运筹学与控制论

DOI:10.11721/cqnuj20210417

基于 PSO 的载荷受限的灾后救援任务分配问题研究*

高　凯[1,2]，耿　娜[1]，张馨月[1]，Quang A. Nguyen[3]
(1. 江苏师范大学 电气工程及自动化学院，江苏 徐州 221116；2. 西北工业大学 自动化学院，西安 710129；
3. The Data Driven Research and Innovation Programme，Centre for Intelligent Healthcare，Coventry University，Coventry CV4 7AL，UK)

摘要:【目的】针对灾后救援问题，研究多直升机多任务含约束灾后救援任务分配问题，给出基于微粒群优化的问题求解方法，以获取各直升机的救援分配方案。考虑救援时间和直升机载荷有限，期望在有限的时间内救援最多的被困人员。【方法】首先，针对问题特性，考虑上述约束和目标，建立问题的数学模型；其次，采用改进的微粒群算法对所建模型进行求解，主要包括微粒的解码方法、微粒的位置和速度更新公式、全局极值更新方法以及局部搜索方法等。【结果】采用所提方法对不同的救援场景进行求解，并与自适应大规模邻域搜索算法进行了比较。【结论】仿真结果验证了所提方法的有效性。

关键词:灾后救援；载荷受限；直升机；任务分配；微粒群算法
中图分类号:O224；TP301　　**文献标志码**:A　　**文章编号**:1672-6693(2021)04-0010-11

灾后救援一直受到国际社会的关注，及时有效的救援工作成为社会关注的焦点。由于受灾原因不同，灾后救援现场的情况也不尽相同，这大大增加了制订救援方案和规划救援路线的难度；进一步，当受灾区域较大且地形环境复杂时，若被困人员分布分散，陆地交通工具及救援人员可能无法及时赶到救援现场，势必会出现被困人员伤亡的情况；尤其是受各种不确定性因素的影响，很有可能再次发生灾难，对救援人员和被困人员的生命安全造成极大的威胁。

考虑到机器人可以有效代替人类深入灾区完成救援任务，采用无人机和直升机来实施救援必然成为一种趋势。目前，无人机技术日渐成熟，因为体积小、成本低等优点，可以很好地实施前期侦察任务[1]，从而探测到被困人员的具体位置和生命体征，以协助制定救援方案，规划救援路线。此外，直升机可以有效协助无人机来完成救援任务，救援直升机能够在恶劣环境下执行飞行任务，并搭载完善的医疗救援设施，从而对被困人员及时实施医疗救助，降低伤亡率[2]。无人机前期侦察技术已经非常成熟，因此，本研究在无人机获取灾后环节信息的基础上，采用直升机来调度救援，保证救援任务快速有效的实施。

直升机在搭载医疗设备时存在载荷受限，且灾后环境中被困人员的存活时间也有限，因此，采用直升机进行救援的关键在于救援任务分配方案的制定[3]，以期在有限的时间和资源内完成尽可能多的救援。一个较优的救援任务分配方案，对于减少被困人员的伤亡率，保证救援人员的安全有着重要的实际意义。

救援任务分配方案实际上为资源调配问题，本质上为一个组合优化问题，多架救援直升机只有合理分配好相应的救援任务，达到资源的最优化利用，才能在有限的时间内救援尽可能多的被困人员，减少伤亡损失。目前国内外对于任务分配的研究方法主要有两类。一类是基于市场法的优化方法[4]；一类是智能优化方法，如微粒群优化算法(Particle swarm optimization，PSO)[5]、遗传算法(Genetic algorithm，GA)[6]、蚁群算法(Ant colony optimization，ACO)[7]、模拟退火算法(Simulate anneal，SA)[8]、自适应大规模邻域搜索算法(Adaptive large neighborhood search，ALNS)[9]等。其中，基于市场法算法可以对问题进行求解，但是该方法不能收敛到全局最优解；智能优化算法中，微粒群算法因参数较少、收敛快和精度高而易于实现，是目前应用较为广泛的群体智能优化算法。

经过多年的发展，PSO 优化方法不断完善，取得了许多成果。例如，王雅琳等人[10]针对任务分配问题提出

* **收稿日期**:2020-11-14　**网络出版时间**:2021-07-07　07:52
资助项目:国家自然科学基金 (No. 61703188；No. 61873105)
第一作者简介:高凯，男，研究方向为机器人灾后救援，E-mail:kaigao@mail.nwpu.edu.cn；**通信作者**:耿娜，女，讲师，博士，E-mail:gengna@jsnu.edu.cn
网络出版地址:https://kns.cnki.net/kcms/detail/50.1165.N.20210706.1426.004.html

了一种离散微粒群优化方法，通过引入反正切函数对微粒的位置进行处理，实现了连续数到整数的转换。张峰[11]将 PSO 算法应用于进港航班排序问题，较好地提升了管制运行的效率。万婧[12]针对可重入作业车间调度中传统微粒群算法容易陷入局部极值的问题，提出了一种混合离散 PSO 算法，通过引入变异机制，有效地解决了该问题。Nie 等人[13]针对机器人路径规划问题，提出了一种基于改进的非线性惯性权重系数的微粒群优化算法。此外，考虑到多模态优化问题中，传统 PSO 算法容易出现重复解和最优值遗漏的问题，Li 等人[14]提出了一种基于连续生态技术的微粒群改进算法（BNPSO），对目标函数进行自适应修正，有效解决了多路径规划问题。而 Seo 等人[15]给出了多分组微粒群优化算法（MGPSO）的概念，每个分组的微粒独立搜索最佳解决方案，为避免解决方案重叠，引入排斥力速度的概念，防止每组最优解被其他组入侵。

PSO 算法用于任务分配的成果虽然不多，但取得了很好的成效。例如，翟文正等人[16]为解决异构多核环境下相关任务高效调度的难题，提出一种面向 DAG 任务模型的调度算法。卜艳萍等人[17]对离散微粒群算法进行了改进，使之适用于网格任务调度问题，实现网格资源的优化分配。郭研等人[18]提出了一种基于云多目标的微粒群算法，可有效地解决多项目调度问题。Geng 等人[19]针对灾后搜救问题，考虑多任务类型多机器人任务分配问题，给出了基于 PSO 的集中式任务分配求解方法，有效解决了上述问题；Lin 等人[20]考虑灾后某些被困人员可能存在分布相对密集的情况，给出了基于分组方法的改进 PSO 方法，多种灾后救援场景验证了所提方法的有效性。

