Hans Journal of Data Mining 数据挖掘, 2022, 12(2), 161-172
Published Online April 2022 in Hans. http://www.hanspub.org/journal/hjdm
https://doi.org/10.12677/hjdm.2022.122017

Hans汉斯

基于市场权值的二阶段多agent任务分配算法

万宇杰*，伏明兰#，陈得宝，陈吕强

淮北师范大学计算机科学与技术学院，安徽 淮北

收稿日期：2022年3月6日；录用日期：2022年4月6日；发布日期：2022年4月15日

摘 要

当前对于多Agent系统中比较热点问题是多agent的资源分配和多agent的任务分配，即如何将合适的任务分配给合适的agent，以优化整体任务分配结果。但是考虑到其中的agent具有自利性问题的研究却很少。为了有效的解决自利agent任务分配的问题，构造出一个包含最优竞争和市场协作的两阶段任务分配模型，在最优竞争的基础上，市场协作阶段进一步协调和优化最优任务的分配。考虑到候选人之间的关系可能最终影响市场的总体收益，提出“市场权值”的概念来描述更精确的合作关系。基于市场权值的两阶段分配模型，不仅能让任务候选人充分发挥自身优势参与竞争，保护任务候选人利益，而且能优化市场的总体收益。最后，通过仿真结果验证本文提出算法的有效性。

关键词

多agent系统，自利agent，市场，任务分配

Two-Stage Multi-Agent Task Allocation Algorithm Considering Market Weight

Yujie Wan*, Minglan Fu#, Debao Chen, Lvqiang Chen

School of Computer Science and Technology, Huaibei Normal University, Huaibei Anhui

Received: Mar. 6th, 2022; accepted: Apr. 6th, 2022; published: Apr. 15th, 2022

Abstract

At present, the hot issues in multi-agent system are multi-agent resource allocation and multi-agent task allocation, that is, how to allocate appropriate tasks to appropriate agents to optimize the overall task allocation results. However, considering the self-interest of the agent, there is little research on the problem. In order to effectively solve the problem of task allocation of self-interest

*第一作者。
#通讯作者。

文章引用: 万宇杰, 伏明兰, 陈得宝, 陈吕强. 基于市场权值的二阶段多 agent 任务分配算法[J]. 数据挖掘, 2022, 12(2): 161-172. DOI: 10.12677/hjdm.2022.122017

agent, a two-stage task allocation model including optimal competition and market cooperation is constructed. On the basis of optimal competition, the market cooperation stage further coordinates and optimizes the optimal task allocation. Considering that the relationship between candidates may ultimately affect the overall income of the market, the concept of "market weight" is proposed to describe a more accurate cooperative relationship. The two-stage allocation model based on market weight can not only enable task candidates to give full play to their own advantages, participate in competition and protect the interests of task candidates, but also optimize the overall income of the market. Finally, the simulation results verify the effectiveness of the proposed algorithm.

Keywords

Multi-Agent System, Self-Interested Agent, Market, Task Allocation

1. 引言

当对多 agent 系统的中的个体进行任务分配时，需要考虑到 Agent 个体的思维状态是否受到感情、智慧、社会关系等方面因素的影响，Agent 的思考能力不一定表现在某个人的智能行为中，更可能因为社会活动的影响而以各种形式展示出来。当前，多 Agent 系统的核心问题是如何使这些 agent 之间进行相互协作，从而进行问题的求解。与劳务市场中工人的合作行为相似，Agent 能够相互合作的前提是多 Agent 系统中的每个个体 agent 都具有交互功能[1]。Agent 之所以被赋予意识观念，是因为多 Agent 系统本身就是一种用来研究现实世界的抽象工具[2]。联盟可以为那些独立的 agent 提供富有弹性且便捷的就业空间与场景，使得这些 agent 在固定时间内组成一个团体，通力合作以实现协同发展和资源整合，从而能够更加智慧、灵活和便捷的完成特定目标。

随着参与者的持续增加，市场也因此变得更加复杂。参与者不再是纯粹的竞争或者合作关系。如国内最大电动汽车制造商比亚迪与特斯拉之间既有竞争关系又有合作关系[3]。他们都在研发电车电池，这样他们就会互相竞争以获得市场份额。同时，比亚迪也是特斯拉在华市场最大合作伙伴之一。相较于传统电池而言，比亚迪为特斯拉提供的刀片电池可以将利用效率比传统电池包提高 50%以上，同时可将成本降低了 20%~30%，以供特斯拉用于开拓低端产品线。在这方面，它们又相互合作。在劳务市场任务分配的过程中，这种情况也是存在的。这意味着在竞争模式下，每个参与者可以相互竞争以获得最大利益，在其他的一些情况下，他们会相互合作，使自己的利益最好。近年来，人们对博弈决策者之间的合作问题进行了大量的研究，然而却很少去研究非合作博弈中自利 agent 的应对机制。本文在联盟技能博弈模型下[4]，对自利 agent 任务分配问题进行了探讨。考虑到多自利 agent 系统是一组受自身利益驱使的参与者，他们对其他参与者的行为做出反应，加入使自己的个体成本函数最小化的联盟,但是传统的任务分配算法往往会忽视这些。

由于上述差异，现有的多 agent 任务分配算法无法直接解决自利 agent 背景下复杂任务的分配问题。本文通过一个自由的劳务市场中任务分配问题为研究对象，设计了一个包含最优竞争和协商合作的两阶段任务分配算法(two-stage multi-agent task allocation algorithm based on market weights)。与其他算法不同的是，基于市场协作的两阶段任务分配算法不仅充分保护了参与者的主体性、保障了市场参与者的利益，

且保证了市场的整体收益。

本文的剩余部分组织如下。第 2 节总结分析了相关问题的国内外研究现状。第 3 节建立了市场中复杂任务效益分配和任务分配的问题模型。第 4 节描述了(TSABMW)算法的基本思想，并对其收敛性进行了分析。第 5 节的仿真结果表明，本文提出的算法能够有效地将复杂任务和所得收益合理有效地分配给理性用户。第 6 节对本文的工作进行了总结。

