摘 要

任务分配问题是一类典型的组合优化问题。多处理器系统上的最优任务分配的研究是有效利用系统资源处理实际问题的热点课题，这方面的研究结果在大规模数值计算、VLSI和计算机网络技术等方面都有很好的应用背景。在理论方面，由于任务分配问题是被公认的NP难问题，所以如何构造有效的启发式算法或近似算法是目前研究的热点领域。

蚁群优化算法是受自然界中的蚂蚁觅食行为启发而提出的一种新颖的仿生进化算法，适用于求解复杂组合优化问题。目前，蚁群优化算法已成功应用于求解旅行商问题、二次分配问题，取得了很好的实验效果。受其影响，国内外许多学者对其进行了大量的研究工作，将其推广到了诸多优化领域，并已经取得了相当丰富的研究成果。

虽然蚁群优化算法的应用范围几乎涉及到各个优化领域，但是还存在很多不足。比如：对于蚁群优化算法求解分布式系统中的任务分配问题的研究大都是在对该问题试验条件或约束条件进行简化的前提下进行的。

本文将蚁群优化算法应用于求解约束条件更复杂的任务分配问题：一个任务只能分配给一个处理机处理，而一个处理机可以处理多个任务，其中每个处理机都有固定成本和能力限制。将该任务分配问题表示成完全二部图，通过蚂蚁在完全二部图上搜索较优路径来寻求该问题的较优解。选择不同规模的几组数据进行实验，对每一组数据，通过反复试测探索了信息素挥发系数，信息素启发式因子和期望值启发式因子的合理设定，并将所得的计算结果与禁忌搜索和随机方法作比较。结果表明蚁群优化算法对不同规模的任务分配问题都有较优的结果，具有比禁忌搜索算法和随机方法更优的性能。

将另一任务分配问题抽象为一种新的有别于二部图的图形表示形式。针对蚁群优化算法易陷入局部最优的不足，提出了一种求解任务分配问题的混合算法，该算法将简单禁忌搜索算法嵌入蚁群优化算法，利用禁忌搜索算法较强的局部搜索能力，提高了蚁群优化算法的优化能力，改善了任务分配问题解的质量。仿真实验表明混合算法的性能优于基本蚁群算法。

最后，对本文的研究工作进行了总结，并指出了蚁群优化算法在该领域进一步还要研究的问题。

关键词：蚁群优化算法；任务分配问题；分布式系统；组合优化

Abstract

The task allocation problem is a typical combinatorial optimization problem. The research about the optimal task allocation on multi-processor system is a hotspot problem of making use of system resources to solve practical problem effectively. In theory, the task allocation problem is a recognized NP-hard problem, so how to construct effective heuristic or approximate algorithm is an important domain of research until now.

Inspired by foraging behavior of ants in the nature, the ant colony optimization algorithm was proposed as a novel bionic evolutionary algorithm for solving complicated combinatorial optimization problems, such as the traveling salesman problem, the quadratic assignment problem. At present, the research about ant colony optimization algorithm has already aroused attention from more scholars and experts gradually. They have made a lot of researches on it, extended it to many optimization domains and acquired considerably abundant research achievements.

Although application scope of the ant colony optimization algorithm is almost related to every optimization domain, there still exists deficiency. For example, the research on solving the task allocation problem in distributed system by the ant colony optimization algorithm is carried out under the premise of simplified experiment condition or constrained condition.

In this thesis, the ant colony optimization algorithm is applied to solve the task allocation problem with more complicated constrained condition, which each task can be assigned to only one processor, but each processor with capacity constraint and fixed cost can execute a number of tasks. The task allocation problem is expressed as a complete bipartite graph, sub-optimal solutions are obtained by ants which search sub-optimal routes in the graph. Several groups of data with different scales are chose to make experiments, for every group of data, optimum configuration of concerned parameters, are speculated by trial-test repeatedly, furthermore, the results of ant colony optimization algorithm is compared to tabu search and random method. The experimental results manifest that the ant colony optimization algorithm performs better under different problem scales and has great advantages over tabu search and random method in performance.

Another task allocation problem is turned into a new graph representation which is different from the bipartite graph. Aiming at easily plunging into local optimization of the ant colony optimization algorithm, a hybrid algorithm which combined ant colony optimization algorithm with tabu search for task allocation problem is proposed. Owing to the stronger local search ability of tabu search, the proposed hybrid algorithm can enhance optimization

ability of the ant colony optimization algorithm and improve quality of solution of the task allocation problem. The simulation results demonstrate that the hybrid algorithm performs significantly better than ant colony algorithm in performance.

In the end, the conclusion is given and the further research direction is pointed out.

Key words: Ant colony optimization algorithm; Task allocation problem; Distributed system; Combinatorial optimization

插图索引

附表索引

兰州理工大学学位论文原创性声明和使用授权说明

原创性声明

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写的成果作品。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律后果由本人承担。

作者签名：王灵霞　　日期：2008年 6 月 13 日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，即：学校有权保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权兰州理工大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。同时授权中国科学技术信息研究所将本学位论文收录到《中国学位论文全文数据库》，并通过网络向社会公众提供信息服务。

作者签名：王灵霞　　日期：2008年 6 月 13 日

导师签名：[illegible]　　日期：2008年 6 月 13 日

第1章 绪 论

1.1 分布式系统概述

分布式系统是二十世纪九十年代以来计算机领域的研究热点之一。作为网络一体化和并行处理分布化的产物，分布式系统解决了系统透明性及处理机动态分配等问题，表现出了强大的生命力。

一个被大家公认的分布式系统的定义是："分布式系统是一些独立的计算机的集合，但在用户看来却是一台计算机"[1]。在硬件结构上，尽管所有的分布式系统都是由多个CPU构成的，但可以用不同的方法将硬件组织起来，形成不同的系统。按照体系结构划分，当前典型的分布式系统可分为两种：共享存储多处理机与分布存储多计算机。如果结点间通过公用存储器中的共享变量实现相互通信，就称为多处理机(Multiprocessors)；如果结点间使用消息传递方式来实现相互通信，就称为多计算机(Multicomputers)[2]。

共享存储多处理机属于紧密耦合的分布式系统，采用共享主存通信。一个典型的共享存储多处理机系统由m个存储模块、p个处理器和d个I/O通道组成，(m,p,d)的数目按情况而定。由于处理机与主存之间互连网络带宽有限，所以当处理机数增多时，访问主存的冲突概率会加大，互连网络会成为系统性能的瓶颈。但由于是单一地址空间，所以较易于编程。

分布存储多计算机属于松散耦合的分布式系统，它的每个处理单元都是一个独立性较强的节点，系由CPU、本地存储器、I/O设备和消息传递通信接口所组成。由于每个计算节点的本地存储器容量较大，所以运算时所需的绝大部分指令和数据均可取自本地存储器。当不同计算节点上的进程需要通信时，就可通过接口进行消息交换。在有的松散耦合的多计算机系统中，其计算节点本身就是一台完整的计算机，这样就演变成了现代的机群系统。这种松散性耦合结构易于扩展，但多地址空间却使编程较困难。

分布式系统与集中式系统相比，有其自身的优点。分布式系统有较高的性能价格比，它还可能具有任何价位上的大型机都无法达到的性能。分布式系统的另一个优点是它有更高的可靠性，通过将任务交由多个机器共同承担，单个芯片的故障只会让一台机器停下来，而其它的机器将继续工作。对于一些关键性应用，例如控制核反应堆或飞机，使用分布式系统来达到高可靠性是最主要的考虑因素，并且在负载增加时分布式系统较集中式系统更容易扩展。

1.2 论文选题的背景和意义

众所周知，一般情形下（处理机个数大于3）的任务分配问题是NP难的。人们不

会盲目地去寻求解决这类问题的最优解。但对于具体的分布式系统，在适当假设条件下，寻找不一定最优但实际可行且效果较满意的方法，仍是现今十分活跃的研究课题[3]。在这方面已经研究出了许多各具特色的算法，较有代表性的一些算法是：基于图论的分配算法、数学规划方法、启发式算法、各种负载共享策略、动态投标算法以及专家系统方法等。它们可粗略地分为两大类：静态分配策略和动态分配策略。前者是在系统运行的初始时刻，将用户提交的任务一次性分配给系统中各处理机，此后直到这些任务运行完毕，各处理机上的任务一般不再变更。其特点是实现简单，但效果有限。后者是在运行过程中，将任务分配给各处理机，并对其上的任务数进行动态调整，尽可能使系统中各处理机上的负载达到基本平衡。其主要特点是能充分发挥各处理机的能力，但实现起来复杂程度高。

本节介绍分布式系统中的几种任务分配（task allocation）策略，这些策略的着眼点就是设法减少系统中各处理机间的通信开销和执行模块所需的开销，以此来提高整个系统的性能。

处理机间的通信（简称 IPC）开销是由位于不同处理机上的模块互相传递数据所引起的。因此，IPC 是模块间的通信（简称 IMC）和模块分配的函数，这里 IMC 是指每对模块间的数据传递。显然，如果两个模块同驻在一个处理机上，那么，它们就不会引起 IPC（假定同驻模块间的通信开销忽略不计）。

通常，若干模块构成一个任务，一个任务是单一的处理实体。任务分解和任务分配是用于减少 IPC 的两个必要步骤。任务分解的目的是把一个提交的任务分划成若干独立的、具有最小 IMC 的模块；任务分配则是指把这些模块分配给处理机，使得它们由于 IPC 所引起的开销最小。假定任务分解已由软件设计者在设计阶段完成，提交给系统的任务已经分解成若干模块。本节主要讨论如何以最优方式将模块分配给处理机。注意模块的个数通常多于（甚至远远多于）处理机的个数，因此，多个模块可能分配给一个处理机。下面将讨论任务分配环境、影响系统性能的因素。

1.2.1 任务分配环境

一般的分布式系统的示意图如图 1.1 所示，其中$(T_1,T_2,\cdots,T_m)$是一组待处理的模块，$(P_1,P_2,\cdots,P_n)$是系统中的n个处理机，它们经由互联网互相通信；模块分配机制S把m个模块合理地分配给n个处理机中的某个处理机，通常$m>n$。位于不同处理机上的模块彼此交换数据是通过它们所在的处理机彼此通信来实现的。任何一对模块的 IMC 是由设计阶段确定的，而且在模块分配期间是模块的一个固定特性。由于 IMC 的总量将随着模块对的不同而变化，因此，把这些模块分配给处理机的方式可能影响整个系统的处理开销。例如，若两个模块T_i和T_j的 IMC 量很大，通信也较频繁，而且它们又分配在不同的处理机上，那么，相应的 IPC 开销就会增大。如果把T_i和T_j分配到同一处理机上，就可能减少系统的总处理开销。此外，若两个模块的 IMC 较少，而且也无优先制约关

系，把它们分配给不同的处理机显然可能加速整个处理过程。

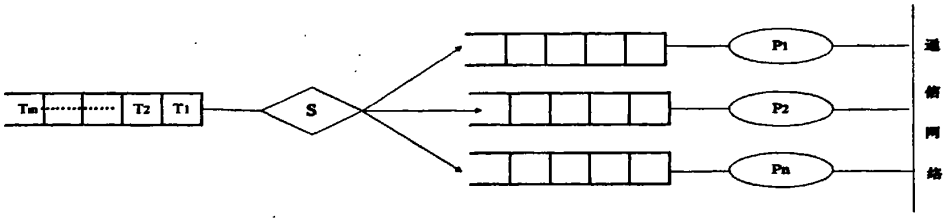

图 1.1 分布式系统示意图

1.2.2 影响系统性能的因素

均衡负载分配策略是把模块合理地分配给系统中的所有处理机，使得这些处理机差不多是均匀负载的。显然，这种分配策略可最大限度地提高系统的吞吐量。如果只注重减少 IPC 而不考虑系统的均衡负载，就可将待处理的模块全部分配给同一处理机处理。显然，这种分配策略可使 IPC 降至最低，但对同一任务的处理时间却成倍地增加了。

不难看出，均衡负载和减少 IPC 是相互冲突的两个因素，它们左右着任务分配策略，影响着系统的性能。负载平衡可以提高整个系统的吞吐量，因为它试图将模块尽可能均等地分配给处理机。但由于 IPC 的开销又迫使分配策略不得不尽可能把较多的模块分配给尽可能少的处理机。任务分配策略就是要设法平衡这两个因素，将模块分配给处理机，使整个系统的性能提至最高。

1.3 研究现状

计算机技术与应用的飞速发展，使得分布式系统的应用越来越广泛[4]。但随着分布式计算机系统的发展，分布式计算机领域中出现了很多亟待解决的问题，其中任务的分配和调度是一个关键性的问题。因为当分布式系统中处理器个数增多时，任务间的通信开销和处理机间等待时间增大，系统效率趋于饱和甚至下降，所以合理的任务分配策略是提高分布式系统效率不可缺少的措施。任务的分配将关系到系统的吞吐量、资源的利用率等，对于发挥各处理单元的作用，从整体上提高系统资源的利用率等具有非常重要的意义。因此在分布式系统中，任务分配问题一直是重要的研究课题之一。

任务分配问题是被公认的 NP 难问题[4]。任务分配的目的是减少并行运行任务间的通信开销和等待时间，改善负荷的均衡性，提高实时任务的响应速度和系统效率。目前基本的任务分配方法有三类：图论法[5][6]，数学规划法[7][8]和启发式方法[4][9][10][11]。图论法简单直观，适用于任务数和处理机数较小的系统；数学规划法是用数学规划的方法求解任务、处理机、系统资源间约束关系对应的一组方程式的最优解，由于求解的是 NP 问题，从而失去其工程应用价值；只有启发式方法实用简单，它是从系统结构和任务特点出发，运用一些有效的分配原则，使某些代价函数接近最优的任务分配策略。很多文章已证明合理的启发式方法求得的解虽不是最优解，却是次优解，尤其在其它两类方法

中难以考虑的资源共享、时间链约束、通信开销和延迟、存取 I/O 设备均可在启发式方法中得以考虑。

1. 图论法

图论方法中，应用于任务分配最重要的方法是偶图的匹配算法。将系统描述成一个偶图，在这个偶图中存在两个集合，分别为任务集合与结点集合，并用结点与任务间的边表示任务的分配。加入任务间通信费用的考虑，文献[12]提出了相应的结点匹配算法。

在文献[13]中，使用图中的结点来表示任务。每两个任务间的通信费用预知，并由连接这两个结点的无向边的权重来表示。如果费用为 0 则表示这两个任务间不存在通信；而费用为无穷则表示这两个任务必须被分配到同一个处理器(结点)。

2. 数学规划法

这种方法的主要思想是把任务分配问题概括成一种优化问题，然后利用数学规划方法去解决它[7][8]。

3. 启发式方法

文献[13]指出：有些应用由于自身的限制，无法在很重要的时间限度内获得一个最优解。对于这些应用，启发式方法能够提供快速而有效的算法。在该方法中，使用很多假设以减少任务分配的计算时间和，这些假设包括网络处理器是同构的、完全连接的，或各任务间的先后次序可被忽略。在这些假设条件下，可使用启发式模型簇算法，来实现基于实时与存储空间限制的任务分配。

文献[14]指出：可用图表示的启发式算法可被大致分为：交互式算法与非交互式算法。非交互式算法利用 TFG 的图论属性来得到一个最优化某一费用函数的方案。交互式算法则是生成一个初始的随机方案，然后逐步优化，直到费用函数值最小。由于这两类方案的不同，费用函数对其得到有效方案的影响也不同。一般的费用标准，如最小化总体执行时间，或最小化最重负载处理器的负载，被这两类算法共同采用。

另外，其它启发式算法也为解决此类问题提供了新的途径。遗传算法是一类模拟生物界自然进化和遗传过程的随机搜索和优化算法，具有在大问题空间求解近似最优解的能力。正因遗传算法的强健性和对复杂问题的求解能力，有人将其应用于多处理机系统的任务分配与调度[15][16]。

模拟退火算法是基于 Monte Carlo 迭代求解策略的一种随机寻优算法，其出发点是基于物理中固体物质的退火过程与一般组合优化问题之间的相似性。文献[17][18]应用模拟退火算法求解任务分配问题。

文献[19]运用改进的蚂蚁算法来求解简单的任务分配问题，即一个任务只能分配给一台机器处理，且每台机器只能处理一个任务。本文主要研究应用蚁群优化算法求解分布式系统上约束条件更复杂的任务分配问题，即一个任务只能分配给一个处理机处理，而一个处理机可以处理多个任务，其中每个处理机都有固定成本和能力限制，同时假设每项任务在不同的处理器上完成具有不同的费用，并且不同的处理机存在着通讯开销。

1.4 本文的研究内容及结构安排

第一章介绍了分布式系统中的任务分配问题、选题背景及研究现状。

第二章介绍了较有代表性的任务分配算法：基于图论的分配策略、整数规划方法和启发式算法，并对这三类典型算法进行比较和分析。

第三章首先从组合优化问题、NP问题及计算复杂性分析入手，对近年来被广泛应用的，具有代表性的仿生进化算法的算法原理、特点、研究现状等进行了综述，重点介绍了蚁群优化算法的原理、特点、改进算法以及研究与应用现状。