基于上述分析，本文考虑采用 PSO 算法来求解救援任务分配问题。首先，考虑直升机载荷受限和时间约束，以成功救援的任务个数为目标函数，建立该问题的数学模型；其次，采用微粒群优化方法求解已建立的数学模型，主要包括：微粒的解码方法、微粒更新公式、全局极值更新策略以及局部搜索方法等；最后，针对不同的救援场景进行仿真，验证所提方法的有效性。

1 问题描述和数学模型

1.1 问题描述

为了简化问题求解，不失一般性，现作出以下假设：

1）直升机的油量能够保证完成救援任务；

2）直升机在救援过程中的飞行速度恒定，且每个被困人员的救援时间均相同，记为 t_{re}；

3）救援前采用无人机对所有被困人员进行生命体征探测，并获取每个被困人员的位置和存活时间；

4）在整个搜救过程中，所有被困人员的位置保持不变。

基于上述假设，再假设在某一灾后环境中，分散着 N 个受伤程度不同的被困人员，且他们的位置已知，表示为 $\boldsymbol{X}_S=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$；$M$ 架直升机对 N 个被困人员实施救援，且 $N>M$，即 1 架直升机需要对多个被困人员实施救援；每架直升机有载荷限制，记为 ΔW，一旦满员需立即返航，待送回起始点后方可继续救援；由于救援环境恶劣，每个被困人员的存活时间有限，要求直升机在有限时间内完成最大的救援任务个数；如果被困人员在存活时间之内被救起，则认为救援任务是成功的，否则，救援任务失败。因此，对 M 架直升机制定合理有效的任务分配方案是保证成功救援被困人员数最多的关键。

综上所述，本文求解的问题可以描述为：在有限的时间和载荷限制的约束下，采用直升机来完成对被困人员的救援任务，拟对每个直升机的救援任务进行分配，期望成功完成的任务个数最多。

救援示意图见图 1，黑色圆圈表示任务，菱形表示直升机，折线表示直升机的救援任务序列，以直升机Ⅰ的救援路径为例，救援顺序为 14→8→9→7→2→I→1，其中：I 表示直升机Ⅰ在执行完任务 2 后，达到它的最大载荷，返航回到起始点，然后再继续执行救援任务，完成对任务 1 的救援。

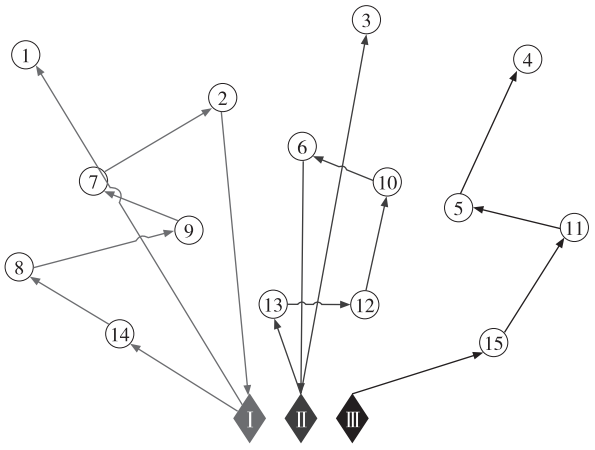

图 1　多机器人多任务救援示意图

Fig. 1　Rescue schematic of multi-helicopter multi-task

1.2 数学模型构建

假设所有直升机的初始位置均为 $P_H(X_0,Y_0)$，直升机飞行速度设为恒定值 v_h。建立一个救援任务分配

矩阵 $\boldsymbol{A}_{(M*\mu)}$,表示 M 架直升机的救援序列,其中 $\mu=\mathrm{ceil}\left(\frac{N}{M}\right)$,表示将 N 个救援任务较为均等地分配给 M 架直升机。由于可能存在 N 个任务无法平均分配给 M 架直升机的情况,所以矩阵 $\boldsymbol{A}$ 中的元素可能存在空缺位置,为了保证矩阵 $\boldsymbol{A}$ 的完整性,本文将 $\boldsymbol{A}$ 中空缺的位置补0,从而构成一个完整的任务分配矩阵 $\boldsymbol{A}$,表示为:

$$\boldsymbol{A}=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1j} & \cdots & a_{1\mu} \\ a_{21} & a_{22} & \cdots & a_{2j} & \cdots & a_{2\mu} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{ij} & \cdots & a_{i\mu} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ a_{M1} & a_{M2} & \cdots & a_{Mj} & \cdots & a_{M\mu} \end{bmatrix}。$$

其中:$1\leqslant i\leqslant M$,$1\leqslant j\leqslant \mu$,第 i 行的元素表示第 i 架直升机需要救援的被困人员序列,救援顺序是从第1列到第 μ 列:$(a_{i1}\rightarrow a_{i2}\rightarrow\cdots\rightarrow a_{ij}\rightarrow\cdots\rightarrow a_{i\mu})$。

假设第 i 架直升机的飞行时间(从起飞开始计算)为 T_i,第 j 位被困人员的生存时间为 t_j。从矩阵 $\boldsymbol{A}$ 第1行第1列的元素开始计算直升机的飞行时间 T_i,t 表示两次救援任务之间所用时间,且 $t=\frac{|a_{ij}a_{i(j-1)}|}{v}$,其中:$i=1,2,\cdots,M$;$j=1,2,\cdots,N$;$a_{i0}$ 表示直升机的起始点。

那么,直升机从起始点到任务点所用的时间为 $T_i=T_i+t+t_{\mathrm{re}}$,如果直升机到达任务 j 所在位置时,t_j 大于 T_i,表示救援成功,救援成功数 F_i 加1;否则,救援失败,F_i 保持不变。那么,$F_i=\sum\left\lfloor\frac{\mathrm{sgn}(t_j-T_i)+1}{2}\right\rfloor$,其中:$\lfloor\cdot\rfloor$表示向下取整函数。