2. 相关工作

随着 MAS 的发展，研究人员在多 agent 任务分配方面取得了很多的成果。现行 MAS 的很多问题都与资源分配和约束条件有关[5]，其中最主要的是地理位置限制。因此，在分布式多 agent 任务分配环境下，通过协商方式形成联盟进行任务的分配与完成的方法得到了广泛的研究，有很多方法可以解决这个问题，比如通过谈判分配任务，基于市场的方法和团队组件问题[6]。文献[7] [8] [9] [10]在联盟形成问题上进行了深入研究，假设一项任务可以由一组 agent 完成，agent 个体试图找到团队成员，系统将任务的适当部分分配给每个成员，以实现效率最大化。kara 和 Bektas [11]提出的数学规划方法通常追求最优解，然而随着问题规模增长，计算成本呈指数增长。Wooldridge 等人[12] [13]于 2006 年首次提出 CRG (Co-operative Resource Games)，详细分析了合作伙伴问题的时间复杂度，并讨论 CRG 与 QCG (Quality Coalition Games)的关系，特别是在什么情况下两者之间切换。针对代理资源和能力的流失，张国富等[14]提出了合作伙伴和代表的想法，对基于复杂联盟的连续生成算法进行了精密设计，提出了有效的多颗粒凝集，可以加入多个合作伙伴以进行任务处理。Sandholm 等[15]研究了联盟形成产生的边界问题。得出该问题是 NP-hard 的结论。对双边联盟结构绘制的充要性进行了完整论证。文献[16]提出了一种联盟结构最优生成算法，通过对联盟上界的剪枝大大缩小了搜索面积。进化算法，[17]提出了一种基于解的算法，多代进化和杂交蚁群遗传算法的低效率。对于单一代理联盟的创建，[18]建设性地引入了“贡献量元素”来评估个体对联盟的贡献量[19]。使用了记录该团体过去记录的高质量解决方案，并在最合适的粒子突变的基础上使用多组相互平行搜索，防止陷入局部最大值，并筛选粒子簇以加速收敛。这类进化算法虽然拥有比数学规划法更低的计算复杂度，但是得到解的质量会随着问题规模(agent 个数与任务个数)的上升而迅速下降。参考文献[20]使用差分进化算法用于解决单一 agent 伙伴问题和粒子质量的速度，并肯定了建立单一工作伙伴关系结构的研究也很有意义，因为在 MAS 中，多个任务之间并不总是存在优先级，单一实体只允许参与一个联盟，它不能同时支持两个或多个操作。在多智能体系统中单个任务的最佳合作伙伴结构应该有尽可能多的合作伙伴来完成这项任务。在实际中拥有最适合特定任务的 agent 所组成的联盟被认为是最合适的合作伙伴结构。

3. 问题模型定义

本节给出问题模型的定义，所研究的模型是一个建立于联盟技能博弈模型之上的自由劳务市场。联盟技能博弈中的自立 agent 符合现实中市场参与者具有相互冲突的偏好。当参与者是自利的情况下，形成稳定的联盟的最为重要的条件是用恰当的合作方式和合理的收益分配。博弈论中的联盟机制考虑了收益如何分配的问题，并提供了若干解决的思路。一个核心的思路是联盟之间可以通过协商，通过任务选择权力之间的交换，达到所有参与者都没有异议的结果。

该联盟主要包括三个集合，用户集 $R \coloneqq \{r_1, r_2, \cdots, r_n\}$，技能集 $S \coloneqq \{s_1, s_2, \cdots, s_l\}$，和任务集 $T \coloneqq \{t_1, t_2, \cdots, t_m\}$，其中 $n = |R|, l = |S|, m = |T|$。其中 $|*|$ 代表每一个集合中所包含元素的个数。当 $i = 1, 2, \cdots, n$，$j = 1, 2, \cdots, l$，$k = 1, 2, \cdots, m$，$RS_{i,j} = 1$ 表示用户 r_i 拥有技能 s_j。$ST_{j,k} = 1$ 完成任务 t_k 需要技能 s_j，否则不需要，任务相应的效益使用向量 $U: \{u_1, u_2, \cdots, u_m\}$ 表示。TS 表示所有任务的效益分配方案，当 $k = 1, 2, \cdots, m$，

$j=1,2,\cdots,l$，$TS_{k,\tau}(\tau)$表示在τ时刻，t_k分配给s_j的收益份额。如果$ST_{i,j}=0$，则$TS_{k,j}(\tau)=-2$。

用户的状态：在τ时，用户r_i的状态用一个6元组表示：

$r_i(\tau):=\langle RS_{i,\cdot},cost_{i,\cdot},w_i(\tau),RT_{i,\cdot}(\tau),provide_{i,\cdot}(\tau),ru_{i,\cdot}(\tau),RInsys_i(\tau)\rangle$，矩阵$RS$的第$i$行用$RS_{i,\cdot}$表示，代表了用户$r_i$拥有的技能。$cost_{i,\cdot}$为矩阵$cost$的第$i$行，表示用户$r_i$使用相应技能所需花费的成本。$w_i(\tau)$表示在$\tau$时刻该用户$r_i$的市场权重。$RT_{i,\cdot}(\tau)$为$RT(\tau)$矩阵的第$i$行，代表用户$r_i$的当前任务选择情况。$provide_{i,\cdot}(\tau)$代表技能分配矩阵$provide(\tau)$第$i$行，$provide_{i,\cdot}(\tau)$是一个包含$l$个元素的向量，$provide_{i,j}(\tau)=1$代表用户$r_i$为当前选择的任务提供了技能$s_j$，否则不为其提供。$ru_{i,\cdot}(\tau)$为一个包含$m$个元素的向量，$ru_i(\tau):=\{ru_{i,1}(\tau),\cdots,ru_{i,k}(\tau),\cdots,ru_{i,m}(\tau)\}$，其中$ru_{i,k}(\tau)\geq 0\,(k\in\{1,2,\cdots,m\})$代表如果用户$r_i$在时刻$\tau$选择任务$m$将会获得的收益。$RInsys_i(\tau)$表示当前用户在系统中的存在状态，$RInsys_i(\tau)=1$（$RInsys_i(\tau)=0$）代表当前用户已经选定一个能够完成的任务并且退出市场(在系统中待选择任务)。

