软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2017,28(7):1898−1925 [doi: 10.13328/j.cnki.jos.005109]
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于信任度的合一系统社会任务分配*

武丹凤[1,2], 于思淼[1], 张绳昱[1], 张锐文[1]

[1](辽宁工程技术大学 软件学院,辽宁 葫芦岛 125105)

[2](北京科技大学 计算机与通信工程学院,北京 100083)

通讯作者: 武丹凤, E-mail: wudanfengby@163.com

摘 要: 在多智体社会网络中,传统的任务分配模型一般采用直接面向任务执行者的分配机制.它们不考虑社会网络组织结构对任务分配性能的巨大影响,也很少透彻地研究不可靠社会中的任务分配.针对这些问题,开创性地研究了软/硬件合一系统的任务分配,即按递阶、分层的思想设计了协作组织模型,并基于此提出了面向社区基于社会协调"软件人"的任务分配模型.模型研究过程中,提出了基于直接信任度和社区声誉的社区信任度评估机制、基于社区信任度和社区物理能力的节点选择机制、基于负载均衡的社区内任务分配机制和基于上下文资源的任务再分配策略.实验结果表明:与常见的直接面向任务执行者和基于资源的任务分配模型相比,所提出的模型具有更优的任务分配性能,且对社会任务环境变化具有更好的鲁棒性;社区内基于负载均衡的分配机制和基于上下文资源的再分配策略也有效提高了分配性能,降低了网络中的通信密度.

关键词: 社会网络;任务分配;组织模型;信任度;负载均衡;资源上下文

中图法分类号: TP316

中文引用格式: 武丹凤,于思淼,张绳昱,张锐文.基于信任度的合一系统社会任务分配.软件学报,2017,28(7):1898−1925. http://www.jos.org.cn/1000-9825/5109.htm

英文引用格式: Wu DF, Yu SM, Zhang SY, Zhang RW. Research on task allocation based trust degree for social network of syncretic system. Ruan Jian Xue Bao/Journal of Software, 2017,28(7):1898−1925 (in Chinese). http://www.jos.org.cn/1000-9825/5109.htm

Research on Task Allocation Based Trust Degree for Social Network of Syncretic System

WU Dan-Feng[1,2], YU Si-Miao[1], ZHANG Sheng-Yu[1], ZHANG Rui-Wen[1]

[1](School of Software, Liaoning Technical University, Huludao 125105, China)

[2](School of Computer & Communication Engineering, University of Science & Technology Beijing, Beijing 100083, China)

Abstract: In the multi-agent social network, the traditional task allocation models generally adopt allocation mechanism that directly orients to task performers. They do not consider the huge impact of social network structure on the performance of allocation, and seldom thoroughly research the allocation mechanism for the unreliable society. Aiming at these problems, this paper firstly designs a collaborative organization model according to the hierarchical and layering methodology, then proposes a community oriented task allocation model for software-hardware syncretic systems. In the process of model research, the paper develops a community trust evaluation mechanism based on direct trust and community reputation, a community node selection mechanism based on trust degree and physical ability of community, a load balancing mechanism which is applied to the task allocation in interior community, and a task redistribution strategy based on the context of community resources. The results of experiments show that the proposed model has better allocation performance and robustness compared with other classical models, and also validate that the load balancing allocation

* 基金项目: 国家自然科学基金(61404069, 61401185, 61540056); 辽宁省教育厅科学技术项目(LJYL052)

Foundation item: National Natural Science Foundation of China (61404069, 61401185, 61540056); Science and Technology Project of Education Hall of Liaoning Province, China (LJYL052)

收稿时间: 2016-01-06; 修改时间: 2016-03-03, 2016-04-26; 采用时间: 2016-05-31; jos 在线出版时间: 2016-10-11

CNKI 网络优先出版: 2016-10-12 16:26:57, http://www.cnki.net/kcms/detail/11.2560.TP.20161012.1626.024.html

© 中国科学院软件研究所 http://www.jos.org.cn

mechanism and the redistribution strategy can not only effectively improve the allocation performance but also reduce the communication density of the social network.

Key words: social network; task allocation; organization model; trust degree; load balancing; resources context

随着机器人技术向智能化、开放式、柔性化方向发展,机器人已变为了拥有大量先进传感器和执行器的复杂智能计算体[1,2],其功能的动态配置与在线重构是近年来的热点研究问题[3,4].2011 年,“软件人(SoftMan,简称 SM)”[5]课题组提出了“软件人”与机器人合一的思想,旨在为机器人与“软件人”所在的 PC 机之间构造一个对等、柔性、动态的协同模式,实现对机器人功能的实时更替,进一步提升机器人系统的柔性智能能力,改善其环境的适应协调能力.

“软件人”是对 Agent[6,7]的继承与发展,是从广义人工生命观点出发,为了延伸、扩展人的生命而提出来的[8].它生存于软件环境、活动于网络世界,是具有类似于人的活性(思维、感知、行为特性与信息处理、获取和利用功能)的软件人工生命[5].“软件人”借鉴了 Agent 研究领域的大量已有研究工作,它是在 Agent、数字生命、虚拟人、网络化身、游戏角色等概念综合集成的基础上产生的,它的主要科学技术基础如图 1 所示[5].

“软件人”与机器人合一一方面是指“软件人”与机器人控制系统的合一,即不同的“软件人”可通过迁移机制附着在机器人控制系统中,作为机器人的控制中心,使机器人具备不同的功能;另一方面是指“软件人”系统与机器人系统的合一,即被附着“软件人”后的机器人可与“软件人”系统中的“软件人”协同完成任务.随着“软件人”与机器人合一机制的技术研究与实现[9,10],“软件人”与机器人合一系统应运而生.合一系统是一种递阶、多级、协调、开放、松散耦合的分布式大系统,是多 Agent 系统的一个特殊实现形式,其逻辑层次结构如图 2 所示.

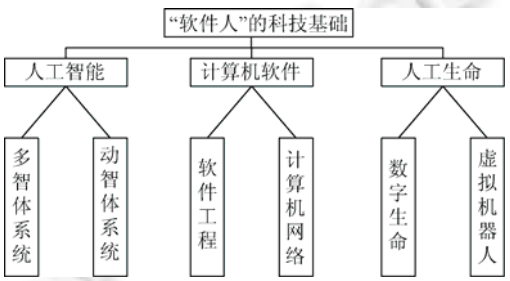


Fig.1 The science and technology foundation of SoftMan[5]

图 1 “软件人”的科技基础[5]

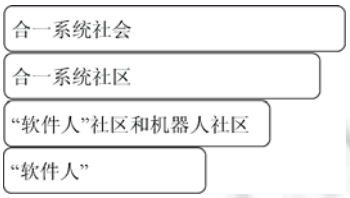


Fig.2 The logical hierarchy of syncretic system

图 2 合一系统逻辑层次结构

由图 2 可见,合一系统社会(syncretic system society,简称 SSS)由若干个合一系统社区(syncretic system community,简称 SSC)组成,而合一系统社区由一个“软件人”社区(SoftMan community,简称 SMC)和一个机器人社区组成(robot community,简称 RobC).合一系统社区内的“软件人”社区位于一个计算机节点中,其中包含多个承担不同功能角色的“软件人”;机器人社区则由 $n(n\geq 1)$个地理位置相近且拥有直接交互关系的机器人构成,每个机器人中也驻扎着承担不同功能角色的“软件人”.

在合一系统提出后,亟需研究其任务分配协调机理及协作模式.但机器人系统存在计算资源有限、实时性差的弱点,因此,如何根据机器人系统的特点以及“软件人”与机器人合一机制设计合一系统社会基于“软件人群”的组织模型,成为了研究合一系统社会任务分配首要解决的问题.在 SSS 中,每一个 ssc 都是任务入口,本文将任务 t 的入口社区称为任务 t 的管理方,将执行 t 的社区称为任务的承包方.在任务管理方寻求承包方的过程中,由于 SSS 的开放性和异质性,有些社区可能会采取不诚信的行为去追求自己的利益[11,12],而诚信社区则能提供自身真实的资源状态与执行任务能力的信息,并在获得任务后按约定完全贡献其可用资源执行任务[13].因此,在任务分配时,需要一种信任评估机制将任务分配给诚信且富有能力的社区,所以合一系统的信任度评估机制是本文研究的重要问题.此外,任务管理社区在分配任务时,既可以直接面向每个 ssc 推荐的“软件人”进行分配,也可以面向 ssc 进行分配.在已有的关于多 Agent 社会网络任务分配研究中,一般采用的是直接面向 Agent 的分配

© 中国科学院软件研究所 http://www.jos.org.cn

机制(与直接面向每个 ssc 推荐的“软件人”进行分配类似),而本文在 SSS 中将探索面向合一系统社区的任务分配.面向社区的分配将基于组织模型结构采用 2 级制的分配机制:第 1 级为面向任务投标社区的协调分配,第 2 级为面向社区内部处于社会最底层的“软件人”的协调分配.

本文的主要工作及成果陈述如下.

1) 按大系统广义模型化的递阶、分层思想[14,15]设计了 SSS 的任务协作组织模型.该模型可以方便地实现“软件人”群体的多级递阶和集中管理,同时也突出了多“软件人”之间主动交互、自由迁移、相互通信、协调合作等特性.模型的协调以本地社区内为先,最终起到全局协调的作用,避免了大量通信资源的消耗.

2) 提出了基于直接信任度和社区信誉的社区信任度评估机制.在本文中,信任度的计算与带有奖励、惩罚性质的满意度反馈机制息息相关,并兼顾了社区的历史表现与当前表现,提高了诚信社区获得任务的概率;设置了推荐偏差阈值,尽量避免采纳恶意社区的推荐信任度.

3) 提出了基于社区信任度的社区节点选择机制,扩展了已有的面向合作、诚信系统的任务分配模型.在对社区信任度进行评估的基础上,结合任务预计执行时间、通信距离、执行任务成功率和资源 4 个方面对社区节点进行了选择.

4) 提出了社区内部基于负载均衡的任务分配机制,减少了任务的等待时间.此外,该机制结合了“软件人”执行任务成功率进行负载均衡,提高了任务分配成功率.需要说明的是,结合了任务预计执行时间的社区节点选择机制与社区内部基于负载均衡的任务分配机制保证了整个社会网络面向社区的负载均衡和社区内部面向具体“软件人”的负载均衡.社区间的负载均衡由作为任务管理方的社区负责,具体借助预计执行时间平衡负载;“软件人”层面的负载均衡由任务承包社区控制.这两个层面的负载均衡有先后顺序,先是社区间的负载均衡,后是社区内“软件人”间的负载均衡.

5) 提出了社区内基于上下文资源的任务再分配机制.任务失败后,任务承包社区的管理“软件人”可自主地在社区资源允许的情况下在社区内部进行再分配,减少了任务管理社区的招标次数,降低了通信密度,节省了通信时间,使得任务分配更加具有柔性,优化了分配性能.

6) 通过广泛实验,将提出的任务分配模型与其他经典模型进行了分配性能对比和鲁棒性对比,并通过单独实验验证了模型中提出的信任度评估机制、负载均衡分配机制和基于社区上下文资源的再分配机制在提高分配性能方面的有效性.

本文第 1 节给出相关工作.第 2 节构造 SSS 的协作组织模型,描述基于协作组织模型的任务协调分配过程.第 3 节对 SSS 任务分配的相关问题进行定义,研究合一系统社会的任务分配模型,模型主要包括 4 个部分,即社区信任度评估机制、基于社区诚信度和能力的社区节点选择机制、社区内部基于负载均衡的“软件人”选择机制和基于社区上下文资源的任务再分配机制.第 4 节进行相关实验验证与对比分析.最后,对本文工作进行总结,提出下一步的研究方向.

1 相关工作

1.1 不可靠多 Agent 系统中的信任和信誉系统研究

在不可靠多 Agent 系统(multi-agent systems,简称 MASs)中,自私的 Agent 可能会为了自身利益采取策略性的不诚信行为,如谎报资源状态信息、破坏合同等;而恶意的 Agent 会毫无理智地破坏合作、危害网络.在本文中,我们将这两者统称为不诚信的 Agent.不诚信 Agent 的自私或恶意行为,会导致合作的任务失败,系统的性能也将大为降低.为解决此类问题,任务分配模型往往需要结合信任评估机制,以尽量保证任务分配的可靠性.信任评估是需要付出代价的,但是它可以促进可靠的分配协商,通过有限的代价达到任务分配目标[16].

与人类社会类似,信任通常来源于两种途径:一种是直接信任,是评价 Agent 依靠自己的知识和以往交互中获得的直接经验判断得出的信任,反映评价 Agent 的自身观点;另一种称为信誉,是通过收集评价 Agent 所处的社会中其他 Agent 对目标 Agent 的观点,经推理得出的信任[17,18].现在,人们往往利用信誉系统有效地综合直接信任与信誉,来完成信任评价.

© 中国科学院软件研究所 http://www.jos.org.cn

在开放的、复杂的、不可靠的多 Agent 系统中,信誉系统体系结构、组织模型、评估对象的选取、评估粒度的粗细、信息是否过滤、信息时效性处理影响着信誉系统的性能[19].针对这些影响因素,领域专家们做了很多研究工作,提出了不同的声誉系统模型.下面,我们将首先对这些因素进行简要介绍,然后将较为成功的信誉系统模型(如 SPORAS[20]、ReGreT[21]和 FIRE[17]模型)与近 5 年提出的信誉系统模型进行不同因素下的特征归类(见表 1),以分析多 Agent 系统信任和信誉系统的研究现状,指引我们在合一系统信任度评估方面应开展的研究工作.

在表 1 中,"?"表示在相关模型中获取不到相关确切信息,"–"表示模型所对应的任务分配环境不需要进行相关方面内容的研究.下面,对表 1 中每列分类作一个具体解释.

控制方式.信誉系统的控制方式主要分为集中式和分布式两种.集中式信誉系统设有一个信誉中心,负责收集与汇总所有的交互评定信息,对评定简单地进行存储或作简单计算供 Agent 将来查询,单个 Agent 并不存储信任相关信息.分布式信誉系统每次交互完成后,Agent 各自保存自己的评定,在与目标 Agent 合作之前,评价 Agent 通过询问其他 Agent 得到目标 Agent 的信誉,依靠存储的直接信任和获得的信誉信息,对目标 Agent 进行评价[18].

组织.组织描述信誉系统中的节点是如何联系在一起的.组织可以分为结构化的和非结构化的两种.在结构化的组织中,当新节点进入网络时,会设计其与其他节点的组织关系;在非结构化的组织中,网络不存在任何组织结构与关系设计,新节点进入网络时随机地与其他节点建立联系.

对象.Mui 等人[22]提出信誉系统的评价主体和目标对象可以是原子个体,也可以是团体/组合.

评估粒度.在评价某一目标对象时,信誉系统可按应用需求针对目标对象的某一方面(如原子功能或种类资源)进行评价,即细粒度评价;也可结合目标对象的各个方面进行综合评价.在综合评价时,不同评价因素设置不同的权值,最终形成综合评价结果.

信息过滤.评价 Agent 和目标 Agent 在交互前,由第 3 方提供的报告值与交互后得到的观察值之间往往存在差异,造成信息不精确的问题,因此需对不精确信息进行过滤.若不进行信息过滤,则不进行信息偏差统计,不判断信息提供者是否诚信,对信誉信息提供者的信息全部利用;若进行信息过滤,则会通过统计等手段检测和排除不诚信者提供的信息.

信息时效.信息时效利用时间的流逝降低对已有信任信息的可信性,该机制适用于 Agent 诚信度或能力变化的动态系统.最新的直接信任信息或信誉信息一般会赋予较大的权重,目标对象最近的负面行为会极大地影响评价 Agent 的决策.若不考虑信息时效,则已有的直接信任度信息或信誉信息会无限期地保留下来;若考虑时效,则随着时间流逝,信息拥有者会降低已有信息的可信性,或根据时间设定,消除已有历史信息.