第四章应用蚁群优化算法来解决多处理器分布式系统上约束条件更复杂的任务分配问题：一个任务只能分配给一个处理机处理，而一个处理机可以处理多个任务，其中每个处理机都有固定成本和能力限制。仿真结果表明该算法比禁忌搜索和随机方法具有更好的求解能力。

在第五章中，为解决蚁群优化算法易陷入局部最优的不足，提出了一种求解任务分配问题的混合算法。该算法将简单禁忌搜索算法嵌入蚁群优化算法，利用禁忌搜索算法较强的局部搜索能力，提高了蚁群优化算法的优化能力，改善了任务分配问题解的质量。仿真实验表明混合算法的性能优于基本蚁群算法。

最后，对本文的研究工作进行了总结和展望。

第 2 章 分布式系统中的任务分配问题

在分布式系统中，任务分配问题一直是重要的研究课题之一。任务分配是在分布式系统环境中，将给定的一组子任务，按照系统的当前状态，按照一定的执行时序分配到网络结点上，以期获得较好的系统执行性能。现阶段，分布式系统任务分配问题的求解算法大致可分为三类：基于图论的分配策略、整数规划方法和启发式算法，下面对这三类典型算法进行比较和分析。

2.1 基于图论的分配策略

基于图论的分配策略的基本思想[3]是把待分配的一组模块作为图中的节点集，连接两个节点的有向连线上的权表示每对模块之间的 IMC 开销。IMC 开销为 0 意指相应的两模块之间无通信发生，因此它们在图中无连线；IMC 开销为∞意指相应的两模块间通信量很大，必须分配给同一处理机。可见，图中节点间连线上的权表示分配在不同处理机上每对模块之间的通信开销。假定任何一对同驻模块的 IPC 开销为 0。若把系统的总开销定义为系统的处理开销和 IPC 开销之和，那么，这种分配策略的目的就是实现最小的总开销。处理开销和 IPC 开销都是模块到处理机分配的函数，为了表示模块到处理机的分配，可定义下面的分配矩阵 X：$x_{i,k}=1$表示模块T_i分配到处理机P_k处理；$x_{i,k}=0$表示模块T_i未被处理机P_k处理。而处理开销则由下面的Q矩阵给出：

$$Q=\{q_{i,k}\},\quad i=1,2,\cdots,m,\quad k=1,2,\cdots,n$$

其中，$q_{i,k}$表示模块T_i在处理机P_k上的处理开销，它是该模块处理要求的度量。$q_{i,k}=\infty$隐含模块T_i不可能在处理机P_k上执行。

令$C_{i,j}$表示模块T_i和T_j之间的 IMC 开销。于是处理给定任务的总开销T可表示为X的函数：

$$T(X)=\sum_{k}\sum_{i}\left\{q_{i,k}x_{i,k}+\sum_{h\neq k}\sum_{j\neq i}C_{i,j}x_{i,k}x_{j,h}\right\} \tag{2.1}$$

其中，第一项表示每个模块在它所分配的处理机上的处理开销；第二项则表示非同驻模块间的 IPC 开销。最小开销分配则是在这种图上执行最小分割算法来获得。

这种分配策略的最大优点是它的简明性。但它也存在一些限制和不足：第一，所用的最小分割算法是一种基本的最小分割算法，它只能用于实现两个或三个处理机间的最小开销分配。若将它扩充成处理任意个数的处理机，则需要n维（n个处理机分配）的最小分配算法，该算法是极其复杂的。这就限制了这种任务分配策略的应用范围；第二，这种方法既未提供有关存储空间或处理时间等方面的限制手段，也未提供负载均衡方面的机制；第三，它没有能力去观察排队延迟效果。当一个以上的模块分配给同一个处理机时，就不可避免地会出现延迟。因此，基于图论的任务分配策略不是任务分配问题的

最好解决办法。

2.2 整数规划方法

该方法的基本思想[3]是仍用前面定义的Q矩阵来表示执行开销，但用一个$m\times m$矩阵V来定义IMC：

$$V=\{v_{i,j}\mid 1\le i\le m\ \&\ 1\le j\le m\ \&\ v_{i,j}\text{为}T_i\text{向}T_j\text{传送的数据量}\}$$

同时引进一个$n\times n$的距离矩阵D：

$$D=\{d_{i,j}\mid 1\le i\le n\ \&\ 1\le j\le n\ \&\ d_{i,j}\text{为}P_i\text{与}P_j\text{间的距离}\}$$

模块分配函数用$m\times n$矩阵x来定义：

$$x=\{x_{i,j}\mid 1\le i\le m\ \&\ 1\le j\le n\ \&\ x_{i,j}=1\text{表示}T_i\text{分配}P_j\text{上}\}$$

该分配方法的目标函数定义为：

$$T(x)=\sum_k\sum_i\left\{q_{i,k}x_{i,k}+\sum_{r\ne k}\sum_{j\ne i}wv_{i,j}d_{k,r}x_{i,k}x_{j,r}\right\} \tag{2.2}$$

其中，常数w用来调节通信开销和执行开销间的差异。

此时，任务分配即要找使T最小的那个x的指派。因此，这种方法实质是带有某些限制条件的隐式枚举算法，其时间复杂度随问题规模成指数增长。这显然限制了算法的实用性。这种方法的优点是很容易加入适当的限制条件，以满足实际环境的需要。

Maetal提出了一种分支界限法[20]，它用一棵搜索树来表示分配问题，每个叶子结点处表示一个分配方案。除了存贮限制和实时限制外，它还引入了以下限制条件：

$m\times n$的模块优先矩阵P：$\begin{cases}p_{i,j}=1\text{表示}m_i\text{不能分配给}P_j\\ p_{i,j}=0\text{否则}\end{cases}$

$m\times m$的模块互斥矩阵E：$\begin{cases}e_{i,j}=1\text{表示}T_i\text{与}T_j\text{不能分配给同一处理机}\\ e_{i,j}=0\text{否则}\end{cases}$

以及允许一个任务(模块)的多份拷贝，以提高系统的可靠性。

该算法在每个分支处检查P，E关系和存贮及实时限制条件，若不满足，则剪枝；若满足，则检查这次分配后的部分开销是否超过已得到的最小全部开销，若超过，则剪枝；否则就分配，并选择下一扩充结点。若全部可能的路径都被探查完，或规定的执行时间已到，则算法结束。

该算法的空间复杂度较小，只需记录当前最小开销的分配方案和此开销即可，约为$O(mn)$。但时间复杂度仍可能是指数级的，而且没有保护模块优先规定的机制和实施负载平衡的机制。

2.3 基于遗传算法和模拟退火算法的任务分配策略

(1) 算法说明

设有一个由n个处理机$P=\{p_1,p_2,\cdots,p_n\}$组成（可以是不同的处理机）的分布式系统[3]，需要执行m个任务$T=\{t_1,t_2,\cdots,t_m\}$，一般$n<m$。为了便于分析问题，可以建立下述五元组：

$$\sum=(T,<,Q,C,X)$$

其中，$T=\{t_1,t_2,\cdots,t_m\}$是任务的集合；“$<$”是T上的任务优先关系，$t_i<t_j(1\le i\le m,1\le j\le m)$表示任务$t_i$必须在任务$t_j$执行之前完成；$Q$是一个$m\times n$矩阵，其元素$q_{i,j}$表示任务$t_i$在处理机$p_j$上的执行时间（假设每个任务的运行时间已知）；$C$是一个$m\times n$矩阵，$c_{i,j}$表示任务$t_i$与$t_j$之间的通信开销；$X$是一个$m\times n$的任务分配矩阵，其中$x_{i,j}=1$表示$t_i$分配到处理机$p_j$上执行，否则$x_{i,j}=0$。

为了实现选择，还必须设计一个目标函数cost。cost是一个包含多种因素折衷的函数，它应能体现出设计者对系统的性能要求，例如可以采取下述方法：

$$\mathrm{cost}=\sum_{i=1}^{m}\sum_{k=1}^{n}\left(q_{i,k}\cdot x_{i,k}+w\cdot\sum_{r=1}^{k-1}\sum_{j=1}^{i-1}c_{k,r}\cdot x_{i,k}\cdot x_{j,r}\right) \tag{2.3}$$

其中常数w用来调节通信开销和执行开销之间的差异。在实际设计过程中，还可以选取不同的w的值或其它目标函数。

(2) 算法描述及简单分析

第一阶段 初始化

① 随机产生一个任务分配矩阵集：先设定一个全零的任务分配矩阵X，然后在X中的每一行由系统随机选定一个元素为1。如表2.1所示，在3个处理机和7个任务的情况下，t_1、t_2、t_3被分配给p_1；t_4、t_5被分配给p_2；t_6、t_7被分配给p_3。用同样的方法可产生多个任务分配矩阵。

表2.1 任务分配矩阵

任务/处理机	p_1	p_2	p_3
t_1	1	0	0
t_2	1	0	0
t_3	1	0	0
t_4	0	1	0
t_5	0	1	0
t_6	0	0	1
t_7	0	0	1

② 描述和确定初始任务分配集：对一个给定的任务分配方案，可用一个数据结构$TA=\{S,R[1..n]\}$来描述：

其中S是一个$\lceil \log_2 n \rceil \times m$位的二进制串，每位$\lceil \log_2 n \rceil$为一节，从左到右第$i$节表示任务$t_i$所在的处理机情况，这样表 2.1 中的任务分配矩阵可以表示为S=00 00 00 01 01 10 10。因为有 3 个处理机，所以两位为一节，00，01，10 分别表示任务被分配到p_1，p_2，p_3上执行，11 是无效编码。

R是一个n链表数组，$R[i]$表示处理机p_i上的任务执行顺序，仍以表 2.1 为例，若任务之间满足“<”优先关系$\{<t_1,t_2><t_4,t_5><t_6,t_7>\}$,则可确定三种任务分配方案：

p_1上执行顺序为t_1，t_2；p_2上执行顺序为t_4，t_5；p_3上执行顺序为t_6，t_7。R表示为：

$R[1]$：1→2→3　$R[2]$：4→5　$R[3]$：6→7

$R[1]$：1→3→2　$R[2]$：4→5　$R[3]$：6→7

$R[1]$：3→1→2　$R[2]$：4→5　$R[3]$：6→7

由于分配到同一处理机上的任务之间有多种排列，所以从一个任务分配矩阵可以产生多种方案。在这些方案中，有一些是违背“<”优先关系的，称为无效分配方案，否则，称为有效分配方案。上面列出的就是三种有效分配方案。从每一个任务分配矩阵所产生的分配方案中各选取一种或几种有效分配方案，便形成初始任务分配方案集。

③ 设置模拟退火算法中的初始温度$temp_0$和收敛率α：温度$temp_{i+1}=\alpha \cdot temp_i$逐步降低，这里$0<\alpha<1$。下标$i$既表示第$i$次迭代，也指称遗传算法中的第$i$代个体。实验表明，在任务数较多的情况下，选取$temp_0=1000$，$temp_\infty=1$，$\alpha$=0.9，可取得较好效果。

第二阶段 循环

直到$temp=1$或者连续多代未产生更好的分配方案为止。在循环过程中若落入局部最优的“陷阱”时要采用静态爬山方法。

① 交叉繁殖：从整个任务分配方案集中以一定的百分比随机选择出一个供交叉的子集。利用它们进行交叉繁殖。具体实现方法是对前述$TA.S$中某些节进行重组，以 2 个处理机和 5 个任务为例，设有两个父代分配方案TA_1和TA_2，其中

$TA_1.S$=0 0 0 1 1　$TA_1.R[1]$：1→2→3，$TA_1.R[2]$：4→5

$TA_2.S$=0 1 0 0 1　$TA_1.R[1]$：1→3→4，$TA_1.R[2]$：2→5

在$TA_1.S$中随机选定几节替换到$TA_2.S$中就形成了一个新的二进制串S'，假如选定$TA_1.S$中的第 1、第 4、第 5 节（从左到右），替换后为S'=0 1 0 1 1，其意义为：将t_1、t_3分配到p_1，将t_2、t_4、t_5分配到p_2。同时，为保持数组R的一致性，从$TA_1.R[1]$中删除任务t_2，并插入到$TA_1.R[2]$中，这时有三种插入位置，对应三种分配方案，从中选取一种或几种有效分配方案，就得到下一代的一组分配方案。

② 变异：除了交叉之外，还要对任务分配方案集以某个较小的比例，选出一个子集进行变异。变异的方式有多种，可以是对$TA.S$中的某些节求反，也可以是互换一方案集中几个处理的任务，还可以是其它方式。由于变异不会造成个体数量变化，所以每次

变异后对一种原方案只保留一种有效变异方案。

③ 复制：对没有交叉和变异的方案，直接加入到新的任务分配方案集中。

④ 选择：交叉和变异产生的方案称为新方案。计算新方案的目标函数 $\cos t$，并令 $\Delta\cos t = \cos t_{新方案} - \cos t_{旧方案}$，这时会出现两种情况：

$\Delta\cos t < 0$。表明新方案优于原方案，将新方案加入到新任务分配方案集中。

$\Delta\cos t \geq 0$。表明原方案优于新方案，此时计算概率值：$prob = \exp\left\{-\frac{\Delta\cos t}{k \cdot temp}\right\}$，这里 k 是玻尔兹曼常数，$temp$ 是当前温度。由系统产生一个 $(0,1)$ 上的随机数 $rand$，如果 $prob > rand$，则将新方案加入到新任务分配方案集中；如果 $prob \leq rand$，则放弃新方案，仍保留原方案。

可以对 $prob$ 算式作一个简单分析：k 是常数，在一定温度进行方案选择时，$temp$ 也可看作常数，所以此时 $prob$ 只与 $\Delta\cos t$ 有关，而且随 $\Delta\cos t$ 递增而递减。$\Delta\cos t$ 越小，表明新方案虽次于原方案，但接近原方案，因而是较好的方案，而此时 $prob$ 也就越大，$prob > rand$ 成立的可能性也越大，越容易被接受。反之同理。

2.4 蚂蚁算法

任务分配问题[19]最简单的一种情形可描述为，有 n 个任务需要分配给 n 个机器去完成，每个任务只能分配给 1 台机器处理，且每台机器只能处理 1 个任务，不同的分配花费不同的代价。任务分配问题要求找到 1 种分配方案所花费的代价最小。

若第 i 台机器完成第 j 项任务的代价 $C_{ij} \geq 0$，则可构成代价矩阵 $C_{n\times n}$。求解如何分配机器，才能在完成总任务的条件下，使总的花费代价最小。

设

$$R_{ij} = \begin{cases} 1 & \text{表示由第}i\text{台机器去完成第}j\text{项任务} \\ 0 & \text{表示不由第}i\text{台机器完成第}j\text{项任务} \end{cases} \quad i,j = 1,2,\ldots,n \tag{2.4}$$

则分配问题是求解任务分配阵 $R_{n\times n}$。约束条件为

$$\sum_{i=1}^{n} R_{ij} = 1 \qquad j = 1,2,\ldots,n \tag{2.5}$$

$$\sum_{j=1}^{n} R_{ij} = 1 \qquad i = 1,2,\ldots,n \tag{2.6}$$

$$R_{ij} = 0\text{或}1 \qquad i,j = 1,2,\ldots,n$$

在满足上述约束条件下，使目标函数 $C_{\min} = \sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij}R_{ij}$ 成立，$C_{ij} \in C_{n\times n}$，$R_{ij} \in R_{n\times n}$。

求解任务分配问题的蚂蚁算法的步骤如下：

(1) 令 $N_c = 1, s = 1, k = 1$ (从第 1 只蚂蚁开始)，首先计算第 k 台机器完成 n 项任务的概率 $P_{ij,k}\ (j = 1,2,3,\ldots,n)$，选出其中最大的概率，记为 $P_{ij_{\max},k}$，即由第 k 台机器来完成第

j_{max}项任务。然后，置第k只蚂蚁禁忌矩阵$N_{n\times n,k}$的第k行所有元素为INF，表明此机器已经有任务。同时置阵$N_{n\times n,k}$的第j_{max}列为INF，表明此任务已经分配了；置第k只蚂蚁的任务分配矩阵$R_{n\times n,k}$的元素$R_{kj_{max},k}=1$；代价向量D_{n,N_c}的元素$D_{k,N_c}=C_{kj_{max}}\circ P_{ij,k}$由式(2.7)计算，即

$$P_{ij,k}=\frac{T(i,j)^{\alpha}\times V(i,j)^{\beta}}{\sum_{j=1}^{n}T(i,j)^{\alpha}\times V(i,j)^{\beta}} \tag{2.7}$$

