分配，剩下 6 台从处理机进行任务处理，即$N=6$，则一种可能的编码如表 3.1 所示：

表3.1 含 6 台从处理机同构系统的编码示例

I	n	c_2	c_3	c_4	c_5	c_6
编码	6	1	0	1	1	0

其中，第一位$n=6$，表示参与计算的处理机数目，初始阶段设定为处理机总数；第二位$c_2=1$，表示第 1 台处理机和第 2 台处理机之间满足约束 I；第三位$c_3=0$表示第 2 台处理机和第 3 台处理机之间满足约束 II；第四位$c_4=1$表示第 3 台处理机和第 4 台处理机之间满足约束 I；第五位$c_5=1$表示第 4 台处理机和第 5 台处理机之间满足约束 I；第六位$c_6=0$表示第 5 台处理机和第 6 台处理机之间满足约束 II。

对种群中的任意一个个体，如果给定个体I，则可以知道参与计算的处理机数目n和混合时序约束C。根据n和C就可以知道系数矩阵A和对应的向量b，最终将问题转化为递推公式(3-15)的标准形式。通过线性规划方法求解该标准式就可以得到β，进而得到任务分配方案α的解。验证求解出的α是否满足模型所有的约束条件，尤其是约束 2 和约束 5。如果个体I满足模型的所有约束条件，则个体I对应唯一一个类似于图 3.4 的混合时序可分任务调度图，调度方案α即为可行解，将该方案对应的任务完成时间$T=r_1+(z+w)\alpha_1+E+F$作为个体I的适应度值。如果α不满足模型的部分约束条件，则表明并不需要这么多的处理机参与计算，令$n=n-1$，更新个体I，重复这个过程，直至求出一个满足模型全部约束条件的解。

3.3.2 交叉和变异

交叉操作在遗传算法中起核心作用。交叉操作保证了遗传算法种群个体的多样性和全局搜索能力。为了使种群个体更加多样，本章采用均匀交叉和单点交叉这两种交叉方式生成新个体，即对种群中的前一半个体采用均匀交叉方式进行交叉，对种群中

中国知网 https://www.cnki.net

后一半个体采用单点交叉方式进行交叉。

均匀交叉对基因中的每一位，随机产生一个 0~1 之间的浮点数，如果大于 0.5，则将第二个父代个体对应的值赋值给第一个子代个体，第一个父带个体对应的值赋值给第二个子代个体；如果该随机数小于等于 0.5，则将第一个父代个体对应的值赋值给第一个子代个体，第二个父带个体对应的值赋值给第二个子代个体。由于个体第 1 位表示参与计算的处理机数目，因此交叉后的后代个体第一位都置为处理机总数 N 。

例如，有两父代个体（假设系统中处理机总数目为 6）：

$$P_1 = (6\,1\,0\,1\,1\,0)$$

$$P_2 = (5\,0\,1\,0\,0\,1)$$

假设产生的 5 个 0~1 之间的浮点数分别为 0.4，0.5， 0.6， 0.3 以及 0.9，则产生的两个后代为：

$$O_1 = (6\,1\,0\,0\,1\,1)$$

$$O_2 = (6\,0\,1\,1\,0\,0)$$

单点交叉随机产生一个整数 p ，满足 $2 \le p < N$ 作为交叉点。然后交换第一个父代和第二个父代基因段中 $p \sim N$ 范围内的基因，生成两个子代个体。由于个体第 1 位表示参与计算的处理机数目，因此交叉后的后代个体第一位都置为处理机总数 N 。

例如，有两父代个体（假设系统中从处理机数目为 6）：

$$P_1 = (6\,1\,0\,0\,1\,0)$$

$$P_2 = (5\,0\,1\,1\,0\,1)$$

假设单点交叉随机产生的整数为 $p = 4$ ，则产生的两个后代为：

$$O_1 = (6\,1\,0\,0\,0\,1)$$

$$O_2 = (5\,0\,1\,1\,1\,0)$$

变异是推动遗传算法进化的重要手段，通过一定概率的基因变异，增加种群多样性，强化了遗传算法的局部搜索能力。为了使得变异后产生的种群更加丰富多彩，本章采用均匀变异和对换变异两种方式生成新个体。

均匀变异对父代基因的每一位，产生一个 0~1 之间的随机数，若大于阈值，则不进行变异，若小于阈值，则将该位上的数字取反；同时，将后代个体的第一位置的值置为处理机总数 N 。

例如，有一个父代个体（假设系统中从处理机数目为 6）：

$$P = (6\,1\,0\,1\,1\,0)$$

假设阈值为 0.1，若产生的 5 个 0~1 之间的浮点数分别为 0.05，0.5， 0.07， 0.3 以及 0.9，则产生的后代为：

$$O = (6\,0\,0\,0\,1\,0)$$

对换变异随机生成两个整数 p 和 q 满足 $2 \le p < q \le N$ 作为变换点，交换 p 和 q 上的值，即可得到变异对换变异后的个体。

中国知网 https://www.cnki.net

例如，有一个父代个体（假设系统中从处理机数目为 6）：

$$P=(611010)$$

假设产生的随机数 p 和 q 分别为 4 和 5，则变异后的个体为：

$$O=(611100)$$

3.3.3 选择算子

本章采用的是精英保留和联赛选择方法相结合的选择算子。精英保留是对后代个体按照适应度值进行排序，将种群中某个指定数目（通常取种群规模的 5%左右）的最优个体直接保留到下一代，这样可以避免进化出来的最好的个体被抛弃。剩余的后代个体采用联赛（Tournament Selection）选择的方法选出。联赛选择法是最简单也是最实用的选择方法之一。在该方法中，从种群中任选一定数目称之为联赛规模的个体，这里我们设定联赛规模为 2。算法步骤如下：

第一步：每次从种群中选择两个个体；

第二步：选出这两个个体中适应度值更大的个体；

第三步：重复前两步骤，直到选择的个体总数达到种群规模或预期的目标需求则停止。

例如，种群中有 10 个个体，适应度值分别为[1, 3, 2, 6, 5, 7, 8, 4, 9, 10]，适应度值越大表示个体越好。先对种群进行精英保留，保留这 10 个个体中最好的 1 个个体，加入新的种群中，然后在进行联赛选择。假设从这 10 个个体中选出的两个适应度值分别为 3 和 2，则将适应度值为 3 的个体放入新种群里面，一直进行 9 次，即可得到整个新种群。

3.3.4 算法框架

基于前面设计的编码以及交叉、变异、选择等遗传算子，我们给出求解混合时序约束的可分任务调度遗传算法框架，如算法 3.1 所示：

算法 3.1 同构系统下混合时序约束可分任务调度遗传算法

1:（**初始化**） 确定种群大小 $PopSize$，交叉概率 p_{cros} 和变异概率 p_{mut} 等参数，根据编码规则随机生成初始种群 $P(0)$，令进化代数 $t=0$。

2:（**交叉**） 以概率 p_{cros} 从 $P(t)$ 之中选择父代个体进行交叉，交叉获得的全部后代个体定义为集合 O_1。

3:（**变异**） 以概率 p_{mut} 从集合 O_1 中选择个体进行变异，新的后代个体定义为集合 O_2。

4:（**选择**） 从集合 $P(t)\cup O_1\cup O_2$ 中选择最优的 EL 个个体直接保留到下一代种群以加快算法的收敛速度；使用联赛选择操作从集合 $P(t)\cup O_1\cup O_2$ 选择 $N-EL$ 个个体

中国知网 https://www.cnki.net

保留到下一代种群 $P(t+1)$ 中；令 $t=t+1$。

5:（**终止条件**） 如果满足终止条件，则终止算法；否则转向步骤 2。

3.4 实验仿真与结果分析

上一节我们详细介绍了混合时序约束可分任务调度遗传算法的设计过程，本节我们将通过实验仿真对上述可分任务调度模型和算法进行详细分析。

通过 3.2 节的分析可知，任务调度算法的任务完成时间和处理机的释放时间有着直接的联系，表 3.2 给出四组处理机释放时间，用于作为下面考察处理机释放时间对任务调度时间的影响。

表3.2 随机生成的不同处理机释放时间参数

r_{avg}	2	4	6	8
r_1	0.18	0.17	0.76	0.16
r_2	0.28	0.22	1.18	0.63
r_3	0.31	0.40	1.48	0.79
r_4	0.94	0.57	1.87	1.35
r_5	1.09	0.99	2.25	3.02
r_6	1.19	1.23	2.59	4.58
r_7	1.33	2.21	2.60	5.64
r_8	1.68	2.84	2.81	6.74
r_9	1.82	3.49	3.61	7.15
r_{10}	2.09	3.62	4.77	9.34
r_{11}	2.66	3.85	5.99	9.59
r_{12}	2.85	3.98	6.24	10.83
r_{13}	2.85	4.76	7.29	11.74
r_{14}	3.38	5.34	8.69	11.83
r_{15}	3.40	5.68	9.99	12.90
r_{16}	3.85	7.10	10.17	13.57
r_{17}	3.86	7.78	10.66	14.08
r_{18}	3.98	8.35	11.07	15.50
r_{19}	4.18	9.26	12.87	16.14
r_{20}	5.15	10.01	14.03	17.57

同构并行与分布式系统星型网络拓扑结构下可分任务调度模型参数设置如下：处

中国知网 https://www.cnki.net

理机总数 $N=20$， $z=0.75$， $w=1.0$， $E=0.005$， $F=0.002$。处理机的释放时间详见表 3.2，其中 $r_1 \sim r_{20}$ 分别对应处理机 $P_1 \sim P_{20}$ 的释放时间，四组不同的处理机释放时间 $r_1 \sim r_{20}$ 分别符合均值 *avg* 为 2、4、6 和 8 的指数分布的随机数。另外，在遗传算法中采用如下参数：种群大小 $PopSize=100$，交叉概率 p_{cros}=0.6，变异概率 $p_{mut}=0.02$，精英保留个数 $EL=5$，终止条件为进化代数 $t=100$。

取 r_{avg}=2，表 3.3 给出了不同任务量情况下（$W_{total}=1.0 \sim 10.0$）两种算法对比的实验结果，其中，GA 代表本文提出的全局优化遗传算法，ExA 代表文献[42]提出的穷举算法(Exhaustive Algorithm)。

表3.3 不同任务量情况下两种算法对比实验结果

任务量	运行结果	GA	ExA
1.0	参与计算处理机数目	4	4
	任务完成时间	1.05756	1.05756
	算法运行时间（秒）	0.996	76.13
2.0	参与计算处理机数目	8	8
	任务完成时间	1.73304	1.73304
	算法运行时间（秒）	0.883	75.32
3.0	参与计算处理机数目	10	10
	任务完成时间	2.4839	2.4839
	算法运行时间（秒）	0.842	75.08
4.0	参与计算处理机数目	10	10
	任务完成时间	3.2367	3.2367
	算法运行时间（秒）	0.842	75.21
5.0	参与计算处理机数目	11	11
	任务完成时间	3.98842	3.98842
	算法运行时间（秒）	0.817	75.21
6.0	参与计算处理机数目	11	11
	任务完成时间	4.74002	4.74002
	算法运行时间（秒）	0.840	75.18
7.0	参与计算处理机数目	11	11
	任务完成时间	5.49161	5.49161
	算法运行时间（秒）	0.834	75.36
8.0	参与计算处理机数目	12	12
	任务完成时间	6.24269	6.24269

中国知网 https://www.cnki.net

	算法运行时间（秒）	0.811	75.58
9.0	参与计算处理机数目	12	12
	任务完成时间	6. 9936	6. 9936
	算法运行时间（秒）	0.822	75.47
10.0	参与计算处理机数目	12	12
	任务完成时间	7.74451	7.74451
	算法运行时间（秒）	0.814	78.55