Tabel 1 Summary of reputation system

表 1 信誉系统总结

信誉系统	控制方式 (C-集中式, D-分布式)	组织模型 (S-结构化,U-非结构化)	对象 (I-原子个体,G-团体)	评估粒度 (A-原子资源种类或能力,H-综合)	信息过滤 (N-无信息过滤研究,Y-有信息过滤相关研究)	信息时效 (N-不进行信息折旧或消除;D-进行信息可信性的衰减或消除)
SPORAS[20]	C	?	I	H	N	D
Amazon[23]	C	?	I	H	N	N
Taobao[24]	C	?	I	H	N	N
ReGreT[21]	D	U	I	H	N	D
FIRE[17]	D	U	I	?	Y	N
TRM[25]	D	U	I	?	N	N
NR1[13]	D	U	I	H	–	–
NR2[26]	D	U	I	A	–	–
CRM[27]	D	U	I	H&A	N	N
Model by Basheer, *et al*.[28]	D	U	I	?	N	N
TMS[29]	D	U	I	?	N	N

下面,我们根据表 1 来分析现有声誉系统模型的相关工作,并指出本文的任务分配模型在信任度评估部分拟进行的研究.

© 中国科学院软件研究所 http://www.jos.org.cn

1) 缺乏对信誉系统控制结构的改进研究.电子商务领域中,多使用集中式信誉系统,如淘宝、亚马逊等;学术研究领域中,对分布式信誉系统研究居多.集中式方法的优点是实现简单,评价失败的风险小,即使新加入的Agent也能从信誉中心获得评定信息.不足之处在于:① 不能区分直接信任和信誉,而一般情况下前者比后者无论从可靠性还是针对性方面都强得多;② 统一的汇总方法不利于评价 Agent 个性化地利用信誉信息;③ 不能快速适应环境的动态变化.分布式信誉系统充分发挥了智能 Agent 自治、灵活的优势,但面临的问题也很突出,如:① 找不到信誉推荐者会造成评价失败;② 推荐者视角不同会产生有偏差的评定,进而影响对目标 Agent 的综合评定;③ 推荐者会为私利而撒谎或不提供信息等[18].两种方式各有利弊,因此,我们拟探求一种混合、开放的多层级综合控制结构,但针对该类结构信誉系统的研究鲜少.

2) 缺乏对信任度评估所依靠的组织的研究,普遍基于非结构化的网络.在本文中,我们拟借鉴人类社会特征来组织系统,为每个"软件人"个体分派不同的角色和责任,并设计角色间的关系.结构化的组织模型可为信誉系统的对象之间提供一种协调机制,提高了系统的运行效率.

3) 评价主体和目标对象一般为 Agent 个体,且以对目标对象的综合评估为主,不能很好地解决同一节点在不同领域、不同方面的可信度问题.在合一系统中,社会是由合一系统社区构成的,因此本文将以社区作为信誉系统中的对象.此外,现实情况下,每个社区都有自己擅长执行的任务类型,如果基于全部任务类型统计其直接信任度和推荐信任度,有可能得出的社区信任度很高,但其并不适合执行某类任务.因此,本文将细化评估粒度,设计一个社区基于不同功能类型对另一个社区进行信任度评估.

4) 在进行声誉信息推荐后,对于推荐信息进行过滤对保证声誉系统性能具有非常重要的意义.但在所列研究工作中,只有 FIRE 模型设计评价 Agent 将每次得到的评定值与实际观察值的差值记录下来作为自己的直接经验,以判断信息提供者的可信性,解决评价信息不精确问题.我们将借鉴该统计方法,由评价社区单独统计推荐社区的推荐度偏差,并设置一个推荐偏差阈值,将超过该阈值的社区所推荐的信息过滤.此外,设计竞争同一任务的社区互相之间不进行推荐,避免因竞争带来的策略型不诚信推荐.

5) 在开放、动态的多 Agent 系统中,信息时效处理机制是保证信息即时性、精确性的重要手段.在表 1 中,除SPORAS和ReGreT信誉系统模型外,其他信誉系统没有考虑信息时效问题.本文将在直接信任度历史统计信息和当前服务质量间设置权重,并设计社区不保留其他社区的信誉信息,而是在任务实时分配时,实时收集目标社区的信誉信息.在对新收集的信誉信息进行过滤后,结合存储的直接信任度数据,计算目标社区的信任度.

1.2 多智体社会网络中的任务分配

在大规模多智体系统或社会网络中,领域内研究者通常设计每个智体在网络中只能与它的邻居进行交互,并基于网络结构模型和智体间的社会关系,研究基于邻居的任务分配模型.如 Weerdt 等人[11]在研究多 Agent 社会网络中的任务分配问题时,设定 Agent 只能向地理位置相近、有直接连接关系的邻居请求资源,但没有考虑将邻居作为中介向社会中更远的 Agent 寻求协作.这种分配模型使网络社会中的资源得不到充分利用,且限制了任务分配的成功率.Jiang 和 Li[30]提出的任务分配算法是通过邻居间信息传递、在对整个网络拓扑掌握后进行的 Agent 选取,但此种算法并不符合大规模系统的实际应用;Jiang 和 Li 在文献[31]中还提出了一种基于邻居物理情境和社会情境的任务分配方法,但却没有考虑 Agent 的历史表现和能力;Hunt 等人[32]提出在分配每一个任务时,Agent 之间需要交换各自的信息,最后达到一致性的分配结果,但这种机制只适合于合作、可靠的社会网络.由此可见,完全分布式的基于邻居的任务分配模型使每个 Agent 要不只在自己的视野范围或通信范围分配任务[33],任务分配达不到最优;要不通过邻居间的交互获取网络拓扑和资源信息,使得网络中的横向通信密集,也不适合动态的拓扑变化环境.此外,基于邻居的任务分配模型并不具有通用性,网络拓扑很大程度上决定了基于邻居的任务分配模型的性能.

针对上述基于固有网络结构和社会关系而建立的基于邻居的任务分配模型的缺点,一些学者提出通过调整网络结构或 Agent 间的社会关系以提高任务分配性能.在一些文献中提出可设计一个协调者(或中介者)扩大 Agent 的交互范围,分散通信密度.如 Abdallah 和 Lesser[34]设计了一个拥有中介者的网络结构,中介者与其他 Agent 相连,并且是社区任务的接收者,它负责分解任务,并做出任务分配决策;Xu 等人[35]在分布式的框架中使

© 中国科学院软件研究所 http://www.jos.org.cn

用了“黑板”协作结构,使 Agent 间可以互相交流信息,“黑板”不光负责收集和处理信息,还是任务分配的协调者和决策者.此类任务分配方法虽然扩展了每个 Agent 的协商范围,但是,无论是中介者,还是“黑板”,工作负荷都很大,当它们出现故障时,系统将停止运转.Kota 等人[36]提出了一种分布式的任务分配方法,该方法可以通过修改 Agent 间的社会关系来实现更好的任务分配,但是他们的工作没有涉及不可靠的社会网络.而本文的研究考虑了实体的社会性(诚信的或欺诈的),并通过社区之间的信任度动态地加以调整,结合社区物理能力完成任务分配.Val 等人[37]通过比较邻居与熟人产生的收益,决定何时选择最适宜的熟人替代已有的某个邻居,但是他们没有考虑 Agent 的能力.Wang 和 Jiang[38]设计了一种 Agent 迁移机制,即将 Agent 从一个节点迁移到另一个节点集中起来建立联系以完成一个任务.这种任务分配机制方便了 Agent 间的协商,减少了社会通信次数,但却没有考虑迁移 Agent 受目标节点资源的限制,且会对目标节点本地任务的接收与执行产生影响,也没有考虑迁移过程中可能带来的错误,如迁移不完整等.

在合一系统网络社会中也设有一个中介者,称其为社会协调“软件人”.社会协调“软件人”只是单纯的协调,任务的分配决策是由各个 ssc 做出的;任务的入口是各个社区,而不像有的相关研究中将中介者作为任务接收者.这种集中加分散的结构,增强了系统的鲁棒性,减少了横向通信密度,增加了任务分配成功率.本文将在第 2 节具体阐述合一系统社会的组织结构模型,且当网络拓扑发生改变时,此种组织模型能够迅速获取变化,具有很强的适应性.此外,在不可靠的网络社会中,虽然邻居的距离最近,但是它有可能是不诚信的,因此,本文设计的任务分配机制是通过协调“软件人”发出标书,在 SSS 中寻找诚信且能力评估最优的社区.

2 合一系统社会“软件人”协作组织模型

在合一系统社会中,合一系统社会通过网络关系组织社区,形成的组织结构直接影响社区间的协作效率.因此,研究 SSS 的任务分配之前,首先要做的就是设计合理的协作组织模型结构.在本节中,我们首先设计了合一系统社会中基于“软件人”群的任务协作组织模型,然后描述了基于协作组织模型的任务协调分配过程.

2.1 “软件人”协作组织模型设计

“软件人”群的组织管理是多“软件人”合作、协调的基本问题,合理的组织结构和分工会减少冲突,有利于系统行为的协调.本文将按照大系统广义模型化的递阶、分层思想[14,15]设计合一系统社会的协作组织模型,如图 3 所示.

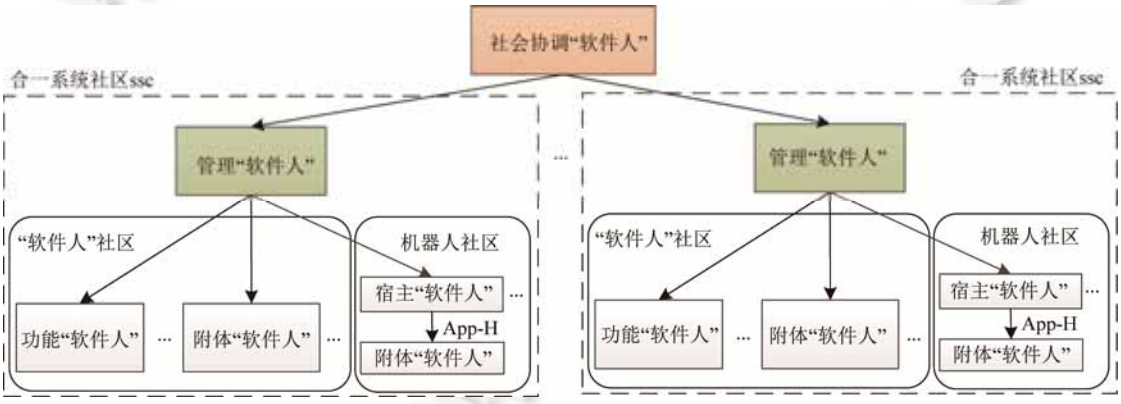


Fig.3 The structure diagram of collaborative organization model of SSS

图 3 合一系统社会(SSS)“软件人”协作组织模型结构图

下面,分别对协作组织模型中的不同“软件人”个体所承担的社会角色作一介绍.

● 社会协调“软件人”(**SoftMan for social coordination**,简称 **SM.coor**). 社会协调“软件人”负责汇总全社会的资源数据和运行状态,各社区管理“软件人”有义务向它汇报相关数据,它是社会中各节点社区间的协调联络

© 中国科学院软件研究所 http://www.jos.org.cn

员,但对各社区没有绝对控制权.

● 管理“软件人”(**SoftMan for management**,简称 **SM**.**man**). 管理“软件人”是合一系统社区 ssc 的总管,一个管理“软件人”代表了一个 ssc.它负责组织关系的建立与维护、社区任务的分配、社区活动的决策、多“软件人”任务协同和各社区间失误协调参与.管理“软件人”的物理位置在“软件人”社区 PC 节点中.

● 功能“软件人”(**SoftMan for executing function**,简称 **SM**.**fun**). 功能“软件人”是完成某些任务的“专业”员工,它是任务逻辑的承载者,同时也是用户请求的最终执行者.这类“软件人”可以在社会网络中“游走”,到达指定的目的地(主机)去工作.在“软件人”社区中,一个功能“软件人”一般是一个功能程序,它所具有的功能类别用 fid 表示,以对应执行 SSS 中社会功能需求代码为 fid 的任务.

● 宿主“软件人”(**SoftMan for host**,简称 **SM**.**host**). 宿主“软件人”不仅是机器人系统的管理守护中心、机器人与 PC 端“软件人”社区交互的枢纽,同时也负责接收附体“软件人”,为其搭建运行支撑环境.具体职责包括:机器人系统初始化、消息通信、附体“软件人”的接收、附体“软件人”的控制、系统容错控制和上下文环境资源监控.需要指出的是,由于宿主“软件人”所在的机器人节点计算资源有限,我们将其决策行为托管给“软件人”社区中的管理“软件人”.因此,“软件人”社区和机器人社区构成的合一系统社区的决策者是管理“软件人”.详细了解宿主“软件人”可参见文献[9].

● 附体“软件人”(**SoftMan for appendage**,简称 **SM**.**app**). 附体“软件人”本质上是可以迁移的特殊功能“软件人”,它是机器人系统的功能控制中心,通过对附体“软件人”的在线迁移与动态替换,可实现机器人功能的实时更替.附体“软件人”的迁移方式有两种:一种是完全迁移,即该附体“软件人”从“软件人”社区 A 中迁移到目标机器人 B,当在 B 中完成子任务后,再返回到“软件人”社区;另一种是复制迁移,即在“软件人”社区 PC 端复制生成和该附体“软件人”完全相同的克隆体,把它们派发到不同的机器人中并行工作,执行完毕后在目标机器人中结束生命.在我们的研究中,选择复制迁移的迁移方式.详细了解附体“软件人”可参见文献[10].

一个机器人可以具备多种能力(功能),每一种能力需要“软件人”社区中对应的附体“软件人”迁移其上才可实现.对此,本文提出假设 1.

假设 1. 在合一系统社区中,具有相同功能类型的不同附体“软件人”个数大于等于 1,但机器人的每一种能力只能由一个附体“软件人”去控制实现它,一个附体“软件人”可对应多个($n\geq 1$)具有该功能的机器人.

为将“软件人”与机器人合一系统的任务分配协调统一到“软件人”群个体之间,在此,拟引入一个在“软件人”与机器人合一后产生的特殊功能“软件人”——App-H.

定义 1(**App-H**). 作为机器人的管理守护中心,每一个 SM.host 可代表一个机器人,当某一 SM.app 与 SM.host 有迁移接收关系时,表示接收该 SM.app 后的机器人可以作为用户请求的最终执行者,具有执行某一任务的能力,因此,代表一个 SM.app 与一个 SM.host 合一的 App-H 可以看作是具有某一功能的特殊功能“软件人”.

作为机器人系统的管理中心,每一个 SM.host 代表一个机器人,一个 SM.app 对应一个机器人等于一个 SM.app 对应一个 SM.host.根据假设 1,机器人的每一种功能有一个 SM.app 去控制实现,该机器人具有唯一性,SM.app 也具有唯一性,因此,每一个 App-H 在 ssc 中具有唯一性.

定义 2(**管理类“软件人”和非管理类“软件人”**). 在协作组织模型中,按照业务功能的划分,“软件人(SM)”被定制为功能 SM“SM.fun”、附体-宿主 SM“App-H”、社区管理 SM“SM.man”和社会协调 SM“SM.coor”4 种类型,其中,SM.fun 和 App-H 属于非管理类“软件人”,其他 SM 属于管理类“软件人”.

合一系统社会协作组织模型既有各个分散社区管理“软件人”直接的、及时的局部控制,又有社会协调“软件人”集中的、间接的、全局的协调,兼有集中协调和分散协调的优点.社会结构和递阶关系可使任务协调相对简化,协调控制与观测信息量相对较小.此外,合一系统“软件人”协作组织模型还具有以下特点.

1) 根据机器人系统计算资源有限、实时性差的特点设计了管理“软件人”和宿主“软件人”角色功能,将机器人社区的管理中心——宿主“软件人”的决策行为托管给位于 PC 机上的管理“软件人”,此种设计也能规避迁移协商阶段“软件人”社区和机器人社区间频繁的通信.

2) 随着机器人控制系统中宿主“软件人”和附体“软件人”的引入,“软件人”与机器人的任务协作转变为功

© 中国科学院软件研究所 http://www.jos.org.cn

能“软件人”与 App-H 之间的协作,使软/硬件无缝地结合.

3) 各类“软件人”各司其职,增强了协作可靠性.

4) 模型结构更加类似于现实社会结构,社会协调“软件人”只是协调联络员,没有绝对控制权,每个社区在管理“软件人”的集中控制下有充分的自治性.在任务分配时,以本地社区为先,避免了通信资源的浪费,且可实现远距离的合作,扩展了物理空间.