式中：α,β为可调系数，分别表示先前蚂蚁的工作循环中由第i台机器来完成第j项任务的信息素量及第i台机器完成第j项任务的效率的相对重要性；

(2) $s=2$，计算遍历其余$n-1$项任务分配给其余$n-1$台机器的概率$P_{ij,k}$，选出其中最大的概率，记为$P_{i_nj_n,k}$，即由第i_n台机器来完成第j_n项任务。然后，置第k只蚂蚁禁忌矩阵$N_{n\times n,k}$的第i_n行为INF，表明此机器已经分配了任务。同时置阵$N_{n\times n,k}$的第j_n列为INF，表明此任务已经分配了；置第k只蚂蚁的任务分配矩阵$R_{n\times n,k}$的元素$R_{i_nj_n,k}=1$；代价向量D_{n,N_c}的元素$D_{k,N_c}\le D_{k,N_c}+C_{i_nj_n}$；$s+1$，重复步骤(2)，直到$s=n$结束；

(3) $k\le n$，重复步骤(1)和(2)，直到$k=n$结束；

(4) 更新矩阵T的值。由于信息量的值是不断增加的，为了防止信息量的无限增长设一个蒸发系数$\rho<1$，从第2次算法循环开始，在每次算法循环开始时，根据$T=T\times\rho$来计算T阵，以此来限制信息量的无限增长。引入全体蚂蚁的信息量的增量ΔT，由于每只蚂蚁撒的信息量与其工作循环所得到的代价成反比，因此$\Delta T=\sum_{k=1}^{n}\frac{Q}{D_{k,N_c}}$，其中：$Q$为常数；

(5) 求出代价向量D_{n,N_c}中最小的元素，记为$D_{N_c,min}$。N_c+1，重复以上各步，直到$N_c=N_{cmax}$。需要注意的是，每次算法循环所得到的$D_{N_c,min}$与前一次算法循环得到$D_{N_c-1,min}$相比较，取其中的较小值作为$D_{N_c,min}$。

2.5 本章小结

一般说来，在由任意多个处理机组成的分布式系统中实现最优的任务分配是一个NP难问题。本章主要阐述了分布式系统中任务分配的一些经典算法的过程及其优缺点，并对近年来出现的一些新算法进行了探讨。

第3章 蚁群优化算法

易于表述但难于求解的组合优化问题始终是研究中的一个热点。许多应用问题都属于NP难问题，也就是说，人们普遍认为在多项式的计算时间内无法获得这些问题的最优解。正因为如此，在解决大规模的实际应用问题时，人们不得不采取近似的方法以求在一个相对较短的时间内获得近似最优解。在非严格的定义下，人们把这类近似算法称为启发式算法(heuristics)。通常这种算法通过利用针对具体问题的相关知识来建立或者改进所求得的解。

最近，许多研究者把焦点集中于一种叫做元启发式算法(metaheuristics)的新型算法上。这里所谓的元启发式算法指的是一类算法的集合，它可以用来定义一个大范围内应用于不同问题的启发式方法。对于难解的、与实际问题有关的组合优化问题，元启发式算法的使用可以明显提高算法在一个合理的时间内找到高质量解的能力。

其中一种特别成功的元启发式算法的灵感来源于真实蚂蚁的行为。从蚂蚁系统(Ant System，AS)的提出，到建立大量基于相同原理的算法，这些算法在学术研究和实际应用等领域都非常成功地解决了多种组合优化问题。本章我们将介绍一种元启发式算法的框架－蚁群优化[21](Ant Colony Optimization，ACO)，这种框架将覆盖上面提到的蚂蚁算法的实现。蚁群优化算法已经成为针对现有的应用和多种蚂蚁算法的一种共同框架。凡是符合蚁群优化框架的算法在本文都被称为ACO算法。

3.1 组合优化

优化方法涉及的领域很广，问题种类与性质繁多。归纳而言，最优化问题可分为函数优化问题和组合优化问题两大类，其中函数优化的对象是一定区间内的连续变量，而组合优化的对象是解空间中的离散状态。

组合优化[22](Combinatorial Optimization)是通过对数学方法的研究去寻找离散事件的最优编排、分组、次序或筛选等，是运筹学(Operations Research)中一个经典且重要的分支，所研究的问题涉及经济管理、工业工程、交通运输、通信网络等诸多领域。组合优化问题分为最小化问题和最大化问题，由于两个问题可以互相转化，所以本文只对最小化问题进行说明。该问题可用的数学模型描述为：

$$\min f(x) \tag{3.1}$$

$$st.\ \ g(x) \geq 0 \tag{3.2}$$

$$x \in D \tag{3.3}$$

其中，$f(x)$为目标函数，$g(x)$为约束函数，x为决策变量，D表示有限个点组成的集合。

组合优化问题的一个实例可以用这三个参数(D,F,f)表示，其中D表示决策变量的定

义域；F表示可行解区域，即$F=\{x\mid x\in D, g(x)\geq 0\}$，F中的任何一个元素称为该问题的可行解；f是目标函数，满足$f(x^*)=\min\{f(x)\mid x\in F\}$的可行解$x^*$称为该问题的最优解。组合优化问题的特点是可行解集合为有限点集。由直观可知，只要将D中有限个点逐一判别是否满足$g(x)$的约束并比较目标值的大小，就可以得到该问题的最优解。

典型的组合优化问题有旅行商问题、加工调度问题、0-1 背包问题、装箱问题、图着色问题、聚类问题等。

组合优化的特点是可靠解的数量有限。虽然从理论上讲这种有限问题可以通过简单枚举找到最优解，然而在实际情况中通常无法实现。特别当实际问题的规模很大，可行解的数量非常多时更是如此。解决这种困难的主要方法是采用智能优化算法。

3.1.1 优化算法及其分类

所谓优化算法，其实就是一种搜索过程或规则，它是基于某种思想和机制，通过一定的途径或规则来得到满足用户要求的问题的解。

就优化机制与行为而分，目前工程中常用的优化算法主要可分为：经典算法、构造型算法、改进型算法、基于系统动态演化的算法和混合型算法等[23]。

(1) 经典算法。包括线性规划、动态规划、整数规划和分枝定界等运筹学中的传统算法，其算法计算复杂性一般很大，只适于求解小规模问题，在工程中往往不实用。

(2) 构造型算法。用构造的方法快速建立问题的解，通常算法的优化质量差，难以满足工程需要。譬如，调度问题中的典型构造型方法有：Johnson 法、Palmer 法、Gupta 法、CDS 法、Daunenbring 的快速接近法、NEH 法等。

(3) 邻域搜索算法。从任一解出发，对其邻域的不断搜索和当前解的替换来实现优化。根据搜索行为，它又可分为局部搜索法和指导性搜索法。局部搜索法：以局部优化策略在当前解的邻域中贪婪搜索，如只接受优于当前解的状态作为下一当前解的爬山法；接受当前解邻域中的最好解作为下一当前解的最陡下降法等。指导性搜索法：利用一些指导规则来指导整个解空间中优良解的探索，如模拟退火、遗传算法、禁忌搜索等。

(4) 基于系统动态演化的方法。将优化过程转化为系统动态的演化过程，基于系统的动态演化来实现优化，如神经网络和混沌搜索等。

(5) 混合型算法。将上述各算法从结构或操作上相混合而产生的各类算法。

优化算法当然还可以从别的角度进行分类，如确定性算法和不确定性算法，局部优化算法和全局优化算法。

3.1.2 计算复杂性与 NP 问题

由于有些优化算法所需的计算时间和存储空间难以承受，因此算法可解的问题在实践中并不一定可解。只有了解所研究问题的复杂性，才能有针对性地设计算法，进而提高优化效率。

算法的时间和空间复杂性对计算机的求解能力有重大影响。算法对时间和空间的需

要量称为算法的时间复杂度和空间复杂度。问题的时间复杂度是指求解该问题的所有算法中时间复杂度最小的算法的时间复杂性，问题的空间复杂度也可类似定义。按照计算复杂性理论研究问题求解的难易程度，可把问题分为 P 类，NP 类和 NP 完全类(NP-Complete，NPC)[23]。

算法或问题的复杂度一般表示为问题规模 n 的函数，时间复杂度记为 $T(n)$，空间复杂度记为 $S(n)$。在算法分析和设计中，沿用实用性的复杂性概念，即把求解问题的关键操作，如加、减、乘、比较等运算指定为基本操作，算法执行基本操作的次数则定义为算法的时间复杂度，算法执行期间占用的存储单元则定义为算法的空间复杂度。在分析复杂性时，可以求出算法的复杂性函数 $p(n)$，也可用复杂性函数主要项的阶 $O(p(n))$来表示。若算法 A 的时间复杂度为 $T_A(n)=O(p(n))$，且 $p(n)$为 n 的多项式函数，则称算法 A 为多项式算法。时间复杂度不属于多项式时间的算法统称为指数时间算法。

P 类问题指具有多项式时间求解算法的问题类。但迄今为止，许多优化问题仍没有找到求得最优解的多项式时间算法，通常称这种比 P 类问题更广泛的问题为非多项式确定问题，即 NP 类问题。NP 类的概念通常由判定问题引入，下面介绍相应的若干概念。

定义 3.1：实例是问题的特殊表现，所谓实例就是确定了描述问题特性的所有参数的问题，其中参数值称为数据，这些数据占有计算机的空间称为实例的输入长度。

定义 3.2：若一个问题的每个实例只有“是”或“否”两种回答，则称这个问题为判定问题，并称肯定回答的实例为“是”实例，否定回答的实例为“否”实例或非“是”实例。

定义 3.3：若存在一个多项式函数 $g(x)$和一个验证算法 H，对一类判定问题 A 的任何一个“是”的判定实例 I 都存在一个字符串 S 是 I 的“是”回答，满足其输入长度 $d(S)$不超过 $g(d(I))$，其中 $d(I)$为 I 的输入长度，且验证算法验证 S 为 I 的“是”回答的计算时间不超过 $g(d(I))$，则称判定问题 A 为非多项式确定问题，简称 NP 类。

由此可见，判定问题是否属于 NP 类的关键是对“是”的判定实例是否存在满足上述条件的一个字符串和算法，其中字符串在此可理解为问题的一个解，而定义中没有强调字符串和算法是如何得到的。可见，P⊂NP。

归约和转换是描述问题特性的常用方法。如果能够将几类问题归结为一个问题，则一旦解决了一个归结后的问题，其它几类问题也就解决了。

定义 3.4：给定问题 A_1 和 A_2，称 A_1 可多项式转换为 A_2，若存在一个多项式函数 $g(x)$和一个字符串，满足：（1）对 A_1 的任何一个实例 I_1，在其输入长度的多项式时间 $g(d(I_1))$内构造 A_2 的一个实例 I_2。，使其长度不超过 $g(d(I_1))$；（2）由此构造使得实例 I_1 和 I_2 的解一一对应，且 d_1 为 I_1 的“是”回答的充要条件为 d_1 对应的解是 I_2 的一个“是”回答。

定义 3.5：给定判定问题 A_1 和 A_2，称 A_1 可多项式归约为 A_2，若存在多项式函数 $g_1(x)$和 $g_2(x)$，使得对 A_1 的任何一个实例 I，在多项式时间内构造 A_2 的一个实例，其输入长度不超过 $g_1(d(I))$，并对 A_2 的任何一个算法 H_2，都存在问题 A_1 的一个算法 H_1，使得 H_1

调用 H_2 且计算时间 $f_{H_1}(d(I)) \le g_2(f_{H_2}(g_1(d(I))))$。

定义 3.6：称判定问题 A∈NPC，若 A∈NP 且 NP 中的任何一个问题可多项式归约为问题 A，称问题 A 为 NP 难问题，只要上述第二个条件成立。

由此可见，NPC⊂NP-hard，而两者的区别仅在于 NPC 必须判断问题属于 NP 类。同时，若已知一个问题为 NP 完全问题或 NP 难问题，当遇到一个新问题时，若已知问题可多项式归约为新问题，则新问题为 NP 难问题，进而若可验证新问题属于 NP 类，则新问题为 NPC 类。

上述四类问题的关系，按大多数学者的观点，较可能的表述见下图：

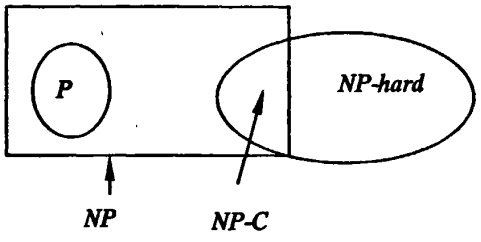

图 3.1 四类问题的关系图

NPC 类问题具有重要的实际意义和工程背景，目前已有许多问题被证明为 NPC 类，如背包问题、装箱问题、调度问题、集合覆盖问题、旅行商问题等。这些问题的求解需要兼顾解的质量和求解时间，一方面可以采用简单、直接的策略和规则来构造近似算法；另一方面可设计平均性能良好但在最坏情况下仍然是指数时间的概率搜索算法。

3.2 蚁群优化算法

蚁群算法是意大利学者Dorigo于20世纪90年代提出的一种新型的模拟进化算法[24][25][26]，应用该算法求解旅行商问题[26][27]，调度问题[28]等，取得了一系列较好的实验结果。虽然研究时间不长，但是初步研究已显示蚁群算法在求解复杂优化问题（特别是离散优化问题）方面具有一定的优势，这表明它是一种很有发展前景的方法。

3.2.1 蚁群算法生物学原理

个体行为显得盲目的蚂蚁在组成蚁群后却能产生惊人的自组织行为，在蚂蚁寻找食物过程中，总能找到一条从蚁穴到距离很远的食物之间的最短路径。仿生学家经过研究发现，蚂蚁寻找食物的奥妙在于，它在行进的过程中，会不断分泌一种特殊的挥发性物质－信息素(pheromone)。信息素主要有两方面的作用：一是蚂蚁之间通过它来相互通信，它可以吸引后来的蚂蚁沿信息素浓度高的路径行进；二是信息素的挥发作用，这使得寻径初期的距离较长的路径和长期没有经过的路径对蚂蚁的影响逐渐减小[25]。

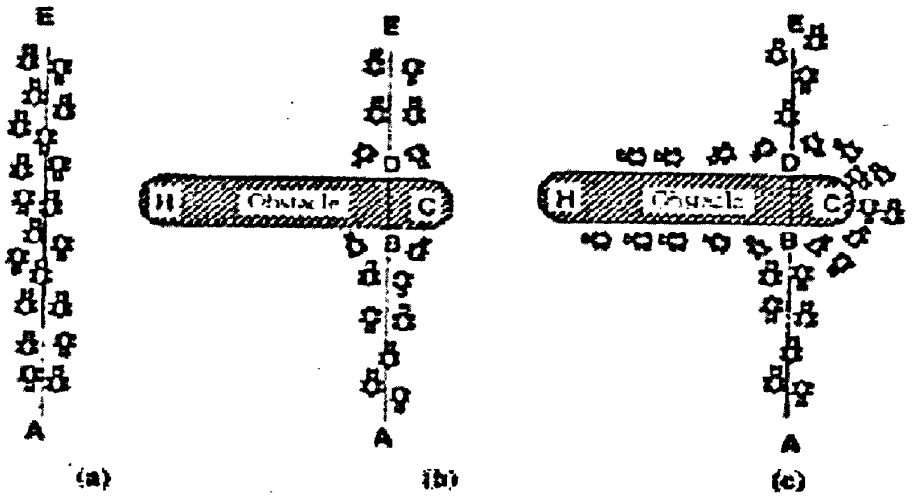

图 3.2 蚂蚁寻找最短路径示意图

如图 3.2 所示，我们假设 A 点是食物源，而 E 点是蚁穴。如果 A、E 两点间没有任何障碍物，虽然开始时蚂蚁所走的路径无序，但经过一段时间后，蚁群将沿着 AE 间的直线路径往来于这两点之间，如图 3.2(a)。如果在点 A、E 之间出现一个障碍物，蚂蚁就必须在 B 点挑选一条路径前行，或者从 H 点绕过去，或者从 C 点绕过去，如图 3.2(b)。这种决定会受到各条路径上以往蚂蚁留下的信息素浓度的影响。如果向 C 点的路径上的信息素浓度比较大，那么向 C 点的路径被蚂蚁选中的概率也就会相对较大一些。对于最早到达 B 点的蚂蚁而言，因为没有信息素的影响，所以它们选择向左或者向右的可能性是一样的，有的选择向左，有的选择向右。由于路径 BCD 比路径 BHD 要短，因此选择路径 BCD 的蚂蚁要比选择 BHD 的蚂蚁早到达 E 点，指向路径 DCB 的信息素浓度要比指向路径 DHB 的信息素浓度大。因此当蚂蚁返回时，从 E 点经 D 点达到 A 点的蚂蚁选择路径 DCB 比选择 DHB 路径的可能性要大得多。从而使路径 BCD(或 DCB)上信息素浓度比路径 DHB(或 BHD)上信息素浓度大得更多。而信息素浓度差变大的结果是选择路径 BCD(或 DCB)的蚂蚁进一步增多，这又导致信息素浓度差进一步加大，如图 3.2(c)。这个过程的结果会使所有的蚂蚁都走最短的那一条路径。