从不同任务量情况下两种算法对比实验结果可以看出，两个算法任务量相同，运用遗传算法和穷举法能够得到相同的调度方案，即参与计算处理机数目相同，且任务完成时间相同，然而算法的运行时间相差很大。本章提出的 GA 算法的运行时间远小于穷举法所需要的时间，从表 3.3 可知遗传算法所需的时间不到一秒，而 ExA 算法需要的时间超过一分钟，由此可知本章提出的 GA 算法具有很好的时间性能。

下面我们分两种情况考察任务调度算法的任务完成时间和处理机的释放时间之间的联系。图 3.5 和图 3.6 分别表示的是在任务量较小（$W_{total}=1.0\sim10.0$）的情况下参与计算的处理机数目和任务最短完成时间随任务大小和处理机平均释放时间的变化规律和趋势。图 3.7 和图 3.8 分别表示的是在任务量较大（$W_{total}=20.0\sim100.0$）的情况下参与计算的处理机数目和任务最短完成时间随任务大小和处理机平均释放时间的变化规律和趋势。

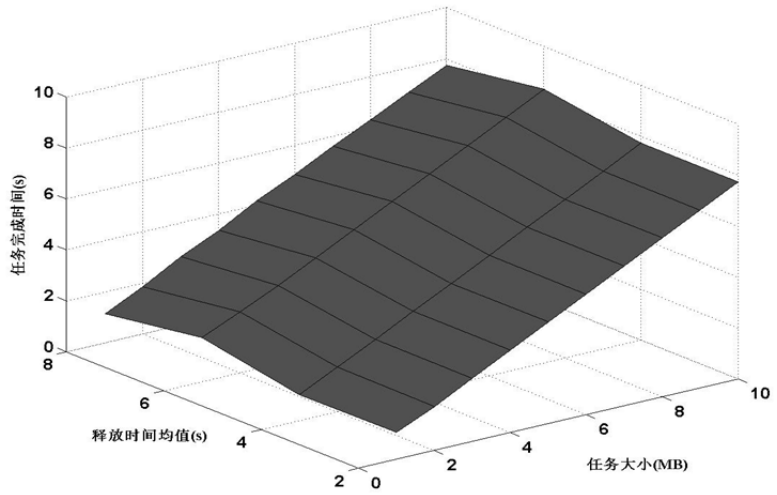

图3.5 在任务量较小时任务的最短完成时间的变化趋势

根据图 3.5 的曲线可以看出，在任务量较小的情况下，随着任务大小的不断增大，

中国知网 https://www.cnki.net

任务完成时间也不断增大，两者近似成线性的变化趋势；而随着释放时间均值的增大，任务完成时间不一定增大，两者没有明确的关系，因为任务完成时间和参与计算的处理机的释放时间有很大的关系，然而释放时间均值大，并不意味着参与计算的处理机的释放时间大，而且任务量小的时候，参与计算的处理机数目较小，此时前面处理机释放时间与结果有很大的关系。

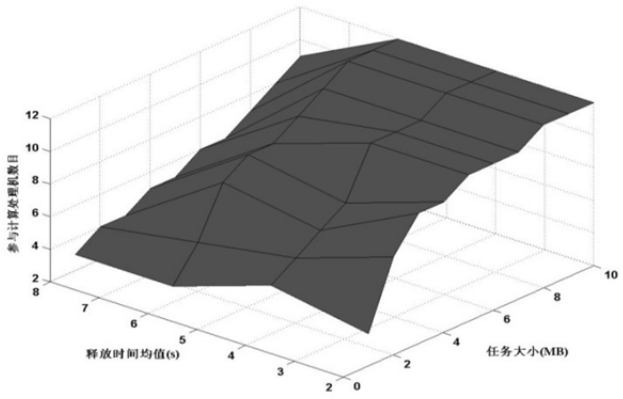

图3.6 在任务量较小时参与计算处理机数目的变化趋势

根据图3.6的曲线可以看出，在任务量较小的情况下，随着任务大小的不断增大，参与计算处理机数目也不断增大；而随着释放时间均值的增大，参与计算处理机数目不一定增大，两者没有明确的关系。因为参与处理机的数目和释放时间有关系，释放时间均值大，并不意味着参与计算的处理机的释放时间大，而且任务量小的时候，参与计算的处理机数目较小，此时前面处理机释放时间与结果有很大的关系。

通过上面的分析可以得出，当任务量较小的时候，处理机的释放时间会在很大程度上影响任务的最短完成时间和参与计算的处理机数目。

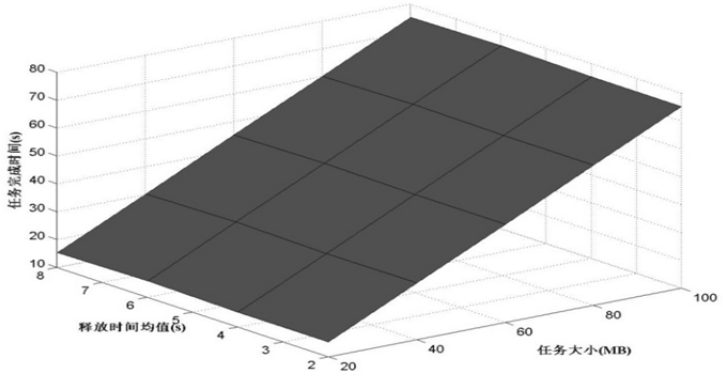

图3.7 在任务量较大时任务的最短完成时间的变化趋势

中国知网 https://www.cnki.net

根据图 3.7 的曲线可以看出，在任务量较大的情况下，随着任务大小的不断增大，任务完成时间也不断增大，两者近似成线性的变化趋势；而随着释放时间均值的增大，任务完成时间不一定增大，两者没有明确的关系，只与释放时间有关系。

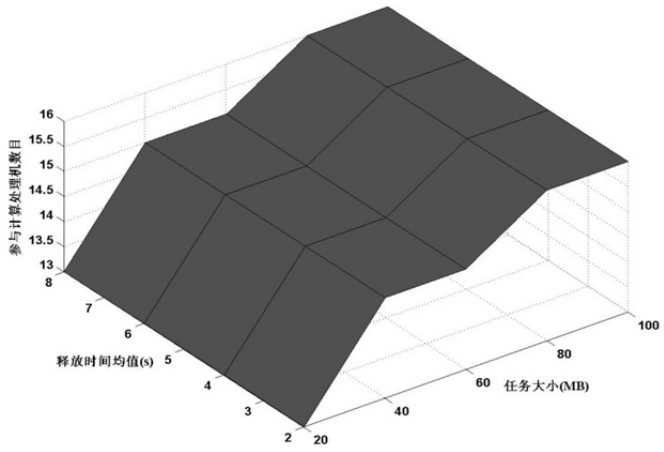

图3.8 在任务量较大时参与计算的处理机数目的变化趋势

根据图 3.8 的曲线可以看出，在任务量较大的情况下，随着任务大小的不断增大，参与计算处理机数目也不断增大；而随着释放时间均值的增大，处理机的释放时间对任务最短完成时间的影响几乎可以忽略。这主要是因为模型采用的是阻塞通信模式，后分配任务的处理机需要等待先分配任务的处理机完成数据的传输后才能开始接收任务，当任务量足够大的时候，等待时间已经超过了处理机的释放时间，所以释放时间对任务的最短完成时间不再构成影响。

3.5 本章小结

本章我们建立了更接近于实际并行与分布式平台下的可分任务调度模型，重点考虑了存在释放时间和启动开销的情况，详细分析了同构并行与分布式平台星型网络拓扑结构下的三种不同时序约束条件下的可分任务调度过程，进而提出了一种新的混合时序约束的可分任务调度模型，并设计了高效的全局遗传算法对其进行求解，最后通过实验证明了任务调度模型的正确性和任务调度算法的高效性。

中国知网 https://www.cnki.net

中国知网 https://www.cnki.net

第四章 带释放时间的异构系统可分任务调度模型与算法

在上一章中，我们研究了同构并行与分布式平台星型网络拓扑结构下，考虑释放时间和启动开销的可分任务调度模型，并设计了高效的遗传算法进行模型的求解。然后，在实际生活中，异构并行与分布式平台也大量存在。基于此，我们提出了异构并行与分布式平台下，考虑释放时间和启动开销的混合约束可分任务调度模型，并设计了遗传算法对模型进行求解。

4.1 问题描述

异构并行与分布式平台下的可分任务调度问题今年来也得到了广泛的关注和研究，取得了众多的研究成果。虽然同构平台可以看做是异构平台的特殊情况，但是调度算法也存在很多的不同。异构并行与分布式平台下处理机之间各不相同，计算速度，链路传输速度和启动开销等值大小各异，传输任务时不能单纯的按照某一个顺序进行传输，故在建立模型的时候，相比同构平台，还需要考虑处理机的调度顺序，这样问题的求解空间将大大增大。文献[2]已经证明了异构平台下多数问题属于 NP-hard 难题，因此当前大多数研究都是围绕启发式调度算法展开[52-55]。文献[52]提出了自适应索引可分负载任务调度算法来帮助提升系统设计；文献[54]基于异构并行与分布式平台，重点考虑了可分任务调度中任务在异构平台上的分配问题，设计了静态和动态两种启发式算法对问题进行求解;文献[55]基于异构并行与分布式，除了考虑 CPU 和存储外，还重点考虑了带宽这一资源，并设计了带宽感知智能算法进行任务调度，取得了很好的结果。文献[56]详细讨论了博弈论在任务调度中的应用。文献[57]基于异构并行与分布式平台星型网络和树型网络拓扑结构，考虑了处理机的计算启动开销和传输启动开销，详细分析推导了单趟和多趟可分任务调度模型的建立和求解。文献[58]基于异构并行与分布式平台，在满足任务约束的情况下，以最短完成时间为目标，重点讨论了处理机上任务的分配问题。

任务在异构并行与分布式平台中有以下的特征：

1. 由于处理机的计算速度各异，同一任务在不同的处理机上有着不一样的计算时间；
2. 不同的任务在同一处理机上所需计算时间不相等；
3. 由于主处理机连接从处理机的链路传输速度各异，故同一任务传输到不同的从处理机上所需的时间不相等。

异构并行与分布式平台下的可分任务调度模型需要重点考虑三个方面：

中国知网 https://www.cnki.net

1. 处理机选择问题：处理机是否需要全部参与运算，如果不需要，应该选择多少台处理机参与运算，选哪些参与任务的处理；
2. 处理机调度顺序问题：在选好了处理机的情况下，以什么样的顺序调度这些处理机是另一个需要着重考虑的问题；
3. 每台处理机分配任务大小：处理机顺序确定之后，为每台处理机分配多大的任务，使得完成时间最短。

本章在上一章的基础上，继续研究异构星型网络系统下，考虑处理机存在释放时间和启动开销的可分任务调度模型与算法。

4.2 模型建立

本节我们基于异构并行与分布式平台星型网络拓扑结构进行建模，如图 4.1 所示：

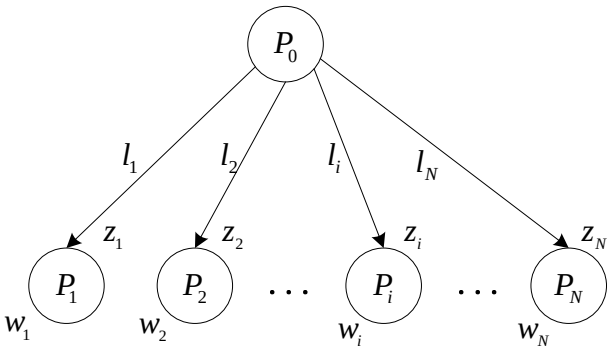

图4.1 异构并行与分布式系统星型网络拓扑结构

假设该异构并行与分布式平台由 $N+1$ 台处理机组成，处理机之间通过星型网络拓扑结构互连，其中 P_0 为主处理机，$\{P_i | i \in \{1,2,...,N\}\}$ 为从处理机，主处理机到每个从处理机都有一条通信传输链路，连接 P_0 和 P_i 的通信链路为 l_i。由于是异构系统，处理机之间的链路传输速率和每台处理机的计算速率不完全相同。从处理机 P_i 的释放时间为 r_i，其中 $i=1,2,...,N$。需要处理的任务最开始位于主处理机 P_0 上，任务总大小为 W_{total}。

