中国科学院大学

University of Chinese Academy of Sciences

博士学位论文

基于容器化多终端协同服务技术研究

作者姓名：赵然

指导教师：倪宏 研究员 中国科学院声学研究所

学位类别：工学博士

学科专业：信号与信息处理

培养单位：中国科学院声学研究所

2019 年 6 月

LaTeX Thesis Template
of
The University of Chinese Academy of Sciences $\pi\pi^{\pi}$

A thesis submitted to the

University of Chinese Academy of Sciences

in partial fulfillment of the requirement

for the degree of

Doctor of Engineering

in Signal and Information Processing

By

Zhao Ran

Supervisor: Professor Ni Hong

Institute of Acoustics, Chinese Academy of Sciences

June, 2019

中国科学院大学
学位论文原创性声明

本人郑重声明：所呈交的学位论文是本人在导师的指导下独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明或致谢。本人完全意识到本声明的法律结果由本人承担。

作者签名：

日　　期：

中国科学院大学
学位论文授权使用声明

本人完全了解并同意遵守中国科学院大学有关保存和使用学位论文的规定，即中国科学院大学有权保留送交学位论文的副本，允许该论文被查阅，可以按照学术研究公开原则和保护知识产权的原则公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存、汇编本学位论文。

涉密及延迟公开的学位论文在解密或延迟期后适用本声明。

作者签名：　　导师签名：

日　　期：　　日　　期：

摘 要

本文是中国科学院大学学位论文模板 ucasthesis 的使用说明文档。主要内容为介绍 LaTeX 文档类 ucasthesis 的用法，以及如何使用 LaTeX 快速高效地撰写学位论文。

关键词：中国科学院大学，学位论文，LaTeX 模板

Abstract

This paper is a help documentation for the LaTeX class ucasthesis, which is a thesis template for the University of Chinese Academy of Sciences. The main content is about how to use the ucasthesis, as well as how to write thesis efficiently by using LaTeX.

Keywords: University of Chinese Academy of Sciences (UCAS), Thesis, LaTeX Template

目 录

图形列表

表格列表

符号列表

字符

Symbol	Description	Unit
R	the gas constant	$m^2 \cdot s^{-2} \cdot K^{-1}$
C_v	specific heat capacity at constant volume	$m^2 \cdot s^{-2} \cdot K^{-1}$
C_p	specific heat capacity at constant pressure	$m^2 \cdot s^{-2} \cdot K^{-1}$
E	specific total energy	$m^2 \cdot s^{-2}$
e	specific internal energy	$m^2 \cdot s^{-2}$
h_T	specific total enthalpy	$m^2 \cdot s^{-2}$
h	specific enthalpy	$m^2 \cdot s^{-2}$
k	thermal conductivity	$kg \cdot m \cdot s^{-3} \cdot K^{-1}$
S_{ij}	deviatoric stress tensor	$kg \cdot m^{-1} \cdot s^{-2}$
τ_{ij}	viscous stress tensor	$kg \cdot m^{-1} \cdot s^{-2}$
δ_{ij}	Kronecker tensor	1
I_{ij}	identity tensor	1

算子

Symbol	Description
Δ	difference
∇	gradient operator
$\delta^{\pm}$	upwind-biased interpolation scheme

缩写

CFD	Computational Fluid Dynamics
CFL	Courant-Friedrichs-Lewy
EOS	Equation of State
JWL	Jones-Wilkins-Lee
WENO	Weighted Essentially Non-oscillatory
ZND	Zel’dovich-von Neumann-Doering

第 1 章　基于容器化服务资源提供技术

1.1　引言

随着计算机技术的快速发展和边缘计算技术的逐渐成熟，位于网络边缘的用户终端设备在用户的数字化生活中正扮演着越来越重要的角色，其定位逐渐从单一的用户服务发起者向用户服务的提供者转变。这会使用户终端承担越来越多的计算任务，这也对终端设备的服务质量和计算能力提出了越来越高的要求。为了满足用户对于服务质量的要求，人们提出了计算迁移技术，将终端设备上的计算任务通过网络迁移到云端服务器上来完成。但是引入云端协同的计算迁移技术，还是存在着时延较大、数据隐私性不能保证、成本较高等问题，难以满足用户需求。而另一方面，边缘终端设备距离用户更近，而且拥有很多空余资源没有得到充分利用，如何将这部分距离用户更近、成本更低的资源组织利用起来为用户设备提供计算迁移服务也成为了计算迁移技术中的一个值得研究的问题。本文以 HTML5 中的 Web Worker 方法为例，研究 Web 应用透明计算迁移到以容器形式部署在边缘设备上的服务端，设计并实现了一种基于容器的 Web Worker 透明边缘计算迁移系统，并利用若干实验来验证该系统能够将周围设备的资源利用起来，减少计算总执行时间，提升用户体验。

1.2　相关研究

1.2.1　计算迁移技术

传统的终端设备的服务模式通常是终端设备在本地进行计算任务的运行，依靠终端设备自身的计算能力来保证服务质量。而随着边缘智能终端设备所要承担的计算任务越来越重，单一终端设备本地计算难以满足终端服务和任务对终端计算能力的要求，计算迁移技术逐渐成为了一个可行的解决方案。计算迁移技术，也被称为计算卸载技术（computation offloading），是指将终端设备上的计算任务，通过网络的形式发送到其他设备上进行计算，再将计算结果通过网络返回给终端设备。

在边缘计算中，通常使用的方法是将边缘终端设备上的计算任务根据具体需求迁移一部分或迁移全部到云端计算资源上进行处理。云端计算资源通常是部署在云数据中心的机房，可以提供计算、存储、网络等资源。这种计算模式又被称为 Mobile Cloud Computing (MCC)。使用 MCC 的计算模式，可以让终端设

备提供更加复杂的计算服务，并且减少终端设备的能源消耗。但另一方面，终端设备与云数据中心之间会出现大量数据交互，产生大量网络延迟，并且可能会出现信息泄露等隐患，终端服务的实时性与安全性都不能得到保证。

为了拉近云数据中心与用户终端之间的距离，云端计算资源也会部署在网络边缘，例如基站、边缘服务器等等。这种计算模式又被称为 Mobile Edge Computing (MEC)。与 MCC 类似，MEC 也是一种集中式的云计算资源，需要资源服务器、中央管理器、移动网络等基础设施的支持，部署起来会比较复杂。目前对于 MEC 中计算迁移技术的的研究，主要集中在计算迁移决策、计算资源管理及移动性管理三个方面，而对与 MEC 系统中的计算迁移方案的设计与实现研究较少。