2.2 基于协作组织模型的任务分配过程

合同网[39,40]是一种重要的协同问题求解模型,在多 Agent 系统的任务协调中有着广泛的应用.因此,本文将基于协作组织模型结构,借助合同网协议的协商过程,研究合一系统社会中面向社区的任务分配过程.分配过程描述如下.

1) 用户产生任务需求,就近将本地 ssc 作为请求入口;

2) 社区专门负责任务分解的功能“软件人”根据社会协调“软件人”提供的社会功能信息表对任务进行分解(由于本文关注的是任务分配问题,因此任务分解不作具体研究),分解后得到的每个子任务对应于一类具有相同功能的 SM.fun 或 App-H;

3) 管理“软件人”过滤出本地能完成相应任务的 SM.fun 或 App-H,获取与社区内部任务分配决策有关的这些“软件人”的信息,根据社区内的任务分配策略做出任务分配决策;

4) 若本地社区不能完成某些任务,则管理“软件人”向社会协调“软件人”发出招标书;

5) 社会协调“软件人”向具有该任务需求功能的社区发送带有招标方地址信息的招标书;

6) 有合作意愿的社区的管理“软件人”按照招标书中的地址在投标截止时间前向招标社区提供投标书;若无投标社区,则转 9);

7) 招标社区的管理“软件人”根据面向社区的任务分配模型确定并通知中标社区,中标社区确认后,招标社区的管理“软件人”完成任务分配过程;若此时投标社区不进行及时确认,或撤销了投标书,则招标社区的管理“软件人”选择次优的社区发出中标通知,依此类推;若中标社区都不进行确认,则转 9);

8) 确认后的中标社区执行 3),完成社区内面向非管理类“软件人”的任务分配;

9) 若任务分配失败,则任务入口社区将失败信息返还给用户.

由上可见,合一系统社会的任务协调以本地社区为先,最终起到全局协调的作用,避免了大量通信资源的消耗.合一系统社会组织模型结构中的社会协调“软件人”既不集中控制、做出分配决策,也不接收任务,只是向任务管理方提供全社会的功能信息,并作为任务招标的中介,辅助任务管理方扩大其任务分配的协商范围.鉴于社会协调“软件人”在任务分配过程中扮演的重要角色,我们可以设置一个与运行中的社会协调“软件人”信息同步更新的备用协调“软件人”,当运行的协调“软件人”发生故障失效时,可立即代替其服务社会.此外,合一系统社会并不能准确地归入“大”或“小”的分类之中,因为在其实际运行过程中,经常会有些社区节点加入或离开,因此,社会的规模是动态变化的.如社会规模变大使得协调“软件人”工作负荷沉重,或没有备用协调“软件人”的情况下现有协调“软件人”失效,合一系统社会各社区可以借助其所在的物理局域网络和社会关系网络[33]与其他社区进行完全分布式的任务分配协商.

通过分配过程的描述,也可得出在任务协调分配过程中,社区节点的选择和社区内部的任务分配策略是高效率完成任务分配、降低任务执行时间、提高任务执行成功率的关键所在,也是本文研究的重点、难点.因此,本文将在第 3 节对此进行深入研究.

在合一系统社会任务分配过程中,招标社区和投标社区之间的协商对任务分配的成败起着关键性作用.招标社区管理“软件人”的目标是在投标的社区中选出能够高质量完成任务的社区;而投标社区管理“软件人”的目标则是在负载条件允许的情况下尽可能多地获得新任务.由于本文研究的是任务分配模型,因此将重点放在了招标社区的分配策略上.在此,我们只简要介绍投标社区在任务分配过程中与招标社区的协商算法.

对于任务的投标社区来说,其协商算法如下.

1) 投标社区管理“软件人”SM.man 收到社会协调“软件人”发送的管理社区 ssc_i 发布的关于任务 t 的招

© 中国科学院软件研究所 http://www.jos.org.cn

标书;

2) SM.man 根据招标书中 t 的功能需求代码查询本社区功能信息表,匹配出可以完成该项任务的本地 SM.fun 或 App-H.若有相应的本地 SM.fun 或 App-H,则结合 t 的起止时间、资源需求等约束条件,以及"软件人"社区或机器人的系统资源状态、SM.fun 或 App-H 的队列大小和处理速率,估算出完成 t 的预计执行时间;

3) 若可在任务截止时间内完成 t,则将社区执行 t 的预计执行时间、相关资源信息、执行此类型任务的成功率等写入投标标书,按招标书中的任务管理社区地址将投标书在投标截止时间内返回;

4) 投标社区的 SM.man 如收到任务管理社区发送的合同,首先核实此时的系统资源状态是否可在任务截止时间内满足任务需求,若满足,则发送合同确认消息给任务管理社区,否则,发送取消投标消息.

若不诚信的社区参与投标,为提高中标率或故意危害社会,社区的管理"软件人"可能会将任务预计执行时间、执行此类任务的成功率、社区资源等信息造假,而上述投标协商算法描述的是诚信社区的投标过程.

3 任务分配模型

当任务以社区为入口进入合一系统社会时,入口社区作为任务管理方首先需要将其分配给能执行该任务的承包方(若任务管理方可以执行该任务,则其既是管理方,又是承包方);任务分配到承包社区后,社区的管理"软件人"再将其分配给具体的 SM.fun 或 App-H 执行.本文将此二级分配机制称为合一系统社会中面向社区的任务分配.

3.1 SSS 中的任务与不诚信社区

在不可靠的合一系统社会中,同样包含自私、恶意和诚信 3 种社区.如前文所述,自私的社区可能会为了自身利益采取策略性的自私行为,如谎报资源状态信息、破坏合同等;而恶意的社区会毫无理智地破坏合作、危害网络.在本文中,我们把这两者统称为不诚信的社区.本文研究不可靠 SSS 中的任务分配问题,因此,本节将对 SSS 中的任务进行规范化定义,对不诚信社区的不诚信行为进行具体描述.

定义 3(SSS 中的任务). SSS 中的任务是由合一系统社区中负责分解任务的功能"软件人"根据社会功能信息列表将原始入口任务进行分解后得到的.每一个分解后的任务 t 可以描述为一个五元组:

$$t=\{id, fid, td, \boldsymbol{q}, dl\},$$

其中,id 表示任务标识符,对任务起唯一标识作用;fid 表示任务的社会功能需求代码,每个任务对应一种社会功能 fid;td 表示任务的文本描述;$\boldsymbol{q}$ 表示任务所需的各项资源集合,$q=\{q_1,...,q_r,...,q_M\}$,$M$ 表示 SSS 中所有社区的资源种类数,q_r 表示任务对第 r 种资源的需求量;dl 表示完成任务的截止时间.

假设 2. 为研究方便,本文在研究任务分配模型时,假设任务间具有并行关系.

接下来,我们将对不诚信社区的不诚信行为进行具体描述.

若社区具有如下任一行为,则这个社区是不诚信的.

1) 在任务分配过程中,投标社区中的管理"软件人"伪造资源状态信息、虚报类型任务执行成功率或任务预计执行时间.

当社区 ssc_i 在投标任务 t 时,社区中的管理"软件人"为提高竞争力,将社区所拥有的一种或多种 t 所需种类资源信息、t 对应 fid 类型任务执行成功率或 t 的预计执行时间策略性修改,汇报给招标社区,则 ssc_i 是不诚信社区.

2) 中标确认后不执行任务或执行中途擅自终止任务.

设社区 ssc_i 已中标确认执行任务 t,完成 t 的预期收益为 $v(t)$.在执行 t 前或执行 t 的过程中,有正在招标的任务 t',且 t'的预期收益大于 $v(t)$.但此时 ssc_i 的剩余资源并不满足 t'的投标需求,如其选择了不执行任务 t,或伪造资源信息投标获得 t'后终止执行 t,则 ssc_i 为不诚信社区.

3) 任务执行失败时不调用社区内空闲资源重新分配任务.

当社区 ssc_i 内的 SM.fun 或 App-H 执行 t 失败时,ssc_i 可调用其空闲资源在任务截止时间到来前完成 t.但 ssc_i

© 中国科学院软件研究所 http://www.jos.org.cn

选择放弃在社区内部对 t 的重新分配,转而向 t 的任务管理社区汇报任务执行失败,则 ssc_i 为不诚信社区.

4) 伪造推荐信任度信息.

社区 ssc_i 在与社区 ssc_j 的合作过程中,会针对每一类型任务形成对 ssc_j 的直接信任度评价信息.如社区 ssc_k 正在进行任务 t 的协商分配,向 ssc_i 征求关于社区 ssc_j 在执行 t 所对应的 *fid* 类型任务方面的信任度信息,但 ssc_i 向 ssc_k 汇报的关于 ssc_j 的推荐信任度不等于其对 ssc_j 的直接信任度,则社区 ssc_i 为不诚信社区.

3.2 社区信任度评估

在合一系统社会中,对信任的定义是:对一个合一系统社区行为的可信度评估.对一个社区节点的信任与这个节点的能力有关,还与以往的交互历史有关,并结合对社区近期行为的重新评估结果进行不断的修正.

本文根据两个因素计算社区信任度:直接信任(direct trust,简称 DT)和社区信誉(reputation,简称 REP).在不可靠的合一系统社会中,直接信任是指一个社区对另一个社区完成某一任务的直接相信程度;社区信誉和推荐信任(recommendation trust,简称 RT)直接关联,任务招标社区综合非投标社区的推荐信任度,得出每一个投标社区的社区信誉.

假设 3. 社区对于以自身为入口的任务都是诚信的.

根据假设 3,本文的社区信任度评估是任务管理方在分配任务时对自身之外的投标社区的评估.

本小节将具体研究任务管理方 ssc_i(招标社区)对投标社区 ssc_j 的社区信任度 $CT_{ij}, i\neq j$.下面,给出与计算社区信任度相关的一些定义和公式.

- ***DoS*$_{ij}$(*t*)**:满意度(**degree of satisfaction**,简称 **DoS**).

即任务承包社区 ssc_j 在完成任务管理社区 ssc_i 的任务 t(t 的功能需求代码为 *fid*)后,ssc_i 对 ssc_j 的满意度评价,$DoS_{ij}(t)\leq 1$.

在实时、动态的合一系统任务环境下,满意度主要与任务执行时间有关.因此,本文设计任务管理社区对任务承包社区的满意度与任务完成截止时间 *dl*、任务承包社区投标时给出的预计执行时间(expected execution time,简称 EET)和中标接收任务后的实际执行时间(actual execution time,简称 AET)有关,如公式(1)所示.

$$DoS_{ij}(t)=\begin{cases}1, & EET(t)\geq AET(t)\\ 1-\dfrac{AET(t)-EET(t)}{dl(t)}, & EET(t)<AET(t)\wedge AET(t)\leq dl(t)\\ -\xi, & \text{failed}\end{cases} \tag{1}$$

若任务承包方在其竞标时向任务管理方汇报的预计执行时间内完成任务,则任务管理方对其的满意度反馈为 1;超出预计执行时间越多,满意度越低.这种设计可抑制投标社区为了中标在投标时故意压减预计执行时间的不诚信行为;若任务执行失败,或未在任务截止时间 *dl*(t)内完成,则满意度为$-\xi$,$\xi\geq 0$.一般情况下,将ξ值取为 5,因为如果一个社区执行某个任务失败,或者意味着该社区是不诚信的,或者意味着该社区暂时失去了完成该类型任务的能力,因此满意度惩罚远远大于奖励.

- ***CQoS*$_{ij}$(*fid*)**:当前服务质量(**the current quality of service**,简称 **CQoS**).

即在最近一次进行 *fid* 类任务合作后,任务管理社区 ssc_i 对任务承包社区 ssc_j 执行关于 *fid* 类型任务的平均满意度评价,$CQoS_{ij}(fid)\leq 1$.$CQoS_{ij}(fid)$ 的计算如公式(2)所示.

$$CQoS_{ij}(fid)=\frac{\sum_{t\in T_{ij}(fid)} DoS_{ij}(t)}{\left|T_{ij}(fid)\right|} \tag{2}$$

其中,$T_{ij}(fid)$ 表示 ssc_i 分配给 ssc_j 的任务功能需求代码为 *fid* 的任务集合,$\left|T_{ij}(fid)\right|$ 表示任务个数.

- ***DT*$_{ij}$(*fid*)**:直接信任度(**direct trust**,简称 **DT**).

即在合作过程中,任务管理社区 ssc_i 对任务承包社区 ssc_j 积累的关于 ssc_j 执行功能需求代码为 *fid* 类型任务的直接信任程度,$DT_{ij}(fid)\leq 1$.$DT_{ij}(fid)$ 与 ssc_i 对当前服务质量和历史直接信任度的重视程度有关.用λ表示对

© 中国科学院软件研究所 http://www.jos.org.cn

历史直接信任度的重视程度,1–λ表示对当前服务质量的重视程度,$\lambda\in[0,1]$.λ越小,则对当前服务质量的重视程度越高. $DT_{ij}(fid)$ 的计算如公式(3)所示,该公式也间接反映出了初次合作后仍需考量的设计.直接信任度更新算法见算法 1.

$$DT_{ij}(fid)'=\lambda\cdot DT_{ij}(fid)+(1-\lambda)\cdot CQoS_{ij}(fid) \tag{3}$$

● $\boldsymbol{RT_i(k,j,t,fid)}$: 推荐信任度(**recommendation trust**,简称 **RT**).

即 ssc_i 分配任务 t 时,社区 ssc_k 向社区 ssc_i 推荐的关于社区 ssc_j 在执行 fid 类任务方面的可信任程度.每次 ssc_i 确定是否将任务分配给投标社区 ssc_j 时,会征求其他未投标社区对 ssc_j 的信任度评价,诚信的社区会将自己对 ssc_j 的直接信任度 DT_{kj} 作为推荐值提供给 ssc_i,即 $RT_i(k,j,t,fid)=DT_{kj}(fid)$.

● $\boldsymbol{CRD_i(k,j,fid)}$: 当前推荐偏差(**the current recommended deviation**,简称 **CRD**).

即 ssc_i 对 ssc_k 向其推荐的关于 ssc_j 在执行 fid 类任务方面的信任度偏差度量.由于一些恶意社区可能会向 ssc_i 提供不真实的推荐信息,因此,ssc_i 需要计算 ssc_k 向其推荐的关于 ssc_j 在 fid 类任务方面信任度的偏差,以便为是否采纳 ssc_k 的推荐信任度提供参考.当有多个社区投标竞争 ssc_i 的任务 t 时,这些投标的社区构成了投标社区集,我们用 $BCT_i(t)$表示.本文设定 $BCT_i(t)$中的社区不能向 ssc_i 推荐信任度,避免社区间由于竞争而故意将对手的推荐信任度降低的情况发生.ssc_i 与 ssc_j 合作后,ssc_i 向承包方 ssc_j 给出满意度反馈 $DoS_{ij}(t)$, 并统计 ssc_k 对于 ssc_j 的推荐偏差 $\left|RT_i(k,j,t,fid)-DoS_{ij}(t)\right|$, 则当前推荐偏差 $CRD_i(k,j,fid)$ 的计算方法如公式(4)所示.

$$CRD_i(k,j,fid)=\frac{\sum\left|RT_i(k,j,t,fid)-DoS_{ij}(t)\right|}{RN_i(k,j,fid)} \tag{4}$$

其中, $\sum\left|RT_i(k,j,t,fid)-DoS_{ij}(t)\right|$ 表示 ssc_i 对 ssc_k 向其推荐的关于 ssc_j 在 fid 类任务方面信任度的推荐偏差累加值, $RN_i(k,j,fid)$ 表示 ssc_k 向 ssc_i 推荐关于 ssc_j 执行 fid 类型任务且 ssc_j 在其推荐时中标的次数.

● $\boldsymbol{RD_i(k,j,fid)}$: 推荐偏差(**recommended deviation**,简称 **RD**).

根据对当前推荐偏差和历史推荐偏差的重视程度,可得出推荐偏差 $RD_i(k,j,fid)$. 用θ表示对历史推荐偏差的重视程度,1–θ表示对当前推荐偏差的重视程度,$\theta\in[0,1]$.θ越小,则对当前推荐偏差的重视程度越高. $RD_i(k,j,fid)$ 计算方法如公式(5)所示.

$$RD_i(k,j,fid)'=\theta\cdot RD_i(k,j,fid)+(1-\theta)\cdot CRD_i(k,j,fid) \tag{5}$$

ssc_i 的管理"软件人"可将 $RD_i(k,j,Fid)'$ 与其设定的推荐偏差限定值 RDL_i 进行比较,决定是否采纳 ssc_k 的推荐信任度.社区 ssc_j 执行任务 t 后,任务管理社区 ssc_i 对于提供推荐信任度并被采纳的社区 ssc_k 的推荐偏差更新算法见算法 1.