若某一直升机成功救援一名被困人员后,则该架直升机的任务量 W_i 加1。若 $W_i=\Delta W$,表示直升机达到最大载荷,需立即返航,此时 W_i 清零。因此,W_i 需要根据 F_i 的值进行更新,即 $W_i^*=W_i\times\mathrm{sgn}(\Delta W-W_i)$,其中:

$$W_i=\Delta W\times\left[\frac{F_i}{\Delta W}-\left\lfloor\frac{F_i}{\Delta W+1}\right\rfloor\right]。$$

返航后,直升机飞行时间 T_i 需加上返航时间 $t'=\frac{|a_{ij}a_{i0}|}{v}$,则直升机所用时间 $T_i=T_i+t'$。

基于以上分析,本文所要求的救援任务分配问题可以建模如下:

$$\max\ \sum_{i=1}^{M}F_i, \tag{1}$$

$$\text{s.t.}\ \ t_j>T_{\mathrm{i}}, \tag{2}$$

$$a_{ij}\in N, \tag{3}$$

$$0\leqslant a_{ij}\leqslant N, \tag{4}$$

$$A_i\bigcap A_j=\varnothing。 \tag{5}$$

其中:(1)式表示 M 个直升机的成功救援的任务个数;(2)式表示任务成功救援的前提是被困人员的存活时间要大于直升机的到达时间;(3),(4)式表示对变量的约束,均为整数,且任务的编码要小于等于任务的总个数,0表示直升机的起始点;(5)式表示任意两个不同的直升机执行的救援序列没有重复的任务。

2 微粒群求解方法

2.1 PSO 算法

微粒群优化方法是一种基于种群的随机优化技术,1995年由 Eberhart 和 Kennedy 博士共同提出[21]。算法的灵感来源于一群觅食的飞鸟,所有飞鸟都不知道食物的具体位置,但可以判断自己与食物之间的距离。整个鸟群可以共享各自与食物之间的距离信息,从而向距离食物最近的鸟的周边区域进行搜寻,最终找到食物。在PSO算法中,所求优化问题的解对应一只觅食的飞鸟,称之为"微粒"。每一个微粒都有它的位置 X_i 和速度 V_i,通过适应度函数进行计算得到一个适应值,也即每一只飞鸟与食物之间的距离。所以每个微粒都知道自己当前最优位置,也即个体极值,表示为 $\boldsymbol{p}_{\mathrm{best}}$,以及种群中所有微粒的最优位置,也即全局极值,表示为 $\boldsymbol{g}_{\mathrm{best}}$。通过不断

的迭代计算，种群中的微粒可以依据个体极值 $\boldsymbol{p}_{\text{best}}$ 和种群的全局极值 $\boldsymbol{g}_{\text{best}}$ 进行判断，调整自己的方向，在下一次迭代中朝着更好的方向飞行，经过多次迭代，整个种群可以收敛到一个最优解，即找到飞鸟的“食物”。

2.2 微粒编码和解码方法

由于任务分配问题的变量多为离散型，本文考虑到标准 PSO 算法的速度和位置更新过程为实数形式，而任务分配问题的适应度计算为整数形式，所以提出采用实数编码并解码成整数的方法来解决灾后救援任务分配问题。因此，微粒编码为实数，而解码后的整数序列则为分配的救援任务序列，用来计算微粒个体的适应值。

假设任一微粒表示为 $\boldsymbol{X}_i=[x_{i1}\quad x_{i2}\quad x_{i3}\quad\cdots\quad x_{ij}\quad\cdots\quad x_{iN}]$，对 $\boldsymbol{X}_i$ 进行解码后得到救援分配矩阵 $\boldsymbol{A}_i$，解码过程分为两个部分：首先，对 $\boldsymbol{X}_i$ 中的实数进行从小到大排序，并返回元素位置的索引，从而得到一个整数矩阵：$\boldsymbol{X}_i'=[x_{i1}'\quad x_{i2}'\quad x_{i3}'\quad\cdots\quad x_{ij}'\quad\cdots\quad x_{iN}']$；然后，再对 $\boldsymbol{X}_i$ 进行分组，M 个元素为一组，则 $\boldsymbol{X}_i'$ 可以分成：

$$[x_1'\quad x_2'\quad\cdots\quad x_M'],[x_{M+1}'\quad x_{M+2}'\quad\cdots\quad x_{2M}'],\cdots。$$

每组中数据分别赋值矩阵 $\boldsymbol{A}_i$ 的每一列，如 $a_{11}=x_1'$，$a_{21}=x_2'$，$\cdots$，$a_{M1}=x_M'$。若矩阵中的元素个数大于救援序列中的元素个数，那么未被赋值的元素置 0，从而可以得到一个完整的矩阵 $\boldsymbol{A}_i$。

下面举例说明本文所提编解码方法。假设直升机数量 $M=3$，被困人员数量 $N=10$，被困人员用 1～10 进行标号，假设任一微粒 $\boldsymbol{X}_i$ 表示为：

$$\boldsymbol{X}_i=[3.74\quad 1.21\quad 5.21\quad 8.45\quad 1.93\quad 4.05\quad 9.11\quad 5.99\quad 6.78\quad 9.99]。$$

首先，对 $\boldsymbol{X}_i$ 中的实数进行排序，返回序列索引，得到整数矩阵 $\boldsymbol{X}_i'=[2\quad 5\quad 1\quad 6\quad 3\quad 8\quad 9\quad 4\quad 7\quad 10]$。以 3 个数为一组，分别为 $[2\quad 5\quad 1]$，$[6\quad 3\quad 8]$，$[9\quad 4\quad 7]$，$[10]$，按照上述方法进行分配，救援分配矩阵为 $\boldsymbol{A}_i=\begin{bmatrix}2&6&9&10\\5&3&4&0\\1&8&7&0\end{bmatrix}$。