文献【Cloudlets: Bringing the cloud to the mobile user】中介绍了一种计算迁移系统，利用部署在用户身边的可信的、计算资源丰富的计算设备（被称为 cloudlets）为移动用户提供服务。当用户在终端设备上产生服务请求时，终端设备可以在 cloudlets 上快速建立定制化的虚拟机，将计算任务迁移到对应虚拟机中并利用其资源来运行，用户可以利用轻量级客户端通过无线网络进行访问。cloudlets 模式是一种细粒度的计算迁移技术 []。这种计算迁移模式并没有在远端虚拟机中运行整个应用程序，而是将应用动态分割成多个可单独运行的组件，综合考虑每个组件的资源消耗情况、当前 cloudlets 状态、组件之间的依赖关系以及计算迁移决策等问题，来决定每个组件是否进行计算迁移。cloudlets 计算迁移模式相比 MCC 和 MEC，与用户之间的距离更近，传输时延更小，非常适合图书馆、咖啡厅、办公室等聚集场所。但 cloudlets 的缺点是需要在区域内单独进行部署，需要额外的硬件、场地、服务器等成本，使用范围有局限性。

1.2.2 Web Worker

Web Worker 是一种基于 HTML5 的多线程方法，本文的研究对象就是面向 Web Worker 的计算迁移系统。基于 Web Worker 的服务系统可以拥有更好的跨平台性，应用开发者在开发服务应用的过程中可以不必考虑底层的系统架构。基于 Web Worker 的计算迁移方法分为透明迁移和非透明迁移两类。基于 Web Worker 的非透明迁移通常采用的方法是重写标准的 Web Worker 的接口 API，并在编写 Web 应用的时候以新的库的形式导入修改，使得 Web 应用在运行的时候通过 WebSocket 通知运行在云端的服务端程序生成相应的 Web Worker 来进行计算 [1234]。文献 [] 中提出了一种基于 Web Worker 的透明迁移方法，通过修改 Web Worker 运行环境代码来实现计算迁移的过程，但是没有做具体实现。

1.2.3 虚拟化技术

很多计算迁移系统中也用到了虚拟化技术。虚拟化技术是一种资源管理技术，利用底层虚拟、上层隔离等方法，将计算机中的物理资源，如计算、存储和网络等，进行抽象、切割，以更好的形式提供给用户使用 []。当管理多个计算机资源的时候，利用虚拟化技术还可以将整个计算机集群进行虚拟化，形成资源池，可以更方便地管理计算机集群中的资源，也可以按需求给用户提供相应的虚拟资源，提高资源利用率，并且可以对用户保持透明，用户可以不必关心底层的资源是如何进行虚拟化和提供的。传统的抽象方法通常是硬件虚拟化，在计算机底层上建立 hypervisior 层，来对底层硬件进行虚拟化，并在其上运行虚拟化的操作系统。基于硬件虚拟化技术运行的实体通常被称为虚拟机，也叫 Virtual Machine (VM)。另外一种比较流行的抽象方法是操作系统 (OS) 虚拟化，这种虚拟化技术利用 LXC 技术在宿主机的操作系统之上进行隔离、封装。基于 OS 虚拟化技术运行的实体通常被称为容器，也叫 Container。

很多计算迁移的研究 [1-4] 都使用了虚拟机 VM 来作为云端承载迁移服务程序的基础。然而因为需要对底层硬件做虚拟化，所以使用虚拟机 VM 会引入极高的启动时延，这对于响应时间敏感的终端用户服务来说是难以接受的。如果考虑提前部署虚拟机 VM 的方案，虽然能够减少启动时间带来的时延，但相对重型的虚拟机 VM 解决方案也会带来较大的额外开销，这也是一种不太能够接受的服务费用以及资源浪费 []。文献 [] 中提出使用基于 OS 虚拟化技术的容器来代替虚拟机 VM，承载云端迁移服务程序。这种轻量级的虚拟化技术不仅能够大大减少启动时间带来的时延，也能够大大降低虚拟化技术带来的额外资源开销。

1.3 基于边缘容器的 Web Worker 透明计算迁移系统设计

为了将用户身边终端设备的空闲资源利用起来，并且方便部署管理，对 Web 应用开发者透明，本文提出一种基于边缘容器的 Web Worker 计算迁移系统。计算迁移系统的实质是利用冗余来提高终端服务系统的服务质量，而对于用户的终端环境来说，用户身边还有着大量的终端设备都处于空余状态，拥有大量的空闲资源可以用来提供计算迁移服务。同时，相比其他边缘计算迁移系统，这种身边的终端距离用户所使用的终端距离更近，网络状况、时延等情况更优，计算迁移服务质量也会更好。虽然终端空闲资源总数庞大，但是与集中式云端资源相比，终端资源分布得比较零散，不方便部署，因此我们使用 swarm 容器集群来对提供迁移服务的终端空余资源进行管理和应用。另一方面，我们修改 Web 运行

环境代码，设计对 Web 应用开发者透明的计算迁移系统，将 Web Worker 迁移到服务端进行运行。

1.3.1 系统设计，结构，模块，流程

基于边缘容器的 Web Worker 计算迁移系统分为客户端与服务端。图？为 Web Worker 计算迁移系统的模块图。客户端也就是用户终端上的 Web 运行环境，可以接收用户请求，通常会根据用户请求生成以多个 Web Worker 形式承载的计算任务。当用户终端资源不足以完成该计算任务，或者完成时间较长，希望能够缩短计算时间提高用户体验的时候，客户端程序可以将计算任务以 Web Worker 的形式迁移到服务端来完成。服务端是一个能够接收计算迁移任务请求的应用程序，并在本地生成 Web Worker 运行迁移过来的计算任务。服务端通常以容器的形式部署在周围有空余计算资源的其他终端上，并受整个边缘容器集群管理。需要指出的是，边缘容器集群中是由多个用户终端设备组成，每个终端设备在有计算任务的时候都可以成为客户端，向其他设备迁移计算任务，在没有计算任务的时候也都可以成为服务端，接受其他设备迁移过来的计算任务。

