众包系统复杂任务中预算有限的技能冗余问题

151496 时鹏

摘　要

众包系统是一种新兴的群集智能平台，它借助互联网将任务发布到广泛的环境当中，使得众多的用户能够参与到任务的执行过程中，以此能够借助众人的智慧，以较少的花费，较好地完成任务。但是由于众包用户相对自由、约束较少，在完成任务可靠性上缺乏保证。为了保证任务能够顺利完成，本文提出了一种针对众包系统中复杂任务所需技能的冗余模型，并抽象成一个组合优化问题，该问题被证明是NP完全问题。针对该问题，本文采用了局部搜索、模拟退火和遗传算法的元启发式算法以及深度搜索算法。并通过实验对以上算法进行比较。实验结果表明，模拟退火和遗传算法在得到与深度搜索（得到最优解）相近的结果的情况，花费的时间更少，效果较好。

关键字：众包，复杂任务，技能，冗余

一、 背景

近些年，随着全球化趋势和互联网的普及，传统外包的中雇主和被雇佣者的关系也发生了变化，众包（crowdsourcing）作为一种新兴的平台应运而生。众包是指将一个任务或者问题发布到开放环境中，使其能够被众多的用户所发现，以发挥众人的智慧和能力来解决任务或者问题的过程[1]。相对于外包，众包面向的是更广阔的对象，对被雇佣者的限制也相对较少，因而众包系统中工人数量庞大且价格低廉，减轻了雇主的经济负担，也提高了执行效率。

当前，随着众包的广泛发展，针对不同适用情景，也有了不同演化和创新。众包可以按照任务类型进行分类：众包可分为复杂任务众包和简单任务众包，例如，复杂任务可以是软件开发[2]、产品设计[3]等等，简单任务可以是图片标识[4]、段落翻译[5]等等。复杂与简单之间并没有清晰的分界，主要考虑任务的花费时间和技能需求等方面。复杂任务可以通过任务分解来解决，也有学者提出复杂任务不分解的方法，比如利用用户的社会网络对任务进行分工和协作[6]。

众包系统虽然得到了广泛应用，但是仍然存在一些需要解决的问题，以往的学者主要关注如下两方面的问题。

众包任务特性：众包任务通常相对简单，单个用户需要较少时间即可完成[2]。事实上，现实中也有许多复杂任务，例如任务需要多种技能或者任务工作量大，需要多个人合作完成[7]。在面对较为复杂的任务上，系统也可以做出一定的调整，比如有学者研究如何将任务分解为子任务，如何分配子任务和如何聚合任务等等[8]。众包任务的难度和所需技能是有差异的，而用户技能也具有一定差异性，如何将任务递送给“合适”的用户，从而提高任务完成质量和节约人力资源是一个值得关注的问题[9]。众包任务的预算或者资源通常是有限的，如何有效地分配预算达到花销与效益的平衡也是一个重要的问题[10]。

众包用户特性：众包用户多存在于互联网上，用户与任务发布者间具有较弱的关系制约，因此，对于众包用户的能否如期完成任务、完成任务的质量都很难有保障。另外，还有可能存在“恶意”用户，通过完成大量低质量工作牟利。为了改善用户的可靠性，通常的解决方法是对任务执行者进行冗余[11]，达到发布者要求的再给予报酬；也有学者研究对众包用户建立信誉机制[12]，但是这样的信誉机制可能难以达到，系统中的用户具有短暂特性，因为

用户通常都是有正式工作，利用碎片时间来完成任务；因此，有学者研究通过观测用户过去行为[13]或者利用发布者与用户的交互[14]来保证用户可靠性。

本文主要考虑众包系统中用户具有不可靠的特性，任务类型为复杂任务，即任务需要多种技能，在预算有限情况下，如何通过对用户的进行冗余的雇佣来保证任务完成的可靠性。可以简单举例说明该问题，如一个复杂任务需要 A、B、C 三种技能，同时有一组具有部分技能的工人（以下工人与用户通用），且每个工人都一定的花销，在总预算有限的情况下，如何雇佣工人，使得该任务所需的三种技能都得到满足（即任务可以被完成）。假如雇佣的工人使得 A、B、C 三种技能分别被 3、1、2 个工人所拥有，那么技能 B 看作是一个薄弱环节，因为拥有该技能的工人一旦缺失，就会造成整个任务无法被完成。因此，要考虑在预算一定的情况下，如何对任务进行恰当的冗余，尽量提高薄弱环节，以保证任务完成的可靠性。