针对微粒中实数重复出现的问题，也即出现不可行解的情况，对排序完后的救援序列 $\boldsymbol{X}_i'$ 进行检查，若出现重复元素，则使用更新之前的 $\boldsymbol{X}_i'$。

本文在研究的问题中考虑直升机容量和任务时间的约束，一般地，被困人员距离直升机越近越先得到救援，被困人员的存活时间越短越先得到救援，两种策略在一定程度上都能保证直升机救援的任务个数尽可能的多，基于此，本文的编码综合考虑距离和时间，设定直升机救援的优先目标。

具体地，首先计算每个任务距离每架直升机之间的距离，表示为 d_{ij}，且 $d_{ij}=|P_Ha_{ij}|$。由此，可以得到距离矩阵为：$\boldsymbol{D}=\begin{bmatrix}d_{11}&d_{12}&\cdots&d_{1j}&\cdots&d_{1N}\\d_{21}&d_{22}&\cdots&d_{2j}&\cdots&d_{2N}\\\vdots&\vdots&\cdots&\vdots&\cdots&\vdots\\d_{i1}&d_{i2}&\cdots&d_{ij}&\cdots&d_{iN}\\\vdots&\vdots&\cdots&\vdots&\cdots&\vdots\\d_{M1}&d_{M2}&\cdots&d_{Mj}&\cdots&d_{MN}\end{bmatrix}$。

接着，计算直升机 i 首先救援被困人员 j 所需要的时间，并计算所需时间与被困人员的存活时间之差，表示为 Δ_{ij}，且 $\Delta_{ij}=t_j-\dfrac{d_{ij}}{v_r}$，由此，可以得到一个时间差矩阵为：$\boldsymbol{\Delta}=\begin{bmatrix}\Delta_{11}&\Delta_{12}&\cdots&\Delta_{1j}&\cdots&\Delta_{1N}\\\Delta_{21}&\Delta_{22}&\cdots&\Delta_{2j}&\cdots&\Delta_{2N}\\\vdots&\vdots&\cdots&\vdots&\cdots&\vdots\\\Delta_{i1}&\Delta_{i2}&\cdots&\Delta_{ij}&\cdots&\Delta_{iN}\\\vdots&\vdots&\cdots&\vdots&\cdots&\vdots\\\Delta_{M1}&\Delta_{M2}&\cdots&\Delta_{Mj}&\cdots&\Delta_{MN}\end{bmatrix}$。

检测矩阵中元素 Δ_{ij} 有无小于 0 的情况，如果存在 $\Delta_{ij}<0$ 的情况，则表示即使第 i 架直升机最先对被困人员 j 实施救援，依然不能在被困人员 j 的存活时间内到达，此时，设定 $\Delta_{ij}=\infty$，表示直升机 i 无法完成对被困人员 j 的救援，也即救援被困人员 j 对于直升机 i 是一个不可行的任务。

进一步，针对距离矩阵和时间差矩阵的每一行，即针对每一架直升机，按照从小到大的顺序进行排列，并返

回序列索引,则每个矩阵都可以得到一个整数序列,然后将得到的两个矩阵相加,得一个优先级矩阵:$\boldsymbol{P}_R=$

$$\begin{bmatrix} p_{r_{11}} & p_{r_{12}} & \cdots & p_{r_{1j}} & \cdots & p_{r_{1N}} \\ p_{r_{21}} & p_{r_{22}} & \cdots & p_{r_{2j}} & \cdots & p_{r_{2N}} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ p_{r_{i1}} & p_{r_{i2}} & \cdots & p_{r_{ij}} & \cdots & p_{r_{iN}} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ p_{r_{M1}} & p_{r_{M2}} & \cdots & p_{r_{Mj}} & \cdots & p_{r_{MN}} \end{bmatrix}$$

,其中:$p_{r_{ij}}$ 表示直升机 i 对任务 j 救援的优先级别,值越小越先得到救援。

上述优先级矩阵是由距离和时间差相加所得,本文两者的权重设置为1。由于距离和时间差加法权重值的设定并不容易,无法准确设定,考虑上述问题,将优先级矩阵进行分级,根据 $p_{r_{ij}}$ 的取值,分成3个优先等级,每个等级具体取值为:任务 j 等级$=\begin{cases}1, p_{r_{ij}}\in[2,N]\\ 2, p_{r_{ij}}\in(N,2N]\\ 3, p_{r_{ij}}=\infty\end{cases}$。

基于任务等级,可以将优先矩阵分成3级,表示如下:

$$\boldsymbol{P}_R=\begin{bmatrix} p_{r_{11}} & p_{r_{12}} & \cdots & p_{r_{1j}} & \cdots & p_{r_{1N}} \\ p_{r_{21}} & p_{r_{22}} & \cdots & p_{r_{2j}} & \cdots & p_{r_{2N}} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ p_{r_{i1}} & p_{r_{i2}} & \cdots & p_{r_{ij}} & \cdots & p_{r_{iN}} \\ \vdots & \vdots & \cdots & \vdots & \cdots & \vdots \\ p_{r_{M1}} & p_{r_{M2}} & \cdots & p_{r_{Mj}} & \cdots & p_{r_{MN}} \end{bmatrix}。$$

等级1 等级2 等级3

根据排序后的 $\boldsymbol{P}_R$ 对矩阵 $\boldsymbol{A}_i$ 进行排序,按照等级1在前,等级3在最后的顺序进行排序,获得优先分级矩阵 $\boldsymbol{P}'_R$。

为了进一步说明上述方法,针对某一矩阵 $\boldsymbol{A}_i$,假设$\boldsymbol{P}'_R=\left[\begin{array}{cccc|cccc|c} 1 & 3 & 5 & 8 & 2 & 4 & 10 & 9 & 7 & 6 \\ 9 & 8 & 1 & 4 & 3 & 7 & 6 & 5 & 2 & 10 \\ 10 & 3 & 6 & 4 & 1 & 9 & 7 & 8 & 5 & 2 \end{array}\right]$,则微粒 $\boldsymbol{X}_i$ 最终可以解码为:$\boldsymbol{X}_i=\begin{bmatrix} 2 & 10 & 9 & 6 \\ 3 & 4 & 5 & 0 \\ 1 & 8 & 7 & 0 \end{bmatrix}$。