算法 1. 任务管理社区 ssc_i 对 ssc_j 执行 t 对应的 fid 类型任务的直接信任度更新、对于提供推荐信任度并被采纳的社区 ssc_k 的推荐偏差更新.

输入:任务承包方执行任务结果,预计执行时间 $EET(t)$,实际执行时间 $AET(t)$,任务完成截止时间 $dl(t)$,ssc_k 向 ssc_i 推荐关于 ssc_j 执行 fid 类型任务且 ssc_j 在其推荐时中标的次数 $RN_i(k,j,fid)$, 推荐社区 ssc_k 对于 ssc_j 关于任务 t 的历史推荐偏差 $RD_i(k,j,fid)$, ssc_i 分配给 ssc_j 的任务功能需求代码为 fid 的任务个数 $\left|T_{ij}(fid)\right|$ 及对 ssc_j 的执行 fid 类型任务的满意度累加和 $\sum_{t\in T_{ij}(fid)}DoS_{ij}(t)$, 任务执行失败后的满意度取值 $-\xi$,ssc_i 对 ssc_j 关于 fid 类型任务的历史直接信任度 $DT_{ij}(fid)$, ssc_k 向 ssc_i 推荐的关于 ssc_j 在 fid 类型任务方面的推荐信任度 $RT_i(k,j,t,fid)$, ssc_i 统计的 ssc_k 对 ssc_j 的历史推荐偏差累加和 $\sum\left|RT_i(k,j,t,fid)-DoS_{ij}(t)\right|$.

输出:更新后的直接信任度 $DT_{ij}'(fid)$; 更新后的推荐社区的推荐偏差 $RD_i'(k,j,Fid)$.

```
1  begin
2  set λ,θ
3  set  CRD_i(k,j,fid)=0                    //当期推荐偏差
```

© 中国科学院软件研究所 http://www.jos.org.cn

set $CQoS_{ij}(fid)=0$ //当前服务质量
set $DoS_{ij}(t)=0$ //满意度
if 任务成功完成 **then**
  if $EET(t)\geq AET(t)$ **then**
    $DoS_{ij}(t)=1$
  else if $EET(t)<AET(t)\wedge AET(t)\leq dl(t)$ **then**
    $Dos_{ij}(t)=1-\dfrac{AET(t)-EET(t)}{dl(t)}$
  end if
else //任务执行失败或未在任务截止时间内完成
  $DoS_{ij}(t)=-\xi$
end if
$\sum_{t\in T_{ij}(fid)} DoS_{ij}(t)+=Dos_{ij}(t)$ //ssc_i 对 ssc_j 执行 *fid* 类型任务的满意度累加
$\sum\left|RT_i(k,j,t,fid)-DoS_{ij}(t)\right|+=\left|RT_i(k,j,t,fid)-DoS_{ij}(t)\right|$ //ssc_i 对 ssc_k 向其推荐的关于 ssc_j 在 *fid* 类型任务方面信任度的推荐偏差进行累加
$\left|T_{ij}(fid)\right|+=1$
$CQoS_{ij}(fid)=\dfrac{\sum_{t\in T_{ij}(fid)} DoS_{ij}(t)}{\left|T_{ij}(fid)\right|}$ //计算当前服务质量
$DT_{ij}(fid)=\lambda\cdot DT_{ij}(fid)+(1-\lambda)\cdot CQoS_{ij}(fid)$ //计算直接信任度
$CRD_i(k,j,fid)=\dfrac{\sum\left|RT_i(k,j,t,fid)-DoS_{ij}(t)\right|}{RN_i(k,j,fid)}$ //计算当前推荐偏差
$RD_i(k,j,fid)'=\theta\cdot RD_i(k,j,fid)+(1-\theta)\cdot CRD_i(k,j,fid)$ //计算推荐偏差
Return 更新直接信任度 $DT_{ij}(fid)$, 更新推荐社区 ssc_k 的推荐偏差 $RD_i(k,j,fid)$
end

● ***REP*$_j$ (*fid*)**: 社区信誉(**reputation**).

ssc_i 可根据非投标社区的推荐获得 ssc_j 在执行 *fid* 类型任务的社区信誉,即其他社区对 ssc_j 执行 *fid* 类型任务的认可程度,$REP_j(fid)\leq 1$.计算方法如公式(6)所示.

$$REP_j(fid)=\left(\sum_{\substack{\forall ssc_k\in(SSC-ssc_i-ssc_j)\\ \wedge ssc_k\notin BCT_i(t)\\ \wedge\forall RD_i(k,j,fid)<RDL_i}} RT_i(k,j,t,fid)\right)\Big/\left|ssc_k\right| \tag{6}$$

其中,$\left|ssc_k\right|$ 表示 ssc_i 对任务 *t* 进行招标时所采纳的推荐信任度的社区的个数.

● ***CT*$_{ij}$ (*fid*)**: 社区信任度(**community trust**,简称 **CT**).

此时,可根据直接信任度、社区信誉计算出 ssc_i 对于 ssc_j 在 *fid* 功能类型方面的社区信任度 $CT_{ij}(fid)$, $CT_{ij}(fid)\leq 1$,如式(7)所示.其中,$\alpha\in[0,1]$.α越大,表示对自身直接信任度的重视程度越高.

$$CT_{ij}(fid)=\alpha\cdot DT_{ij}(fid)+(1-\alpha)\cdot REP_j(fid) \tag{7}$$

社区信任度计算算法将在下一节基于社区信任度的社区节点选择算法中一并给出.需要指出的是,当其他社区基于某一 *fid* 在对新加入合一系统社会的社区进行信任度评估时,会由社区的管理"软件人"依据自身知识库中的直接信任度信息表集合,计算出其对整个社会 *fid* 功能的平均信任度,作为本社区对新加入社区 *fid* 功能的直接信任度.而诚信社区对其进行推荐时,会将推荐信任度等值于直接信任度,最终由任务管理方得出其社区声誉.

© 中国科学院软件研究所 http://www.jos.org.cn

由本节内容可见,每个社区分布式存储对其他社区直接信任度的方法,以及在任务分配时计算其他社区推荐信任度得出实时社区信誉的方法,避免了社区信任度集中计算、集中存储、缺乏实时性的问题.社区信任度评估是基于递阶多级式的集中协调加分布式评估的控制结构,基于该结构的组织模型中不同的"软件人"角色配置、关系设计以及基于"软件人群"的协作,解决了第 1.1 节所提到的"缺乏对系统控制结构的改进研究、缺乏对信任度评估所依靠的组织关系研究"这两个问题;无论是直接信任度,还是推荐信任度,都是面向合一系统社区不同任务功能类型角度进行的细化评估,解决了第 1.1 节中提到的"评估对象和目标对象一般为个体,且以对目标对象的综合评估为主,不能很好地解决同一节点在不同领域、不同方面的可信度"这一问题;在计算时兼顾了投标社区当前与历史的执行任务表现,且针对不可靠的社会环境,设计了推荐偏差阈值,通过推荐偏差统计,尽量降低采纳不诚信社区的恶意推荐的概率.此外,进行了竞标社区不能相互推荐的设定,从一定程度上解决了第 1.1 节中提到的"不精确信息过滤和信息时效性"的问题.

在不可靠的合一系统社会中,社区信任度评估其实很大层面上衡量的是社区的诚信,在任务分配选择社区节点时,还应对社区实时的上下文资源状态和其他关键指标进行综合考量.接下来,将研究基于社区信任度的社区节点选择策略,完成任务管理方的任务分配.

3.3 基于社区信任度的社区节点选择策略

当一个社区对以自身为入口的任务基于社会功能进行分解后,它需要将任务分配到社区.本文设定合一系统社会的任务协调以本地社区为先,即对于本社区可以在任务截止时间内完成的任务,管理"软件人"可选择不对其进行招标.

假设 4. 社区内部"软件人"之间的通信时间忽略不计.

当管理"软件人"根据所在社区资源状态决定对任务进行招标分配时,本文选择 4 个因素作为选择任务承包方的衡量指标.实时的任务环境和需求对于任务的等待时间和处理时间具有很高的要求,因此,本文将任务等待时间和处理时间合并统称为投标社区的预计执行时间,以此作为第 1 个衡量指标,该指标也有利于社区间的负载均衡;一个社区往往对一些任务类型比较擅长,即社区完成此类任务的成功率较高,因此,本文将社区基于 *fid* 任务类型的执行成功率作为第 2 个衡量指标;当一个社区对于任务需求的资源充裕时,为减少任务的等待时间,它可以复制启动新的功能"软件人"去执行任务(针对由"软件人"社区中功能"软件人"完成的任务),或当某一个"软件人"执行任务发生故障时,可在任务截止时间及系统资源允许的情况下,将任务转到社区内其他具有相同功能的 SM.fun 或 App-H 上执行,免去了任务管理社区对任务的二次分配,因此,社区拥有任务所需资源的大小将作为第 3 个衡量指标.在任务分配时,这 3 个指标数值由投标社区汇报给招标社区,代表了投标社区的实时能力.此时,一些不诚信的社区为了获得任务会报出有利于中标的虚假数据.因此,这 3 个衡量指标值需要招标社区基于社区信任度进行考量,得出投标社区的能力评估值.此外,任务执行时或完成后,任务管理方都需与任务承包方进行通信,社区间距离越近,通信时间越少,任务完成时间越短,因此,本文将通信距离纳入社区选择的第 4 个衡量指标,结合能力评估值得出招标社区对投标社区的合作倾向因子.

下面,首先基于社区信任度,结合投标社区汇报的 3 个指标数值对投标社区的能力进行评估,得出 ssc_i 对于 ssc_j 的能力评估值 $AA_j(t)$.

定义 4. 设 ssc_i 为任务 t 的招标社区,t 的功能需求代码为 *fid*,ssc_j 为任务 t 的投标社区,则 ssc_i 对于 ssc_j 的能力评估值 $AA_j(t)$ 表示为

$$AA_j(t)=CT_{ij}(fid)\cdot(\beta_1\cdot RET_j(t)+\beta_2\cdot RSR_j(fid)+\beta_3\cdot RR_j(t)) \tag{8}$$

其中,

$AA_j(t)\le 1$;

β_1、β_2、β_3 为权系数,$\beta_1\in[0,1],\beta_2\in[0,1],\beta_3\in[0,1]$,且 $\beta_1+\beta_2+\beta_3=1$;

$CT_{ij}(fid)$ 为 ssc_i 对 ssc_j 在执行 *fid* 功能类型任务的社区信任度,$CT_{ij}(fid)\le 1$;

© 中国科学院软件研究所 http://www.jos.org.cn

$RET_j(t)$ 为相对执行时间, $RET_j(t)=\dfrac{1/EET_j(t)}{\sum_{\forall ssc_j\in SSC(t)}1/EET_j(t)}$, $EET_j(t)$ 表示 ssc_j 投标时汇报的预计执行时间, $SSC(t)$ 表示投标 t 的社区集合, $RET_j(t)\in[0,1]$;

$RSR_j(fid)$ 为 ssc_j 执行 fid 功能类型任务的相对成功率, $RSR_j(fid)=\dfrac{SR_j(fid)}{\sum_{\forall ssc_j\in SSC(t)}SR_j(fid)}$, 其中, $SR_j(fid)=\dfrac{|ST_j(fid)|}{|T_j(fid)|}$, $T_j(fid)$ 为 ssc_j 执行过的功能需求类型为 fid 的任务集合, $|T_j(fid)|$ 表示任务个数, $ST_j(fid)$ 为 ssc_j 成功执行的功能需求类型为 fid 的任务集合, $|ST_j(fid)|$ 表示成功执行的任务个数, $SR_j(fid)\in[0,1]$;

$RR_j(t)$ 为 ssc_j 拥有的 t 需求资源种类的相对总量, $RR_j(t)=\dfrac{R_j(t)}{\sum_{\forall ssc_j\in SSC(t)}R_j(t)}$, $R_j(t)$ 表示 ssc_j 拥有的 t 需求资源的总量, $RR_j(t)\in[0,1]$.

此时,可根据 ssc_i 对 ssc_j 的能力评估值(如 ssc_i 所招标的是必须在一个社区内执行的多种类型任务的任务集 T,$T=\{t_1,t_2,\ldots,t_l,\ldots\}$,则公式(8)中的 $CT_{ij}(fid)$ 变为 T 中每个任务相对的功能需求类型信任度的乘积,即 $CT_{ij}(Fid(T))=\prod_{\forall t_l\in T}CT_{ij}(fid(t_l))$. 同理, $RSR_j(Fid(T))=\prod_{\forall t_l\in T}RSR_j(fid(t_l))$, $RET_j(T)$ 为 ssc_j 完成整个 T 的相对预计执行时间, $RR_j(T)$ 为 ssc_j 拥有的 T 需求资源类型的相对总量)和通信距离计算出 ssc_i 与 ssc_j 进行合作的倾向因子.

定义 5. 设定 ssc_i 为任务 t 的招标社区,ssc_j 为任务 t 的投标社区,则 ssc_i 与 ssc_j 进行合作的倾向因子 $CTF_j(t)$ 如下所示:

$$CTF_j(t)=\upsilon_1 RAA_j(t)+\upsilon_2 RCT_{ij} \tag{9}$$

其中, $\upsilon_1\in[0,1],\upsilon_2\in[0,1],\upsilon_1+\upsilon_2=1$; $RAA_j(t)$ 为 ssc_i 对 ssc_j 执行 t 的相对能力评估值, $RAA_j(t)=\dfrac{AA_j(t)}{\sum_{\forall ssc_j\in SSC(t)}AA_j(t)}$;

RCT_{ij} 为 ssc_i 和 ssc_j 的相对通信时间, $RCT_{ij}=\dfrac{1/d_{ij}}{\sum_{\forall ssc_j\in SSC(t)}1/d_{ij}}$, d_{ij} 为 ssc_i 和 ssc_j 之间的距离;$SSC(t)$表示投标任务 t 的社区集合; $RCT_{ij}\in[0,1]$.

最终,根据下式确定任务承包社区.

if $\exists ssc_j\in\{SSC(t)\}$ and $CTF_j(t)=\max_{\forall ssc_j\in SSC(t)}\{CTF_j(t)\}$, 则 ssc_j 为任务 t 的承包社区.

社区信任度计算(算法 2 中第 3 行~第 13 行)及基于社区信任度的社区节点选择(算法 2 中第 14 行~第 28 行)如算法 2 所示.

算法 2. 社区信任度计算及基于社区信任度的社区节点选择.

输入:推荐社区 ssc_k 对社区 ssc_j 执行 fid 类型任务的推荐信任度 $RT_i(k,j,t,fid)$, 推荐偏差 $RD_i(k,j,fid)$, ssc_i 对 ssc_j 执行 fid 类型任务的直接信任度 $DT_{ij}(fid)$, 投标社区列表 $SSC(t)$, 投标社区 ssc_j 对任务 t 的预计执行时间 $EET_j(t)$, 投标社区 ssc_j 执行 fid 类型任务的成功率 $SR_j(fid)$, ssc_j 拥有的 t 需求资源的总量 $R_j(t)$, ssc_i 与 ssc_j 之间的距离 d_{ij}, 推荐偏差限定值 RDL,关于 t 任务的推荐社区列表 $recoCommList\{t\}$;

输出:中标社区.

1 **begin**

2 初始化 $\beta_1,\beta_2,\beta_3,\upsilon_1$和$\upsilon_2$, 被采纳推荐信任度的推荐社区的推荐偏差累加和 $sumRecoValue$=0,投标社区任务预计执行时间的倒数累加和 $sumEstimateTime$=0,成功率总量 $sumSuccRate$=0,所有投标社区资源累加和 $sumResource$=0,社区间距离的倒数累加和 $sumDistance$=0,能力评估值累加和 $sumAblity$-

© 中国科学院软件研究所 http://www.jos.org.cn

$Assess$=0,社区信任度列表 $List\{CT_{ij}(fid)\}$, 社区信任度 $CT_{ij}(fid)=0$, 投标社区名声 $REP_j(fid)=0$, 社区合作倾向因子 $CTF_j(t)=0$, 推荐信任度被采纳的社区个数 $|ssc_k|=0$, 投标社区能力评估值 $AA_j(t)=0$.