1.3.2 透明迁移的客户端

客户端包含 Web 应用和重写的 Web 运行环境两个部分。Web 运行环境在本研究中主要指运行在用户终端上的 Web 浏览器，它能够为 HTML5 及 Javascript 提供标准支持。因为设计的计算迁移系统为透明迁移，所以 Web 应用只需要调用标准的 Web Worker 接口就可以生成 Web Worker 并与其进行通信，而不需要对 Web 应用本身做任何特殊的改动，Web 应用开发者也不需要了解底层是如何实现的。底层的 Web 运行环境被重新改写，主要包含 Web Worker 管理端和 websocket 通信客户端。当系统决定要进行计算迁移的时候，Web Worker 管理端在收到上层 Web 应用通过标准接口传来的 Web Worker 生成的请求的时候，会将该请求进行翻译并重新封装，Web 运行环境中的 websocket 通信客户端模块会通过 websocket 将封装后的请求发送到服务端进行处理。

在后续的通信中，客户端与运行在服务端的 Web Worker 也要进行很多信息交互，这些通信也是以 websocket 的形式进行的，因此需要为 websocket 设置几种通信标志，来区分不同的通信类型。本研究中主要设置三种通信标志：

- ESTABLISH: 客户端请求服务端创建新的 Web Worker，由服务端直接进行处理
- COMMUNICATE: 客户端向服务端发送的通信信息，由服务端接收后转发

给服务端对应的 Web Worker

• TERMINATE: 客户端请求服务端停止对应的 Web Worker，由服务端直接进行处理

添加通信标志的操作是在客户端中的 websocket 通信客户端重新封装请求的过程中实现的。这些通信标志可以让服务端区分该信息是控制类型（ESTABLISH、TERMINATE）还是数据类型（COMMUNICATE），并根据通信类型的不同采取不同的处理方式。

1.3.3 基于容器的服务端

图？中右侧为服务端程序的模块。服务端程序包括底层的 Javascript 运行环境、websocket 通信服务端以及 Web Worker 管理端。Javascript 运行环境用来提供 Javascript 标准接口，支撑整个服务端程序以及 Web Worker 的运行。websocket 通信服务端用来接受客户端发来的消息，并且对其解封装，根据通信标志的不同，做相应的处理。Web Worker 管理端，接受通信服务端传来的指令，调用 Web Worker 标准接口，实现对 Web Worker 生命周期的管理，包括 Web Worker 的创建、信息交互及销毁。

图？为整个系统的整体架构图。服务端程序是以容器的形式部署在边缘 Docker Swarm 集群上的。这个过程需要使用 Dockerfile 生成安装有对应运行环境及服务端程序的 Docker 镜像，并将生成的镜像上传到本地镜像仓库中。接下来需要由 Docker Swarm 集群从镜像仓库中拉取对应镜像，按照对资源以及服务规模的需求生成相应的迁移服务，并通过 Docker Swarm 的调度器在合适的终端节点上启动相应数量的副本实例，也就是运行着服务端程序的容器，打开对应端口，为用户终端提供计算迁移服务。每个容器在哪个节点上运行是由 Docker Swarm 调度器上执行的调度策略来决定的。

计算迁移的过程被分为 4 个阶段，运行时准备阶段，网络连接阶段，数据传输阶段，任务执行阶段 [?]。在运行时准备阶段中，服务端使用 Dockerfile 生成安装有对应运行环境及服务端程序的 Docker 镜像，或者直接从本地镜像仓库中下载对应镜像，创建对应服务，在节点上部署对应容器，并对外暴露服务端口。在网络连接阶段中，客户端接收到用户请求，Web 应用根据用户请求向底层 Web 运行环境发送 Web Worker 的创建请求，客户端的 Web Worker 管理端会将带有创建 Web Worker 所需要的信息的 JS 文件的 URL 加上一个 ESTABLISH 标志，封装后交给客户端的 websocket 客户端。此时 websocket 客户端会向服务端暴露出

的服务 IP 和端口请求建立连接，通过服务端 Docker Swarm 的调度以后，客户端与某个容器上运行的服务端的 websocket 服务端端建立网络连接。websocket 服务端在接收到消息后会进行解封装，读取通信类型标志并根据消息内容中包含的 URL 下载对应 JS 文件。然后服务端上的 Web Worker 管理端会在对应容器中运行该 JS 文件，创建 Web Worker。在数据传输阶段中，websocket 客户端发送的信息带有 COMMUNICATE 通信标志，代表信息中包含的内容是 Web 应用发送给 Web Worker 的数据信息，websocket 服务端解析到该标志后，会直接将信息转发给在容器中运行的 Web Worker。在任务执行阶段中，运行在边缘容器中的 Web Worker 会利用容器中配置的资源，执行对应计算任务，并将计算结果通过网络返回给用户的客户端设备。当该客户端接收到某个 Web Worker 返回的全部数据，则会向该服务端发送带有 TERMINATE 标志的信息，服务端在接收到该消息后，会销毁对应容器中运行的 Web Worker，释放相关资源。此时该容器并不会随之销毁，而是可以等待下一次计算迁移任务，直到整个 Docker Swarm 集群对该容器的生命周期进行调整。

1.4 实验结果及分析

1.4.1 实验配置

本研究中使用树莓派设备来模拟用户终端设备及边缘终端设备，搭建基于容器的 Web Worker 透明边缘计算迁移系统，并测试其对于用户服务体验的提升效果。本实验中的客户端以及服务端集群使用的树莓派型号为 raspberry pi3，配置为 1G 内存，4 核处理器。服务端 Docker Swarm 集群由局域网内的 4 个树莓派组成，其中一个作为 master，另外三个作为 worker，每个树莓派节点上都可以作为计算迁移服务的执行节点。另外为了与基于容器的边缘集群的性能做对比，我们还使用了一台局域网内的 Dell PowerEdge R 730 服务器作为服务端来进行实验。本实验中客户端的 Web 运行平台为 chromium 浏览器，服务端的 JS 执行环境为基于 V8 引擎的 Node.JS。

本实验中的测试用例为利用一个利用 Web Worker 的图形渲染 Web 应用 Ray Tracing[http: / / nerget. com / rayjs-mt / rayjs. html]。在这个应用中，Web 应用需要渲染加载一张图片，具体过程是将图片切割成若干份（本实验中为 20 份），启动一定数量的 Web Worker 来进行渲染加载。这个测试用例中，图片总大小是一样的，也就是总的任务量一定，使用的 Web Worker 数量越多，每个 Web Worker 计算的任务量就越少，同时 Web Worker 数量与切割份数不一定相同，可能出现一