大量蚂蚁组成的蚁群的集体行为实际上就构成了一种信息正反馈现象：某一路径上走过的蚂蚁越多，则后来者选择该路径的概率就越大。该算法的本质在于：

(1) 选择机制，信息素越多的路径，被选择的概率越大；

(2) 更新机制，路径上面的信息素会随蚂蚁的经过而增长，而且同时也随时间的推移逐渐挥发消失；

(3) 协调机制，蚂蚁之间实际上是通过信息素来相互通信、协同工作的。

3.2.2 基本蚁群算法及其实现

为了便于理解，以使用蚁群算法求解平面上 n 个城市的旅行商问题（用1,2,...,n 表

示城市序号）为例，来说明蚁群算法具体的实现步骤。对于其它问题，可以对此模型稍加修改便可应用。虽然它们从形式上看略有不同，但基本原理是相同的，都是通过模拟蚁群行为达到优化目的。

旅行商问题，也叫货郎担问题，即为给定相互连通的城市的集合$\{1,2,...,n\}$，及城市i和j之间的距离$d_{ij}\ (i,j=1,2,...,n,d_{ii}=0)$，模型的目标是寻找一条经过$n$个城市各一次且仅一次，最终回到出发点的最短回路。若将每个顶点看成是图上的节点，距离d_{ij}为连接顶点i、j边上的权，则旅行商问题就是在一个具有n个节点的完全图上找到一条花费最小的 Hamilton 回路。如果d为欧式距离则有$d_{ij}=d_{ji}$，这样的旅行商问题称为对称的旅行商问题。

给定一个有n个城市的旅行商问题，设人工蚂蚁的数量为m，每个人工蚂蚁的行为符合下列规律：

(1) 根据路径上的信息素浓度，以相应的概率来选取下一步路径；

(2) 不再选取自己本次循环已经走过的路径为下一步路径，用一个数据结构$tabu_k$来控制这一点；

(3) 当完成了一次循环后，根据整个路径长度来释放相应浓度的信息素，并更新走过路径上的信息素浓度。

为模拟实际蚂蚁的行为，首先引进如下记号：设m为蚁群中蚂蚁的数量，d_{ij}是城市i、j间距离，那么有：$d_{ij}=\sqrt{(x_i-x_j)^2+(y_i-y_j)^2}$，式中$x_i$、$y_i$、$x_j$、$y_j$分别为$i$、$j$的坐标，$b_i(t)$表示$t$时刻位于城市$i$的蚂蚁个数，$m=\sum_{i=1}^{n}b_i(t)$。$\tau_{ij}(t)$表示在$t$时刻边$(i,j)$上的信息素浓度。当蚂蚁完成一次循环后，相应边上的信息素浓度为$\tau_{ij}(t+1)=\rho\tau_{ij}(t)+\Delta\tau_{ij}$，即等于上一时间段残留下的信息素浓度加上当前时间段新增加的信息素浓度。其中ρ为一个取值范围在0到1之间的常数，显然，$1-\rho$表示在时间段t到$t+1$之间信息素浓度的挥发强度因子。

$\Delta\tau_{ij}=\sum_{k=1}^{m}\Delta\tau_{ij}^{k}$。其中$\Delta\tau_{ij}^{k}$是第$k$只蚂蚁在时间$t$到$t+1$之间在边$(i,j)$上增加的信息素浓度。Dorigo 针对$\Delta\tau_{ij}^{k}$取值的不同给出了蚂蚁算法的三种模型[25]，分别称之为蚁周系统(Ant-Cycle System)、蚁量系统(Ant-Quantity System)、蚁密系统(Ant-Density System)。

对于蚁周系统，$\Delta\tau_{ij}^{k}$定义为：

$$\Delta\tau_{ij}^{k}=\begin{cases}\dfrac{Q}{L_k}, & \text{若第}k\text{只蚂蚁在时间}(t,t+1)\text{之间经过边}(i,j)\\ 0, & \text{否则}\end{cases} \tag{3.4}$$

对于蚁量系统，$\Delta\tau_{ij}^{k}$定义为：

$$\Delta\tau_{ij}^{k}=\begin{cases}\dfrac{Q}{C_{ij}}, & \text{若第}k\text{只蚂蚁在时间}(t,t+1)\text{之间经过边}(i,j)\\ 0, & \text{否则}\end{cases} \tag{3.5}$$

对于蚁密系统，$\Delta\tau_{ij}^{k}$ 定义为：

$$\Delta\tau_{ij}^{k}=\begin{cases}Q, & \text{若第}k\text{只蚂蚁在时间}(t,t+1)\text{之间经过边}(i,j)\\ 0, & \text{否则}\end{cases} \tag{3.6}$$

在上述公式(3.4)、(3.5)和(3.6)中 Q 是一个常量，用来表示蚂蚁完成一次完整的路径搜索后，所释放的信息素总量。L_k 表示第 k 只蚂蚁的总路径长度，它等于第 k 只蚂蚁经过的各段路径长度的总和。显然，蚂蚁不会在其没有经历过的路径上释放信息素。值得注意的是，后两种模型中（如公式(3.5)和(3.6)所示）利用的是路径的局部信息，而公式(3.4)利用的是路径的整体信息，在求解旅行商问题时，第一个模型性能较好。

定义第 k 只蚂蚁当前时刻在顶点 V_i，下一步选择顶点 V_j，即选择路径 (i,j) 的概率为：

$$p_{ij}^{k}(t)=\begin{cases}\dfrac{[\tau_{ij}(t)]^{\alpha}\times[\eta_{ij}(t)]^{\beta}}{\sum_{l\in allowed_k}[\tau_{il}(t)]^{\alpha}\times[\eta_{il}(t)]^{\beta}}, & \text{若}j\in allowed_k\\ 0, & \text{否则}\end{cases} \tag{3.7}$$

其中 $\eta_{ij}=\dfrac{1}{C_{ij}}$，$C_{ij}$ 为路径 (i,j) 长度。α,β 为两个参数，分别用来控制信息素浓度和路径长度的相对重要程度。$allowed_k$ 是第 k 只蚂蚁在满足前面所述人工蚂蚁行为规律 3 个条件的前提下，下一步可以选择的路径集合。

根据上面的分析，再引入一个控制“蚂蚁不再选取自己本次循环已经走过的路径作为下一步路径”这一行为特性的数据结构 $tabu_k$，表示第 k 只蚂蚁访问过的城市集合，其中 $1\le k\le m$。在初始化的时候，m 只人工蚂蚁被放置在不同的城市上，赋予每条边的信息素浓度初值为一常数 $\tau_{ij}(0)=C$。每只蚂蚁的 $tabu_k$ 的第一个元素赋值为它所在的城市。当蚂蚁完成一次完整的寻径过程后（即从某个城市出发走完所有城市一次并回到出发点），计算 $\Delta\tau_{ij}^{k}$，并更新每条边上的信息素浓度，然后开始新一轮循环。直到迭代次数达到事先定义好的最大循环次数 NC 或所有的蚂蚁都选择了同一路径时，程序终止。

蚁周系统描述如下：

(1) 初始化：$t=0$，$nc=0,\tau_{ij}(t)=C,\Delta\tau_{ij}=0$，并置 m 只蚂蚁于 n 个城市上，其中 nc 为迭代循环的次数，t 是一个“伪”时间控制变量；

(2) 将各蚂蚁的初始出发点置于当前解集 $tabu_k$ 中，对每只蚂蚁 $k(k=1,2,...,m)$ 按概率 p_{ij}^{k} 移至下一顶点 j，并将 j 置于当前解集 $tabu_k$ 中；

(3) 计算各蚂蚁的目标函数值 $L_k(k=1,2,...,m)$，记录当前的最好解；

(4) 按信息素浓度调整规则更新每条边的信息素浓度；

(5) 对各边 (i,j) 置 $\Delta\tau_{ij}=0$，$nc=nc+1$；

(6) 若 $nc<NC$ 且无退化行为（即找到的都是相同解），则转步骤(2)，否则结束并输

出当前最好解。

3.2.3 蚁群优化算法的优点与不足之处

1. 蚁群算法的优点

蚁群算法主要是采用信息正反馈原理和某种启发式算法的有机结合。在构造解的过程中，ACS采用确定性选择和随机选择相结合的选择策略，在搜索过程中动态地调整确定性选择的概率，当进化到一定代数后，进化方向已经基本确定，这时对路径上信息量作动态调整，缩小最好和最差路径上的信息浓度的差别，并且适当加大随机选择的概率，以利于对解空间的更广泛地搜索。ACS属于自适应方法，具有很强的自适应性。更重要的是，问题的最优解并不因为其中少数蚂蚁的死亡或者不良表现而受到影响，是一个蚂蚁群体的活动而非单个蚂蚁的活动。与其它算法，如禁忌搜索、人工神经网络、遗传算法相比，它的优点有：

(1) 蚁群算法是一种正反馈的算法。从真实蚂蚁的觅食过程中我们不难看出，蚂蚁能够最终找到最短路径，直接依赖于最短路径上信息素的堆积，而信息素的堆积却是一个正反馈的过程。对蚁群算法来说，初始时刻在环境中存在完全相同的信息素，给予系统一个微小扰动，使得各个边上的轨迹浓度不相同，蚂蚁构造的解就存在了优劣，算法采用的反馈方式是在较优的解经过的路径留下更多的信息素，而更多的信息素又吸引了更多的蚂蚁，这个正反馈的过程使得初始的不同得到不断的扩大，同时又引导整个系统向最优解的方向进化；

(2) 蚁群算法本质上是一种并行的算法，具有本质并行性，易于并行实现，个体之间不断进行信息交流和传递，有利于发现较好解，单个个体容易收敛于局部最优，多个个体通过合作，可很快收敛于解空间的某一子集，有利于对解空间的进一步探索，从而发现较好解；

(3) 蚁群在满足问题的约束条件下搜索，因而蚁群算法对处理约束满足问题有一定的优越性；

(4) 易于与其它启发式算法结合，以改善算法的性能；

(5) 蚁群算法的鲁棒性较好，相对于其它算法，蚁群算法对初始路线要求不高，并且对蚂蚁算法模型稍加修改，便可以应用于其它问题的求解；

(6) 相对于某些需要人工干预的算法，蚁群算法的搜索过程不需要人工的调整。

2. 蚁群算法的几个缺陷

虽然蚁群算法具有很强的全局寻优能力，在很多领域获得了广泛的应用，但也存在一些缺陷，主要有以下两点：

(1) 与其它的寻优算法相比较，蚁群算法的搜索时间过长。蚁群中各个个体的运动是随机的，虽然通过信息素交换能够向着最优路径进化，但是当群体规模较大时，很难在较短的时间内从大量杂乱无章的路径中找出一条较好的路径。这是因为在进化的初

期，各个路径上的信息素差别不大，通过信息正反馈，使得较好路径上的信息素逐渐增大，经过较长一段时间，才能使得较好路径上的信息素明显高于其它路径，随着这一过程的进行，差别越来越明显，从而最终收敛，这一过程一般需要较长的时间。一般地，蚁群算法的时间复杂度为$O(NC \cdot n^2 \cdot m)$, NC为循环次数，大部分的时间被用于解的构造；

(2) 蚁群算法的执行过程中容易出现停滞现象。蚁群算法中搜索进行到一定的程度后，所有蚂蚁个体发现的解趋于一致，此时，即使使用随机搜索策略，也不可能在解空间中进一步搜索，这样就存在陷入局部极小的可能性，不利于发现更好的解。原因在于信息素轨迹更新规则中不被选用的弧段上的信息素轨迹和选中的弧段上的信息素轨迹的差异会变的越来越大，而蚂蚁始终沿着信息素轨迹高的弧段爬行，这就导致当前不被选用的弧段今后被蚂蚁选择的概率变的越来越小，进而使算法只会在某些局部最优解附近徘徊，出现停滞现象。

3.2.4 改进的蚁群优化算法

1991年，第一个蚁群优化算法的模型－蚂蚁系统[21][24](Ant System，AS)用于求解旅行商问题等组合优化问题，实验结果表明AS算法具有较强的鲁棒性和发现较好解的能力，但同时也存在一些缺陷，如收敛速度慢、易出现停滞现象等。在随后的几年中，Dorigo等人从两方面对AS进行了发展，一方面针对AS中的问题进行改进，提出了新的算法模型；另一方面将AS及其改进模型不断应用到各种具体问题之中。

在众多改进的算法中，以蚁群系统和最大－最小蚂蚁系统这两种算法效果最好。

1. 蚁群系统

蚁群系统[26]是Dorigo和Gambardella在1996年提出的，在AS的基础上主要做了三个方面的改进：使用了伪随机比例规则；将全局更新规则应用于最优的蚂蚁路径上；使用局部更新规则。结果表明ACS大大改善了蚂蚁系统。

(1) 采用伪随机比例选择规则选择下一个城市，即对位于城市i的蚂蚁，按照公式(3.8)选择下一个城市j：

$$j = \begin{cases} \max\left\{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}\right\}, & \text{若} q \le q_0 \\ S, & \text{否则} \end{cases} \tag{3.8}$$

其中：$0 \le q \le 1$是初始设定的参数，q_0是一个随机数，$q \in [0,1]$，S是根据式(3.7)服从概率分布P_{ij}^{k}的随机变量，其它符号的定义如前。状态转移规则的引入可使蚂蚁在进行路径选择时，不仅可以利用前面同伴提供的反馈信息，还可以进行一定程度的随机探索操作。

(2) 局部信息更新规则。蚂蚁从城市i转移到城市j后路径ij上的信息素按照公式(3.9)进行更新：

$$\tau_{ij} = (1-\sigma)\cdot\tau_{ij} + \sigma\cdot\tau_0 \tag{3.9}$$

其中τ_0为常数，$\sigma \in (0,1)$为可调参数。

(3) 全局信息更新规则。针对全局最优解所属的边按公式(3.10)进行信息素更新：

$$\tau_{ij}(t+1)=(1-\rho)\cdot\tau_{ij}(t)+\rho\cdot\Delta\tau_{ij}^{gb} \tag{3.10}$$

$$\Delta\tau_{ij}^{gb}=1/L^{gb} \tag{3.11}$$

其中L^{gb}为当前全局最优长度。

2. 带精英策略的蚂蚁系统

带精英策略蚂蚁系统的基本思想就是：在每次循环结束后即对所有已经发现的最优解给予额外的信息素增强，这样做的目的是为了使到目前为止所找出的最优解在下一次循环中对蚂蚁更具有吸引力，当信息素水平被更新后，这条路径就可以看成已经被一定数量的所谓精英蚂蚁所走过，这条路径上的某些边可能就是最优解的一部分，该策略的目的就是为了在持续迭代中指导搜索的进行。该算法信息素根据下式进行更新。

$$\tau_{ij}(t+1)=\rho\tau_{ij}(t)+\Delta\tau_{ij}+\Delta\tau_{ij}^{*} \tag{3.12}$$

$$\Delta\tau_{ij}^{*}=\begin{cases}\sigma\cdot\dfrac{Q}{L^{*}}, & \text{若边}(i,j)\text{是所找出最优解的一部分}\\ 0, & \text{否则}\end{cases} \tag{3.13}$$

其中$\Delta\tau_{ij}^{*}$表示精英蚂蚁引起的路径(i,j)上的信息素量的增加；σ是精英蚂蚁的个数；L^{*}为所找出的最优解的路径长度；$\Delta\tau_{ij}$和$\Delta\tau_{ij}^{k}$的定义与蚁周公式相同。

在AS_{elite}中，将与精英蚂蚁所走过的路径对应的行程记为Tgb，这样的解被称为全局最优解(global-best solution)，当进行信息素更新时，对这些行程予以加权，同时，将找出这个解的蚂蚁称为精英蚂蚁(elitist ants)。从而增大较好行程的选择机会，这种算法能够以更快的速度获得更好的解。

3. 最大最小蚂蚁系统

最大一最小蚂蚁系统[29][30](Max-Min Ant System，MMAS)是德国学者Stützle和Hoos在1996年提出的一种目前最优的模型。这是目前为止解决TSP和QAP等组合优化问题最好的一类蚁群算法。与AS主要有三方面的不同：

(1) 在MMAS中只有一只蚂蚁用于在每次循环后更新信息素轨迹。

(2) 为避免搜索的停滞，将各条路径上可能的信息素限制在$[\tau_{min},\tau_{max}]$上，可以有效地避免某条路径上的信息素远远大于或小于其余路径情况的发生。

(3) 为使蚂蚁在算法的初始阶段更多地搜索新的解决方案，将信息素轨迹初始化为τ_{max}，而在AS中没有这样设置。

4. 蚁群混合策略

将蚁群算法和其它智能优化算法相融合，形成优势互补，是改进和完善蚁群算法的又一种重要途径。文献[31][32]将遗传算法和蚁群算法相结合，用遗传算法生成信息素分布，利用蚁群算法求精确解，形成了一种时间效率和求解效率都比较好的启发式算法；文献[33]把免疫思想引入到蚁群算法中，综合了蚁群和免疫系统的优点。