任务的状态：在τ时刻任务$t_k\in T$的状态由一个5元组表示：

$t_k(\tau):=\langle ST_{\cdot,k},TSN_k,u_k,RT_{\cdot,k}(\tau),T_kInSys(\tau)\rangle$。其中$ST_{\cdot,k}$代表$ST$的的第$k$列，是一个包含$l$个元素的列向量。$j\in\{1,2,\cdots,l\}$，$k\in\{1,2,\cdots,m\}$，$ST_{i,k}=1(ST_{i,k}=0)$代表完成任务$k$需要(不需要)技能$S_j$。$TSN_k$代表任务$k$需要技能的个数。$u_k$是任务$t_k$对应的收益。

$RT_{\cdot,k}(\tau)$代表任务分配矩阵$RT(\tau)$的第k列，代表当前任务被哪些用户选择。$T_kInsys_i(\tau)=1$代表在时刻τ任务已经获得所有的技能，并且能被完成，下一步将该任务从市场中剔除。$T_kInsys_i(\tau)=-1$代表在时刻τ，任务t_k没有找到全部所需技能或者完成该任务t_k所需的用户的技能成本大于所能提供的收益，则将该任务从市场中删除。$T_kInsys_i(\tau)=0$表示在时刻τ，任务t_k处于待完成状态。

任务分配本质上是一个多目标优化问题，而联盟技能博弈问题的目标是找到一个任务分配方式使系统总收益最高。故可设在时刻τ_0对n个用户$\{r_1,r_2,\cdots,r_n\}$和k个任务$\{t_1,t_2,\cdots,t_n\}$进行任务分配得到任务选择矩阵$RT^{n\times k}$矩阵中元素为：

$$RT_{ij}=\begin{cases}1 & r_i\text{选择任务}t_j\\ 0 & r_i\text{不选择任务}t_j\end{cases}$$

对任务分配方案的比较在于其收益矩阵和Sum和代价矩阵C，如下所示：

$$\begin{cases}Sum=\begin{bmatrix}sum_{11}&\cdots&sum_{1k}\\ \vdots&\ddots&\vdots\\ sum_{n1}&\cdots&sum_{nk}\end{bmatrix}\\ Cost=\begin{bmatrix}cost_{11}&\cdots&cost_{1k}\\ \vdots&\ddots&\vdots\\ cost_{n1}&\cdots&cost_{nk}\end{bmatrix}\end{cases}$$

sum_{ij}代表任务t_j分配给用户agent r_i的份额。$cost_{ij}$代表用户agent r_i完成任务t_j需要付出的技能成本。

技能提供矩阵$provide_i^{k\times j}$代表了参与者$agentr_i$为任务提供技能的情况，技能收益矩阵$revenue_i^{k\times j}$代表了参与者$agentr_i$为任务提供技能分别带来的收益。

$$\begin{cases}provide\begin{bmatrix}provide_{11}&\cdots&provide_{1j}\\ \vdots&\ddots&\vdots\\ provide_{k1}&\cdots&provide_{kj}\end{bmatrix}\\ revenue\begin{bmatrix}revenue_{11}&\cdots&revenue_{1j}\\ \vdots&\ddots&\vdots\\ revenue_{k1}&\vdots&revenue_{kj}\end{bmatrix}\end{cases}$$

其中 $sum_{ij}=\sum_{i=1}^{k}\sum_{j=1}^{j}provide_{ij}revenue_{ij}$，同理可得 $cost_{ij}=\sum_{i=1}^{k}\sum_{j=1}^{j}provide_{ij}sigcost_{ij}$。

以上两个指标用于衡量任务分配的效果，其对应的总收益和总代价仍如下：

$$\begin{cases}SumRT=\sum_{i=1}^{n}\sum_{j=1}^{k}sum_{ij}RT_{ij}\\ SumCost=\sum_{i=1}^{n}\sum_{j=1}^{k}sum_{ij}cost_{ij}\end{cases}$$

对于给定的 $provide_i^{k\times j}(\tau)$ 和 $revenue_i^{k\times j}(\tau)$ 则系统总收益定义为：

$$SysRev(\tau)=SumRT(\tau)-SumCost(\tau)$$

假设用户具有两种市场状态：调度和待调度状态。调度和待调度状态由该用户 r_i 的市场权重 w_i 在 τ 时刻的排序先后来决定。处于待调度状态下的用户在选择任务时，首先考虑当前能带来最大收益的任务，而不考虑任务的完成情况。处于待调度状态下的用户 r_i 会放弃当前已获得的个体收益跟随调度者 r_i 去选择共同完成一个可能会带来更大收益的任务。调度者和待调度者的关系以及数量会随着市场复杂情况产生动态的变换。

4. 基于市场协作的多 agent 任务分配模型

4.1. 阶段 1 的任务选择策略

为了算法描述方便，首先给出一些定义。

(1) $P(i,k,\tau)$ 代表在时刻 τ，用户 r_i 能为任务 t_k 提供的技能的集合，记为 $P_{num}(i,k,\tau)$，其计算公式如下所示：

$$p_{num}(i,k,\tau):=\begin{cases}P_{num}(i,k,\tau-1)+left(\tau),RT_{i,k}(\tau-1);\\ need_{i,k}-\sum_{j=1}^{l}I_2(i,j,k,\tau),\text{否则}\end{cases}$$