3 **for all** ssc_k **in** $recoCommList\{t\}$ **do**
4 **if** $RD_i(k,j,fid)<RDL_i$ **then**
5 $|ssc_k|+=1$
6 $sumRecoValue+=RT_i(k,j,t,fid)$
7 **end if**
8 **end for**
9 **for all** ssc_j in $SSC(t)$ **do**
10 $REP_j(fid)=sumRecoValue/|ssc_k|$
11 $CT_{ij}(fid)=\alpha\cdot DT_{ij}(fid)+(1-\alpha)\cdot REP_j(fid)$
12 $List\{CT_{ij}(fid)\}\cdot add(CT_{ij}(fid))$
13 **end for**
14 **for all** ssc_j **in** $SSC(t)$ **do**
15 $sumEstimateTime+=1/EET_j(t)$
16 $sumSuccRate+=SR_j(fid)$
17 $sumDistance+=1/d_{ij}$
18 $sumResource+=R_j(t)$
19 **end for**
20 **for all** ssc_j **in** $SSC(t)$ **do**
21 $get\ CT_{ij}(fid)$ **from** $List\{CT_{ij}(fid)\}$
22 $$AA_j(t)=CT_{ij}(fid)\cdot\left(\beta_1\cdot\frac{1/EET_j(t)}{sumEstimateTime}+\beta_2\cdot\frac{SR_j(fid)}{sumSuccRate}+\beta_3\cdot\frac{R_j(t)}{sumResource}\right)$$
23 $sumAblityAssess+=AA_j(t)$
24 **end for**
25 **for all** ssc_j **in** $SSC(t)$ **do**
26 $$CTF_j(t)=\upsilon_1\frac{AA_j(t)}{sumAblityAssess}+\upsilon_2\frac{1/d_{ij}}{sumDistance}$$
27 **end for**
28 **return** 具有最大 $CTF_j(t)$ 值的中标社区
29 **end**

3.4 社区内基于负载均衡的任务分配机制

当任务分配到社区后,管理“软件人”需要将其分配给具体的SM.fun或App-H执行.对于不诚信的社区,它可能在争取到任务之后不执行或中途擅自终止任务,导致任务分配失败;也可能优先执行回报价值高的其他任务,导致任务未在任务截止时间或预计执行时间内完成.对于这些不诚信的社区,已经用社区信任度去筛选.因此,本节将讨论任务分配到诚信社区后,社区管理“软件人”对最终执行任务的SM.fun或App-H的选择.

假设5. 一个SM.fun或App-H一次只能执行一个任务.

一般来说,一个SM.fun或App-H的任务执行成功率越高,处理速率越快,它的任务等待队列越长.因此,在存在多个功能类型相同的SM.fun或App-H的社区内,需要考虑负载均衡.负载均衡的性能主要由任务的等待时间决定,等待时间可简单考虑为只与“软件人”任务等待队列的大小和处理速率有关.因此,社区内的任务分配目标是均衡多个功能类型相同的SM.fun或App-H的负载量,并尽量将任务分配给执行成功率高的“软件人”.

本文以将任务 t 分配到SM.fun为例,阐述最终执行者的选择方法.

© 中国科学院软件研究所 http://www.jos.org.cn

设合一系统社区中具有功能 fid 的 SM.fun 集合为 $\mathbf{A}$, $A=\{SM.fun|SM.fun\to fid\}$,$|A|\geq 1$.集合 $\mathbf{A}$ 中任一 $SM.fun_x$ 的等待队列为 Q_x,Q_x 的大小为 s_x,s_x 与队列中每一个任务的时间复杂度有关,这里设 s_x 的最小值为正在分配的任务 t 的大小. $SM.fun_x$ 的处理速率为 v_x,任务执行成功率为 sr_x,则管理"软件人"将任务分配给 $SM.fun_x$ 的倾向因子 TF_x 为

$$TF_x=\frac{rv_x}{rs_x}\cdot rsr_x \tag{10}$$

其中,rv_x 为 $SM.fun_x$ 在集合 $\mathbf{A}$ 中的相对处理速率, $rv_x=\dfrac{v_x}{\sum\limits_{\forall SM.fun_x\in A}v_x}$, $rv_x\in(0,1]$; rs_j 为 $SM.fun_x$ 在集合 $\mathbf{A}$ 中的相对队列大小, $rs_x=\dfrac{s_x}{\sum\limits_{\forall SM.fun_x\in A}s_x}$, $rs_x\in(0,1]$; rsr_x 为 $SM.fun_x$ 在集合 $\mathbf{A}$ 中的相对成功率, $rsr_x=\dfrac{r_x}{\sum\limits_{\forall SM.fun_x\in A}r_x}$, $rsr_x\in[0,1]$.

最终,管理软件人选择 TF_x 值最大的 SM.fun 执行任务.同理,按此方法选择最佳的 App-H 执行任务.

社区内基于负载均衡的任务分配过程如算法 3 所示.

算法 3. 社区内基于负载均衡的任务分配.

输入:基于某一 fid 的 SM.fun 或 App-H 列表 $List\{App-H(fid)\,||\,SM\cdot fun(fid)\}$, 其中,任一 $App\text{-}H_x$ 或 $SM.fun_x$ 的任务等待队列为 Q_x,Q_x 的大小为 s_x,$App\text{-}H_x$ 或 $SM.fun_x$ 的处理速率为 v_x,$App\text{-}H_x$ 或 $SM.fun_x$ 的历史成功率为 sr_x;

输出:具体执行任务的 SM.fun 或 App-H.

```
1  begin
2  set sumSR=0,sumV=0,sumS=0   //备选 SM.fun 或 App-H 的成功率累加和、处理速率累加和以及队列大小累加和
3  set TF_x=0
4  for all SM.fun_x or App-H_x in List do
5      sumSR+=sr_x, sumV+=v_x, sumS+=s_x
6  end for
7  for all SM.fun_x or App-H_x in List do
8      TF_j = (v_x/sumV) × (sr_x/sumSR) / (s_x/sumS)
9  end for
10 return 具有最大 TF_x 的功能软件人 SM.fun_x 或 App-H_x
```

3.5 社区内基于上下文资源的任务再分配机制

本文设定当选择的 SM.fun 或 App-H 在执行任务过程中发生故障时,任务承包社区的管理"软件人"会在任务截止时间和上下文资源允许的情况下,根据公式(10)选择社区内具有相同功能的其他 SM.fun 或 App-H 执行任务.这也是面向社区的任务分配机制的一个优点,即任务承包社区可在获得任务后在社区内部自主进行分配,选择执行任务的软件人;当任务执行失败时,社区也可基于上下文资源,采用自主再分配的策略,以期避免任务管理社区的惩罚.该策略有效减少了招标社区重新对任务进行分配的次数,省去了任务管理社区再次招/投标的时间,也避免了再分配时将任务分到不诚信的社区情况的发生.因此,面向社区进行任务分配时,将资源作为第 3 项衡量指标的意义也更显重大.如某一"软件人"执行任务失败后,社区内没有符合的 SM.fun 或 App-H 以及充足的系统资源,则需将任务失败消息汇报给招标社区,由招标社区重新对任务进行分配.

社区内基于上下文资源的任务再分配策略具体描述如算法 4 所示.

算法 4. 社区内基于上下文资源的任务再分配.

输入:社区此次执行任务 t 的结果(是否成功),动态备选"软件人"列表 $backMap\{SM.fun||App\text{-}H\}$;

输出:社区内基于上下文再分配的结果.

```
1  begin
2  if 任务执行成功 then
3  if 以本社区为入口的任务
```

© 中国科学院软件研究所 http://www.jos.org.cn

```
4        宣布任务执行结果
5     else
6        将执行任务结果传回给任务管理社区
7     end if
8     else
9     在 backMap{SM.fun(fid)||App-H}内移除执行任务失败的 SM.fun 或 App-H
10     if backMap{SM.fun||App-H}内有其他备选“软件人”and 预计执行时间小于截止剩余时间
11                  call 算法 3,交给最优备选“软件人”SM.fun 或 App-H 执行任务
12                  return 执行结果,back line 2
13     else  //backMap{SM.fun||App-H}内无其他备选“软件人”或预计执行时间大于截止剩余时间
14     if 以本社区为入口的任务
          if 未到截止时间
15                  面向社区基于协调“软件人”重新招标 call 算法 2
                    else
                      任务失败,返回失败信息给用户
                    end if
16     else
17                 将执行任务结果传回给任务管理社区
18                 end if
            end if
19     end if
20     end
```

4 实验验证与分析

拟进行 4 部分实验,完成以下工作.

1) 将提出的面向社区基于社会协调“软件人”的任务分配模型与其他具有代表性的任务分配模型进行性能对比,并分析实验结果.

2) 对模型中提出的社区信任度评估机制进行效果验证.

3) 对模型中提出的社区内基于负载均衡的分配机制、基于上下文资源的再分配机制进行效果验证.

4) 当诚信社区发生变化时,测试面向社区基于社会协调“软件人”的任务分配模型是否具有鲁棒性,并与直接面向“软件人”的分配模型进行比较.

实验仿真平台在 MyEclipse IDE 下使用 Java 语言自主开发.社会网络由 10 个合一系统社区组成,按实验需求设置每个社区的诚信度.当不诚信社区推荐信任度时,按设置的社区诚信度概率进行恶意推荐,即推荐信任度随机降低 50%;不诚信社区在投标时,按其社区诚信度概率谎报自己的能力,即预计执行时间缩短 30%,成功率提高到 100%.

社会中共有 60 个 SM.fun,按每个社区 3~9 个的数量随机分散在 10 个合一系统社区中;30 个 App-H,按每个社区 3~7 个的数量随机分散在 5 个合一系统社区中.每个 SM.fun 分属于 1~6 功能类型中的一种,每个 App-H 分属于 7~9 功能类型中的一种.每个社区的初始资源不同,相同类型任务所需资源大小相等.任务的资源需求种类除 CPU、内存、磁盘、网络 4 种系统资源外,还包括执行任务相应的 SM.fun 或 App-H,也就是说,SM.fun 或 App-H 也是任务执行所需的资源.SM.fun 的处理速率随机设定为 3~9 个任务每秒,App-H 的处理速率随机设定为 1~4 个任务每秒.当社区内系统资源充足时,可复制 SM.fun 执行任务.App-H 执行任务除需 CPU、内存、磁盘、网络 4 种资源外,还对应机器人硬件设备,硬件不能复制,因此 App-H 不能复制.根据现实情况,设置 App-H 比 SM.fun 的处理速率低,因此,在机器人社区执行任务的排队时间一般较“软件人”社区要长.一个合一系统社区可有多个机器人,每个机器人最多拥有 3 种不同功能类型的 App-H,且每一类型的 App-H 只有 1 个.

设置每一个任务经任务管理社区分配后(可多次分配)都可在社会网络中完成.由于“软件人”社区中 SM.fun 的可复制性与“机器人”社区中 App-H 的不可复制性,以及 SM.fun 与 App-H 任务处理速率的较大差距,将分别基于“软件人”社区类型任务(1~6 类型任务)和机器人社区类型任务(7~9 类型任务)进行相关实验.

© 中国科学院软件研究所 http://www.jos.org.cn

4.1 模型间性能对比与分析

4.1.1 实验设置

实验分别设社会中诚信社区的个数为 3 和 7,直接信任度的权值设为 0.7.分别针对“软件人”社区任务类型和机器人社区任务类型,观察不同任务分配模型的性能.当诚信社区数量为 3 时,设置诚信社区的资源较不诚信社区的要多,且底层“软件人”的处理速率和成功率也较高;当诚信社区数量为 7 时,设置不诚信社区的资源较多,但处理速率和成功率较低.随机产生由“软件人”社区执行的任务 1 000 个,分 10 轮投放,每轮投放 100 个,每个社区 10 个;随机产生由机器人社区执行的任务 500 个,分 10 轮投放,每轮投放 50 个,每个社区 5 个.任务分配模型的性能指标选取以下两种.

任务执行总时间.任务执行总时间是指任务集从不同社区(10 个社区同时作为任务入口)进入合一系统社会到任务全部执行完毕所经历的时间.执行总时间与任务分配模型的性能成反比.分别针对“软件人”社区任务和机器人社区任务,记录不同的任务分配模型下、每一轮任务完成后与之前各轮任务的累积执行总时间.实验重复进行 50 次,得出累积平均执行总时间.

任务完成步骤.任务完成步骤的基数为任务个数,每一个任务进入本地社区后即为一步,当本地社区(即任务管理社区)通过招投标交由其他社区执行 1 次,或其他社区执行失败,再次回到本地社区被本地社区执行时,任务完成步骤累加 1 次.分别针对“软件人”社区任务和机器人社区任务,观察不同的任务分配模型下、每一轮任务完成后与之前各轮任务的累积完成步骤.实验重复进行 50 次,得出累积平均完成步骤.

为了验证所提出的面向社区基于社会协调“软件人”任务分配模型的性能,本文将其与直接面向“软件人”的任务分配模型、透明的面向社区基于社会协调“软件人”的任务分配模型和基于资源的任务分配模型进行对比.我们将实际应用下本文提出的任务分配模型称为“不透明的面向社区基于协调‘软件人’的模型”,因为它不可能完全准确地甄别出社会中的不诚信社区,且每个社区执行每个功能类型任务的成功率、社区的上下文资源状态以及预计执行时间对于任务管理社区来说都是黑盒子,是不透明的,它只能在任务投标社区汇报后,借助于本文提出的信任度评估机制和节点选择机制进行任务分配.

其余 3 个模型简单介绍如下.

直接面向“软件人”的任务分配模型.即直接面向 SM.fun 或 APP-H 的任务分配模型,投标者为社会中的 SM.fun 或 APP-H.当中标的“软件人”执行任务失败后,由任务管理社区重新对任务进行面向 SM.fun 或 APP-H 的招标.与传统的直接面向 Agent 的分配模型[11,30,38,41]不同,在投标者所在社区中,具体由哪个或哪些 SM. fun 或 APP-H 投标,其决策是由该社区的管理控制中心——管理“软件人”做出的.

透明的面向社区基于社会协调“软件人”的任务分配模型.透明的模型可以侦测到网络中全部的不诚信社区,且每个社区执行每个功能类型任务的成功率、社区的上下文资源状态以及预计执行时间对于任务管理社区来说都是透明的.这种模型在任务分配时,毋庸置疑是最优的,但在真实不可靠的网络环境中并不存在.实验只是将其作为基准,评估其他模型在不可靠合一系统社会网络中的任务分配性能.虽然在透明的任务分配模型中,招标社区对于投标社区的相关信息可全部掌握,但为比较不透明模型的学习性能,更好地观测不透明模型与透明模型的差距,透明模型的分配机制还应与不透明的相同,即还需将社区信任度评估与基于社区信任度的社区节点选择等所产生的计算时间和通信开销计入其中(虽然分配决策时不采用这些数据).

基于资源的任务分配模型.此种模型中,任务将被分配给拥有其所需资源总量最多的社区[26,30,31,42,43].在做实验时,依然采用本文提出的社区信任度评估、基于负载均衡的任务分配以及社区内基于上下文资源的任务再分配机制,只是在选择社区节点时,不考虑社区执行此类任务的成功率和预计执行时间,重点考虑社区的系统资源以及可执行此任务的“软件人”资源.