到目前为止，关于蚁群算法收敛性的证明并不多，Thomas[34]，孙焘[35]和段海滨[36]对蚁群算法的全局收敛性给予了初步的证明，该算法收敛性的理论研究将是蚁群算法的一个重要内容。

其它改进算法：Bulinheimer等人将遗传算法中的排序概念扩展应用到蚂蚁系统中，称为基于优化排序的蚂蚁系统[37]。近年来，国内学者在基本蚁群算法的基础上，也提出了一些改进算法。其中有代表性的主要有：吴庆洪等人提出的具有变异特征的蚁群算法[38]；郝晋等人提出的具有扰动特性的蚁群算法[39]；丁建立等人提出的遗传算法与蚂蚁算法融合的算法[40]。

3.2.5 蚁群优化算法的应用

蚁群优化算法主要用于求解不同的组合优化问题，一类为静态组合优化问题，另一类为动态组合优化问题。静态组合优化问题是指一次性给出问题的特征，在解决问题过程中，问题的特征不再改变。这类问题的范例就是经典旅行商问题。动态问题被定义为一些量的函数，这些量的值由隐含系统动态设置。因此，问题在运行时间内是变化的，而优化算法须在线适应不断变化的环境。这类问题的典型例子是网络路由问题。

1. 蚁群优化算法在静态组合优化问题中的应用

(1) 旅行商问题(Traveling Salesman Problem，TSP)[26][27]

蚁群优化算法首先应用于旅行商问题。TSP是组合优化中研究最多的NP难问题之一，许多研究表明，应用蚁群优化算法求解TSP优于模拟退火法、遗传算法、神经网络算法、禁忌搜索等多种优化方法。

(2) 二次分配问题(Quadratic Assignment Problem，QAP)[41][42]

二次分配问题是指把n台设备分配到n个地点，从而使得分配的代价最小，其中代价是设备被分配到位置上方式的函数。QAP是继TSP之后蚁群算法应用的第一个问题。

(3) 车间任务调度问题(Job-shop Scheduling Problem，JSP)[28]

JSP指已知一组M台机器和一组T个任务，任务由一组指定的将在这些机器上执行的操作序列组成。车间任务调度问题就是给机器分配操作和时间间隔，从而使所有操作完成的时间最短，并且规定两个工作不能在同一时间在同一台机器上进行。运用蚁群算法求解JSP，实验结果表明蚁群算法的求解性能优于遗传算法和模拟退火算法。

(4) 车辆路线问题(Vehicle Routing Problem，VRP)[43][44]

VRP来源于交通运输。已知M辆车，每辆车的容量为D，目的是找出最佳行车路线，在满足某些约束条件下使得运输成本最小。应用蚁群算法研究VRP，结果表明该方法优于模拟退火和神经网络，稍逊于禁忌算法。

(5) 有序排列问题(Sequential Ordering Problem，SOP)[45]

给定一个有向图，图上的弧和节点都加了权，服从于节点间先后次序的约束，SOP指在有向图上找出一个具有最优权值的哈密尔顿路径。SOP是NP难问题，应用扩展的蚂

蚁算法解决SOP，结果非常理想。

2. 蚁群优化算法在动态组合优化问题中的应用

在动态组合优化问题中，涉及到一些与静态优化组合不同的特征。通讯网络就是一个典型的例子，比如网络中信息的分布和计算分布具有随时间动态变化的特性。普通路由问题简单地说就是要建立一个路由表使得网络性能的一些量度最大化。将蚁群算法用于高速有向连接网络系统中，达到公平分配效果最好的路由。国内王颖、谢剑英等人提出了基于改进蚁群算法的多点路由调度方法[46]。

3. 蚁群优化算法在其它问题中的应用

蚁群优化算法在其它的组合优化问题中也取得了一些好的效果，其应用领域正在不断扩大。

(1) 电力系统

电力系统中的机组最优组合、电网规划、负荷分配等问题都是典型的组合优化问题。例如重庆大学郝晋将此算法应用到火电系统的机组最优组合选择问题中，并从动态规划中引入“状态”的概念，从而把选择机组的过程同蚂蚁选择城市的过程等同起来，取得了较好的效果[47]。

(2) 军事运筹

贺晋忠等人试图将该算法运用于军事领域，来解决武器-目标分配问题，当在允许使用的武器同敌方目标之间的联接上人为分布信息素后，就可以将武器选择目标的过程视为TSP的城市模式，结果显示同样可以取得较为理想的分配方案[48]。

(3) 连续空间优化及系统辩识

有人研究将主要用于离散空间优化问题的蚁群算法扩展到连续空间的寻优问题求解中，将离散问题中两点间信息素的修改规则扩展为影响附近的一个区域，各单蚁的信息量分布函数对整个解空间所处区域均有影响，只是影响程度随离各单蚁所在解空间位置距离的增加而递减。从而实现了在连续的解空间内逐步收敛至最优解所处的邻近区域。将此思想应用于系统辩识，可以实现系统的最优参数辩识。

3.3 其它启发式算法

3.3.1 邻域函数与局部搜索

邻域函数[23]是优化中的一个重要概念，其作用就是指导如何由一个（组）解来产生一个（组）新的解。邻域函数的设计往往依赖于问题的特性和解的表达方式（编码）。由于优化状态表征方式的不同，函数优化与组合优化中的邻域函数的具体方式将明显存在差异。

函数优化中的邻域函数的概念比较直观，利用距离的概念通过附加扰动来构造邻域函数是最常用的方式，如 $x'=x+??$，其中 x' 为新解，x 为旧解，$?$ 为尺度参数，$?$ 为满足

某种概率分布的随机数或白噪声或混沌序列或梯度信息等。显然，采用不同的概率分布（如高斯分布、柯西分布、均匀分布等）或下降策略，将实现不同性质的状态转移。

在组合优化中，传统的距离概念已不再适用，但其基本思想仍旧是通过一个解产生另一个解。邻域函数的的一般性定义是：

定义 3.7：令(S,F,f)为一个组合优化问题，其中 S 为所有解构成的状态空间，F 为 S 上的可行域，f 为目标函数，则一个邻域函数可定义为一种映射，即 $N:S\rightarrow 2^S$。其涵义是，对于每个解 $i\in S$，一些“邻近” i 的解构成 i 的邻域 $S_i\subset S$，而任意 $j\in S_i$ 称为 i 的邻域解或邻居。通常约定，$j\in S_i \Leftrightarrow i\in S_j$。

定义 3.8：若 $\forall j\in S_i\cap F$，满足 $f(j)=f(i)$，则称 i 为 f 在 F 上的局部极小解；若 $\forall j\in F$，满足 $f(j)=f(i)$，则称 i 为 f 在 F 上的全局最小解。

局部搜索算法是基于贪婪思想利用邻域函数进行搜索的，它通常可描述为：从一个初始解出发，利用邻域函数持续地在当前解的邻域中搜索比它好的解，若能够找到如此的解，就以之成为新的当前解，然后重复上述过程，否则结束搜索过程，并以当前解作为最终解。可见，局部搜索算法尽管具有通用易实现的特点，但搜索性能完全依赖于邻域函数和初始解，邻域函数设计不当或初值选取不合适，则算法最终的性能将会很差。同时，贪婪思想会使算法丧失全局优化的能力，即算法在搜索过程中无法避免陷入局部极小。因此，若不在搜索策略上进行改进，那么要实现全局优化，局部搜索算法采用的邻域函数将导致解的完全枚举。而这在大多数情况下是无法实现的，并且穷举法对于大规模问题在搜索时间上是不允许的。

鉴于局部搜索算法的上述缺点，智能优化算法，如模拟退火、遗传算法、禁忌搜索和混沌搜索等，从不同的角度利用不同的搜索机制和策略实现对局部搜索算法的改进，来取得较好的全局优化性能。

3.3.2 遗传算法

遗传算法[23] (Genetic Algorithm, GA)是基于“适者生存”的一种高度并行、随机和自适应的优化算法，它将问题的求解表示成“染色体”的适者生存过程，通过“染色体”群的一代代不断进化，包括复制、交叉和变异等操作，最终收敛到“最适应环境”的个体，从而求得问题的最优解或满意解。GA 是一种通用的优化算法，其两个最显著的特点则是隐含并行性和全局最优解空间搜索。作为强有力且应用广泛的随机搜索和优化方法，遗传算法可能是当今影响最广泛的进化计算方法之一。

通常，遗传算法的设计是按以下步骤进行的：

(1) 确定问题的编码方案；

(2) 确定适配值函数；

(3) 遗传算子的设计；

(4) 算法参数的选取；

(5) 确定算法的终止条件。

3.3.3 禁忌搜索算法

禁忌搜索[23] (Tabu Search，TS)的思想最早是由 Glover 在 1986 年提出，它是对局部邻域搜索的一种扩展，是一种全局逐步寻优算法，是对人类智力过程的一种模拟。TS 算法通过引入一个灵活的存储结构和相应的禁忌准则来避免迂回搜索，并通过藐视准则来赦免一些被禁忌的优良状态，进而保证多样化的有效探索以最终实现全局优化。相对于模拟退火和遗传算法，TS 是又一种搜索特点不同的元启发式算法。它重要的思想是标记对应已搜索到的局部最优解的一些对象，并在进一步的迭代搜索中尽量避开这些对象（而不是绝对禁止循环），从而保证对不同的有效搜索途径的探索。

简单 TS 算法的基本思想是：给定一个当前解（初始解）和一个邻域，然后在当前解的邻域中确定若干候选解；若最佳候选解对应的目标值优于“best so far”状态，则忽视其禁忌特性，用其替代当前解和“best so far”状态，并将相应的对象加入禁忌表，同时修改禁忌表中各对象的任期；若不存在上述候选解，则在候选解中选择非禁忌的最佳状态为新的当前解，而无视它与当前解的优劣，同时将相应的对象加入禁忌表，并修改禁忌表中各对象的任期。如此重复上述迭代搜索过程，直到满足停止准则。

一般而言，要设计一个禁忌搜索算法，需要确定算法的以下环节：

(1) 初始解和适配值函数

(2) 邻域结构和禁忌对象

(3) 候选解选择

(4) 禁忌表及其长度

(5) 藐视准则

(6) 集中搜索和分散搜索策略

(7) 终止准则

其中禁忌表大小、邻域函数结构与数量是影响禁忌搜索性能的关键因素。

简单禁忌搜索算法的具体步骤可描述如下：

(1) 给定算法参数，选取蚁群算法产生的较优解作为禁忌搜索算法的初始解 x，置禁忌表为空；

(2) 判断算法终止条件是否满足？若是，则结束算法并输出优化结果；否则，继续以下步骤；

(3) 利用当前解 x 的邻域函数产生其所有（或若干）邻域解，并从中确定若干候选解；

(4) 对每个候选解，判断藐视准则是否成立。若是，则用满足藐视准则的最佳状态 y 替代 x 成为新的当前解，即 x=y，以及用其替换“best so far”状态，并将相应的对象加入禁忌表，同时修改禁忌表中各对象的任期，然后转步骤(6)，否则，继续以下步骤；

(5) 对每个非禁忌对象对应的候选解，选择最佳候选解为新的当前解，并对相应于该候选解的禁忌对象设置禁忌值，同时更新其它禁忌对象的禁忌值；

(6) 转步骤(2)。

禁忌搜索是对人类思维过程本身的一种模拟，它通过对一些局部最优解的禁忌（也可以说是记忆）达到接纳一部分较差解，从而跳出局部搜索的目的。与传统的优化算法相比，TS 算法的主要特点为：

(1) 搜索过程中可以接受劣解，因此，具有较强的“爬山”能力；

(2) 新解不是在当前解的邻域中随机产生，或是优于“best so far”的解，或是非禁忌的最佳解，因此选取优良解的概率较大。

TS 算法是一种局部搜索能力很强的全局迭代寻优算法。但是 TS 也有明显的不足，比如：

(1) 对初始解有较强的依赖性，好的初始解可使 TS 在解空间中搜索到好的解，而较差的初始解则会降低 TS 的收敛速度；

(2) 迭代搜索过程是串行的，仅是单一状态的移动，而非并行搜索。

为了进一步改善禁忌搜索的性能，一方面可以对禁忌搜索算法本身的操作和参数选取进行改进，另一方面则可以与模拟退火、蚁群算法、遗传算法以及基于问题信息的局部搜索相结合。

3.4 本章小结

在现实世界中存在很多组合优化问题，这些问题都有一个共同的特征，随着问题规模的扩大，求解的复杂度呈现指数级增长，在现有条件下，通过穷举方法无法在多项式时间内求解。为了在求解性能和求解复杂度间取得一个平衡，人们提出了各种启发进化算法。而模拟生物界的某些生物机理的仿生进化算法，运用到一些组合优化问题的求解上，很好地解决了这个问题。

蚁群算法自产生以来表现出强大的生命力，较之以往的智能优化算法，不论是在搜索效率上，还是在算法的时间复杂度方面都取得了令人满意的效果。目前广泛应用于组合优化、人工智能等多个领域。较强的鲁棒性、寻径过程的并行性以及易于与其它启发式算法结合的优越性，使得蚁群算法吸引了越来越多研究者的注意，不断对其进行更深入的研究。初步的研究表明，蚁群算法在求解复杂优化问题方面尤其是离散优化问题，比目前广泛应用的遗传算法、模拟退火算法等进化算法具有一定的优势。

第 4 章 蚁群优化算法求解分布式系统任务分配问题

蚁群优化算法是近年新出现的一种从群体智能思想演变而来的新算法，也是目前国内外人工智能算法研究的热点。蚁群优化算法是人们对自然界中蚂蚁群体行为的研究而提出的一种基于种群的模拟进化算法，它模拟了真实蚂蚁群体搜寻食物的行为，成为解决NP难问题的锐利武器。目前其应用已涉及到诸多领域，如旅行商问题、调度问题等，并取得了良好的寻优效果。

任务分配问题是一类典型的组合优化问题。多处理器分布式系统上的任务分配问题是指：在分布式计算环境下，如何最有效地利用系统资源来完成一组有限的任务集合。这方面的研究结果在大规模数值计算、VLSI和计算机网络技术方面都有很好的应用背景。近年出现的启发式算法：禁忌搜索[49]以及蚂蚁算法[19]等为解决此类问题提供了新的途径。文献[19]运用改进的蚂蚁算法来求解简单的任务分配问题，即一个任务只能分配给一台机器处理，且每台机器只能处理一个任务。本文应用蚁群优化算法来解决分布式系统上约束条件更复杂的任务分配问题，即一个任务只能分配给一个处理机处理，而一个处理机可以处理多个任务，其中每个处理机都有固定成本和能力限制。仿真结果表明该算法比禁忌搜索算法和随机方法具有更好的求解能力。

4.1 任务分配问题的数学描述

研究任务分配问题时，对问题描述所给出的假设往往是有差别的。不失一般性，假设系统是非抢先式的，即同一处理机上的一个任务在没有完成前，其它分配到该处理机上的任务不能执行。本文所考虑的任务分配问题的模型[49]描述如下：

给定一个任务集合$T=\{1,2,\cdots,m\}$，一个处理机集合$P=\{1,2,\cdots,k\}$，v_i表示任务i的KOP(thousand operations per second)生产能力。处理机j的固定成本和能力分别用f_j和c_j表示。$e_{ii'}$表示任务i与任务i'之间的通讯开销，并且只有当任务i与任务i'被分配给不同的处理机时才有$e_{ii'}$，假设$e_{ii'}=e_{i'i}$。x_{ij}是0-1决策变量，$x_{ij}=1$意味着第i个任务由第j台处理机处理，$x_{ij}=0$意味着第i个任务不由第j台处理机处理。如果处理机j被使用过，则$y_j=1$；否则$y_j=0$。该问题的数学模型为：

$$Z=\min\left(\sum_{i=1}^{m-1}\sum_{i'=i+1}^{m}e_{ii'}\left(1-\sum_{j=1}^{k}x_{ij}x_{i'j}\right)+\sum_{j=1}^{k}f_jy_j\right) \tag{4.1}$$

$$s.t.\ \sum_{i=1}^{m}v_ix_{ij}\le c_jy_j,\quad j=1,\cdots,k \tag{4.2}$$

$$\sum_{j=1}^{k}x_{ij}=1,\quad i=1,\cdots,m \tag{4.3}$$

$$x_{ij}, y_j \in \{0,1\},\ \forall i = 1,\cdots,m,\ j = 1,\cdots,k \tag{4.4}$$

限制条件(4.2)保证了所有分配到处理机 j 的任务的KOP需求总和不会超过处理机 j 的能力 c_j，同样说明任意一台处理机可以处理多个任务。限制条件(4.3)确保了每个任务只能在一台处理机处理。我们的任务是求解如何分配处理机，才能在完成总任务的条件下，使得总的花费代价最小。

