大连海事大学

学 号 1120140423

姓 名 曹立禄

学 院 交通运输管理学院

学 科 管理科学与工程

指导教师 薛大伸

日 期 2016.xx

# 第1章 绪论

## 1.1 研究背景

近年来，计算机技术和互联网技术得到了极大的发展。国家总理李克强在国家创新技术大会上创新性的指出，要让互联网技术同传统行业相结合的发展，提出互联网加的战略。但同时，随着技术的高速发展，数据量也在急剧增加，从互联网刚起步的时候以M字节作为计算单位，到G字节作为计算单位，在数据量上有了质的提升。但是现如今要用T级别甚至P级别才能描述当今数据量的庞大。因此，如何更加高效、更加快速的对这些数据中进行分析处理，从而挖掘出对我们的生产发展有价值，可利用的知识，成为了当今国家各类研究机构以及企业讨论的热门话题。为了解决这种局面，云计算技术的出现及兴起顺应了时代的需求和发展。在传统的数据处理框架中，一般海量的数据都是由大型机来处理的。这种处理框架的缺点在于成本较高，维护比较复杂，并且很难扩展。而云计算一改传统的大型服务器处理的方式，所提倡的分布式计算框架和并行分析计算方式能够充分利用廉价的服务器，并且具有维护方便，极易扩展的优势。目前最具有代表性的大数据处理框架是Apache的Hadoop大数据分析计算平台，其特有的文件存储系统hdfs能将大数据按照一定机制分别存储在不同的服务器节点上，并且有多个备份。在数据存储速度以及容错性上有极大的优越性，并且其并行处理框架MapReduce可以充分利用各个廉价服务器节点的性能，使得对大数据的并行处理成为可能。同时也使得以前需要大量计算而不能推广的算法求解框架也变成了现实，比如遗传算法。

遗传算法的编程思想来自于生物群体的遗传变异和进化规律。生物进化的目的在于保持种群的稳定和增加种群的适应性，以达到能与生存环境和谐发展。对于种群进化中的遗传变异属性则使种群具有优胜劣汰的属性，保证种群的繁荣发展。种群在进化的过程之中，每一次进化都将最适应环境的个体保留下来，而不适应环境的个体将遭到淘汰，这样保持了种群的最优解和适应度总是处于最优水平。因此根据遗传算法的特点，它非常适合于解决大规模求解问题，比如集群作业调度、非线性求解问题等。另外，由于遗传算法特有的并行计算的特性，使得它可以通过群体分组的方式进行计算求解。

从上述论述可知，遗传算法有着得天独厚的优势可以运行在目前最流行的并行分析计算框架Hadoop平台上。如今，在信息数据急剧增加的时代，很多运行良好的传统系统也逐渐面临着数据量过多无法有效处理的瓶颈。另一方面，对于数据分析统计也成为当前的热门话题，比如用户行为分析，决策支持系统等等。因而大数据处理平台的出现结合传统的算法模型使得需要进行大量数据处理的大规模求解问题，成为可能。另外，随着我国社会经济的快速发展及其带来的数据量和约束条件的增多。将传统算法结合大数据处理平台进而解决当前传统求解框架无法应对的问题，也是适应时代进步的需要。

## 1.2 国内外研究现状

早在20世纪50年代遗传算法被首次提出之后，经过将近十年的时间也就是20世纪60年代，美国的密歇根大学的John Holland教授可是着手从自然行为与人工系统的自适应性方面开始应用遗传算法。一开始，他想创造一种应用程序与机器的理论，想要使得程序和机器人对自然具有相当的适应能力，于是他意识到使用群体方式进行自然搜索、进化、选择和交换的重要性。后来，经过将近十多年的发展，也就是到20世纪70年代到20世纪80年代中期的时候，基于逻辑数学和语言智能的人工智能开始兴盛起来，从而很多著名大学的学者开始怀疑基于自然搜索进化的人工智能学说，但是Holland教授依旧坚持这一方面的研究。之后Bagley发明了“遗传算法”一词并且沿用至今，后来他创新型的提出了一种双倍体的编码方式并对以后的遗传算法的各种算子比如交换算子、变异算子等的发展起到了很大的促进作用。同时，他还敏锐的观察到采用何种方式能方式整个算法早熟的原理，并且发展了自组织的遗传算法概念。

Cavicohio 1970年研究了基于遗传算法的子程序选择和模式识别问题，在模式识别问题上，采用整数编码，检索空间很大，他提出了以预选择策略保证群体多样性，对遗传法参数进行中心控制的方法。同年，Weinberg研究了生物体的计算机仿真，他的贡献在于提出运用多层遗传算法来进行遗传算法的参数自优化。1968至1971年，Holland提出了重要的模式理论，建议采用二进制编码。与前面几位博士不同，Holland首次采用二进制编码来研究函数优化问题，并指出了运用Gray码的一些优点，他研究了从生物系统引申出的各种不同的选择和配对策略。1972年，Frantz的博士论文中研究了许多新的问题，如基因非线性(异位显性)现象，基因迁移操作及多点交换操作等，由于没有设计出诸如GA－deception之类合适的非线性优化问题，实验结果并不具备说服力。这一年，Holland的模式理论也渐趋成熟，但在编码策略上出现了至今仍执争论的二派，一派根据模式定理建议用尽量少的符号编码，一派以数值优化计算的方便和精度为准采用一个基因一个参数的方法，并把相应的基因操作改造成适合实数操作的形式，Bosworth，Zoo和Zeigler是后者的开创者。1975年竖立了遗传算法发展史上的两块里程碑，一是Holland出版了经典著作《Adaptation in Nature and Artificial System》，该书是作者十几年间许多思想及其实现的结晶，详细阐述了遗传算法的理论，并为其奠定了数学基础，发展了一整套模拟生物自适应系统的理论;二是De Jong完成了具有指导意义的博士论文《An Analysis of the

Behavior of a Class of Genetic Adaptive System》，他深入领会了模式定理并做了大量严格的计算实验，给出了明确的结论，他还建立了著名的De Jong五函数测试平台，定义了性能评价标准，并以函数优化为例，对遗传算法的六种方案的性能及机理进行了详细实验和分析，他的工作成为后继者的范例并为以后的广泛应用奠定了坚实的基础。为克服De Jong的轮盘赌(随机)选择操作中的随机误差，Brindle 于1981年在她的博士论文中研究了六种复制策略。 进入八十年代，随着以符号系统模仿人类智能的传统人工智能暂时陷入困境，神经网络、机器学习和遗传算法等从生物系统底层模拟智能的研究重新复活并获得繁荣。Goldberg在遗传算法研究中起着继往开来的作用，他在1983年的博士论文中第一次把遗传算法用于实际的工程系统—煤气管道的优化，从此，电子科技大学硕士论文 遗传算法的理论研究更为深入和丰富，应用研究更为广泛和完善，自1985年起，遗传算法及其应用国际会议每二年召开一次，有关人工智能的会议和刊物上大多有遗传算法的专题，Goldberg的课本，及其他学者的专著的出版有力地推动了遗传算法的传播。

进入九十年代，以不确定性、非线性、时间不可逆为内涵，以复杂问题为对象的科学新范式得到学术界普遍认同，如广义进化综合理论。由于遗传算法能有效地求解属于NPC类型的组合优化问题及非线性多模型、多目标的函数优化问题，从而得到了多学科的广泛重视，一些学者也认识到求解复杂问题最优解是不现实的，故而寻求满意解，而遗传算法是最佳工具之一，生物进化的历史比任何数学证明都强有力，问题是遗传算法在吸收遗传学、进化论及分子生物学最新成果和在实验得到证明和证伪的同时本身也在进化。

目前，遗传算法引起包括数学、物理、化学、生物学、计算机科学等领域的科学家的极大兴趣。遗传算法的研究领域和内容十分的广泛，如遗传算法的设计与分析，遗传算法的理论基础及其在各个领域的应用。随着理论的深入研究和应用领域的不断拓展，遗传算法必将取得更大的发展。

Internet 的发展促使研究学者们必须处理多约束规则的大数据信息，而且处理这些数据信息可行的方法只有采用可增长数据（大数据）的处理架构。遗传算法逐渐被用于大规模数据问题求解，像非线性优化、聚类和作业调度等。进化算法的内部并行特征使得它们对于并行化成为优先的候选者。虽然大多数并行进化计算算法的研究可以应用于实践，但是在在可增长数据的并行进化算法研究还很少。

2006 年南洋理工大学计算机科学院的 Dudy Lim 等人在论文中提出GE-HPGA(Grid-Enabled Hierarchical Parallel Genetic Algorithm)算法及算法模型，此算法是基于网格计算提出，并基于标准网格计算技术实现。此模型有两个显著的特征，第一个是扩展的 GridRPC API 隐藏了复杂的网格计算环境，第二个是无缝的资源发现和选择调度器。

2009 年伊利诺伊大学计算机科学系的 Abhishek Verma 等人在论文中首次提出并行化遗传算法在 Map/Reduce 中实现的思想。论文中介绍了第二个并行遗传算法实现和两个并行实现算法的比较结果。并且第二个并行遗传算法采用 Hdaoop 的Map/Reduce 编码实现。在这篇论文中的主要研究点是提出了遗传算法并行化的新模型，并采用 Hadoop的 Map/Reduce 实现此模型。

随后马里兰大学计算机科学系的Di-Wei Huang等人用Map/Reduce实现遗传算法用于求解销售作业调度。通过实验证明大规模种群下需要更少的迭代次数就能达到最优值。

在2012年，来自四川大学的李剑峰和彭舰教授讲改进的遗传算法用在云计算的环境中并加上了很多任务调度的任务。他们提出了一种新的双适应度的遗传算法的仿真实验，很好的证明了这种新的方法不仅可以在短时间内完成任务调度任务，并且可以在平均完成时间上也做了相应的优化。

在2013年3月份，中科院的两位院士提出来一种全新的编码策略，这种策略创新型的提出了如何将计算机资源作为遗传算法运算过程中的基本单元，从而在此基础上提出了一种区域杂交算子和变异算子，并且基于Hadoop大数据分析计算平台进行研究在大数据平台上的实现也分析。根据大量的实验证明给予Hadoop大数据分析计算平台上的进行有效作业调度的可行性。

## 1.3 研究目的和意义

Hadoop大数据处理平台在当今社会提供了一种新的资源处理方式，它将数据分布式存储在其各个服务器节点上，并就其架构来说具备高可用，高容错，存取效率高、扩展性强等特点。这使得当今计算机技术飞速发展下的数据量急剧增加不再成为阻碍生产力发展的瓶颈。相应的很多经典算法比如本文研究的遗传算法是很多领域问题求解的解决框架。但是很多时候，由于数据量的过大，使得算法求解显得力不从心。传统的做法在于增加计算服务器的性能（比如CPU，内存等设备），但是这样做的缺点在于增加了维护成本，并且扩展性也很不灵活。

如今很多科学领域的经典求解算法慢慢的已经发展成为大数据的处理问题。例如娱乐圈的用户行为分析，互联网大文本的处理运算，天文学、地理学问题、基因重组，更高层面的军事卫星导航、军事侦察问题。

针对如上提出的各类问题，Hadoop大数据平台与传统算法模型的结合有着深远意义。它不但可以帮助传统无法解决的问题提供一种有效的解决方案，同时可以为社会的发展和生产力的提高产生重要的推动作用。虽然理论和研究现在还处于技术前沿阶段，投入真实的实践环节还需要一段实践，但是这同时可以为后续的大力研究产生重要的指导意义。综上所述，将传统的遗传算法与最新规定Hadoop大数据分析计算平台相结合进行讨论研究，不仅仅对社会与经济的发展具有理论指导意义，同时也具有相当的现实意义和应用价值。

本文以Hadoop大数据计算平台为技术，利用传统遗传算法的并行特性，将其与Hadoop大数据分析计算平台相结合，提出了新的遗传算法的求解框架。同时，本文选用传统的TSP（旅行商）问题作为框架的验证案例，进一步说明两者结合正确性。并且比较传统算法，在遗传算法的全局与局部收敛速度，是否能保持种群物种的多样性，能得到全局最优解而不是局部最优解等方面进行系统的比较说明，证明框架的优越性。