异构并行与分布式平台下的可分任务调度模型需要重点考虑三个方面，其中处理机的调度顺序对可分任务调度完成时间有着很直接的影响，所以针对异构平台，不能随便假定一个处理机调度顺序，而应该在建立模型的时候加以考虑。记 $\sigma=(\sigma_1,\sigma_2,...,\sigma_N)$ 为 $1\sim N$ 的一个排列，则 $P_S=(P_{\sigma_1},P_{\sigma_2},...,P_{\sigma_N})$ 表示处理机的一个调度顺序。不是所有的处理机都参与计算，故假设参与计算的处理机数目为 n，即 P_S 的前 n 台处理机 $P_{\sigma_1},P_{\sigma_2},...,P_{\sigma_N}$ 参与计算。主处理机 P_0 将任务划分为 n 个子任务 $\alpha_1,\alpha_2,...,\alpha_n$，并将这些子任务依次按顺序分配到从处理机 $P_{\sigma_1},P_{\sigma_2},...,P_{\sigma_n}$ 上进行处理，其中 $\sum_{i=1}^{n}a_i=W_{total}$。与第

中国知网 https://www.cnki.net

三章类似，考虑到并行与分布式系统以处理大型任务为主，计算时间较长，因此从处理机反馈结果给主处理机的时间可以忽略不计。

对于异构星型网络，我们令 e_i 和 f_i 分别表示处理机 P_i 的传输启动开销和计算启动开销； z_i 表示主处理机 P_0 与 P_i（$i=1,2,...,N$）之间的通信链路传输单位任务所需要的时间，则传输任务块 α_i 给从处理机 P_i 的时间为 $e_i+z_i\alpha_i$；令 w_i 表示处理机 P_i（$i=1,2,...,N$）计算单位任务所需要的时间，则其计算分配给它的任务块的时间为 $f_i+w_i\alpha_i$。

对于异构并行与分布式平台，我们需要确定参与计算的处理机、最优的调度顺序以及最优的任务块分配策略使得任务完成的时间最短。下面，我们将通过详细的数学推导建立基于异构并行与分布式平台的可分任务调度模型。

假设参与计算的前 n 台处理机调度顺序为 $P_{\sigma_1},P_{\sigma_2},...,P_{\sigma_n}$，其中 $\sigma_1,\sigma_2,...,\sigma_n$ 为 $1\sim N$ 任一排列的前 n 个数，记处理机 $P_{\sigma_i}\ (i=1,2,...,n)$ 开始从主处理机接收任务的时刻为 s_{σ_i}。给定所有处理机的释放时间，我们分三种情况讨论处理机释放时间和开始接收任务时刻可能满足的约束关系。图 4.2、4.3 和 4.4 分别给出了满足约束条件 I、II 和 III 情况下的可分任务调度图。

I. $r_{\sigma_{i+1}}\le s_{\sigma_i}+z_{\sigma_i}\alpha_i+\mathrm{e}_{\sigma_i}, i=1,2,...,n-1$：不等式左边为从处理机 $P_{\sigma_{i+1}}$ 的释放时间，右边表示主处理机 P_0 向从处理机 P_{σ_i} 的传输任务结束时刻。该约束表示的是处理机 $P_{\sigma_{i+1}}$ 的释放时间 $r_{\sigma_{i+1}}$ 要早于 P_0 给 $P_{\sigma_{i+1}}$ 分配任务的时刻，即 $P_{\sigma_{i+1}}$ 由忙碌状态转为空闲状态发生在 P_0 给 P_{σ_i} 传输任务 α_i 的过程中，一种可能的调度图如 4.2 所示。

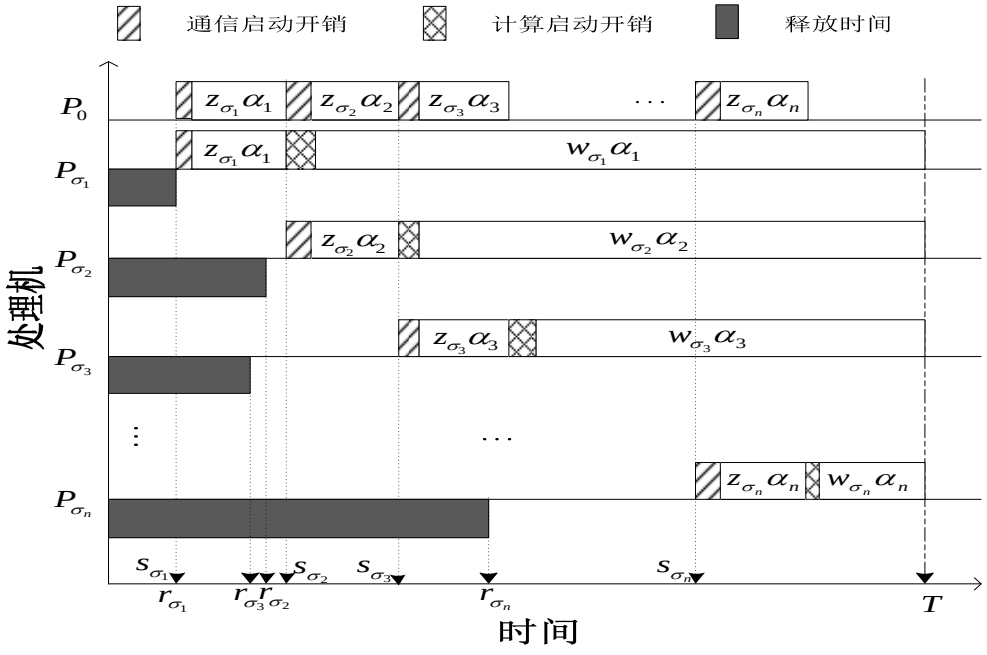

图4.2 一种满足约束条件 I 的异构系统可分任务调度图

中国知网 https://www.cnki.net

由于需要处理的任务总量为W_{total}，即所有从处理机上任务之和为W_{total}，则可得到递推公式(4-1)：

$$\sum_{i=1}^{n} a_i = W_{total} \tag{4-1}$$

文献[15]证明了只有当所有处理机同时完成计算的时候，任务的完成时间最短。这个结论在上一章的同构并行与分布式平台下得到了运用，并取得了很好的结果，同样也可以运用在异构并行与分布式平台中，得到递推公式(4-2)：

$$s_{\sigma_i} + e_{\sigma_i} + z_{\sigma_i}\alpha_i + f_{\sigma_i} + w_{\sigma_i}\alpha_i = s_{\sigma_{i+1}} + e_{\sigma_{i+1}} + z_{\sigma_{i+1}}\alpha_{i+1} + f_{\sigma_{i+1}} + w_{\sigma_{i+1}}\alpha_{i+1},\ i = 1,2,...,n-1. \tag{4-2}$$

等式左边表示处理机P_{σ_i}的任务完成时间，即任务开始传输时刻s_{σ_i}加上主处理机P_0传输任务块α_i给从处理机P_i的时间$e_i + z_i\alpha_i$再加上处理机P_i计算任务块α_i所需的时间$f_i + w_i\alpha_i$。同理，等式的右边表示处理机$P_{\sigma_{i+1}}$的任务完成时间，即任务开始传输时刻$s_{\sigma_{i+1}}$加上主处理机P_0传输任务块α_{i+1}给从处理机P_{i+1}的时间$e_{i+1} + z_{i+1}\alpha_{i+1}$再加上处理机$P_{i+1}$计算任务块$\alpha_{i+1}$所需的时间$f_{i+1} + w_{i+1}\alpha_{i+1}$。

由图 4.2，我们可以得到：

$$s_{\sigma_i} = s_{\sigma_{i-1}} + z_{\sigma_{i-1}}\alpha_{i-1} + e_{\sigma_{i-1}},\ \ i = 2,3,...,n \tag{4-3}$$

递推公式(4-3)表示那个处理机P_i的开始时间等于处理机P_{i-1}的传输任务结束时间，由于$s_1 = r_1$，递推可以得到s_i关于r_1的递推式

$$\begin{cases} s_{\sigma_1} = r_{\sigma_1} \\ s_{\sigma_i} = s_{\sigma_{i-1}} + z_{\sigma_{i-1}}\alpha_{i-1} + e_{\sigma_{i-1}} \\ s_{\sigma_n} = s_{\sigma_{n-1}} + z_{\sigma_{n-1}}\alpha_{n-1} + e_{\sigma_{n-1}} \end{cases} \Rightarrow\ s_i = r_{\sigma_1} + \sum_{j=1}^{i-1}(z_{\sigma_j}\alpha_j + e_{\sigma_j}),\quad i = 2,3,...,n \tag{4-4}$$

将递推公式(4-2)中的s_i用递推公式(4-4)得出的递推式进行替换，化简后可以得出α_i关于α_{i-1}的表达式：

$$\alpha_i = \frac{w_{\sigma_{i-1}}\alpha_{i-1}}{z_{\sigma_i} + w_{\sigma_i}} + \frac{f_{\sigma_{i-1}} - e_{\sigma_i} - f_{\sigma_i}}{z_{\sigma_i} + w_{\sigma_i}},\ \ i = 2,3,...,n \tag{4-5}$$

中国知网 https://www.cnki.net

此处我们令 $q_i = \frac{w_{\sigma_{i-1}}}{z_{\sigma_i} + w_{\sigma_i}}$，$v_i = \frac{f_{\sigma_{i-1}} - e_{\sigma_i} - f_{\sigma_i}}{z_{\sigma_i} + w_{\sigma_i}}$，则递推公式(4-5)可以重写为：

$$\alpha_i = q_{i-1}\alpha_{i-1} + v_{i-1},\ i = 2,3,...,n \tag{4-6}$$

由于异构并行与分布式平台数学模型较为复杂，涉及的变量很多，无法像第三章一样准确的求出任务最短完成时间的递归公式，但是我们可以得到每台从处理机分配任务大小的递推公式，进而求得任务最短完成时间。由递推公式可知，每个处理机分配任务大小与处理机调度顺序，处理机计算速度，链路传输速度，计算启动开销和传输启动开销均有直接关系。

II. $r_{\sigma_{i+1}} > s_{\sigma_i} + z_{\sigma_i}\alpha_i + \mathrm{e}_{\sigma_i}, i = 1,2,...,n-1$：该约束表示的是从处理机 $P_{\sigma_{i+1}}$ 的释放时间 $r_{\sigma_{i+1}}$ 要晚于主处理机 P_0 给 P_{σ_i} 分配完任务 α_i 的时刻，即 P_0 在给 P_{σ_i} 传输任务 α_i 后需要等待一段时间直到 $P_{\sigma_{i+1}}$ 恢复空闲才能为其传输任务 α_{i+1}，调度图如 4.3 所示。