个 Web Worker 需要处理多份任务的情况。在本实验中，Ray Tracing 应用可以不做任何修改，直接运行在客户端树莓派的 chromium 上，然后改写后的浏览器将计算任务迁移到边缘树莓派集群或服务器上，创建 Web Worker 并进行计算，最后将结果返回给客户端，浏览器将图片显示给用户。本实验中对基于容器的 Web Worker 透明边缘计算迁移系统服务性能的评价指标为 Web Worker 的总体执行时间，即从第一个 Web Worker 开始创建，到最后一个 Web Worker 执行完成所消耗的时间。

1.4.2 实验结果

为了测试不迁移、迁移到服务器以及迁移到边缘树莓派集群上的服务性能，我们做了实验一，图片大小为 200*200，切分为 20 份 200*10 的小图片，交给不同数量的 Web Worker 来进行计算，实验结果如图？所示，其中横坐标表示每一次实验中的 Web Worker 数量，纵坐标表示该次实验中的 Web Worker 执行总时间。

从图? 中可以看出，不使用计算迁移系统的时候，执行总时间最长，服务性能最差，而且随着 Web Worker 数量的增加，性能变差得非常明显，让用户的服务体验变得非常差。迁移到边缘树莓派集群上的时候，相比不迁移的情况，性能提升明显，虽然执行时间比迁移到服务器的情况稍差，但相差不是很大，还是能够保持在用户可接受范围内。另外随着 Web Worker 数量的增加，迁移到边缘树莓派集群的执行总时间会慢慢增加，但增长趋势比较缓慢，不会出现不迁移时候的明显变差的情况。迁移到服务器上的时候，执行时间最短，尤其是在 Web Worker 数量为 1 的时候，优势非常明显，但这更多的是基于服务器本身性能要远远好于单个树莓派的原因。在实际应用场景中，使用服务器来提供计算迁移服务的成本要远远高于使用边缘终端设备，而且实际场景中的服务器通常位于云端数据中心，距离用户终端设备较远，网络通信开销会增大，同时还会受限于网络状况，服务性能很可能达不到这么好的效果。

为了测试透明迁移到边缘树莓派集群与非透明迁移到集群的服务性能差异，我们做了实验二，图片大小为 400*400，切分为 20 份 400*20 的小图片，交给不同数量的 Web Worker 来进行计算，实验结果如图？所示，其中横坐标表示每一次实验中的 Web Worker 数量，纵坐标表示该次实验中的 Web Worker 执行总时间。

从图？中可以看出，相比实验一，图片大小变大，总任务量变多，总的计算

时间也相应增加。不使用计算迁移系统的时候，执行总时间还是最长，服务性能最差，而且随着 Web Worker 数量的增加，性能变差得非常明显，让用户的服务体验变得非常差。另外值得注意的一点是，不适用计算迁移系统的时候，在使用 4 个 Web Worker 来执行任务的时候，总执行时间最短，这可能与树莓派设备是 4 核处理器有关，当 Web Worker 数量小于 4 个的时候，会出现树莓派的计算能力没有被充分利用的情况，当 Web Worker 数量大于 4 个的时候，会出现各个线程之间抢占 CPU 而导致切换时间大大增加影响总执行时间的情况。使用计算迁移后，总的执行时间都会大大缩短，当 Web Worker 数量多于 4 个的时候，边缘树莓派集群中多设备多核处理器的优势就体现出来了，当 Web Worker 数量增加的时候，总执行时间也不会随之增加很多，稳定在一个比较短的时间内。对比透明计算迁移方式和非透明计算迁移方式的实验结果，可以看出，透明计算迁移方式的总执行时间也要比非透明计算迁移方式的总执行时间要短一些，整体服务效果更好。

1.5 本章小结

本章。。。。。

第 2 章　资源受限终端任务调度策略

2.1　引言

任务调度问题是指系统在同时接收到多个服务请求任务的时候，将这些任务合理地分配给多个智能终端上的容器进行处理，由于不同的容器所在智能终端的处理能力不同，每个容器所占用的资源也不同，导致不同的调度方案结果会有差异，需要针对在容器化智能终端协同服务场景下的一些特点来进行取舍和进一步的优化，以追求对于终端资源利用的最大化。在该任务场景下，最常见的用户需求就是实时性需求，也即要求任务能够被快速响应、快速执行、且执行结果能够快速回传给用户，因此最小化任务完成时间是任务调度问题最主要的目标。除此之外，需要考虑的因素还涉及硬件设备功耗、分布式设备负载均衡度等。

2.2　相关工作

2.2.1　传统任务调度问题求解方法

任务调度问题是 NP-hard 问题，已有的任务调度算法主要分为两大类，一类是基于局部贪心策略的算法，另一类是启发式的智能搜索方法。贪心策略的算法，例如优先调度任务量最重的算法，或者优先调度计算时间最短的算法，其调度结果很容易陷入局部最优解，不能够合理利用所有资源。启发式算法则是确定搜索策略，通过迭代、评价等方式，逐步逼近全局最优解。

任务调度问题是将多个任务计划到约束下的多个节点的问题。任务调度问题是一个优化问题。应用多种算法来解决任务调度问题。基于最佳资源选择 (BRS) 的算法, 如最大最小、最小、苦难等, 是解决任务调度问题的传统方法。一些元启发式算法, 如 PSO 和基于 pso 的改进算法, 是处理任务调度问题的新方法。

2.2.2　启发式算法求解方法

近年来, 对任务调度问题进行了大量的研究。随着研究的进行, 许多元启发式算法被用来处理复杂的优化问题。元启发式算法具有简单的操作和较少的开销, 能够找到全局最优。

启发式算法中，如遗传算法（Genetic Algorithm），是模拟达尔文生物进化论的自然选择和遗传学机理的生物进化过程的计算模型，通过模拟自然进化过程

搜索最优解。遗传算法将一个调度方案表示为一个染色体，并利用交叉、变异等运算符实现调度方案的进化，最终得到的结果比较好，但是收敛速度较慢且运算量较大。模拟退火算法（Simulated Annealing Algorithm），仿照固体退火原理，是一种基于概率的算法，从某一较高初温出发，伴随温度参数的不断下降, 结合概率突跳特性在解空间中随机寻找目标函数的全局最优解，即在局部最优解能概率性地跳出并最终趋于全局最优。Kennedy 和 Eberhart 在 1995 年提出粒子群算法（Particle Swarm Optimization），该算法最初是受到飞鸟集群活动的规律性启发，进而利用群体智能建立的一个简化模型，通过追随当前搜索到的最优值来寻找全局最优。

