摘　要

随着科技的发展，新一代的计算机，无论计算能力和计算速度都比旧的计算机优越。但人类对高性能计算的需求，也不断提高。除了增强处理器本身的计算能力外，并行处理是一种提高计算能力的有效手段。以前并行处理要采用昂贵的专用计算机，随着个人计算机及网络成本的下降，现已广泛用分布式网络计算机系统进行并行处理。本文是在 MPI 网络并行环境中，将二维热传导方程参数反演问题用并行遗传算法进行数值求解。

本文首先介绍本课题研究的背景与意义，然后介绍并行计算的基本理论、计算机机群系统和 MPI 消息传递机制。在此基础上，建立了基于 Linux 和 MPI 的 PC 机群实验环境。然后基于网络并行环境中并行算法的设计原则，结合遗传算法的并行性，对陶瓷/金属材料热物性反问题用并行遗传算法进行求解。由于并行遗传算法将并行计算机的高速并行性和遗传算法固有的并行性相结合，极大地提升了遗传算法的求解速度和质量。在主从式、细粒度和粗粒度这三类遗传算法并行化模型中，粗粒度模型以其较小的通讯开销和对种群多样化，获得了最广泛的应用。本文研究了并行遗传算法中不同迁移间隔和交叉点数对并行算法的性能影响，并对实验结果进行了比较和分析。最后总结了本文所做的工作，并指出本领域有待于进一步研究的问题。

本文总共分为六章,其内容如下:

第一章，主要介绍本课题研究的背景与意义及本文所做的主要工作。

第二章，主要介绍并行计算机的发展及分类，并行计算的基本理论。

第三章，讨论了遗传算法及其并行性,介绍了它的理论背景及算法描述。

第四章，介绍了 MPI 系统,详细的给出了在实际计算中所使用的 MPI 机群系统的配置。同时介绍了一些基于 MPI 的并行程序设计技巧。

第五章，给出了本文研究的热物性参数反问题模型的数值求解过程，对并行遗传算法的几种不同交叉点数及迁移间隔进行分析比较。

第六章，给出了本文的结论，并对下一步的工作做了展望。

本文得到了国家自然科学基金项目(批准号:60173046)的资助。

关键词： 热传导，反问题，并行计算，并行遗传算法，MPI

Abstract

With developments of technology, computing power and speed of new generation computers are much better than the former ones. However people's demands for high performance computing are increasing and infinite in some sense, so that in addition to enhancing the computing power of a processor, parallel processing is also an efficient way to enhance the computing power of a system. In the past, the parallel processing can only be performed on the expensive and special computers. Along with the cost of PCs and network descending, the distributed parallel computing concept is widely used in the parallel computing. This thesis focuses on determination of parameters in a two dimension heat conduction equation on the MPI network parallel environment, by solving an inverse problem using the parallel Genetic Algorithm.

Firstly this thesis introduces research background and significances related with the subject, then shows basic theory of the parallel computing, introduces the cluster concept and the message passing system of MPI, after that, discusses how to establishes a parallel computing environment based on the MPI and Linux; and then based on network background, integrates principle of parallel algorithm with parallel characteristic of parallel genetic algorithm (PGA), using the PGA to solve the thermophysical properties inverse problem of ceramic/metal combine material, The PGA combines high-speed parallel-ability of supercomputers with the inherent parallelity of GA, and improves greatly the efficiency and accuracy of GA s. Among master-slave, fine-grained and coarse grained parallel approaches, the coarse-grained model is most widely used for its little communication overhead and its diversifying of the population. This thesis studies the influence to performance of parallel algorithm of difference migration step and crossover points of PGA, analyzes and compares some experiment results. Finally, conclusions of this thesis and suggestions for further research are given.

This thesis has six chapters.

Chapter 1 introduces research background and significances related with the

subject and main work which has been done.

Chapter 2 introduces development and the classification of the parallel computer, discusses the basic theory of parallel computing.

Chapter 3 discusses the GA and its parallelity, introduces its theory background and algorithm description.

Chapter 4 introduces the MPI system; a detail configuration of MPI cluster system which this thesis has used is given. The techniques of MPI parallel programming also have been addressed.

Chapter 5 gives the detail solution process of thermophysical properties inverse problem. The influence to performance of parallel algorithm of difference migration step and crossover points of PGA has been analyzed.

Chapter 6 summarizes this thesis, and suggestions for future research are given.

This subject is supported by National Natural Science Foundation of China (NSFC Grant No. 60173046).

Key words: heat conduction, inverse problem, parallel computing, parallel genetic algorithm, MPI

第1章 绪 论

1.1 本课题研究的背景与意义

高温材料反问题方法的研究始于上世纪50年代，近几十年来，它发展很快，各国研究者根据不同对象的特点，采取各自的方法和技巧进行分析研究，但与正问题相比，毕竟是年轻的，还没有成熟到可以系统化为完整学科的地步[1, 2]。反问题的求解有广泛的应用价值。在大型航天器、新型核反应堆、大型超导发电机等许多无法测量从而要求逆推相应物性参数的工程问题中，在研究新材料时，往往需要考虑反问题。例如，Cornell 大学力学与空间机械学院的一组科学家讨论了合金定向固化过程的反问题并指出了一种反问题求解的面向对象编制程序的思路[3, 4]。对于实际工程问题，很难得到反问题的精确解，一般用数值方法进行求解。Jonas(1999)，Abdullah(1999)，Delaunay(1998)分析了反问题，他们认为无论是线性反问题或非性线反问题，其计算量远大于正问题的计算量，研究计算速度快的算法是有实际意义的。

并行计算能缩短计算时间，但并行计算机价格昂贵，一般单位很少购买并行计算机。以局域网为依托的机群计算，可以利用各单位已有的计算资源，使局域网增加新的功能，一网两用。一方面作为通常意义的网络，用于信息交换和资源共享；另一方面又可以用作并行计算[5, 6]。

分布式并行处理系统的实质是在逻辑关系上采用消息传递(Message Passing)方式进行并行计算；在物理结构上经过相应的通信接口把许多自治的处理器相连。它与计算机网络有许多相似相通之处。因此，这类系统上的并行算法易与实施机群计算的并行算法相互移植。近年来，随着计算机网络技术的发展，在局域网上开发并行系统进行机群计算有着巨大的潜在价值。基于机群计算的分布式并行算法将为大规模工程与科学计算开辟一个新的领域。

1.2 本文所做的主要工作

本文研究了在 PC 机群环境中实现并行遗传算法解热传导反问题的相关问

题。

论文首先研究利用现有计算机资源，构建小型的机群系统，建立基于 Linux 和 MPI 的实验环境。在此基础上，介绍了开发 MPI 并行程序的过程，并且在实验环境中测试了并行程序的性能。

本文对机群环境作的研究和探索，主要包括以下工作：

第一，对并行计算体系结构进行讨论，通过建立基于 Linux 和 MPI 的并行机群实验环境，探索可扩展 PC 机群系统的构建和实现方法。

第二，研究了消息传递模式下机群系统并行程序模型以及影响并行程序性能的因素；讨论了 MPI 程序设计的基本模式和方法，以及在 MPI 机群环境下开发并行程序的框架和过程。

第三，给出了机群 MPI 环境下遗传算法的并行程序实现。

第四，在建立的并行实验平台上，对并行遗传算法的几种不同影响因素做了实验和比较。

最后，根据理论研究和实际实验的结果，总结了在机群系统中利用并行遗传算法求解热传导反问题的可行性，以及分析并行遗传算法进行改进的思路。

第2章 并行计算基础

2.1 并行计算的意义和发展前景

并行计算（Parallel Computing），简单的讲，就是在并行计算机系统上所作的计算，它和常说的高性能计算（High Performance Computing）、超级计算（Super Computing）含义基本相同，因为任何高性能计算和超级计算总要使用并行技术。

并行计算机的发展基于人们在两方面的认识：第一，单机性能不可能满足大规模科学与工程问题的计算需求，而并行计算机是实现高性能计算、解决挑战性计算问题的唯一途径；第二，同时性和并行性是物质世界的一种普遍属性，具有实际物理背景的计算问题在许多情况下都可以划分成能够并行计算的多个子任务。针对某一具体应用问题，我们可以利用它们内部的并行性，设计并行算法，将其分解成为互相独立但彼此又有一定联系的若干个子问题，分别交给各台处理机，而所有的处理机按并行算法完成初始应用问题的求解[7]。例如，根据几十个常用应用软件的统计，60%—80%的标量计算可以被向量化，而90%左右的串行计算可以并行化[8]。

当今，计算科学已经和传统的两种科学，即理论科学和实验科学，并列成为第三门科学，它们彼此相辅相成的推动科学发展与社会进步。在许多情况下，或者是理论模型复杂甚至理论尚未建立、或者实验费用昂贵甚至实验无法进行，此时计算科学就成为求解问题的唯一或主要手段。计算科学极大的增强了人们从事科学研究的能力，大大地加速了把科技转化为生产力的过程，深刻地改变着人类认识世界和改变世界的方法和途径。计算科学的理论和方法，作为新的研究手段和新的设计与制造技术的理论基础，正推动着当代科学与技术向纵深发展。

人类对计算机性能的要求是无止境的，在诸如预测模型的构造和模拟、工程设计和自动化、能源勘探、医学、军事以及基础理论研究等领域中都对计算机提出了极高的具有挑战性的要求。例如，在制造业领域，工程计算和模拟如果可能的话必须在几秒或几分钟内完成。在设计环境中，如果进行一次模拟需

要两星期才能得到结果，这通常是不可能接受的。因为只有模拟完成时间足够短时，设计者方可高效地工作[9]。当系统变得更复杂时，就需要增加更多时间对系统进行模拟。有一些应用问题的计算对时间有特定的期限（最著名的要数气象预报），花两天时间来获得当地第二天精确的天气预报将使得这种预报毫无意义。某些研究领域，如对大型 DNA 结构建模以及进行全球天气预报均具有巨大挑战性问题。所谓巨大挑战性问题是指无法用当今计算机在合理的时间内完成求解的那些问题。

另一个需要巨大计算量的应用问题是预测太空中天体运动。每个天体由于万有引力而相互吸引，这些远距离的力可用简单公式加以计算，而每个天体的运动可以通过计算天体所受合力的计算加以预测。如果共有 N 个天体，则每个天体将总共需计算 N－1 个力，约 N^2 次计算。例如，银河系可能有 10^{11} 个星体，此时就需重复计算 10^{22} 次，即使使用一个仅需 $\log_2 N$ 次计算的高效的近似算法（但每次计算中含有更多的计算量），计算的总量仍异常巨大（$10^{11}\log_2 10^{11}$）。若在单处理器系统上运算将需要很长时间，即使每次计算仅需 1 微妙（10^{-6} 秒，这是非常乐观的估计，因为一次计算中含有多次乘法和除法），则使用 N^2 算法是，迭代就需要 10^9 年，而用 $N\log_2 N$ 算法时，一次迭代也需几乎是一年的时间。

最后，计算机运算处理速度的提高和存储容量的增大主要靠电子工程学的元件技术成果。随着处理器系统规模日益庞大、技术日益复杂，其运算处理速度的提高不论在理论上还是技术上都会受到限制。一是计算机电路中信息传输是由电子运动实现，传输速度约为光速的一半；二是元器件的物理尺寸受氢原子直径 dH 的限制不可能无限变小，使用硅材料的 VLSI 器件，其特征尺寸以 0.1 μm 为极限；三是器件的发热和冷却问题，如 1 亿次的 Cray-1 超级计算机用了几吨的液态氟进行冷却。据资料估计，单处理机的极限速度为每秒 $10^9 \sim 10^{11}$ 浮点运算，目前的技术水平已趋近这个极限。即使微电子技术的进步还有很大的潜力，如采用量子效应集成电路工艺生成的芯片，可比现在的硅芯片的速度快几百倍，但一个 CPU 满足不了多个 I/O 设备的处理速度要求，出现了信号的拥挤堵塞现象，即所谓的“冯.诺依曼隘口”，使得整个系统的性能降低。因此，并行计算就成为提高计算机系统速度的一个新思路[10]。

总之，并行计算是具有战略意义、影响深远的研究领域，将成为一个国家综合实力的重要指标之一，无论是科技、经济、社会的发展都越来越离不开高性能计算。它不仅是一种产业能力，更是国家尊严和国力的表现。正因为如此，

世界各国都争相发展高性能计算机技术，以图在科技和经济发展的较量中占得先机。

2.2 当代主要的并行计算机系统及其结构模型

大型并行机系统一般可分为 6 类：单指令多数据流机 SIMD（Single－Instruction Multiple-Data）；并行向量处理机 PVP（Parallel Vector Processor）；对称多处理机 SMP（Symmetric Multiprocessor）；大规模并行处理机 MPP（Massively Parallel Processor）；工作站机群 COW（Cluster of Workstation）和分布共享存储 DSM（Distributed Shared Memory）多处理机[10]。SIMD 计算机多为专用，其余的 5 种均属于多指令多数据流 MIMD（Multiple-Instruction Multiple-Data）计算机。但随着计算机的发展，曾经风行一时的向量机和 SIMD 计算机已退出历史舞台，而 MIMD 类型的并行机却占了主导地位。当代主流的并行机是可扩放的并行机，包括共享内存的对称多处理机（SMP）和分布存储的大规模并行处理机（MPP）和工作站机群（COW）。

2.2.1 对称多处理机 SMP

对称多处理机的结构示于图 2-1。SMP（Symmetric Multiprocessors）系统使用商品微处理器（具有片上或外置高速缓存），它们经由高速总线（或交叉开关）连向共享存储器。这种机器主要应用于商务，例如数据库、在线事务处理系统和数据仓库等。其结构具有如下特征：（1）对称性：系统中任何处理器均可访问任何存储单元和 I/O 设备；（2）单地址空间：单地址空间有很多好处，例如因为只有一个 OS 和 DB 等副本驻留在共享存储器中，所以 OS 可按工作负载情况在多个处理器上调度进程从而易达到动态负载平衡，又如因为所有数据均驻留在同一共享存储器中，所以用户不必担心数据的分配和再分配；（3）高速缓存及其一致性：多极高速缓存可支持数据的局部性，而其一致性可有硬件来增强；（4）低通信延迟：处理器间的通信可用简单的读/写指令来完成（而多计算机系统中处理器间的通信要多条指令才能完成发送/接收操作）[11]。

目前大多数商用 SMP 系统都是基于总线连接的，占了并行计算机很大市场，但是 SMP 也具有如下问题：（1）欠可靠：总线、存储器或 OS 失效均会造成系

统崩溃，这是 SMP 系统最大的问题；（2）可观的延迟：尽管 SMP 比 MPP 通信延迟要小，但相对处理器速度而言仍然相当可观（竞争会加剧延迟），一般为数百个处理器周期，长者可达千个指令周期；（3）慢速增加带宽：有人估计，主存和磁盘容量为每 3 年增加 4 倍，而 SMP 存储器总线带宽每 3 年只增加 2 倍，I/O 总线带宽增加速率则更慢，这样存储器带宽的增长跟不上处理器速度或存储容量的步伐；（4）不可扩放性：总线是不可扩放的，这就限制最大的处理器数一般不能超过 10。为了增大系统的规模，可改用交叉开关连接，或改用 CC-NUMA（高速缓存的一致性的非均匀存储器访问）或机群结构[12]。

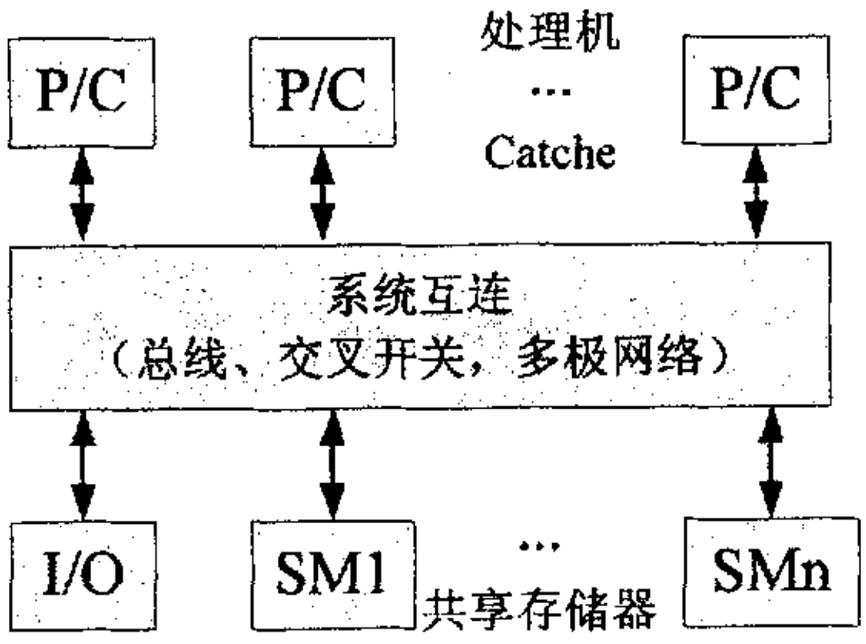


图 2-1 SMP 并行机结构模型

2.2.2 大规模并行处理机 MPP

MPP（Massively Parallel Processor）一般是指由成百上千处理器组成超大型（Very Large-Scale）计算机系统。所有的 MPP 均使用物理上分布的存储器，且使用分布的 I/O 也变多。它的结构示于图 2-2。其中每个节点有一个或多个处理器和高速缓存（P/C）、一个局部存储器、有或没有磁盘和网络结构电路（NIC：Network Interface Circuitry），它们均连向本地互连网络，而节点间通过高速网络（HSN：High Speed Network）相连。它具有如下特征：（1）处理器节点采用商用微处理器；（2）系统中有物理上的分布式存储器；（3）采用高通信带宽和低延迟的互连网络（专门设计和定制的）；（4）能扩放至成百上千个处理器；（5）它是一种异步的 MIMD 机器，程序系有多个进程组成，每个都有其私有地址空间，进程间采用传递消息相互作用。MPP 的主要应用是科学计算、工程模拟和