如果 $p_{num}(i,k,\tau)>0$，则在当前的任务分配情形之下，用户 r_i 能为其选择的任务 t_k 提供技能，若 $p_{num}(i,k,\tau)=0$，则代表任务 t_k 所需的技能不能被用户 r_i 提供。$\sum_{i'\in\{1,\cdots,n\},i'\neq i}RT_{i',k}(\tau)\cdot provide_{i',j}(\tau)=0$ 代表当用户 r_i 在 τ 时刻选择任务时，任务 t_k 所需技能 s_j 不被 $R/\{r_i\}$ 中的用户提供。$\sum_{i'\in\{1,\cdots,n\},i'\neq i}RT_{i',k}(\tau-n)\cdot provide_{i',j}(\tau-n)\geq 1$ 则代表了用户 r_i 在 τ 时刻选择任务时，任务 t_k 所需技能 s_j 已被 $R/\{r_i\}$ 中的一个用户提供。

(2) $N_i(\tau):=\{t_k\in T\,|\,P(i,k,\tau)\neq\varnothing\}$。

(3) $complete(i,k,\tau)$ 代表如果用户 r_i 在时刻 τ 选择任务 k，任务 k 能不能完成。

$$complete(i,k,\tau):=\begin{cases}1,RT_{i,k}(\tau)=1\wedge\sum_{j=1}^{l}I_1(i,k,\tau)=TSN_k\\ 0,\text{否则}\end{cases}$$

在 $RT(\tau)$ 的任务分配情境下，若任务 t_k 拥有所需的技能 s_j，则 $I_1(j,k,\tau)=1$，任务 t_k 没有获得所需技能 s_j 或所需技能在市场中所有用户 r_i 都无法提供时，则 $I_2(j,k,\tau)=0$。

(4) $C_i(\tau):=\{t_k\in T\,|\,complete(i,k,\tau)=1\}$。

(5) $ru_{i,k}(\tau)$ 代表用户 r_i 在时刻 τ 选择任务 t_k，则用户 r_i 能获得个体收益，其计算公式如下：

$$ru_{i,k}(\tau)=\begin{cases}\dfrac{u_k}{TSN_k}\cdot p_{num}(i,k,\tau), & complete(t_k,i,\tau)\wedge p_{num}(i,k,\tau)\neq 1\\ 0, & \text{否则}\end{cases}$$

其中 u_k 为任务 t_k 对应的收益，所有任务对应的收益用向量 $U:\{u_1,u_2,\cdots,u_m\}$ 表示。TSN_k 代表完成任务 t_k 总共需要技能的个数。

用户轮流进行任务选择的过程最优竞争如算法 5.1 所示。3~10 行为单个用户 r_i 在时刻 $\tau+1$ 的选择意向。1~12 行为一轮任务选择。

算法 1：最优竞争

1：WHILE 出现用户 agent 在前后两轮中任务选择方案不同的情况
2：FOR $i=0$ to n
3：IF $N_i(\tau)\cap C_i(\tau)\neq\varnothing$
4：$T_{\max k}=\underset{t_k\in N_i(\tau)\cap C_i(\tau)}{\arg\max}\ ru_{i,k}(\tau)$
5：r_i 选择任务 $\max k$ 且得到个人效益 $ru_{i,k}$
6：ELSE IF $N_i(\tau)\neq\varnothing$ and $C_i(\tau)=\varnothing$
7：$T_{\max k}=\underset{t_k\in N_i(\tau)}{\arg\max}\ ru_{i,k}(\tau)$
8：r_i 选择任务 $\max k$ 且得到个人效益 $ru_{i,k}$
9：ELSE
10：r_i 不进行任何操作，保留在系统中
11：END FOR
12：END WHILE

4.2. 下面简单说明算法第一阶段的收敛性

对于任意的两个用户 agent r_a' 和 agent r_b' （$r_a'\in R$，$r_b'\in R$），r_a' 和 r_b' 只有拥有相同的技能和拥有不同的技能这二种情况。第 1 种情况下，对于双方来说，对方的技能是多余的，所以不会选择对方与其合作，转而去各自寻找能给自己带来最大收益的任务。在第 2 种情况中，r_a' 只能选择协助 r_b' 去完成某个任务，或是分别选择不同的任务。这是因为如果 r_a' 倾向与 r_b' 合作而 r_b' 不意愿与 r_a' 合作的话，r_a' 选择的任务将不能被完成，r_a' 所获得的收益就会为 0。则在下一回合选择的过程中，r_a' 将不再愿意与 r_b' 合作，将会选择其他能够提供 r_b' 所拥技能的其他用户进行合作，故在下一回合，r_a' 也不再需要 r_b'。

4.3. 阶段二的任务选择策略

算法 2：市场协作

1：FOR $i=0$ TO $i=$ CheckNum //CheckNum 代表当前市场中权值最大的 τ 个工人
2：$T_{MaxTask}\leftarrow\underset{t_k\in T\cap T_k InSys\cap RT_{i,k}=1}{\arg\max}\ AverageU_k$
3：发起者选择任务 $T_{maxtask}$
4：WHILE 任务当前无法被完成 and 合作参与者在系统中 and 合作参与者被当前任务需要
5：合作参 Rh 与者选择该任务
6：记录合作参与者 Rh 的选择并剔除

Continued

7： IF 任务当前可以被完成
8：将刚才没有参与合作的参与者从市场中剔除
9：其他用户用最优竞争策略再次达到纳什均衡
10：计算系统收益
11：IF Step2Revenue >Step1Revenue
12： 保留任务选择状态
13： ELSE 考察下一个发起者发起的待合作任务