二、 相关工作

相关工作我们主要从 3 个领域进行介绍：分别是多机器人（multi-robot）、多智能体系统（multi-agent system）与团队形成（team formation）和众包（crowdsourcing）。

2.1 多机器人领域

多智能体领域通常会考虑机器人团队的健壮性（robustness），因为当一个团队具有冗余性，才能在单个机器人失败后，不影响最终任务的完成。

Liemhetcharat[15]考虑了每个机器人的每个部件的出现损坏的概率，如何组成一个具有健壮性的机器人团队，但是文中没有考虑代价和预算。Okimoto[16]的这篇论文中，团队中的每个机器人可以完成一个集合的任务，目标是选取一个机器人团队来完成目标任务集合。在可允许的代价范围内，“k-健壮”首次被定义，指的是在这个形成的团队中，任意删除 k 个机器人，仍能形成对目标任务集合的覆盖。他把这个问题看做一个双目标约束问题，并提出了一种分支限界法，但该方法复杂度较高，对于数据量庞大、实时性的众包环境不适用。Manisterski[17]假设有一些特殊任务，它包含了许多子任务，且不同的 Agent 有不同的能力来完成不同子任务和花销，他提出一种子任务的分配算法。该文与本文所考虑的众包环境不同，实际情况下，复杂任务通常也难以直接划分。

2.2 多智能体系统与团队形成领域

Lappas[18]和 Li[19]考虑了在社会网络环境下情景，关注如何形成团队能够满足任务所需的技能的覆盖，同时也使得形成团队在社会网络中通信代价最小。而本文中没有考虑社会网络，因为在大多数的众包环境中，用户之间的关联相对较弱，所以暂时没有考虑社会网络。Bachrach[20]考虑任务和 Agent 的技能，只有一个团队的技能覆盖任务所需的技能，任务才能够被完成，但文中没有考虑到技能的冗余性。Marcolino[21]给出了一个具有多样性的团队优于统一性的团队的条件，提出了一种最优的投票策略来形成团体，文中考虑了团队内的合作。Golshan[22]考虑有一个任务集合，其中的每一个任务被完成后都会有一定的收益，现在有在一定预算的情况下，形成一个专家团队，使得该专家团队能完成该任务集合某一子集，该子集所获得的总收益最大。文中考虑的多个任务的情况，而本文只考虑一个任务完成的可靠性。

2.3 众包领域

Karger[23]考虑在在预算有限的情况下，通过任务的冗余分配给众包系统中的用户，来保证任务的可靠性。Venanzi[24]为了保证任务能够顺利的完成，提出了通过建立信用机制来评估众包工人的可靠性，进而保证任务的可靠性。Donmez[25]和 Jung[26]考虑了工人的能力动态变化，通过预测的办法来提高任务的完成质量。本文暂时没有考虑任务的完成质量，仅以任务的技能的覆盖次数来衡量任务的被完成的可靠性。直观上，技能被冗余的次数越多，发生技能缺失导致的任务失败的概率也越低。

三、 问题描述

我们假设有一个复杂任务 a，比如是一个软件开发任务，这个复杂任务需要多种专业的技能。这个任务 a 的技能被表示为一个集合$K_a = \{k_1, k_2, \dots k_m\}$，共需要$m$种技能。该任务假设被公布在众包的平台上，在众包平台上，工人们可以看到这个任务，工人们如果对这个任务感兴趣可以向该任务报名，对该任务报价并提供技能清单。假设有个工人 i 报价为 p_i，并且他所提供的自己所能完成的技能的集合，表示为$S_i = \{s_1, s_2, \dots, s_{m'}\}$。同时，规定任务具有一定的预算$B$，且预算允许雇佣多个工人。