信号处理等以计算为主的领域。

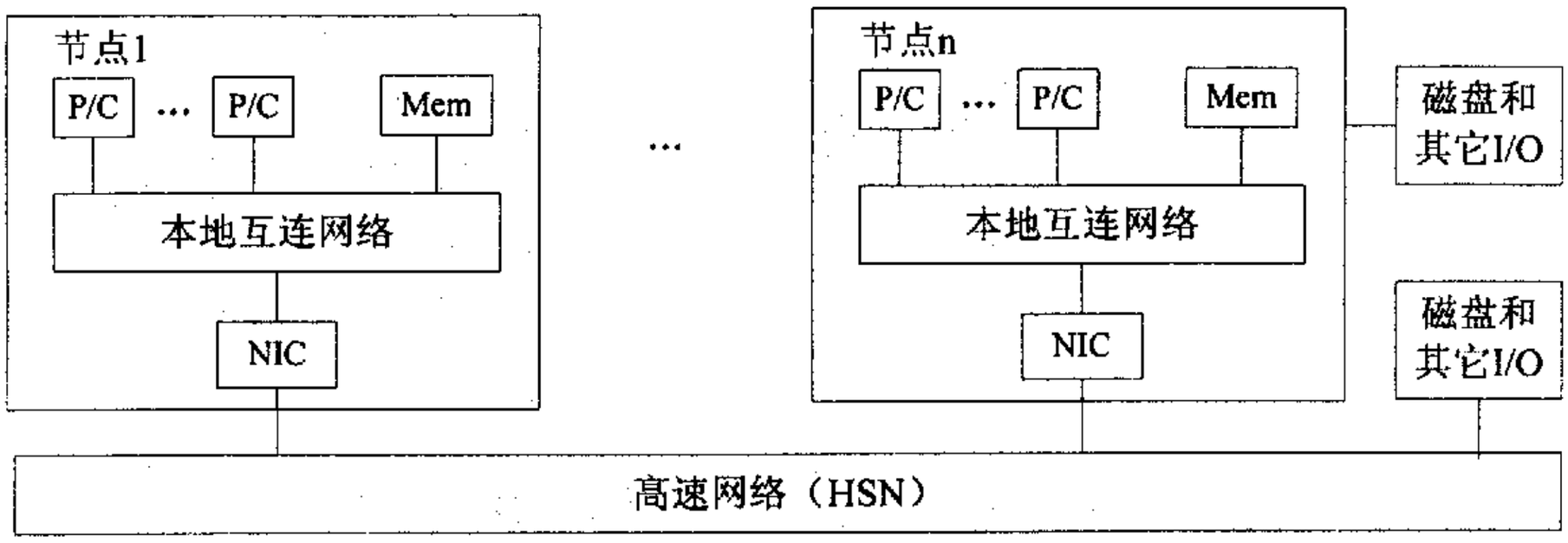


图 2-2 MPP 并行机结构模型

2.2.3 工作站机群 COW

随着工作站性能迅速提高和价格日益下降以及高速网络产品陆续问世，一种新型的并行计算系统便应运而生。这种系统将一群工作站用某种结构的网络互连起来，充分利用各工作站的资源，统一调度、协调处理，以实现高效并行计算[13]。如图 2-3 所示。它由工作站和互连网络两部分组成，互连网络可以是普通的 LAN（如以太网、令牌环和 FDDI 等），也可以是高速开关网络（如 ATM，交换式高速以太网等）。工作站是个广义的称呼，它可以是高档微机，甚至也可以是个对称多处理机 SMP。一个实用的 COW（Cluster Of Workstation）还应有一个高效的软件环境，包括操作系统、可由用户调用的通信原语库以及并行程序设计环境与工具等[14,15,16]。从用户、程序员和系统管理员的角度看，COW 相当于单一并行系统，感觉不到多个工作站的实际存在；从程序设计模式的角度看，它与 MPP 一样可采用面向消息传递的 SPMD（Single Program Multiple Data）编程方式，各个工作站均运行一个程序，但分别加载不同的数据，从而支持粗粒度的并行应用程序。

和前面所介绍的对称多处理机 SMP 和大规模并行处理机 MPP 相比，COW 在实用上具有一些明显的优点：

（1）投资风险小：用户在购置传统巨型机或 MPP 系统时，总是担心使用效率不高或性能发挥得不好，如果购置后在一定程度上确实出现此问题，就相当于浪费了大批资金，但 COW 不存在此问题，因为即使 COW 在技术上不够先进，

但每台高性能的工作站仍可照旧使用，不浪费资金；（2）编程方便：用户无需学用新的并行程序设计（如并行 C、C++和 Fortran 等），只需利用所提供的并行程序设计环境，在常规的 C，C++和 Fortran 等程序中相应的地方插入少量的几条原语，即可使这些程序在 COW 上运行，这一点最受用户欢迎；（3）系统结构灵活：用户将不同性能的工作站使用不同的体系结构和互连网络构成同构或异构的工作站机群系统，从而可弥补单一体系结构使用面窄的弱点，可更充分的满足各类应用的需求；（4）性能/价格比高：一般一台巨型机或 MPP 都很昂贵，而一台高性能工作站相对便宜 ，一个 COW 系统从浮点运算能力来看虽然有限，但一群工作站的总体运算性能可接近一些巨型机的性能，但价格却低了很多。（5）能充分利用分散的计算资源：当个人工作站处于空闲状态时，COW 可在空闲时间内给这些工作站加载并行计算任务，从而工作站资源可得到充分利用；（6）可扩放性好：用户可根据需要增加工作站的数目，以高带宽和低延迟的网络技术支持获得高的加速比，从而获得应用问题的高可扩性。

实现工作站机群需要解决的主要问题是通信性能和并行编程环境。因为组成 COW 的硬件环境中工作站的性能已相当高，且还在不断的提高，相对而言工作站性能不是关键问题；相反，负责数据通信的互连网络却是一项关键技术，因为 COW 系统中并行计算时各工作站之间需要经过互连网络交换数据，某些工作站上的程序所需要的数据也往往要通过网络获取或提交。如果网络通信延迟时间很长，再加上用于通信的软件开销（此部分不可忽略）可能就限制了 COW 技术对某些问题的适用性。近几年来网络技术发展很快，从而很好的支持理论工作站的互连[17]。

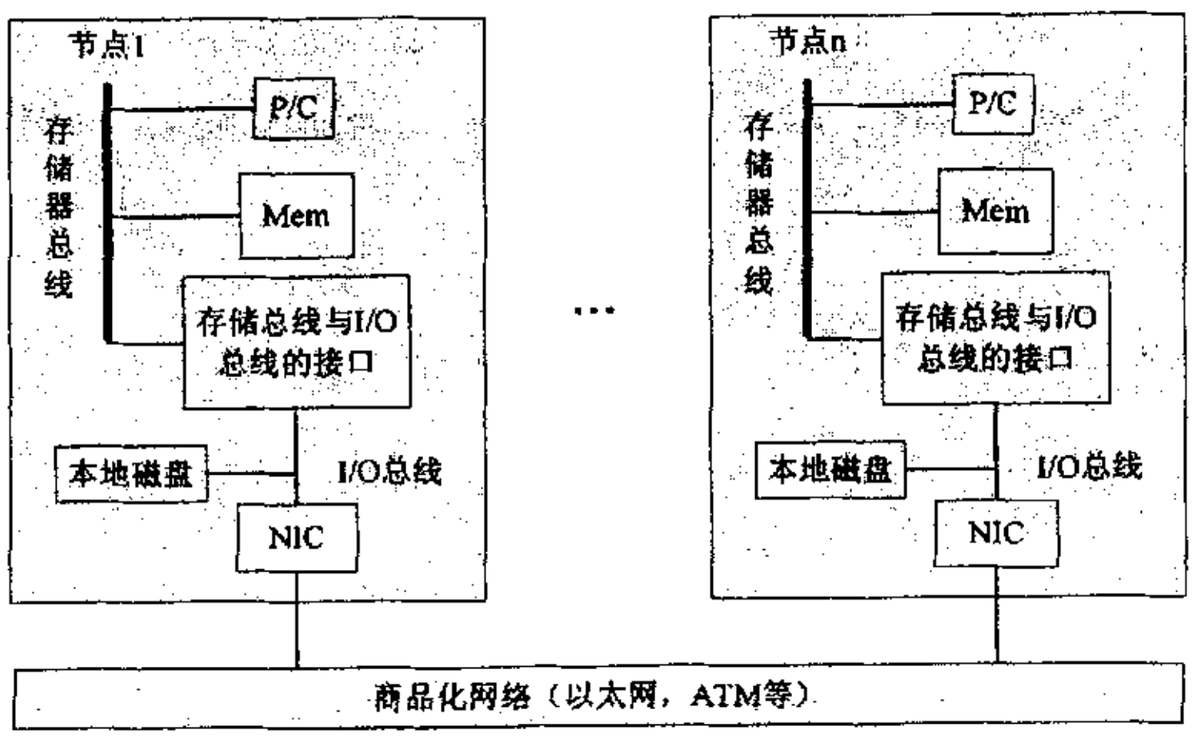


图 2-3 COW 并行机结构模型

伴随高速网络而产生的另一关键技术就是工作站到网络的主机接口网络接口的设计，它应尽量保持网络的传输速度与主机数据收发速度相匹配，其中增加高速缓存或采用 DMA 是可供选择的两种技术。网络接口使工作站可以利用网络传输数据，但也增加了通信延迟，它占用了工作站 CPU 时间，从而影响了 COW 的性能。一般一次通信时间延迟可如图 2-4 所示，其中 D 为操作系统调度时间，A 用于分配缓冲，C 用于拷贝数据到缓冲区，S 为启动发送/接收时间，T 为链路传输时间，F 用于定位中断源，X 用于从接收队列获取数据。一般减少延迟的方法有：（1）设计精简通信协议以减少数据移动的次数和协议处理时间；（2）采用主动消息（Active Message）携带数据处理命令以重叠计算和通信；（3）定制通信处理单元以消除通信对工作站 CPU 的依赖。

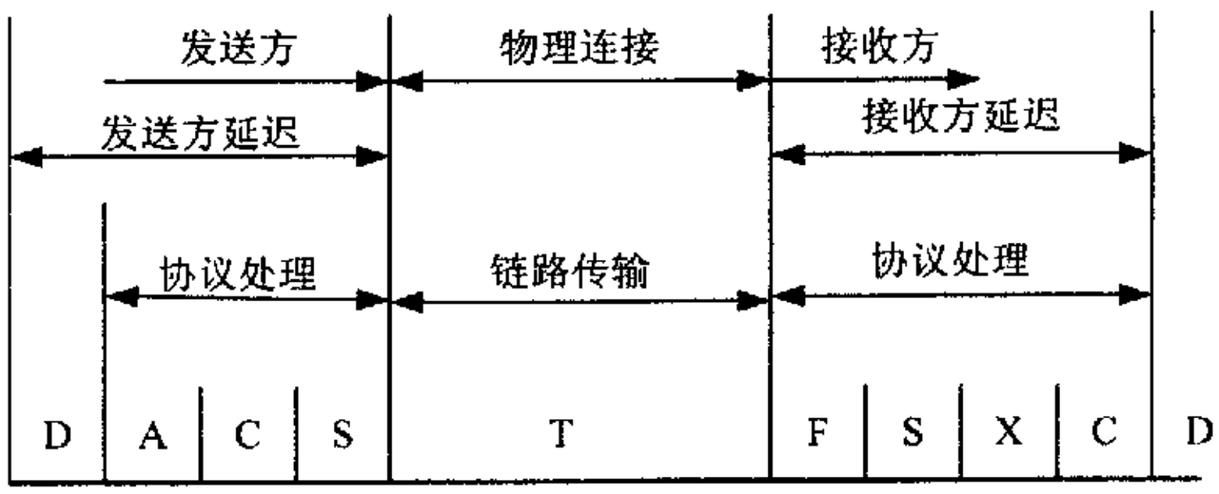


图 2-4 一次通信的时间延迟

另外，COW 要走向实用必须为用户提供一个良好的使用环境和一套完整的工具系统，主要包括：（1）并行编程环境，如一些通信原语库、多种编程语言（Express、PVM、Linda、P4、MPI）的支持以及并行编译器等；（2）可视化监视/调试器（典型的断点调试工具如 Cray 的 Mtdbx 和 TotalView 等）；（3）并行图形库（如基于 Express 的 Plotix 等）；（4）并行文件系统和数据库，以适用分布式事务处理的研究与开发[18]。

2.3 机群化的基本概念

前面我们多处提到机群这个概念，在本节中，我们将对有关计算机机群的基本概念作进一步的详细描述。

2.3.1 机群特征

机群是全体计算机（节点）的集合，这些计算机由高性能网络或局域网（LAN）物理地互连。典型情况下，每个计算机节点是一台 SMP 服务器、一台工作站或是一台 PC 计算机。更重要的是，所有机群节点必须能一起集体工作，如同一个单一集成资源，除了满足由交互用户单独地使用每个节点的协定任务之外。

下面描述有关机群的五个系统结构概念。

- **机群节点**：每个节点是一台完整计算机。这就意味着每个节点有自己的处理器、高速缓存、磁盘以及某些 I/O 适配器。此外在每个节点上驻留有完整、标准的操作系统。一个节点可拥有多台处理器，但只有一份 OS（操作系统）映像拷贝。
- **单一系统映像**：一个机群是一个单一计算资源。与此相反的是分布式系统（例如一个局域计算机网），在那里将节点作为单独资源。机群借助若干单一系统映像（SSI：Single System Image）技术，实现单资源概念。SSI 使机群变得更易使用和管理。目前大多数商品化可用机群还没到可完全实现 SSI 操作的程度。
- **节点间连接**：机群中的节点，通常用商品化网络，如以太网、FDDI、光纤通道以及 ATM 开关进行连接。此外使用标准协议以平滑节点间通信。
- **增强的可用性**：机群化提供了一个成本有效方法以增强一个系统的可用性，这是指一个系统可为用户使用的时间百分比。
- **更好的性能**：在若干服务领域中，机群应能提供更高性能。其中一个服务领域是将机群作为超级服务器使用。如果一个具有 n 个节点的机群中的每个节点能为 m 个客户服务，则该机群就能同时为 mn 个客户服务。另一个服务领域是机群通过分布式并行处理方法，用最短时间去完成一个大型作业的执行。

2.3.2 机群的优越性和难点

机群概念带来了许多好处，同时也带来了挑战。其中最重要的是能用性(usability)、可用性、可扩张性、可用利用率（available utilization）和性能/价格比。

将一批工作站用以太网简单的互连起来，并称其为机群，几乎不可能形成一个高可用性和高性能系统。必须要采用一些技术（大部分软件）以获得这些潜能。这些技术涉及到的方面有：

（1）**能用性** 由于机群中每个节点均是传统平台，故用户能在熟悉和成熟环境中开发和运行他们的应用程序。平台提供了所有功能很强的工作站编程环境工具，并允许几千个现有的（顺序）应用程序无需修改便可运行。因此一个机群可视为一个巨型工作站，它能为多个顺序用户作业运行提供大为增加的吞吐率并减少响应时间。

（2）**可用性** 术语可用性是指一个系统从事生产性使用的时间百分比。传统的整体系统，如主机和容错系统依靠昂贵的定制设计来获取高可用性。机群不使用定制组件，而使用廉价的商品化组件以提供含有大量冗余的较高可用性：

- **处理器和存储器**：机群有多个存储器和处理器部件，当某个部件失效时，其它的仍可使用，因此机群仍能运行。与此相反，SMP 机的共享存储器失效时，整个系统将崩溃。
- **磁盘阵列**：一个机群有多个局部磁盘。因此如果其中一个损坏，不会使整个系统崩溃。事实上，具有失效磁盘的节点可使用远程磁盘，因而可继续工作。
- **操作系统**：一个机群有多个操作系统映像，每个驻留在分离节点上。当一个系统映像崩溃时，其他节点仍能工作。与此相反，SMP 只有一个操作系统映像，驻留在共享存储器，若该映像被毁，则整个系统便将崩溃。

实现机群潜在可用性的关键在于有关软件技术。此外也需要使共享部件（如互连）具有高可用性的技术，否则它们将成为单失效点（single point of failure）。

（3）**可扩展性能** 一个机群的计算能力随节点增加而增加。其次，机群的可扩展性是群体可扩展性。关于这一点，通过机群和 SMP 的比较可得到最好的了解。SMP 是处理器可扩展系统，而机群扩展是多组件的，包括处理器、存储器、磁盘，甚至 I/O 部件。因为是松耦合，机群能扩展至几百个节点，而对于 SMP 来讲，要超过几十个节点就非常困难。

在 SMP 中，共享存储器（以及存储器总线）是瓶颈。当几个顺序程序运行于一个 SMP 上时，它们不得不竞争使用存储器，尽管这些程序是完全独立的。相同的程序集运行于机群时，不存在存储器瓶颈。每个程序可在一个节点上执

行，使用局部存储器。

对于这类应用，机群可提供更高的总体存储器带宽和减少存储器时延。机群的局部磁盘也聚集为大磁盘空间，可容易地超过集中式 RAID 磁盘空间。增强的处理、存储和 I/O 能力使得机群只需使用良好开发的，如 PVM 或 MPI 那样的并行软件包，就可求解大型应用问题。

(4)**性能/成本比** 机群能成本有效的获取上述优点。传统的 PVP(Parallel Vector Processor）超级计算机以及 MPP 的成本很易达到几千万美元。与此相比，具有相同峰值性能的机群价格则要低 1 到 2 个数量级。机群大量的采用商品化部件，它们的性能和价格遵循 Moore 定律，从而使机群的性能/成本比的增长速率远快于 PVP 和 MPP。

2.4 并行算法基础知识

2.4.1 并行算法定义与分类

算法是解题的精确描述，是一组有穷的规则，它规定了解决某一特定类型问题的一系列运算。并行计算是可同时求解的诸进程的集合，这些进程相互作用和协调动作，并最终获得问题的求解。并行算法就是对并行计算过程的精确描述[19]。

并行算法可以从不同的角度分类为数值计算并行算法和非数值计算并行算法；同步并行算法和异步并行算法；共享存储并行算法和分布存储并行算法。

数值计算是指基于代数关系运算的计算问题，如矩阵运算、多项式求值、线性代数方程组求解等。求解数值计算问题的算法称为数值算法（Numerical Algorithm）。科学与工程中的计算问题如计算力学、计算物理、计算化学等一般是数值计算问题。非数值计算是指基于比较关系运算的一类诸如排序、选择、搜索、匹配等符号处理，相应的算法也称为非数值算法（Non-numerical Algorithm）。非数值计算在符号类信息处理中获得广泛应用，如数据库领域的计算问题、海量数据挖掘等，近年来广泛关注的生物信息学主要也是非数值计算。

2.4.2 并行算法的复杂性

对于并行算法，除了研究所需的运行时间之外还需要研究其算法所需处理器数目以及研究两者在最坏情形下与问题规模n的变化关系。

- **运行时间t（n）**：它通常包含两部分的时间，一是数据从一个处理器经由互连网络或共享存储器到达另一个处理器所需要的选路时间t_r（Routing Time，即通信时间）；二是数据在一个处理器内作算术、逻辑运算所需的计算时间t_c。
- **处理器数目p(n)**：求解给定问题所需的处理器数目，可以固定大小，也可以随问题规模n变化而变化。在实际应用中，通常将处理器数目限制为$p(n)=n^{1-e}$，其中$0<e<1$。

2.4.3 并行算法性能评测

那么如何评价一个并行算法的优劣呢？显然，单纯考虑其所需的时间或者其所使用的处理器数目都是不全面的。一般地，需要综合考虑以下各项指标：

（1）并行算法的代价c（n）：将其定义为并行算法所需的运行时间t（n）和所需处理器数目p（n）的乘积，即：c（n）=t（n）* p（n）。如果求解一个问题的并行算法的执行代价在阶的意义上等于最坏情形下串行求解此问题所需的运行时间（步数），则称这样的并行算法是代价最佳的并行算法。

（2）加速比Sp（n）：设t_s（n）是求解某个问题的最快速的串行算法在最坏情形下所需的运行时间；t_p（n）是求解同一个问题的并行算法在最坏情形下所需的运行时间，则Sp（n）定义为：Sp（n）=t_s（n）/t_p（n），Sp（n）的意义是度量算法并行性对求解问题所需运行时间的改进程度。显然，若Sp（n）越大，并行算法就越好。在理想的情形下，Sp（n）=p（n），即p（n）台处理器去并行求解某个问题要比只用一个处理器去求解同一个问题快p（n）倍。但在绝大多数情况下，一方面所求解的原始问题不可能分解成n个子任务且每个子任务运行在一个处理器上所需时间为1/n；另一方面并行计算机在并行求解过程中，还需要进行数据交换或通信等方面的额外开销，因此实际上Sp（n）=p（n）是不可能的。事实上，由于任何一个并行算法都能在一台串行计算机上进行模拟，所以：t_s（n）$\leqslant$ p（n）t_p（n），从而有：1 $\leqslant$Sp（n）$\leqslant$p（n）。