## 1.4 论文的研究内容和研究思路

本文在查阅大量国内外研究成果及文献之后，首先专注于研究Hadoop大数据分析计算平台的实现及其工作原理，并搭建简易的Hadoop大数据运行平台。紧接着对传统遗传算法在解决传统问题领域的应用做了相当的研究，之后对遗传算法在Hadoop大数据分析计算平台的上并行的可能性做了深入的论证，并根据传统问题解决方案的不足，提出了在最新的Hadoop大数据分析计算平台上搭建本文提出的混合并行遗传算法框架。之后利用其求解TSP问题做了框架的可行性验证。主要的研究内容如下：

（1） 对最新的Hadoop大数据分析计算平台及其数据库Hbase进行深入研究，并搭建简易的计算平台。

（2） 对传统的遗传算法的求解过程做了深入的论述并详细分析了遗传算法并行运算的可能性。

（3） 根据对Hadoop大数据分析计算平台和遗传算法的深入研究，提出了基于Hadoop大数据平台的混合并行遗传算法模型，并通过TSP问题进行了编程实现及论证分析。

（4） 通过得出的求解数据与传统的求解框架进行对比分析，从而阐述改进后的计算框架的优点。

（5） 对最新提出的基于Hadoop大数据分析计算平台的混合并行遗传算法求解框架进行总结并提出其中的不足和未来需要优化改进的方面。

本文将传统计算平台上的遗传算法求解问题进行分析研究并将其在最新的Hadoop大数据分析计算平台上实现。并通过实验平台数据充分证明了本文研究内容在处理持续增长数据的可行性。具体论文结构如下：

第一章提出本文的研究内容及其所需要用到的技术，并详细阐述了研究内容的国内外研究背景。最后给出了本论文的内容及研究思路安排。

第二章着重研究Hadoop大数据分析计算平台以及基于Hadoop平台的Hbase数据库。主要包括Hadoop平台的架构搭建及其MapReduce计算核心。

第三章主要是对传统的遗传算法如何进行求解运算的研究，以及对于TSP问题进行深入的介绍，包括其传统的求解思想和框架。

第四章为本文的核心章节之一，核心内容主要是根据上述各个章节研究的成果，创造性的提出了基于Hadoop大数据分析计算平台的混合并行遗传算法求解框架。

第五章是本文的另一个核心章节，本章主要是将上一章节提出的并行遗传算法求解框架应用在TSP问题上，在经过反复实验之后得出实验数据，从而验证了本文研究内容的可行性。

第六章主要针对以上的研究内容和思路进行反思和总结，并提出研究过程中的不足已经将来要做的改进。

# 第2章 Hadoop大数据平台

## 2.1 Hadoop大数据平台背景

#### 2.1.1 Hadoop大数据平台的产生背景

随着互联网技术的迅速发展，数据量的爆发式增长。Google的传统的搜索存储方式已经不能很好的适应这种环境。因此为了解决这个问题，Google内部实现了一套针对自身运营和发展的大数据平台，这个平台主要由三部分组成，GFS、MAPREDUCE和BIGTABLE。他们分别是分布式的文件存储系统、数据处理框架和基于分布式存储系统的列式存储数据库。

大约 2000年，谷歌将其三项技术以论文的形式发表出来，Apache的一位员工Doug Cutting基于这三篇论文产生了创建大数据平台的灵感。因此他基于Apache已有的项目Lucune和Nutch，加上Google大数据平台的实现思想，创建了现在的大数据平台Hadoop。

Hadoop大数据平台和Google分布式大数据平台一样，也是由三个核心部分组成，它们分别是分布式文件存储系统hdfs、数据分析框架MapReduce和基于分布式文件存储系统的列式存储数据库Hbase。

传统的数据存储及其分析方式一般是通过大型机进行批量存储，以及通过建立存储索引，链接分析等方式进行数据的处理。这种方式的缺点在于随着数据量的逐渐增长，扩展大型机的存储将越来越困难，维护成本也会越来越高，并且扩展性越来越差，并且对于中小型企业来说很难承受起昂贵的运行和维护费用，从而使得在大数据时代很难发挥出群体的智慧，在一定程度上阻碍了技术的发展。

自从Hadoop出现之后，一改传统的数据存储及其分析方式，数据将会以文件的形式分布存储在各个成本较低的服务器上，并且通过其特有的MapReduce分析计算框架来进行大规模的数据分析。这样就使得数据量的爆发式增长对于服务器来说将不再成为瓶颈，只需要增加价格低廉的服务器即可进行数据的无限扩展，从而也使得中小型企业也能很好的参与进大数据时代的数据分析与计算工作。这将为新时代的技术发展提供得天独厚的优势。

#### 2.1.2 Hadoop 大数据平台生态圈

随着Hadoop大数据平台的出现，基于Hadoop大数据平台的技术如雨后春笋般蓬勃发展，因此出现了基于Hadoop大数据平台的生态圈。这些技术极大地促进了大数据平台的发展，同时也极大的增加了用户的使用方便度，使得基于Hadoop的大数据平台成为了当今主流的大数据存储和分析计算平台。以下列出集中主要的Hadoop生态圈技术。

Pig：Haoop的整个运行过程是任务制的，而pig则是基于Hadoop而产生的一种编程语言，它的出现极大的简化了Haoop的任务分发机制。另外，对于半结构化的数据而言，比如日志文件，pig的出现使其变得更加具有现实意义。同时，pig的扩展性极强，它可以在日常的编程中自定义添加java的数据类型并进行转换。

Hive：对着Hadoop的文件式存储系统hdfs系统的逐步发展，基于hdfs的数据仓库也在慢慢兴起，比如Hbase。但是如何将传统的结构化比如传统的二维表数据直接映射到数据仓库中是Hadoop生态圈亟待解决的问题。Hive的出现则解决了这个问题，Hive作为一个数据仓库工具的一种实现，其提供了简单的sql查询功能，并且可以直接让sql语句在MapReduce中运行使用。其学习成本不高，可以十分方便的扩展数据仓库的功能。

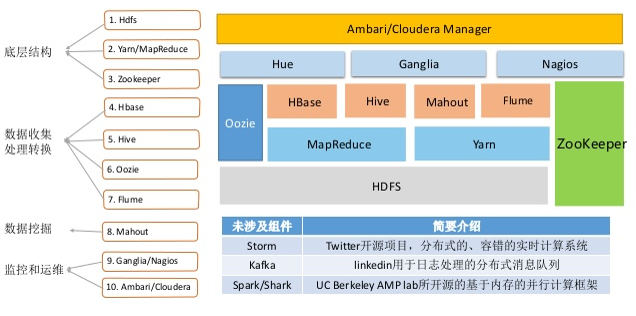
Mahout：Mahout也是Apache的一个开源项目之一，从属于Apache Software Foundation（ASF），它的核心领域为机器学习中的经典算法的实现。其出现的目的在于帮助开发人员更加方便的构建智能化的应用程序。常见的Mahout实现包括聚类分析，频繁子项挖掘，筛选过滤分析等。另外，Mahout也在部署在云端进行使用，可以很好的同云计算进行集成应用。

Zookeeper：Zookeeper 是一个源码开放的、分布式的应用协调服务。它以Fast paxos 算法为基础，实现Hadoop分布式系统的同步服务，配置维护和命名服务等分布式应用。

## 2.2 Hadoop核心组件及其文件系统

#### 2.2.1 Hadoop核心组件

从狭义上来区分，Hadoop的核心组件主要分为三个部分，HDFS（Hadoop Distributed File System）分布式文件系统、YARN（Yet Another Resource Negotiator）资源管理调度系统和MapReduce分布式运算框架。



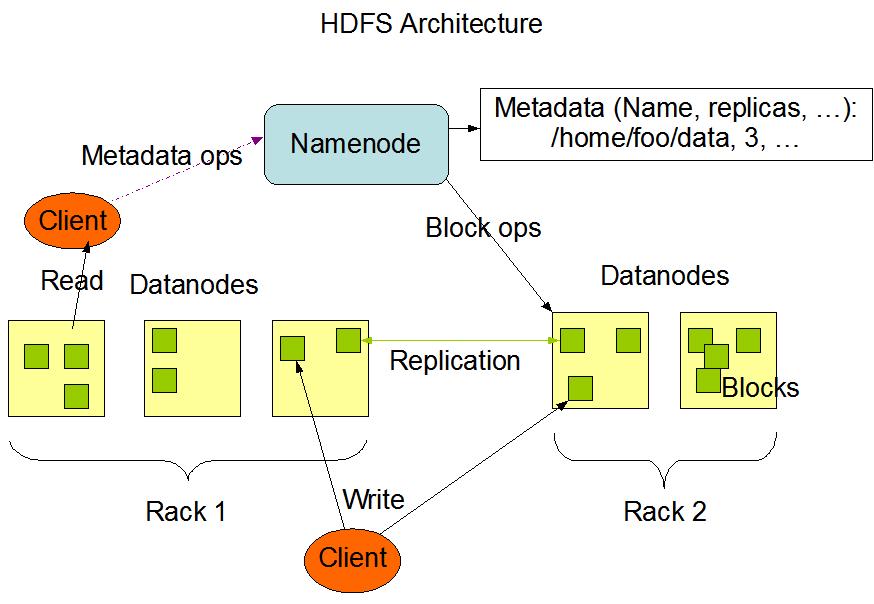
HDFS：hdfs是Hadoop大数据平台的分布式数据存储系统，这种存储系统与现在传统的存储系统有很大的相似点，但同时，它有自己特有的优势。比如，它有很强的容错性，这使得它可以部署在廉价的机器上。并且，hdfs提供了可以承载高吞吐量的数据访问能力，这使得hdfs非常适合大规模数据的分析和计算。

YARN：yarn是Hadoop大数据平台上的硬件资源协调者，负责平台的硬件资源分配、调控等功能。可以为Hadoop的上层应用提供可靠的资源统一调度和管理。这使得Hadoop大数据平台在集群使用效率上更加可靠、有效和统一。

MapReduce：是Hadoop大数据平台上特有的数据分析计算框架，主要用于 TB级别的大数据分析计算。这种计算方式极大的简化的编程人员在并行编程方式下的编程和运行。

#### 2.2.2 Hadoop文件式存储系统hdfs及实现机制

Hadoop的文件存储系统hadfs是一个分布式的数据存储系统，它为分布式数据分析提供了最底层的存储基础，也是Hadoop大数据平台的核心之一。Hdfs是基于本机文件存储系统的抽象层，如同Windows的NTFS文件存储格式屏蔽磁盘读写细节一样，hdfs文件存储系统则屏蔽了各个存储节点的文件存储细节，使得分布式存储如同在一台机器上存储一样。通过分布式存储处理来解决大数据文件的存储问题，具有分布式管理，性能高，容错性高等特点。



如上图所示，整个Hadoop的文件存储系统主要是主从结构。具体分为主节点（namenode）和多个从节点（datanode）。其主要负责功能如下：

namenode节点主要负责接收用户的发出的操作请求、对所存储的文件系统和目录结构索引进行相应的维护工作，另外，对文件系统和block之间的关系、block和 datanode之前的关系进行管理和维护。

而datanode的主要任务则是复制实际的文件存储，保证所存储的文件被切分为block存储在相应的磁盘上。此外，这些文件将会被复制成为多个副本，这极大的保证了数据的安全性。