许多元启发式算法被引入到优化问题中。GA 是戈德伯格在 1988 年提出的经典元启发式算法, 它将自然选择理论引入到优化过程中, 包括突变、交叉和选择 citefonseca1995an,Whitley1994, Tanese1989DGA, Ng 广州市 2018。虽然 GA 的性能相当好, 但遗传算法的操作过于复杂, 无法实现, 不适合某些情况。一些元启发式算法的灵感来自昆虫、鱼类、鸟类和其他群体生物的自然行为。粒子群算法 (PSO) 是肯尼迪 1995 年提出的经典元启发式算法。PSO 的原理简单, 性能显著 1995 年的柠檬素颗粒, Liao2007, Gomathi2013。蚁群优化算法 (ACO) 是受蚂蚁在鸟巢和食物来源之间自然觅食行为的启发。ACO 利用化学信息素在中国的人群中进行交流。

最近提出了一些新的元启发式优化算法。关于改进这些算法的研究并不多。蚂蚁狮子优化器提出 (ALO) 在 2015 年是受狮子 citemirjilili2013 蚂蚁的狩猎行为的启发。鲸鱼优化算法 (WOA) 于 2016 年提出。WA 模拟鲸鱼的自然狩猎行为。2016 年提出的蜻蜓算法 (DA) 是受蜻蜓群在自然界中的静态和动态行为的启发。

2017 年, Shahrzad Saremi 和 Seyedali Mirjalili 提出了一种新的元启发式优化算法, 称为蝗虫优化算法 (GOA)。GOA 利用群体内部的相互作用和蜂群外的风的影响来模拟蝗虫群的迁徙行为, 寻找目标食物 citesare2013 2017 蝗虫。GOA 算法利用群智能, 通过在蝗虫群之间分享经验, 确定搜索方向, 找到最佳或近似的最佳位置。GOA 还使用了具有多个迭代的进化方法, 以提高群智能的效率。

开发了一些基于 GOA 的改进算法。OBLGOA 是由艾哈迈德 • 埃维斯在 2018 年提出的 ceewees 2013 改进。根据目前的搜索位置, 引入了基于对立面的学习策略, 以生成相反的解作为候选方案。OBL 策略可以提高算法的收敛速度, 但由于缺乏随机性, 改进有限。桑卡拉阿罗拉提出了混沌蝗虫优化算法在 2018 年 柠檬酸乱糟糟。将混沌映射应用到算法中, 提高了 GOA 的性能。采用 10 幅混沌映

射来评价混沌理论的影响。结果并不是特别理想, 因为混沌因素在处理许多基准函数时并不合适。提出了一种基于 GOA 的新算法来解决优化问题和任务调度问题。

2.3 GOA 算法

蝗虫优化算法模拟蝗虫的昆虫群行为。蝗虫蜂拥而至, 远距离迁徙, 寻找一个有食物的新栖息地。在这个过程中, 蝗虫之间的互动在蜂群内部互相影响。风的力量和蜂群外的重力影响蝗虫的轨迹。食物的目标也是一个重要的影响因素。

受上述三个因素的影响, 移民过程分为勘探和开发两个阶段。在勘探阶段, 鼓励蝗虫快速、突然地移动, 寻找更多潜在的目标区域。在开发阶段, 蝗虫往往在当地移动, 以寻找更好的目标地区。蝗虫自然实现了勘探开发寻找食物来源的两种迁徙趋势。此过程可以抽象为优化问题。蝗虫群被抽象为一群搜索代理。

Seyedali Mirjalili 在文献 [] 中提出了蝗虫群体迁移的数学模型。具体的模拟公式如下：

$$X_i = S_i + G_i + A_i \tag{2.1}$$

这里变量 X_i 是第 i 个搜索单位的位置，变量 S_i 代表蝗虫集群内部搜索单位间社会交互对第 i 个搜索单位的影响因子，变量 G_i 代表蝗虫集群外部重力因素对第 i 个搜索单位的影响因子，变量 A_i 代表风力的影响因子。变量 S_i 的定义公式如下：

$$S_i = \sum_{j=1, j\neq i}^{N} s(d_{ij})\widehat{d_{ij}} \tag{2.2}$$

这里变量 d_{ij} 代表第 i 个搜索单位和第 j 个搜索单位之间的欧式距离，计算方法如下：

$$d_{ij} = |x_j - x_i| \tag{2.3}$$

变量 $\widehat{d_{ij}}$ 代表第 i 个搜索单位和第 j 个搜索单位之间的单位向量，计算方法如下：

$$\widehat{d_{ij}} = \frac{x_j - x_i}{d_{ij}} \tag{2.4}$$

s 是一个函数，用于计算蝗虫集群之间的社会关系影响因子，该函数定义如下：

$$s(r) = fe^{\frac{-r}{l}} - e^{-r} \tag{2.5}$$

这里 e 是自然底数，变量 f 代表吸引力因子，参数 l 代表吸引力长度。当应用于解决数学优化问题的时候，为了优化数学模型，公式 1 中需要加入一些适当的改动。代表集群外部影响因子的变量 G_i 和 A_i 需要被替换为目标食物的位置。这样计算公式就变成了如下的样子：

$$x_i = c\Big(\sum_{j=1,j\neq i}^{N} c\frac{u-l}{2}s(|x_j - x_i|)\frac{x_j - x_i}{d_{ij}}\Big) + \widehat{T}_d \tag{2.6}$$

这里参数 u 和参数 l 分别代表搜索空间的上界和下界。变量 $\widehat{T}_d$ 是目标食物的位置，在优化问题的数学模型中代表所有搜索单位在整个搜索过程中所能找到的最优的解的位置。另外，参数 c 是搜索单元的搜索舒适区控制参数，改变参数 c 的大小可以平衡搜索过程中的“开拓”和“探索”两个阶段。参数 c 的计算方式如下：

$$c = cmax - iter\frac{cmax - cmin}{MaxIteration} \tag{2.7}$$

这里参数 *cmax* 和参数 *cmin* 分别是参数 c 的最大值和最小值，参数 *iter* 代表当前的迭代次数，参数 *MaxIteration* 代表最大迭代次数。