（3）并行算法效率Ep（n）：虽然某个并行算法能获得好的加速，但是有时候其处理器利用率却可能很低，特别对于处理器数目不固定的情形更是如此。因此，仅衡量Sp（n）就断定一个并行算法是“好”的显然不全面。为此，人们引入并行算法效率Ep（n）的概念，将它定义为算法的加速比与处理器数目之比，即：Ep（n）＝ Sp（n）/p（n），0<Ep（n）≤1，Ep（n）可以度量并行计算机系统中处理器能力发挥的程度。

（4）并行算法伸缩性：给定处理器数目p，如果并行算法的效率Ep（n）随着问题规模（大小）n增加而单调递增，那么这种算法为可伸缩性的并行算法。当处理器数目增加时，可以通过增加问题规模使得伸缩性的并行算法仍能保持在理想的状态。

对于不同的并行计算机体系结构，为了保持可伸缩性的并行算法效率不变，问题规模随处理器数目增长的速度应是不同的。这个增长速度是处理器数目的函数，并称此函数为并行算法的等效率函数（Iso-Efficiency，简记为IsoE（p））。如果IsoE（p）属于指数型函数，那么相应并行算法和并行计算机体系结构的组合具有低伸缩性。如果IsoE（p）属于线性函数，那么我们说并行算法和并行计算机体系结构的组合具有高伸缩性。等效率函数是衡量并行算法性能的一个重要指标。

利用等效率可以通过简单的解析表达式来判断并行算法的伸缩性。设Tc表示并行算法所需的计算时间，Tt表示并行算法所需的额外时间（包括通信、同步和空闲等待时间等），则这时并行算法的加速比表示为：Sp（n）＝p*Tc/（Tc+Tt），而并行算法的效率表示为：Ep（n）＝Sp（n）/p（n）＝1/（1＋Tt/Tc）。这样，为了保持并行算法的效率不变，必须使Tc＝（E/（1－E））Tt。当求出Tc和Tt之后，通过简单的代数变换即可获得等效率曲线，从而可以据此判断并行的算法伸缩性。并行算法的可伸缩性问题对于网络并行计算环境显得尤为重要[20]。

2.4.4 并行计算模型

计算模型是对计算机的抽象。对于算法设计者而言，计算模型为设计、分析和评价算法提供了基础。冯.偌依曼机就是一个理想的串行计算模型，但现在还没有一个通用的并行计算模型。按照普渡（Purdue）报告，这种模型必须能

刻画并行计算机中那些对并行计算十分重要的能力，可以期望大多数专用并行计算机能提供这些能力。

这种抽象无须包含任何结构信息，如处理器数和处理器间通信结构，但它应能蕴式地表示并行计算的相对成本。这种模型应能对性能进行足够精确的表示而无须对实现细节进行过于显式描述。这种抽象模型向计算机系统结构设计者、软件开发人员、程序员以及算法设计者提供了很多好处。每一台并行计算机有一个自然模型，它能非常近似地反映自身的体系结构。

具体而言，并行计算模型的主要作用如下：

- 为并行算法的研究提供了一个比较通用的物质基础。
- 为并行算法的设计与分析提供了一种简单、方便的框架。
- 使得设计的并行算法具有一定的生命力，可以适用于多种具体的并行机。

迄今为止，已经有多种并行计算模型存在，如 PRAM 模型、LogP 模型、C3 模型、BDM 模型等，但其中的每一种只抽象了实际并行机的一个或几个方面，尚无一种适用于所有并行机。

第3章　遗传算法及其并行性

遗传算法是一类借鉴生物界自然选择和自然遗传机制的随机化搜索算法，由美国 J. Holland 教授提出，其主要特点是群体搜索策略和群体中个体之间的信息交换，搜索不依赖于梯度信息。它尤其适用于处理传统搜索方法难于解决的复杂和非线性问题，可广泛用于组合优化、机器学习、自适应控制、规划设计和人工生命等领域，是 21 世纪有关智能计算中的关键技术之一[21]。

3.1 遗传算法基本思想

遗传算法是从代表问题可能潜在解集的一个种群（population）开始的，而一个种群则由经过基因(gene)编码(coding)的一定数目的个体(individual)组成。每个个体实际上是染色体（chromosome）带有特征的实体。染色体作为遗传物质的主要载体，即多个基因的集合，其内部表现(即基因型)是某种基因的组合。它决定了个体的形状的外部表现，如黑头发的特征是由染色体中控制这一特征的某种基因组合决定的。因此，在一开始需要实现从表现型到基因型的映射即编码工作。由于仿照基因编码的工作很复杂，我们往往进行简化，如二进制编码。初代种群产生之后，按照适者生存和优胜劣汰的原理，逐代（generation）演化产生出越来越好的近似解。在每一代，根据问题域中个体的适应度（fitness）大小挑选(selection)个体，并借助于自然遗传学的遗传算子（genetic operators）进行组合交叉（crossover）和变异（mutation），产生出代表新的解集的种群。这个过程将导致种群像自然进化一样的后生代种群比前代更加适应于环境，末代种群中的最优个体经过解码（decoding），可以作为问题近似最优解[22]。设 P（t）和 C（t）分别表示第 t 代的双亲和后代，遗传算法的一般结构可描述如下[23]：

遗传算法过程

begin

t←0;

初始化 P（t);

```
评估 P（t）;
while 不满足终止条件 do
begin
    重组 P（t）获得 C(t);
    评估 C（t）;
    从 P（t）和 C（t）中选择 P（t+1）;
    t←t+1;
    end
end
```

3.2 遗传算法的应用步骤

遗传算法提供了一种求解复杂系统优化问题的通用框架，它不依赖于问题的领域和种类。对一个需要进行优化计算的实际应用问题，一般可按下述步骤来构造求解该问题的遗传算法。

第一步：确定决策变量及其各种约束条件，即确定出个体的表现型 X 和问题的解空间。

第二步：建立优化模型，即确定出目标函数的类型（是求目标函数的最大值还是求目标函数的最小值）及其数学描述形式或量化方法。

第三步：确定表示可行解的染色体编码方法，也即确定出个体的基因型 X 及遗传算法的搜索空间。

第四步：确定解码方法，即确定出由个体基因型 X 到个体表现型 X 的对应关系或转换方法。

第五步：确定个体适应度的量化评价方法，即确定出由目标函数值 f（x）到个体适应度 F（X）的转换规则。

第六步：设计遗传算子，即确定出选择运算、交叉运算、变异运算等遗传算子的具体操作方法。

第七步：确定遗传算法的有关运行参数，即确定出遗传算法的 M（群体大小，一般取为 20－100）、T（遗传运算的终止进化代数，一般取为 100－500）、Pc（交叉概率，一般取为 0.4-0.99）、Pm（变异概率，一般取为 0.0001-0.1）等参数。

由上述构造步骤可以看出，可行解的编码方法、遗传算子的设计是构造遗传算法时需要考虑的两个主要问题，也是设计遗传算法时的两个关键步骤。对不同的优化问题需要使用不同的编码方法和不同操作的遗传算子，它们与所求解的具体问题密切相关，因而对所求解问题的理解程度是遗传算法应用成功与否的关键。图 3-1 所示为遗传算法的主要构造过程示意图。

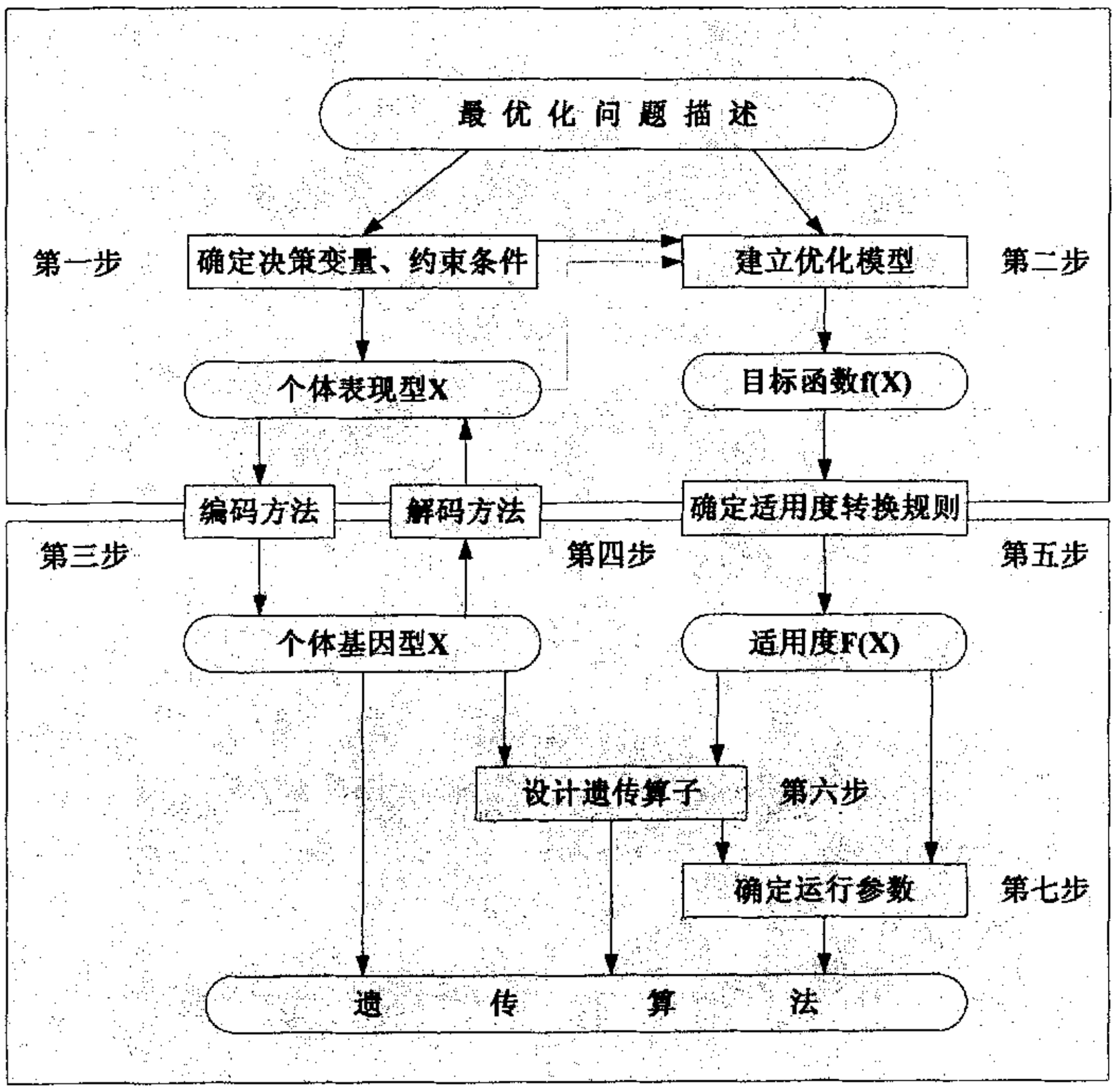


图 3-1 遗传算法的主要构造过程示意图

3.3 遗传算法的基本实现技术

在遗传算法的运行过程中，它不对所求解问题的实际决策变量直接进行操作，而是对表示可行解的个体编码施加选择、交叉、变异等遗传运算，通过这种遗传操作来达到优化的目的，这是遗传算法的特点之一。遗传算法通过这种

对个体编码的操作，不断搜索出适应度较高的个体，并在群体中逐渐增加其数量，最终寻求出问题的最优解或近似最优解。

3.3.1 编码方法

在遗传算法中如何描述问题的可行解，即把一个问题的可行解从其解空间转换到遗传算法所能处理的搜索空间的转换方法就称为编码。

编码是应用遗传算法时要解决的首要问题，也是设计遗传算法时的一个关键步骤。编码方法除了决定了个体的染色体排列形式之外，它还决定了个体从搜索空间的基因型变换到解空间的表现型时的解码方法，编码方法也影响到交叉算子、变异算子等遗传算子的运算方法。由此可见，编码方法在很大程度上决定了如何进行群体的遗传进化运算以及遗传进化运算的效率。一个好的编码方法，有可能会使得交叉运算、变异运算等遗传操作可以简单地实现和执行。而一个差的编码方法，却有可能会使得交叉运算、变异运算等遗传操作难以实现，也有可能会产生很多在可行解集合内无对应可行解的个体，这些个体经解码处理后所表示的解称为无效解。虽然有时产生一些无效解并不完全都是有害的，但大部分情况下它却是影响遗传算法运行效率的主要因素之一[24,25]。

迄今为止人们已经提出了许多种不同的编码方法。总的来说，这些编码方法可以分为三大类：二进制编码、浮点数编码、符号编码。下面我们从具体实现的角度出发介绍其中的几种主要编码方法。

（1）二进制编码

二进制编码方法是遗传算法中最常用的一种编码方法，它使用的编码符号集是由二进制符号 0 和 1 所组成的二值符号集{0，1}，它所构成的个体基因型是一个二进制编码符号串。

二进制编码符号串的长度与问题所要求的求解精度有关。假设某一参数的取值范围是{ Umin，Umax }，我们用长度为 l 的二进制编码符号串来表示该参数，则它总共能够产生 2^l 种不同的编码，若使参数编码时的对应关系如下：

$$\begin{aligned}
&00000000\cdots00000000=0 && \longrightarrow \text{Umin}\\
&00000000\cdots00000001=1 && \longrightarrow \text{Umin}+\delta\\
&\quad\vdots \quad\quad \vdots \quad\quad \vdots &&\\
&11111111\cdots11111111=2^l-1 && \longrightarrow \text{Umax}
\end{aligned}$$

则二进制编码的编码精度为：

$$\delta=\frac{\mathrm{Umax}-\mathrm{U\,min}}{2^{l}-1}$$

假设某一个体的编码是：

$$\mathrm{X}:\ b_l b_{l-1} b_{l-2}\ldots b_2 b_1$$

则对应的解码公式为：

$$x=U_{min}+(\sum_{i=1}^{l} b_i.2^{i-1}).\frac{U\max-U\min}{2^{l}-1}$$

二进制编码方法有下述一些优点：

- 编码、解码操作简单易行。
- 交叉、变异等遗传操作便于实现。
- 符合最小字符集编码原则。
- 便于利用模式定理对算法进行理论分析。

（2）浮点数编码方法

对于一些多维、高精度要求的连续函数优化问题，使用二进制编码来表示个体时将会有一些不利之处。

首先是二进制编码存在着连续函数离散化时的映射误差。个体编码串的长度较短时，可能达不到精度要求；而个体编码串的长度较长时，虽然能提高编码精度，但却会使遗传算法的搜索空间急剧扩大。例如，若使用二进制编码方法来处理一个含有 100 个决策变量的优化问题，其中每个决策变量的取值范围是[-250，250]，要求精度取小数点后 5 位小数，为达到这个精度要求，每个变量必须用 26 位长的二进制编码符号串来表示，这是因为：

$$2^{25}=33554432<\frac{500}{0.00001}=50000000<67108864=2^{26}$$

这样每个个体必须用 100 x 26 位长的二进制编码符号串表示。亦即此时遗传算法的搜索空间大约是 2^{2600}。在如此之大的搜索空间寻优肯定会使得遗传算法的运行性能相当差，甚至可能无法进行下去[26,27,28]。

其次是二进制编码不便于反映所求问题的特定知识，这样也就不便于开发针对问题专门知识的遗传运算算子，人们在一些经典优化算法的研究中所总结出的一些宝贵经验也就无法在这里加以利用，也不便于处理非平凡约束条件。

为改进二进制编码方法的这些缺点，人们提出了个体的浮点数编码方法。

所谓浮点数编码方法，是指个体的每个基因值用某一范围内的一个浮点数来表示，个体的编码长度等于其决策变量的个数。因为这种编码方法使用的是决策变量的真实值，所以浮点数编码方法也叫做真值编码方法。

浮点数编码方法有下面几个优点：

- 适合于在遗传算法中表示范围较大的数。
- 适合于精度要求较高的遗传算法。
- 便于较大空间的遗传搜索。
- 改善了遗传算法的计算复杂性，提高了运算效率。
- 便于遗传算法与经典优化方法的混合使用。
- 便于设计针对问题的专门知识的知识型遗传算子。
- 便于处理复杂的决策变量约束条件。

（3）符号编码方法

符号编码方法是指个体染色体编码串中的基因值取自一个无数值含义、而只有代码含义的符号集。这个符号集可以是一个字母表，如{A，B，C，D，…}；也可以是一个数字序号表，如{1，2，3，4，5，…}；还可以是一个代码表，如{A1，A2，A3，A4，A5，…}等等。

符号编码的主要优点是：

- 符合有意义积木块编码原则。
- 便于在遗传算法中利用所求解问题的专门知识。
- 便于遗传算法与相关近似算法之间的混合使用。

但对于使用符号编码方法的遗传算法，一般需要认真设计交叉、变异等遗传运算的操作方法，以满足问题的各种约束要求，这样才能提高算法的搜索性能。

3.3.2 适用度函数

遗传算法在进化搜索中基本上不用外部信息，仅用目标函数即适应度函数为依据。遗传算法的目标函数不受连续可微的约束且定义域可以为任意集合。对目标函数的唯一要求是，针对输入可计算出能加以比较的非负结果。这一特点使得遗传算法应用范围很广。

由于遗传算法中，适应度函数要比较排序并在此基础上计算选择概率，所

以适应度函数的值要取正值。由此可见，在不少场合，将目标函数映射成求最大值形式且函数值非负的适应度函数是必要的。

最优化问题分为两大类，一类为求目标函数的全局最大值，另一类为求目标函数的全局最小值。对于这两类优化问题，由解空间中某一点的目标函数值 f（x）到搜索空间中对应个体的适用度函数值 F（x）的转化方法如下：

1）对于求最大值的问题，作如下转换：

$$F(X)=\begin{cases} f(X)+Cmin, & \text{if } f(X)+Cmin>0 \\ 0, & \text{if } f(X)+Cmin\le 0 \end{cases}$$

Cmin 为一个相对较小的数，它可用下面几种方法之一来选取。

（1）预先指定的一个较小的数。

（2）进化到当前代为止的最小目标函数值。

（3）当前代或最近几代群体中的最小目标函数值。

2）对于求最小值的问题，作如下转换：

$$F(X)=\begin{cases} Cmax-f(X), & \text{if } f(X)<Cmax \\ 0, & \text{if } f(X)\ge Cmax \end{cases}$$

Cmax 为一个相对较大的数，它可用下面几种方法之一来选取。

（1）预先指定的一个较大的数。

（2）进化到当前代为止的最大目标函数值。

（3）当前代或最近几代群体中的最大目标函数值。

3.3.3 选择算子

从群体中选择优胜的个体，淘汰劣质个体的操作叫选择。选择算子有时又称为再生算子（reproduction operator）。选择的目的是把优化的个体(或解)直接遗传到下一代或通过配对交叉产生新的个体再遗传到下一代。选择操作是建立在群体中个体的适应度评估基础上的，目前常用的选择算子有以下几种。