#### 2.2.3 Hadoop的MapReduce分析计算框架

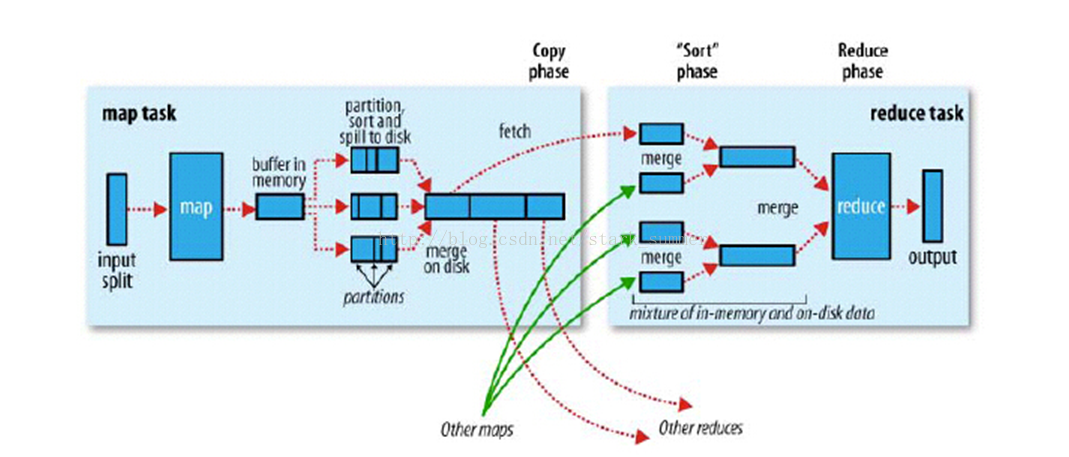
在传统的计算机的分析运算中，常规的思想是将数“数据”移向“计算”去进行数据的分析和统计，这样做的好处在于实现简单，符合常规的思考方式，从而也使得编程人员容易思考和掌握。但是这种方式的缺点在于，由于现今的I/O以及网络传输的瓶颈，这种方式不适合大规模数据的分析和计算，在当今大数据背景下显得力不从心。

而最新的Hadoop大数据计算平台一改传统的由“数据”移向“计算”的分析计算方式 ，演变成了既有传统的计算方式，又结合从“计算”移向“数据”的计算方式。正如其命名，它的计算步骤主要分为两个步骤，Map数据统计计算和Reduce数据分析阶段。

Map阶段正是从“计算”移向“数据”的分析计算，这种计算方式充分发挥了分布式系统的优越性，可以使得所有的分析计算节点一同进行数据的分析和统计，之后将分析统计的结果，通俗来说，其实就是过滤的结果一同交给接下来的Reduce框架。

Reduce分析计算框架的主要任务是将map阶段的计算结果进行数据汇总和分析。如同map阶段一样，Reduce分析计算框架也是由很多节点组成的，在实际的数据分析计算中，可以根据业务的具体需求和环境，将不同的map阶段完成的结果分发给不同的Reduce节点来进行数据的分析和统计。

MapReduce的图解如下：



综上所述，Hadoop的MapReduce分析计算框架主要有以下优点：

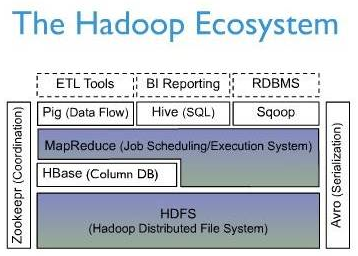
第一，在实际的业务运行中，它将一般的业务在技术层面划分成了两个部分，Map数据分析过滤阶段和Reduce数据计算统计阶段，这样使得在大数据量的环境下可以进行更加细致的数据处理，并且充分发挥的分布式计算的优点。

第二，MapReduce分析计算框架将分布式计算中面临的公共问题，比如jar包的分发，任务的启动，任务的容错，节点之间的任务调度，中间节点的分组传递等封装成了计算框架。使得编程开发人员无需关注整个框架的底层是如何实现的，只需要在框架层面上专注于业务的开发即可，这样的使得整个开发过程变得更加可靠，节省了大量的数据处理成本和时间。最重要的是，编程开发人员无需过多的考虑此框架是否稳定可靠，因为框架的稳定性和可靠性已经有专门的人员去开发维护。

## 2.3 基于 Hadoop的列式存储数据库Hbase

#### 2.3.1 Hbase简介及其表结构

Hadoop来自于Google的bigtable，它是基于hdfs文件存储系统而存在的，提供高可性能、高可靠、列存储、可伸缩、实时读写的数据库系统。如同传统的关系数据库与Windows文件存储系统的关系一样，Hbase是一种架设在hdfs文件系统上的非关系型数据库。传统的数据库分为RDBMS和NOSQL，而Hbase则是介于两者之间的为大数据而生的数据库，其查询只能通过主键（main key）和主键的range来查询检索里面的数据，对于事物的支持度比较低，仅支持单行事物。因此其数据结构既不是结构化的也不是松散的，而是介于两者之间的。



由于文件存储系统的存储缺点，比如随机检索比较慢，复杂的较高等缺点，不适合结构型数据的分析计算，因此Hbase的出现解决了此类问题。

Hbase和传统的关系型数据库不同，它是一种列式存储结构的数据库。其主要目标是依靠横向扩展，通过不断增加廉价的服务器来增加其运算和存储的能力。其表结构都是按照列为单位来进行数据存储的，即每一行是一个记录，并且每一行的字段数是任意的，每一个字段都要存储字段名和字段值。

Hbase数据库一般具有以下几个特点：

1、可以存储海量数据。其一个表的存储字段可以达到上亿行，百万列级别。因此它是专门用来处理大数据的数据库。

2、 面向列。Hbase是面向列族的数据存储和权限控制，每一个列族有其独立的索引系统来增强数据检索的效率。

3、稀疏存储。对于数据为空（null）的列并不占用存储空间，因此，表的设计可以非常灵活。这个特性极大的增强了Hbase数据库在大数据结构化松散业务中的应用。

Hbase的数据逻辑视图如下：

HBase以表的形式存储数据。表有行和列组成。列划分为若干个列族(row family)

|  |  |  |  |
| --- | --- | --- | --- |
| **Row Key** | **column-family1** | **column-family2** | **column-family3** |
| **column1** | **column1** | **column1** | **column2** | **column3** | **column1** |
| **key1** | **t1:abc t2:gdxdf** |  | **t4:dfads t3:hello t2:world** |  |  |  |
| **key2** | **t3:abc t1:gdxdf** |  | **t4:dfads t3:hello** |  | **t2:dfdsfa t3:dfdf** |  |
| **key3** |  | **t2:dfadfasd t1:dfdasddsf** |  |  |  | **t2:dfxxdfasd**  **t1:taobao.com** |

**Row Key**

与nosql数据库们一样,row key是用来检索记录的主键。访问hbase table中的行，只有三种方式：

1 通过单个row key访问

2 通过row key的range

3 全表扫描

Row key行键 (Row key)可以是任意字符串(最大长度是 64KB，实际应用中长度一般为 10-100bytes)，在hbase内部，row key保存为字节数组。

存储时，数据按照Row key的字典序(byte order)排序存储。设计key时，要充分排序存储这个特性，将经常一起读取的行存储放到一起。(位置相关性)

注意：

字典序对int排序的结果是1,10,100,11,12,13,14,15,16,17,18,19,2,20,21,…,9,91,92,93,94,95,96,97,98,99。要保持整形的自然序，行键必须用0作左填充。

行的一次读写是原子操作 (不论一次读写多少列)。这个设计决策能够使用户很容易的理解程序在对同一个行进行并发更新操作时的行为。

**列族**

hbase表中的每个列，都归属与某个列族。列族是表的chema的一部分(而列不是)，必须在使用表之前定义。列名都以列族作为前缀。例如*courses:history* *，* *courses:math* *都属于* *courses* 这个列族。

访问控制、磁盘和内存的使用统计都是在列族层面进行的。实际应用中，列族上的控制权限能 帮助我们管理不同类型的应用：我们允许一些应用可以添加新的基本数据、一些应用可以读取基本数据并创建继承的列族、一些应用则只允许浏览数据（甚至可能因 为隐私的原因不能浏览所有数据）。

**时间戳**

HBase中通过row和columns确定的为一个存贮单元称为cell。每个 cell都保存着同一份数据的多个版本。版本通过时间戳来索引。时间戳的类型是 64位整型。时间戳可以由hbase(在数据写入时自动 )赋值，此时时间戳是精确到毫秒的当前系统时间。时间戳也可以由客户显式赋值。如果应用程序要避免数据版本冲突，就必须自己生成具有唯一性的时间戳。每个 cell中，不同版本的数据按照时间倒序排序，即最新的数据排在最前面。

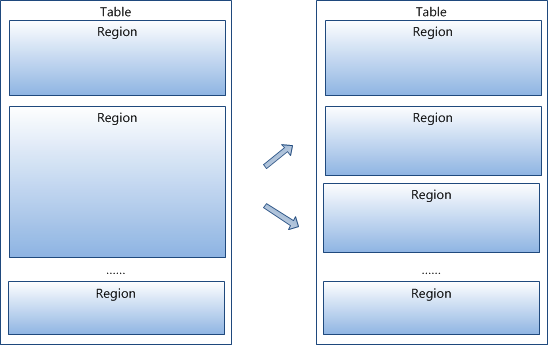
为了避免数据存在过多版本造成的的管理 (包括存贮和索引)负担，hbase提供了两种数据版本回收方式。一是保存数据的最后n个版本，二是保存最近一段时间内的版本（比如最近七天）。用户可以针对每个列族进行设置。

**Cell**

由*{row key, column(* =<family> + <label>*), version}* 唯一确定的单元。cell中的数据是没有类型的，全部是字节码形式存贮。

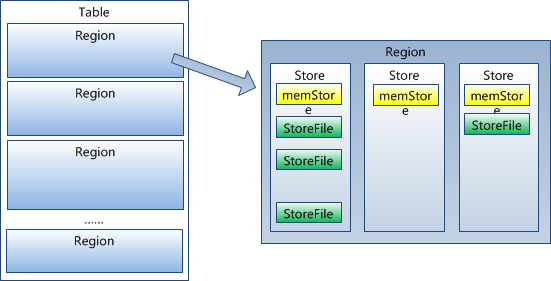
#### 2.3.2 Hbase的存储机制及寻址机制

虽然Hbase是列式存储数据库，但是其表中的所有行都是按照row key进行字典排序的。由于Hbase需要适应大数据量的存储，因此，其按照行的方向分割成了多个Hregion。Hregion是按照数据表的大小进行分割的，当数据量比较小的时候，每一个表只有一个region，随着数据量的不断增加，region会不断的扩大，当增大到一定程度的时候，Hregion就会等分成两个新的Hregion，相应的，随着数据表中的数据越来越多的时候，Hregion也会越来越多。如下如Hregion分割情况所示：



Hregion是Hbase中分布式存储和负载均衡的最小单元，当数据表中数据比较多的时候，一个表中不同的Hregion肯能会分别存储到不同的服务器节点中，但是一个最小单元Hregion不会存储到不同的服务器节点中。

注意的是，Hregion是分布式存储的最小单元，但却不是存储的最小单元。存储的最小单元是Store，一个Store保存一个列族。而Hregion则由一个或者多个Store组成。每个Store又由一个memStore和0到多个StoreFile组成。StoreFile以Hfile的形式保存在Hadoop的文件系统hdfs上。如下图所示：



HFile分为六个部分：

Data Block 段–保存表中的数据，这部分可以被压缩

Meta Block 段 (可选的)–保存用户自定义的kv对，可以被压缩。

File Info 段–Hfile的元信息，不被压缩，用户也可以在这一部分添加自己的元信息。

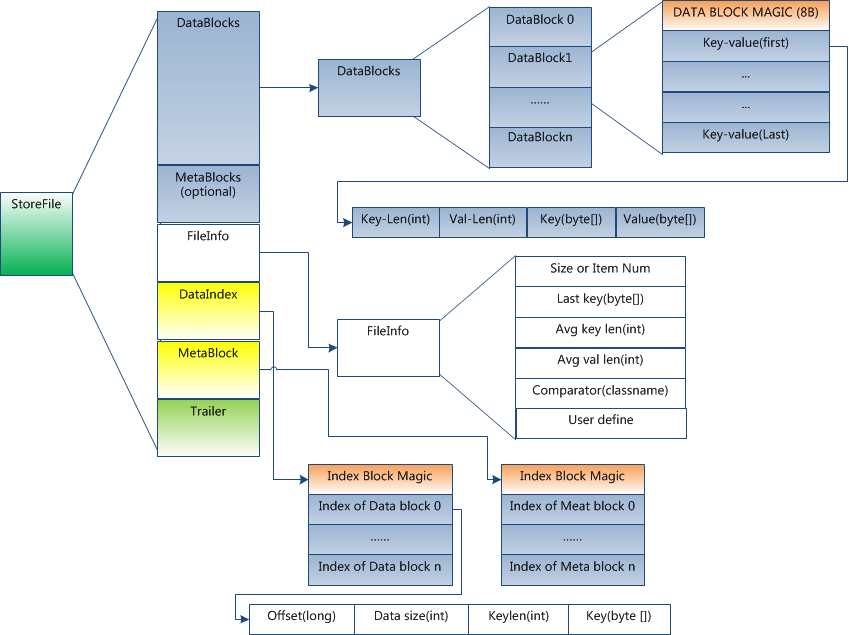
Data Block Index 段–Data Block的索引。每条索引的key是被索引的block的第一条记录的key。