对于第1架直升机,由于任务2,10和9都在等级2中,故将任务9和10依次提到任务2后面,然后将任务6放入最后;对于第2架直升机,任务5属于等级2,任务3和4属于等级1,因此,将3和4提前,将5移到后面;对于直升机3,任务1,8和7都属于同一等级,故保持不变。

根据上述方法来解码,充分考虑了被困人员的存活时间,以及他与直升机之间的距离,可以为将微粒解码为一个违反约束较小或不违反约束的解,加快微粒群的搜索过程。

2.3 微粒速度和位置更新公式

本文采用的是带线性递减惯性权重的微粒群优化算法[22],假设第 i 个微粒的位置向量表示为 $\boldsymbol{X}_i=(x_{i1},x_{i2},\cdots,x_{in})^{\mathrm{T}}$,速度向量表示为 $\boldsymbol{V}_i=(v_{i1},v_{i2},\cdots,v_{in})^{\mathrm{T}}$。第 i 个微粒的最优位置为 $\boldsymbol{p}_{\mathrm{best}\,i}=(p_{i1},p_{i2},\cdots,p_{in})^{\mathrm{T}}$,整个微粒群的全局最优解为 $\boldsymbol{g}_{\mathrm{best}}=(p_1,p_2,\cdots,p_n)^{\mathrm{T}}$,本文还针对PSO算法采取了改进措施,引入全局最差解的概念,以一定的概率接受较差的解,进而找到全局最优解。其中:$p_{wj}(t)$ 和 $\boldsymbol{g}_{\mathrm{best}}$ 相反,即微粒群中最差的解,t 为当前迭代的代数,$t_{\max}$ 是最大迭代次数。因此,最差解的参与次数会随着迭代次数 t 的增加而越来越少,在算法运行后期,最差解对算法的影响则相应会很小。

那么,微粒的速度和位置的更新公式如下:

$$v_{ij}(t+1)=\omega v_{ij}(t)+c_1 r_1(p_{ij}(t)-x_{ij}(t))+c_2 r_2(p_{gj}(t)-x_{ij}(t))+\frac{t_{\max}-t}{t_{\max}}(p_{wj}(t)-x_{ij}(t)), \tag{6}$$

$$x_{ij}(t+1)=x_{ij}(t)+v_{ij}(t+1)。 \tag{7}$$

其中：ω 为线性递减惯性权重，c_1 和 c_2 为学习因子，r_1 和 r_2 为 0 到 1 之间的随机数。

2.4 个体与全局极值更新

为避免 PSO 算法陷入局部最优，本文在微粒个体最优解和全局最优解更新部分引入了模拟退火法，以一定的概率接受较差的解，以摆脱局部最优的位置，从而找到真正的全局最优解。以下是具体的方法描述：首先将模拟退火算法中的温度 T 转换为控制参数 u，能量差 ΔE 转换为适应度之差 Δf，表示为 $\Delta f=f_{\text{上一代最优解}}-f_{\text{更新解}}$。

根据模拟退火算法的公式：$P(\Delta E)=\exp\left(\frac{\Delta E}{kT}\right)$，定义本文模拟退火接受准则的公式为 $P(\Delta f)=\exp\left(\frac{\Delta f}{ku}\right)$，初始控制参数为 u_0，参数下限为 $u_{\min}$，降温系数为 q。具体的流程如下：

步骤 1：初始化控制参数，令 $u=u_0$；

步骤 2：产生更新解后，计算适应度差 Δf；

步骤 3：判断 Δf 是否大于 0。若大于 0，则说明产生了适应度更好的更新解，接受这个解，此时 $\boldsymbol{g}_{\text{best}}$ 更新为这个解，并进行步骤 6；否则说明产生了较差的更新解，进行步骤 4；

步骤 4：随机产生一个 0～1 之间的随机数 rand，若 $\exp\left(\frac{\Delta f}{ku}\right)<\text{rand}$，则进行步骤 5；否则无操作，进行步骤 6；

步骤 5：接受适应度较差的更新解，并放入一个储备集 archive 中，统计 $\boldsymbol{g}_{\text{best}}$ 没有发生变化的次数，如果超过设定的阈值，则将 $\boldsymbol{g}_{\text{best}}$ 更新为 archive 中的随机一个解；

步骤 6：“降温”，$u=u\times q$，重复步骤 2 到 6。

2.5 局部搜索方法

PSO 算法有着优异的全局搜索能力，但是在算法运行后期存在局部搜索能力差的问题，导致无法找到全局最优解。因此，本文引入局部搜索方法，在算法运行后期（即算法运行到 $\frac{2}{3}$ 最大迭代次数时）对 $\boldsymbol{g}_{\text{best}}$ 进行局部搜索。首先，根据优先等级，判断出 $\boldsymbol{g}_{\text{best}}$ 中每一行的等级；接着，对等级 1 和 2 分别进行全排列，则产生多个解，计算这些解的适应值，从中选择最大的解，与 $\boldsymbol{g}_{\text{best}}$ 进行比较，如果大于 $\boldsymbol{g}_{\text{best}}$，则 $\boldsymbol{g}_{\text{best}}$ 更新为当前解；否则，$\boldsymbol{g}_{\text{best}}$ 保持不变。

局部搜索操作步骤如算法 1 中伪码所示。

算法 1　局部搜索操作

输入：原全局最优解：$\boldsymbol{g}_{\text{best}_i}=[x_{i1}\quad x_{i2}\quad x_{i3}\quad\cdots\quad x_{ij}\quad\cdots\quad x_{iN}]$；

输出：局部搜索后全局最优解：$\boldsymbol{g}^*_{\text{best}_i}=[x^*_{i1}\quad x^*_{i2}\quad x^*_{i3}\quad\cdots\quad x^*_{ij}\quad\cdots\quad x^*_{iN}]$；