一个复杂任务所需的任意一个技能被看作是完成该任务必不可少的一部分。每个工人不必覆盖所有的技能，但是可以选取多个工人形成一个团队（每个工人只能被选取一次）。如果选取的工人团队（或者工人组合，以下通用）里面，有一个技能一个工人也没有拥有，那么该任务就无法完成。例如，这里有一个开发安卓手机软件的任务，它需要 Java 开发的技能，如果工人团队没有人具有 java 的开发能力，那么这个任务将无法完成。到目前为止，该问题可以看作一个集合覆盖问题，任务的技能被看成集合的元素，工人所具有的技能是这个任务技能的集合的子集。

但是在众包系统中，对于工人具有较弱的约束，工人的工作时间无法得到保障。因此，在预算允许的情况下，我们要对工人进行冗余的雇佣，以保证任务可以顺利完成。因为对于完成任务来说每个技能是必须的，所以我们要尽量保证每个技能都有一定数量的工人来“覆盖”。

为了量化技能冗余的程度，我们定义在所雇佣的工人组合中，如果中间有 3 个工人拥有技能k'，我们就说技能k'的冗余度为 3，用字母 $r_{k'}$表示。因为每个技能对任务来说都是必须的，所以需要尽量保证每个技能都能保证较好的冗余，这里假设任务的每个技能的重要性是相同的。直觉上，冗余度最小的技能可以被看作一个任务完成关键因素。这个可以从木桶效应的角度理解，冗余度最小的技能可以被看作是一个薄弱环节，它最容易缺失从而导致任务无法被完成。所以我们定义任务健壮性（robustness）表示为冗余度最小的技能的冗余度。所以问题就是在满足约束的条件，如何尽可能的增大任务的健壮性，即**最大化最小的技能冗余次数**。问题的数学定义表示如下：

$$\max min(r_1, r_2, \dots, r_m)$$

其中

$$\sum_{i=1}^{n} s_{ij} x_i = r_j$$

$$\text{for j} = 1,2, \dots, m$$

s.t.

$$\sum_{i=1}^{n} p_i x_i \le B$$

我们将该问题称为众包任务的技能冗余问题，其判定问题描述为：

给定一个有限的任务所需的技能集合 U= $\{k_1, k_2, \dots k_m\}$，给定一个有限的工人集合$W = \{w_1, w_2, \dots, w_n\}$，每一个工人$i$ 都有一个技能向量$\mathrm{S_i} = \{s_{i1}, s_{i2}, \dots, s_{im}\}$，其中的元素$s_{ij} \in \{0,1\}$，和价格$p_i \in Z^+$。另给定正整数 B ，表示预算和正整数 R 代表最低技能冗余的最低值。

问题：是否存在一个二进制向量$\mathrm{X} = (x_1, x_2, \dots, x_n)$，其中$x_i \in \{0,1\}$，$\mathrm{i} = 1,2, \dots, \mathrm{n}$，使得满足以下不等式：

$$\sum_{i=1}^{n} p_i x_i \le B$$

$$\sum_{i=1}^{n} s_{ij} x_i \ge R$$

$$\text{for j} = 1,2,\dots,m$$

定理 1：众包任务的技能冗余问题是 NP 完全问题（NP-complete）

为了更好证明，NP 完全的 0-1 背包问题可以规约到众包任务的技能冗余问题，我们先给出 0-1 背包问题的定义如下：

问题：0-1 背包问题判定问题

实例：给定一个有限的物品集合 U= $\{u_1, u_2, \dots, u_n\}$，对于每一个物品$u_i \in U$,$\text{i} = 1,2,\dots,\text{n}$，都有一个重量$\omega_i \in Z^+$和价值$p_i \in Z^+$。另给定正整数$W$ ，表示背包容量和正整数 P 代表总价值 P。