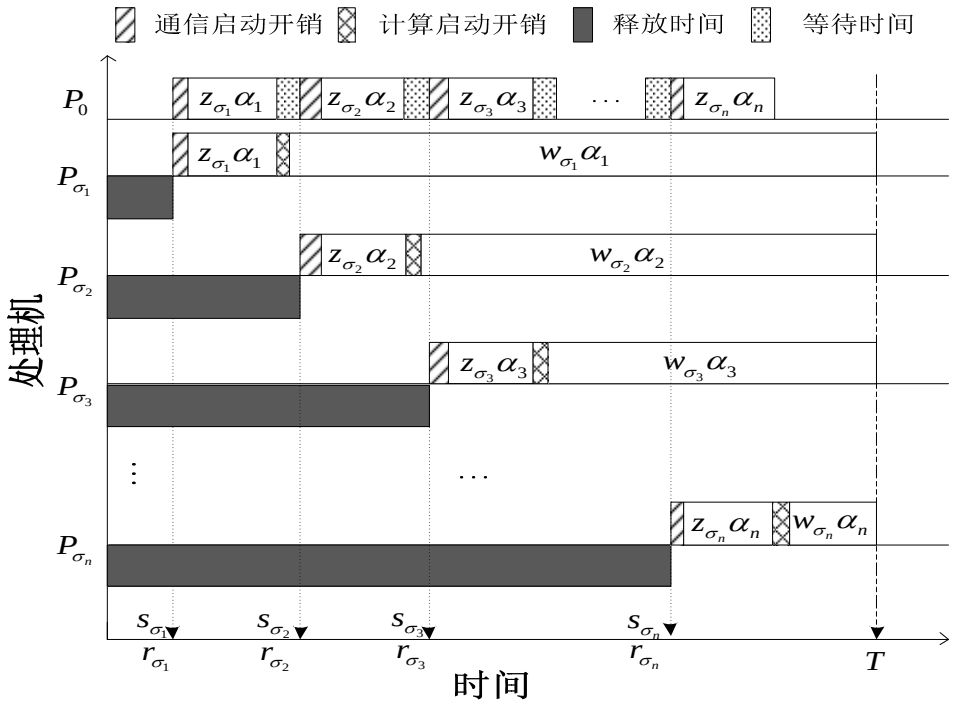

图4.3 一种满足约束条件 II 的异构系统可分任务调度图

同样地，根据所有参与计算处理机同时完成计算可以得到递推公式(4-2)。由图 4.3 可以看出，处理机 $P_{\sigma_i}\ (i = 1,2,...,n)$ 的释放时间 r_{σ_i} 和开始时间 s_{σ_i} 是相同的，即 $r_{\sigma_i} = s_{\sigma_i}$。将等式递推公式(4-2)中的 s_{σ_i} 替换为 r_{σ_i} 可以得到：

$$\alpha_i = \frac{z_{\sigma_1} + w_{\sigma_1}}{z_{\sigma_i} + w_{\sigma_i}}\alpha_1 + \frac{r_{\sigma_1} - r_{\sigma_i} + e_{\sigma_1} + f_{\sigma_1} - e_{\sigma_i} - f_{\sigma_i}}{z_{\sigma_i} + w_{\sigma_i}},\ i = 2,3,...,n. \tag{4-7}$$

中国知网 https://www.cnki.net

结合递推公式(4-1)，此处我们记 $t_i = \frac{z_{\sigma_1} + w_{\sigma_1}}{z_{\sigma_i} + w_{\sigma_i}}$， $u_i = \frac{r_{\sigma_1} - r_{\sigma_i} + e_{\sigma_1} + f_{\sigma_1} - e_{\sigma_i} - f_{\sigma_i}}{z_{\sigma_i} + w_{\sigma_i}}$。

将 t_i 和 u_i 代入递推公式(4-6)，由 $\sum_{i=1}^{n} a_i = W_{total}$，可得：

$$\alpha_1 = \left(W_{total} - \sum_{i=2}^{n} u_i \right) \Big/ \sum_{i=1}^{n} t_i \tag{4-8}$$

约束 II 情况下，我们可以得到每台处理机分配任务大小的公式，由公式可知，每个处理机分配任务大小与处理机调度顺序，处理机计算速度，链路传输速度，计算启动开销和传输启动开销均有直接关系。

III. 混合时序约束：在实际的并行与分布式系统中，任意两个相邻的从处理机对之间可能满足约束条件 I，也可能满足条件 II。假设有 n 台处理机参与计算，所有可能的情况有 2^{n-1} 种。图 4.4 给出了一种可能的混合时序约束的可分任务调度图。

同样地，此处我们用 $C = (c_2, c_3, ..., c_n)$ 表示参与计算的 n 个处理机满足的一种可能的混合时序约束条件，其中 c_i=1或0，$i = 2,3,...,n$。若 $c_i = 1$，表明主处理机 P_0 在给调度序列中第 $i-1$ 台处理机 $P_{\sigma_{i-1}}$ 传输完任务数据块后紧接着为调度序列中第 i 台处理机 P_{σ_i} 传输数据，中间没有空闲等待。此时，调度序列中第 i 台处理机 P_{σ_i} 的开始时间满足 $s_{\sigma_i} = s_{\sigma_{i-1}} + z_{\sigma_{i-1}} \alpha_{\sigma_{i-1}} + e_{\sigma_{i-1}}$。若 $c_i = 0$，表明主处理机 P_0 在给调度序列中第 $i-1$ 台处理机 $P_{\sigma_{i-1}}$ 传输完任务数据块后需要等待调度序列中第 i 台处理机 P_{σ_i} 由忙碌状态转为空闲状态才能开始传输数据，中间存在空闲等待时间。此时，调度序列中第 l_i 台处理机 P_0 的开始时间满足 $s_{\sigma_i} = r_{\sigma_i}$。图 4.4 给出了一种满足混合约束条件 C 的可分任务调度图，其中 $C = (1,...,1,0,0,...,0,1,...,1)$。

中国知网 https://www.cnki.net

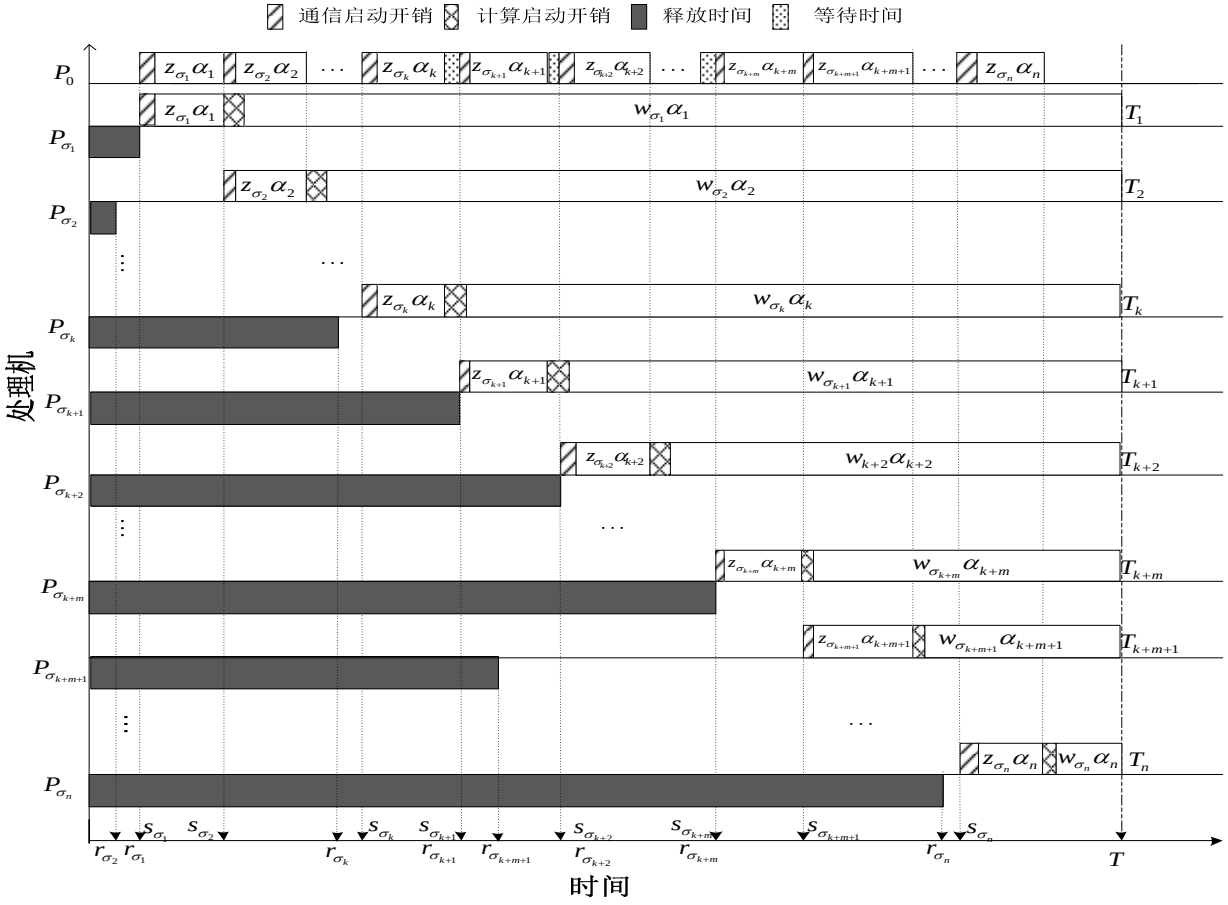

图4.4 一种可能的混合时序约束的异构系统可分任务调度图

根据图 4-4 满足的时间约束条件C，可以得到处理机开始时间与释放时间之间的递推公式(4-9)所示。

$$
\begin{cases}
s_{\sigma_1} = r_{\sigma_1} \\
s_{\sigma_2} = s_{\sigma_1} + z_{\sigma_1}\alpha_1 + e_{\sigma_1} \\
\vdots \\
s_{\sigma_k} = s_{\sigma_{k-1}} + z_{\sigma_{k-1}}\alpha_{k-1} + e_{\sigma_{k-1}} \\
s_{\sigma_{k+1}} = r_{\sigma_{k+1}} \\
s_{\sigma_{k+2}} = r_{\sigma_{k+2}} \\
\vdots \\
s_{\sigma_{k+m}} = r_{\sigma_{k+m}} \\
s_{\sigma_{k+m+1}} = s_{\sigma_{k+m}} + z_{\sigma_{k+m}}\alpha_{k+m} + e_{\sigma_{k+m}} \\
\vdots \\
s_{\sigma_n} = s_{\sigma_{n-1}} + z_{\sigma_{n-1}}\alpha_{n-1} + z_{\sigma_{n-1}}
\end{cases}
\tag{4-9}
$$

中国知网 https://www.cnki.net

又根据所有处理机同时完成计算的优化条件，将递推公式(4-2)，代入递推公式(4-9)可得：

$$\begin{cases} \alpha_1 = \alpha_1 \\ \alpha_2 = q_1\alpha_1 + v_1 \\ \vdots \\ \alpha_k = q_{k-1}\alpha_{k-1} + v_{k-1} \\ \alpha_{k+1} = t_{k+1}\alpha_1 + u_{k+1} \\ \alpha_{k+2} = t_{k+2}\alpha_1 + u_{k+2} \\ \vdots \\ \alpha_{k+m} = t_{k+m}\alpha_1 + u_{k+m} \\ \alpha_{k+m+1} = q_{k+m}\alpha_{k+m} + v_{k+m} \\ \vdots \\ \alpha_n = q_{n-1}\alpha_{n-1} + v_{n-1} \end{cases} \tag{4-10}$$