4.2 蚁群优化算法求解任务分配问题

为了将蚁群优化算法应用到任务分配问题中，可以将任务分配问题表示为一个完全二部图 $G = (T, P, E)$。对应于二部图一侧的 m 个结点，T 是 m 个点的集合，表示 m 个任务；对于二部图的另一侧的 k 个结点，P 是 k 个点的集合，表示 k 个处理机；E 是连接任务结点和处理机结点的边，$E = \{e_{ij} \mid i = 1,2,\cdots,m; j = 1,2,\cdots,k\}$。

在用于任务分配问题求解的蚁群优化算法中，每一个人工蚂蚁是具有下述特性的智能体：

(1) 当它选择把任务 i 指派给处理机 j 时，它会在连接 (i, j) 上留下一种称为信息素的物质 τ_{ij}；

(2) 它以一定的概率 P_{ijr} 为一指定任务 i 选择处理机 j，该概率为连接 (i, j) 上的启发式信息 η_{ij} 和的信息素量 τ_{ij} 的函数；

(3) 在构造一个完整的解时，已经被处理的任务被加以禁止，所有分配到处理机 j 的任务的KOP需求如果超过该处理机的能力限制，即 $c_j y_j - \sum_{i=1}^{m} v_i x_{ij} \le 0$，$j = 1,\cdots,k$，则该处理机也被禁止，如此重复直到所有的任务都已被处理机处理为止。

该启发式算法使用由 m 只蚂蚁组成的群体来一步步地进行解的构造。每只蚂蚁代表建立解的一个独立过程，这个过程分两步来完成：第一，蚂蚁选择将要被分配的任务；第二，对每个已经选择的任务，分配处理机执行它。若干蚂蚁过程之间通过信息素来交换信息，合作求解并不断优化。所有任务均被处理机处理完意味着蚂蚁建立解过程的结束。

类似于TSP，在分配之前，可先为每只蚂蚁建立一个数据结构，称为禁忌表。$tabu_r$ 表示第 r 只蚂蚁的禁忌表，$tabu_r(s)$ 是这个禁忌表的第 s 个元素。禁忌表 $tabu_r$ 记忆了已经指派过的任务，并在一次迭代结束（即构造了一个完整的解）前禁止蚂蚁为它们分配一个新的处理机。当一次迭代完成后，禁忌表被清空，而且蚂蚁又可以自由选择。

考虑该问题中每台处理机能力的限制，我们给每一台处理机定义一个标识表 $flag(j)$：如果 $c_j y_j - \sum_{i=1}^{m} v_i x_{ij} \ge 0$，则 $flag(j) = 1$，意味着处理机 j 可以处理任务 i；否则 $flag(j) = 0$，所有分配到处理机 j 的任务的KOP需求超过了该处理机的能力限制，处理

机 j 被禁止直到所有的任务都被处理完。

在每次迭代开始时，$\forall\ j=1,\cdots,k$，$flag(j)=1$，也就是说，所有处理机可以处理任意一个任务，当处理机 j 处理完某个任务 i 后，$y_j=1$，更新处理机 j 的容量 $c_j=c_j-v_i$，$i=1,\cdots,m$。当蚂蚁选择了第 $i+1$ 个即将分配的任务时，计算 c_j-v_{i+1}，如果 $c_j-v_{i+1}\geq 0$，则 $flag(j)=1$，任务 $i+1$ 可以被处理机 j 处理；否则 $flag(j)=0$，处理机 j 被禁止。当一次迭代完成后，标识表 $flag(j)$ 重新赋值为1。

第 r 只蚂蚁为第 i 个任务分配第 j 台处理机的概率是：

$$P_{ijr}=\frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{l\in allowed_r}\tau_{il}^{\alpha}\eta_{il}^{\beta}},\quad j\in allowed_r,\quad i=1,\cdots,m \tag{4.5}$$

其中，α，β 是体现信息素和启发式信息对蚂蚁决策重要性的两个参数，$allowed_r$ 是所有符合 $flag(j)=1$ 的处理机的集合，即可用处理机的集合，启发式信息 η_{ij} 用下式(4.6)进行计算：

$$\eta_{ij}=\frac{1}{cost((i,j)\cup A_{i-1})-cost(A_{i-1})} \tag{4.6}$$

其中，A_{i-1} 是第 r 只蚂蚁 $i-1$ 个任务的分配方案集合，$cost(A_{i-1})$ 表示 $i-1$ 个任务的分配方案所对应的花费代价，$cost((i,j)\cup A_{i-1})$ 表示 i 个任务的分配方案所对应的花费代价，η_{ij} 与将 (i,j) 添加到 A_{i-1} 时新增加的额外成本成反比。

在建立一个解决方案的过程中，蚂蚁应用式(4.7)的局部更新规则对它们所经过的连接 (i,j) 进行信息素更新。

$$\tau_{ij}\leftarrow(1-\gamma)\cdot\tau_{ij}+\gamma\cdot\tau_0 \tag{4.7}$$

其中 γ 是一个常数，$0<\gamma<1$。

在每只蚂蚁均完成了所有任务的分配之后，选择总的花费代价最小的那只蚂蚁，对其任务分配方案中连接 (i,j) 上的信息素按下式进行全局更新：

$$\tau_{ij}\leftarrow(1-\rho)\cdot\tau_{ij}+\Delta\tau_{ij} \tag{4.8}$$

$\rho(0<\rho<1)$ 为信息素的挥发系数，$1-\rho$ 表示随着时间的推移信息素浓度的残留程度；在较优分配方案中，如果任务 i 被分配给了处理机 j，则 $\Delta\tau_{ij}=Q/Z$，其中，Q 是一个常数，表示此方案中蚂蚁释放的信息素总量，默认情况下，Z 表示一次迭代中不同任务分配方案所对应的花费代价的最小值，即迭代最优解；而只在每固定循环次数（比如10次）使用全局最优解进行信息素更新。否则 $\Delta\tau_{ij}=0$。式(4.8)表明只有属于分配方案中连接 (i,j) 上的信息素才会得到增强。

上述过程重复直到满足终止条件，蚁群优化算法求解任务分配问题的过程描述如下：

begin 设置参数，初始化信息素浓度；

while 不满足终止条件时

for 蚁群中的每只蚂蚁

for 每个任务(直到所有任务都被处理完)

判断$allowed_r$中处理机的个数，蚂蚁根据式(4.5)为每个任务分配处理器；

根据式(4.7)对属于分配方案中的边进行局部信息素更新；

根据式(4.8)对属于分配方案中的边进行全局信息素更新；

end

end

4.3 仿真实验与结果分析

4.3.1 实例分析

实验环境：P4 PC机，内存为512M，操作系统为WinXP，开发软件为Delphi 7.0。用蚁群优化算法仿真计算时，给τ_{ij}赋相同的初始值1，最大循环次数$NC=1000$，$Q=100$，蚂蚁个数与任务总数相等，对α,β,ρ取不同值进行测试。参数的默认值设为$\alpha=2,\beta=3,\rho=0.5$，每次实验只有一个参数被改变，被测试的值为：$\alpha\in\{0,1,2,3,4,5\}$，$\beta\in\{0,1,2,3,4,5\}$，$\rho\in\{0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9\}$。

为了验证方法的可行性及效率，对不同规模的几组数据（使用文献[49]中产生随机问题集的方法得出）进行实验。用$m\times k$（m表示任务数，k表示处理机数）表示问题的大小，六个问题的规模分别为8×5、10×6、13×5、15×5、18×7和20×8。以规模为13×5的问题为例来说明蚁群优化算法的求解情况，测试数据为：任务的KOP生产能力$v=\{93,103,86,81,94,98,103,84,98,105,102,90,87\}$，处理机固定成本$f=\{564,1149,404,1461,192\}$，处理机的能力$c=\{282,383,202,487,192\}$，当两个任务分配给不同的处理机时产生的通信开销矩阵为：

$$c=\begin{bmatrix}
0 & 47 & 48 & 46 & 49 & 44 & 45 & 44 & 42 & 48 & 41 & 50 & 43\\
0 & 0 & 44 & 40 & 42 & 48 & 47 & 49 & 45 & 51 & 52 & 49 & 48\\
0 & 0 & 0 & 45 & 42 & 47 & 48 & 45 & 44 & 44 & 42 & 41 & 49\\
0 & 0 & 0 & 0 & 51 & 52 & 40 & 44 & 43 & 45 & 48 & 49 & 50\\
0 & 0 & 0 & 0 & 0 & 43 & 44 & 49 & 48 & 42 & 41 & 40 & 53\\
0 & 0 & 0 & 0 & 0 & 0 & 45 & 47 & 46 & 43 & 42 & 40 & 44\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 50 & 43 & 44 & 46 & 48 & 52\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 49 & 48 & 44 & 41 & 43\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 42 & 43 & 45 & 49\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 49 & 48 & 44\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 47 & 46\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 50\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

4.3.2 算法主要参数的选取

蚁群优化算法是一种新颖的仿生进化算法，这一性质决定了蚁群优化算法的参数选择对求解结果的优劣有很大的影响。因此，研究参数的最佳配置，对发挥蚁群优化算法在实际问题中的作用有很重要的意义。

在蚁群优化算法中，影响求解结果的参数主要包括以下几个：

1. 信息素启发式因子α；
2. 期望值启发式因子β；
3. 信息素挥发系数ρ；

下面就具体分析一下这几个参数对蚁群优化算法求解性能的影响。

1. 信息素启发式因子α和期望值启发式因子β

参数α反映了人工蚂蚁在运动过程中所积累的信息素(即残留信息素τ_{ij})在指导蚁群搜索中的相对重要程度，参数β则反映了人工蚂蚁在运动过程中期望信息η_{ij}在指导蚁群搜索中的相对重要程度。参数α的大小反映了蚁群在搜索过程中随机性因素作用的强度，其值越大，蚂蚁在以后的循环中选择之前循环走过的路径的可能性越大，搜索的随机性减弱，会导致搜索过早陷于局部最优(“停滞”现象)；参数β的大小反映了蚁群在路径搜索中先验性、确定性因素作用的强度，其值越大，蚂蚁在某个局部点上选择局部较优分配的可能性越大，虽然搜索的收敛速度得以加快，但蚁群在搜索过程中随机性减弱，易于陷入局部最优。

其次，算法的全局寻优性能，要求蚁群的搜索过程必须有很强的随机性；而算法的快速收敛性能，又要求蚁群的搜索过程必须要有较高的确定性。因此，两者对蚁群优化算法性能的影响和作用是相互配合、密切相关的，算法要获得优化解就必须在这二者之间选取一个平衡。取信息素启发式因子α和期望值启发式因子β不同数值的组合，参数α及β对算法性能影响的仿真结果如表4.1所示。

从实验结果不难看出，蚁群优化算法中信息素启发式因子α和期望值启发式因子β的不同选取对算法性能有极大的影响：当α过大时，相当于对二部图连接(i,j)上的信息素τ_{ij}在蚂蚁的搜索过程中的重要性给予充分的重视，蚂蚁完全依赖τ_{ij}的引导进行搜索，如果此时期望信息η_{ij}的启发式因子β相对也比较大，则将导致局部最优解上的信息正反馈作用极强，算法必将出现过早的停滞现象；当α过小时，相当于对二部图连接(i,j)上的信息素τ_{ij}在蚂蚁的搜索过程中的重要性未予足够重视，蚂蚁完全依赖η_{ij}的引导进行搜索，如果此时期望信息η_{ij}的启发式因子β相对也较小，则将导致蚁群陷入纯粹的、无休止的随机搜索中，在这种情况下所进行的搜索一般也很难找到最优解；适当地选择α及β的取值范围，蚁群优化算法均能获得较好的搜索结果。本例中$\alpha=1,\beta=2$或$\alpha=2,\beta=4$时算法取得较好的结果5929。

表 4.1 启发式因子α和β的不同组合对算法性能的影响

α \ β	0	1	2	3	4	5
0	—	6705	6705	6705	6705	6705
1	6375	5932	5929	6639	5943	6637
2	6351	5963	5952	5946	5929	5935
3	6351	5938	6639	5952	5951	5946
4	6351	6646	5946	6639	5952	5943
5	6351	6642	5932	5952	6638	5952

2. 信息素挥发系数ρ

在算法中，人工蚂蚁是具有记忆功能的，随着时间的推移，以前留下的信息素将要逐渐消逝。信息素挥发系数ρ的大小直接关系到蚁群优化算法的全局搜索能力及其收敛速度：由于信息素挥发系数ρ的存在，当要处理的问题规模比较大时，会使那些从来未被搜索到的路径(可行解)上的信息量减小到接近于0，因而降低了算法的全局搜索能力。而且当ρ过大时以前搜索过的路径被再次选择的可能性过大，也会影响到算法的随机性能和全局搜索能力；反之，通过减小信息素挥发系数ρ虽然可以提高算法的随机性能和全局搜索能力，但又会使算法的收敛速度降低。

因而，关于蚁群优化算法中信息素挥发系数ρ的选择，必须综合考虑算法的全局搜索能力和收敛速度两项性能指标，针对具体问题的应用条件和实际要求，在全局搜索能力和收敛速度两方面作出合理或折中的选择。ρ的不同取值对算法结果的影响如图4.1所示，由图4.1可见，当$\rho = 0.7$时该算法取得较好的结果。所以本例中，蚁群优化算法中信息素挥发系数的选择宜取$\rho = 0.7$。

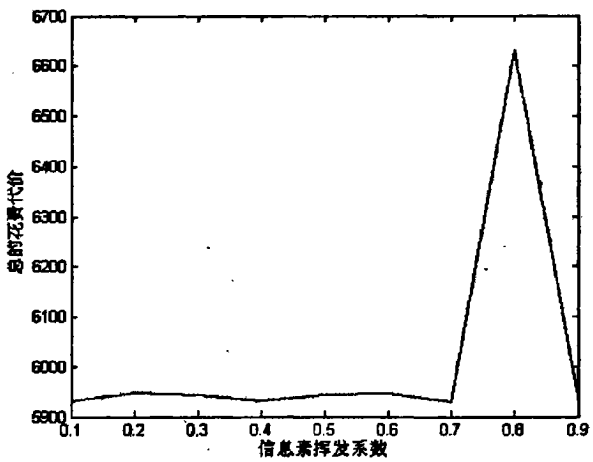

图 4.1 信息素挥发系数与总的花费代价的关系

在蚁群优化算法中，参数的设定是非常重要的环节，合理的参数可以缩短算法计算

的时间，保证最终解的可靠性和优良性。但是至今为止，还没有一种方法可作为确定参数值的最优策略，各个参数一般要凭借经验来选取。本例中，在完成了系统仿真程序编写的基础上，经过大量的实验，最终确定各参数值如下：蚁群数目为13，局部和全局信息素挥发系数为0.7，信息素更新参数Q取100，信息素启发式因子α和期望值启发式因子β取如下组合：$(\alpha=1,\beta=2)$或$(\alpha=2,\beta=4)$，各参数设定如下图所示：