问题：是否存在一个二进制向量$\text{X} = (x_1, x_2, \dots, x_n)$，其中$x_i \in \{0,1\}$，$\text{i} = 1,2,\dots,\text{n}$，使得满足以下不等式(1)：

$$\sum_{i=1}^{n} \omega_i x_i \le W \; and \sum_{i=1}^{n} p_i x_i \ge P \qquad (1)$$

然后我们通过将 0-1 背包问题规约到众包任务的技能冗余问题，来证明该问题为 NP 完全问题。

证明：

首先，显然众包任务的技能冗余问题属于 NP，即可以在多项式时间内判定一个解是否满足该问题的约束。

接下来我们将 0-1 背包问题规约到众包任务的技能冗余问题。我们令所有的工人用二进制向量表示$\text{X} = (x_1, x_2, \dots, x_n)$，其中$x_i \in \{0,1\}$，$\text{i} = 1,2,\dots,\text{n}$

$$x_i = \begin{cases} 1 & \text{工人 } x_i \text{ 被选中} \\ 0 & \text{工人 } x_i \text{ 未选中} \end{cases}$$

所有被选中的工人的工资需要满足总预算B的约束，所以

$$\sum_{i=1}^{n} p_i x_i \le B$$

类似 0-1 背包问题的中不等式(1)，ω_i和W 分别被替换为了 p_i和B.

我们忽略了那些工人拥有但是任务a不需要的技能。任务a所需的技能被表示为集合$K_a = \{k_1, k_2, \dots k_m\}$，为了表示简单，工人的技能集合使用列表 $S_i = \{s_1, s_2, \dots, s_m\}$表示，如果工人 i 拥有技能 k_j，则s_{ij}的值为 1，否则为 0。

$$\sum_{i=1}^{n} s_{ij} x_i \ge R$$

$$\text{for j} = 1,2,\dots,m$$

0-1 背包问题可以看作是众包任务的技能冗余问题的一个实例。如果我们令 0-1 背包问题中的 P 等于众包任务的技能冗余问题的R，但是仅有一个技能是任务所需要的，即$m = 1$。0-1

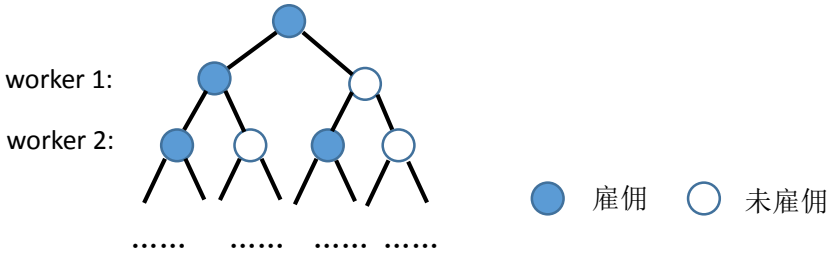
背包问题可以看作是该问题的一维的某种特殊情况。换句话说，如果众包任务的技能冗余问题能够有效的解决那么 0-1 背包问题也将解决。这意味着，本问题要比 0-1 背包问题更难。因此，我们可以得出结论 0-1 背包问题可以规约到本问题。众包任务的技能冗余问题是 NP 完全问题。■

四、 算法设计

众包的技能冗余问题是一个组合优化问题。候选集合是众包工人集合，集合中每个工人提供所拥有的技能集合和工资，在满足预算约束的条件下，雇佣一定数量的工人（不允许重复雇佣），目标是最大化最小的技能冗余数量。问题中，每个工人只有被雇佣和未雇佣两种状态，假设工人的数量为 n，那么所有情况就是 2^n 个，问题的解空间随着数据规模呈现指数爆炸趋势。尤其在众包环境中，常常面临的是大规模的数量的用户。因此，本节先考虑传统的深度搜索法。然后应用元启发式算法，例如基础局部搜索、模拟退化和进化计算。