4.4. 简单说明阶段二算法的有效性

由于任务分配是一个 NP 难问题，最优竞争阶段无法保证可以为用户分配到最优的任务。因为在最优竞争模型中，用户 r_i 所作出的反应只依托于个体的效益追求，这就意味着可能存在用户 agent 不满意目前负责的任务。在竞相选择能给自己带来最大收益的任务后，任务参与者通常出现合作行为。例如，在劳务市场中，一个工程项目往往由几个部分组成，如家庭装修需要刷漆工，电工，水工等。拥有最强能力的工人往往拥有工程承办权(即有权力发起合作)。在赢得工程承包的机会后，会邀请要价最低的工人参与其合作以相互配合完成整个项目。各部分的参与者发挥自己的长处配合完成整个项目。在考虑工人愿意放弃当前收益较小任务以获得更高收益的情况下。

5. 实例分析

如图 1 所示是一个简单的联盟技能博弈模型。在图中 $R=\{r_1,r_2\}$ 代表 2 个用户 agent， $S=\{s_1,s_2\}$ 表示两个技能的集合， $T=\{t_1,t_2,t_3\}$ 表示 3 个任务的集合。对于任务 t_1 和服务 t_3，因为其被完成所需技能个数只有一个，所以任务收益将全部都分配给其唯一需要的技能。对于任务 t_2 由于需要两个技能，则将收益在所需技能之间平均分配(u_k/TSN_k)。

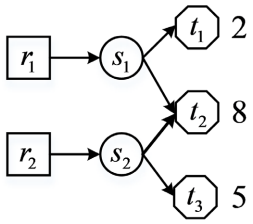

Figure 1. Graphic representation of coalitional skill game model
图 1. 联盟技能博弈的图形表示

对于“阶段 1”，若 r_1 和 r_2 都选择任务 t_2，则 r_1 和 r_2 的收益均为 4。因为用户 agent 是自利的，在下一回合任务选择中， r_2 将优先选择能够给自己带来最大利益的任务 t_3。若 t_2 没有获得所需全部技能的情况下， r_1 也会选择能给自己带来最大收益的任务 t_1。所以对于阶段一来说，无论 r_1 和 r_2 在开始选择了哪个任务，最终得到的纳什均衡点都是 r_1 选择 t_1， r_2 选择 t_3，系统总收益为 7。

在“阶段 2”，用户权值为所拥有技能的出度的权重之合。 $s_1\to t_1$ 为 2， $s_1\to t_2$ 为 4， $s_2\to t_2$ 为 4， $s_2\to t_3$ 为 5，则 $w_{r_1}=0.4$， $w_{r_2}=0.6$。 $w_{r_2}>w_{r_1}$，所以服务 agnet r_2 比 r_1 更优先享有任务 t_2 的合作发起权，任务 t_2 所缺少技能恰好为 r_1 所拥有，且系统总收益大于原来总收益，故用户 r_1 选择任务 t_2。第二谈判的过程包括合作待参与者是否愿意参与合作以及参与合作以后所获得是效益如何分配。这里将系统增加的收

入按照权值分配给参与合作的每个任务，则 r_1 的收入为 $2+\left(Sum_2-Sum_1\right)*w_1=2.4$ （$Sum_1=8, Sum_2=7$），r_2 的收入为 $4+\left(Sum_2-Sum_1\right)*w_2=5.6$，不仅系统收益增加，每个人收益也得到提升。

如图 2 所示的是在给定一个由合作发起者产生的联盟 $R^1=\{r_1,r_2,r_3,\cdots,r_n\}$，联盟内成员由竞争状态转变为合作状态的过程。假设 t_1 是一个未获得完成任务全部所需技能的待完成状态，r_1 是合作发起者，不同的颜色代表着不同的选择意向。

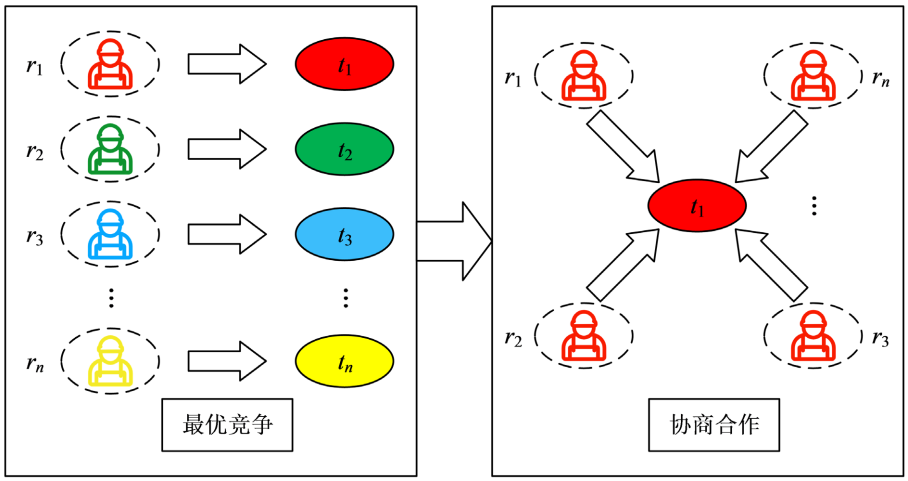

Figure 2. The shift in Market conditions
图 2. 市场状态转变

6. 仿真结果

仿真环境：内存 8 GB；CPU，AMD Ryzen 5 4600U with Radeon；主频，3.6 GHz；操作系统，Win10。

在本节中，将 TSABMW 的运行结果同以下 5 种算法运行结果进行了比较：非自利的普通遗传算法(General Genetic Algorithm, GGA)、VAA 算法(Vig & Adams'Algorithm, VAA) [21]和 SAA 算法(Service and Adams' Algorithm, SAA) [22]，以及具有自利性的组合拍卖算法(Combinatorial Bids based Algorithm, CBA) [23]、有效的虚拟对策算法(Computationally Efficient Sampled Fictitious Play, CESFP) [24]。仿真所用数据集为随机产生和基于 iMPOSE [25] [26]产生。