在优化问题的求解过程中，公式 4 作为演进公式，被不断循环迭代来寻找最优解，直到达到迭代终止条件为止。通常迭代终止条件为达到预设的最大迭代次数，或者所得到的最优解满足预设的最优解条件。在本研究所涉及到的优化问题中，迭代终止条件均为达到预设的最大迭代次数。在迭代演进的过程结束后，该算法可以得到一个近似的最优解的位置以及相应的最优解的值。

蝗虫优化算法的算法伪代码如算法 1 所示：

2.3.1 蝗虫算法优缺点分析

GOA 是最近受到蚱蜢自然迁移行为启发的元启发式算法。虽然 GOA 具有简单的理论基础并且易于实现，但 GOA 的性能更优越。原始的 GOA 改变了蚱蜢的舒适区域，这可以使目标通过迭代收敛到全局最优解。与一些经典算法相比，GOA 的收敛速度要快得多。GOA 可显着提高蝗虫的平均适应度，改善蝗虫

算法 1 蝗虫优化算法

initialize the swarm and set the position boundaries u and l
initialize the factors including cmax, cmin, MaxIteration
initialize all the search agents position with random origin matrix
calculate the target fitness and mark the target position
while $(iter < MaxIteration$ and $target fitness > destination fitness)$ **do**
    calculate $d_i j$ and $\widehat{d_i j}$ by equation(2)
    calculate $s(d_i j)$ by equation(3)
    update x_i by equation(4)
    calculate the fitness
    if current fitness is better than target fitness **then**
        update the target fitness and the target position
    end if
    $iter = iter + 1$
end while
Return target fitness and target position

的初始随机种群。虽然 GOA 具有这些优点，但它也存在一些阻碍算法获得更好解决方案的缺点。在对 GOA 公式进行一些理论分析和用 MATLAB 代码进行的一些实验之后，给出了 GOA 的几个缺点。

首先，GOA 使用线性递减参数使搜索过程收敛，这很难区分过程的两个阶段，即开发阶段和探索阶段。其次，在搜索过程的早期阶段的每次迭代期间，最佳解决方案的位置急剧波动。似乎最终的最佳解决方案只受搜索过程后期的影响，无论前一阶段的结果如何。GOA 理论无法充分利用所有搜索迭代。当最大迭代次数增加时，GOA 的最佳适应性不是很突出，例如，从实验结果中发现的 500 到 1500。最后，搜索过程很容易陷入局部最优。GOA 易于实现的优势无助于摆脱局部最优，这可能是 GOA 的劣势。

2.4 带随机跳出机制的动态权重蝗虫优化算法

2.4.1 动态权重

GOA 使用线性递减参数来限制搜索空间并使所有搜索代理移动到目标位置。线性递减参数不能增强搜索过程的两个阶段的影响，即开发阶段和探索阶

段。在探索阶段，GOA 无法在目标搜索区域周围快速收敛，搜索代理只是在整个搜索空间中游荡，这无法为后期搜索阶段奠定坚实的基础。在开发阶段，参数通常使搜索代理滑过局部最佳位置，就像搜索代理超速运动一样。

线性递减参数机制无法使算法充分利用整个迭代。引入动态权重参数机制以提高算法的利用率。搜索进度分为三个阶段，即早期阶段，中间阶段阶段和后期阶段。在早期阶段，目标位置的权重应该更高，以使搜索过程快速收敛。在中间阶段，参数应该是稳定的，以使算法探索搜索空间。在后期阶段，搜索代理中的重力的权重应该更小，以深入利用局部最优解位置。具有动态权重参数的 GOA 公式表示为算法 [1]。新的迭代方程如下：

$$x_i = m * c\Big(\sum_{j=1,j\neq i}^{N} c\frac{u-l}{2}s(|x_j - x_i|)\frac{x_j - x_i}{d_{ij}}\Big) + \widehat{T}_d \tag{2.8}$$

where m is the dynamic weight parameter to adjust the search process. To correspond to the feature of the three phases of the search process, the parameter m is set as follows:

$$m = \begin{cases} 0.5 - \frac{(0.5-0.1)*iter}{MaxIteration*0.2} & 0 < iter \leq MaxIteration * 0.2 \\ 0.1 & MaxIteration * 0.2 < iter \leq MaxIteration * 0.8 \\ 0.05 & MaxIteration * 0.8 < iter \leq MaxIteration \end{cases} \tag{2.9}$$

2.4.2 随机跳出机制

原始的 GOA 算法没有使用跳跃机制，并且所有搜索代理仅根据社交互动和目标食物吸引力的影响而移动。原始 GOA 的机制可能导致算法陷入局部最优位置。

引入随机跳跃策略以帮助 GOA 算法提高跳出局部最佳位置的概率。参数 p 被设置为跳跃阈值。在迭代结束之前，将当前最佳适合度与最后一次迭代的最佳适应度进行比较。如果当前最佳适应度和最后一个最佳适应度的比率高于阈值 p，则可以假设该算法没有找到更好的解决方案，并且随机跳跃机制应该启动。根据随机初始化规则在最佳位置周围生成新的搜索代理。随机初始化规则如下：

$$tempPos = curPos * ((0.5 - rand) * iniRan + 1); \tag{2.10}$$

其中 *tempPos* 是新搜索代理的位置，*curPos* 是当前最佳解决方案的位置。*iniRan* 是管理随机跳跃边界的初始化范围参数。如果 *iniRan* 更高，则算法的全局搜索能力就像模拟退火算法一样得到增强。

2.4.3 DJGOA 算法流程

本文提出了一种随机跳跃动态权重蝗虫优化算法（DJGOA）。DJGOA 的过程分为初始阶段，参数设置阶段，计算阶段和适应度更新阶段四个阶段。在初始化阶段，设置包括位置边界在内的一些因素，并在边界内随机初始化原始群体位置。在参数设置阶段，搜索过程进入迭代循环，动态权重参数根据当前迭代由 emph equation（7）设置。在计算阶段，群体内的社会交互由 emph equation（6）计算。在适应度更新阶段，如果当前适应度优于历史的最佳目标适应度，则更新目标解决方案的适合度。如果当前适应度与最后一次迭代的适应度之比高于之前设置的阈值 emph p，则使用随机跳转策略生成新的搜索代理以尝试跳出局部最优位置通过 emph equation（8）。具有随机跳跃的动态权重蝗虫优化算法（DJGOA）的伪代码显示为算法2。