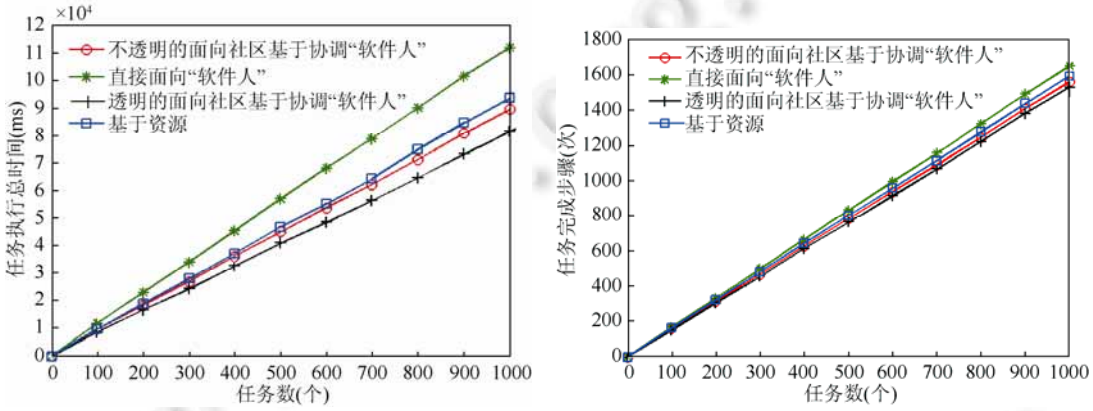
4.1.2 结果与分析

图 4 和图 5 分别展示的是 3 个诚信社区、4 种模型分配“软件人”社区类型任务和机器人社区类型任务最终消耗的任务执行总时间和任务完成步骤.

由图 4 和图 5 可以看出,透明的面向社区基于社会协调“软件人”的分配模型执行时间最短,任务完成步骤最

© 中国科学院软件研究所 http://www.jos.org.cn

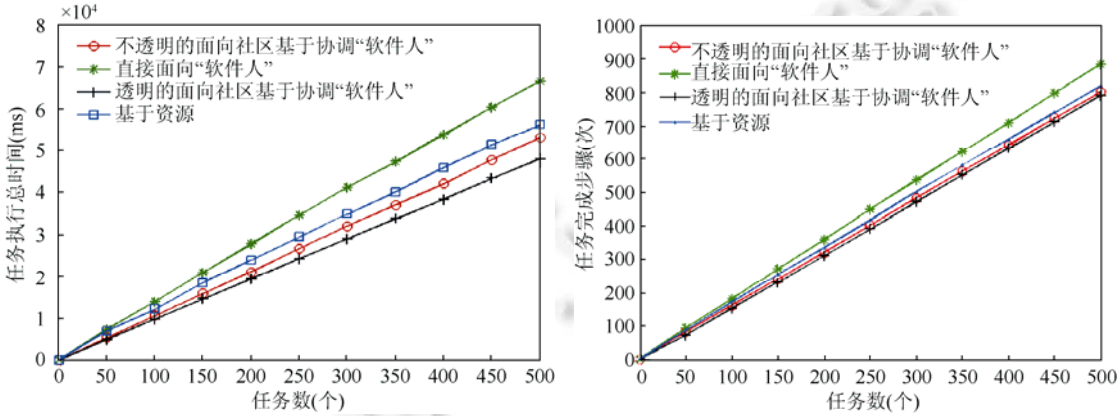
少,性能最优.这是由于社区是否诚信、社区执行每个功能类型任务的成功率、社区的上下文资源状态以及对招标任务的预计执行时间对于任务管理社区来说都是透明的.其次是不透明的面向社区基于社会协调“软件人”的分配模型,通过实验结果可以看出它的性能与透明的模型差距不大,验证了社区信任度的评估机制以及基于社区信任度的社区节点选择机制在不可靠社会任务分配中的有效性.直接面向 SM.fun 或 App-H 的分配模型产生的执行总时间最长,任务完成步骤最多,这是由于该分配模型面对的是 SM.fun 或 App-H 个体,没有基于社区上下文的再分配机制,当中标的 SM.fun 或 App-H 执行任务失败时,任务承包方将失败信息传回给任务管理社区,由任务管理方面向社会中的 SM.fun 或 App-H 进行再分配,增加了任务管理方再次招投标分配任务的时间开销,且有可能分配到不诚信的社区,导致任务的再次分配.基于资源的模型与本文提出的模型性能接近,这是由于,在实验设置时,诚信社区资源较多,速率和成功率也较高,基于资源的分配模型使用社区信任度评估机制在寻找资源多的诚信社区时,恰好弥补了其不考虑速率和成功率的缺点.但因是不透明的信任度评估,该模型也会将少量任务分配给不诚信的社区,但由于只考虑资源,不考虑社区执行该任务的预计执行时间和成功率,其性能仍逊色于不透明的面向社区基于社会协调“软件人”的分配模型.



(a) 任务执行总时间 (b) 任务完成步骤

Fig.4 Performance comparison of four kinds of models on SoftMan community type tasks under condition of having three honest communities

图 4 3 个诚信社区下 4 种模型分配“软件人”社区类型任务的性能对比



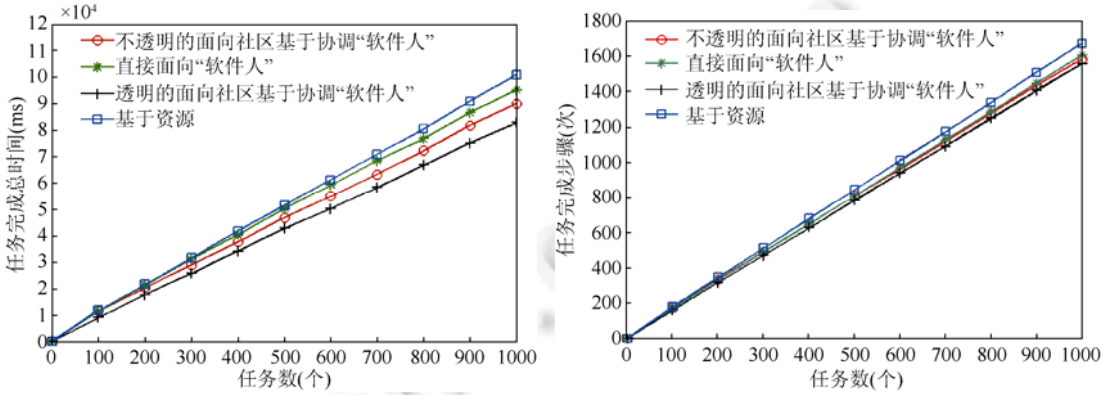
(a) 任务执行总时间 (b) 任务完成步骤

Fig.5 Performance comparison of four kinds of models on robot community type tasks under condition of having three honest communities

图 5 3 个诚信社区下 4 种模型分配机器人社区类型任务的性能对比

© 中国科学院软件研究所 http://www.jos.org.cn

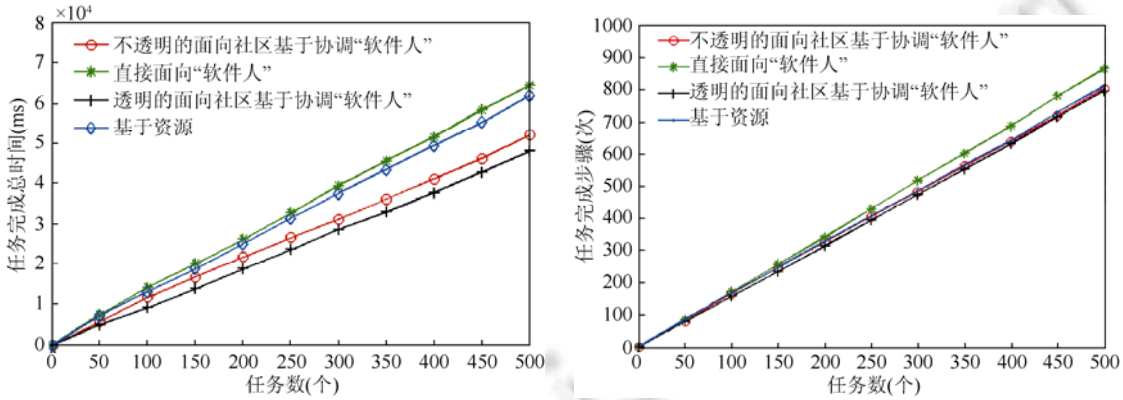
图 6 和图 7 分别展示的是 7 个诚信社区、4 种模型分配“软件人”社区类型任务和机器人社区类型任务最终消耗的任务执行总时间和任务完成步骤.由图 6 和图 7 可以看出,透明的面向社区基于社会协调“软件人”的分配模型执行时间仍然最短,任务完成步骤最少,性能最优.其次是不透明的面向社区基于社会协调“软件人”的分配模型.在分配“软件人”社区类型任务时,基于资源的任务分配模型的执行总时间最长,任务完成步骤最多,其次是直接面向“软件人”的任务分配模型;在分配机器人社区类型任务时,直接面向“软件人”的任务分配模型的执行总时间最长,任务完成步骤最多,其次是基于资源的任务分配模型.综合图 4~图 7 所示结果可以得出:在不可靠社会网络中,无论诚信社区比例是大是小、社区资源和“软件人”成功率如何设置,一般情况下,面向社区基于社会协调“软件人”的任务分配模型性能优于直接面向“软件人”的任务分配模型和基于资源的任务分配模型.



(a) 任务执行总时间 (b) 任务完成步骤

Fig.6 Performance comparison of four kinds of models on SoftMan community type tasks under condition of having seven honest communities

图 6 7 个诚信社区下 4 种模型分配“软件人”社区类型任务的性能对比



(a) 任务执行总时间 (b) 任务完成步骤

Fig.7 Performance comparison of four kinds of models on robot community type tasks under condition of having seven honest communities

图 7 7 个诚信社区下 4 种模型分配机器人社区类型任务的性能对比

此外,图 6、图 7 相对于图 4、图 5 来说,基于资源的分配模型与不透明的面向社区基于社会协调“软件人”的分配模型相比,任务执行总时间差距增大.这是由于,在实验设置时,不诚信社区资源较多,在基于资源的任务分配机制下,虽有社区信任度评估,但仍有部分任务分配到这些社区,而这些社区的速率设置和成功率设置较低,因此执行任务时间会增加;其次,因是不透明的信任度评估,其在将任务分配给诚信社区时,也只会考虑资源,

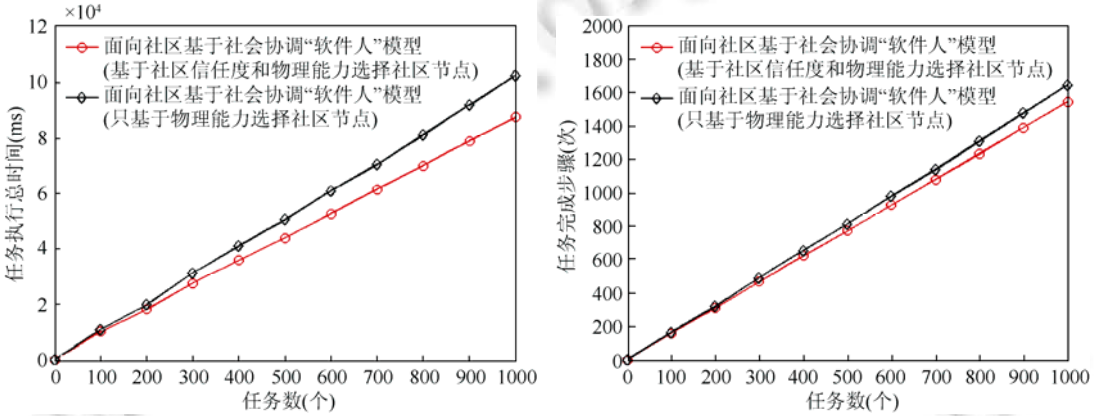
© 中国科学院软件研究所 http://www.jos.org.cn

不考虑社区执行该任务的预计执行时间和成功率.两者合一,任务完成总时间差距增大.但基于资源的分配模型与不透明的面向社区基于社会协调“软件人”的分配模型的任务完成步骤差距变化并不明显,这是由于,虽然诚信社区资源较少,但是由于社区信任度评估机制良好的性能,大多数情况下仍能找到诚信社区分配的任务,因此,任务完成步骤差距变化不大.

4.2 社区信任度评估机制效果验证

在不可靠的合一系统社会中,通过社区信任度评估机制可以将任务分配给诚信度、执行特定任务类型成功率较高的社区.本节实验中,模型将以分配“软件人”社区任务为例,分别采用基于社区信任度和社区物理能力的社区节点选择策略、只基于社区物理能力的社区节点选择策略,面向社区分配任务,并通过任务执行总时间和任务完成步骤的对比,验证所提出的社区信任度评估机制的有效性.当任务管理社区只基于物理能力进行任务承包社区选择时,将不再需要满意度评价形成直接信任度,也不需要其他社区的信任度推荐,其通信开销将大为降低.因此,实验不再比较通信次数.

设置包含 10 个社区的社会网络中诚信社区的数量为 3,直接信任度权重为 0.7.随机产生由“软件人”社区执行的任务 1 000 个,分 10 轮投放,每轮 100 个,每个社区 10 个,进行 50 次重复实验,计算得出任务分配模型分别采用这两种社区节点选择策略时的累积平均执行总时间和任务完成步骤,实验结果如图 8 所示.



(a) 任务执行总时间 (b) 任务完成步骤

Fig.8 Performance comparison of using and not using community trust evaluation mechanism when allocating SoftMan community type tasks

图 8 分配“软件人”社区类型任务时采用与不采用社区信任度评估机制的性能对比

由图 8 可以看出,在任务管理社区基于社区信任度和社区物理能力选择任务承包社区时,较只基于社区物理能力分配任务降低了任务执行总时间和任务完成步骤,验证了本文提出的社区信任度评估机制的有效性以及在提升模型任务分配性能方面的积极作用.

4.3 社区内任务分配机制效果验证

在本文提出的面向社区基于社会协调“软件人”的分配模型中,在社区内部分配任务时采取了负载均衡的任务分配机制,且针对 SM.fun 或 App-H 执行任务失败的情况,提出了基于社区上下文资源的任务再分配策略.本节将对模型中的这两个关键部分进行效果验证.

4.3.1 负载均衡的分配机制

负载均衡可以降低任务的等待时间,缩短合一系统社会执行任务的总时间.本节将采用社区内负载均衡的分配机制与不采用此机制作一性能对比,以验证负载均衡机制对缩短任务执行时间方面的作用.在社区内基于负载均衡分配时,考虑了 SM.fun 或 App-H 的处理速率、队列大小以及历史成功率;而不采用负载均衡机制分配

© 中国科学院软件研究所 http://www.jos.org.cn

任务时,只考虑历史成功率.

我们首先以一个机器人社区为例,观察同一类型任务在相应功能 App-H 的任务分配数量,验证公式(10)在负载均衡方面的效果,并在相同的 App-H 成功率设置下,测试基于处理速率、队列大小的负载均衡对任务执行总时间的影响.设该机器人社区机器人的个数为 3,每个机器人中的 App-H 一次只能执行一个任务;单独向该社区发送 400 个 *fid* 类型代码为 7 的机器人任务,每一个任务的大小相等.机器人社区中的功能类型代码为 7 的 App-H 具体设置如表 2 中 1~4 列所示.

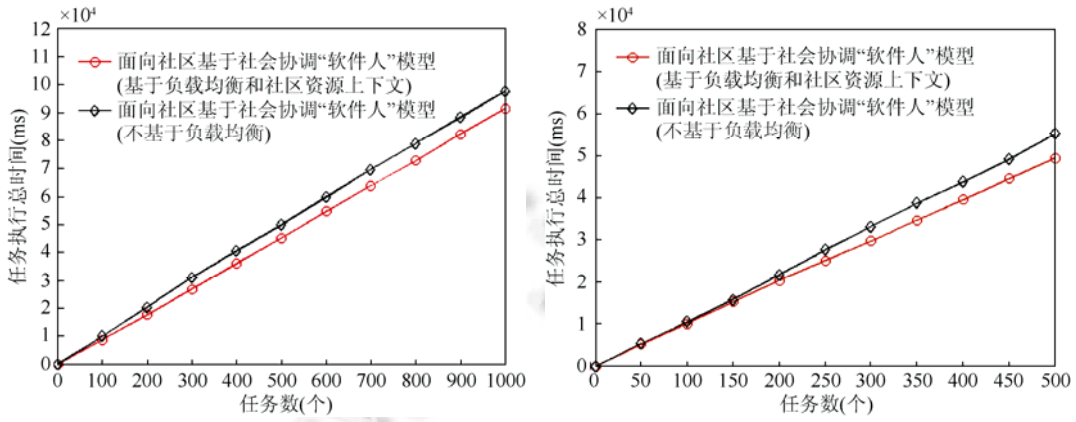
任务集重复分配执行 50 次,得出采用和不采用社区内基于负载均衡的任务分配机制时,每个机器人中的 App-H 最终执行成功的平均任务数量,如表 2 中第 5 列和第 6 列所示;得出整个机器人社区完成任务的平均执行总时间(任务执行总时间定义与第 4.1.1 节中的定义相同),如表 2 中第 7 列和第 8 列所示.由实验结果可见,社区内采用基于负载均衡的任务分配机制时(同时考虑 App-H 的处理速率、队列大小以及历史成功率),不同机器人但相同功能类型的 App-H 间最终执行成功的任务数量相差不大,分别为 130、143 和 127,而只基于任务成功率分配时每个 App-H 最终执行成功的任务数量为 49、31、320,负载很不均衡;最终,负载均衡分配机制下的任务执行总时间最短,这是因为任务的分配较为均衡,降低了等待时间.