6.1. 普通遗传算法最优参数的选取

为了获得普通遗传算法的最佳参数，任意生成一组随机数据，生成方法为：$n=30$，$l=15$，$m=30$。用户具有的技能数量为 $random(1,5)$，任务需要的技能个数为 $random(1,5)$。对于该数据集，在不同的交叉概率(crossover probability, CP)和变异概率(mutation probality, MP)下，普通遗传算法运行 30 次，平均系统收益如表 1 所示。从实验结果可以看出，当交叉概率为 0.1 实，变异概率为 0.4 时，普通遗传算法的平均系统收益得到最大值在表 2、表 3、表 4 和表 5 所示实验结果中，普通遗传算法的参数设置均为：交叉概率为 0.1，变异概率为 0.4，种群规模为 100。

6.2. 算法性能比较

从表 2~5 可以看出，在大多数情况下，TSABMW 算法所得的平均系统收益大于其他自利和非自利算法。因为对于计算有效的采样虚拟对策中的用户 agent 只知道自己的历史行动和相应的收益，较其他算法

中用户 agent 所获指导行动信息数量较少，其任务分配的效果相对其他算法来说较差。VAA 算法中对于联盟形成适应性较为精准的把控，是其能在部分数据集中取得较好收益的保障。从以上表中可以看出 TSABMW 算法的运行时间小于 GGA、VAA 和 CESFP 算法，持平于 CBA 和 SAA 算法，其运行时间在多数情况下都是可以接受的。

Table 1. The average system returns of common genetic algorithms
表 1. 普通遗传算法的平均系统收益

MP \ CP	0.01	0.05	0.1	0.15	0.2	0.4	0.6	0.8
0.1	11346	11935	12586	13268	13020	8370	7285	6479
0.2	11439	13113	13423	13640	12493	8091	7223	7037
0.25	12927	12865	14632	14291	12028	8432	7099	6758
0.3	12927	13330	14539	14136	11346	7998	7037	6944
0.35	12555	14136	13795	14291	10912	8370	7285	6789
0.4	12276	12741	**14818**	14539	10385	7874	7099	6572
0.5	12586	13609	14508	14074	10354	7905	7254	6882
0.6	12648	13392	14105	14694	10292	8029	7037	6913
0.8	12772	13981	13702	13888	9765	7936	7068	6851

Table 2. The system revenue and uptime under randomly generated dataset 2
表 2. 随机产生的数据集 2 下的系统收益和运行时间

data set1	TSABMW	time	GGA	time	CBA	time	VAA	time	SAA	time	CESFP	Time
1.1	13826	0.01	14756	6.09	13826	0.01	**16120**	1.11	10912	0.01	7068	0.76
1.2	**15376**	0.01	14849	7.1	11532	0.01	14632	1.08	10974	0.01	6076	0.88
1.3	**15066**	0.01	13578	5.88	14167	0.01	14446	1.17	10075	0.01	9548	0.76
1.4	11935	0.01	13454	5.87	12958	0.01	**15500**	1.1	10385	0.01	9486	0.78
1.5	17391	0.01	**18228**	5.33	17577	0.01	17918	1.05	13051	0.01	11160	0.79
1.6	**20336**	0.01	17484	5.88	12834	0.01	17577	0.99	13578	0.01	10044	0.91
1.7	13671	0.01	13237	5.08	15345	0.01	**15810**	1.04	11904	0.01	11625	0.81
1.8	15624	0.01	13485	5.82	8246	0.01	15252	1.18	14663	0.01	10106	0.88
1.9	**19995**	0.01	15407	5.88	10757	0.01	**17856**	1.06	14663	0.01	9238	0.83
1.1	19313	0.01	15190	5.91	16213	0.01	19189	1.17	15066	0.01	10633	0.89
1.11	**15779**	0.01	15283	5.83	11625	0.01	13795	1.04	14395	0.01	12865	0.89
1.12	15004	0.01	17329	6.75	**18228**	0.01	18166	1.11	12617	0.01	10788	0.86
1.13	**15190**	0.01	13144	5.55	11532	0.01	14942	1.08	11563	0.01	10137	0.84
1.14	14849	0.01	13764	6.16	10385	0.01	**16833**	1.03	10230	0.01	8525	0.91
1.15	**18445**	0.01	16244	6.38	12741	0.01	16399	1.03	13516	0.01	9765	0.81

Table 3. The system revenue and uptime under randomly generated dataset 3
表 3. 随机产生的数据集 3 下的系统收益和运行时间

data set2	TSABMW	time	GGA	time	CBA	time	VAA	time	SAA	time	CESFP	Time
2.1	**108942**	0.01	108741	5.5	90450	0.01	126630	1.05	81606	0.01	73566	0.8
2.2	98088	0.01	974 85	5.63	73164	0.01	**102912**	1.04	73767	0.01	73365	0.84
2.3	**93264**	0.01	89244	6.57	79194	0.01	93063	1.04	63315	0.01	48039	0.79
2.4	**116178**	0.01	99696	6.2	79998	0.01	107736	1.02	95274	0.01	60702	0.82
2.5	91053	0.01	91857	5.54	49848	0.01	**106530**	0.98	81606	0.01	80601	0.87
2.6	**123615**	0.01	106731	6.42	73566	0.01	123012	0.95	100098	0.01	69948	0.8
2.7	**96681**	0.01	74772	5.6	58491	0.01	81003	1.02	80601	0.01	41607	0.79
2.8	88641	0.01	**111756**	5.6	60099	0.01	82410	0.96	77385	0.01	49848	0.87
2.9	**108339**	0.01	93063	6.28	82008	0.01	95877	0.94	72561	0.01	58893	0.85
2.10	79998	0.01	98490	5.97	65727	0.01	**113565**	0.95	83214	0.01	57084	0.94
2.11	**123012**	0.01	119193	5.27	102510	0.01	117987	1.03	90651	0.01	70953	0.89
2.12	100500	0.01	**125022**	6.39	104118	0.01	99093	1.16	76983	0.01	67536	0.87
2.13	87837	0.01	88239	6.51	53466	0.01	**101103**	1.03	83013	0.01	55878	0.81
2.14	85626	0.01	82410	5.84	69144	0.01	**92460**	0.99	78390	0.01	57687	0.84
2.15	**93264**	0.01	91656	5.79	59697	0.01	90852	1.16	50853	0.01	60903	0.78