4.1 深度搜索

深度搜索算法，将解空间考虑成一棵搜索树，每一层代表一个工人，该工人有 2 个状态，即雇佣和未雇佣，因此整个搜索树为高度为 n 的二叉树。如图 1：



本文采用递归的方式来完成搜索树的遍历，算法如下：

算法 1

```
def DeepSearch( i, cur_v, cur_w):
    // cur_w is current selected workers list, cur_v is current cost by selected workers
    // p is the list of every worker's price
        if i >= n:
            //计算当前解的最小的技能冗余值为 min_robust
            if min_robust > best:
                best = min_robust;    best_select_worker = cur_w
            return
        else:
            if cur_v + p[i] <= Budget:
                temp_cur_w = Copy(cur_w) + i // add worker i to the selected workers list
                DeepSearch(i + 1, cur_v + p[i], temp_cur_w)
            temp_cur_w = Copy(cur_w)
            DeepSearch(i + 1, cur_v, temp_cur_w)
```

4.2 基础局部搜索

基础局部搜索也被称为迭代更新，每一步的迭代都向着好于当前解的方向进行更新和迭代。这个算法迭代停止于发现了一个局部最优解。

本算法中，工人总数量为 n，一个解是一个长度为 n 的 0-1 向量，每个元素表示的是每个工人的是否被雇佣，1 表示工人被雇佣，0 表示工人未雇佣。

算法如下：

算法 2

```
s = RandomGenerateSolution()
Repair(s)
while t < timelimited:
    s = Improve(N(s))
    t++
```

N(s)：是求解当前解的邻域函数，本算法中定义一阶邻域指的是取反当前解的一位的得到的解的集合，同时还需要对邻域解进行判断是解是否是可行解。如果不在可行解范围内，则不作为邻域进行考虑。

Improve(N(s))：是从邻域解集中找到一个比当前解要好的解。这里有两个策略：一是找到第一个比当前的解要好的解就返回；二是找到比当前的解要好的解集中最好的一个再返回。我们分别定义这两种策略的邻域搜索为 BLS-First 和 BLS-Opt。适应度函数就是目标函数，即最小的技能冗余值。

4.3 模拟退火

模拟退火算法是一种常见的元启发式算法。如上基础局部搜索算法主要的问题是容易陷入局部最优解，模拟退火算法对其进行改进，增加了一种策略使得解在迭代过程中能够从局部最优解中逃离出来，进而得到更好的解。该策略的基本思想是以一定的概率允许在每步迭代中解朝着更差解的方向移动，这样就有一种可能从当前的局部最优解中跳出来。算法如下：

算法 3

```
S = RandomGenerateSolutionSets()
T= CalT(S)
Best_solution = Select_best(S)
While t < timelimited:
    s ‘= Improve(N(Best_solution))
    if Fitness(s’) > Fiteness(Best_solution):
        Best_solution = s’
    else:
        if Random() < p( T, Best_solution, s’ ):
            Best_solution = s’
    Update(T)
```

RandomGenerateSolutionSets()：是随机产生一个解集合，其结果主要用于选择产生一个较优的初始解，另外用于计算温度值。

因为我们求解的问题是最大化问题，所以概率计算方式为$P(T,s,s')=exp^{(-\frac{fitness(s)-fitness(s')}{T})}$，$s'$是新产生的解，s是当前解，如果新产生的解比当前解s要差，即$fitness(s)-fitness(s')>0$，则按照概率P更新当前解为新产生的解s'。可以看出，当新产生的解s'与当前解的差距越大，更新的概率越小。T为温度参数，它在迭代过程中递减，较差解更新当前的解的概率也越小。这样做的目的是使得在刚开始的搜索过程更趋多样性，在后来的搜索过程更趋深入性。

Update(T)：是更新温度的函数，由于温度参数在搜索过程中逐渐递减，所以温度参数的更新方式为$T'=\alpha\cdot T$，其中$\alpha\in[0,1]$。