2.4.4 实验结果

2.4.4.1 实验设置

为了评估所提算法的性能，进行了一系列实验。我们将所提出的 DJGOA 与 GOA 的原始算法（最近的 Dragonfly 算法（DA）的元启发式算法）和粒子群算法（PSO）的经典启发式算法进行了比较，该算法通过在 cite saremi2017grasshopper 中使用的 13 个基准函数。13 个基准功能分为两种类型。函数 emph f1-f7 是单峰函数，它测试了算法的收敛速度和局部搜索能力。函数 emph f8-f13 是多模函数，当存在多个局部最优时，它测试算法的全局搜索能力。表2.1中列出了 7 个单峰测试函数的详细信息和表达式, 表2.2中列出了 6 个多峰测试函数的详细信息和表达式。

为了解决测试功能，采用了 30 个搜索代理，最大迭代次数设置为 500. 每个算法的每个实验进行 30 次以产生统计结果。GOA，DA 和 PSO 的参数设置为论文引用引用 mirjalili2016dragonfly 引用文件。 citemi2017grasshopper。

2.4.4.2 实验结果

表 [？] 中给出了 30 次实验的平均值，标准偏差，最佳值和最差值，以描述 DJGOA，GOA，DA 和 PSO 的性能。

从表 [？] 可以看出，DJGOA 的性能优于其他 3 种算法。所有 13 个基准函数的 10 个结果表明，DJGOA 的平均值比其他几个数量级的表现要好。而对于其他三个测试功能的结果，DJGOA 的表现略差于其他测试功能。实际上，DJGOA 可以通过 GOA，DA 和 PSO 获得相同数量级的结果。对于上面提到的三个测试功能

算法 2 带随机跳出机制的动态权重蝗虫优化算法

initialize the swarm and set the position boundaries u and l
initialize the factors including cmax, cmin, MaxIteration and iniRan
initialize all the search agents position with random origin matrix
calculate the target fitness and mark the target position
while $(iter < MaxIteration$ and $targetfitness > destinationfitness)$ **do**
    set the dynamic weight parameter m by equation(7)
    calculate d_ij and $\widehat{d_ij}$ by equation(2)
    calculate $s(d_ij)$ by equation(3)
    update x_i by equation(6)
    calculate the fitness
    if current fitness is better than target fitness **then**
        update the target fitness and the target position
    end if
    if $(currentfitness/lastfitness) > p$ **then**
        generate a new search agent by equation (8)
        if the new fitness is better than target fitness **then**
            update the target fitness and the target position
        end if
    end if
end while
Return target fitness and target position

表 2.1 F1-F7 单峰测试函数

Function	Dim	Range	f_{min}
$F_1(x) = \sum_{i=1}^{n} x_i^2$	30	[-100,100]	0
$F_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	[-10,10]	0
$F_3(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	30	[-100,100]	0
$F_4(x) = max_i\{\lvert x_i \rvert, 1 \le i \le n\}$	30	[-100,100]	0
$F_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	[-30,30]	0
$F_6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	30	[-100,100]	0
$F_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0,1)$	30	[-1.28,1.28]	0

中的两个，DJGOA 可以在获得最佳价值方面做得更好。实验结果表明，DJGOA 在标准偏差和最差值方面表现较好，表明 DJGOA 在搜索中更稳定。对于 12 个实验结果，DJGOA 可以获得更好的最佳值结果，这意味着 DJGOA 能够比其他 3 种算法更好地找到最佳适应度。

对于单峰测试功能，GOA 的性能非常好，DJGOA 可以大大提高 GOA 的性能。DJGOA 将搜索结果提高了几个数量级。对于多模式测试功能，DJGOA 可以帮助算法提高很多，特别是在寻找更准确的目标适应度时。总之，跳跃策略可以帮助算法进行大量的搜索，并且 DJGOA 在全局搜索和本地搜索中具有比 DA，GOA 和 PSO 更好的搜索最佳能力。

2.4.4.3 Wilcoxon Signed Rank Test

A experiment of wilcoxon signed rank test was conducted to verify the confidence level of the PSO, GOA and DJGOA. From the results shown in table[?], all the p values of DJGOA in the 13 test functions were smaller than 0.05, which meant that the conclusion obtained was significant.

2.4.4.4 Convergence Analysis

The results of the best fitness value along the iterations of DA, GOA and DJGOA for all the 13 benchmark functions are shown in Fig[?]. The convergence curves showed

表 2.2 F8-F13 多峰测试函数

Function	Dim	Range	f_{min}
$F_8(x)=\sum_{i=1}^{n}-x_i\sin(\sqrt{\lvert x_i\rvert})$	30	[-500,500]	$-418\times Dim$
$F_9(x)=\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)+10]$	30	[-5.12,5.12]	0
$F_{10}(x)=-20exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2})-$ $exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i))+20+e$	30	[-32,32]	0
$F_{11}(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	30	[-600,600]	0
$F_{12}(x)=\frac{\pi}{n}\{10\sin(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+$ $(y_n-1)^2\}+\sum_{i-1}^{n}u(x_i,10,100,4)+\sum_{i=1}^{n}u(x_i,10,100,4)$ $y_i=1+\frac{x_i+1}{4}$ $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m x_i>a\\0-a<x_i<a\\k(-x_i-a)^m x_i<-a\end{cases}$	30	[-50,50]	0
$F_{13}(x)=0.1\{\sin^2(3\pi x_1)+\sum_{i=1}^{n}(x_i-1)^2[1+\sin^2(3\pi x_i+1)]+$ $(x_n-1)^2[1+\sin^2(2\pi x_n)]\}+\sum_{i=1}^{n}u(x_x,5,100,4)$	30	[-50,50]	0

that DJGOA could make the search process converge more rapidly than others, and the results also demonstrated that the dynamic weight mechanism could help the original GOA algorithm to make full utilization of every iteration.

2.1。

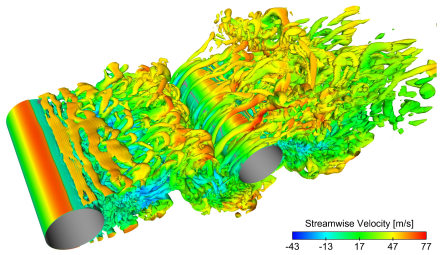

图 2.1 Q 判据等值面图，同时测试一下一个很长的标题，比如这真的是一个很长很长很长很长很长很长很长很长的标题。

Figure 2.1 Isocontour of Q criteria, at the same time, this is to test a long title, for instance, this is a really very long very long very long very long very long title.