01：开始；

02：将 $\boldsymbol{g}_{\text{best}_i}$ 进行解码，得救援分配矩阵 $\boldsymbol{A}$，并计算适应度为 f_i；

03：根据优先级矩阵 $\boldsymbol{P}_R$，对 $\boldsymbol{A}$ 进行优先分级，得优先分级矩阵 $\boldsymbol{P}'_R$；

04：根据 $\boldsymbol{P}'_R$，对其中等级 1 和 2 的元素序列进行全排列，产生多个邻域解 X_{g_j}；

05：计算 X_{g_j} 的适应度，从中选出最大的解 X_g，适应度为 f_g；

06：比较 f_i 和 f_g，若 $f_g>f_i$，则 $\boldsymbol{g}^*_{\text{best}_i}=X_g$，否则 $\boldsymbol{g}^*_{\text{best}_i}=\boldsymbol{g}_{\text{best}}$；

07：结束。

2.6 算法步骤

按照上述思路，本文的算法步骤如下：

步骤 1：初始化环境参数和微粒群优化方法参数；

步骤 2：采用 2.2 节方法对微粒进行解码，并计算适应值，选择微粒的个体极值和全局极值。

步骤 3：根据(6)，(7)式更新微粒的位置和速度；

步骤 4：计算更新后的微粒适应值，判断变异操作条件，若满足条件则进行变异操作；

步骤 5：更新微粒的全局极值和个体极值；

步骤 6：判断是否满足停止循环的条件，若满足则停止循环，否则，返回步骤 3。

3 仿真

3.1 仿真环境及参数设置

为验证本文所述算法的有效性,首先,确定算法运行参数;接着,将所提方法与自适应大规模邻域搜索算法[23]进行比较。仿真环境如下:Windows 10,I5-8250U,8 GB,Matlab R2018a。PSO算法的参数设置如下:微粒的个数设置为200,最大迭代次数由实验确定,进化因子 $c_1=c_2=2$,线性权重系数 $\omega=0.9-0.5\times\frac{k}{N_{\max}}$,$t$ 表示当前迭代次数,微粒最大飞行速度为 $V_{\max}=5$。救援环境为一个 300 km×600 km 的长方形区域,救援起点和终点都为原点(0,0),被困人员的存活时间为0～5 h之间的随机数,每个被困人员的救援时间为10 min。直升机飞行速度 v_r 为250 km·h^{-1},最大载荷量为4人。

3.2 参数影响

由文献[22]可知,参数对PSO算法的运行结果有很大影响,因此,首先分析PSO算法中各参数对计算结果的影响,并据此确定参数的具体值。

3.2.1 *微粒个数对算法的影响* 首先,在被困人员数量为14,直升机数量为3的实验场景下,验证微粒个数对算法结果的影响,其他参数不变,观察微粒个数变化对最大救援人数和收敛时间的影响,具体结果如表1、表2所示。

从表1可以看出,当微粒个数为100时,10次实验中,只有9次实验结果达到最大值11,而随着微粒个数的增加,设置为200,300时,10次全部都可以达到最大值11。这说明增加微粒数对于PSO算法的最优取值有正面作用。从表2的收敛时间可以发现,微粒个数的增加仅会对全部运行时间(即运行完最大迭代次数的时间)产生影响,而对收敛到最优解的时间没有特别明显的影响。综合表1和表2的实验结果,本文选择微粒数为200。

3.2.2 *最大迭代次数对算法的影响* 进一步,验证最大迭代次数变化对算法结果的影响,保证被困人员数量($N=14$)和其他参数不变,观察最大迭代次数的变化对最大救援人数和收敛时间的影响,结果见表3和表4。

表1 微粒个数变化对最大救援人数的影响

Tab.1 The influence of partide number change on the maximum number of people rescued 人

微粒数	实验次数									
	1	2	3	4	5	6	7	8	9	10
100	11	11	11	11	11	11	11	10	11	11
200	11	11	11	11	11	11	11	11	11	11
300	11	11	11	11	11	11	11	11	11	11

表2 微粒个数变化对收敛到最大值的时间的影响

Tab.2 The influence of partide number change on the time to converge to the maximum value s

微粒数	实验次数									
	1	2	3	4	5	6	7	8	9	10
100	6.5	1.4	3	1.9	1	4.5	51	2.5	4.2	3.3
200	1.1	1.3	1.5	3.2	5.7	1.6	3.4	1.2	5	6.2
300	3.1	2.6	1.4	3.2	1.2	0.9	1.1	1	2.7	1.3

注:适应值 F 未达到11的实验中,时间表示运行完最大迭代次数的时间,下同

表3 最大迭代次数对最大救援人数的影响

Tab.3 The influence of maxmum iteration number change on the maximum number of people rescued 人

迭代次数	实验次数									
	1	2	3	4	5	6	7	8	9	10
500	10	11	11	11	11	11	11	11	11	11
800	11	11	11	11	11	11	11	11	11	11
1 000	11	11	11	11	11	11	11	11	11	11

表4 最大迭代次数对收敛到最大值的时间

Tab.4 The influence of maxmum iteration number change on the time to converge to the maximum s

迭代次数	实验次数									
	1	2	3	4	5	6	7	8	9	10
500	49	3.2	2	4	5.4	1	2.6	4.4	3.3	1.2
800	1.1	1.3	1.5	3.2	5.7	1.6	3.4	1.2	5	6.2
1 000	3.1	2.7	1	1.7	3.5	1.8	4.4	3.9	4.5	5.9

从表3可以看出,最大迭代次数为500时,算法运行结果达到最大值11的个数最少,而当最大迭代次数设置为800和1 000时,10次运行的适应值都为11。这说明迭代次数的增加对PSO算法的最优取值也有正面作用。而从表4中可以看出,PSO算法的收敛时间(即达到最大值11的时间)具有较大的随机性,与最大迭代次数无关。易知,算法的运行时间与最大迭代次数呈正相关,因此,本文选择最大迭代次数为800。

3.3 对比实验

为验证本文所用 PSO 算法的合理性和优越性，将本文所提的改进 PSO 算法与 ALNS 算法在 5 个不同场景下进行比较，针对 5 个场景，分别为场景 1～5。首先，利用 Matlab 软件自带的 rng()函数，保证被困人员位置 X_S 和剩余生存时间 T_S 在两个算法中产生的随机数保持一致，分别为 rng(8)和 rng(2)。具体实验环境设置如表 5 所示，实验结果见表 6～8。其中，加黑数字为最优值。