Table 2 Comparision of successful task number among the same function App-Hs and comparision of total task execution time of robot community using and not using load balancing mechanism

表 2 采用与不采用社区内负载均衡任务分配机制时,社区中相同功能类型 App-H 执行成功任务数量和社区任务执行总时间对比

机器人编号	App-H 功能类型	App-H 速率设置(任务个数/S)	App-H 成功率设置(%)	执行成功任务数量(负载均衡)	执行成功任务数量(不基于负载均衡)	任务执行总时间(ms)(负载均衡)	任务执行总时间(ms)(不基于负载均衡)
1	7	3	75	130	49		
2	7	2.5	80	143	31	72 786	198 880
3	7	2	90	127	320		

下面,我们将在合一系统社会中以各个社区为并发任务入口分配任务,验证社区内基于负载均衡的任务分配机制对提高整个系统性能的积极作用.合一系统社会包含 10 个合一系统社区,诚信社区的数量设为 3,直接信任度权重设为 0.7,随机产生由“软件人”社区执行的任务 1 000 个,分 10 轮投放,每轮投放 100 个,每个社区 10 个;随机产生由机器人社区执行的任务 500 个,分 10 轮投放,每轮投放 50 个任务,每个社区 5 个.分别针对“软件人”社区任务和机器人社区任务,进行 50 次重复实验,计算得出任务分配模型采用与不采用该机制时,任务的累积平均执行总时间.任务执行总时间定义与第 4.1.1 节中的定义相同,实验结果如图 9 所示.



(a) 分配“软件人”社区类型任务 (b) 分配机器人社区类型任务

Fig.9 Comparision of total task execution time using and not using load balancing mechanism

图 9 采用与不采用负载均衡机制的任务执行时间对比

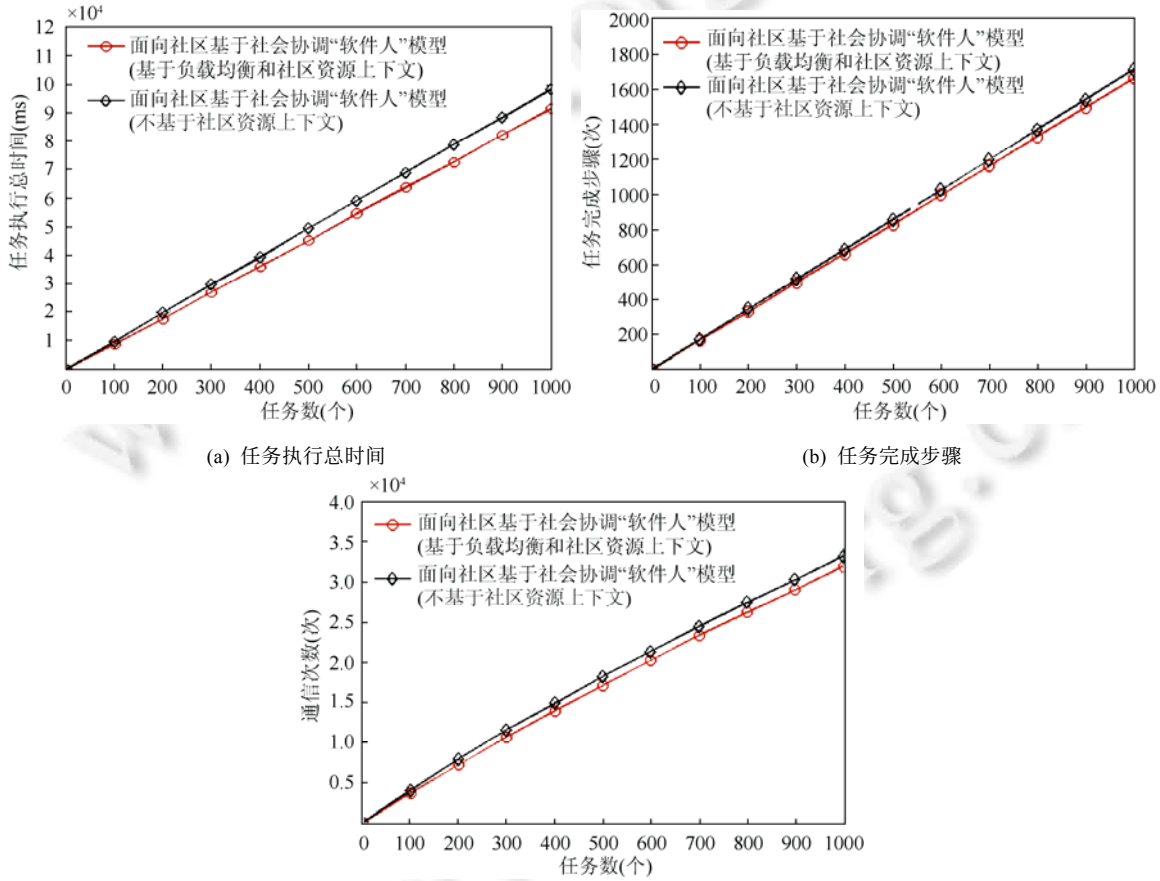
© 中国科学院软件研究所 http://www.jos.org.cn

由图 9 可以看出,管理“软件人”在社区内部分配任务时,采用负载均衡的分配机制可以降低任务的执行总时间,提高模型分配性能.

4.3.2 基于上下文的任务再分配机制

前文提到,当选择的 SM.fun 或 App-H 在执行任务过程中发生故障时,任务承包社区的管理“软件人”会在任务截止时间和上下文资源允许的情况下,在社区内选择其他具有相同功能的其他 SM.fun 或 App-H 重新执行任务,以期避免任务管理社区的惩罚.本实验将对社区内基于上下文的任务再分配机制对模型任务分配性能的积极作用作一验证.

设置包含 10 个社区的社会网络中诚信社区的数量为 3,直接信任度权重为 0.7.随机产生由“软件人”社区执行的任务 1 000 个,分 10 轮投放,每轮投放 100 个任务,每个社区 10 个;随机产生由机器人社区执行的任务 500 个,分 10 轮投放,每轮投放 50 个任务,每个社区 5 个.分别针对“软件人”社区任务和机器人社区任务,进行 50 次重复实验,计算得出本文所提出的任务分配模型采用与不采用该机制时的累积平均执行总时间、完成步骤和通信次数.任务执行总时间和任务完成步骤的定义与第 4.1.1 节中的定义相同,通信次数指的是从任务分配到任务完成整个过程中,不同社区与社会协调“软件人”或不同社区之间通信次数的总和,根据假设 4,社区内部“软件人”之间的通信不计入在内.实验结果如图 10 和图 11 所示.



(a) 任务执行总时间 (b) 任务完成步骤

(c) 通信次数

Fig.10 Model performance comparison of using and not using the redistribution mechanism based on context when allocating SoftMan community type tasks

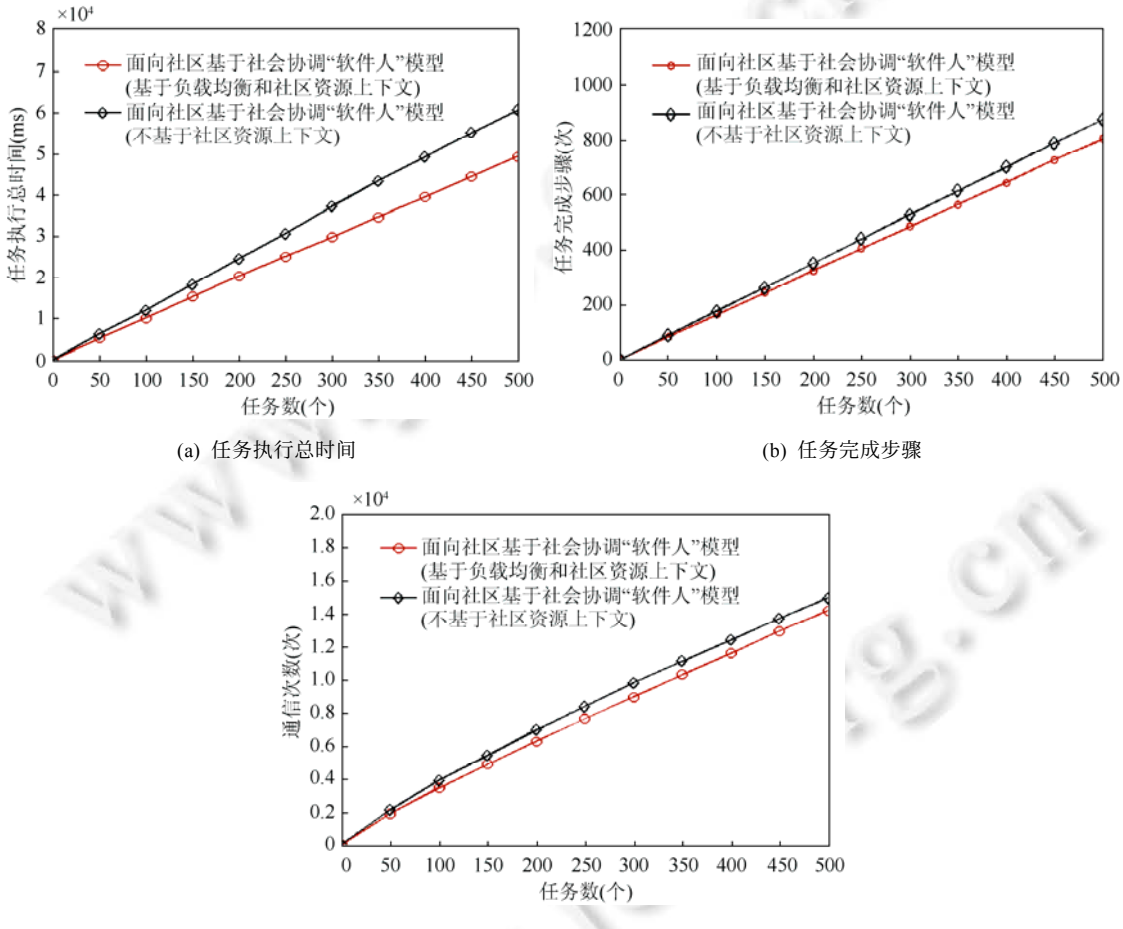
图 10 分配“软件人”社区类型任务采用与不采用基于上下文的任务再分配机制的性能对比

© 中国科学院软件研究所 http://www.jos.org.cn

由图 10 和图 11 可以看出,社区内采用基于上下文的任务再分配机制时,降低了任务执行总时间、完成步骤和通信次数,提高了模型的分配性能.这是由于,当社区执行任务失败时,其在社区内部采用基于上下文的任务再分配机制可省去任务管理方对该任务再次招投标的时间和通信开销,也避免了任务管理方再次招投标时将任务分配给不诚信社区、不诚信社区执行任务失败的情况发生.

4.4 模型鲁棒性

因面向社区基于协调“软件人”任务分配模型和直接面向“软件人”任务分配模型在面向实体方面的截然不同,这部分将测试此两种模型在诚信社区动态变化环境下的鲁棒性,并对实验结果进行对比、分析.诚信社区动态变化是指社会中的诚信社区将发生变化,即原本不诚信的社区会变为诚信社区,原本诚信的社区中会有一部分变为不诚信社区.



(a) 任务执行总时间 (b) 任务完成步骤

(c) 通信次数

Fig.11 Model performance comparison of using and not using the redistribution mechanism based on context when allocating robot community type tasks

图 11 分配机器人社区类型任务采用与不采用基于上下文的任务再分配机制的性能对比

随机产生 100 个面向“软件人”社区的任务,这 100 个任务将被重复分配 30 轮,每轮每个社区固定分配 10 个任务.第 1 轮任务分配前,将从 10 个社区中选择 3 个社区作为不诚信社区;在第 11 轮任务分配开始时,这 3 个社区将动态地变为诚信社区,而在原先诚信社区中将随机挑选 3 个变为不诚信社区.实验重复进行 50 次,得出每

© 中国科学院软件研究所 http://www.jos.org.cn

轮任务的平均执行总时间和平均完成步骤,实验结果如图 12 所示.

随机产生 100 个面向机器人社区的任务,这 100 个任务将被重复分配并执行 30 次(轮),每轮每个社区固定分配 10 个任务.第 1 轮任务分配前,将从 5 个拥有机器人的社区中选择 2 个社区作为不诚信社区,社会中其余社区为诚信社区;在第 11 轮任务分配开始时,这两个社区将动态变为诚信社区,而原先拥有机器人的 3 个诚信社区中的其中 2 个将变为不诚信社区.实验重复进行 50 次,得出每轮任务的平均执行总时间和平均完成步骤,实验结果如图 13 所示.

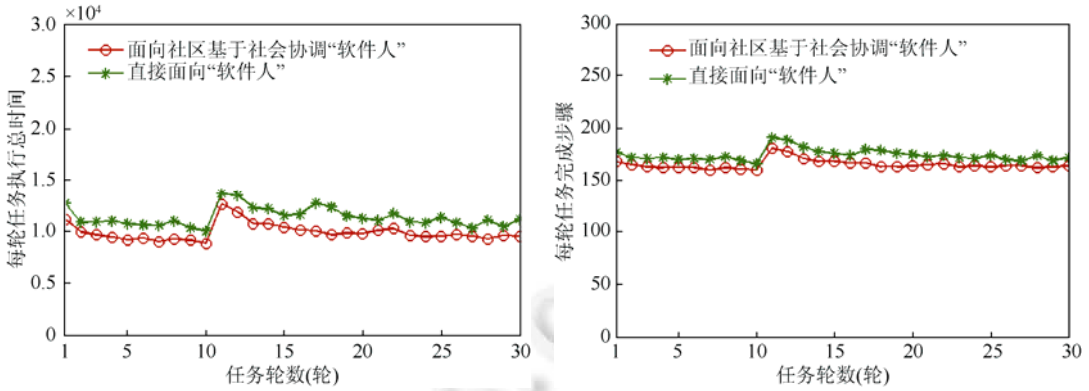


Fig.12 The experimental results of using two allocation models allocating SoftMan community type tasks under the social circumstance of the honest communities changing dynamicly

图 12 诚信社区动态变化时,分配"软件人"社区类型任务两种模型的实验结果

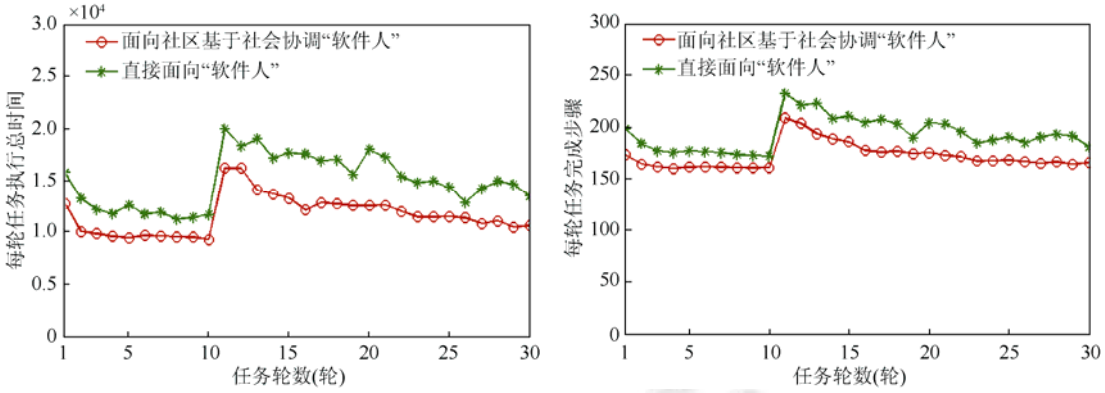


Fig.13 The experimental results of using two allocation models allocating robot community type tasks under the social circumstance of the honest communities changing dynamicly

图 13 诚信社区动态变化时,分配机器人社区类型任务两种模型的实验结果

由图 12 和图 13 可以看出,在第 1 轮~第 10 轮任务,随着轮次的增加,两种模型每轮任务的平均执行总时间和平均完成步骤逐渐下降,说明这两种模型都有自学习功能;随着分配经验的累积,性能越来越优,最后趋于稳定,说明了模型中针对社区或"软件人"的信任度评估机制以及基于信任度的社区节点或"软件人"选择策略的有效性.在第 11 轮,诚信社区发生变化,面向社区基于社会协调"软件人"的任务分配模型和直接面向"软件人"的任务分配模型在每轮任务执行总时间和完成步骤上有一个大幅度的升高.这是由于,诚信社区发生变化,第 1 轮~第 10 轮积累下来的社区信任度不再适用.此后,随着任务轮次的增加,执行总时间和完成步骤总体为下降趋势,但在下降过程中,面向社区基于社会协调"软件人"的任务分配模型下降趋势稳定,波动较小,且较直接面向"软

© 中国科学院软件研究所 http://www.jos.org.cn

件人”的任务分配模型恢复性能快;而直接面向“软件人”的任务分配模型在性能恢复过程中每轮任务的执行总时间和完成步骤波动较大,且恢复性能较慢.这是由于动态变化的是社区的诚信度,但直接面向“软件人”的任务分配模型面对的是“软件人”个体.每次任务执行后任务管理方只能修正对一个“软件人”的信任度,当社区变为不诚信社区(或变为诚信社区)后,如社区中的另一“软件人”投标任务时,其仍以之前对该“软件人”形成的信任度历史进行分配,而不能顾及该“软件人”所在社区的诚信度已发生变化、一个“软件人”是否诚信的决策是由社区管理“软件人”决定的实际情况,因此性能恢复较慢,且波动性较大.总的来说,两种模型都具有鲁棒性,但面向社区基于社会协调“软件人”分配模型的鲁棒性比直接面向“软件人”的分配模型要强,且稳定性要好.