1.适应度比例方法（fitness proportional model）

适应度比例方法是目前遗传算法中最基本也是最常用的选择方法。它也叫轮盘赌或蒙特卡罗（Monte Carlo）选择。在该方法中，各个个体的选择概率和其适应度值成比例。

设群体大小为 M，其中个体 i 的适应度值为 fi，则 i 被选择的概率 Psi 为

$$Psi=fi/\sum_{i=1}^{M} fi \quad (i=1,2,3,\cdots M)$$

2.最佳个体保存方法（elitist model）

该方法的思想是把群体中适应度最高的个体不进行配对交叉而直接复制到下一代中。此种选择操作又称复制(copy)。

此选择方法的优点是，进化过程中某一代的最优解可不被交叉和变异操作所破坏。但是，这也隐含了一种危机，即局部最优个体的遗传基因会急速增加而使进化有可能限于局部解。也就是说，该方法的全局搜索能力差，它更适合单峰性质的搜索空间搜索，而不是多峰性质的空间搜索。所以此方法一般都与其他选择方法结合使用。

3.期望值方法（expected value model）

期望值方法主要思想如下：

（1）计算群体中每个个体在下一代生存的期望数目：

$$M=fi/\bar{f}=fi/\sum fi/n$$

（2）若某个体被选中并要参与配对和交叉，则它在下一代中的生存的期望数目减去 0.5；若不参与配对和交叉，则该个体的生存期望数目减去 1。

（3）在（2）的两种情况中，若一个个体的期望值小于零时，则该个体不参与选择。

De Jong 曾对比了适应度比例选择法，最佳个体保存法和期望值法用于函数优化中的性能。对比实验表明，采用期望值法的遗传算法离线性和在线性都高于采用另外两种方法的遗传算法性能。

4.排序选择方法（rank-based model）

排序选择方法的主要思想是：对群体中的所有个体按其适应度大小进行排序，基于这个排序来分配各个个体被选中的概率。其具体操作过程是：

（1）对群体中的所有个体按其适应度大小进行降序排序。

（2）根据具体求解问题，设计一个概率分配表，将各个概率值按上述排列次序分配给各个个体。

（3）以各个个体所分配到的概率值作为其能够被遗传到下一代的概率，基于这些概率值用比例选择(赌盘选择)的方法来产生下一代群体。

由于排序选择方法的主要着眼点是个体适应度之间的大小关系。对个体适应度是否取正值或负值以及个体适应度之间的数值差异程度并无特别要求。

3.3.4 交叉算子

遗传算法中的所谓交叉运算，是指对两个相互配对的染色体按某种方式相互交换其部分基因，从而形成两个新的个体。交叉运算是遗传算法区别于其他进化算法的重要特征，它在遗传算法中起着关键作用，是产生新个体的主要方法。

交叉算子的设计和实现与所研究的问题密切相关，一般要求它既不要太多地破坏个体编码串中表示优良性状的优良模式，又要能够有效地产生出一些较好的新个体模式。另外，交叉算子的设计要和个体编码设计统一考虑。

交叉算子的设计包括以下两方面的内容：

（1）如何确定交叉点的位置?

（2）如何进行部分基因交换?

最常用的交叉算子是单点交叉算子。但单点交叉操作有一定的适用范围，故人们发展了其他一些交叉算子。下面介绍几种适合于二进制编码个体或浮点数编码个体的交叉算子。

单点交叉 单点交叉（One-point Crossover）又称为简单交叉，它是指在个体编码串中只随机设置一个交叉点，然后在该点相互交换两个配对个体的部分染色体。

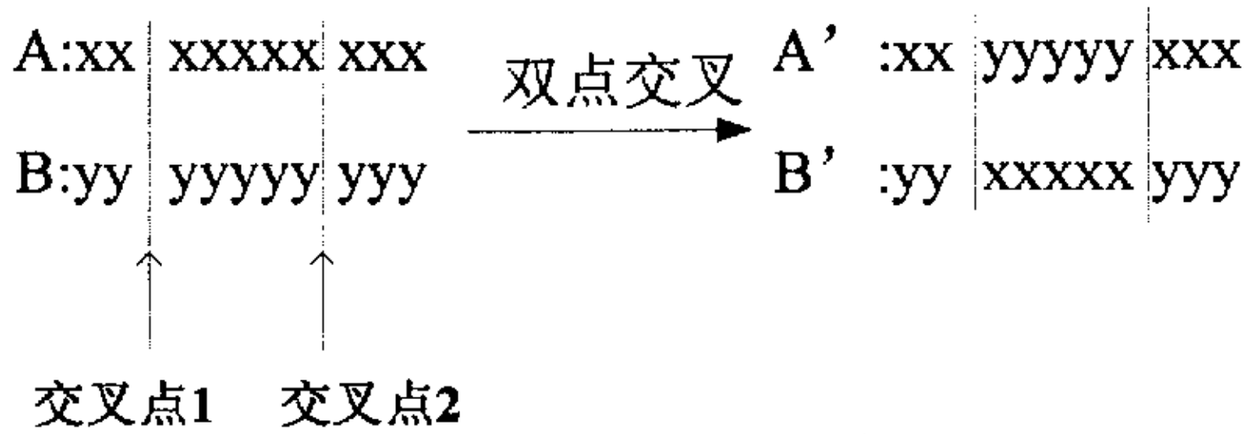
单点交叉的重要特点是：若邻接基因座之间的关系能提供较好的个体性状和较高的个体适应度的话，则这种单点交叉操作破坏这种个体性状和降低个体适应度的可能性最小。

双点交叉与多点交叉 双点交叉（Two-point Crossover）是指在个体编码串中随机设置了二个交叉点，然后再进行部分基因交换。双点交叉的具体操作过程是：

（1）在相互配对的两个个体编码串中随机设置两个交叉点。

（2）交换两个个体在所设定的两个交叉点之间的部分染色。

例如，双点交叉操作的示例如下：



将单点交叉和双点交叉的概念加以推广，可得到多点交叉（multi-point crossover）的概念。即多点交叉是指在个体编码串中随机设置了多个交叉点，然后进行基因交换。多点交叉又称为广义交叉，其操作过程与单点交叉和双点交叉相类似。

均匀交叉（Uniform Crossover） 是指两个配对个体的每一个基因座上的基因都以相同的交叉概率进行交换，从而形成两个新的个体。均匀交叉实际上可归属于多点交叉的范围，其具体运算可通过设置一屏蔽字来确定新个体的各个基因如何由哪一个父代个体来提供。

算术交叉（Arithmetic Crossover） 是指由两个个体的线性组合而产生出两个新的个体。为了能够进行线性组合运算，算术交叉的操作对象一般是由浮点数编码所表示的个体。

3.3.5 变异算子

遗传算法中的所谓变异运算，是指将个体染色体编码串中的某些基因座上的基因值用该基因座的其它等位基因来替换，从而形成一个新的个体。例如，对于二进制编码的个体，其编码字符集为{0、1}，变异操作就是将个体在变异点上的基因值取反，即用 0 替换 1，或用 1 替换 0；对于浮点数编码的个体，若某一变异点处的基因值的取值范围为{ Umin，Umax }，变异操作就是用该范围内的一个随机数去替换原基因值；对于符号编码的个体，若其编码字符集为{A，B，C，…}，变异操作就是用这个字符集中的一个随机指定的且与原基因值不相同的符号去替换变异点上的原有符号。

从遗传运算过程中产生新个体的能力方面来说，交叉运算是产生新个体的主要方法，它决定了遗传算法的全局搜索能力；而变异运算只是产生新个体的辅助方法，但它也是必不可少的一个运算步骤，因为它决定了遗传算法的局部

搜索能力。交叉算子与变异算子的相互配合，共同完成对搜索空间的全局搜索和局部搜索，从而使得遗传算法能够以良好的搜索性能完成最优化问题的寻优过程。

3.4 并行遗传算法

遗传算法固有的并行性和大规模并行机的快速发展，促使许多研究者开始研究遗传算法的并行化问题，研制数量更加接近自然物种的软件群体将成为可能。遗传算法与并行计算机的结合，能把并行机的高速性和遗传算法固有的并行性两者的长处彼此结合起来，从而也促进了并行遗传算法的研究与发展。

3.4.1 源于自然的并行性

在自然进化过程的任何时刻，总是同时有大量的物种在彼此独立地向前进化（当然，不同物种间的相互竞争使它们的进化过程会有不同程度的相互影响）。在同一物种内部，也是同时存在着大量的个体在通过自然选择、交配和基因突变而进化着。显然，自然界的进化过程本身就是一个并行过程。遗传算法来源于自然进化，与自然进化有密不可分的关系，很自然地也就继承了自然进化过程所固有的并行性。在遗传算法理论的形成过程中也充分体现了这一点。

3.4.2 遗传算法的并行化途径

遗传算法固有的并行性可通过不同的方法和途径去挖掘和实现，从而产生了不同类型的并行化方法。大体可分为三种计算拓扑模式[29]:主从式、粗粒度和细粒度。

1）主从式（master-slave）并行化方法

主从式模型(图 3-2)将选择、交叉、变异等全局操作交给主处理器(master)串行执行，而将适用度评估等局部操作交给从处理器网络（slaves）并行执行。主从式模型对计算机体系结构没有严格的要求，适合运行在共享存储和分布式存储的并行计算机上。

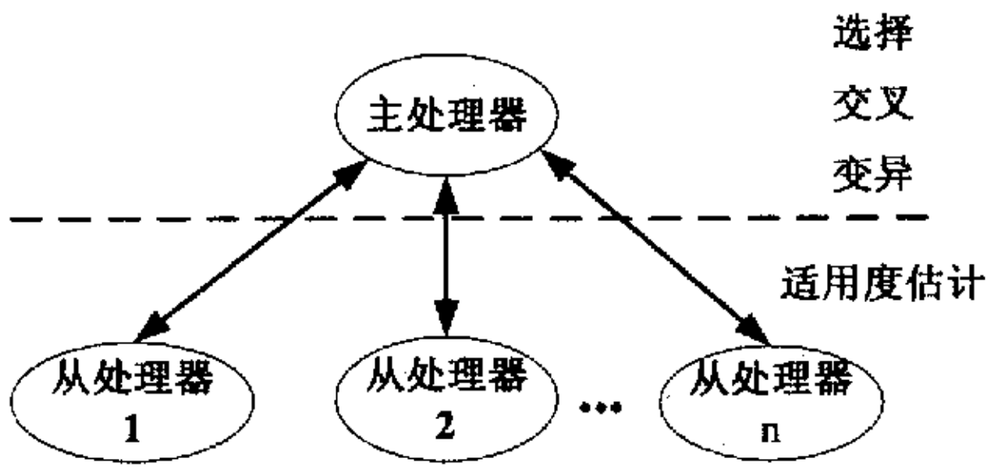


图 3-2 主从式模型

总的来说，主从式比较直观且容易实现，它并没有对标准遗传算法的框架结构作任何改动，所以不会影响其解决具体问题的效果。但是它不可避免地存在有负载不均衡的问题，而且通信量仍然很大，这使得它的效率不高，从而限制了它的实用性。

2）粗粒度模型（coarse-grain model）

将种群分成若干个子群并分配给各自对应的处理器，每个处理器不仅独立计算适应度，而且独立进行选择，重组交叉和变异操作，还要定期地相互传送适应度最好的个体，从而加快满足终止条件的要求。

粗粒度模型也称孤岛模型(island model)，基于粗粒度模型的遗传算法也称为分布式遗传算法（Distributed Genetic Algorithm, DGA），它是目前应用最广泛的一种并行遗传算法。Pettey 等已证明，综合考虑选择、交叉和变异的效应，粗粒度模型对遗传模式积木块的搜索次数的上限可用指数函数描述，这一结果从理论上表明该模型是有效的。粗拉度模型对并行系统平台要求不高，可以是松散耦合并行系统，特别适合基于 Transputer 的 MIMD 系统，并且效果很好，它主要开发群体之间的并行性。算法描述如下：

```
begin
   (1) 产生一个初始群体并将它划分成 p 个子群体
   (2) for i=1 to p par-do
       (2. 1) 初始化
       (2. 2) 评估第一代子群体的适用度
       (2. 3) while running do
          (2.3.1) for j=1 to n (generations) do
             (a) select parents
```

```
    (b) reproduce
    (c) mutate offspring
    (d) replace parents   /*形成新一代子群体*/
    (e) evaluate sub-population
end for
(2.3.2) select emigrants   /*选择要迁出去的个体*/
(2.3.3) do step(a) and (b) in parallel
          (a) send emigrants        /*发送要迁出的个体*/
          (b) receive immigrants    /*接收 迁入的个体*/
        endwhile
    endfor
end
```

其中，p 是处理器个数，也是子群体的个数。另外，发送要迁出的个体和接收迁入的个体这两个进程是并发执行的，这是为了避免通信过程中的死锁[28,29]。

3)细粒度模型（fine-grained model）

为种群中的每一个个体分配一个处理器，每个处理器进行适应度的计算，而选择、重组交叉和变异的操作仅在与之相邻的一个处理器之间互相传递个体中进行。

在细粒度模型中,通常处理器被连接成平面网格（grid），每个处理器上仅分配一个个体，选择和交叉只在网格中相邻个体之间进行（根据一个预先定义的领域结构来判定个体之间是否相邻）。这种细精度的并行遗传算法被称作扩散式（diffusion）或邻域模型。

3.4.3 并行遗传算法的迁移策略

迁移（migration）是并行遗传算法引入的一个新的算子，它是指在进化过程中子群体间交换个体的过程，一般的迁移方法是将子群体中最好的个体发给其它的子群体，通过迁移可以加快较好个体在群体中的传播，提高收敛速度和解的精度。与单种群相比只需要较少的个体评价计算工作量。因此，即使是采用单一处理器的计算机上以串行方式(伪并行)实现并行算法也能产生较好的结果。因此迁移算子的采用，使并行算法更适合于全局寻优，并且计算量较小。

以基于粗粒度模型的并行算法为例，其迁移策略可分为以下两种：

（1）一传多　每个处理器对应有若干个相邻处理器，每个处理器产生新一代个体后，都将自己最好的一个个体传送给其所有相邻处理器，并且接受来自相邻处理器的最好的个体，将这些个体与自己的个体同时考虑，淘汰适应度差的个体。

（2）一传一　考虑到染色体的多样性、每个处理器都将自己最好的个体仅传给与之相邻的一个处理器，同时增加两个参数：一是 send_rate 决定处理器之间通讯的频率，如 send_rate=3 时表示当遗传代数是 3 的倍数时，各处理器之间相互传送个体；二是 send_best 决定每次传送给最好个体的数目，如 send_best＝5 时表示每个处理器把最好的前 5 个个体传给各自的相邻处理器。

3.4.4 并行遗传算法的性能与参数选取关系

对于并行遗传算法，由于搜索的随机性，仅使用加速比这个指标来衡量其性能的优劣程度是比较困难的。比较现实的方法是，设计出一些具有不同几何特性的测试函数，通过它们来测量和统计达到最优点时的平均进化代数和平均计算时间，根据这些测试结果来比较不同的并行遗传算法的优劣。并行遗传算法的性能主要体现在收敛速度和精度两个方面，它们除了与迁移策略有关，还与一些参数选取的合理性密切相关，如遗传代数、群体数目、群体规模、迁移率和迁移间隔。

（1）遗传代数和群体规模

目前大多数研究者采用固定的遗传代数作为算法终止条件，遗传代数越大，求解精度越高，但时间开销越大。如何针对一个具体问题确定一个合理的遗传代数，仍没有一个很好的办法。群体规模是群体个体的数目，群体规模增大有利于解的精度和群体多样性的提高，但同时也增加了求解时间。

（2）迁移率与迁移间隔

将每次被迁移的个体数称为迁移率。迁移率的选取是一个很复杂的问题：由于被迁移者一般均是各子群体中的最优个体，所以迁移率较大，则有利于优良个体在整个群体中的传播和收敛速度的提高，但同时也会增加通信的开销，使加速比下降，也可能导致群体多样性的下降，不利于开发并行遗传算法在多个方向同时进行搜索的特征。应该针对具体问题选取合适的迁移率。

迁移间隔是指相邻两次迁移的时间间隔。迁移间隔小有利于子群体之间的融合，使得优良个体及时传播到所有子群体中，对群体的进化方向可以起到良好的指导作用，有利于提高解的精度和群体的收敛速度。但同时也会明显地增大通信及同步开销，不利于加速比的提高，而且某些优良个体在群体中的统治地位会产生不利于群体保持多样性的负面影响，使得整个群体类似于串行遗传算法中的随机交配群体(panmixis population)，不利于并行遗传算法发挥其并发搜索多个方向的特性，并有可能使群体进化陷入局部最小点。如果迁移间隔较大，则各子群体之间比较隔绝，其优点是降低了通信开销，提高了加速比，但同时会导致优良个体不能被及时传播，不能充分发挥其导向作用，不利于提高解的精度和收敛速度。

总之，如何获得较好的性能是并行遗传算法中的重要课题。选取合理的参数是十分困难的，这方面目前还没有指导性的实验结论。

第4章 MPI并行程序设计

4.1 MPI概述

MPI（Message Passing Interface）是消息传递并行程序设计的标准之一，当前通用的是MPI 1.1规范。正在制定的MPI 2.0规范除支持消息传递外，还支持MPI的I/O规范和进程管理规范。MPI正成为并行程序设计事实上的工业标准。MPI的实现包括MPICH、LAM、IBM MPL等多个版本，最常用和稳定的是LAM、MPICH[30,31]。

MPICH含三层结构，最上层是MPI的API，基本是点到点通信，和在点到点通信基础上构造的集群通信（Collective Communication）；中间层是ADI层（Abstract Device Interface），其中device可以简单地理解为某一种底层通信库，ADI就是对各种不同的底层通信库的不同接口的统一标准；底层是具体的底层通信库，例如工作站机群上的p4通信库、曙光1000上的NX库、曙光3000上的BCL通信库等。

MPICH的1.0.12版本以下都采用第一代ADI接口的实现方法，利用底层device提供的通信原语和有关服务函数实现所有的ADI接口，可以直接实现，也可以依靠一定的模板间接实现。自1.0.13版本开始，MPICH采用第二代ADI接口[32,33]。

对MPI的定义，可从以下三个方面来认识：

(1) MPI是一个库，而不是一门语言。MPI库可以被FORTRAN或C语言调用，它遵守语言语法中对过程或函数的调用规则。

(2) MPI是一种标准或规范的代表，而不特指某一个对它的具体实现。现在，几乎所有的并行计算机制造商都提供对MPI的支持，可以在网络上免费得到MPI在不同并行计算机上的实现，一个正确的MPI程序，可以不加修改的在所有并行机上执行。

(3) MPI是一种消息传递编程模型，并成为这种编程模型的代表和事实上的标准。MPI虽然很庞大，但是它的最终目的是服务于进程间通信这一目标[32]。

4.2 MPI 通信

并行程序设计时使用的几种主要通信模式：阻塞通信和非阻塞通信，组通信。MPI 中引入的虚拟拓扑结构的概念，使得对进程的分配易于管理。

4.2.1 阻塞通信

MPI 的标准发送及接收函数都是阻塞的。接收者在调用函数 MPI_Recv 后被阻塞，直至收到匹配的消息为止；但发送者的阻塞比较复杂。当发送者调用函数 MPI_Send 后被阻塞，在此期间，系统会先把发送者的发送缓冲区中的内容，拷贝到系统缓冲区中，由系统发送消息。发送者的操作不必等待发送完成，只要拷贝操作成功，发送者就可返回，甚至可重用存放发送缓冲区（不是系统缓冲区）。但是，如果系统缓冲区不足，或消息过长，令拷贝失败的时候，发送者将被阻塞到消息发送为止。如图 4-1 所示。