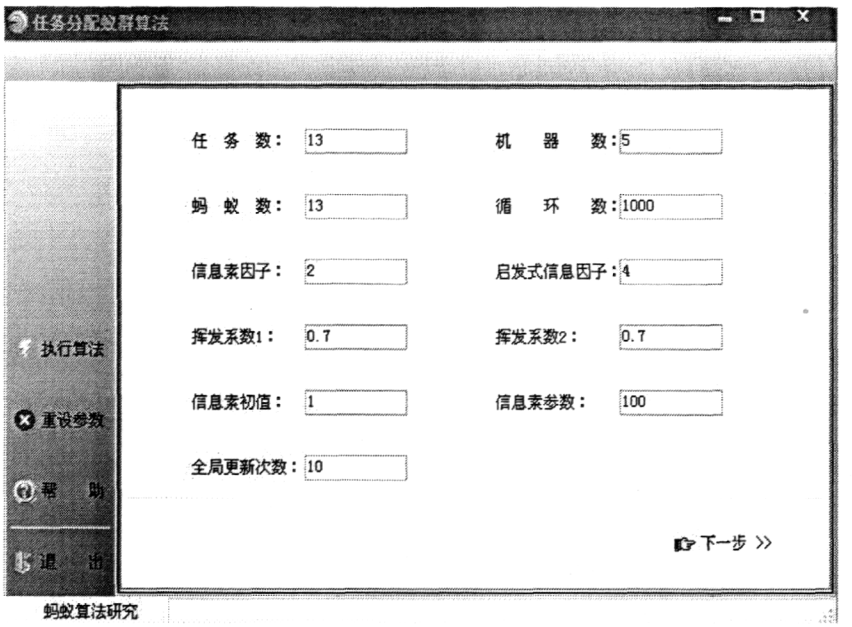

图 4.2 参数设定界面

程序运行结果如图4.3所示：

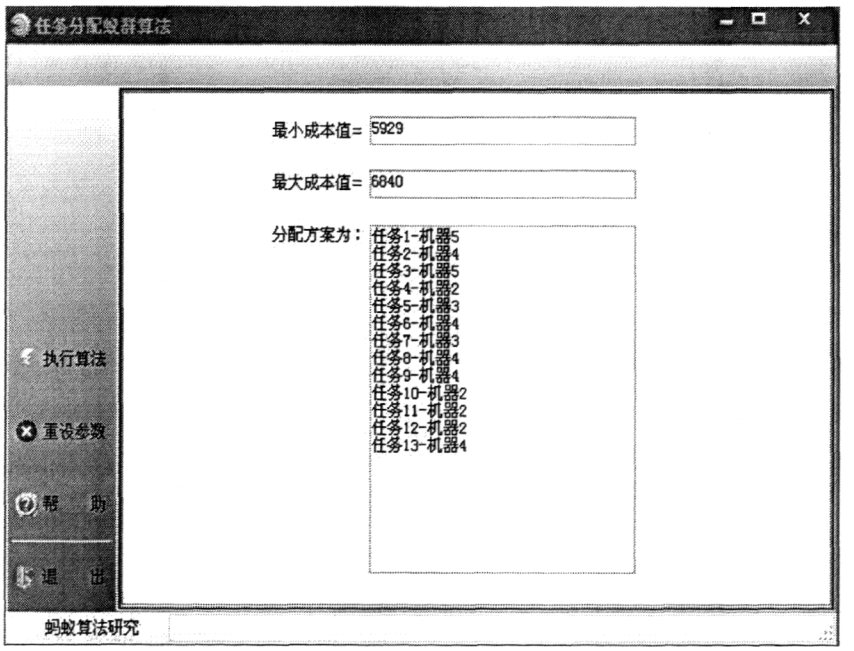

图 4.3 程序运行结果界面

4.3.3 仿真结果

对不同规模的问题运用蚁群优化算法求解之后，针对相同的数据集再分别应用随机

方法和禁忌搜索求解。禁忌搜索是是对局部领域搜索的一种扩展，是一种全局逐步寻优算法。在对禁忌搜索的实现中，禁忌长度设置为4，候选解的个数为8，最大迭代步数是100。仿真结果是10次实验的平均值。表4.2给出了三种算法在六个问题上的目标函数值。实验表明蚁群优化算法对不同规模的任务分配问题都有较好的结果。从这一系列实验结果可以发现，蚁群优化算法具有很强的发现较优解的能力。

表 4.2 不同算法的计算结果比较

规模	随机方法	禁忌搜索	蚁群优化算法
8×5	5838	6049	5807
10×6	5213	5397	4938
13×5	6658	6840	5929
15×5	7079	8677	6870
18×7	10251	10787	10443
20×8	14262	14498	13958

4.4 本章小结

本文应用蚁群优化算法求解分布式系统中约束条件更复杂的任务分配问题，通过仿真实验及比较分析，证明蚁群优化算法可以有效地解决任务分配问题。但是该算法也存在一些缺点，比如求解需要较长的搜索时间（本文蚁群优化算法的时间复杂度是$O(NC\cdot m^3\cdot k)$），而且该方法容易出现停滞现象，即搜索进行到一定程度后，所有个体所发现的解完全一致，不能对解空间进一步搜索，不能发现更好的解。鉴于此，可以考虑将蚁群优化算法与其它启发式算法，比如遗传算法、禁忌搜索、模拟退火算法等结合来求解任务分配问题，这一点值得进一步研究。

第 5 章 基于混合算法的分布式系统任务分配问题求解

任务分配问题是一类典型的组合优化问题。多处理器系统上的最优任务分配的研究是有效利用系统资源处理实际问题的热点课题，分布式计算系统中的一个根本问题是任务模块在处理机上的合理分配，以使总费用最小。在理论方面，由于任务分配问题是被公认的 NP 难问题[4]，所以如何构造有效的启发式算法或近似算法是目前研究的热点领域。

近年出现的启发式算法为解决此类问题提供了新的途径。其中，蚁群优化算法是受自然界蚂蚁觅食过程中，基于信息素的最短路径搜索食物行为启发而提出的一种智能优化算法。初步的研究表明，在求解复杂优化问题（特别是离散优化问题）方面该算法具有一定的优越性。文献[19]运用改进的蚂蚁算法来求解简单的任务分配问题。另外，近年来，混合优化策略得到了较广泛的一个应用，并取得了理想的效果，比如文献[50][51]分别运用结合蚁群优化算法和禁忌搜索算法的混合算法求解车间作业调度问题以及配电网规划问题。本文应用该混合算法来解决分布式系统上的任务分配问题。实验表明，与基本蚁群算法相比，该混合算法具有较好的性能。

5.1 问题定义

设在一个由 M 个处理机 $P=\{P_1,\cdots,P_M\}$ 组成的分布式系统中，$T=\{T_1,\cdots,T_N\}$ 是 N 个待处理的任务。假定 $W=\{w_1,\cdots,w_N\}$ 表示各个任务执行所需要的资源数量，$R=\{r_1,\cdots,r_M\}$ 是各个处理机拥有的最大资源数量，因此任务 T_i 可以被分配给处理机 P_j 当且仅当分配在处理机 P_j 上的任务的资源数量之和加 w_i 小于等于处理机 P_j 的最大资源数量，即 r_j。定义 K 为一个通信成本矩阵，其中，元素 k_{ijpq} 表示在处理机 j 上执行的任务 i 与分配在处理机 q 上的任务 p 之间的通信成本。v_j 表示处理机 j 相对于系统中最慢的处理机的处理速度，其中最慢处理机的速度是1。t_i 是任务 i 在最慢处理机上的执行时间，则 $t_j^e=\sum_{i:x_{ij}=1}t_i/v_j$ 表示处理所有分配在处理机 j 上的任务的时间和。设 X 是二进制矩阵，元素 $x_{ij}=1$ 表示第 i 任务分配给第 j 台处理机处理，$x_{ij}=0$ 则表示第 i 任务不由第 j 台处理机处理。本文所考虑的任务分配问题的数学模型为[52]：

目标函数 $f(X)$ 定义为：

$$f(X)=\min\left(\alpha_1\sum_{j=1}^{M}t_j^e+\alpha_2\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{\substack{p=1\\p\neq i}}^{N}\sum_{\substack{q=1\\q=j}}^{M}k_{ijpq}x_{ij}x_{pq}\right) \tag{5.1}$$

$$s.t.\quad \sum_{j=1}^{M}x_{ij}=1\qquad i=1,2,\cdots,N \tag{5.2}$$

$$\sum_{i=1}^{N} w_i x_{ij} \le r_j \qquad j=1,2,\cdots,M \tag{5.3}$$

其中α_1和α_2（$\alpha_1+\alpha_2=1$）表示目标函数中两个成本函数的相对重要性。限制条件(5.2)确保了每个任务只能在一个处理机上处理。条件(5.3)意味着每个处理机的资源限制条件必须满足。该系统上的任务分配问题是指在满足一定约束的条件下，将每个任务分配给其中的某个处理机，并使得相应的成本函数（总的执行时间与分配在不同处理机上的任务之间的通信成本之和）最小。

5.2 混合算法求解任务分配问题

蚁群优化算法是一种应用于组合优化问题的启发式搜索算法，它充分利用蚁群行为中所体现的正反馈机制进行求解，同时，利用分布并行计算方式在全局的多点进行解的搜索。将蚁群优化算法应用于旅行商问题等经典优化问题,取得了较好的效果。但是该算法也存在一些缺点：(1)当蚁群优化算法陷于停顿时，使某些不在最优解上的信息素得到有效增强，正反馈机制使得算法无法跳出局部状态；(2)与其它优化算法相比，蚁群优化算法搜索时间过长，特别是当初始信息匮乏时，需要大量的搜索时间。TS算法是一种局部搜索能力很强的全局迭代寻优算法，计算速度比较快，并且在整个搜索域内，TS算法能提供比其它算法更好的引导。但是TS算法也有其局限性：(1)解的质量高度依赖于算法的运行次数；(2)优化的效果依赖于初始解的好坏。

为此，将简单TS算法嵌入蚁群优化算法，构造混合优化算法。其基本思想是：信息素初始化；在每次迭代过程中，先产生每只蚂蚁的任务分配方案，比较每只蚂蚁分配方案所对应的目标函数值，选取最小的目标函数值作为蚁群优化算法的较优解，并记录该只蚂蚁的任务分配方案；简单禁忌搜索算法以蚁群优化算法得到的较优解作为初始解来开始其局部搜索，并将TS搜索后的较优解作为此次迭代较优解；最后信息素全局更新。算法一次迭代的流程如图5.1所示。

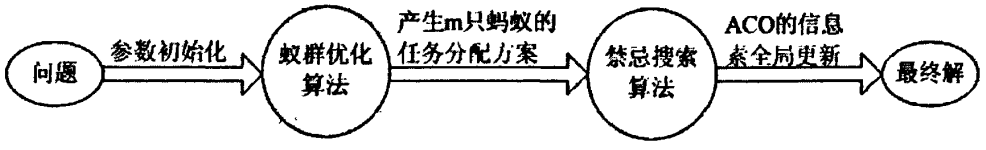

图 5.1 混合算法求解任务分配问题的总体框架

5.2.1 问题的图形表示

本文提出的算法首先将被求解的问题抽象为有向图，如图5.2所示。借助于图5.2，可以产生图5.1中m只蚂蚁的任务分配方案。在图5.2中，第一层只包含一个结点$root$，每只蚂蚁都从该结点开始遍历，其余的N层分别代表N个任务可允许选择的M个处理机。每一层上的处理机结点个数依$\sum_{i=1}^{N} w_i x_{ij}$与$r_j$的大小关系动态变化，若$\sum_{i=1}^{N} w_i x_{ij} \le r_j$，则

第 j 个处理机结点保留；否则，遍历时第 j 个处理机结点不予考虑。每只蚂蚁构造候选解的过程是从 *root* 结点出发，依次遍历每一层上的一个结点（若第 i 层选择了第 j 个结点，则意味着第 i 个任务分配给了第 j 个处理机）。一组蚂蚁如此反复地在图上运动，最终得到处理机的一组序列即任务对应的分配方案。

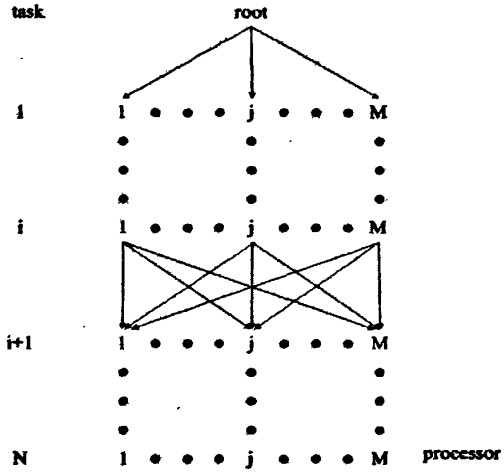

图 5.2 任务分配问题的图形表示

5.2.2 混合算法求解

初始时刻，图5.2所有可能的边上均被赋予一定的信息素初始值 τ_0。

在每一次迭代过程，蚁群中的每只蚂蚁均要产生问题的解。在解的建立过程中，不失一般性，假设第 r 只蚂蚁遍历了位于第 i 层的第 j 个结点，接着按照伪随机比例（pseudorandom proportional）规则选择位于第 $i+1$ 层的第 k 个结点，这个规则由如下公式给出：

$$k=\begin{cases}\arg\max\limits_{l\in allowed_r}\left\{\tau_{ijl}^{\alpha}\eta_{i+1,l}^{\beta}\right\} & \text{若} q\le q_0\\ S, & \text{否则}\end{cases} \tag{5.4}$$

其中，q 是均匀分布在区间[0,1]中的一个随机变量，$q_0(0\le q_0\le 1)$ 是一个参数，S 是根据式(5.5)给出的概率分布产生出来的一个随机变量，$allowed_r$ 是满足式(5.3)要求的处理机集合。

$$P_{ijk}=\begin{cases}\dfrac{\tau_{ijk}^{\alpha}\eta_{i+1,k}^{\beta}}{\sum\limits_{l\in allowed_r}\tau_{ijl}^{\alpha}\eta_{i+1,l}^{\beta}}, & \text{若} i=0,1,\ldots,N-1; k\in allowed_r\\ 0, & \text{否则}\end{cases} \tag{5.5}$$

其中，α，β 是体现信息素和启发式信息对蚂蚁决策重要性的两个参数，$\eta_{i+1,k}$ 可按式(5.6)计算：

$$\eta_{i+1,k}=\frac{1}{cost(A_{i+1})},\ i=0,1,\cdots,N-1;\ k=1,2,\cdots,M \tag{5.6}$$

其中A_{i+1}是$i+1$个任务的分配方案集合，$cost(A_{i+1})$是根据$f(x)$计算出的$i+1$个任务分配方案所对应的成本值，$\eta_{i+1,k}$与$i+1$个任务分配方案所对应的成本值成反比。与真实蚁群不同，人工蚁群系统具有记忆功能。为了满足蚂蚁分配任务时处理机的资源约束限制(5.3)，为每只蚂蚁都设计了一个数据结构，称为禁忌表$tabu_r$。$tabu_r$用来记录违背资源约束限制(5.3)的处理机，不允许该蚂蚁在本次迭代中再选择这个处理机来处理任务。当本次迭代结束后，禁忌表被清空，该蚂蚁又可以自由地进行选择。

在所有蚂蚁完成了所有任务的分配之后，运用简单禁忌搜索算法改进解。

邻域函数、禁忌对象、禁忌表和藐视准则的设计是禁忌搜索算法的关键，所以在算法的设计和实现之前，分别定义禁忌搜索算法所需的几个参数：

1. 初始解(Initial solution)

禁忌搜索算法以蚁群优化算法得到的较优解作为初始解来开始其局部搜索，并令该初始解为当前最好解。

2. 邻域函数(Neighborhood function)

邻域函数为可应用于当前解的移动的集合。禁忌搜索从一个初始解开始，迭代地搜索这个解的邻域，使解从当前解移到一个新的邻域，目的是使评价函数最小。常用的邻域操作有互换(swap)、插入(insert)、逆序(inverse)等，本文采用互换法来实施邻域操作。其邻域操作方法为：

(1) 互换操作指从按顺序排列的任务序列$(1,2,...,N)$中选择两个元素，比如s和u，若任务s和任务u被分配到的处理机$A2M[s]$和$A2M[u]$不同，并且满足$A1S[A2M[s]]-A1W[s]+A1W[u]\geq 0$与$A1S[A2M[u]]-A1W[u]+A1W[s]\geq 0$，则交换任务$s$和$u$对应的处理机$A2M[s]$和$A2M[u]$。其中，$A2M[s]$表示执行任务$s$的处理机，$A2M[u]$表示执行任务$u$的处理机，$A1S[A2M[s]]$表示执行任务$s$的处理机$A2M[s]$上的剩余资源数量，$A1S[A2M[u]]$表示执行任务$u$的处理机$A2M[u]$上的剩余资源数量，$A1W[s]$是任务$s$执行时所需的资源数量，$A1W[u]$是任务$u$执行时所需的资源数量。

(2) 考虑到邻域搜索的效率，通常仅取当前解的邻域的一个子集作为候选解集。至于最佳候选解集的选取，一般的做法是选择候选解集中满足藐视准则或非禁忌的最优解。

3. 禁忌表(Tabu list)

禁忌表是为避免循环和陷入局部最优解而采用的一种具有短期记忆功能的方法。根据本文采用的邻域操作方法，将禁忌表设定成禁忌长度$l_1=3$的先进先出的循环队列。

4. 藐视规则(Aspiration criterion)

算法采用了两条藐视规则：

(1) 基于适配值的准则。若某个禁忌候选解的适配值优于“best so far”状态，则解

禁此候选解为当前状态和新的“best so far”状态；

(2) 基于最小错误规则。如果迭代后得到的候选集中所有对象都被禁忌时，且都不比以前所取得的解好，此时从候选集的所有元素中选一个评价值最小的状态解禁。

5. 评价函数(Evaluation function)

禁忌搜索算法的评价函数是候选集合元素选取的一个评价公式，也是用于对搜索状态的评价，进而结合禁忌准则和藐视准则来选取新的当前状态。本文直接采用目标函数作为评价函数。

6. 终止准则(Termination criterion)

禁忌搜索中的停止规则有两种：(1)设置最大迭代次数作为停止算法的标准；(2)在给定迭代次数内所搜索的最好解不再改进时，算法就停止。本文的终止准则选用(1)。

使用简单禁忌搜索算法改进解之后，信息素按下式进行全局更新：

$$\tau_{ijk} = (1-\rho)\cdot\tau_{ijk} + \Delta\tau_{ijk} \tag{5.7}$$

$\rho(0 \le \rho \le 1)$为信息素的挥发系数，$1-\rho$表示随着时间的推移信息素浓度的残留程度；在较优的遍历路径中，如果任务i被分配给了处理机j，同时任务$i+1$被分配给了处理机k，则连接(j,k)上的信息素得到增强，即$\Delta\tau_{ijk} = Q/f_{\min}$，其中，$Q$是一个常数，表示蚂蚁释放的信息素总量；$f_{\min}$表示该任务分配方案所对应的成本值；否则$\Delta\tau_{ijk} = 0$。