Meta Block Index段 (可选的)–Meta Block的索引。

Trailer–这一段是定长的。保存了每一段的偏移量，读取一个HFile时，会首先 读取Trailer，Trailer保存了每个段的起始位置(段的Magic Number用来做安全check)，然后，DataBlock Index会被读取到内存中，这样，当检索某个key时，不需要扫描整个HFile，而只需从内存中找到key所在的block，通过一次磁盘io将整个 block读取到内存中，再找到需要的key。DataBlock Index采用LRU机制淘汰。

HFile的Data Block，Meta Block通常采用压缩方式存储，压缩之后可以大大减少网络IO和磁盘IO，随之而来的开销当然是需要花费cpu进行压缩和解压缩。

目标Hfile的压缩支持两种方式：Gzip，Lzo。



## 2.4 本章小结

本章先从Hadoop生态系统的宏观角度把握什么是最新的Hadoop大数据分析计算平台，之后分别从Hadoop大数据平台的背景，Hadoop核心组件及其文件系统和基于Hadoop的列式存储数据库Hbase来系统的介绍Hadoop分析计算平台的由来及其核心功能。

对于Hadoop大数据平台的背景，本章主要做了一个大概的介绍，旨在能对Hadoop大数据平台的实际应用能有一个宏观的把握，从而更好的在下文的应用中清晰流畅。

对于Hadoop大数据平台的核心组件及其文件系统是本章介绍的重点，原因在于它是一切文件及其数据存储的基础，也是整个架构最底层的部分。只有这一层搭建好，才能在让其上搭建的Hbase数据库更好的发挥数据存储和检索的功能。

本章的介绍的另外一个重点是基于Hadoop文件存储系统hdfs的列式存储数据库Hbase。这种分布式的，高性能，高可靠的数据在Hadoop大数据平台上有着得天独厚的优势，能够改善hdfs文件存储系统在数据检索方面的不足，做到对数据进行更加灵活，扩展性更强的处理。

# **第3章 传统遗传算法和TSP问题**

## 3.1 遗传算法

#### 3.1.1遗传算法简介

遗传算法（Genetic Algorithm）是受生物进化学说和遗传学说的启发发展起来的。在生物进化的的过程中主要有以下几个特点：不断繁殖，生存竞争，适者生存和遗传变异。而生物的遗传规律有分离规律和自由组合规律，生物变异的特点有基因重组、基因突变和染色体变异。

而遗传算法则是模仿了生物在整个遗传进化中的这些特点，从而模拟出的一种求解模型。具体来说，遗传算法是一种搜索寻优的技术或者说是一种优化的算法，其原理在于模仿生物遗传和进化，根据生存竞争和优胜劣汰的原则，复制，交换变异等操作，使得要解决的问题从初始解开始一步一步逼近最优解。

遗传算法的研究对象主要是各种非线性的、多变量的、多目标的复杂的自适应系统问题。

#### 3.1.2遗传算法的发展历史

遗传算法的发展历史向前可以追溯到上个世纪50年代，大概有着60多年的历史，其具体的发展历史可以分为以下几个阶段：

萌芽期，50年代后期到70年代中期。由一个数学家提出遗传算法。之后随着科学技术的不断发展，美国一位教授Hadoman和其学生对其做了比较大的贡献，在其基础上发表非常多的相关研究论文及专著。

成长期，70年代中期到80年代末期。遗传算法产生了很多固定的算子和技术。也就意味着遗传算法的发展趋向于成熟阶段。其每两年一次的学术交流会也成为其成熟的标志。并且遗传算法不仅仅停留在数学方面，其在物理领域、生物领域、营销领域等有了长足的发展。对经济、社会产生了重要影响。

发展期，90年代。遗传算法的发展处于成熟期。其每年的都有定量的论文及专著发表，学术交流也会定期举行。

今后的发展。随着遗传算法的不断发展，其渐渐衍生出来遗传规划的概念，遗传 规划在结构化设计、人工智能、复杂问题优化、复杂系统分析等方面有着比较优越的特性。

虽然遗传算法已经比较成熟，但是仍然有许多可以继续改进的地方：

1、 编码技术和程序表达技术的改进。

2、 复制、交换、突变等遗传操作的改进。

3、 适应度的表达和适应度的改进。

4、 寻求其它有效遗传算子，防止近缘杂交、过早收敛。

5、 进一步探讨遗传算法和遗传规划的机理。

6、 开发遗传算法和遗传规划的商业软件。

#### 3.1.3遗传算法的特点

1. 智能式搜索。依据适应度（目标函数）进行的只能搜索。
2. 渐进式优化。利用复制、交换、突变等操作使下一代结果优于上一代。
3. 全局最优解。 采用交换和突变操作产生新个体，使得搜索得到的优化结果逼近全局最优解。
4. 黑箱式结构。根据问题的特征进行编码（输入）和确定适应度（输出），具有只考虑输入与输出关系的黑箱式结构，并不深究输入与输出关系的原因。
5. 通用性强。不要求明确的数学表达式，只需要一些简单的原则要求，可应用于解决离散问题、函数关系不明确的复杂问题。
6. 并行式运算。每次迭代计算都是对群体中的所有个体同时进行运算，是并行式运算方式，搜索速度快。

#### 3.1.4遗传算法的运算过程

遗传算法的运算过程主要是通过循环语句来控制5个主要的步骤。

第一步、编码、随机产生初始群体。这一步是遗传算法求解中非常关键的一步，是所有其他步骤的基础。编码的好坏直接决定遗传算法求解的速度，效率。当编码完成之后紧接着产生初始群体。所有的初始群体是用码的方式表现出来的。

第二步、个体评价、选择、确定是否输出。在初始群体产生之后，要设定一个适应度函数对所有的群体进行评价。检测初始群体中的各个个体是否达到预想的要求。适应度函数是遗传算法中用来判定是否进行下一代循环的判别条件，也是判别是否得到全局最优解的一个判断指标。当要进入下一层循环迭代的时候，要像生物进化一样进行优秀个体的选择，对于优秀的个体还要进行相应的复制，使其在群其中的比例增大，相应的对于比较差的个体筛选掉。

第三步、随机交叉运算。由于遗传变异的过程中是存在杂交现象，因此这一步则是模拟遗传变异种的现象。

第四步、随机变异运算。由于在生物的遗传变异过程中同时也存在个体变异的现象，因此在这一步则是模拟生物的进化的变异现象而做出的操作。上一步的交叉 和此步骤的变异运算是遗传算法中产生新解的重要操作，也是防止在计算过程中出现过早收敛，避免只得出局部最优解而不是全局最优解的手段。

第五步、转向个体评价，开始新的循环。这一步是对上一代结果的评价，检查是否达到了预期的输出条件。若没有达到输出条件则进入下一代的循环迭代。



另外，在整个遗传算法运算的过程中，有许多关键点需要值得注意，比如如何进行编码，个体适应度的计算，以及各个遗传算子的选择等，具体如下：

编码过程。编码过程是整个遗传算法运行的基础，编码的好坏直接决定遗传算运行的效率是是否能得到全局最优解而不是局部最优解。一般来说，遗传算法工作的对象是字符串和数字。因此对编码的要求要能反应所要研究问题的本质属性，并且所得到的初始码能都便于计算机的运行和计算。

适应度计算。在整个遗传算法的计算过程中，第二步就是计算种群中个体的适应度，适应度的大小直接决定这一个个体是有多大几率会被遗传到下一代。因此适应度是衡量种群个体的优劣的重要尺度，遗传算法中的适应度同时也是驱动整个遗传算法运行的核心动力。

选择操作。选择操作是在种群个体在计算出适应度并根据个体适应度大小进行排序后的一个操作，目的在于选出能都最大程度适应环境的个体进入下一代个体。

复制算子。在进行个体选择之后，接下来就是对复制完毕的个体进行复制操作。这个体现了自然界的“适者生存”的原则。整个复制得数量也是根据种群个体在按照个体适应度进行排序后决定的。因此适应度越大的个体，其复制的数量也就越多，适应度比较小的个体相应的复制的个数也就越少，但是最终结果要保证整个个体的总数是不变的。J.H.Holland教授提出了一种按照轮盘赌的方式进行个体的复制。这种方式很好的解决了复制所需要的个体数目。

交换算子。整个遗传算法在运算过程中的一个重要操作就是淘汰掉个体之后产生新的个体，而交换操作则是产生新个体的主要办法。这个算子的操作主要是两个个体的一部分字符进行两两交换。整个交换过程中个体的选择是完全随机的。这样极大的保证了整个过程的自然选择的特点。根据遗传算法多年求解的历史经验，一般来说交换的概率一般会定在50%-80%左右。

变异算子。变异是整个遗传算法运算过程中产生新个体的另一种方法，这个算子借鉴了整个生物在自然选择的过程中的突变现象。这个算子的重要作用在于尽量保持整个遗传算法的计算不会陷入局部最优解而是求得全局最优解。

终止条件。遗传算法的一个重要的特点就是通过反复的迭代和分析计算逐渐逼近整个求解域最优解的过程。因此整个算法必须有有个能够进行终止的条件。在整个遗传算法应用的历史中，最常用的终止条件是规定迭代的代数。当整个算法迭代到一定代数的时候进行终止操作。另一种方式是采用控制变差的方式进行算法的终止，比如整个算法的最终的求解结果不会产生很大的变化，这时候我们可以对算法进行终止操作。第三种方式则是检查适应度变化来决定是否对算法进行终止操作。

## 3.2 TSP问题

#### 3.2.1TSP问题简介

TSP问题，又名为旅行商问题，是在数据求解模型中的一种著名模型，最早是由Dantzig在1959年提出的算法求解模型。其算法的基本思路是一位旅行商要向给定的一些城市去推销自己的货物，而城市与城市之间的路程不一样，因此，这位旅行商要选取一个最优的行走方案挨个城市去推销自己的货物。这同时也是一个最优组合问题的原型。

TSP问题在实际的生产和生活中有着广泛的应用，比如我们日常生活中的物流配送系统。如果在比较的小的国家中，此类问题可能不太突出，但是在我国这种地域范围广，城市数量较多的情况下，一种比较优秀的配送系统模型无疑能为整个物流系统的配送流程节约大量的人力物力成本。

旅行商问题最简单的理解方式为：有一个旅行商，他要到给定数量的城市取推销自己的商品，为了节约人力和物力成本，他要找出一个最短的行走路线去推销自己的商品。TSP有着非常悠久的历史，最早的描述是在1759年欧拉研究一位骑士的周游问题，即对国际棋盘上的64个方格，每一个方格都走一次并且最终要返回到起点。后来，TSP问题由美国的RAND公司在1948年引入，得益于该公司的声誉以及线性规划这一新方法的出现使得TSP成为一个知名且流行的问题。

#### 3.2.2TSP问题的传统求解思想

对于旅行商的求解一直以来是研究的热门话题，原因在于随着城市数量的增多，其旅行商的求解组合将是成指数级数量增加，因此，如何构建一个优秀且效率出众的求解算法模型也成为了求解旅行商问题的关键。传统的旅行商求解问题在城市数量较少的情况下构建出了很多经典的模型，当然这些模型也可以用作城市数量较多的情况下，但是运算效率确也是值得深思的一个方面。