Table 4. The system revenue and runtime based on the data set 3 generated by iMPOSE
表 4. 基于 iMPOSE 所产生的数据集 3 下的系统收益和运行时间

data set3	TSABMW	time	GGA	Time	CBA	time	VAA	time	SAA	time	CESFP	Time
3.1	49698	0.01	48708	6.88	36531	0.01	**63162**	1.12	41778	0.01	24948	0.87
3.2	57816	0.01	47124	5.59	49797	0.01	**62073**	1	39105	0.01	28908	0.84
3.3	**53361**	0.01	44253	5.88	36828	0.01	49797	1.02	36630	0.01	29601	0.85
3.4	53658	0.01	60489	5.83	37323	0.01	55935	0.92	48312	0.01	32472	0.86
3.5	**58509**	0.01	52470	6.71	48114	0.01	56133	1.04	45441	0.01	33264	0.91
3.6	**51084**	0.01	42966	5.78	30789	0.01	50193	1.05	36927	0.01	29205	0.75
3.7	**56727**	0.01	51777	6.6	43362	0.01	55242	1.1	37917	0.01	31482	0.84
3.8	**49401**	0.01	44748	5.56	35442	0.01	44946	1	36234	0.01	35937	0.78
3.9	**59400**	0.01	57420	6.13	36432	0.01	57321	1.04	47025	0.01	44352	0.85
3.10	**46431**	0.01	45738	6.58	21087	0.01	55539	1.14	40293	0.01	29403	0.84
3.11	**58608**	0.01	56034	6.45	43461	0.01	58212	1.08	45738	0.01	31581	0.76
3.12	33363	0.01	38115	6.37	39006	0.01	**41976**	1.1	38907	0.01	24750	0.79
3.13	43659	0.01	**48411**	6.14	28215	0.01	38808	0.97	34551	0.01	27819	0.83
3.14	**50886**	0.01	49401	5.95	39996	0.01	54945	1.08	41085	0.01	32175	0.83
3.15	**57123**	0.01	53163	6.65	28809	0.01	53262	0.99	42075	0.01	43164	0.79

Table 5. The system revenue and runtime based on the data set 4 generated by iMPOSE
表 5. 基于 iMPOSE 产生的数据集 4 下的系统收益和运行时间

data set4	TSABMW	time	GGA	Time	CBA	time	VAA	time	SAA	time	CESFP	time
4.1	37210	0.01	35380	6.94	**39833**	0.01	39162	1.1	25315	0.01	22570	0.95
4.2	**37149**	0.01	34709	6.29	36783	0.01	36783	1.28	28060	0.01	19947	1.03
4.3	**37820**	0.01	36051	6.49	25132	0.01	37210	1.36	26413	0.01	21960	0.94
4.4	**35807**	0.01	34526	7.49	24034	0.01	34160	1.13	26169	0.01	20618	1.06
4.5	30927	0.01	**36478**	6.7	15799	0.01	33489	1.23	25376	0.01	12627	1.03
4.6	**37149**	0.01	35136	7.24	30378	0.01	36295	1.27	30317	0.01	17812	1.08
4.7	**36661**	0.01	36234	7.62	27389	0.01	35685	1.26	26047	0.01	21594	0.98
4.8	25071	0.01	35502	7.28	27877	0.01	**37149**	1.42	21411	0.01	23119	0.98
4.9	**37149**	0.01	37027	7.35	19764	0.01	35563	1.14	29524	0.01	20496	1.02
4.10	**33794**	0.01	32879	7.13	26230	0.01	32940	1.12	24644	0.01	17995	0.88
4.11	32574	0.01	31903	7.28	22509	0.01	**32879**	1.27	26108	0.01	24278	0.98
4.12	29707	0.01	33611	7.32	26962	0.01	**34892**	1.37	22875	0.01	18910	0.87
4.13	**36234**	0.01	35624	7.26	28792	0.01	32452	1.27	24705	0.01	22997	0.9
4.14	28609	0.01	33611	7.38	19703	0.01	**33062**	1.21	22082	0.01	18544	0.89
4.15	**35197**	0.01	35624	7.12	24156	0.01	31903	1.28	25620	0.01	21533	0.83

7. 总结

本文提出了一种考虑市场权值的二阶段多 agent 任务分配优化算法。市场中的工人是理性的人，他们总是选择能给他们带来最大个人收益并且容易完成的任务。任务请求者的目标完成任务，即获得全部所需技能。对于理性的市场参与者，只有当效益合理分配并且任务分配达到 Nash 均衡时，才能保证任务分配结果的稳定性，但自利性必然影响市场总收入的提高。通过市场权值能让用户自发合作去完成为自己带来更大个体收益的任务。在根据最优反应策略选择任务的过程中，执行顺序将会在一定程度上影响威客的个体收益。根据这一特点，本文提出的基于市场权值的协商阶段，进一步调整了市场中复杂任务分配的效益分配不均衡、选择不合理的问题。最后的仿真结果验证了算法的有效性。未来的研究工作将进一步总结威客任务分配的动态特性，考虑威客的诚信度，技能水平等特性，完善动态任务分配模型。并在此基础上提出满足个体理性、预算有效性、隐私保护性、任务分配结果的稳定性等特性的任务分配算法。

参考文献

[1] Sycara, P.K. (1998) Muti-Agent Systems. *AI Magazine*, **19**, 79-92.

[2] Jennings, N.R. (1995) Controlling Cooperative Problem Solving in Industrial Multi-Agent Systems Using Joint Intentions. *Artificial Intelligence*, **75**, 195-240. https://doi.org/10.1016/0004-3702(94)00020-2

[3] Kraus, S. and Arkin, R.C. (2001) Strategic Negotiation in Multiagent Environments. MIT Press, Cambrigde.