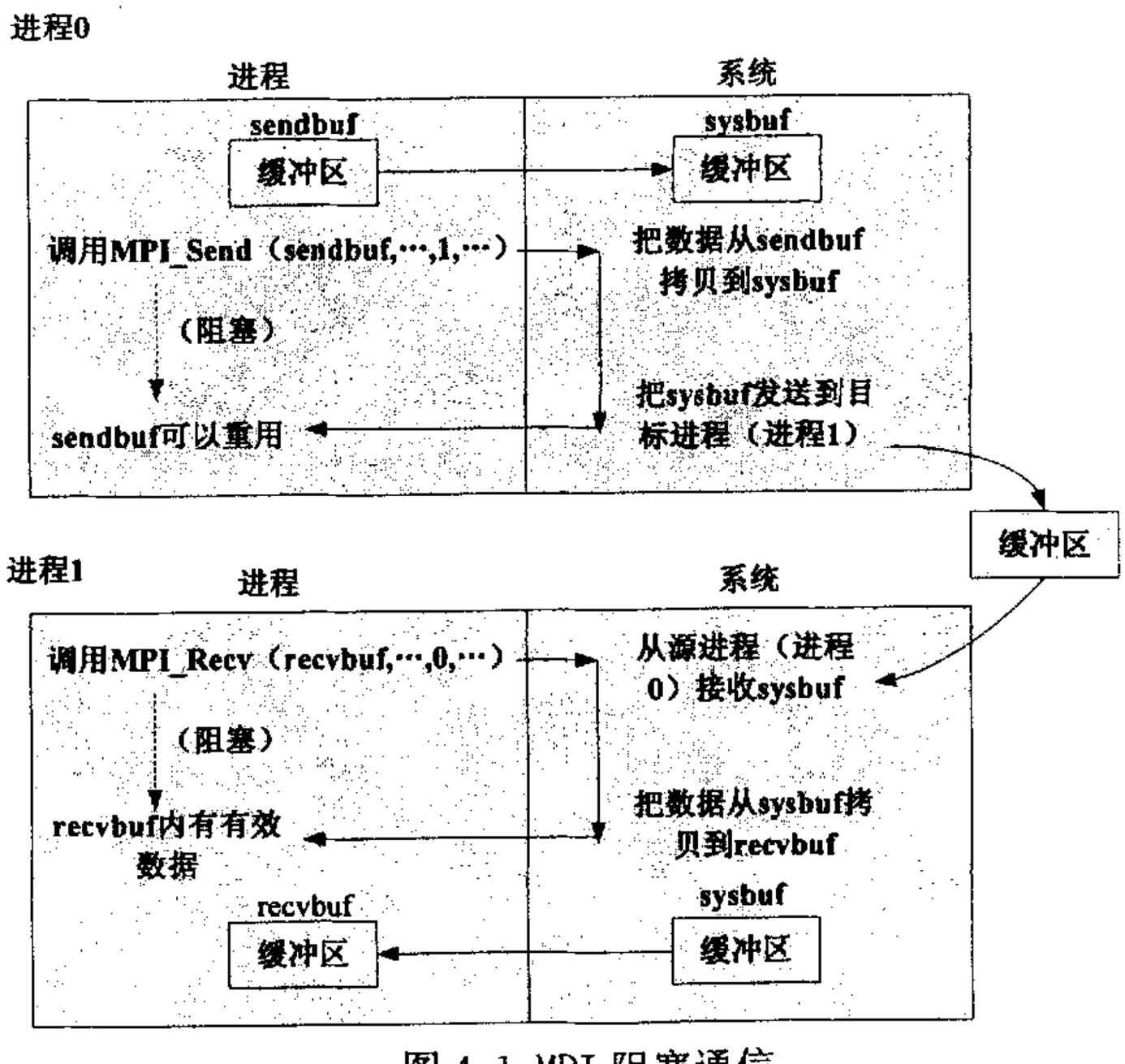


图 4-1 MPI 阻塞通信

4.2.2 非阻塞通信

非阻塞通信主要用于实现计算与通信的重叠[33]，从而提高整个程序的执行的效率，非阻塞通信还可实现一些特殊的控制功能，对于非阻塞通信，不必等到通信操作完全完成便可返回，该通信操作可以交给特定的通信硬件去完成，在该通信硬件完成该通信操作的同时，处理机可以同时进行计算操作，这样便实现了通信与计算的重叠。如图 4-2 所示。

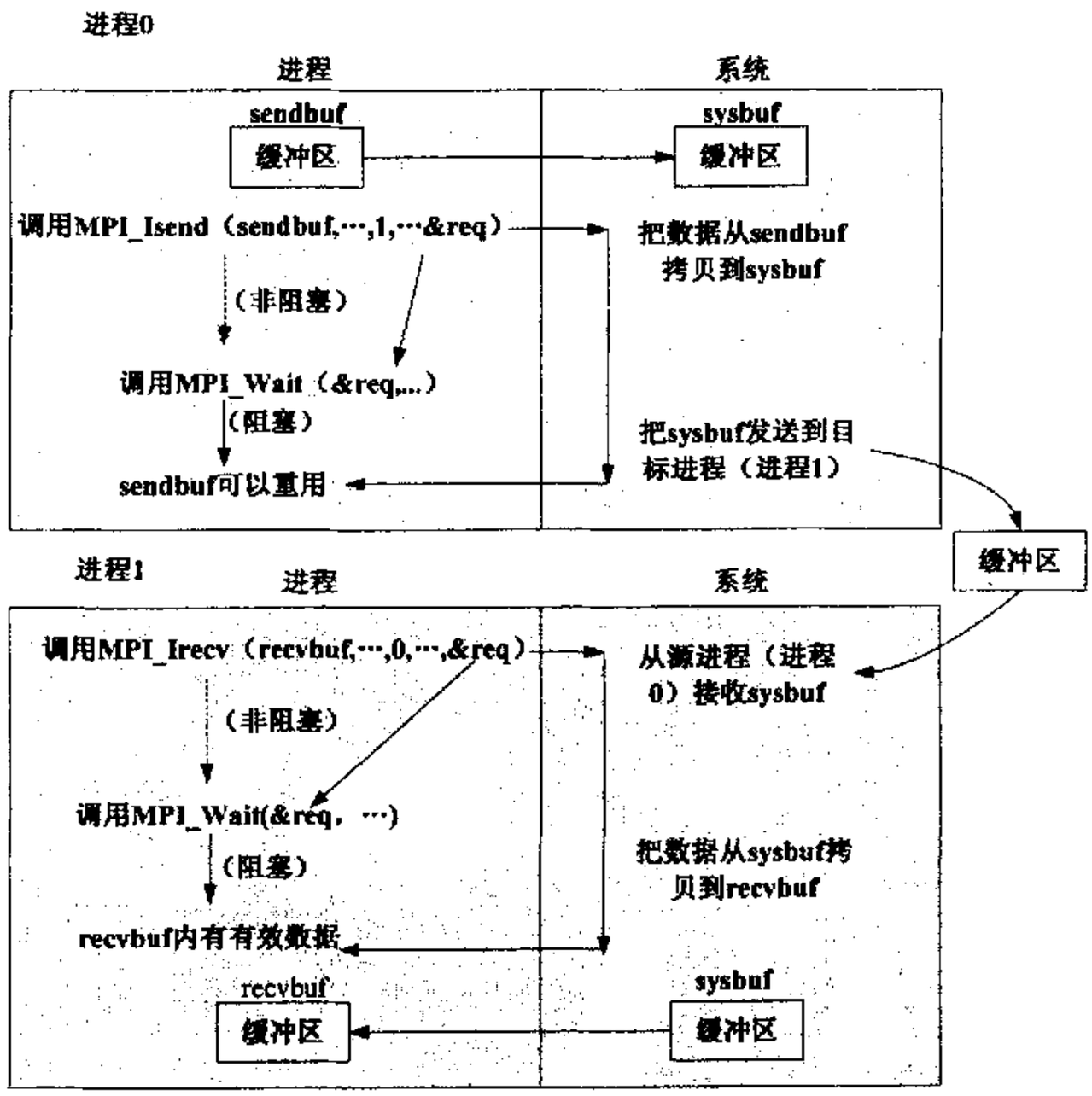


图 4-2 MPI 非阻塞通信

MPI 的非阻塞通信操作分开始及完成两个阶段，发送者调用函数 MPI_Isend 或接收者调用函数 MPI_Irecv 后，可执行其它计算。在发送或接收操作完成前，发送者不应该更改发送缓冲区的内容；接收者不宜使用接收缓冲区的内容。发送者（接收者）需要检测发送（接收）操作是否完成。

发送(接收)操作开始时，发送者(接收者)使用请求句柄(request handler)，MPI 通过检查请求来决定发送（接收）操作是否完成。发送者（接收者）调用函数 MPI_Test 后，会马上返回，根据返回值确定发送（接收）操作是否完成。若

发送者（接收者）调用函数 MPI_Wait，则发送者（接收者）会被阻塞直到发送（接收）操作完成才能返回。

4.2.3 组通信

MPI 组通信和点到点通信的一个重要区别，就在于它需要一个特定组内的所有进程同时参加通信，而不是像点到点通信那样只涉及到发送方和接收方两个进程。组通信在各个不同进程的调用形式完全相同，而不像点到点通信那样在形式上就有发送和接收的区别。

组通信一般实现三个功能：通信、同步与计算。通信功能主要完成组内数据的传输；而同步功能实现组内所有进程在特定的地点在执行速度上取得一致；计算功能要对给定的数据完成一定的操作。

4.2.4 MPI 通信模式

MPI 共有四种通信模式：标准通信模式（standard mode）、缓存通信模式（buffered－mode）、同步通信模式（synchronous-mode）和就绪通信模式（ready-mode）[34]。这几种通信模式对应于不同的通信需求，MPI 为用户提供功能相近的不同通信方式，为用户编写高效的并行程序提供了可能。

（1）标准通信模式

在 MPI 采用标准通信模式（图 4-3） 时，是否对发送的数据进行缓存是由 MPI 自身决定的，而不是由并行程序员来控制。如果 MPI 决定缓存将要发出的数据，发送操作不管接收操作是否执行，都可以进行，而且发送操作可以正确返回而不要求接收操作收到发送的数据[35]。

缓存数据是需要付出代价的，它会延长数据通信的时间，而且缓冲区也并不是总可以得到的，这样 MPI 也可以不缓存将要发出的数据，这样只有当相应的接收调用被执行后，并且发送数据完全到达接收缓冲区后，发送操作才算完成。对于非阻塞通信，发送操作虽然没有完成，但是发送调用可以正确返回，程序可以接下来执行其它的操作。

（2）缓存通信模式

当用户希望直接对通信缓冲区进行控制时，可采用缓存通信模式（图 4-4）。

在这种模式下由用户直接对通信缓冲区进行申请、使用和释放，因此，缓存模式下对通信缓冲区的合理与正确使用是由程序设计人员自己保证的。

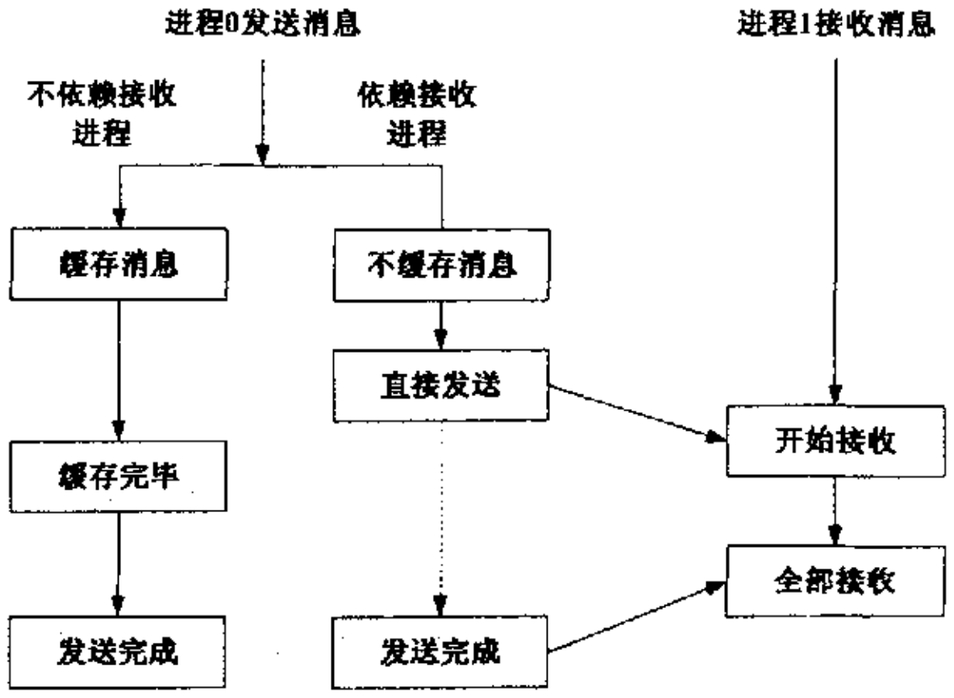


图 4-3 标准通信模式

缓存通信模式不管接收操作是否启动，发送操作都可以执行，但是在发送消息之前必须有缓冲区可用，这由用户保证，否则该发送将失败返回。对于非阻塞发送，正确退出并不意味者缓冲区可以被其它的操作任意使用，但阻塞发送返回后其缓冲区是可以重用的。

采用缓存通信模式时，消息发送能否进行及能否正确返回不依赖于接收进程，完全依赖于是否有足够的通信缓冲区可用，当缓存发送返回后，并不意味者该缓冲区可以自由使用，只有当缓冲区中的消息发送出去后，才可以释放该缓冲区。

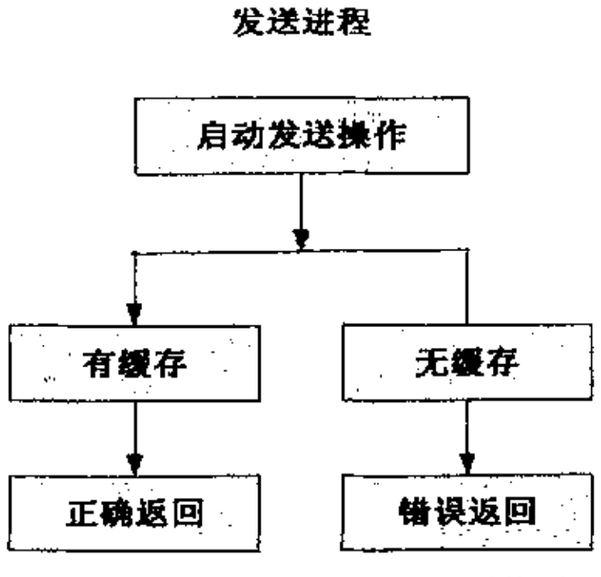


图 4-4 缓存通信模式

（3）同步通信模式

同步通信模式（图 4-5） 的开始不依赖于接收进程相应的接收操作是否已经启动，但是同步发送却必须等到相应的接收进程开始后才可以正确返回。因

此，同步发送返回后，意味着发送缓冲区中的数据已经全部被系统缓冲区缓存，并且已经开始发送。这样当同步发送返回后，发送缓冲区可以被释放或重新使用。

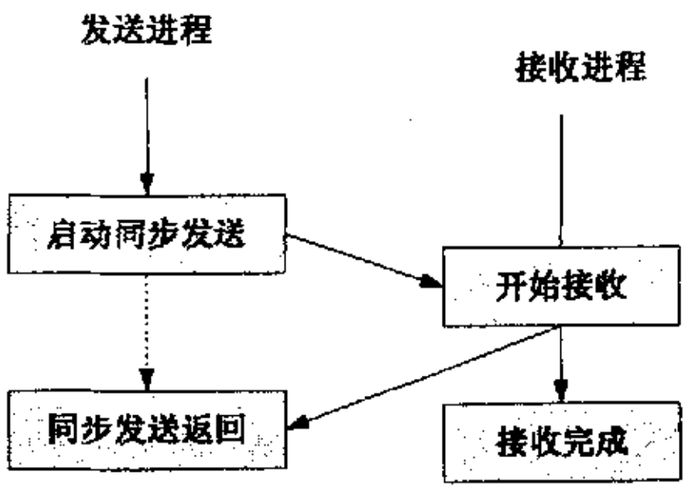


图 4-5 同步通信模式

（4）就绪通信模式

在就绪通信模式（图 4-6）中，只有当接收进程的接收操作已经启动时，才可以在发送进程启动发送操作，否则，当发送操作启动而相应的接收还没有启动时，发送操作将出错。对于非阻塞发送操作的正确返回，并不意味着发送已完成，但对于阻塞发送的正确返回，则发送缓冲区可以重复使用。

就绪通信模式的特殊之处就在于它要求接收操作先于发送操作而被启动，因此，在一个正确的程序中，一个就绪发送能被一个标准发送替代，它对程序的语义没有影响，而对程序的性能有影响。

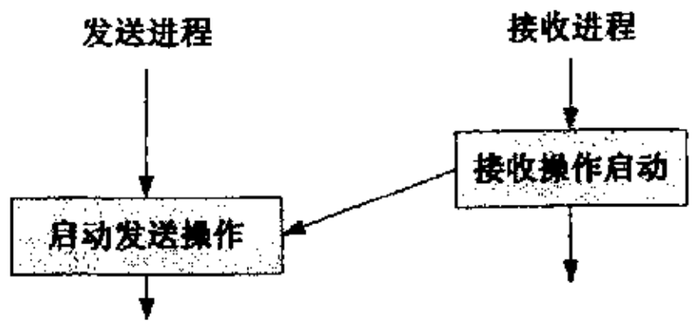


图 4-6 就绪通信模式

4.3 基于 Linux 环境的 MPI 配置

4.3.1 Linux 操作系统

Linux 是一套免费使用和自由传播的类 Unix 操作系统，是一个基于 POSIX 和 Unix 的多用户、多任务、支持多线程和多 CPU 的操作系统。它能运行主要的

Unix 工具软件、应用程序和网络协议。它支持 32 位和 64 位硬件。Linux 继承了 Unix 以网络为核心的设计思想，是一个性能稳定的多用户网络操作系统。它主要用于基于 Intel x86 系列 CPU 的计算机上。这个系统是由全世界各地的成千上万的程序员设计和实现的。其目的是建立不受任何商品化软件的版权制约的、全世界都能自由使用的 Unix 兼容产品。

Linux 是 1991 年由当时的芬兰赫尔辛基大学计算机系的学生 Linus Torvalds 独立完成的。此后，Linux 在因特网上广泛的传播，并由大批的遍布各地的系统软件设计专家共同对它进行改进和提高。