对于传统的旅行商求解问题思路主要有以下几种：

分支界限法：

基本求解思路：广度优先搜索或者是最小损耗的方式对解空间树的搜索方式是分支界限法求解问题的基本思路。整个问题的解空间树是整个问题解空间的一课有序树，最常见的是子集树和排列树两种。在搜索问题的解空间的时候，回溯法与分支定界法最根本的区别在于他们对于当前节点的扩展所采用的扩展方式不同。在传统的分支定界法进行分支扩展时，每一个活节点只有一次机会成为扩展结点。当活节点一旦成为扩展结点后，就会一次性的将其子孙节点全部生成。在所有生成的子孙节点中，导致产生不可行解活着导致非最优解的子孙节点将会被舍弃，其余的儿子节点都将加入到活节点列表中。此后，从活节点列表中取出下一个节点作为当前节点并继续进行相应的扩展，重复上述步骤过程。根据上述步骤一直进行搜索并一直找到所需要的最优解或者直到活节点列表为空为止。以下是常用的两种分支定界的方法：

第一种，方式队列（FIFO）分支定界法

这种方式是简单的将整个活节点列表组成一个队列，整个选择的原则则是按照先进先出的原则进行选择作为当前的扩展结点。

第二种，优先队列式分支定界法

这种方法是将所有的活节点列表组织称一个具有优先权重的优先队列，并按照优先队列中规定的节点的权重进行选取，优先级较高的优先称为下一个节点的活节点，优先级低的反之。

此类求解算法在进行初步无向图规划模拟的时候计算量一般是比较一般的，但是如果随着城市数量的增多，其生成最小二叉树的过程的计算量将会非常巨大。

动态规划法：

对于动态规划求解TSP问题的求解思想在于，对于此类问题，旅行商从哪个城市出发对于最终的求解结果都是一样的。因此我们可以假设从第一个城市出发。

近似算法：

对于近似算法求解，可以采用一个给定完全无向图G=(V,E)，并且其假设每一条边(u,v)∈E都有一组非负整数的费用。之后我们只需要找出图的最小哈密顿回路即可。其求解最小哈密顿回路的算法可以采用图G的最小生成树来求解。

这种近似算法有其自身的优越性，比如在初始节点不多的情况下求解是比较方便的。但是同样面临初始节点过多的情况下，求求解计算过程也是相当大庞大，并且其计算难度也是呈指数增长的。上述描述的采用图G的最小生成树的方式来进行求解，我们可以假设初始节点已经达到上百个，这样其求解框架虽然比较容易构建，但是求解的时间复杂度却不是一般的机器可以衡量的。因此此类求解框架在数据量较大的情况下是有很大的局限性的。

#### 3.2.3TSP问题的智能优化算法求解思想

随着社会生产力和科学技术的发展，智能优化算法越来越成为很多经典问题求解的首选方法。智能优化算法有着可扩展性强，求解效率高效，符合人类生产和生活中的理解问题的思路。因此，在在以往很多不容易求解的问题中，智能优化算法发挥了其得天独厚的优势。对于本文的旅行商问题，同样有着非常多的智能优化算法来进行求解，比如常见的模拟退火算法，蚁群算法，蜂群算法以及本文将要提出的遗传算法。对于每种算法的求解思路及其介绍如下。

模拟退火算法：

模拟退火算法是一种通用概率算法，其旨在一个给定的大的空间内进行最优解的搜索。该算法的原型是来自于对热力学中退火过程的描述，在某一给定的初始温度下，通过缓慢的降温过程，使得算法在多项式的空间内给出一个近似的最优解。对于TSP问题，模拟退火算法可以通过其贪心算法在整个全局范围内进行近似求解。其算法思路如下：

第一步，编码选择。一般来说采用描述TSP问题常用的策略即路径编码策略。

第二步，对SA状态产生函数设计。由于编码选择策略选取的事路径编码策略，因此可将其设计为三种操作方式即，呼唤操作方式、逆序操作方式和插入操作方式。

第三步，SA状态接受函数的设计。min{1，exp（-△/t）}>random[0，1]准则是一种SA接受新状态的条件中最常用的设计方案，△为新旧状态的目标值差，t为”温度”。

第四步，初始温度和初始状态设计。在SA设计中，最常用的一种初始温度设计方案是，首先随机产生一组状态，之后确定其中任意两个状态之间的最大的目标差值即|Δmax|，然后进行公式t0=-Δmax/lnpr的计算，其中pr为初始接受概率（理论上应接近1，但是实际设计时可以取为0.1）。在初始状态可以采用启发式算法求解从而快速得到一个解，并将此作为SA的初始状态。

第五步，退温函数设计。对于常规的退火算法求解问题，指数退温函数是最为常规的退温策略（tk=λtk-1，λ为退温速率）。

第六步，温度修改法则和算法终止条件设计。这里可以采用阀值设计的”温度修改”和”算法终止”两准则。

蚁群算法：

蚁群短发的研究依据在于蚂蚁在寻找食物的过程中，会在自己走过的路上留下自己的一种分泌物，我们称之为信息素，这样在一定的范围内当其他的蚂蚁嗅到以前蚂蚁留下的信息素之后，会根据这些信息调整自己的觅食路径。这样，当一条路上的蚂蚁越来越多的时候，其他蚂蚁对这条路的选择概率也越来越大，同时这条路的信息激素也会越来越强。

运用蚁群算法求解旅行商问题就是根据蚂蚁的这种觅食行为而得出的模型。其求解思路主要分为以下几个步骤：

第一：初始化蚂蚁求解种群，即给定一定数量的蚂蚁作为初始求解种群，并且这个种群的蚂蚁数量应该远大于城市的数量。

第二：对其中的每一只蚂蚁选择到达下一个城市，对每一个蚂蚁计算下一个访问城市，访问城市不可重复，直至所有的蚂蚁走完所有的城市。

第三：计算各个蚂蚁经过的路径长度，记录当前问题的最优解，同时更新各个链接路径的信息素浓度。

第四：判断是否达到最大的迭代次数，是则终止，否则重复进行第二步。

从以上的描述中我们可以看出，蚁群算法的核心求解思路其实就是穷举法，这种方法对于小数据量的问题求解具有很高效的求解效率，但是不适合于大规模的问题求解模型。

遗传算法：

遗传算法是一种模拟大自然自选择的一种智能优化算法。这种算法的优势在于其模型了整个自然选择的过程，并且整个计算过程中遗传算法是可以并行计算的，因此在遗传算法求解过程中，可以进行大规模的并发计算，从而提高问题求解的效率。对于遗传算法求解TSP问题的详细过程在此不做过多的赘述。

## 3.3本章小结

本章主要介绍了遗传算法和TSP问题两个问题，对于遗传算法的介绍，本章节主要从遗传算法的发展历史作为开端，引出了在整个的遗传算法的发展流程中，遗传算法主要有哪些重大的改进。同时紧接着介绍了遗传算法的算法特点，并且根据这些总结出为什么在现如今科学技术和生产力急剧发展的时代，遗传算法成为了众多热门算法之一。然后介绍了整个遗传算法是如何仿真自然选择过程的，如何对传统的自然选择过程进行了有效的改进，使得遗传算法在求解问题的时候更加有效率，更加符合经典问题的求解模型。同时，对于遗传算法如何符合并行求解框架也做出了简要的介绍，为后文引出混合并行遗传算法求解做了铺垫。

对于本章节的后半部分主要介绍TSP旅行商求解问题，主要从TSP问题的简介，TSP问题的传统求解思想及其TSP问题的智能优化算法求解模型两个方面进行介绍。详尽的列出了传统求解问题的框架及其思想，并对其各个求解模型的求解思想进行简单的概括说明并进行适当的改进评判。

# 第4章 基于Hadoop的并行遗传算法模型

## 4.1 基于Hadoop的并行遗传算法实现依据

遗传算法在社会的诸多领域有着非常广泛的应用，比如生产调度、图像处理、函数优化等。虽然随着社会的不断发展，数据量的增大，求解问题的复杂性增大，使得传统的算法模型在求解问题上力不从心。由于遗传算法的思想是从种群出发，因此自身有着并行的固有属性，非常适合在并行的分布式的计算机集群上运行，随着并行计算平台的普及也同时为遗传算法的并行计算奠定了基础。比如遗传算法中个体适应度值的计算以及评价可以独立进行，互相之间没有任何依赖，并且计算过程中不需要任何通讯。

在并行遗传算法（PGA）中，要做的不仅仅是要将并行遗传算法变成一种方案，更重要的是修改遗传算法的计算模型，使其能够很好的适应分布式并行计算平台。在并行遗传算法改进的过程中，一是要注意如何将大规模的初始种群进行划分，使得并行计算平台能够更有效率的运行；二是在其选择，交换和变异的过程中如何处理各个种群之间的通讯机制，使得遗传算法在整体运行中处于稳定状态。

以下是遗传算法可以进行大规模并行计算的理论依据。

#### 4.1.1个体适应度计算的并行性

在遗传算法的运算过程中，个体的适应度计算是非常关键的一步，适应度的计算决定了产生下一代的初始群体。但是群体中个体适应度的计算也是整个遗传算法中最消耗时间的一步操作。

但是对于个体的适应度来说，其计算只与所选取的适应度函数有关，和其他因素没有什么大的关系，并且，个体适应度的计算顺序也不会对遗传算法的最终结果产生影响。因此可以将个体适应度的计算通过一定方法进行分组执行，比如在分布式的环境中将群体分成不同的组，然后将各个组分别部署到不同的机器上，这样个体的适应度计算就可以分开进行了，通过这种方法可以提高遗传算法的计算效率。

#### 4.1.2整个群体中各个个体适应度评价的并行性

在遗传算法的求解过程中，当各个个体的适应度计算出来之后，紧接着就是对个体适应度进行科学的评价，而评价的优劣直接决定下一代群体初始群体的选择。但是对于种群较多的情况下，对个体适应度进行逐一评价，毫无疑问将非常耗时并且浪费计算机资源。因此如果能通过一定的方法，比如借助最新的分布式平台策略无疑是提高遗传算法运算效率的途径之一。

因此我们可以首先通过一定的算法将待评价的种群分成不同的组，这样将各个组分布在不同的分布式计算服务器上，由各个服务器只关注处理自己机器上的个体评价，之后通过一定通讯机制进行汇总，这无疑是提高遗传算法运行效率的一种优化方法。

#### 4.1.3子代群体产生过程的并行性

在遗传算法的求解过程中，产生子代的主要步骤是选择、交换和变异。这三种操作中，选择操作只与个体的适应度有关。而交换和变异操作只是和参与运算的编码有关，与之前是否有过选择、交换或是变异操作无关，因此在理论上是可以将其三种操作互相分开进行。

根据上述遗传算法的子代产生过程的并行特性，可以通过将其选择、交换和变异这三种操作分布在不同的主机上进行，使其独立运行，从而提高遗传算法的运算效率。

#### 4.1.4基于群体分组的并行性

以上几种并行求解理论都是整体算法的背景下进行的局部改进。从原理上来讲，遗传算法的总群体是有不同的群组组成的，根据生物界一个物种不同种群可以共同进化的特点，因此遗传算法的进化迭代同样也可以分成不同的物种种群进行，这就在理论上实现了遗传算法并行的基础。

在种群应用遗传算法计算的过程中，将一个种群规模较大的初始种群根据一定的方法将其分成不同的种群，之后对不同的种群应用一样的遗传算法分析求解框架。这样，在大数据分布式的环境下，不同的种群分析计算可以分布在不同的主机上进行，毫无疑问这样可以提高整体遗传算法的求解效率。

## 4.2 传统的并行遗传算法实现模型

#### 4.2.1全局PGA模型

全局PGA模型并行遗传算法中一种主从结构，这种解决方案是保留初始种群的数量而不进行二次分组。在整个遗传算法运行的过程中，它只有一个种群，因此不管是种群的适应度函数还是适应度评价都只是一套方案。并且在种群的选择、交换和变异操作中，任何一个个体都有机会同全部的个体进行选择、交换和变异操作，因此这种模型是全局性的。其并行的核心在于主从服务器机构，主服务负责统一安排其他服务进行局部的操作运算，之后将运算的结果在整合到主服务器上进行分析汇总。这种实现模型主要有两种测量，第一种是对种群的个体适应度计算进行并行处理。第二种是对各个遗传算子进行从节点的并行处理。