5 结 论

在合一系统中,“软件人”之间的协作很大程度上受组织模型结构的影响,本文所研究的任务分配作为协作的重要一环,首先即设计了合理的基于“软件人”群协作组织模型.

任务分配的主要目标是最大化系统的整体效能并且尽可能快地完成任务.最大化系统的整体效能即充分利用系统资源;尽可能快地完成任务即最小化每个任务的执行时间.每个任务的执行时间应包括 4 个部分:分配任务时与其他社区的协商时间、在非管理“软件人”中的等待时间、执行过程中任务管理社区与任务承包社区间的通信时间以及任务在非管理“软件人”中的处理时间.因此,在不可靠的合一系统社会应用背景下,为达到任务分配的目标,分配模型在社区间进行任务分配时采取了借助社会协调“软件人”的协商机制、基于直接信任度和社区声誉的社区信任度评估机制和基于社区信任度的社区节点选择机制;在社区内进行任务分配时采取了基于负载均衡的分配机制和基于上下文资源的任务再分配策略.实验结果表明,基于社会协调“软件人”面向社区的任务分配模型与直接面向“软件人”的任务分配模型和基于资源的任务分配模型相比,任务执行总时间最短,满足了不可靠社会网络中的任务分配优化目标.此外,所提出的模型对动态的社会环境具有鲁棒性和自适应性,可适应除合一系统社会外的多种现实应用.

本文提出的任务分配模型是基于固定的合一系统社会组织模型设计的,社会协调“软件人”的通信压力较大,因此下一步的工作将研究如何使每个社区基于自身任务分配的历史经验,建立自己的社会上下文环境,即熟人社会关系,以扩充任务分配的路径,缓解社会协调“软件人”的通信压力,降低社会网络中的通信密度.此外,对于模型中的权重参数,实验时是通过设置多个参数组合,每一个组合经过多次实验,根据最小化任务执行时间的任务分配目标取最优设置值.但这种参数取值方法较为传统,工作量也较大,因组合设置有限,最终参数取值也较难很近似地接近最优.此外,当社区诚信动态变化时也不能及时调整.因此下一步工作我们将专门研究参数取值及在实时分配过程中的自适应调节问题.

References:

[1] Kim JH, Yang W, Sincak P, Xu P. Robot Intelligence Technology and Applications 3. Springer Int'l Publishing, 2015.

[2] Tan M, Wang S. Research progress on robotics. Acta Automatica Sinica, 2013,39(7):963−972 (in Chinese with English abstract). [doi: 10.3724/SP.J.1004.2013.00963]

[3] Navaravong L, Zhen K, Shea J M, Dixon WE. Formation reconfiguration for mobile robots with network connectivity constraints. Network IEEE, 2012,26(4):18−24. [doi: 10.1109/MNET.2012.6246748]

[4] Nigl F, Li S, Blum J E, Lipson H. Structure-Reconfiguring robots: Autonomous truss reconfiguration and manipulation. IEEE Robotics & Automation Magazine, 2013,20(3):60−71. [doi: 10.1109/MRA.2012.2201579]

[5] Zeng GP, Tu XY, Wang HB. SoftMan Research and Applications. Beijing: Science Press, 2007 (in Chinese).

[6] Wilensky U, Rand W. An Introduction to Agent-Based Modeling: Modeling Natural, Social, and Engineered Complex Systems with NetLogo. MIT Press, 2015. [doi: 10.1063/PT.3.2884]

[7] Liu DY, Yang K, Chen JZ. The research status and development trend of agent. Ruan Jian Xue Bao/Journal of Software, 2000,11(3):315−321 (in Chinese with English abstract). http://www.jos.org.cn/ch/reader/create_pdf.aspx?file_no=20000305&journal_id=jos

[8] Lu Q, Zeng G, Zhang W, Tu X. SoftMan and agent. In: Proc. of the 2005 IEEE Networking, Sensing and Control. 2005. 904−907. [doi: 10.1109/ICNSC.2005.1461313]

© 中国科学院软件研究所 http://www.jos.org.cn

[9] Wu DF, Zeng GP, Xiao CE, Zhang QC. Open robot agent: Construction of host SoftMan. Journal of Computer Applications, 2015,35(6):1766−1772 (in Chinese with English abstract). [doi: 10.11772/j.issn.1001-9081.2015.06.1766]

[10] Wu DF, Zhang QC, Zhang X, Xiao C. Research on appendage SoftMan based on machine cogmatics. Open Automation and Control Systems Journal, 2014,6:1480−1490. [doi: 10.2174/1874444301406011480]

[11] Weerdt MMD, Zhang Y, Klos T. Multiagent task allocation in social networks. Autonomous Agents and Multi-Agent Systems, 2012,25(1):46−86. [doi: 10.1007/s10458-011-9168-3]

[12] Ohtsuki H, Hauert C, Lieberman E, Nowak MA. A simple rule for the evolution of cooperation on graphs and social networks. Nature, 2006,441(7092):502−505. [doi:10.1038/nature04605]

[13] Jiang Y, Zhou Y, Wang W. Task allocation for undependable multiagent systems in social networks. IEEE Trans. on Parallel & Distributed Systems, 2013,24(8):1671−1681. [doi: 10.1109/TPDS.2012.249]

[14] Tu XY, Z. Wang Z and Y. Guo YH. Large Scale Systems Control Theory. Beijing: Beijing University of Posts and Telecommunications Press, 2005 (in Chinese).

[15] TU XY. Generalized model, intelligent simulation, Soft-Man. Computer Simulation, 2011,28(7):224−228 (in Chinese with English abstract). [doi: 10.3969/j.issn.1006-9348.2011.07.056]

[16] Aref A, Tran T. FTE: A fuzzy logic based trust establishment model for intelligent agents. In: Proc. of the 2015 IEEE/WIC/ACM Int'l Conf. on Web Intelligence and Intelligent Agent Technology (WI-IAT). IEEE, 2015. 133−138. [doi 10.1109/WI-IAT.2015.105]

[17] Huynh TD. Trust and reputation in open multi agent systems [Ph.D. Thesis]. Southampton: Electronics and Computer Science, University of Southampton, 2006.

[18] He LJ, Huang HK, Zhang Wei. A survey of trust and reputation systems in multiagent systems. Journal of Computer Research and Development, 2013,39(7):963−972 (in Chinese with English abstract).

[19] Hendrikx F, Bubendorfer K, Chard R. Reputation systems: A survey and taxonomy. Journal of Parallel & Distributed Computing, 2014,75:184−197. [doi: 10.1016/j.jpdc.2014.08.004]

[20] Zacharia G, Moukas A, Maes P. Collaborative reputation mechanisms for electronic marketplaces. Decision Support Systems, 2000,29(4):371−388. [doi: 10.1016/S0167-9236(00)00084-1]

[21] Sabater J, Sierra C. Social ReGreT, a reputation model based on social relations. ACM SIGECOM Exchanges, 2002,3(1):44−56. [doi: 10.1145/844331.844337]

[22] Mui L, Mohtashemi M, Halberstadt A. Notions of reputation in multi-agents systems: A review. In: Proc. of the 2002 Int'l Conf. on Trust, Reputation, and Security: Theories and Practice. Springer-Verlag, 2002. 280−287. [doi: 10.1145/544741.544807]

[23] Amazon. 2016-06-03. http://www.amazon.com/

[24] Taobao. 2016-06-03. https://www.taobao.com/

[25] Geetha G, Jayakumar C. Implementation of trust and reputation management for free-roaming mobile agent security. IEEE Systems Journal, 2015,9:1−11. [doi: 10.1109/JSYST.2013.2292192]

[26] Jiang Y, Zhou Y, Li Y. Reliable task allocation with load balancing in multiplex networks. ACM Trans. on Autonomous & Adaptive Systems, 2015,10(1):1−32. [doi: 10.1145/2700327]

[27] Khosravifar B, Bentahar J, Gomrokchi M, Alam R. CRM: An efficient trust and reputation model for agent computing. Knowledge-Based Systems, 2012,30(2):1−16. [doi: 10.1016/j.knosys.2011.01.004]

[28] Basheer GS, Ahmad MS, Tang AYC, Graf S. Certainty, trust and evidence: Towards an integrative model of confidence in multi-agent systems. Computers in Human Behavior, 2015,45:307−315. [doi: 10.1016/j.chb.2014.12.030]

[29] Ntemos K, Kalouptsidis N, Kolokotronis N. Managing trust in diffusion adaptive networks with malicious agents. In: Proc. of the Signal Processing Conf. (EUSIPCO), the 23rd European. IEEE, 2015. [doi: 10.1109/EUSIPCO.2015.7362351]

[30] Jiang Y, Li Z. Locality-Sensitive task allocation and load balancing in networked multiagent systems: Talent versus centrality. Journal of Parallel & Distributed Computing, 2011,71(6):822−836. [doi: 10.1016/j.jpdc.2011.01.006]

[31] Jiang Y, Jiang J. Contextual resource negotiation-based task allocation and load balancing in complex software systems. IEEE Trans. on Parallel & Distributed Systems, 2009,20(5):641−653. [doi: 10.1109/TPDS.2008.133]

[32] Hunt S, Meng Q, Hinde C, Huan g T. A consensus-based grouping algorithm for multi-agent cooperative task allocation with complex requirements. Cognitive Computation, 2014,6(3):338−350. [doi: 10.1007/s12559-014-9265-0]

[33] Su Z, Jiang J, Liang C, Zhang G. A distributed algorithm for parallel multi-task allocation based on profit sharing learning. Acta Automatica Sinaca, 2011,37(7):865−872. [doi: 10.1016/S1874-1029(11)60212-7]

© 中国科学院软件研究所 http://www.jos.org.cn

[34] Abdallah S, Lesser V. Modeling task allocation using a decision theoretic model. In: Proc. of the 4th Int'l Joint Conf. on Autonomous Agents and Multiagent Systems. ACM, 2005. 719−726. [doi: 10.1145/1082473.1082583]

[35] Xu H, Mandal S, Pattipati KR, Kleinman DL, Mishra M. An optimization-based distributed planning algorithm: A blackboard-based collaborative framework. IEEE Trans. on Systems Man & Cybernetics Systems, 2014,44(6):673−686. [doi: 10.1109/TSMC.2013.2276392]

[36] Kota R, Gibbins N, Jennings N. Decentralised approaches for self-adaptation in agent organisations. ACM Trans. on Autonomous & Adaptive Systems, 2012,7:1−28. [doi: 10.1145/2168260.2168261]

[37] Val ED, Rebollo M, Vasirani M, Fernández A. Utility-Based mechanism for structural self-organization in service-oriented MAS. ACM Trans. on Autonomous & Adaptive Systems, 2014,9(3):1−24. [doi: 10.1145/2651423]

[38] Wang W, Jiang Y. Multiagent-Based allocation of complex tasks in social networks. IEEE Trans. on Emerging Topics in Computing, 2015,1−12. [doi: 10.1109/TETC.2015.2403200]

[39] Kinnebrew JS, Biswas G. Efficient allocation of hierarchically-decomposable tasks in a sensor Web contract net. In: Proc. of the 2009 IEEE/WIC/ACM Int'l Conf. on Web Intelligence and Intelligent Agent Technology. IEEE Computer Society, 2009. 225−232. [doi: 10.1109/WI-IAT.2009.154]

[40] Qiu HP, Qin Y, Hu R, Guan L. Study on the task allocate based on improved contract net in multi-agent system. Computer Science, 2012,39(6A):279−282 (in Chinese with English abstract).

[41] Rahimzadeh F, Khanli L M, Mahan F. High reliable and efficient task allocation in networked multi-agent systems. Autonomous Agents and Multi-Agent Systems, 2015,29(6):1−18. [doi: 10.1007/s10458-014-9273-1]

[42] Wang W, Jiang Y. Community-Aware task allocation for social networked multiagent systems. IEEE Trans. on Cybernetics, 2014,44(9):1529−1543. [doi: 10.1109/TCYB.2013.2289327]

[43] Jiang Y, Huang Z. The rich get richer: Preferential attachment in the task allocation of cooperative networked multiagent systems with resource caching. IEEE Trans. on Systems Man & Cybernetics Part A Systems & Humans, 2012,42(5):1040−1052. [doi: 10.1109/TSMCA.2012.2186439]

附中文参考文献:

[2] 谭民,王硕.机器人技术研究进展.自动化学报,2013,39(7):963−972. [doi: 10.3724/SP.J.1004.2013.00963]

[3] 曾广平,涂序彦,王洪伯.“软件人”研究与应用.北京:科学出版社,2007.

[7] 刘大有,杨鲲,陈建中.Agent 研究现状与发展趋势.软件学报,2000,11(3):315−321. http://www.jos.org.cn/ch/reader/create_pdf.aspx?file_no=20000305&journal_id=jos

[9] 武丹凤,曾广平,张青川,肖超恩.开放式机器人智体——宿主“软件人”的构建.计算机应用,2015,35(6):1766−1772. [doi: 10.11772/j.issn.1001-9081.2015.06.1766]

[14] 涂序彦,王枞,郭燕慧.大系统控制论.北京:北京邮电大学出版社,2005.

[15] 涂序彦.广义模型智能仿真软件人.计算机仿真,2011,28(7):224−228. [doi: 10.3969/j.issn.1006-9348.2011.07.056]

[18] 贺利坚,黄厚宽,张伟.多 Agent 系统中信任和信誉系统研究综述.计算机研究与发展,2008,45(7):1151−1160.

[40] 裘杭萍,覃垚,胡汭,管留.多 Agent 系统中基于改进合同网模型的任务分配研究.计算机科学,2012,39(6A):279−282.

武丹凤(1982−),女,山西清徐人,博士生,讲师,CCF 专业会员,主要研究领域为智能机器人,“软件人”技术,社会计算,分布式计算.

张绳昱(1995−),男,本科生,主要研究领域为网络工程,分布式计算.

于思淼(1994−),男,学士,主要研究领域为高性能并行计算,软件工程.

张锐文(1996−),男,本科生,主要研究领域为网络工程,复杂系统建模.

© 中国科学院软件研究所 http://www.jos.org.cn