Linux 以它的高效性和灵活性著称。Linux 模块化的设计结构，使得它既能在价格昂贵的工作站上运行，也能够在廉价的 PC 机上实现全部的 Unix 特性，具有多任务、多用户的能力。Linux 是在 GNU 公共许可权限下免费获得的，是一个符合 POSIX 标准的操作系统。Linux 操作系统软件包不仅包括完整的 Linux 操作系统，而且还包括了文本编辑器、高级语言编译器等应用软件。它还包括带有多个窗口管理器的 X-Windows 图形用户界面，如同我们使用 Windows 一样，允许我们使用窗口、图标和菜单对系统进行操作。

Linux 有多种发行版本，其中 RedHat 是 Linux 的一个重要发行版本，因为其支持多种硬件平台、集成的软件丰富以及拥有友好的用户界面，使得其口碑甚佳，目前十分流行。RedHat 所有的软件包都是以 RPM(RedHat package Manager)方式包装的，这是一个高效的软件包管理系统，它提供了安装、升级、查询、和校验软件的可靠方法，可以让用户彻底卸载应用软件和系统部件。RedHat 集成的软件非常完整和精美，包括大量的 GNU 和自由软件。另外，RedHat 的系统安全性好，还能够提供 PAM，即快速的系统安全补丁。RedHat 目前的版本是 RedHat10.0，Linux 内核为 2.6。

4.3.2 LAM/MPI 的安装和配置

LAM（Local Area Multicomputer）是 Linux 平台下一个免费的 MPI 实现。它由 Ohio 州立大学开发，主要用于异构的网格计算并行系统。LAM/MPI 被认为具有“群集亲和性”，原因是它提供基于守护进程的进程启动或控制以及快速的客户到客户的消息传递协议。LAM/MPI 能够使用 TCP/IP 以及/或者共享内存来传递消息[36, 37]。

LAM 具有对 MPI-1 的完整实现(除了 LAM 不支持取消发送以外)，以及许多 MPI-2 的实现。服从其规则的程序源码可在 LAM 和其它 MPI 实现间移植。除了满足标准外，LAM/MPI 还提供了大量的监视能力来支持调试。监视有两个级别：在一个级别上，LAM/MPI 有允许在程序运行时间的任何时候给进程和消息状态拍快照的钩子。通讯者群组的成员、以及消息的内容。在第二个级别上，MPI 库能够生成一个通讯的累积记录，它可以在运行时间或事后反思时被视觉化[38]。

LAM 是共享软件，可以从网络上免费下载。具体步骤如下:

（1）LAM 的安装

根据机器配置的不同，从 http://www.lam-mpi.org/download/下载相应的 LAM 软件包。以 lam-6.5.9 为例，文件名为 lam-6.5.9-tcp.1.i386.rpm，如果下载的是源程序，则需经历解压，配置，编译，安装等步骤，为简化起见，我们以直接下载 RPM 包为例。6.5.9 可直接下载未压缩的 RPM 包。以 root 用户登陆，在 Linux shell 提示符下通过如下命令开始安装。

```
#rpm  - ivh lam-6.5.9-tcp.1.i386.rpm
```

可以在非 root 用户下执行以下命令来检查 LAM 是否安装成功，如果安装成功的话就会提示 LAM 启动成功。

```
$lamboot
```

（2）LAM 程序的编译、运行

LAM 程序的编译和在 Linux 下的普通程序的编译没有太大的区别，以 C 语言的程序为例，下一命令将程序 test.c 编译成可执行程序 test。

```
$mpicc -o test test.c
```

在执行上面程序之前，必须先执行 Lam 运行期环境，在主节点上执行即可。用以下命令。

```
$lamboot
```

然后用以下命令执行编译好的 LAM 可执行程序。

```
$mpirun  - np n test
```

这里的 n 是你想要启动的进程数，如 2，4，8 等等。mpirun 将在给定的节点机上启动指定数目的进程。当进程数目大于节点机数目时，mpirun 会在一些节点机上启动两个或更多进程。

（3）其他节点的加入

LAM 节点的配置文件保存在/etc/lam/lam_bhots.def 中,用 vi 打开该文件,在该文件的后面加入其它节点的机器名，例如 p1，p2 等。

为了加入机群环境中其它节点机，需要更改以下几项设置

a）关闭节点机上的防火墙

由于在执行并行计算时要远程使用其它节点的 CPU，需远程登陆其它机器。而出于对安全的考虑，防火墙屏蔽掉了这些权限。所以要在安装的时候选择不安装防火墙或者在安装后关闭防火墙。

b）将节点机的 IP 地址和机器名一一对应

在并行计算时要加入其它的计算节点，这时需要在前面提到的节点配置文件中加入其他的机器名，而 LAM 在查找这些机器时要到/etc/hosts 中查找该机器的对应 IP，所以要在/etc/hosts 文件中将各个计算节点的机器名和其 IP 地址一一对应，格式如下：

192.168.13.1 p1

192.168.13.2 p2

...

c）放权给计算用户

要允许计算用户调用该计算节点必须放权给该用户，例如在所有的计算节点上建立一个相同的用户 mpiuser，然后要在各个节点上的/etc/hosts.equiv 文件中添加可信任的机器名和用户名，格式如下：

p1 mpiuser

p2 mpiuser

...

d）修改远程调用的配置文件

在目录/etc/xinetd.d 下面的几个以 r 开头的配置文件如 rsh.conf、rlogin.conf、rexec.conf 是用来管理相应的远程调用的功能如 rsh、rlogin、rexec 的。要在这些配置文件中将其中的 disable=yes 改成 disable=no，以允许其它节点的远程调用。如果该目录下没有这些文件，则可能是在安装时相应的服务没有安装，找到相应的 rsh-server-0.17-10.i386.rpm（0.17-10 是版本号，根据 Linux 的版本有可能不同）包后安装即可。（注意，修改上述配置后需重启服务才能生效，命令：/etc/init.d/xinetd restart 可重启以上服务）。

4.4 基于MPI并行编程技巧

4.4.1 负载平衡

负载平衡的目的是缩小各个节点完成子任务的时间差以获得尽可能高的并行效率。负载平衡分为静态负载平衡和动态负载平衡两大类型。

静态负载平衡是在编译时针对用户程序的各种信息（任务的计算量和通信关系等）及并行系统本身的状况（网络结构、各节点计算能力等）对用户程序中的并行任务做出静态分配决策。

动态负载平衡是在程序运行过程中实现负载平衡的。它通过分析并行系统的实时负载信息，动态地将任务在各处理机之间进行分配和调整，以消除系统中负载分布的不均匀性。动态负载平衡的算法简单，实时控制，但增加了系统的额外开销。

比如在MPI程序设计时，有N个工作，P个进程，进程个数比工作件数少，采用最简单先来先服务的方式平衡负载。主进程先向各子进程分配第一批工作，各子进程进行并行地处理这些工作，工作结果返回主进程。当子进程出现空闲时，主进程就给它分配新工作，直至所有工作完成。主进程通知各子进程结束，整个程序结束[39]。

4.4.2 避免死锁

MPI_Send的阻塞可能导致死锁，可采用以下方法避免死锁：

1）把发送及接收的次序对换：凡进程序号为双数的先发送后接收，进程序号为单数的先接收后发送。

进程0

```
MPI_Send(buf1,bufsize,MPI_INT,
    1,tag1,comm.);
MPI_Recv(buf2,bufsize2,MPI_INT,
    1,tag2,comm,&status);
```

进程1

```
MPI_Recv(buf1,bufsize1,MPI_INT,
    0, tag1,comm.,&status);
MPI_Send(buf2,bufsize2,MPI_INT,
    0, tag2,comm) ;
```

```
⋮
MPI_Comm_rank(MPI_COMM_WORLD,&my_rank);
```

```
if((my_rank%2)==0){
    MPI_Send(buf1,bufsize1,MPI_INT,dest,tag1,comm);
    MPI_Recv(buf2,bufsize2,MPI_INT,source,tag2,comm,&status);
}
else{
    MPI_Recv(buf1,bufsize1,MPI_INT,source,tag1,comm,&status);
    MPI_Send(buf2,bufsize2,MPI_INT,dest,tag2,comm);
}
⋮
```

2）使用缓冲模式通讯。进程为 MPI 提供足够的系统缓冲区，让发送者返回。

```
⋮
MPI_Buffer_attach(userbuf, count*sizeof(int));
MPI_Bsend(buf1, bufsize1, MPI_INT, dest, tag1, comm);
MPI_Recv(buf2, bufsize2, MPI_INT, source, tag2, comm, &status);
⋮
```

3）使用非阻塞通讯。在阻塞通讯中导致死锁的另一个原因是发送及接收的顺序问题，除了把发送者及接收者的顺序匹配外，还可用非阻塞通讯，打破先完成一个操作才执行另一个操作的顺序，避免死锁。

```
⋮
MPI_Isend(buf1, bufsize1, MPI_INT, dest, tag1, comm, &request);
MPI_Recv(buf2, bufsize2, MPI_INT, source, tag2, comm, &status);
MPI_Wait(&request, &status);
⋮
或
⋮
MPI_Irecv(buf2, bufsize2, MPI_INT, source, tag2, comm, &request);
MPI_Send(buf1, bufsize1, MPI_INT, dest, tag1, comm);
MPI_Wait(&request, &status);
⋮
```

4.4.3 非阻塞通信

采用 MPI 非阻塞通信可同时进行通信及计算。先建立及提交持久通信请求。发送者及接收者继续执行其它工作，其间，发送者及接收者调用函数 MPI_Test 检测它们的操作是否完成。若其它工作已完成，但此操作尚未完成，只好调用函数 MPI_Wait 等它完成为止。

```
#include<mpi.h>
⋮
MPI_Init(&argc, &agrv);                          /*初始化 MPI 环境*/
MPI_Comm_rank( MPI_COMM_WORLD, &my_rank);   /*提取本进程号*/
if(my_rank == source)
{
   MPI_Send_Init(&buf, 1, MPI_INT, dest, tag, MPI_COMM_WORLD, &request);
}
else       /*my_rank = dest */
{
   MPI_Recv_Init(&buf, 1, MPI_INT, source, tag, MPI_COMM_WORLD, &request);
}
/*提交请求*/
MPI_Start(&request);
while(!flag && have_more_work_to_do){
      /*…执行其它作…*/
      MPI_Test(&request, &flag, &status);
}
/*其它工作完成，但此操作还未完成，只好等待此操作完成*/
if(!flag)                                        /*检查操作返回值*/
    MPI_Wait(&request, &status);                 /*等待 request 完成*/
/*后面工作*/
⋮
MPI_Finalize();                                  /*终止 MPI 环境*/
```

4.4.4 同步迭代

每个进程对不同部分的数据执行相同迭代计算，在开始下个迭代计算之前，进程必须同步。MPI 提供路障(Barrier)，对进程作全局同步。各进程在本次迭代完成计算后，调用一次路障，等待其它进程完成迭代计算后，才开始后面工作或开始下次迭代计算。

```
⋮
/*在迭代计算后*/
MPI_Barrier();
⋮
/*后面工作*/
⋮
```

4.4.5 设定错误处理

MPI 的预设错误处理为 MPI_ERRORS_ARE_FATAL,程序调用后非正常终止。另一种处理为 MPI_ERRORS_RETURN,调用后返回错误码。以下代码为函数 MPI_Bcast 设定错误处理为返回错误码。当出现错误时，程序先显示错误信息，然后终止。

```
/*错误处理变量*/
char error_message[MPI_MAX_ERROR_STRING];
int message_length;
⋮
/*初始化 MPI 环境*/
MPI_Init(&argc,&argv);
⋮
/*设定错误处理为返回错误码*/
MPI_Errhander_set(MPI_COMM_WORLD,MPI_ERRORS_RETURN);
⋮
error_code=MPI_Bcast(&data,…);
if(error_code != MPI_SUCCESS){
   MPI_Error_string(error_code,error_message,&message_length);
```

```
    fprintf(stderr, "Error in call to MPI_Bcast=%s\n",error_message);
    fprintf(stderr," Exiting function XXX\n");
    MPI_Abort(MPI_COMM_WORLD,-1);
}
⋮
/*终止 MPI 环境*/
MPI_Finalize();
```

第 5 章　热物性反问题的并行遗传算法实现

5.1 二维热传导方程简介

5.1.1 热传导反问题及其数值计算

计算传热学是研究用数值方法求解传热问题的一门科学。它可以理解为：根据所需求的实际问题建立合理的数学模型，利用数值方法将连续模型离散化，再通过用计算机高级语言编制的程序，以计算机作为工具来求解传热问题的，它是与工程实践密切结合的一门应用基础科学。是传热学学科领域中的一个重要分支，也是计算物理领域的一个分支，与计算流体力学，计算燃烧学等互相依存，互相促进[40,41]。

热传导方程是传热分析中最重要的一类方程，同样，热传导反问题（IHCP: Inverse Heat Conduction Problem）是热科学和技术研究的一个重要组成部分，同时也是在传热测量技术有关领域中开展较多研究的问题之一。在航天、核物理、冶金等工业研究领域有着广泛的应用背景[42]。正是因为其数值处理的困难性和重要的价值引起了许多数学家的关注，在 2000 年的日本长野反问题国际论坛会议上这一问题被列为第一个中心论题

现代工程技术，诸如能源、机械、动力、化工、冶金、交通、空调、制冷、电子、航天、建筑、材料、食品等专业领域中，传热学都起着愈来愈重要的作用。近十余年来，计算机技术和计算方法的发展，大大的推动了计算传热学的进展。计算传热学的主要优点是：能以较少的费用和较短的时间预示出有实用意义的研究结果。对投资大及周期长的实验研究课题来说，这个优点更为突出。当然，计算传热学不可能全部代替传热学的实验研究。在还未建立起精确的数学模型的地方，实验传热学是唯一能够给出研究结果的重要方法。事实上，计算传热学与实验传热学在很多地方是相辅相成的[43]。大量成功的算例表明，数值计算确实是一种研究和解决复杂实际传热问题的有效方法。将计算传热学和实验传热学相结合，不仅有助于实验方案的设计和改进，减少实验工作量和缩短实验周期，而且推动和促进了实验传热学的研究，并加深对物理概念和实验

机理的理解。同样，数值计算与解析求解相结合也是一种行之有效的工作方法。因此，在计算机辅助下的数值分析方法是一种有效和经济的传热学研究手段[44]。

本文研究的热物性反问题的高效分布式并行算法，是以内燃机缸体中的陶瓷/金属梯度隔热涂层这种复合材料的材料特性求解为工程背景；其应用前景是：如何通过实测的温度场分布，计算反演出这种功能梯度复合材料的材料特性，例如热传导系数随空间和温度的变化等。显然，功能梯度材料的热传导系数是一种多维、非线性、时变函数。本课题解决了二维、非线性、时变功能梯度材料的热传导系数反演求解问题，提出了一种高效的并行求解方案。本文用遗传算法来寻求热物性反问题数学模型的各个参数的近似值；用正问题的求解来对反问题演化结果进行验证和误差估计。在设计解决方案时，用虚拟边界预测法（VFB）加速正问题的分布式计算，并用不同的预测方法来加速预测过程和抑制过预测现象，从而加速对反问题演化结果的验证和误差估计。根据误差估计重新进行反问题的演化求解，直到误差估计处于要求的区域为止。

大量国内外资料表明，到目前为止，并行遗传算法用于一维常（变）系数热传导方程，二维稳态常（变）系数热传导方程求解的研究有了一定的成果，也有一些成功的算法实现，但对类似本文研究的二维非稳态变系数数学模型采用并行遗传算法求解还未见报道。本文的创新之处在于利用前一个国家自然科学基金已经取得的正问题求解的研究成果，把并行遗传算法和虚拟边界预测法相结合，提出了一种在高速局域网上分布求解热物性反问题的高效途径。

5.1.2 二维热传导方程的数学模型

对传热问题进行数值计算，和用其它理论求解方法一样，只有当实际的传热问题可以给出数学描述时，才能进行理论预测。因此，对给定的传热问题，首先是要写出它的控制方程和定解条件，有了这个前提，才能利用正确的数值方法，借助于数字计算机，得到反映传热过程内涵的数值结果，获得其几何参数，初始状态，工质的物性、流动状况和边界条件等对系统内的温度分布及其变化规律，边界处的热流或换热系数等的影响关系。传热问题的数学描述，一般都基于待求系统所遵循的守恒定律或平衡原理，即满足能量守恒[45,46,47]。

首先列出直角坐标系表示的二维导热控制方程及其边界条件：

$$\frac{\partial u}{\partial t}=\frac{\partial}{\partial x}(k1\frac{\partial u}{\partial x})+\frac{\partial}{\partial y}(k2\frac{\partial u}{\partial y}) \qquad 0<x，y<1，t>0 \qquad (5-1)$$

由于在我们研究的陶瓷/金属模型中，高度 H 远大于厚度 L，也即（H＞＞L），且认为陶瓷/金属梯度隔热涂层的导热系数只在厚度方向有变化，即 x 方向，因此有 k1=k2=k。这样，整个方程可写成$\frac{\partial u}{\partial t}=\frac{\partial}{\partial x}(k\frac{\partial u}{\partial x})+\frac{\partial}{\partial y}(k\frac{\partial u}{\partial y})$，加上边界条件，得数学模型如下：

$$\begin{cases}\dfrac{\partial \mathrm{u}}{\partial \mathrm{t}}=\dfrac{\partial}{\partial x}(\mathrm{k}\dfrac{\partial u}{\partial x})+\dfrac{\partial}{\partial y}(\mathrm{k}\dfrac{\partial u}{\partial y}) & 0<x,y<1,t>0; \\ \mathrm{u}(\mathrm{x},\mathrm{y},0)=0, & \\ \mathrm{u}(0,\mathrm{y},\mathrm{t})=\sin(2\pi t)\sin(\pi y), & \\ u(1,y,t)=0,u(x,0,t)=0,u(x,1,t)=0. & t>0\end{cases} \qquad (5-2)$$

该方程是描述一类新型复合材料中温度周期变化的问题，其中 u 表示温度，k 表示该材料的热传导系数。

当热传导系数随温度变化时，式（5-2）是非线性导热问题，在工程上，一般近似认为材料热传导系数是温度的二次函数，在我们研究的复合模型中，热传导系数还是厚度（即 x）的函数，根据经验有 $k=k(x,u)=(A-x^2)(B-Cu+Du^2)+x^2(E+Fu+Gu^2)$。k 中 A，B，C，D，E，F，G 为未知参数，令$\bar{\theta}=(A,B,C,D,E,F,G)$。假定一点 p 为测点，该点的温度历程为 Tm(L,t)。因此，如果为 k 中未知参数赋值后，可求解(5-2)式得到 p 点处的温度计算值的历程 Tc。显然，Tm(ti)与 Tc（ti）（i=1,2,…,N；N＝总测量点数）之间存在差别，因此，可选取目标函数为

$$J(\bar{\theta})=\| Tm-Tc \|^2=\frac{1}{\mathrm{N}}\sum_{i=1}^{N}[Tm(ti)-Tc(ti,\bar{\theta})]^2 \qquad (5-3)$$

因此，整个辨识问题转化为在给定约束条件下求$\bar{\theta}$使$J(\bar{\theta})$达极小的非线性优化问题。

实际上，不管是设计还是实际工程（或试验）问题，都能利用先验知识得到使 A，B，C，D，E，F，G 具有实际意义的上、下界A^l、A^u、B^l、B^u、C^l、C^u、D^l、D^u、E^l、E^u、F^l、F^u、G^l、G^u。从而原问题转化为以定解方程组（5-2）和紧约束式$A^l\le A\le A^u$，$B^l\le B\le B^u$，$C^l\le C\le C^u$，$D^l\le D\le D^u$，$E^l\le E\le E^u$，