这种模型的好处在于保留了传统遗传算法的全局搜索行为，是站在全局上的遗传算法的处理机制。并且这种算法对并行处理框架要求不高，实现比较简单。但是对于初始种群过大的群体，这种算法无疑又暴露出了传统遗传算法的缺点，种群数量过大又成为了影响计算效率的瓶颈。

#### 4.2.2粗粒度PGA模型

这种算法模型又称之为孤岛遗传算法模型，这种一种彻底的群体分组模型。在初始阶段就将种群分成很多组。每一个组独立运行一个遗传算法，从而全局内选择变成了区域内选择。因此在整个遗传算法的计算过程中，每一个组维护一套算法操作，在进行遗传算子操作的过程中，也是在自己组内进行选择，而不去在全局进行搜索。由于遗传算子不会跨组进行选择，所以在这种算法模型中引入了一种用于组之间通讯的算子，即“迁移”算子。

迁移算子的最核心作用就是维护组与组之间的信息交互，避免各个组在搜索过程中陷入局部最优解。迁移算子主要包含“迁移拓扑”、“ 迁移率”和“迁移周期”三个基本要素。迁移拓扑确定了这种模型在进行种群迁移时候的迁移路径，这与计算机的架构模型也有着一定的关系。而迁移率则是表示在进行种群之间迁移的时候应该迁移的数量，这从全局来看使得整个遗传算法不会陷入局部最优解，而可以在最终结果中找到全局最优解。最后迁移周期则决定多长时间迁移一次，迁移周期的大小直接影响整个遗传算法的计算效率，迁移周期过多，则有损整个算法在计算过程中的时间消耗，如果过小，则达不到迁移算子的作用。一般来说，典型的迁移率控制在10%到20%位佳，而迁移次数则一般一代迁移一次或是几代迁移一次，因为这样比较符合自然界物种迁移的规律。通常来说，迁移率越高，则迁移周期越长。在迁移的过程中，有时候采用同步的方式进行迁移，有时候采用异步的方式进行迁移。另外，在种群迁移的过程中，有时候选择一个或多个最优的个体进行迁移，有时候则规定通过随机选择的方式进行迁移，但大多数情况下，可以采用适度比例或者排列比例进行选择，确定哪些个体将会被替换，以便利于整体算法的顺利运行。

基于国内的现状，分布式PGA为国内PGA研究的主要方向。分布式PGA作为PGA的一种形式，一般实行粗粒度及全局级并行，各子种群间的相互关系较弱，主要靠一些几乎串行GA来加速搜索过程。采用分布式PGA求解问题的一般步骤为：（1）将一个大种群划分为一些小的子种群，子种群的数目与硬件环境有关；（2）对这些子种群独立的进行串行GA操作，经过一定周期后，从每个种群中选择一部分个体迁移到另外的子种群。对于个体迁移存在多种方法，第一种方法，在执行迁移操作时，每次从子种群中随机选择一部分染色体发送出去，接收的染色体数应该与发出的染色体相同。第二种方法，在执行迁移操作时，首先在每个子种群内只使用选择而不使用其它遗传算子繁殖一些后代，这些后代的数目与迁移数相同。然后再将这些后代的原子种群合并成一个大子种群并均匀随即地从该子种群中选择个体进行迁移。这样，待迁移后子种群的规模便又恢复到正常状态。而当子种群接收到从其他子种群迁移来的个体时则均匀随即地替换掉子种群内的个体。第三种方法，将其中一个子种群设置为中心子种群，其他子种群与中心子种群通信。中心子种群始终保持着整个种群中当前的最优个体，其他子种群通过“引进”中心子种群中的最优个体来引导其加快收敛速度，改善个体特征。

#### 4.2.3细粒度PGA模型

细粒度PGA模型又称之为领域模型，其核心思想是对传统遗传算法模型的修改，同样维持一个群体的进化，即对总群体不进行群体的拆分。但是这种模型在种群的平面网格上也就是在切面的模式下对种群进行的极细的小种群划分（理想情况下是每一个个体维护一个种群），并且种群之间有着极强的通讯能力。这种模型即维护了在一个种群的情况下的全局搜索能力，又使得可以并行计算的部分能以并行的方式在各个服务器上进行运行，极大的利用分布式环境下的服务器资源。细粒度模型要解决的主要问题是领域结构和选择测量。

领域结构既决定了种群中个体的空间位置，也确定了个体在种群中传播的路径。领域结构主要受特定并行计算机的内存结构和通信结构影响。领域拓扑确定一个个体的邻居，构成该个体的局部领域。通常，只有一个拓扑的直接领域才属于其局部领域，若把某个固定步数内所能到达的所有个体也包含在内，则可以扩大领域半径。在确定选择策略时，要考虑到选择压力的变化，而选择压力与领域结构有关。与全局匹配选择类似，局部匹配选择可以采用局部适应度比例、排列比例选择，以及随机行走选择。局部生存选择确定局部邻域中被替换的个体，如果子代自动替换邻域中心的那个个体，那么可以直接使用代替换作为局部生存策略。

#### 4.2.4混合PGA模型

混合PGA模型，顾名思义，是以上几种并行遗传算法求解模型的集合，它根据业务场景的不同，在实际情况下选择如何联合以上几种模型来进行求解运算。一般来说，混合PGA模型不是单一结构的，而是层次化的，比如上层采用粗粒度模型，下层则采用细粒度模型来进行求解。或者在求解过程中分阶段，前一个阶段采用一种模型结构，在最后进行总括分析的时候采用另一种求解模型。

## 4.3基于Hadoop的并行遗传算法实现模型

#### 4.3.1 混合并行遗传算法求解模型

传统的并行遗传算法在进行问题求解的时候往往只能局限在某一类问题或是只是针对某以来问题进行重新设计，这大大减少了遗传算法的可用性。比如全局并行遗传算法虽然对算法的计算效率有了很大程度的优化，但是在面对初始种群过多的情况下也显得力不从心。而局部并行的遗传算法虽然解决了在初始种群过多的情况下的效率问题，但是其全局搜索能力却有了很大的限制，虽然在计算过程中引入了“迁移”算子的概念，但是其全局搜索能力的限制在很多问题上也是不可忽略的。实际情况下，既要有全局搜索能力好，并且效率极高的遗传算法在并行环境下是不存在的，因此，如何能协调两者之间的关系，如何针对实际的生产环境进行有效的调节则成为了当前分布式环境下的并行遗传算的当务之急。

针对上述环境，本文提出了混合并行遗传算在分布式大数据分析计算平台Hadoop上的实现，并且针对这种模型进行了扩展性方面的优化，使得既能兼顾全局搜索性能又能有着不错的效率的并行遗传算法成为可能。具体实现思路如下：

本文模型是在传统遗传算法的求解框架基础上，对全局遗传算法和局部并行遗传算法进行结合，并引入种群代数适应度函数。由于局部并行遗传算法具有解决全局初始种群过大的情况，因此，可以将这种模型放在整个算法模型的前端，对于初始种群过大的模型，可以先通过这种计算模型进行前n代的筛选，这样在经过n代种群迭代之后，将边缘化的、进化偏离的种群进行筛选过滤，这样随着种群的前提迭代，会将整个种群的数目降低。当种群的整个种群数量降低到一定程度之后，则由全局并行的遗传算法模型进行下一轮的分析运算，对结果进行下一轮的迭代筛选，种群代数适应度函数的引入意义在于确定种群的前期迭代需要多少轮迭代。此函数的引入不仅仅能确定遗传算法在哪一代的时候进行终止并进入下一轮的迭代，同时还会提升整个遗传算法的扩展性，也就是根据实际问题和大数据的分析计算环境的各个因素比如环境节点的多少，环境最大吞吐量及其环境的负载性能来得出中间平衡点也就是停止前一轮的迭代而进行下一轮模型的迭代。

整个并行模型的流程图如下：

***此处应该有图***

如上图所示，整个并行遗传算法的模型主要分为两个部分，上半部分为局部并行搜索部分，而下半部分为全局搜索部分。需要注意的一点是上半部分和下半部分不是绝对对等的，而是根据实际生产业务情况进行比重划分，有的时候可以让上半部分多一些，有的时候可以让下半部分多一些，当然在特殊情况下，其中一部分也可以舍弃掉。正如前面描述的，上部分主要解决初始种群过大的情况，这样在实际初始种群过大的情况下进行种群分割，让遗传算法的运行分布在不同的节点上进行运行，这样经过n代循环迭代之后有一些种群处于边缘化，成为无意义的数据，因此这个时候可以将其舍弃掉，从而整个初始种群数目会减少。从而紧接着进入下一步的全局遗传算法迭代，在全局迭代的过程中，仅仅对全局搜索没有影响的算子操作，比如适应度计算等进行分节点分布式计算，这样在局部分布式的情况下提高遗传算法的计算性能和效率。

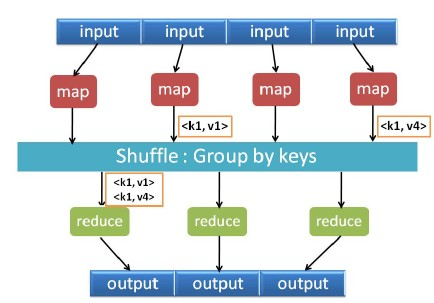
#### 4.3.2 混合并行遗传算法在Hadoop中的实现

Google在以往经典的映射化简的函数式编程思想中进行总结，从而产生了当今流行的Hadoop大数据分析计算平台，并且其MapReduce并行计算框架非常完美的同其文件系统hdfs结合，对大数据的分析效力产生了深远影响。

在整个MapReduce的运算过程中，所有的Map阶段操作都是并行操作的，我们无需去关心一个Map是在一台机器节点上运行还是被多台机器节点同时调用，其框架中的并行机制能保证各个Map的调用是并行运算的。Map在接受参数为key/Value键值对，key为每一行的行号，而value为每一行的内容。

当Map阶段运算完毕之后，也就是表明每一行的数据都被进行了一次全局的预处理，并且其整个处理过程是并行完成了，接下来MapReduce会将整个处理结果全部交付给Map阶段和Reduce阶段的中间链接环节shuffle来进行处理，这个阶段的处理主要是将Map阶段产生的数据进行排序和分组，这就是说会将整个数据按照Map的处理结果分成不同的域，之后将分完组的数据交由接下来的Reduce进行进一步的处理。

第三步则是由Reduce对分好组的数据进行处理，由于第三步Reduce得到的数据是已经在shuffle中排序好的数据，其实也就是在处理一个个的数据集合。因此整个过程的调用也是并行进行的。具体处理过程图如下：



根据上面的描述以及遗传算法固有的并行属相，将其与Hadoop大数据分析计算平台具有天然的优势。由于上文提出的混合并行遗传算法的运算具有两个阶段，第一个阶段是局部并行计算阶段，而第二部分为全局并行阶段。具体阶段实现如下：

第一阶段，由于这个阶段的主要目的是对过大的初始种群进行并行筛选，得到具有很强代表向的种族群体，为下一阶段的运算提供便利。因此这一步的主要任务是通过MapReduce进行初始种群的筛选，得到具有强代表性的种群群体，并作为第二阶段的主要输入。因此这一步是根据MapReduce的整个分析计算过程分成三个阶段，第一个阶段是Map阶段，Map阶段将对整个种群群体进行适应度计算。紧接着Map阶段会将产生的结果输入到第二阶段的shuffle阶段，这一步由于是初始种群的筛选，因此我们采用传统的shuffle算法对整个种群进行排序和排序得出这一步的结果。随后进入第三个阶段即Reduce阶段，这个阶段我们将会对整个带有适应度值的群体进行两种操作，第一种是去重操作，由于这一步的输入结果是以分组的形式输入的，也就代表着这个组中的适应度值是一样的，因此，我们只需要取出这个组中的一个种群个体即可。另外，对相邻的个体进行适应度值的相似性比较。因此我们这一步的结果是筛选出具有代表性的个体，因此对于一些适应度值相似的个体我们只需要取出其中一个作为代表个体即可，不要全部使用，因此取出这个这些具有强代表性的个体作为下一阶段的的种群输入。