表 5　实验场景

Tab. 5　The experimental scenes

实验场景	1	2	3	4	5
被困人员数量 N/人	14	16	20	30	60
直升机数量 M/架	3	4	5	7	15

表 6　不同救援场景下的仿真结果

Tab. 6　Simulation results under different rescue scenarios

场景		PSO			ANLS		
		F/人	T/s	t/s	F/人	T/s	t/s
$M=3$，$N=14$	1	11	89	7.0	11	14	12.0
	2	11	90	1.0	11	9	4.5
	3	11	83	7.4	11	13	0.5
	4	11	85	2.0	11	15	1.9
	5	11	91	2.8	11	14	0.1
	6	11	92	49.0	11	14	1.3
	7	11	90	8.7	11	15	0.8
	8	11	89	1.0	11	10	1.7
	9	11	89	2.3	11	14	1.6
	10	11	91	7.2	11	11	3.6
	平均	**11**	88.9	8.8	**11**	**12.9**	**2.8**
$M=4$，$N=16$	1	13	104	0.7	13	14	0.7
	2	13	99	0.9	13	18	0.1
	3	13	98	1.1	13	19	0.4
	4	13	100	0.1	13	18	1.1
	5	13	101	1.0	13	19	1.7
	6	13	99	0.1	13	15	0.8
	7	13	101	0.1	13	14	0.6
	8	13	98	0.5	13	17	0.2
	9	13	97	1.0	13	12	2.0
	10	13	98	2.0	13	16	0.8
	平均	**13**	99.5	**0.8**	**13**	**16.2**	**0.8**
$M=5$，$N=20$	1	16	121	11.2	16	36	24.0
	2	16	119	1.0	16	32	2.4
	3	16	125	1.0	16	35	9.2
	4	16	124	1.0	16	27	6.0
	5	16	125	6.9	16	19	9.1
	6	16	126	2.8	16	34	8.7
$M=5$，$N=20$	7	16	125	2.0	16	28	1.4
	8	16	120	3.4	16	28	3.7
	9	16	126	2.1	16	34	4.5
	10	16	126	2.1	16	31	3.7
	平均	**16**	123.7	**3.4**	**16**	**30.4**	7.3
$M=7$，$N=30$	1	23	221	57.2	23	115	87.7
	2	23	225	37.6	23	94	23.5
	3	22	230	4.7	23	123	82.1
	4	23	231	153.0	23	123	71.0
	5	23	229	14.0	23	102	26.8
	6	23	231	34.5	22	120	79.8
	7	23	232	59.0	21	136	1.7
	8	22	229	34.9	23	116	65.3
	9	23	228	118.6	23	121	50.4
	10	22	228	134.2	22	124	6.2
	平均	**22.7**	228.4	64.8	22.6	**117.4**	**49.4**
$M=15$，$N=60$	1	45	522	211.7	46	563	304.7
	2	45	601	388.5	44	812	216.2
	3	46	597	280.1	44	755	414.3
	4	45	612	169.7	45	745	398.6
	5	46	527	98.8	45	733	451.7
	6	45	565	174.4	46	633	440.7
	7	45	517	161.6	45	664	434.1
	8	46	591	184.7	46	837	576.5
	9	46	522	424.1	46	701	513.5
	10	46	572	203.1	46	953	754.1
	平均	**45.5**	**562.6**	**229.7**	45.3	739.6	450.4

从上述实验结果可以看出，改进后的 PSO 算法在前期问题规模较小的情况下，所得实验结果（最大救援人

数、算法总用时和收敛时间)相对于ALNS算法稍有劣势。但是当问题规模增大时,PSO算法的最大救援人数已呈现优于ALNS算法的趋势,而在算法总用时和收敛时间这两个指标上,PSO算法所得结果明显要优于ALNS算法,这对于救援任务分配问题尤为关键。因此,上述不同场景的仿真结果验证了本文算法的优越性。

表7为场景1下被困人员的剩余生存时间和位置信息,表8采用本文所提算法运行后的结果,可以看出10次结果均可以收敛到最大救援人数11。图2给出了场景环境1下各直升机的救援路径示意图,从图中可以看出,每架直升机的具体救援路径为:直升机Ⅰ:11-5-7-10-返航-14;直升机Ⅱ:4-12-8-3-返航-9;直升机Ⅲ:6-13-1-2。从各直升机的救援路径可以看出,直升机Ⅰ和Ⅱ在达到最大载荷后,返航到起始点,然后继续从起始点出发来完成下一个任务。

表7 场景1被困人员信息

Tab.7 Information of trapped persons in Scenario 1

被困人员	剩余生存时间/h	位置/km	被困人员	剩余生存时间/h	位置/km	被困人员	剩余生存时间/h	位置/km
1	2.179	(224,286)	6	1.651	(−293,66)	11	3.105	(33,21)
2	0.129	(281,128)	7	1.023	(−42,20)	12	2.645	(26,67)
3	2.748	(222,87)	8	3.096	(−59,295)	13	0.672	(157,118)
4	2.176	(19,292)	9	1.498	(14,38)	14	2.567	(127,269)
5	2.101	(−160,100)	10	1.334	(−13,97)			

表8 场景1的算法运行结果

Tab.8 The algorithm results of Scenario 1

次数	最大救援人数	运算时间/s	最优解	次数	最大救援人数	运算时间/s	最优解
1	11	1.1	11-5-7-10-14-4-12-8-3-9-6-13-1-2	6	11	1.6	7-9-14-6-12-1-5-10-3-13-11-2-8-4
2	11	1.3	12-7-1-10-6-14-9-5-3-8-11-4-2-13	7	11	3.4	10-7-4-12-6-14-9-8-1-11-13-5-3-2
3	11	1.5	13-9-7-1-10-8-14-4-3-2-12-5-6-11	8	11	1.2	12-10-6-7-14-5-9-3-8-11-2-13-1-4
4	11	3.2	11-9-5-10-6-14-7-8-13-12-2-4-3-1	9	11	5.0	11-10-1-6-12-3-5-7-8-13-9-2-4-14
5	11	5.7	7-9-12-10-5-3-4-6-14-8-11-2-13-1	10	11	6.2	5-12-14-4-7-1-11-10-3-6-9-13-2-8