其中：

$q_j = \dfrac{w_{\sigma_j}}{z_{\sigma_{j+1}} + w_{\sigma_{j+1}}}$, $j = 1,2,...,n-1$， $t_i = \dfrac{z_{\sigma_1} + w_{\sigma_1}}{z_{\sigma_i} + w_{\sigma_i}}$, $u_i = \dfrac{r_{\sigma_1} - r_{\sigma_i}}{z_{\sigma_i} + w_{\sigma_i}}$, $i = 1,2,...,n$。

由递推公式(4-10)的$n-1$个等式和递推公式(4-1)组成n个等式，可以得到如下标准式：

$$A \cdot \alpha = b \tag{4-11}$$

其中α表示的是待求的$n \times 1$维的解向量$(\alpha_1, \alpha_2, ..., \alpha_n)$，$A$是$n \times n$的系数矩阵，$b$是$n \times 1$维的列向量。对于图3.4给出的可分任务调度图中，$A$和$b$分别如递推公式(4-12)和递推公式(4-13)所示。

中国知网 https://www.cnki.net

$$A=\begin{bmatrix}
-q_1 & 1 & 0 & \cdots & & & & & & & 0\\
\vdots & \ddots & \ddots & \ddots & \cdots & & & & & & \vdots\\
\vdots & \ddots & \ddots & \ddots & \ddots & \cdots & & & & & \vdots\\
0 & \cdots & 0 & -q_{k-1} & 1 & 0 & \cdots & & & & 0\\
-t_{k+1} & 0 & \cdots & & 0 & 1 & 0 & \cdots & & & 0\\
-t_{k+2} & 0 & & \cdots & & 0 & 1 & 0 & \cdots & & 0\\
\vdots & \vdots & & & \cdots & & \ddots & \ddots & & \cdots & \vdots\\
-t_{k+m} & 0 & & & & \cdots & & 1 & 0 & \cdots & 0\\
0 & 0 & & & & & \cdots & -q_{k+m} & 1 & & \\
\vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots\\
0 & & & & & & & \cdots & 0 & -q_{n-2} & 1 & 0\\
0 & 0 & & \cdots & 0 & 0 & \cdots & 0 & \cdots & 0 & -q_{n-1} & 1\\
1 & 1 & & & \cdots & 1 & 1 & & & \cdots & 1 & 1
\end{bmatrix} \tag{4-12}$$

$$b=\left(v_1,v_2,\ \cdots,\ v_{k-1},\ u_{k+1},\ u_{k+2},\ \cdots,\ u_{k+m},\ v_{k+m},v_{k+m+1},\ \cdots,\ v_{n-1},\ W_{total}\right)^T \tag{4-13}$$

通过上述分析，我们可以知道给定一个调度顺序以及一种混合时序约束条件 $C=(c_2,c_3,...,c_n)$，就可以得到形如递推公式(4-11)的标准式，用线性规划方法可以求出任务分配方案 α 的解。

和上一节定义的一样，$P_{\sigma_1},P_{\sigma_2},...,P_{\sigma_n}\ (0<n\le N)$ 为参与计算的处理机调度序列。基于本节三种约束条件的分析，异构系统下混合时序约束的可分任务调度的优化模型可以表示为：

$$\min_{\sigma,C} T=\min\left(r_{\sigma_1}+\left(z_{\sigma_1}+w_{\sigma_1}\right)\alpha_1+e_{\sigma_1}+f_{\sigma_1}\right)$$

s.t.

1. $\sigma=\left(\sigma_1,\sigma_2,...,\sigma_n\right)$为1~$N$的一个排列, 且有$0<n\le N$;
2. $\sum_{i=1}^{n}\alpha_i=W_{total}$，其中$0<\alpha_i\le W_{total}, i=1,2,...,n$;
3. $s_{\sigma_i}+z_{\sigma_i}\alpha_i+w_{\sigma_i}\alpha_i+e_{\sigma_i}+f_{\sigma_i}=s_{\sigma_{i+1}}+z_{\sigma_{i+1}}\alpha_{i+1}+w_{\sigma_{i+1}}\alpha_{i+1}+e_{\sigma_{i+1}}+f_{\sigma_{i+1}},\ i=1,2,...,n-1$;
4. $C=(c_2,c_3,...,c_n)$, 其中$c_i=0$或1, $i=2,3,...,n$;
5. $\begin{cases} r_{\sigma_{i+1}}\le s_{\sigma_i}+z_{\sigma_i}\alpha_i+e_{\sigma_i}, & \text{当}(c_i=1)\\ r_{\sigma_{i+1}}> s_{\sigma_i}+z_{\sigma_i}\alpha_i+e_{\sigma_i}, & \text{当}(c_i=0)\end{cases}, i=2,3,...,n;$
6. $s_{\sigma_i}=\begin{cases} s_{\sigma_{i-1}}+z_{\sigma_{i-1}}\alpha_{i-1}+e_{\sigma_{i-1}}, & \text{当}(c_i=1)\\ r_{\sigma_i}, & \text{当}(c_i=0)\end{cases}, i=2,3,...,n.$

可分任务调度模型的目标是使得任务的完成时间最短。模型的约束条件 1 表示并不一定要求所有的处理机都参与任务的计算，即参与计算的处理机数目可以小于 N；参与计算的 n 个处理机的调度顺序为：$P_{\sigma_1},P_{\sigma_2},...,P_{\sigma_n}$；约束条件 2 表示每台处理机分配

中国知网 https://www.cnki.net

的任务量非负，且任务量之和为W_{total}；约束条件 3 表示所有处理机必须同时完成任务的处理，当且仅当在满足这个条件的情况下，可分任务调度方案才能够得到最优解；约束条件 4 和 5 限制了C的取值范围：当$c_i=1$，处理机$P_{\sigma_{i-1}}$和P_{σ_i}必须满足情况 I，即$r_{\sigma_{i+1}} \le s_{\sigma_i}+z_{\sigma_i}\alpha_i+e_{\sigma_i}$；当$c_i=0$，处理机$P_{\sigma_{i-1}}$和$P_{\sigma_i}$必须满足情况 II，即$r_{\sigma_{i+1}} > s_{\sigma_i}+z_{\sigma_i}\alpha_i$；约束条件 6 给出了处理机$P_{\sigma_i}$的开始时间$s_{\sigma_i}$的取值范围：当$c_i=1$，$P_0$给$P_{\sigma_{i-1}}$传输完数据后紧接着为$P_{\sigma_i}$传输数据，因此$P_{\sigma_i}$的开始时间为$P_{\sigma_{i-1}}$的开始时间$s_{\sigma_{i-1}}$加上$P_0$为$P_{\sigma_{i-1}}$传输任务的时间$z_{\sigma_{i-1}}\alpha_{i-1}+e_{\sigma_{i-1}}$，即$s_{\sigma_i}=s_{\sigma_{i-1}}+z_{\sigma_{i-1}}\alpha_{i-1}+e_{\sigma_{i-1}}$；当$c_i=0$，$P_0$给$P_{\sigma_{i-1}}$传输完数据后需要等待$P_{\sigma_i}$由忙碌转为空闲才能开始传输数据，因此$P_{\sigma_i}$的开始时间与其释放时间相同，即$s_{\sigma_i}=r_{\sigma_i}$。以上六个约束必须全部满足，才能构成可分任务调度解的一个方案。

本小节我们对异构并行与分布式平台星型网路拓扑结构下混合时序约束可分任务调度进行了详细的讨论和分析，并建立了以任务完成时间最短为目标的带约束的单目标优化模型，下一节我们将根据上面建立的优化模型，设计遗传算法来求解可分任务调度模型。

4.3 可分任务调度算法

4.3.1 编解码和适应度函数

根据上面建立的任务调度模型，我们采用实数编码的形式，将异构并行与分布式平台下混合时序约束的可分任务调度问题表示成一个向量$I=(n,\gamma,\sigma,\mathrm{C})$，其中$n$表示参与计算的处理机数目，假设从处理机数目为$N$，种群初始化时，假定所有从处理机都参与任务处理，即将n的大小设为N，$\gamma=(\gamma_1,\gamma_2,...,\gamma_N)$是一个随机数串，且满足$0\le\gamma_i\le i-1,\ i=1,2,...,n$；每个$\gamma$串都与$\sigma=(\sigma_1,\sigma_2,...,\sigma_N)$串一一对应，可以用来简化交叉变异等操作，$\sigma=(\sigma_1,\sigma_2,...,\sigma_N)$为$1\sim N$的一个排列，表示的是系统中处理机调度顺序。$C=(c_2,c_3,...,c_N)$表示一种混合时序约束条件，$c_i\in\{1,0\}$。若$c_i=1$，则表示处理机对$P_{\sigma_{i-1}}$和$P_{\sigma_i}$满足约束条件 I；反之若$c_i=0$，则表示处理机对$P_{\sigma_{i-1}}$和$P_{\sigma_i}$满足约束条件 II。例如，假设共有 6 台从处理机，则一种可能的编码如表 4.1 所示：

表4.1 含 6 台从处理机异构系统的编码示例

I	n	γ	σ	C
编码	6	0 1 0 1 0 2	2 1 4 6 3 5	1 0 1 1 0

其中，第一位$n=6$，表示参与计算的处理机数目（初始阶段设定为从处理机总数）；第二位到第七位$\gamma_1\sim\gamma_6$是一个随机数串，第八位到第十三位$\sigma_1\sim\sigma_6$表示的是处理机的调度顺序为P_2,P_1,P_4,P_6,P_3,P_5，第八位$c_2=1$，表示调度序列中第 1 台处理机P_2和第 2 台处理机P_1之间满足约束 I；第九位$c_3=0$表示调度序列中第 2 台处理机P_1和

中国知网 https://www.cnki.net

第 3 台处理机P_4之间满足约束 II；第十位$c_4=1$表示第 3 台处理机和第 4 台处理机之间满足约束 I；第十一位$c_5=1$表示第 4 台处理机和第 5 台处理机之间满足约束 I；第十二位$c_6=1$表示第 5 台处理机和第 6 台处理机之间满足约束 II。

编码过程将个体中的σ串转换成为γ串，方便交叉变异等操作。对σ串中的每个数字σ_i，计算σ串中满足下标比i大且值比σ_i小的元素的数目，即为对应σ_{γ_i}的值。假设σ串为（2 1 4 6 3 5），则σ_1为 2，后面比 2 小的值只有一个，则γ_2为 1；σ_2为 1，后面没有比 1 小的值，则γ_1为 0；σ_3为 4，后面比 4 小的值只有一个，则γ_4为 1；以此类推可得γ串为（0 1 0 1 0 2）。

解码过程将随机数γ串转换成σ串。$\gamma=(\gamma_1,\gamma_2,...,\gamma_N)$为一个随机数串，对于其中的任意一位$\gamma_i$，其为$0\sim i-1$之间的一个随机值。最开始的时候，$\sigma$串是一个空串，根据$\gamma$串将$\mathrm{N}\sim 1$依次放入$\gamma$，即完成了一次编码过程。假设现在要将$i$放入$\sigma$串中，先查看$\gamma_i$的值，此时$\gamma_i$的值表示从后往前数，$i$要经过空元素的个数，这样就可以得到对应于$\gamma$串的$\sigma$串。假设随机生成的$\gamma$串为(0 1 0 1 0 2)，则将 6~1 依次放入σ串的过程为：首先将 6 放入σ串中，查看此时的γ_6为 2，则从σ串后面往前数 2 个空元素，下一个空元素的位置即为 6 的存放位置，即可得σ_4为 6；将 5 放入σ串中，查看此时的γ_5为 0，则从σ串后面往前数 0 个空元素，下一个空元素的位置即为 5 的存放位置，即可得σ_6为 5；将 4 放入σ串中，查看此时的γ_4为 1，则从σ串后面往前数 1 个空元素，下一个空元素的位置即为 4 的存放位置，此时σ_4和σ_6为分别含有元素 6 和 5，σ_5为空元素，即可得σ_3为 4；以此类推可得σ串为（2 1 4 6 3 5）。