4.4 遗传算法

以上两种启发式算法（基础局部搜索和模拟退火算法）在每一次迭代过程中都只有一个解，综合起来看，就像是点的轨迹。因此这两种方法也被称为轨迹法。而遗传算法是从生物

进化中得到启发，每步迭代过程中，不再只是一个解，而是多个解类似于一个种群。算法如下：

算法 4

```
S = RandomGenerateSolutionSets()
While t < timelimited:
    SS = {}
    for i from 1 to |S|/2:
        parent1 = RandomSelect(S)
        parent2 = RandomSelect(S)
        newItem1, new Item2 = Crossover(parent1, parent2)
        if Random() < p:
            mutation(newItem1)
            mutation(newItem2)
        SS.append(newItem1)
        SS.append(newItem2)
    S = SelectBestSet(S,SS)
```

由于本文中问题采用长度为n的 0-1 的向量表示一组解，即 1 表示工人被雇佣，0 表示未雇佣，所以这样的 0-1 向量适合直接应用遗传算法。

RandomGenerateSolutionSets()：该方法可以随机产生一个解的种群，内部可以结合修补函数和基础局部搜索，来优化初始解。

Crossover()：交叉函数，是由两个亲代解进行交换，产生两个子代解。

Mutation()：变异函数，以一定的概率进行变异，对解的某一个元素进行取反操作。

SelectBestSet(S,SS)：该函数是从新产生的种群和原始种群中，按照适应度值优劣，选择最优的前 k 个作为下一次迭代的初始种群。 k 为种群大小。

五、 实验

本实验主要从算法的计算结果和算法的消耗时间两个角度进行对比。每组实验随机产生 10 个任务，每个任务每种算法重复运行 10 次取平均。测试程序使用 python 实现，运行环境 CPU：Intel i3 380M，内存 8G。代码参见：
https://github.com/shipengAlan/RobustCrowdsourcing.git

表 5-1

对比算法				
局部搜索（邻居最优）	局部搜索（邻域首优）	模拟退火	遗传算法	深度搜索
BILOpt	BILFirst	SA	EC	DS

5.1 比较算法结果

随机产生 10 个工人，比较四种启发式算法运行 0.01s、0.05s 和 0.1s 后的得出的平均结果，即任务健壮性（最小技能冗余数），深度搜索算法得到的是最优解。结果分别如图 5-1、图 5-2 和图 5-3。

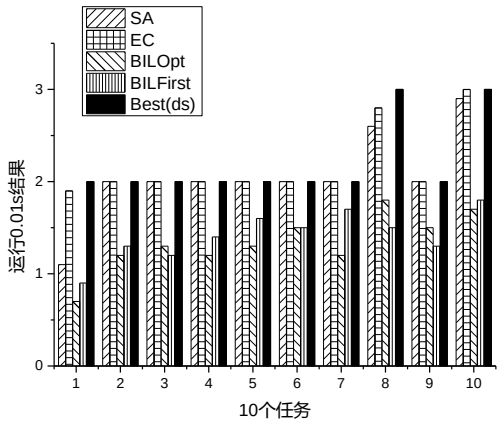


图 5-1

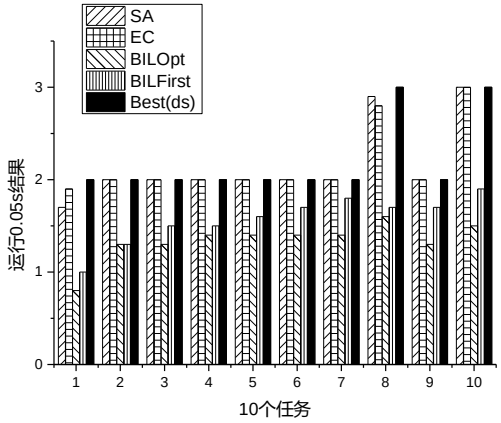


图 5-2

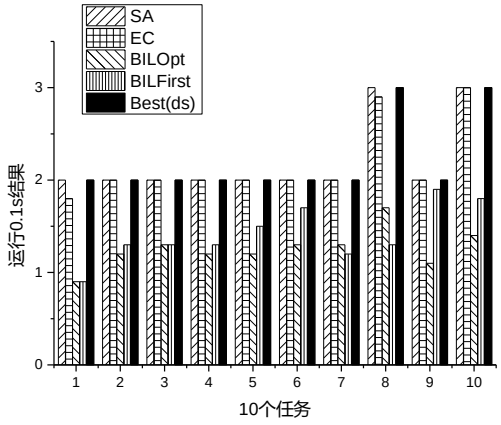


图 5-3

可以看出遗传算法（EC）和模拟退火（SA）效果较好，甚至模拟退火在运行 0.1s 后，达到了最优解。表 5-2 是最优算法的结果和执行时间（秒）。

表 5-2

结果	2	2	2	2	2	2	2	3	2	3
时间	0.0407	0.0354	0.0453	0.0429	0.0307	0.0467	0.0352	0.0448	0.0419	0.0444