2.5 Improved GOA(IGOA)

2.6 任务调度问题

2.6.1 问题描述

2.6.2 任务调度模型

2.6.3 实验结果

2.7 本章小结

附录 A　中国科学院大学学位论文撰写要求

学位论文是研究生科研工作成果的集中体现，是评判学位申请者学术水平、授予其学位的主要依据，是科研领域重要的文献资料。根据《科学技术报告、学位论文和学术论文的编写格式》（GB/T 7713-1987）、《学位论文编写规则》（GB/T 7713.1-2006）和《文后参考文献著录规则》（GB7714—87）等国家有关标准，结合中国科学院大学（以下简称“国科大”）的实际情况，特制订本规定。

A.1　论文无附录者无需附录部分

A.2　测试公式编号

$$\begin{cases} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{V}) = 0 \\ \frac{\partial (\rho \boldsymbol{V})}{\partial t} + \nabla \cdot (\rho \boldsymbol{V}\boldsymbol{V}) = \nabla \cdot \boldsymbol{\sigma} \\ \frac{\partial (\rho E)}{\partial t} + \nabla \cdot (\rho E \boldsymbol{V}) = \nabla \cdot (k \nabla T) + \nabla \cdot (\boldsymbol{\sigma} \cdot \boldsymbol{V}) \end{cases} \quad \ldots \text{(A.1)}$$

$$\frac{\partial}{\partial t} \int_{\Omega} u \, \mathrm{d}\Omega + \int_{S} \mathrm{n} \cdot (u \boldsymbol{V}) \, \mathrm{d}S = \dot{\phi} \quad \ldots \text{(A.2)}$$

A.3　测试生僻字

霜蟾盥薇

参考文献

[1] WIKIBOOK. http://en.wikibooks.org/wiki/latex[M]. On-line Resources, 2014.

[2] LAMPORT L. Document preparation system[M]. Addison-Wesley Reading, MA, 1986.

[3] 陈浩元. 著录文后参考文献的规则及注意事项[J]. 编辑学报, 2005, 17(6):413-415.

[4] 初景利. 图书馆数字参考咨询服务研究[M]. 北京: 北京图书馆出版社, 2004.

[5] STAMERJOHANNS H, GINEV D, DAVID C, et al. MathML-aware article conversion from LaTeX[J]. Towards a Digital Mathematics Library, 2009, 16(2):109-120.

[6] BETTS L R, TAYLOR C P. Aging reduces center-surround antagonism in visual motion processing[J]. Neuron, 2005, 45(3):361-366.

[7] BRAVO H, OLAVARRIA J. Comparative study of visual inter and intrahemispheric corticocortical connections in five native chilean rodents[J]. Anatomy and embryology, 1990, 181 (1):67-73.

[8] 哈里森·沃尔德伦. 经济数学与金融数学[M]. 谢远涛, 译. 北京: 中国人民大学出版社, 2012: 235-236.

[9] 牛志明, 斯温兰德, 雷光春. 综合湿地管理国际研讨会论文集[C]. 北京: 海洋出版社, 2013.

[10] 陈晋镳, 张惠民, 朱士兴, 等. 蓟县震旦亚界研究[M]//中国地质科学院天津地质矿产研究所. 中国震旦亚界. 天津: 天津科学技术出版社, 1980: 56-114.

[11] 袁训来, 陈哲, 肖书海. 蓝田生物群: 一个认识多细胞生物起源和早期演化的新窗口 – 篇一[J]. 科学通报, 2012, 57(34):3219.

[12] 袁训来, 陈哲, 肖书海. 蓝田生物群: 一个认识多细胞生物起源和早期演化的新窗口 – 篇二[J]. 科学通报, 2012, 57(34):3219.

[13] 袁训来, 陈哲, 肖书海. 蓝田生物群: 一个认识多细胞生物起源和早期演化的新窗口 – 篇三[J]. 科学通报, 2012, 57(34):3219.

[14] WALLS S C, BARICHIVICH W J, BROWN M E. Drought, deluge and declines: the impact of precipitation extremes on amphibians in a changing climate[J/OL]. Biology, 2013, 2(1):399-418[2013-11-04]. http://www.mdpi.com/2079-7737/2/1/399. DOI: 10.3390/biology2010399.

[15] ボハンデ. 過去及び現在に於ける英国と会[J]. 日本時報, 1928, 17:5-9.

[16] ДубровинА. И. Открытое письмо Председателя Главного Совета Союза Русского Народа Санкт-Петербургскому Антонию, Первенствующему члену Священного Синода [J]. Вече, 1906:1-3.

作者简历及攻读学位期间发表的学术论文与研究成果

本科生无需此部分。

作者简历

casthesis 作者

吴凌云，福建省屏南县人，中国科学院数学与系统科学研究院博士研究生。

ucasthesis 作者

莫晃锐，湖南省湘潭县人，中国科学院力学研究所硕士研究生。

已发表 (或正式接受) 的学术论文:

[1] ucasthesis: A LaTeX Thesis Template for the University of Chinese Academy of Sciences, 2014.

申请或已获得的专利:

(无专利时此项不必列出)

参加的研究项目及获奖情况:

可以随意添加新的条目或是结构。

致　谢

感激 casthesis 作者吴凌云学长，gbt7714-bibtex-style 开发者 zepinglee，和 ctex 众多开发者们。若没有他们的辛勤付出和非凡工作，LaTeX 菜鸟的我是无法完成此国科大学位论文 LaTeX 模板 ucasthesis 的。在 LaTeX 中的一点一滴的成长源于开源社区的众多优秀资料和教程，在此对所有 LaTeX 社区的贡献者表示感谢！

ucasthesis 国科大学位论文 LaTeX 模板的最终成型离不开以霍明虹老师和丁云云老师为代表的国科大学位办公室老师们制定的官方指导文件和众多 ucasthesis 用户的热心测试和耐心反馈，在此对他们的认真付出表示感谢。特别对国科大的赵永明同学的众多有效反馈意见和建议表示感谢，对国科大本科部的陆晴老师和本科部学位办的丁云云老师的细致审核和建议表示感谢。谢谢大家的共同努力和支持，让 ucasthesis 为国科大学子使用 LaTeX 撰写学位论文提供便利和高效这一目标成为可能。