任意一个个体I，给定n、σ和C，就可以得到对应的$n\times n$的系数矩阵A和向量b，并将问题转化为递推公式(4-11)的标准形式。通过线性规划方法求解该标准式就可以得到任务分配方案α的解。然后我们需要验证求解出的α是否满足模型所有的约束条件，尤其是约束 2 和约束 5。若满足模型的所有约束条件，则个体I对应唯一一个类似于图 4.4 的混合时序可分任务调度图，调度方案α即为可行解，将该方案对应的任务完成时间$T=r_{\sigma_1}+(z_{\sigma_1}+w_{\sigma_1})\alpha_{\sigma_1}+e_{\sigma_1}+f_{\sigma_1}$作为个体$I$的适应度值。如果$\alpha$不满足模型的部分约束条件，则表明并不需要这么多的处理机参与计算，令$n=n-1$，更新个体I，重复这个过程，直至求出一个满足模型全部约束条件的解。

4.3.2 交叉和变异

交叉算子：对个体不同的基因段采用不同的交叉策略，对于第一个基因段，无论两个父代个体为何值，生成的两个子代个体均赋值为N。对于γ基因段，采用双点交叉方式。随机生成两个整数p和q满足$2\le p<q\le N$作为交叉点，将两个父代个体交叉位点γ_p和γ_q之间的基因进行交换。举例来说，处理机总数$\mathrm{N}=6$，随机p和q数分别为 2 和 4，通过父代个体I^1和I^2，生成子代个体I^3和I^4。γ基因段双点交叉方式如

中国知网 https://www.cnki.net

下：

$$I^1=(\mathrm{n}^1,\gamma^1,\sigma^1,\mathrm{C}^1)=(4,\ 0\,1\,0\,1\,0\,2,\ 2\,1\,4\,6\,3\,5,\ 1\,0\,1\,1\,0)$$

$$I^2=(\mathrm{n}^2,\gamma^2,\sigma^2,\mathrm{C}^2)=(5,\ 0\,1\,2\,3\,0\,3,\ 4\,3\,6\,2\,1\,5,\ 0\,1\,0\,1\,0)$$

交换γ 串 2~4 之间的基因片段可得：

$$\gamma^3=(\gamma_1^3\ \gamma_2^3\ \gamma_3^3\ \gamma_4^3\ \gamma_5^3\ \gamma_6^3)=(0\,1\,2\,3\,0\,2)$$

$$\gamma^4=(\gamma_1^4\ \gamma_2^4\ \gamma_3^4\ \gamma_4^4\ \gamma_5^4\ \gamma_6^4)=(0\,1\,0\,1\,0\,3)$$

对于σ基因段，通过对交叉后子代的γ基因段进行解码操作即可得到。

对于C基因段，采用两点交叉方式。随机生成两个整数p和q满足$2\le p<q\le N$作为交叉点，将两个父代个体交叉位点C_p和C_q之间的基因进行交换。假设随机生成的位点为 2 和 5，经过交叉操作生成的两个后代个体如下：

$$I^3=(\mathrm{n}^3,\gamma^3,\sigma^3,\mathrm{C}^3)=(6,\ 0\,1\,2\,3\,0\,2,\ 4\,3\,2\,6\,1\,5,\ 1\,1\,0\,1\,0)$$

$$I^4=(\mathrm{n}^4,\gamma^4,\sigma^4,\mathrm{C}^4)=(6,\ 0\,1\,0\,1\,0\,3,\ 2\,1\,6\,4\,3\,5,\ 0\,0\,1\,1\,0)$$

变异算子：对个体不同的基因段采用不同的变异策略。对于第一个基因段，变异后的个体赋值为N。对于γ基因段，随机选择该基因段内一个基因位，产生一个符合条件的随机数替换相应基因位即可。对于C基因段，随机选择该基因段一个基因位，对其进行取反操作。举例而言$I=(\mathrm{n},\gamma,\sigma,\mathrm{C})=(4,\ 0\,1\,0\,1\,0\,2,\ 2\,1\,4\,6\,3\,5,\ 1\,0\,1\,1\,0)$。对$\gamma$基因段位置 3 进行变异，随机数为 2，则$\gamma_3=2$，$C$基因段位置 4 取反，得变异后代：$I'=(\mathrm{n}',\gamma',\sigma',\mathrm{C}')=(6,\ 0\,1\,2\,1\,0\,2,\ 3\,2\,4\,6\,1\,5,\ 1\,0\,1\,0\,0)$。

4.3.3 局部搜索算子

从上面的编码方案可以看出，异构并行与分布式平台下的可分任务调度算法编码空间需要重点考虑处理机调度顺序和约束条件两个方面。相比第三章的同构并行与分布式平台下的算法搜索空间有了极大的增加，因此为了能够加快算法进化收敛速度，本文引入局部搜索算子来加快优化速度。局部搜素策略的主要过程如下：

1、对种群中的每个个体I，其初始适应度值记为$f(I)$。依次交换$\sigma=(\sigma_1,\sigma_2,...,\sigma_N)$串中相邻的两个位置的值，即从$i=0\sim N-1$依次交换$\sigma_i$和$\sigma_{i+1}$的值，且对于个体$I'$其基因段$n$更新为处理机总数$N$，得到个体$I'$，计算新个体$I'$的适应度值，如果更新后的个体适应度值$f(I')$大于更新之前的适应度值$f(I)$，则$I=I'$；反之，个体$I$保持先前的状态不变。如果连续$N$次交换都不能得到更好地解，则结束这一步骤。

2、经过步骤 1，我们得到了约束条件不变情况下，调度顺序更优的个体，其适应度值为$f(I)$，然后对其C基因段进行优化。对于$n-1$个位点中的每一个位点，我们先将其对应位取反，基因段n更新为处理机总数N，记为个体I''，计算其适应度值$f(I'')$，若$f(I'')>f(I)$，则$I=I''$；反之，个体I保持原状态不变。对C基因段依

中国知网 https://www.cnki.net

次尝试取反，直到最后一位。取反完成后，结束该局部搜索过程。

以上所述过程即为本文采用的局部搜索算子的基本流程，后面的实验结果证明该局部搜索算子能够极大地加快算法的优化速度和收敛效率。

4.3.4 选择算子

本章我们仍采用的是精英保留和联赛选择方法相结合的选择算子，选择算子同3.3.3节。

4.3.5 算法框架

基于前面设计的编码以及交叉、变异、选择等遗传算子，我们给出求解混合时序约束的可分任务调度遗传算法框架，如算法4.1所示：

算法4.1 异构系统下混合时序约束可分任务调度遗传算法

1:（**初始化**） 确定种群大小 $PopSize$，交叉概率 p_{cros} 和变异概率 p_{mut} 等参数，根据编码规则随机生成初始种群 $P(0)$，令进化代数 $t=0$。

2:（**交叉**） 以概率 p_{cros} 从 $P(t)$ 之中选择父代个体进行双点交叉，交叉获得的全部后代个体定义为集合 O_1。

3:（**变异**） 以概率 p_{mut} 从集合 O_1 中选择个体进行变异，新的后代个体定义为集合 O_2。

4:（**局部搜索**） 对集合 $P(t)\cup O_1\cup O_2$ 中的所有个体进行局部搜索，更新后的所有个体集合记为 O_3。

5:（**选择**） 从集合 $P(t)\cup O_1\cup O_2\cup O_3$ 中选择最优的 EL 个个体直接保留到下一代种群以加快算法的收敛速度；使用联赛选择操作从集合 $P(t)\cup O_1\cup O_2\cup O_3$ 选择 $N-EL$ 个个体保留到下一代种群 $P(t+1)$ 中；令 $t=t+1$。

6:（**终止条件**） 如果满足终止条件，则终止算法；否则转向步骤2。

4.4 实验仿真与结果分析

上一节，我们详细介绍了混合时序约束可分任务调度遗传算法的设计过程，本节我们将通过一系列对比实验来对算法进行全面的评价和分析。

根据前面的分析可知，异构并行与分布式平台下任务调度受很多因素的影响，模型也更为复杂，假设从处理机数目为 N，其搜索空间规模达到了 $N!\cdot 2^{N-1}$。随着系统中处理机的增大，即随着变量 N 的增大，搜索空间将增大到惊人的规模， 用穷举法进行求解的开销是十分巨大的。因而本章第三节提出了用遗传算法求解该模型，并设计了相应的遗传算子。本节实验参数设置如下：从处理机总数 $N=15$。链路传输单位

中国知网 https://www.cnki.net

任务时间 $z_i\left(i=1,2,...,N\right)$ 为 0.1~1 之间的随机数，考虑到通常情况下，链路传输速率相较于处理机计算速率较快，因此对于处理机计算单位任务时间 $w_i\left(i=1,2,...,N\right)$，我们取 1~10 之间的随机数，对于处理机释放时间 $r_i\left(i=1,2,...,N\right)$，我们取 1~100 之间的随机数，表 4.2 给出了本章实验中所涉及的参数。遗传算法中的一些重要参数设置如下：种群大小 $PopSize=100$，交叉概率 $p_{cros}=0.6$，变异概率 $p_{mut}=0.02$，精英保留个数 $EL=5$，终止条件为进化代数 $t=2000$。

表4.2 异构系统参数

处理机	z	w	r	e	f	处理机	z	w	r	e	f
p_1	0.53	2.90	4.15	7.06	5.80	p_9	0.99	2.27	4.41	5.89	9.11
p_2	0.77	7.61	28.41	3.02	0.14	p_{10}	0.98	5.34	44.43	6.95	2.44
p_3	0.71	4.14	5.57	8.14	0.45	p_{11}	0.99	1.57	38.77	1.06	6.76
p_4	0.79	9.62	10.61	8.63	3.74	p_{12}	0.10	7.99	76.78	5.75	1.03
p_5	0.06	3.64	82.52	8.71	9.50	p_{13}	0.05	3.82	79.72	2.84	2.96
p_6	0.77	5.92	69.78	5.25	0.54	p_{14}	0.95	4.01	19.50	3.01	9.80
p_7	0.30	6.48	32.39	4.69	6.23	p_{15}	0.16	6.47	49.48	2.78	1.63
p_8	0.28	8.25	95.07	2.64	8.30						

文献[12]和[13]分析了异构分布式系统下处理机调度顺序对任务完成时间的影响。文献[12]和[13]中处理机遵从链路速度递减，即传输单位任务所需时间递增的顺序，记为 IZ；文献[15]中处理机遵从计算速度递减，即处理机计算单位任务所需时间递增的顺序，记为 IW；文献[42]中处理机遵从释放时间递增的顺序，记为 IR。本文采用遗传算法求解最优的处理机调度顺序，记为 GA。

表 4.3 给出了任务量从 100 到 500 时各个算法的运行结果。为了更加直观的对比各算法的优劣，图 4.5 和图 4.6 分别给出了任务的最短完成时间和参与计算的处理机个数随任务量的变化趋势。

表4.3 任务大小为 100-500 时各算法的运行结果

算法	任务大小	处理机数目	完成时间	处理机的调度顺序
GA	100	11	114.72	1, 14, 7, 15, 2, 11, 13, 9, 8, 6, 12
IZ		11	171.818	13, 5, 12, 15, 8, 7, 1, 3, 2, 6, 4