$F' \le F \le F''$, $G' \le G \le G''$ 为约束条件，而式（5-3）定义的指标 J 达到最小为目标的优化问题。

正问题求解是反问题求解的基础，正问题求解精度和速度对反演结果和反演计算量有着重大影响。对于热传导系数随温度变化这种非线性温度场，有多种解法，这里采用迭代法求解。假定我们所求的 k 值具有如下形式：

$$k=(1-x^2)(0.1-0.01u+0.001u^2)+x^2(1.0+0.1u+0.01u^2)$$

方程(5-2)可以被改写为：

$$u_t = k(u_{xx}+u_{yy}) + k_x u_x + k_y u_y$$

$$或者：\quad u_t = k(u_{xx}+u_{yy})+G$$

其中：$G=G(x,y,t)=k_x u_x + k_y u_y$

对 k 求导可以得到：

$$k_x=2x(0.9+0.11u+0.009u^2)+[(1-x^2)(0.002u-0.01)+x^2(0.1+0.02u)]u_x$$

$$k_y=[(1-x^2)(0.002u-0.01)+x^2(0.1+0.02u)]u_y$$

这样方程(5-2)可以写为：

$$(4+\rho)u_{ij}^{(n+1)} = u_{i-1,j}^{(n+1)} + u_{i+1,j}^{(n+1)} + u_{i,j-1}^{(n+1)} + u_{i,j+1}^{(n+1)} + dij \qquad (5-4)$$

其中：

$$\rho = \frac{h^2}{k\nabla t} \qquad d_j = \rho u_j^{(n)} + \frac{h^2}{k}G$$

$$i, j=0, 1, 2, \cdots, M; \qquad n=0, 1, 2, \cdots$$

$$u_{0j}^{(n)} = \sin(2\pi t^{(n)})\sin(2\pi y_j) \qquad u_{Mj}^{(n)} = 0$$

$$u_{ij}^{(0)} = 0 \qquad u_{i0}^{(0)} = 0 \qquad u_{iM}^{(0)} = 0$$

上面的公式中 ∇t 表示时间步长，h 表示空间步长，比如在 1×1 的计算区间，如果计算的点为 480×480 的话，则 h=1/480。

对于某一热传导系数 k，由（5-2）式可惟一确定一个温度场分布，首先假定一个$\bar{\theta}$的值，利用这个假定的$\bar{\theta}$可求得选定的测点的温度历程 Tc（ti，$\bar{\theta}$），用这组值和实测的该点的温度历程 Tm（ti），直到满足要求精度$\|Tm-Tc\|\leq\varepsilon$的非线性温度场的解，从中可以看出，整个计算过程就是一个正问题，反问题的交替过程，其程序框图如图 5-1 所示。

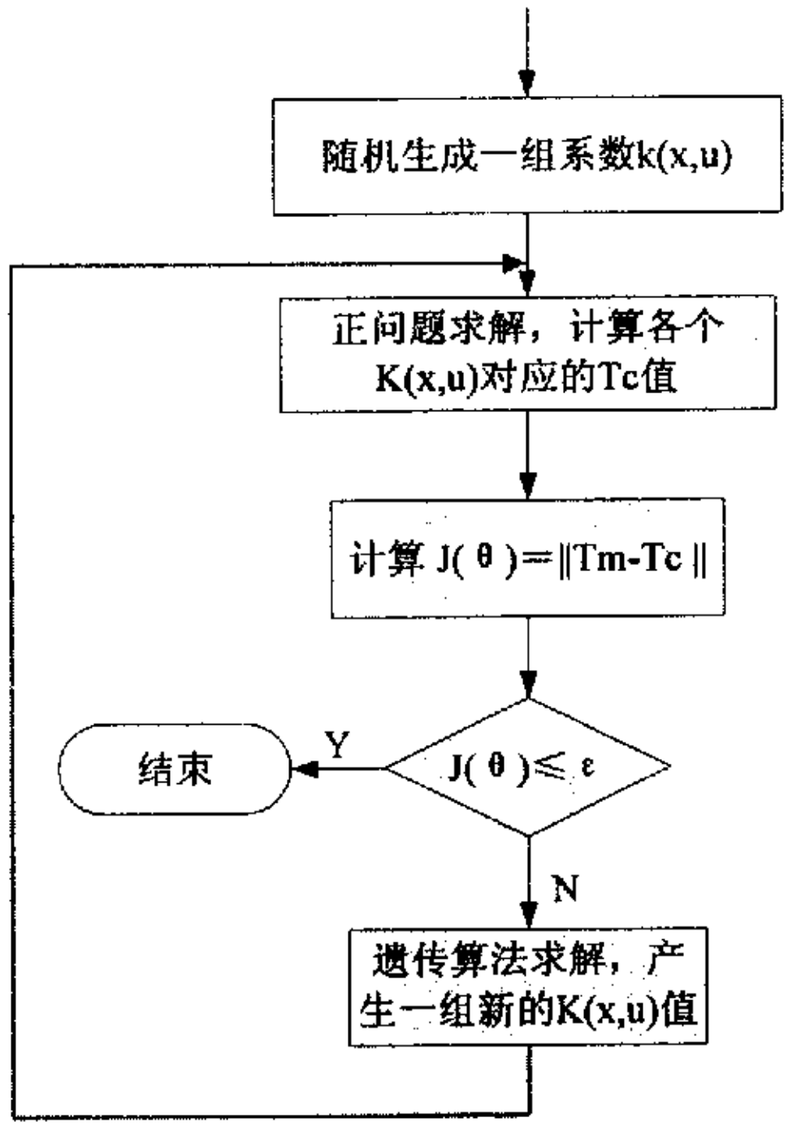


图 5-1　反问题求解程序框图

5.2 反问题并行遗传算法求解

5.2.1 算法的实现步骤

遗传算法（Genetic Algorithms, GA s）已被广泛用来求解 NP 问题和神经网络的训练等复杂的优化问题。由于 GA s 的特点，它适合于用来求解多变量优化问题。遗传算法实现多变量优化的基本步骤如下（假设有 n 个变量）：

Step 1　初始搜索区域。初始搜索区域根据不同问题总可以确定一个大致范围，即给定 $x_i\in[x_i\min,\ x_i\max]$　（i = 1 ,2 ,3, …, n）；

Step 2　基因编码。用一个长度为 L 的二进制数 t_i 来代表 x_i，则二者之间存

在映射关系：$x_i = x_i min + (x_i max - x_i min)\ t_i/(2L-1)$ (5-5)

n 个二进制数级联到一起就得到了一条代表初始解的染色体(个体)；

Step 3　第一代群体。将各自变量的初始搜索区分成（2L－1）等分，令 $t_i i$ = random(2L), i = 1 , 2 , 3, …, n , 则 t_i 级联到一起就得到了一条染色体, 即问题的一个原始解, 如此进行 m 次得到 m 个随机原始解, 构成原始群体；

Step 4　计算个体适用度。 依式(5-5)得各自变量的离散值 x_i, 代入适用度函数得个体适值 f_j, j=1， 2， …， m;

Step 5　根据串复制概率 p_r(F)选择 2 个串，各复制一份，适用度越高，则复制概率越大;

Step 6　在串上随机选择一个位置，在复制的 2 个串上标记为交叉位 csl;

Step 7　以交换概率 p_c 交换 csl 后的基因段;

Step 8　对两个串中的基因按变异概率 p_m 进行翻转;

Step 9　跳至 step 5，直至已复制 m 个串;

Step 10　从 step 4 开始重复进行，直到满足某一性能指标或规定的遗传代数。

以上遗传过程描述了最简单的进化模型，其中复制实施了适者生存的原则；交换的作用是组合父代中有价值的信息，产生新的后代，以实现高效搜索；变异的作用是保持群体中基因的多样性。

5.2.2 程序的具体实现

本文采用的是并行遗传算法中的粗粒度模型，在该模型中，每个子群体是一个独立的进程，在为这些子群体选择合适的拓扑连接时，可以选择使群体内部距离最大的一种最简单的结构。所选定的结构是将子群体连成环状，迁移可以沿环的一个方向进行，这样在整个群体内流动的基因的最佳量有可能比较低。

粗粒度模型在个体迁移方面有一些不同的策略：要迁移的个体可以选择适应度最佳的，也可以只选择适应度不低于子群体内平均值的，或者干脆是随机选择的。对于迁入的个体如何加入子群体中，可以是随机地选择相同个数的个体加以取代，也可以有目的地选择适应度最差或者是和迁入个体一模一样的个体加以取代。后者的效果显然要比前者好。

本文采用迁出群体中最优的个体，迁入的最优个体取代群体中最差的个体，

如图 5-2 所示。其中主控机除完成常规的遗传操作外，还负责整个并行计算的终止判断。

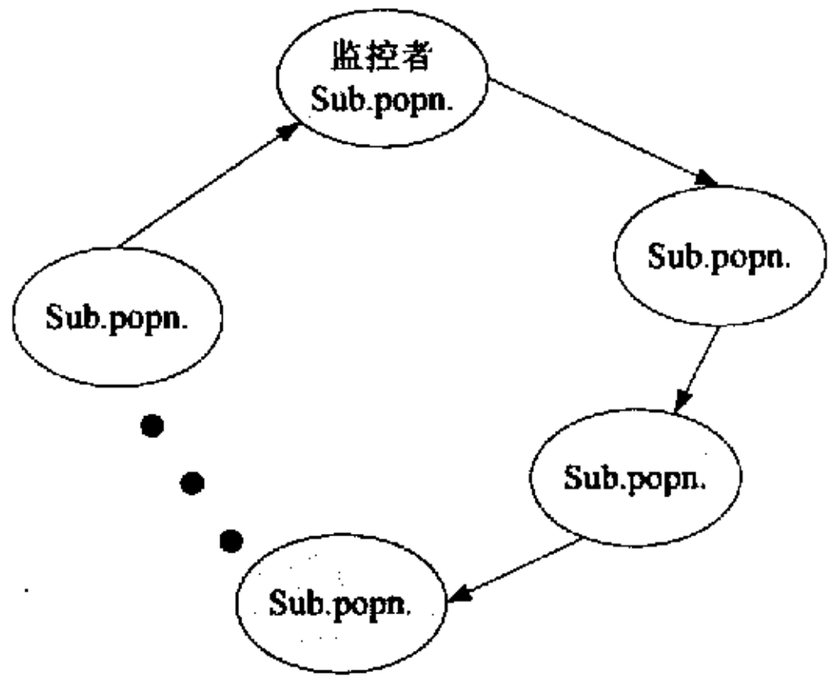


图 5-2 N 个子群体的环状迁移 GA

根据 5.2.1 算法绘制程序流程图如图 5-3 所示。

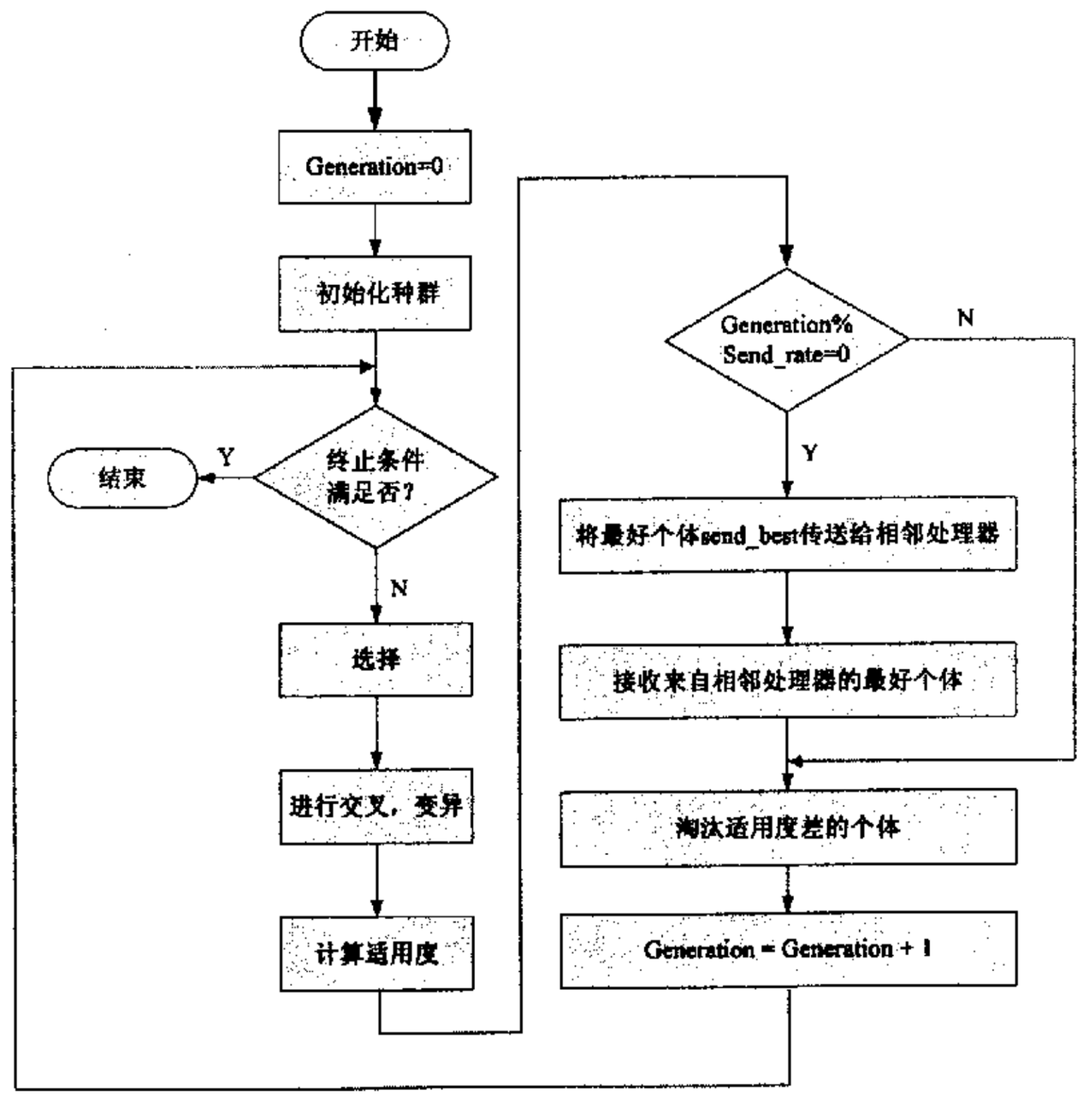


图 5-3 “一传一”的程序流程图

1）数据结构与遗传算法参数

基本遗传算法处理的对象主要是个体，因此设计了结构变量 individual 来描述个体信息，其中包括个体的染色体串 chrom，个体适应度 fitness，个体对应的变量 variale 等。为记录进化历代最佳个体，设计结构变量 bestever，表示最佳个体对应的染色体 chrom，个体适应度 fitness，对应的变量 variable 以及最佳个体产生的代数。根据个体变量定义，设计当前代种群 oldpop 以及新一代种群 newpop 为全局变量。

基本遗传算法运行参数包括：种群大小 popsize，染色体长度 lchrom，进化最大代数 maxgen，交叉率 pcross，变异率 pmutation 等[48,49]。这些参数在运行开始预先定义。

```
    ⋮
typedef struct _Individual{          //个体
    char gen[CLEN];                  //染色体
    double fitness;                  //个体适用度
    double variableA;                //个体对应的变量值
        ⋮
}Individual;
typedef struct _Bestever             //最佳个体
{
    char gen[CLEN];                  //最佳染色体
    double fitness;                  //最佳个体适用度
    double variableA;                //最佳个体对应的变量值
    ⋮
    int generation;                  //最佳个体生成代
}Bestever;
Individual oldpop[PNUM];             //当前代种群
Individual newpop[PNUM];             //新一代种群
Bestever    bestfit;                 //最佳个体
    ⋮
float pmutation=0.01;                //变异概率
float pcross=1.0;                    //交叉概率
```

```
double sumfitness;                    //种群中个体适用度累计
    ⋮
```

2）产生初始种群

为产生初始种群设计的函数为 initpop()。种群中个体的染色体随机产生。在产生染色体编码后，对个体进行解码。该算法程序如下：

```
void Initpop()                        //随机初始化种群，群体规模为PNUM
{
int i,j;
int a,b,c,d,e,f,g;
srand((unsigned)time(NULL));
for (i=0; i<PNUM; i++)
{
    a = (rand()%32768);      //2^CLEN
     b = (rand()%2048);
     c = (rand()%256);
     ⋮
    for(j=0;j<CLEN1;j++)
    {
        oldpop[i].gen[j] = 1&(a>>j);
    }
    for(j=0;j<CLEN2;j++)
    {
        oldpop[i].gen[30+j] = 1&(b>>j);
    }
    for(j=0;j<CLEN3;j++)
    {
        oldpop[i].gen[52+j] = 1&(c>>j);
    }
                ⋮
    Objfun(&oldpop[i]);          //计算对应变量值
}
```

```
    bestfit.fitness = 10;   //设置一个比较大的数
    bestfit.generation = 0;
}
```

3）遗传操作设计

为轮盘赌选择设计的函数 select()，返回种群中被选择的个体编号。方法是产生一个[0，1]随机数 pick，若 pick ＜ sum =oldpop[i]/sumfitness，则第 i 个个体被选中。由于轮盘赌选择对适用度函数的设计有一定要求，而排序选择方法的主要着眼点是个体适应度之间的大小关系。对个体适应度是否取正值或负值以及个体适应度之间的数值差异程度并无特别要求。因此本文采用对种群按适用度先排序，然后按个体在种群中的序位重新计算个体的适用度，对线性排序适用度计算方法如下：

$$Fit(Pos) = 2 - SP + \frac{2(\mathrm{SP}-1)(Pos-1)}{N-1} \qquad SP \in [1.0,2.0]$$

该算法程序如下：

```
void Pregeneration()      //计算排序后个体的适用度
{
    int i;
    double sp = 1.1;     //sp为选择压力
    sumfitness = 0.0;
    for( i=0; i<PNUM; i++)
    {
        gfit[i] = 2.0 -sp + 2.0*(sp-1)*i/(PNUM-1);
        sumfitness +=gfit[i];
    }
}
int Select()   //轮盘赌选择 先排序然后按排序后的适用度选择
{
     float sum,pick;
    int i;
    pick = Randomperc();
```

```
sum = 0.0;
if(sumfitness != 0 )
{
    for( i=0; ( sum < pick) && ( i< PNUM); i++)
    {
        sum += gfit[i] / sumfitness;
    }
    else
        i = Mid(1,PNUM);

    return (i-1);
}
```