可以看出在较少的数据规模的情况下，遗传算法（EC）和模拟退火（SA）算法在相近的时间内，能得到近似于最优解的结果。

5.2 比较算法性能

我们增加工人数量至 20 个，一共有 10 个任务，每个任务每个算法计算 10 遍取平均，然后将这 10 个任务的结果求和。分别比较算法在 0.01s、0.1s、1s、2s、3s 下的结果。如图 5-4。

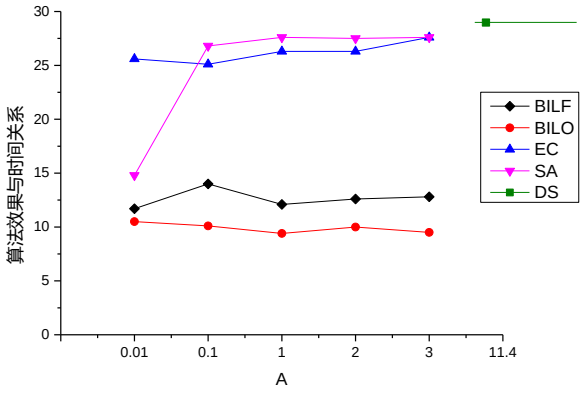


图 5-4

如图 5-4 可以看出，在 1-3s 时，模拟退火和遗传算法就已经接近于深度搜索算法得到的最优结果，且花费时间更短。我们知道深度搜索算法的复杂为$O(2^n)$，其在工人数量众多的情况下，计算代价会呈指数式增长，不适合应用于众包环境中。

另外，我们在实验中发现，温度的选取对模拟退火算法的效果至关重要，如果温度选取较高，在较短的时间内，其结果不一定较好（原因是此时仍有较大概率接受较差解，迭代处于探索阶段），当时间足够时，就会得到较好的结果。如图 5-4 可以清楚的看到这个现象。

六、 总结

由于众包系统的特性导致系统中的工人具有不可靠性，本文考虑了众包系统的中复杂任务，其需要多种技能，而每一种技能对于技能的完成来说都是不可或缺的，这就需要考虑复杂任务的技能的冗余。因为每一个工人能够提供多种技能，在预算有限的情况下，如何雇佣工人才能保证任务的在各个技能的冗余比较均衡，本文提出了最大化最小的技能冗余数的目标，并将该问题抽象成组合优化问题，并证明该问题属于 NP 完全问题。应用了基础局部搜索、模拟退火和遗传算法等启发式算法来求解该问题。通过实验和对比发现，模拟退火和遗传算法在得到与深度搜索（得到最优解）相近的结果的情况，花费的时间更少。

七、 参考文献

[1] J.Howe, “The Rise of Crowdsourcing, Wired Magazine”, 14(6):1-4, 2006

[2] Tran-Thanh, Long, et al. "Efficient crowdsourcing of unknown experts using multi-armed bandits." European Conference on Artificial Intelligence. 2012.

[3] https://www.threadless.com/

[4] http://zoo1.galaxyzoo.org/

[5] Goto, Shinsuke, Donghui Lin, and Toru Ishida. "Crowdsourcing for Evaluating Machine Translation Quality." Proceedings of the 9th International Conference on Lan-guage Resources and Evaluation (LREC-14), 2014.

[6] Chilton, Lydia B., et al. "Frenzy: A Platform for Friendsourcing." First AAAI Conference on Human Computation and Crowdsourcing. 2013.