中国知网 https://www.cnki.net

IW		8	167.802	11, 9, 1, 5, 13, 14, 3, 10
IR		10	135.836	1, 9, 3, 4, 14, 2, 7, 11, 10, 15
GA	200	14	171.005	1, 7, 3, 15, 12, 13, 5, 8, 11, 13, 9, 2, 6, 10
IZ		15	227.589	13, 5, 12, 15, 8, 7, 1, 3, 2, 6, 4, 14, 10, 9, 11
IW		12	257.021	11, 9, 1, 5, 13, 14, 3, 10, 6, 15, 7, 2
IR		13	224.339	1, 9, 3, 4, 14, 2, 7, 11, 10, 15, 6, 12, 13
GA	300	15	221.542	1, 7, 15, 2, 13, 5, 12, 8, 3, 11, 14, 9, 6, 10, 4
IZ		15	278.101	13, 5, 12, 15, 8, 7, 1, 3, 2, 6, 4, 14, 10, 9, 11
IW		14	341.639	11, 9, 1, 5, 13, 14, 3, 10, 6, 15, 7, 2, 12, 8
IR		13	309.551	1, 9, 3, 4, 14, 2, 7, 11, 10, 15, 6, 12, 13
GA	400	15	274.221	1, 15, 7, 12, 13, 5, 8, 3, 2, 6, 11, 14, 9, 10, 4
IZ		15	328.613	13, 5, 12, 15, 8, 7, 1, 3, 2, 6, 4, 14, 10, 9, 11
IW		15	424.198	11, 9, 1, 5, 13, 14, 3, 10, 6, 15, 7, 2, 12, 8, 4
IR		15	391.669	1, 9, 3, 4, 14, 2, 7, 11, 10, 15, 6, 12, 13, 5, 8
GA	500	15	324.226	1, 15, 7, 13, 5, 12, 8, 3, 2, 6, 14, 11, 9, 10, 4
IZ		15	379.125	13, 5, 12, 15, 8, 7, 1, 3, 2, 6, 4, 14, 10, 9, 11
IW		15	506.516	11, 9, 1, 5, 13, 14, 3, 10, 6, 15, 7, 2, 12, 8, 4
IR		15	473.08	1, 9, 3, 4, 14, 2, 7, 11, 10, 15, 6, 12, 13, 5, 8

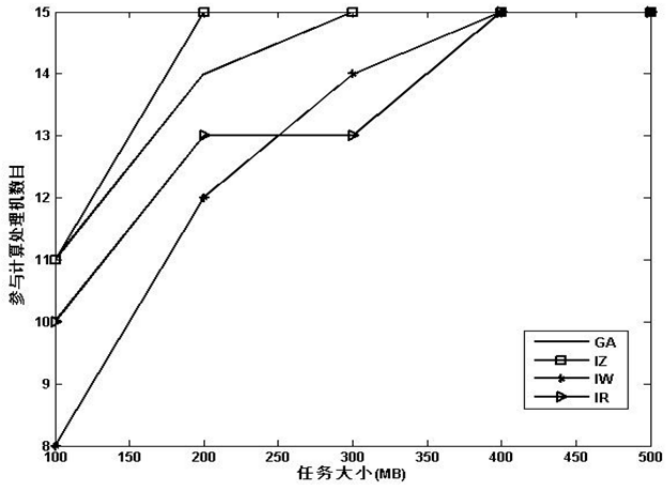

图4.5 参与计算的处理机数目的变化趋势

由图 4.5 可以看出， 任务量较小时，并不是所有处理机都参与计算，随着任务量增大，参与计算处理机不断增大，随着任务量增大到一定程度，所有处理机都参与计算。

中国知网 https://www.cnki.net

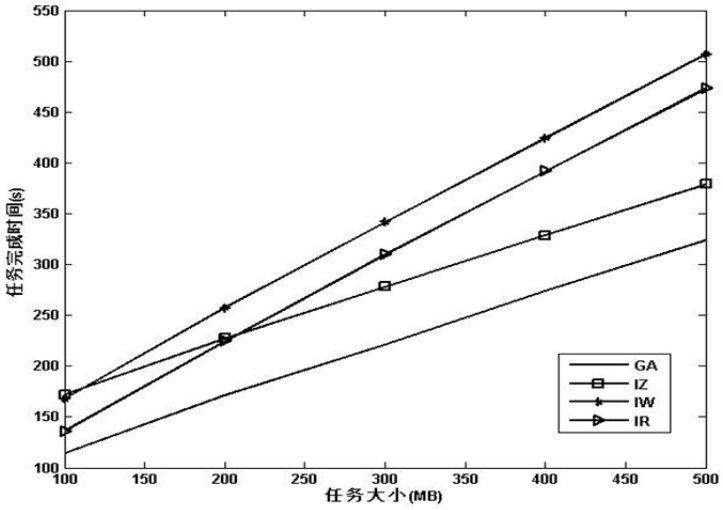

图4.6 任务最短完成时间的变化趋势

由图 4.6 可以看出本文提出的算法 GA 能得到最优的任务完成时间，其结果优于算法 IZ、IW 和 IR，并且从该图可以看出，随着任务量的增大，IR 算法的结果与 GA 算法的结果差距越来越大，一定程度后，其结果甚至比 IZ 算法的性能还要差一些，由此可以看出，随着任务量的逐渐增大，处理机的释放时间和启动开销对任务完成时间的影响逐渐减小，而处理机调度顺序的影响则逐渐增强。

4.5 本章小结

本章基于上一章同构系统混合时序约束可分任务调度模型的分析过程，提出并建立了异构系统下的混合时序约束可分任务调度模型，设计了新的遗传算法进行求解，从理论和实验两方面分析了处理机释放时间对异构系统调度的影响，实验结果证明了算法的有效性。本节提出的算法能够得到更优的调度方案和更短的任务完成时间。

中国知网 https://www.cnki.net

第五章 总结与展望

本文主要围绕并行与分布式系统中的任务调度问题和任务调度算法进行了深入的讨论和研究。本章对全文工作进行总结，并对未来将要进行的工作进行展望。

5.1 工作总结

并行与分布式平台被越来越广泛的推广和使用，成为支撑各类系统和应用的基础设施和平台。任务调度模型和算法的研究是并行与分布式平台推广和应用必须解决的一个关键问题，它直接决定了并行与分布式平台中资源有没有得到有效的利用。本文对并行与分布式平台下任务调度模型与算法进行了研究和探索，主要工作如下：

1、高效的基于遗传算法框架的可分任务调度算法研究

根据我们文章提出的更接近实际情况的可分任务调度模型和算法，针对同构和异构两种星型网络设计了不同的遗传算法进行问题的求解。根据不同系统的特点，抽象成数学模型，进而设计了不同的编码解码、交叉、变异、选择和局部搜索等操作，实验结果表明本文所提出算法具有很好的时间性能并且能够得到更优解。利用遗传算法来求解可分任务调度问题方面的工作和文章还比较少，这也是本文的另一个创新点。

2、考虑释放时间和启动开销的并行与分布式系统下可分任务调度模型的研究

对现有可分任务调度模型进行了分析和总结，将实际处理机调度过程中存在释放时间和启动开销等因素考虑其中，同时考虑到实际并行与分布式系统中存在同构和异构两种并行与分布式平台，针对两种不同的平台，分别建立了星型网络拓扑结构下考虑处理机释放时间和启动开销的可分任务调度模型，并进行了详细的数学推导和理论分析，建立了以任务完成时间最短为目标的可分任务调度优化模型。

5.2 未来的展望

本文基于可分任务理论，研究了更贴近实际问题，以任务完成时间最短为目标的可分任务调度模型和调度算法。我所做的工作是在已有的研究工作成果上的一些改进和扩展。可分任务调度理论在并行与分布式系统中的应用，在未来的研究工作中，主要有以下几个方向：

一 并行与分布式平台中任务调度算法验证与应用

现有的并行与分布式平台中任务调度算法大多数只有数学模型以及模拟求解，在实际平台上效果不得而知，也很难验证。如何能真正的将理论与实际联系起来，理论成果能够很好的转化到实际生产中，并解决实际平台上的任务调度问题是很有挑战性

中国知网 https://www.cnki.net

的。因此在以后的研究中，推进算法验证和实际应用是很值得研究的方向。

二 更接近实际问题的可分任务调度模型

1、如何设计出一种更接近实际问题的可分任务调度模型，且能够同时满足求解的简单性的平衡是一个很有挑战性的问题，这是在建立可分任务调度模型中必须考虑和研究的问题。

2、目前大多数的任务调度模型中任务都是可以任意划分的，而现实中可能有一些任务可能不能划分，即任务是部分可分的，有一些任务之间可能存在依赖关系。这类问题的模型建立也是一个值得深入研究的问题。

3、本文主要考虑的是单趟调度情况下建立的可分任务调度模型，而现在已有的一些文章和成果已经引入了多趟调度的技术。多趟调度在某种程度上能够提高系统的性能，但是多趟调度本身的解决方案还处于探索阶段，最优趟数的确定和求解仍是一个很大的难点。如何建立简单高效的多趟可分任务调度模型是未来一个研究方向。

4、目前大多数的可分任务调度模型是以任务完成时间最短为优化目标和方向的，而现实中还存在很多的系统不是追求最短完成时间，而是追求最大吞吐量等其他目标，对这类问题的研究也是很有意义的。

5、本文中的可分任务调度模型都假设处理机接收任务后可以一直运行，然后在实际的并行与分布式平台中，处理机可能存在下线时间，因而影响对该处理机分配的任务大小和顺序。如何建立考虑下线时间的任务调度模型是值得深入探索的问题。

可分任务调度问题已经取得了相当可观的进展和研究成果，同时也存在着很多急需解决的问题。建立更贴近与实际分布式网络系统的模型，并设计与之相对应的简洁高效的可分任务调度算法依然是未来很长一段时间内的研究重点。

中国知网 https://www.cnki.net

参考文献

[1] 尚明生. 网格计算中的任务调度算法研究[D].电子科技大学, 2007.

[2] Complexity results for scheduling problems[EB]. 2014.10.17. http://www.mathematik.uni-osnabrueck.de/researeh/OR/elass.

[3] CHENG Y C, ROBERTAZZI T G. Distributed Computation with Communication Delays[J]. IEEE Transactions on Aerospace and Electronic Systems, 1988, 24(6):700-712.

[4] ROBERTAZZI T G. Ten Reasons to Use Divisible Load Theory[J]. Computer, 2003, 36(5):63-68.

[5] BHARADWAJ V, GHOSE D, ROBERTAZZI T G. A New Paradigm for Load Scheduling in Distributed Systems[J]. Cluster Computing, 2003, 6(1):7-18.

[6] BHARADWAJ V, GHOSE D, MANI V, et al. Scheduling Divisible Loads in Parallel and Distributed Systems[M]. Los Alamitos CA, IEEE Computer Society Press, Sept. 1996, 292 pages.

[7] SHOKRIPOUR A, OTHMAN M. Categorizing Researches about DLT in Ten Groups[C]. //2009 International Association of Computer Science and Information Technology - Spring Conference, IACSIT-SC 2009, Los Alamitos CA: IEEE Computer Society Press, 2009:45-49.

[8] SHOKRIPOUR A, OTHMAN M. Survey on Divisible Load Theory and Its Applications[C]. //Proceedings-2009 International Conference on Information Management and Engineering, ICIME 2009, Los Alamitos CA: IEEE Computer Society Press, 2009: 300-304.

[9] MANI V, GHOSE D. Distributed Computation in Linear Networks: Closed Form Solutions[J]. IEEE Transactions on Aerospace and Electronic Systems. 1994, 30:471–483.

[10] GHOSE D, MANI V. Distributed Computation with Communication Delays: Asymptotic Performance Analysis[J]. Journal of Parallel and Distributed Computing.1994, 23:293–305.