为单点交叉操作设计的函数crossover()，由父个体parent1和parent2产生子个体child1和child2，若交叉发生处理编码赋值，并返回交叉点位置jcross；否则不做任何处理，返回0。方法是先通过flip（pcross）函数确定是否发生交叉操作，若发生交叉操作，在[1，lchrom]区间随机确定交叉位置jcross，child1继承parent1在jcross之前的编码和parent2在jcross之后的编，child2继承parent2在jcross之前的编码和parent1在jcross之后的编码。编码赋值按染色体的编码位逐一判断处理。该算法程序如下：

```
void Crossover(char pgen1[],char pgen2[],char cgen1[],char cgen2[])   //交叉操作
{
    int i,jcross;
    if(Flip(pcross))
    {
        jcross = Mid(1,CLEN-1);
        for(i=0;i<jcross;i++)
        {
        cgen1[i] = pgen1[i];
        cgen2[i] = pgen2[i];
    }
```

```
        for(i=jcross;i<CLEN;i++)
        {
            cgen1[i] = pgen2[i];
            cgen2[i] = pgen1[i];
        }
        gcross++;
    }
    else
    {
        for(i=0;i<CLEN;i++)
        {
            cgen1[i] = pgen1[i];
            cgen2[i] = pgen2[i];
        }
        gcross =0;
    }
}
```

为变异操作设计的函数mutation()，按变异概率pmutation确定个体child的编码位是否发生操作，若某编码位发生变异，则该编码翻转。该算法程序如下：

```
void Mutation(char gen[])    //变异操作
{
    int i;
    for(i=0;i<CLEN;i++)
    {
        if(Flip(pmutation))
        {
            gen[i] =!gen[i];
            gmutation ++;
        }
    }
}
```

4）世代进化过程实现

为模拟世代进化过程设计的函数 generation()，以种群的处理对象实现了一个世代内的三种遗传操作。首先在当前种群中用 select()函数选择两个个体，然后对两个个体按交叉概率实行可能的交叉操作和变异操作，然后对个体解码、计算适应度、记录亲子信息数据等，生成新一代个体。该算法程序如下：

```
void Generation()   //世代进化过程
{
    int mate1,mate2,j=0;
    gmutation = 0;
    do
    {
        //挑选交叉配对
        mate1 = Select();
        mate2 = Select();
        //printf("Select sel1=%2d , sel2=%2d \n",mate1,mate2);
        //交叉和变异
        Crossover(oldpop[Exchg[mate1]].gen,oldpop[Exchg[mate2]].gen,
                  newpop[j].gen,newpop[j+1].gen);
        Mutation(newpop[j].gen);
        Mutation(newpop[j+1].gen);
        Objfun(&newpop[j]);
        Objfun(&newpop[j+1]);
        j+=2;
    } while(j<PNUM-1);
}
```

5.2.3 数值实验结果

为验证遗传算法并行求解热传导系数的可行性，以及求解精度与遗传算法的一些参数选择的关系，我们利用实验室现有的计算机资源，将 6 台如下配置的 PC 机，通过网线集中到交换机上，从而组建了一个小型的机群，为并行程序的运行建立了一个硬件平台。

软件环境：

操作系统采用 Redhat 8.0，LAM/MPI 选用的为 lam-6.5.9。按 4.3 节所述进行配置。

硬件的基本配置为：

PC 机硬件	主要部件	配置
	CPU	INTEL P4 2.4GHZ
	内存	Kingston 1G DDR RAM
	硬盘	MT 80G 金钻 8 代 7200rpm
	网卡	INTEL 8460 100M
网络交换设备	3COM 3C16980A(24 口，100M 可堆叠)	

我们采用的数学模型如式（5-2）所示，$k=k(x,u)=(A-x^2)(B-Cu+Du^2)+x^2(E+Fu+Gu^2)$表示该新型复合材料热传导系数是 x，u 的函数，数值反演时，取定（A，B，C，D，E，F，G）＝（1.0，0.1，0.01，0.001，1.0，0.1，0.01）作为真实值，根据参考文献[50]中的算法求解正问题温度场分布，然后用遗传算法对上述系数进行反问题求解，由于遗传算法内在的随机性，为了衡量算法的性能，本文采用运算多次取平均值的方法，数值实验结果如下：

表 5-1 辨识值与真实值的关系

	A	B	C	D	E	F	G
实际值	1.0	0.1	0.01	0.001	1.0	0.1	0.01
辨识值	1.0194	0.0982	0.0098	0.00094	0.9865	0.1032	0.00949
绝对误差	0.0194	0.0018	0.0002	0.00017	0.0135	0.0032	0.00051
相对误差（%）	1.94	1.8	2.0	6.0	1.35	3.2	5.1

由表 5-1 可见，辨识结果具有较好的精度，在我们统计的数据中，相对误差的最大值不超过 10%。从辨识得到的结果看，参数 B、C、D 和 E、F、G 辨识值与真实值之间的误差依次增大。这是由于 B、C、D 和 E、F、G 的量级差别较大，B 的辨识值的误差湮没了 C 和 D 的真值造成的，同样，E 的辨识值的误差也湮没了 F 和 G 的真值[51]。实际上，我们真正关心的并非是参数 A、B、C、D、E、F、G 的辨识值与真值的误差，而是热传导系数的值与真值的误差，实验表明，两者的差别并不大。

表 5-2 加速比与并行效率

节点数	2	3	4	5	6
加速比	1.62	2.31	2.89	3.25	3.72
并行效率	0.81	0.77	0.72	0.65	0.62

同样，我们也统计了计算机节点数与加速比的关系，如表 5-2 所示，实验表明，并行遗传算法反演求解热传导系数所获得的加速比还是比较高的。

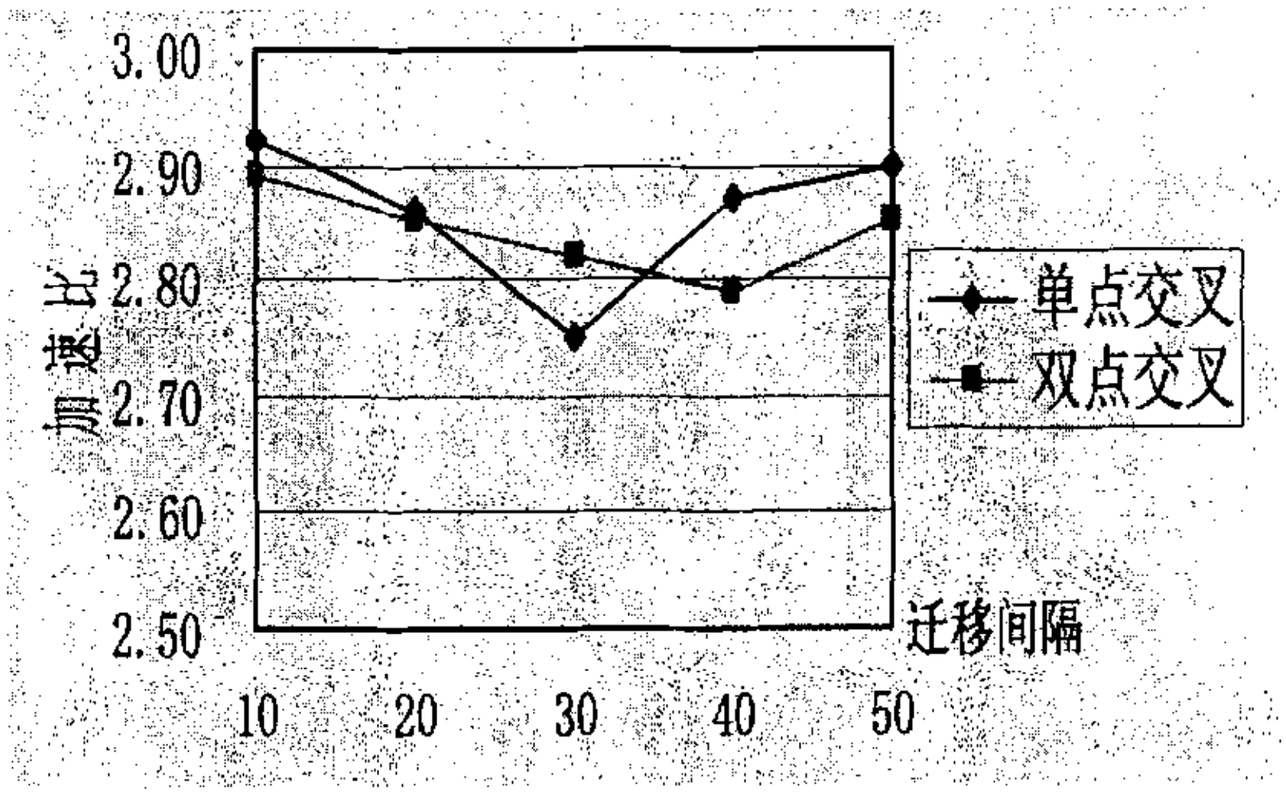


图 5-4 交叉点与加速比的关系（np=4）

同时，对衡量并行遗传算法特有的迁移间隔也做了以下实验，如图 5-4 所示，图 5-4 采用了两种交叉方式：单点交叉和多点交叉（双点），从图中可以看出，不管是单点还是多点（双点）交叉，其效率并不是一直增加或者减少，而是有个转折点，其中单点交叉的转折点发生在迁移间隔等于 30 代的位置，而双点交叉的转折点发生在迁移间隔等于 40 代的位置，从图中可以得出，当迁移间隔小于转折点时，由于迁移间隔较短，此时虽然增加了优良个体在整个循环中的迁移所增加的通信开销，但同时又加速了优良个体的扩散，使得遗传操作较早获得最佳个体，当超过某一转折点后，此时虽然优良个体的扩散减缓，但通信开销也同时减小，因此也获得了较好的效果。

第6章 结束语

随着信息时代得到来，需要处理的信息量越来越庞大、需要解决的问题越来越复杂，使得计算量剧增。比如本文所涉及的并行遗传算法求解二维热传导方程反问题，仅通过提高单个处理器的运算速度和采用传统的“顺序(串行)”计算技术已难以胜任。因此，需要有功能更强大的新的计算机系统和计算技术来支撑。将并行计算机及并行计算技术应用到求解此类问题也就顺理成章了。由于此类问题的应用在科学计算领域占有相当大的份额，并几乎涉及到国民经济的各个领域，因此将该问题并行化求解还是具有一定的实际意义的。

在机群系统中进行并行计算是非常有潜力的，因为它充分利用了现有的可利用的软硬件资源，实践的测试结果说明，对于某些应用问题，其性能优越于其它的并行编程环境。因此，机群系统是一种切实可行的并行计算机系统。本文成功的将并行遗传算法应用于热传导系数反问题求解中，并在局域网机群系统中利用 MPI 进行并行程序设计，并且讨论了并行遗传算法的一些性能参数对程序运行效率的影响，数值实验结果证明了所提方法的可行性及有效性，值得一提的是，在优化设计时，具体的热传导方程可以当作一个黑箱来处理，即具体的方程形式对遗传算法来说是完全未知的，正问题的具体解法跟遗传算法也没有直接关系，只需实验或者经验来确定热传导系数的一般形式。

同时我们也应看到，遗传算法还是处于发展阶段。在特定的反演问题中如何使用该算法，目前还不是很明朗，这些都需要从理论和实验两个角度去深入研究。尽管如此，我们还是看到了遗传算法的一些较好的特性，如并行运算寻找目标函数的极值点等。

参考文献

[1] Delaunay D., Jarug Y., Woodbury K.A. et al · Inverse Problems in Engineering，Theory and Practice · New York ： The American Society of Mechanical Engineers，1998，367-374

[2] 郭方中 · 动态传热学 · 武汉：华中理工大学出版社，1997，62-121

[3] George Z. Yang and Nicholas Zabaras · The Adjoint Method for an Inverse Design Problem in the Directional Solidification of Binary Alloys · JOURNAL OF COMPUTATIONAL PHYSICS，1998，140, 432–452

[4] Rajiv Sampath and Nicholas Zabaras · An object-oriented framework for the implementation of adjoint techniques in the design and control of complex continuum systems · International Journal for Numerical Methods in Engineering, 2000，48:239-266

[5] 孙家昶 · 网络并行计算 · 科学出版社，1997，27~37

[6] Guo Qingping (郭庆平)、Y. Paker 等 · Performance Evaluation of PVM on PC-LAN Distributed Computing · USA :Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'2000), June 2000, Las Vegas, Vol. 4, 1947~1953

[7] Rajkumar Buyya 著，郑纬民，石 威，汪东升等译 · 高性能集群计算 · 北京：电子工业出版社，2001，2~32

[8] Michael.W · The Researeh and Development of Parallel computation · Parallel Computing Theory and Practice，2000，129~135

[9] Barry Wilkinson，Michael Allen 著，陆鑫达等译 · 并行程序设计 · 北京：机械工业出版社，2002，3~89

[10] 黄 凯，徐志伟 著，陆鑫达，曾国荪，邓倩妮 等译 · 可扩展并行计算技术、结构与编程 · 北京：机械工业出社，2000，8~75

[11] David E.Culter 等 · 并行计算机体系结构 · 北京：机械工业出版社，1999.26~58

[12] 康继昌，朱怡安 · 现代并行机原理 · 西安：西北工业大学出版社，1996.145~169

[13] Dhabaleswar K. Panda, et al. · Special Issue on Workstation Clusters and Network-Based Computing · Parallel and Distributed Computing，1997,40~47

[14] David.Culler, Jaswinder.Palsingh Parallel Computing Architecture,1999,9,45~127

[15] 孙家昶，张林波，迟学斌 等 · 网络并行计算与分布式编程环境 · 北京：科学出版社，1996.57~96

[16] 张艳．分布并行算法设计、分析与实现：[博士学位论文]．电子科技大学, 2001.1~17

[17] 王鼎兴，郑纬民，沈美明．并行机群的若干关键技术．清华大学学报（自然科学版），1995.38(51)：78~83

[18] 肖祥锭．并行化程序性能调试环境的设计．湖南大学学报,1998.2：78～84

[19] 莫则尧，李晓梅．工作站网络环镜下的并行计算．计算机报,1997.6：209~215

[20] 罗省贤，何大可．基于 MPI 的网络并行计算环境及应用．成都：西南交通大学出版社，2001.104~197

[21] 陈国良，王煦法，庄镇泉 等．遗传算法及其应用．北京：人民邮电出版社，1996. 3~65

[22] 王小平，曹立明．遗传算法——理论、应用与软件实现．西安：西安交通大学出版社，2002，3~128

[23] [日]玄光男，程润伟 著，汪定伟，唐加福，黄 敏 译．遗传算法与工程设计．北京：科学出版社，2000，12~156

[24] 周 明，孙树栋．遗传算法原理及应用．北京：国防工业出版社，1999，90~256．

[25] Kamal C. Sarma Bilevel．Parallel Genetic Algorithms for Optimization of large Steel Structure．Computer-Aided Civil and Infrastructure Engineering, 2001, 295~304

[26] 邵 成，吴新余．改进的遗传算法在多热源选址中的应用．南京邮电学院学报，1997，17（1）：98~102

[27] 江厚满，张若棋，张寿齐．用遗传算法确定材料物态方程参数．高压物理学报，1998，12(1)：47~53

[28] 孟祥萍，梁志珊，张化光．一种基于二进制编码的优化方法．控制与决策，1998，13(增刊)

[29] David Andre, Jonn R. Koza．A parallel implementation of genetic programming that achieves super-linear performance．Journal of Information Sciences, 1998(106), 201~218

[30] Marc Snir，Steve Otto，Steven Huss-Lederman et al.．MPI，the Complete Reference volume 1，The MPI Core．London，England，The MIT Press, 1999

[31] William Gropp，Steven Huss-Ledeman，Andrew Lumsdaine et al.．MPI，The complete Reference volume2，The MPI extensions

[32] LAM/MPI Parallel Computing ：http://www.lam-mpi.org/

[33] William Gropp，Tutorial on MPI：The Message-Passing Interface, Argonne National Laboratory：http://www-unix.mcs.anl.gov/mpi/tutorial/gropp/talk.html#Node0

[34] Dongarra,J.．Introduction to MPI．The International Journal of Super Computer

Application，1999，369~378

[35] 都志辉．高性能计算之并行编程----MPI 并行程序设计．北京：清华大学出版社，2001，10~125

[36] 车静光．微机集群组建、优化和管理．北京：机械工业出版社，2004，12~125

[37] Brian Barrett, Jeff Squyres, Andrew Lumsdaine ．Integration of the LAM/MPI environment and the PBS scheduling system．In Proceedings, 17th Annual International Symposium on High Performance Computing Systems and Applications, Quebec, Canada, May 2003

[38] Jeffrey M. Squyres, Andrew Lumsdaine, William L. George，et al.．The Interoperable Message Passing Interface (IMPI) Extensions to LAM/MPI ．In Proceedings, MPI Developer's Conference, Cornell, NY, USA, 2000

[39] 魏永明，杨飞月，吴漠霖．Linux 实用教程．北京：电子工业出版社，1999.170~269

[40] Duchateau Pc, Rmndell W．Unity in an inverse problem for an unknown reaction erm in a reaction diffusion equation．J Diff Eqn, 1985, 59(2), 155~164

[41] Korn Rv, Vogelius M．Determining conductivity by boundary measurements．Comm Pure ApplMath, 1984, 37:289~298

[42] Tsien D S, Chen Y M．A numerical method for nonlinear inverse problem in fluid dynamic, proc, int cof．The Uhiv of Texas as Austiu, 1974, 22:943~945

[43] Beck J V, Blackwell B, Clair C R Sjr．Inverse Heat Conduction ill-posed Problems ．New York: John Wiley - Sons, 1985

[44] Tanaka M, Abstracts of International Symposium on Inverse Problems In Engineering Mechanies(ISIP 2000) ．Nagano City: symposium secretariat, 2000, 1~52

[45] 徐长发，李红．实用偏微分方程数值解法．武汉：华中理工大学出版社，2003,42~127

[46] 胡家赣．线性代数方程组的迭代解法．北京：科学出版社，1991，105~152

[47] 陆金甫，关治．偏微分方程数值解法．北京：清华大学出版社，1987，145~209

[48] 候安宁，何樵登．地震弹性参数的非线性数值反演．石油地球物理勘探，1994.29(6)：669~677

[49] 何耀华，韩守木，程尚模．求解多变量优化问题 Gas 方法的实现与改进．华中理工大学学报，1996.24(4)：96~99

[50] 杨磊．基于 PC 机群环境 MPI 的多重网格并行算法研究：[硕士学位论文]．武汉：武汉理工大学计算机学院，2004

[51] 郭彤城，慕春棣．并行遗传算法的新进展．系统工程理论与实践，2002.2：15~23

致 谢

在近三年的学习和生活中，导师郭庆平教授给予了我全面、耐心的指导，导师在科研、教学工作上一丝不苟、严谨求实的精神令我敬佩。在此特向他表达我由衷的感谢和敬意。

在完成本论文的工作过程中，还得到王伟沧，刘继军等老师的悉心指导和大力帮助，在此向他们表示深深的谢意。

感谢实验室的所有同学，他们是欧阳琳，唐国胜，毛黎明，陈俊，杨浩，饶静，李小薪，张峰等。

最后，感谢我的家人、我的父母，是他们给我很大的鼓励和支持，使得我能顺利的完成学业。

申鼎才

2005 年 3 月于武汉理工大学

附 录

攻读硕士期间发表论文情况

[1] 申鼎才，郭庆平．基于 Internet 的分布式数据采集与分析在岩土工程中的应用研究．武汉理工大学学报（交通科学与工程版）．

[2] 郭庆平，申鼎才．The Application of Parallel Genetic Algorithm On Heat Conduction Inverse Problem．DCABES2005，已录用．

攻读硕士期间参加科研项目情况

基于机群计算的热物性反问题高效分布式并行算法研究，国家自然科学基金资助项目（批准号：60173046）