[7] Kittur, Aniket, et al. "Crowdforge: Crowdsourcing complex work." Proceedings of the 24th annual ACM symposium on User interface software and technology. ACM, 2011.
[8] Tran-Thanh, Long, et al. "Budgetfix: budget limited crowdsourcing for interdependent task allocation with quality guarantees." Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems. International Foundation for Autonomous Agents and Multiagent Systems, 2014.
[9] Bragg, Jonathan, Andrey Kolobov, and Daniel S. Weld. "Parallel Task Routing for Crowdsourcing." Second AAAI Conference on Human Computation and Crowdsourcing. 2014.
[10] Tran-Thanh, Long, et al. "Efficient budget allocation with accuracy guarantees for crowdsourcing classification tasks." Proceedings of the 2013 international conference on autonomous agents and multi-agent systems. International Foundation for Autonomous Agents and Multiagent Systems, 2013.
[11] Karger, David R., Sewoong Oh, and Devavrat Shah. "Budget-optimal task allocation for reliable crowdsourcing systems." Operations Research 62.1 (2014): 1-24.
[12] Venanzi, Matteo, Alex Rogers, and Nicholas R. Jennings. "Trust-based fusion of untrustworthy information in crowdsourcing applications." Proceedings of the 2013 international conference on autonomous agents and multi-agent systems. International Foundation for Autonomous Agents and Multiagent Systems, 2013.
[13] Yu, Han, et al. "A reputation-aware decision-making approach for improving the efficiency of crowdsourcing systems." Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems. International Foundation for Autonomous Agents and Multiagent Systems, 2013.
[14] Zhang, Yu, and Mihaela van der Schaar. "Reputation-based incentive protocols in crowdsourcing applications." INFOCOM, 2012 Proceedings IEEE. IEEE, 2012.
[15] Liemhetcharat, Somchaya, and Marco Veloso. "Forming an effective multi-robot team robust to failures." Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on. IEEE, 2013.
[16] Okimoto, Tenda, et al. "How to Form a Task-Oriented Robust Team." Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems. International Foundation for Autonomous Agents and Multiagent Systems, 2015.
[17] Manisterski, Efrat, et al. "Forming efficient agent groups for completing complex tasks." Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems. ACM, 2006.
[18] Lappas, Theodoros, Kun Liu, and Evimaria Terzi. "Finding a team of experts in social networks." Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2009.
[19] Li, Cheng-Te, Man-Kwan Shan, and Shou-De Lin. "On team formation with expertise query in collaborative social networks." Knowledge and Information Systems 42.2 (2013): 441-463.
[20] Bachrach, Yoram, and Jeffrey S. Rosenschein. "Coalitional skill games." Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2. International Foundation for Autonomous Agents and Multiagent Systems, 2008.
[21] Marcolino, Leandro Soriano, Albert Xin Jiang, and Milind Tambe. "Multi-agent team formation: diversity beats strength?." Proceedings of the Twenty-Third international joint conference on Artificial Intelligence. AAAI Press, 2013.

[22] Golshan, Behzad, Theodoros Lappas, and Evimaria Terzi. "Profit-maximizing cluster hires." Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2014.
[23] Karger, David R., Sewoong Oh, and Devavrat Shah. "Budget-optimal task allocation for reliable crowdsourcing systems." Operations Research 62.1 (2014): 1-24.
[24] Venanzi, Matteo, Alex Rogers, and Nicholas R. Jennings. "Trust-based fusion of untrustworthy information in crowdsourcing applications." Proceedings of the 2013 international conference on autonomous agents and multi-agent systems. International Foundation for Autonomous Agents and Multiagent Systems, 2013.
[25] Donmez, Pinar, Jaime G. Carbonell, and Jeff G. Schneider. "A Probabilistic Framework to Learn from Multiple Annotators with Time-Varying Accuracy." SDM. Vol. 2. 2010.
[26] Jung, Hyun Joon, Yubin Park, and Matthew Lease. "Predicting Next Label Quality: A Time-Series Model of Crowdwork." Second AAAI Conference on Human Computation and Crowdsourcing. 2014.