[11] BATAINEH S, ROBERTAZZI T G. Ultimate Performance Limits for Networks of Load Sharing Processors[C].//Proceedings of the 1992 Conference on Information Sciences and Systems, Princeton, NJ, USA: Princeton University, 1992: 794–799.

[12] KIM H J, JEE G I, LEE J G. Optimal Load Distribution for Tree Network Processors[J]. IEEE Transactions on. Aerospace and Electronic Systems.1996, 32(2): 607–612.

[13] BHARADWAJ V, GHOSE D, MANI V. Optimal Sequencing and Arrangement in Distributed Single-Level Networks with Communication Delays[J]. IEEE Transactions on. Parallel and Distributed Systems. 1994, 5:968–976.

[14] BHARADWAJ V, LI X, KO C C. On the influence of start-up costs in scheduling divisible load

中国知网 https://www.cnki.net

on bus networks[J]. IEEE Transactions on. Parallel and Distributed Systems, 2000, 11(12):1288–1305.

[15] SOHN J, ROBERTAZZI T G. Optimal load sharing for a divisible job on a bus network[J]. IEEE Transactions on Aerospace and Electronic Systems, 1996, 32(1):34–40.

[16] SURESH S, MANI V, OMKAR S N. The effect of start-up delays in scheduling divisible load on bus networks: an alternate approach[J]. Computers & Mathematics with Applications, 2003, 46(10-11):1545–1557.

[17] KIM H J. A novel optimal load distribution algorithm for divisible load[J]. Cluster Computing.2003, 6(1): 41–46.

[18] KIM H J, MANI V. Divisible load scheduling in single-level tree networks: optimal sequencing and arrangement in the nonblocking mode of communication. Computers & Mathematics with Applications, 2003, 46(10-11):1611–1623.

[19] SHANG M S. Optimal algorithm for scheduling large divisible workload on heterogeneous system[J]. Applied Mathematical Modeling. 2008, 32:1682–1695.

[20] 尚明生，孙世新，傅彦. 周期性的多趟可分任务调度算法研究[J].系统工程与电子技术，2007, 29(2): 294-299.

[21] BHARADWAJ V, GERASSIMOS B. Efficient Scheduling Strategies for Processing Multiple Divisible Loads on Bus Networks[J]. Journal of Parallel and Distributed Computing, 2002, 62(1): 132-151.

[22] 尚明生. 异构总线网络的可分负载优化调度算法[J].计算机工程，2005, 31(20):30-32.

[23] GHOSE D, KIM H J, KIM T H. Adaptive Divisible Load Scheduling Strategies for Workstation Clusters with Unknown Network Resources[J]. IEEE Transactions on Parallel and Distributed Systems,2005,16(10): 897-907.

[24] DROZDOWSKI M, LAWENDA M, GUINAND F. Scheduling Multiple Divisible Loads[J]. International Journal of High Performance Computing Applications, 2006, 20(1): 19-30.

[25] BEAUMONT O, LEGRAND A, ROBERT Y. Static Scheduling Strategies for Heterogeneous Systems[J]. Computing and Informatics , 2002, 21(4):413-30.

[26] BEAUMONT O, LEGRAND A, MARCHAL L, et al. Steady-State Scheduling on Heterogeneous Cluster: Why and How?[C].//Proceedings of the 18th International Parallel and Distributed Processing Symposium(IPDP'04), Los Alamitos CA: IEEE Computer Society Press, 2004: 171-178.

[27] LEGRAND A, MARCHAL L, ROBERT Y. Optimizing the steady-state throughput of scatter and reduce operations on heterogeneous platforms[C].//Proceedings-18th International Parallel and Distributed Processing Symposium, IPDPS 2004, Los Alamitos CA: IEEE

中国知网 https://www.cnki.net

Computer Society Press, 2004: 2485-2492.

[28] LEGRAND A, MARCHAL L, ROBERT Y. Optimizing the steady-state throughput of scatter and reduce operations on heterogeneous platforms[J]. Journal of Parallel and Distributed Computing, 2005, 65(12): 1497-1514.

[29] BHARADWAJ V, BARLAS G. Scheduling divisible loads with processor release times and finite size buffer capacity constraints in bus networks[J]. Cluster Computing, 2003, 6(1): 63-74.

[30] HU J, KLEFSTAD R. Scheduling divisible loads on bus networks with arbitrary processor release time and start-up costs: Xrmi[C]//Performance, Computing, and Communications Conference, 2007. IPCCC 2007. IEEE Internationa. IEEE, 2007: 356-364.

[31] BEAUMONT O, LEGRAND A, MARCHAL L, et al. Independent and Divisible Tasks Scheduling on Heterogeneous Star-shaped Platforms with Limited Memory[C]. //Proceedings of the 13th Euromicro Conference on Parallel, Distributed and Network-Based Processing (PDP'OS), Los Alamitos CA: IEEE Computer Society Press, 2005: 179-186.

[32] WOLNIEWICZ P. Divisible Job Scheduling in Systems with Limited Memory[D]. PH.D Thesis, Poznan University of Technology Press, 2003.

[33] WOLNIEWICZ P, DROZDOWSKI M. Processing Time and Memory Requirements for Multi-installment Divisible Job Processing[C]. //Parallel Processing and Applied Mathematics: 4th International Conference PPAM'O1, Berlin: Springer-Verlag, 2002:125-133.

[34] DROZDOWSKI M, WOLNIEWICZ P. Out-of-Core Divisible Toad Processing[J]. IEEE Transactions on Parallel and Distributed Systems, 2003, 14(10): 1048-1056.

[35] DROZDOWSKI M, WOLNIEWICZ P. Optimum divisible load scheduling on heterogeneous stars with limited memory[J]. European Journal of Operational Research, 2006,172(2): 545-559.

[36] LI X, BHARADWAJ V, KO C C. Divisible Load Scheduling on Single-Level Tree Networks with Buffers Constraints[J]. IEEE Transactions on Aerospace and Electronic Systems, 2000,36(4):1298-1307.

[37] BHARADWAJ V, YAO J. Divisible load scheduling strategies on distributed mufti-level tree networks with communication delays and buffer constraints[J]. Computer Communications, 2004,27(1): 93-110.

[38] YANG Y, CASANOVA H. A Multi-Round Algorithm for Scheduling Divisible workload Applications: Analysis and Experimental Evaluation[R]. Technical Report CS2002-0721，Dept. of Computer Science and Eng., University of California, San Diego, 2002.

[39] LAWENDA M. Multi-Installment Divisible Loads Scheduling[D]. PH.D Thesis, Poznan University of Technology Press, 2006.

[40] YANG Y, VAN DER RAADT K, CASANOVA H. Multiround Algorithms for Scheduling

中国知网 https://www.cnki.net

Divisible Loads[J]. IEEE Transactions on Parallel and Distributed Systems, 2005,16(11): 1092-1102.

[41] YANG Y, CASANOVA H. UMR: A Multi-Round Algorithm for Scheduling Divisible Workloads[C]. //Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'03), Los Alamitos CA :IEEE Computer Society Press, 2003: 24-32.

[42] CHOI K, ROBERTAZZI T G. An Exhaustive Approach to Release Time Aware Divisible Load Scheduling. International Journal of Internet and Distributed Computing Systems[J]. 2011, 1(2):40-50.

[43] BHARADWAJ V, WONG H M. Scheduling Divisible Loads on Heterogeneous Linear Daisy Chain Networks with Arbitrary Processor Release Times[J]. IEEE Transactions on Parallel and Distributed Systems, 2004, 15(3):273-288.

[44] JOHN HENRY HOLLAND. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence[M]. MIT press, 1992.

[45] FRASER A S. Simulation of genetic systems[J]. Journal of Theoretical Biology, 1962, 2(3): 329-346.

[46] BLICKLE T, THIELE L. A comparison of selection schemes used in genetic algorithms[J]. 1995.

[47] GOLDBERG D E, RICHARDSON J. Genetic algorithms with sharing for multimodal function optimization[C]//Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms. Hillsdale, NJ: Lawrence Erlbaum, 1987: 41-49.

[48] FONSECA C M, FLEMING P J. Genetic Algorithms for Multiobjective Optimization: Formulation Discussion and Generalization[C]//ICGA. 1993, 93: 416-423.

[49] POWELL D, SKOLNICK M M. Using genetic algorithms in engineering design optimization with non-linear constraints[C]//Proceedings of the 5th international conference on genetic algorithms. Morgan Kaufmann Publishers Inc., 1993: 424-431.

[50] GOLBERG D E. Genetic algorithms in search, optimization, and machine learning[J]. Addion wesley, 1989, 1989.

[51] COSTA G D, GELAS J P, GEORGIOU Y, et al. The green-net framework: Energy efficiency in large scale distributed systems. In: Mei A, ed. Proc. of the Int'1 Symp. on Parallel and Distributed Processing. Piscataway: IEEE Computer Society, 2009.1-8.

[52] 石海燕. 无线传感器网络可分负载调度算法研究[D].浙江工业大学, 2013.

[53] KARGER D, STEIN C, WEIN J. Scheduling Algorithms. http://theory.lcs.mit.edu/~karget/ Papers/ scheduling.ps.gz

[54] SIEGEL H J, ALI S. Techniques for mapping tasks to machines in heterogeneous computing

中国知网 https://www.cnki.net

systems.Journal of Systems Architecture,2000,46:627-639

[55] LIN W, LIANG C, WANG J Z, et al. Bandwidth-Aware Divisible Task Scheduling for Cloud Computing [J]. Software: Practice and Experience, 2014, 44(2) :163-174.

[56] 代亮. 无线传感器网络可分负载调度研究[D].西安电子科技大学, 2011.

[57] BEAUMONT O, CASANOVA H, Legrand A, et al. Scheduling Divisible Loads on Star and Tree Networks: Results and Open Problems[J]. IEEE Transactions on Parallel and Distributed Systems, 2005, 16(3):207-218.

[58] MURUGESAN G, CHELLAPPAN C. Multi-Source Task Scheduling in Grid Computing Environment using Linear Programming [J]. International Journal of Computer Science and Engineering, 2014, 9(1):80-85.

中国知网 https://www.cnki.net

中国知网 https://www.cnki.net

致谢

光阴荏苒，时光飞逝，二年半的研究生生活不仅增强了我独立解决问题的能力，也培养了我对学术研究的热情。

值此研究生论文定稿之际，首先需要感谢的是倾心向我传道授业解惑的博士生导师王宇平教授。在读研期间，王老师始终给予我悉心的指导，时时以他严谨求实的科研态度，高瞻远瞩的学术敏感和锐意进取的工作风范影响并激励着我。本文的选题和研究工作是在王老师和王晓丽师姐的热情帮助和悉心指导下完成的，从论文的立题到最终论文的完成，王老师和王晓丽师姐都给予了极大的关怀和鼓励，并提出了许多宝贵的指导意见。

深深的感谢养育我的父母，感谢你们对我的支持和理解。你们对我无限的关爱和默默的支持是我最大的精神动力。

谢谢一直以来支持的朋友们，是你们在我难过痛苦的时候陪我分担，在我开心幸运的时候陪我分享，我很庆幸自己能遇到你们，相聚即是缘，愿我们友谊长存，一辈子兄弟。虽然我们以后工作不在同一个地方，但是我坚信有缘的人还是会相逢的。

最后感谢所有的评审老师，谢谢您百忙之中抽空审阅我的论文。

中国知网 https://www.cnki.net

中国知网 https://www.cnki.net

中国知网 https://www.cnki.net

西安电子科技大学

XIDIAN UNIVERSITY

地址：西安市太白南路2号

邮编：710071

网址：www.xidian.edu.cn