第二阶段，这一阶段主要是利用上一阶段的产生的结果，进行全局并行搜索，得出最终结果。因此，可以将整个遗传算法的整个阶段按照MapReduce算法分成三个阶段。Map阶段主要做初始种群的输入和适应度的计算和终止条件判断。由于整个种群个体的适应度值具有无关性，因此完全可以在Map中进行种群是适应度值的并行计算。其次是终止条件的判断，在本文阐述的模型中终止条件一般选择迭代到一定次数或者最终搜索结果没有极大变化则去终止程序的运行。因此这一步可以刻在Map阶段进行判断。而shuffle阶段会对得出的结果进行分组和排序，在遗传算法中各个个体的适应度值计算完毕之后，很重要的一步就是对整个种群进行排序操作，因此在shuffle阶段将会完成整个种群的排序操作。而在Reduce阶段将对整个种群进行选择复制以及交换变异操作。这一步中的操作一共有三分部，第一部分是选择优秀的个体进行复制操作，并且淘汰适应度值不高的个体，之后将会随机选择一定比例的个体进行交换和变异操作，从而保证整个算法的最终结果不会陷入局部最优解，保证整个进化过程的完整性。

具体流程图如下：



根据上述描述，可得出整体流程如下：

此处应该有图：

根据上图描述，我们可以明确地看出整个算法的模型主要是通过前期并行筛选结合后期全局搜索得出最终求解结果。值得我们注意的是，第二阶段的全局过程也是基于Hadoop大数据分析计算平台并行运算的，因此在一些情况下，我们应该动态的调控第一部分和第二部分的权重关系，甚至有些时候我们可以直接省掉第一部分直接使用第二部分进行遗传算法的并行求解。因此，整个算法模型的实现背景是基于在初始种群过大的群体之上进行的。比如我们将在下文提出的本模型在TSP问题的应用验证。我们知道TSP问题是著名的优化组合问题，但是当参与优化组合的个体过多的时候，其排列组合的种类将是会成指数增长的。例如我们的排列组合初始参选个体有20个，其排列组合的种类就是20的阶乘个数，也就是135161222676480000个。在实际生活中，排列组合的初选个数是20个左右甚至更多的情况是很常见的，但是其排列组合的种类却是成指数增长的，因此一种优秀的算法解决模型会对这类问题的求解产生深远的意义和影响。本文提出的混合并行遗传算法正是解决此类问题的算法模型，为这类问题的求解解决了传统算法无法达到的效率上的优势。

#### 4.3.3混合并行遗传算法的优越性

在以上论述中，本文根据传统的遗传算法求解框架模型，并结合其优点，提出了混合遗传算法求解模型，这种模型根据不同遗传算法模型的分层求解框架，使得并行遗传算法既保留的全局的搜索能力，有极大的提到的算法是计算性能和效率。具体优越性如下：

保留了全局搜索能力。本文提出的遗传算法在第二层计算框架下主要采用全局并行的遗传算法计算模型，虽然在第一层进行了一次全局筛选，但是这种淘汰制的筛选只是将没有用的种群个体进行淘汰，并不会影响最终算法全局搜索能力。同时，这样做的目的是最大化的保留了全局搜索能力，避免了算法模型进入局部最优解的危险。

在全局范围内提高了算法的搜索效率。本模型的两个模型部分都采用的分布式并行策略。虽然求解的模型不同，但是第一层求解模型在初始种群较多的情况下采用分片并行策略提高了整个算法的求解性能。而第二层则是在于全局搜索无关的算子上进行了并行计算从而提高算法的计算性能和效率。这使得模型整体在全局范围内的搜索效率的了提升。

充分利用Hadoop大数据平台的能力。本文提出的模型与最新的分布式大数据平台hadoop进行结合，借助Hadoop的文件系统hdfs和其计算框架mapreduce。使得并行框架结合大数据计算平台，充分发挥了两者的性能。

充分发挥了遗传算法的扩展性。本文提出的混合遗传算法求解框架，在原有算子的基础上提出了迭代终止条件算子，这个算子的主要核心目的在于确定第一层计算框架和第二层计算框架的结合点，并且可以根据实际问题对这个算子进行评估，从而对第一层框架和第二层框架的比重进行相应的衡量，从而确定两层计算框架的比重。

提高具体问题求解的灵活性。在实际的问题求解中，往往不是拿一种固定的框架进行计算求解，而是根据实际问题确定计算框架的结构。而本文提出的混合模型同样给出了很多的框架变形接口，使得在实际问题中可以根据实际的业务情况进行适当的框架变形。从而让求解框架更加灵活，适应度更好。

## 4.5本章小结

本章节为整个论文的核心章节之一，主要的研究任务是先分析基于Hadoop的遗传算法并行的可行性，由于Hadoop是天生的并行计算平台，因此整个研究的核心是如何对遗传算法的计算过程进行相应的改进，在不失去遗传算法的核心计算分析计算优势的前提下对其算法的流程最早Hadoop进行改进。改进的着重点主要从遗传算的个体适应度的并行性，整个群体中各个个体适应度评价的并行性，子代群体产生过程的并行性以及基于群体分组的并行性这几个方面进行分析。最终的得出遗传算法在在Hadoop平台上并行的依据。

由于在遗传算法的并行模型前人已有了比较多的研究，因此第二部分的着重点在于分析前人们研究的成果，从全局PGA模型，粗粒度PGA模型以及细粒度的PGA模型这几方面进行相应的研究，并从中吸取有益的经验，结合自己的研究成果，提出了混合并行遗传算法在Hadoop大数据计算平台上的实现。

最后一部分主要是结合前面两部分的研究成果，基于前几部分的得出的混合并行遗传算法在Hadoop可以运行的依据，具体说明混合并行遗传算法在Hadoop大数据分析计算平台的实现模型，并对此模型进行了伪代码实现。最后通过分析论证对混合并行遗传算法在Hadoop大数据平台上实现的优越性进行总结概述。为下一章节的实际运用做了充分的铺垫。

# 第5章 TSP问题下的模型验证

## 5.1 TSP问题传统遗传算法求解模型

当旅行商问题中的城市数量不是很多时，采用传统的遗传算法进行求解无疑是一种比较高效，扩展性比较强的方法，整个过程的求解也会比较顺畅。针对具体的求解方法，我们先建立一个包含所有城市的数组，并且将这个数组的无序不重复组合的结果作为整个求解方法的初始种群数。之后对其整个种群进行选择复制、遗传变异操作，并且在整个进化过程中对每个种群个体进行个体适应度检查。当迭代到一定代数的时候进行终止操作，并从中选出最优个体。

流程图：



流程图解释：

如上图所示，TSP问题的整个求解过程严格按照遗传算法的求解流程，具体的求解过程主要分为如下几步：

第一步，定义初始城市群体。这一步的主要任务是定义哪些城市要进入整个求解过程的候选队列中。在组合优化问题中，也就是有哪些元素进行参加组合优化问题。

第二步，初始化种群。由于遗传算法的计算搜索对象是一个给定的种群，并且当我们取得已经定义好的城市之后，我们要将所有的城市放入一个数组，并且将这个数组中城市的排列顺序进行随机打乱，尽量做到不重复。最终得到各种组合类型的城市数组，这些数组的集合即整个算法的初始种群，其每一种组合形式即种群的一个个体。

第三步，最城市数组集合中的各个个体进行适应度计算。这一步中个体的适应度计算即城市的排列中依次走过路程的长短。我们将这个值作为每一个个体的适应度大小。

第四步，进行种群的选择复制操作。在这一步中我们将按照自然界中的淘汰比率20进行个体淘汰。由于上一步我们已经计算出了各个个体的适应度值，因此我们可以按照种群中个体的适应度值进行由小到大的排序，由于我们最终的结果求得是最小路径，因此，路径长的相应的适应度值也就是越小，我们按照这个规律对整个种群中的个体进行选择淘汰，我们先选取适应度值比较大的进行复制，并淘汰掉其中适应度值比较小的个体。但最终结果保持整个种群的数量不变。

第四步，进行遗传变异操作。这一步是前半阶段是将上一步中选中的个体进行复制之后遗传到下一阶段。之后对新的种群进行变异操作，变异操作我们可以对其中的选定个体中城市的顺序进行重新排序。这样做的目的也是为了避免最终的结果会陷入局部最优解而得不到全局最优解。

第五步，对是否满足终止条件进行判断。对于整个计算过程，我们选定的终止条件是看临近10代的结果有没有比较大的变化，如果变化较大，说明整个优化搜索过程还在继续，如果临近10代的结果没有较大的变化，我们可以判定整个搜索结果都在最优解的周围进行徘徊，因此我们可以终止整个迭代过程，并输出最优解。

模型伪代码：

(1) TSP问题的中体调用类

class TSP\_GA {

public static void main(String[] args) {

1：调用城市个体类并进行城市个体的初始化并保存在数组中

2：初始化整个种群

3：调用选择复制、交换变异算法进行种群迭代

4：输出最终结果

}

}

(2) 整个遗传算法的城市个体类

class City {

1：定义城市类的具体属性

double distanceTo(City city){

2：计算到给定城市的距离

}

}

(3) 保存的整个种群个体和进行种群个体适应度计算的类

class Population {

1: 定义数组保存整个种群个体

Tour[] tours;

// Construct a population

Population(int populationSize, boolean initialise) {

2： 保存整个种群的信息

}

Tour getFittest() {

3： 获取当前种群中最优的个体

}

}

(4)种群的每个个体具体信息类

public class Tour{

1：保持所有城市的路径

private ArrayList tour = new ArrayList<City>();

public void generateIndividual() {

2：创建一个随机个体并将内部的城市顺序打乱

}

public double getFitness() {

3：计算每个个体的适应度值（距离小的适应值较大）

}

public int getDistance(){

4：获取当前个体中路径的当前顺序下的路径总距离

}

}

(5)选择复制，交换变异调用过程

public class GA {

1：定义交换概率和突变概率

double mutationRate = 0.015;

int tournamentSize = 5;

Population evolvePopulation(Population pop) {

1：调用种群交换操作方法

2：调用个体变异方法

}

Tour crossover(Tour parent1, Tour parent2) {

2：对给定的个体进行交换操作

}

void mutate(Tour tour) {

3：对于给定的个体，进行突变操作

}

Tour tournamentSelection(Population pop) {

4：对于给定的个体，进行交换操作

}

}

## 5.2 Hadoop下并行遗传算法模型求解TSP问题

#### 5.2.1 初始种群与编码设计

遗传算法中，第一步也是非常关键的一步就是初始种群的生成以及编码，只有对初始种群及其编码进行合理的设计，才会让以后的步骤顺利的进行。很多时候，一个设计合理的初始种群和编码将会对整个算法的求解起到事半功倍的效果。

对于本文论述的遗传算法求解TSP问题。由于整个问题是实际问题，因此本文将不采用虚拟编码的方式，而采用仿真模拟的方式进行初始种群的生成以及编码。由于在TSP问题的本质为排列组合优化问题，因此我们可以假设所有的城市都在一个二维坐标轴中（都在二维坐标轴的第一象限），因此每一个城市都有一个有序实数对作为其唯一标示，城市与城市之间的距离我们可以采用毕达哥拉斯定理来计算。城市与城市之间的有序组合将是整个种群中的一个个体。由于我们要搜索所有解的可能，因此，整个种群的生成我们将采用穷举法来做。种群初始种群的生成及其编码步骤如下：