[4] 陈琦. 2020 年 9 月新能源车销售三强:上汽通用五菱、比亚迪、特斯拉[J]. 汽车与配件, 2020(20): 43.

[5] Bachrach, Y., Parkes, D.C. and Rosenschein, J.S. (2013) Computing Cooperative Solution Concepts in Coalitional Skill Games. *Artificial Intelligence*, **204**, 1-21. https://doi.org/10.1016/j.artint.2013.07.005

[6] 蒋建国, 苏兆品, 张国富, 夏娜. 多任务联盟形成中的 Agent 行为策略研究[J]. 控制理论与应用, 2008, 25(5):

853-856.

[7] Cisneros-Cabrera, S., Sampaio, P. and Mehandjiev, N. (2018) A B2B Team Formation Microservice for Collaborative Manufacturing in Industry 4.0. 2018 *IEEE World Congress on Services* (*SERVICES*), San Francisco, 2-7 July 2018, 37-38. https://doi.org/10.1109/SERVICES.2018.00032

[8] Hayano, M., Hamada, D. and Sugawara, T. (2014) Role and Member Selection in Team Formation Using Resource Estimation for Large-Scale Multi-Agent Systems. *Neurocomputing*, **146**, 164-172. https://doi.org/10.1016/j.neucom.2014.04.059

[9] Juárez, J. and Brizuela, C.A. (2018) A Multi-Objective Formulation of the Team Formation Problem in Social Networks: Preliminary Results. *Proceedings of the Genetic and Evolutionary Computation Conference*, Kyoto, 15-19 July 2018, 261-268. https://doi.org/10.1145/3205455.3205634

[10] Zzkarian, A. and Kusiak, A. (1999) Forming Teams: An Analytical Approach. *IIE Transactions*, **31**, 85-97. https://doi.org/10.1080/07408179908969808

[11] Kara, I. and Bektas, T. (2006) Integer Linear Programming Formulations of Multiple Salesman Problems and Its Variations. *European Journal of Operational Research*, **174**, 1449-1458. https://doi.org/10.1016/j.ejor.2005.03.008

[12] Wooldridge, M. and Dunne, P.E. (2004) On the Computational Complexity of Qualitative Coalitional Games. *Artificial Intelligence*, **158**, 27-73. https://doi.org/10.1016/j.artint.2004.04.002

[13] Wooldridge, M. and Dunne, P.E. (2006) On the Computational Complexity of Coalitional Resource Games. *Journal of Artificial Intelligence*, **170**, 835-871. https://doi.org/10.1016/j.artint.2006.03.003

[14] 张国富, 蒋建国, 夏娜, 苏兆品. 基于离散粒子群算法求解复杂联盟生成问题[J]. 电子学报, 2007, 35(2): 323-327.

[15] Larson, K.S. and Sandholm, T.W. (2000) Anytime Coalition Structure Generation: An Average Case Study. *Journal of Experimental & Theoretical Artificial Intelligence*, **12**, 23-42. https://doi.org/10.1080/095281300146290

[16] 苏射雄, 胡山立, 郑盛福, 林超峰, 骆剑彬. 基于势结构的任一时间联盟结构生成算法[J]. 计算机研究与发展, 2008, 45(10): 1756-1762.

[17] 梁军, 程显毅. 基于混合蚁群遗传算法的 agent 联盟求解[J]. 计算机科学, 2009, 36(5): 227-231.

[18] 桂海霞, 张国富, 苏兆品, 蒋建国. 一种基于差分进化和编码修正的重叠联盟结构生成算法[J]. 控制理论与应用, 2018, 35(2): 215-223.

[19] 许波, 余建平. 基于 QPSO 的单任务 Agent 联盟形成[J]. 计算机工程, 2010, 36(19): 168-170.

[20] 尹蕾. 分布式智能系统中多任务协作机制研究[D]: [博士学位论文]. 合肥: 合肥工业大学, 2019.

[21] Vig, L. and Adams, J.A. (2006) Multi-Robot Coalition Formation. *IEEE Transactions on Robotics*, **22**, 637-649. https://doi.org/10.1109/TRO.2006.878948

[22] Service T.C. and Adams J.A. (2011) Coalition Formation for Task Allocation: Theory and Algorithms. *Autonomous Agents and Multi-Agent Systems*, **22**, 225-248. https://doi.org/10.1007/s10458-010-9123-8

[23] Lin, L. and Zheng, Z. (2005) Combinatorial Bids Based Multi-Robot Task Allocation Method. *Proceedings of the* 2005 *IEEE International Conference on Robotics and Automation*, Barcelona, 18-22 April 2005, 1145-1150. https://doi.org/10.1109/ROBOT.2005.1570270

[24] Sisikoglu, E., Epelman, M.A. and Smith, R.L. (2011) A Sampled Fictitious Play Based Learning Algorithm for Infinite Horizon Markov Decision Processes. *Proceedings of the* 2011 *Winter Simulation Conference*, Phoenix, 11-14 December 2011, 4086-4097. https://doi.org/10.1109/WSC.2011.6148098

[25] Myszkowski, P.B., Skowroński, M.E. and Sikora, K. (2015) A New Benchmark Dataset for Multi-Skill Resource-Constrained Project Scheduling Problem. *Proceedings of the* 2015 *Federated Conference on Computer Science and Information Systems* (*FedCSIS*), Łódź, 13-16 September 2015, 129-138. https://doi.org/10.15439/2015F273

[26] Myszkowski, P.B., Skowroński, M.E., Olech, Ł.P., *et al*. (2015) Hybrid Ant Colony Optimization in Solving Multi-Skill Resource-Constrained Project Scheduling Problem. *Soft Computing*, **19**, 3599-3619. https://doi.org/10.1007/s00500-014-1455-x