5.3 仿真实验

为证明混合算法解决分布式系统任务分配问题的有效性，对使用文献[52]中的方法产生的规模$N \times M$（N表示任务数，M代表处理器个数）分别为8×4、10×4、15×5、20×5、30×6的几组数据分别运用基本蚁群算法和混合算法进行仿真实验。实验的硬件环境是Pentium4、2.80GCPU、512M内存，软件环境是Windows XP操作系统。基于程序框图5.1，本文以Java为编程工具对算法进行了模拟实现。

仿真实验中基本蚁群算法的参数设置：信息素浓度初始值$\tau_0 = 0.55$，单位信息素浓度$Q = 1$，蚂蚁数与任务数相等，启发式因子α和β以及信息素挥发度ρ的缺省值为$\alpha = 1, \beta = 3, \rho = 0.5$。混合算法另加禁忌搜索算法实现所需的几个参数：禁忌长度为3，候选解的个数为5，最大迭代步数是20。另外，目标函数$f(X)$中的参数$\alpha_1 = 0.6, \alpha_2 = 0.4$。

对于混合算法，从采用的伪随机比例规则我们不难发现，影响算法求解性能的主要因素为：α, β, ρ这三个参数的合理设定。在实验中，我们固定这三个参数中的两个不变，而改变第三个参数。这些参数的测试集合为$\alpha \in \{0,1,2,3,4,5\}$，$\beta \in \{0,1,2,3,4,5,10,15\}$，$\rho \in \{0.1,0.3,0.5,0.7,0.9\}$。如此反复试测，最终确定$\alpha, \beta, \rho$这三个参数的较优组合及算法的计算结果。比如对于15个任务，5个处理器的任务分配问题，与其它α, β, ρ的不同组合相比，当$\alpha = 2$，$\beta = 3$，$\rho = 0.7$时，混合算法取得较好的结果215.35。

针对五组数据分别采用基本蚁群算法和混合算法进行了仿真，两种算法都运行1000次，表5.1给出了两种算法的目标函数值。从中可以看出，蚁群优化算法与禁忌搜索算法

相融合的混合算法具有更好的搜索性能。

表 5.1 混合算法与基本蚁群算法计算结果比较

任务数	处理机数	基本蚁群算法	混合算法
8	4	57.999	52.8
10	4	93.125	86.25
15	5	225.725	215.35
20	5	390.917	381.717
30	6	950.625	938.125

根据上表数据，对不同规模（任务数×处理机数）的任务分配问题，运用matlab绘制了两种算法的计算结果对比图一图5.3。

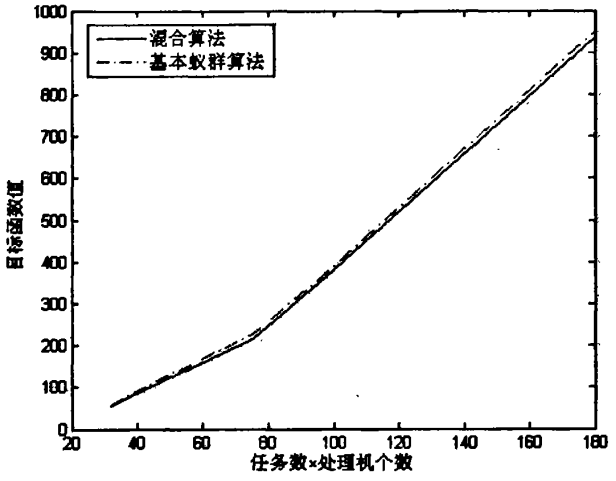

图 5.3 混合算法和基本蚁群算法的性能比较

5.4 本章小结

本文运用混合算法求解分布式系统的任务分配问题。该算法成功地将简单禁忌搜索算法作为局部搜索策略嵌入蚁群优化算法，为蚁群优化算法提供了有效的搜索信息，扩大了搜索范围，弥补了蚁群优化算法易陷入局部最优的不足。通过仿真实验，验证了结合蚁群优化算法和简单禁忌搜索算法各自特点的混合算法，使两种算法的搜索能力得到相互补充，弥补了各自缺点，是一种优化能力、效率和可靠性较高的优化方法。

结论和展望

蚁群优化算法是从真实蚂蚁的觅食行为中受到启发而发展起来的一种新型的模拟进化算法，它具有并行性，所有“蚂蚁”独立行动；它具有协作性，蚂蚁通过信息素进行间接通讯、相互协作来寻找问题的最优解；它具有鲁棒性，只要对算法稍作修改，就可以运用于别的优化问题。目前，该算法在组合优化、函数优化、机器人、路径规划、数据挖掘等领域的应用已显示出其求解复杂优化问题的优势。

本文主要做了以下工作：

1. 对近年来被广泛应用的、具有代表性的仿生进化算法的算法原理、特点、研究现状等进行了综述，详细说明了蚁群优化算法的基本原理、模型实现、特点、改进算法以及发展现状。

2. 应用蚁群优化算法求解分布式系统中约束条件更复杂的任务分配问题，即一个任务只能分配给一个处理机处理，而一个处理机可以处理多个任务，其中每个处理机都有固定成本和能力限制。将该任务分配问题表示成完全二部图，应用蚁群优化算法来解决该问题，选择不同规模的几组数据进行实验，对每一组数据，通过反复试测探索了信息素挥发系数ρ，信息素启发式因子α和期望值启发式因子β的合理设定，并将所得的实验结果与其它算法作比较，结果表明蚁群优化算法比禁忌搜索算法和随机方法具有更好的求解能力。

3. 将另一任务分配问题抽象为一种新的有别于二部图的图形表示形式。针对蚁群优化算法求解问题时易陷入局部最优，出现停滞现象的不足，利用简单禁忌搜索算法和蚁群优化算法的优点，将简单禁忌搜索算法融入到蚁群优化算法中，得到了一种适于求解分布式系统任务分配问题的混合算法。从实验结果可以看出该混合算法能改进基本蚁群算法的计算结果的质量。

本文的研究成果在于拓宽了蚁群优化算法的应用领域，为一些复杂困难的任务分配问题提供了新的求解方法。

虽然经过十几年的发展，ACO 算法已经被证明是一个求解优化问题(特别是组合优化问题)的有效工具。但对该算法的研究还远未像遗传算法、人工神经网络等成熟，另外，由于本人的知识水平和研究时间有限，本文在算法和实现方面也存在着一些缺陷和不足，需要在日后作进一步研究：

1. 在运用简单禁忌搜索算法改进解时，为了提高简单禁忌搜索算法的性能效率，可以使用长期记忆策略来加强或者多元化搜索过程。

2. 针对算法本身的改进与完善仍将是以后蚁群优化算法在应用中的重要研究方向。基本蚁群算法中易出现停滞现象、搜索时间长等缺点，今后应不断改进算法性能，提升算法通用性。

3. 蚁群优化算法的参数是影响其求解性能和效率的关键因素，信息素挥发系数ρ，信息素启发式因子α和期望值启发式因子β等都是非常重要的参数，其选取方法和选取原则直接影响到蚁群优化算法的全局收敛性和求解效率。蚁群优化算法中的参数设定有很多经验因素，目前还没有很好的规律，只能依靠反复试测。如何设置参数使蚁群优化算法在任务分配问题中求解性能更好有待进一步研究。

4. 蚁群优化算法的理论研究方面还存在着一些问题需要进一步解决，比如初始化参数选择问题、算法的收敛性研究等，这些均带有经验性和直觉性，至今没有经过严格的数学论证。

在此，希望我们所作的研究能够对蚁群优化算法和组合优化问题的求解有所推动。

参考文献

[1] Tanenbaun A S. Distributed Operating Systems. 陆丽娜. 北京: 电子工业出版社, 1999

[2] 陈国良, 吴俊敏, 章锋等. 并行计算机体系结构. 北京: 高等教育出版社, 2002

[3] 何炎祥. 分布式操作系统. 北京: 高等教育出版社, 2005, 74-88

[4] Kasahara H, Narita S. Practical multiprocessor scheduling algorithms for efficient parallel processing. IEEE trans on Computers, 1984, C-33 (11): 1023－1029

[5] Stone H S. Multiprocessor scheduling with the aid of network flow algorithms. IEEE Trans on Software Eng, 1977, SE-3(1): 85-93

[6] Stone H S. Critical load factors in two-processor distributed computer systems. IEEE Trans on Software Eng, 1978, SE-4(3): 254-258

[7] Kuolin H. Allocation of processors and files for load balancing in distributed systems. Ph. D Thesis, Berkeley: Univ of California, 1985

[8] Chu W W. Task allocation in distributed data processing. Computer, 1980, 13(11):57-69

[9] Efe K. Heuristic models of task allocation scheduling in distributed systems. IEEE Computer, 1982, 15(6): 50-60

[10] Adam T L. A comparison of list scheduling for parallel processing systems. Commu of ACM, 1974, 17(12): 685-690

[11] Coffman E G Jr. Computer and job-shop scheduling theory. New York: Wiley, 1976

[12] Shen C C, Tsai W H. A graph matching approach to optimal task assignment in distributed computing systems using a minmax criteion. IEEE Trans, on Computers, 1985, C-34(3): 197-203

[13] Kobrosly W. Implementation of an optimal task allocation strategy. Southern Tier Technical Conference, 1988, Proceedings of the 1988 IEEE, 1988: 212-219

[14] Szymanek R, Kuchcinski K. Task assignment and scheduling under memory constraints. Euromicro Conference, 2000. Proceedings of the 26th, 2000, 1(1): 84-90

[15] 张聪, 马义忠. 异构计算机系统中基于遗传算法的任务分配与调度. 微电子学与计算机, 2004, 24(6): 74-78

[16] 钟求喜, 谢涛. 基于遗传算法的任务分配与调度. 计算机研究与发展, 2000, 37(10)

[17] Hamam Y, Hindi K S. Assignment of program modules to processors: A simulated annealing approach. European Journal of Operational Research, 2000(122): 509-513

[18] Attiya G, Hamam Y. Task allocation for maximizing reliability of distributed systems: A simulated annealing approach. Journal of Parallel and Distributed Computing, 2006(66): 1259-1266

[19] 杨冬, 王正欧. 改进的蚂蚁算法求解任务分配问题. 天津大学学报, 2004, 37(4): 373-376

[20] Maetal P R, A task allocation model for distributed computing system. IEEE Trans on Computer, Vcl. 31. No. 1. Jan, 1981

[21] Dorigo M, Stützle T. Ant colony optimization. Cambridge: MIT Press, 2004

[22] 邢文训, 谢金星. 现代优化计算方法. 北京: 清华大学出版社, 1999. 8

[23] 王凌. 智能优化算法及其应用. 北京: 清华大学出版社, 2004, 12-16

[24] Dorigo M. Optimization, learning and natural algorithms. Ph. D. thesis, Dipartimento di Eletronica, Politecnico di Milano, Italy, 1992

[25] Dorigo M, Maniezzo V, Colorni A. Ant system: optimization by a colony of cooperation agents. IEEE Transactions on Systems, Man, and Cybernetics-Part B, 1996, 26(1): 28-41

[26] Dorigo M, Gambardella L M. Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans on Evolutionary Computation, 1997, 1(1): 53-66

[27] Dorigo M, Gambardella L M. Ant colonies for the traveling salesman problem. Technical Report/IRIDIA/96-3, Belgium: Universite Libre de Bruxelles, 1996

[28] Colorni A, Dorigo M, Maniezzo V. Ant system for job-shop scheduling. Belgian Journal of Operations Research, Statistics and Computer Science, 1994, 34(1): 39-54

[29] Stützle T, Hoos H. The max-min ant system and local search for the traveling salesman problem. In T. Baeck, Z. Michalewicz, X. Yao, editors, Proceedings of IEEE-ICEC-EPS'97, IEEE International Conference on Evolutionary Computation and Evolutionary Programming Conference, IEEE Press, 1997: 309-314

[30] Stützle T, Hoos H. Improvements on the ant system: introducing max-min ant system. In Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms, Springer-Verlag, Wien, 1997: 245-249

[31] Marcin L P, Tony W. Using genetic algorithm to optimize ACS-TSP. Proc of 3rd In Workshop ANTS[C]. Brussels, 2002: 282-287

[32] 邵晓魏, 邵长胜, 赵长安. 利用信息量留存的蚁群遗传算法. 控制与决策, 2004, 19(10): 1187-1189

[33] Lee Z J, Lee C Y, Su S F. An immunity-based ant colony optimization algorithm for solving weapon-target assignment problem.Applied Soft Computing, 2002(2):39-47

[34] Thomas S, Dorigo M. A short convergence proof for a class of ant colony optimization algorithms. IEEE Trans on Evolutionary Computation, 2002, 6(4): 358-365

[35] 孙焘, 王秀坤, 刘业欣等. 一种简单蚂蚁算法及其收敛性分析. 小型微型计算机系统, 2003, 21(8): 1562-1564

[36] 段海滨, 王道波. 蚁群算法的全局收敛性研究及改进. 系统工程与电子技术, 2004,

26(10), 1506-1509

[37] Bullnheimer B, Hard R F, Strauss C. A new rank-based version of the ant system: a computational study. Technical Report Pom-03/97, Institute of Management Science, University of Vienna, 1997

[38] 吴庆洪, 张纪会, 徐心和. 具有变异特征的蚁群算法. 计算机研究与发展, 1999, 36(10): 1240-1245

[39] 郝晋, 石立宝, 周家启. 求解复杂TSP问题的随机扰动蚁群算法. 系统工程理论与实践, 2002(9): 88-91

[40] 丁建立, 陈增强, 袁著祉. 遗传算法与蚂蚁算法的融合. 计算机研究与发展, 2003, 40(9): 1531-1536

[41] Maniezzo V, Colorni A. The ant system applied to the quadratic assignment problem. IEEE Transactions on Data and Knowledge Engineering, 1999, 11(5): 769-778

[42] Gambardella L M, Taillard é D, Dorigo M. Ant colonies for the quadratic assignment problem. Journal of the Operational Research Society, 1999, 50(2): 167-176

[43] Bullnheimer B, Hartl R F, Strauss C. An improved ant system algorithm for the vehicle routing problem. Annals of Operations Research, 1999(89): 319-328

[44] Gambardella L M, Taillard é D, Agazzi G. MACS-VRPTW: a multiple ant colony system for vehicle routing problems with time windows. In D. Corne, M. Dorigo, and F. Glover, editors, New Ideas in Optimization, McGraw Hill, London, UK, 1999: 63-76

[45] Gambardella L M, Dorigo M. HAS-SOP: a hybrid ant system for the sequential ordering problem. Technical Report IDSIA-11-97, IDSIA, Lugano, Switzerland, 1997

[46] 王颖, 谢剑英. 一种基于改进蚁群算法的多点路由算法. 系统工程与电子技术, 2001, 23(8): 98-101

[47] 郝晋. 蚁群优化算法及其在电力系统短期发电计划中的应用研究. 重庆: 重庆大学, 2002

[48] 贺晋忠, 王明赞, 赵雪花. 蚁群算法在武器-目标分配问题中的应用. 华北工学院学报, 2004, 25 (S1): 244-247

[49] Chen W H, Lin C S. A hybrid heuristic to solve a task allocation problem. Computers & Operations Research, 2000(27): 287-303

[50] 宋晓宇, 王丹. 求解Job shop调度问题的混合蚁群算法研究. 计算机工程, 2007, 33(4): 218-220

[51] 陈根军, 唐国庆. 基于禁忌搜索与蚁群最优结合算法的配电网规划. 电网技术, 2005, 29(2): 23-27

[52] Sanz S S, Xu Y, Yao X. Hybrid meta-heuristics for task assignment in heterogeneous computing systems. Computers & Operations Research, 2006(33): 820-835

致 谢

在本论文完成之际，我要特别感谢我的导师张远平教授在这三年学习和工作中给予我的悉心指导，以及对我生活上的无微不至地关怀和照顾。张老师严谨的治学态度和亲切宽厚的作风令我深受感动，并时时激励我在学习和工作中不断进取。

同时，我要感谢实验室里和我朝夕相处的同学。感谢他们在生活和学习上对我的关心和鼓励；他们对我的研究工作提出了很好的建议，与他们的讨论使我受益非浅；他们孜孜不倦认真学习的态度和勤勤恳恳的工作作风共同维护了实验室中良好的研究环境与学术风气。

最后，感谢我的亲人对我的亲切关怀和鼓励，正是他们的关爱与支持才使我有了不断前进的动力。

附录 A 攻读学位期间发表的论文

[1] 王灵霞, 张远平, 吴佩莉. 蚁群算法求解分布式系统任务分配问题. 计算机工程与设计, 2008, 29(6): 1472-1474

[2] 唐明珠, 张远平, 杨佳, 王灵霞. 基于背景知识的KNN文本分类. 计算机科学, 2007, 34(10A): 102-104