1. 初始化所有城市在坐标轴中的位置，我们将所有城市按照初始点距离的远近按照1号到n编号，因此每一个城市的坐标点即L(xi,yi)。
2. 根据第一步中每一个城市坐标点的选取，我们可以随机生成旅行商的各种可能出现的城市旅行路径。我们设Tj = {L1…Ln}为第一种可能的旅行路径，因此其它的旅行路径将是这个旅行路径的各种有序组合形式，这里我们将采用穷举法的方式全部生成。具体生成操作不在本文的论述范围之内。
3. 当第二部结束之后我们得到了旅行商全部可能的旅行路径，也就是各个备选排列组合个体的全部有序排列组合情况。因此我们也就得到了整个遗传算法的初始种群，我们设整个初始种群的为S，从而S={Ti..Tj}。

#### 5.2.2求解TSP问题的遗传算子设计

（1）选择操作算子

一般来说，整个遗传算法在进行个体选择和复制得时候是根据个体的适应度来决定的，本文求解的问题将根据通过交叉、变异操作产生的个体与父类个体种群进行混合替换的策略产生下一代种群个体，这也就是跨时代精英选择算法。在本次的实验中将采用轮盘赌的方式来产生新的种群个体。轮盘赌的方式优点具有以下几点：

1. 择优选择。在下一个的个体选择中，适应度值高的个体将有更大的几率被遗传到下一代，而适应度值低的个体机会将会大大，这非常类似于生物物种的遗传进化原理。

2. 保持物种的多样性。由于轮盘赌方式并没有完全否定适应度小的个体被选取的机会，因此能极大的保持物种的多样性。

3. 排序测量解决了比例适应度计算的尺度问题。

（2）交叉操作算子

（3）变异操作算子

（4）相似度比较算子

相似度比较算子是本文为第一层初始种群筛选而引入的一个算子，这个算子的目的旨在在一部分种群中选出具有强代表性的个体进行第二层计算过程。

#### 5.2.3 求解TSP问题第一层算法模型设计

根据上述描述，本部分将采用基于Hadoop的混合遗传算法针对TSP问题给出第一层的算法模型设计步骤，具体步骤如下：

（1）生成大规模的初始种群，初始种群即为所有城市的不同的有序排列组合的集合；

（2）Map阶段：对种群中的所有个体进行适应度计算，的到带有适应度的全体种群个体；

（3）Shuffle阶段：对整个种群按照适应度的大小进行排序操作，得到具有既定顺序的种群；

（4）Reduce阶段：这个阶段主要是采用相似度比较算子进行种群中个体的相似度比较，从众多的个体中选取出具有代表性的个体，组成信息的种群；

（5）若种群个体符合第二阶段的个体要求数量，则终止迭代；若没有符合，则重新进入第二步操作。

图：

#### 5.2.4 求解TSP问题第二层算法模型设计

第二层设计主要是应用第一层的计算结果，由于第一层的运算结果已经得出一个在数量上比较客观的初始种群数量。因此，我们可以种群带入第二层的Hadoop混合并行遗传算法进行求解运算，具体的求解步骤如下：

（1）初始化第一层的计算结果作为这一层的初始种群；

（2）Map阶段：对所有的种群个体进行适应度计算，确定所有的种群个体的适应度

（3）Shuffle阶段：对上一步得到个体适应度的个体进行分组排序，最终得到具有有序数列的群体。为下一步的选择复制操作提供基础数据。

（4）Reduce阶段：由于在上一步中我们已经得出具有有序数列的种群。这一步的主要任务有两个，一个按照上文定义的选择复制算子即轮盘赌的方式进行选择复制操作。第二个任务主要是按照上文定义的交叉算子和变异算子进行整个种群进行交叉和变异操作，从而使得整个搜索结果不会陷入局部最优解。

（5）终止条件检查，本案例主要采用检查在一定代数内是否其最优解是否有大幅度的变化，如果没有，则输出最优解。

图：

## 5.3 算法测试与分析

## 5.4本章小结

本章节为整个论文论述的核心章节之一，首先介绍TSP问题的传统求解模型，之后引出基于本论文论述的模型即基于Hadoop的混合并行遗传算法模型来求解TSP模型。详细介绍了新模型的每个过程的设计方案，包括初始种群的编码设计，遗传算子的设计以及第一层求解算法和第二层求解算法的设计。从而验证在TSP问题下的基于Hadoop的混合并行遗传算法求解的可行性。在本章节论述的过程中，着重论述了本模型的第一层设计和第二层设计在TSP问题中的衔接应用，表述了这种设计在初始种群过大的前提环境下是如何有效求解的，并给出了本模型的优点。

接着，对于本模型在TSP问题中的应用给出了编程实现，并通过实现数据表述本模型的优点，以及给出了在实际的生产情况下，如果高效的应用本模型进行实际问题求解。

# 第6章 总结和展望

纵览整片文章，本文的核心思想主要基于当今信息数据爆发式增长的环境下，如何对大数据进行有效的处理。Hadoop大数据分析计算平台的出现为当前面临的问题提供了一个优秀高效的平台。但同时，如何将传统的算法框架和求解方法同当今的大数据平台相结合也成为了当务之急。比如早在50年代初期提出的遗传算法，为社会的发展和经济的繁荣起到了不可磨灭的作用。但是，如何将遗传算法应用在大数据的平台上，从而产生一种稳定性高，效率优秀的框架却值得深入研究。本文通过上述背景论述，首先对Hadoop大数据分析计算平台进行了深入的研究，了解Hadoop的核心优势。比如Hadoop的文件存储系统是如何工作的，其分析计算框架MapReduce是如何进行并行运算的。此外，还对其生态系统进行深入的研究，比如依托其文件存储系统hdfs出现的Hbase数据库等。

接下来主要研究传统的遗传算法的原理以及前人们是如何利用遗传算法解决当今社会产生的一些接受的问题，比如在图像识别，天文学领域用传统的建模方式无法解决的问题等。由于遗传算法有着天然的并行计算优势，因此一开始就有很多智者提出了遗传算法并行的可能性。本文基于上述观点首先对遗传算法的整个运算流程做了深入的研究，以及其各种优秀的算子，比如选择复制算子，遗传变异算子等都有深入的讨论。然后针对其并行的优势做了比较深入的探讨和论述，主要在其各种并行方式之间做论证对比，并给出哪种并行结构更有利于适用于当今流行的Hadoop大数据分析计算平台。

结合上述两部分的学习和讨论，本文最终进入核心章节。将Hadoop大数据分析计算平台与传统的遗传算法相结合并提出了基于Hadoop的混合并行遗传算法求解模型。本文的核心章节主要分为四个部分，第一部分主要讨论基于Hadoop的并行遗传算法实现的依据，并从多个方面此模型的可行性。第二部分主要论述遗传算法并行的实现模型主要有几种，并对照这几种实现模型进行研究评价。结合上述部分，第三部分主要给出整个论文的核心模型即基于Hadoop的混合并行遗传算法，并对此模型进行了详尽的描述和可行性证明。紧接着对本模型的优越性进行分析论证。最后一分部主要是模型的验证阶段，本文采取了对当今经济发展具有重要指导意义的TSP即旅行商问题模型进行证明，由于TSP问题设计大量的初始求解个体，并且其个体的数量随着城市数量的增多成指数增长。因此用本文提出的模型求解成为了不二之选。本文通过TSP问题对模型进行验证最终得到实验数据，通过实验数据证明了本模型的可行性和正确性。

本文完成的工作：

算法过程中的不足：

# 参考文献

[1] 李伟卫,赵航,张阳,王勇.基于MapReduce的海量数据挖掘技术研究.计算机工程与用,2012,33(06):1-6.

[2] 孟小峰,慈祥.大数据管理:概念、技术与挑战.计算机研究与发展,2013,50(1):146-169.

[3] 维克托•迈尔-舍尔维恩,肯尼斯•库克耶.大数据时代.浙江:浙江人民出版,2013:38-39.

[4] 刘猛.云计算下神经网络算法的研究.成都:电子科技大学,2011,20(10):21-30.

[5] Dudy Lim,Yew-Soon Ong,YaoChu Jin,BernHard Sendhoff,Bu-Sung Lee.Efficient Hierarchi- cal Parallel Genetic Algorithms Using Grid Computing,EI.Nanyang Technological Universi- ty,2006,10:5-15.

[6] Abhishek Verma,XavierLlor`a,David E.Goldberg,RoyH.Campbell.Scaling Genetic Algorith- ms Using MapReduce.International Conference,EI Index,2009,2:1-6.

[7] Di-Wei Huang.A Genetic Algorithm for JobShop Scheduling Problems using MapReduce(EI 检索).International Conference,2013,6:1-6.

[8] 李剑锋,彭舰.云计算环境下基于改进遗传算法的任务调度算法.计算机应用,2012,11(10):1-6.

[9] 徐肖,胡吉明.一种 Hadoop中基于改进遗传算法的作业调度算法.南京:河海大学,2013,6(8):15-25.

[10] 雷万云.后危机时代集团企业信息化建设策略.e 制造杂志,2009,12:2-10.

[11] Peter Fingar.云计算21世纪商业平台.电子工业出版社,2012:20-28.

[12] 雷万云.云计算-技术、平台及应用案例.北京:清华大学出版社,2011:7-250.

[13] 刘义军.基于云计算平台的个人信息融合系统的研究与实现.北京:北京邮电大学,2010:8-12.

[14] 张建.基于Hadoop的云计算模型研究与应.南京:南京信息工程大学,2012:13-27.

[15] 邓自立.云计算中的网络拓扑设计和Hadoop平台研究.合肥:中国科学技术大学,2009,9(2):10-15.

[16] 张继平.云存储解析.人民邮电出版社,2012:30-100.

[17] Brandeau M,Chiu S S.An overview of representative problems in location resear- ch.Management Science,1989,35:645-674.

[18] 葛春景,王霞,关贤军.重大突发事件应急设施多重覆盖选址模型及算法.运筹与管理,2011,20(5):50-56.

[19] Holland J H.Adaptation in natural and artificial systems.University of Michigan Press,Ann Arbor Mich,1975,13:1-7.

[20] Dino Keco,Abdulhamit Subasi.Parallelization of genetic algorithms using Hadoop MapReduce.Southeast Europe Journal of Soft Computing,2012:1-5.

# 攻读硕士学位期间公开发表的论文

*Proceedings - 2015 International Conference****on****Network and Information Systems for Computers, ICNISC 2015*, p 231-235, October 28, 2015, *Proceedings - 2015 International Conference****on****Network and Information Systems for Computers, ICNISC 2015*;  **ISBN-13:** 9781479918423;  **DOI:** 10.1109/ICNISC.2015.62;  **Article number:** 7311875; **Conference:** International Conference **on** Network and Information Systems for Computers, ICNISC 2015, January 23, 2015 - January 25, 2015; **Publisher:** Institute of Electrical and Electronics Engineers Inc.

# 致谢

时光荏苒，研究生生活一晃而过。我深切的感受到，研究生生活将会给我即将要踏上的征途中增添无尽的财富。

在这里，我首先要感谢我的研究生导师薛大伸教授，是他在这三年的研究生生活中给予了我无私帮助。尤其在最后的论文指导中，他一丝不苟的治学精神、深邃的洞察力、远见卓识的战略思想以及强调学以致用的实事求是精神，无不感染着我时刻提醒自己要以一种慎独的态度对待我的毕业论文以及将来的事情。

另外，感谢我的导师在我读研期间给我创造的实习机会，还有给别人做指导的机会，这些经历无不在我的自我成长过程中起到了重要的作用。

我还要感谢在我读研期间给予各种指导的教授，老师们，感谢你们的时间付出，感谢你们的治学精神以及你们对我生活上的帮助和关怀。没有你们，我也许会走更多的弯路，也许会被前进道路上的困难阻碍不得前进。

还有，我的同门师兄，师姐，师弟，师妹们，感谢你们在我的研究生期间的各种支持和关怀。也感谢有你们陪伴的在研究生期间一块度过的快乐时光。在这里衷心祝福你们顺利完成研究生学业，事业有成。

感谢我的母校，大连海事大学，感谢你给了我一个发奋进取的平台。

感谢我的家人，尤其感谢我父母，感谢你们无时不刻的关怀与支持。

感谢导师以及答辩委员会的各位专家、教授、老师以及同我一起并肩作战的同学、朋友以及给予我宝贵意见的各位。