4 结论

本文针对直升机救援规划问题,提出了一种多直升机多救援目标的救援任务分配方法。首先,根据救援问题的特殊性,以最大化救援人数为目标,建立带有约束条件的数学模型。接着,对PSO算法的微粒进行编码和解码,将微粒转换成救援分配矩阵,得到每一架直升机的具体救援顺序。随后,将直升机救援总人数作为微粒的适应度,采用改进的微粒群算法来求解,通过改进微粒的更新公式,全局极值的更新策略,增加局部搜索等来加快算法收敛。最后,不同场景的仿真结果验证了所提方法在处理大中规模救援问题上要优于对比算法。

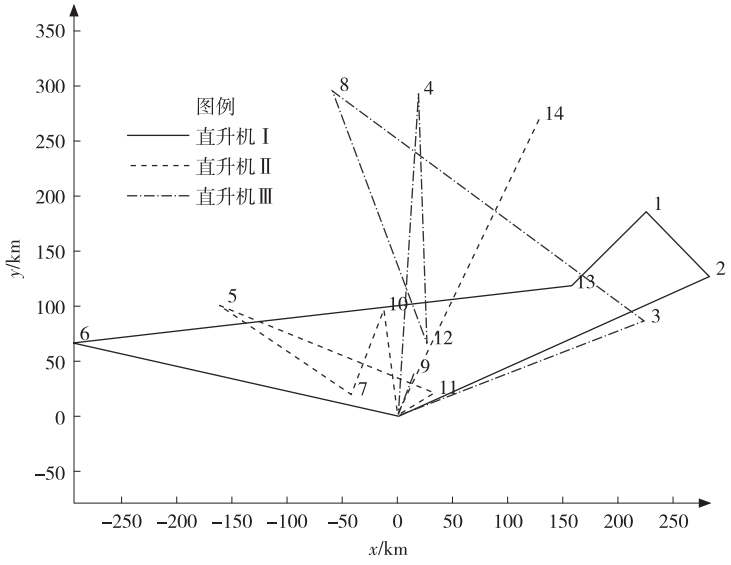

图2 场景1环境下各直升机救援路径示意图

Fig.2 Rescue path diagram of each helicopter under scenario 1

但是本文所用PSO算法仍存在陷入局部极值和收敛速度慢的缺点,因此,在后续研究中可尝试通过结合其

他优化算法来提高 PSO 算法的全局寻优能力，加快算法的收敛速度。

参考文献：

[1] 曹梁文，朱俊志，尚永涛，等. 基于波多黎各飓风灾难的无人机救援规划研究[J]. 信息技术与信息化，2019(7)：167-171.
CAO L W，ZHU J Z，SHANG Y T，et al. Research on UAV rescue planning based on Puerto Rico hurricane disaster[J]. Information Technology and Informatization，2019(7)：167-171.

[2] 修言. 直升飞机在救援中的作用[J]. 中国个体防护装备，2008(3)：9.
XIU Y. The role of helicopters in rescue[J]. China Personal Protective Equipment，2008(3)：9.

[3] 李明龙，杨文婧，易晓东，等. 面向灾难搜索救援场景的空地协同无人群体任务规划研究[J]. 机械工程学报，2019，55(11)：1-9.
LI M L，YANG W J，YI X D，et al. Swarm robot task planning based on air and ground coordination for disaster search and rescue[J]. Journal of Mechanical Engineering，2019，55(11)：1-9.

[4] 赵慧潮，石朝侠. 基于市场法的多机器人协作未知环境探索[J]. 计算机与数字工程，2017，45(11)：2085-2089.
ZHAO H C，SHI C X. Unknown environment exploration of multi-robot based on market[J]. Computer & Digital Engineering，2017，45(11)：2085-2089.

[5] 梁国强，康宇航，邢志川，等. 基于离散粒子群优化的无人机协同多任务分配[J]. 计算机仿真，2018，35(2)：22-28.
LIANG G Q，KANG Y H，XING Z C，et al. UAV cooperative multi-task assignment based on discreteparticle swarm optimization algorithm[J]. Computer Simulation，2018，35(2)：22-28.

[6] 徐正伟，张亮，马芳. 基于改进遗传算法的多植保无人机任务分配研究[J]. 工业控制计算机，2019，32(6)：43-44.
XU Z W，ZHANG L，MA F. Task assignment of multi-plant protection UAVs based on improved genetic algorithm[J]. Industrial Control Computer，2019，32(6)：43-44.

[7] 曹如月，李世超，季宇寒，等. 基于蚁群算法的多机协同作业任务规划[J]. 农业机械学报，2019，50(B07)：34-39.
CAO R Y，LI S C，JI Y H，et al. Multi-machine cooperation task planning based on ant colony algorithm[J]. Transactions of the Chinese Society for Agricultural，2019，50(B07)：34-39.

[8] 周凌超. 基于改进模拟退火算法的导弹目标分配方法[J]. 工业控制计算机，2018，31(1)：95-97.
ZHOU L C. Target assignment for missile based on improved simulated annealing algorithm[J]. Industrial Control Computer，2018，31(1)：95-97.

[9] 李坤，徐铮，田慧欣. 基于自适应变邻域搜索算法的一类混合流水车间调度问题[J]. 系统工程，2015，33(11)：121-129.
LI K，XU Z，TIAN H X. An adaptive variable neighbourhood search algorithm for the hybrid flowshop scheduling problem[J]. System Engineering，2015，33(11)：121-129.

[10] 王雅琳，王宁，阳春华，等. 求解任务分配问题的一种离散微粒群算法[J]. 中南大学学报(自然科学版)，2008，39(3)：571-576.
WANG Y L，WANG N，YANG C H，et al. A discrete particle swarm optimization algorithm fortask assignment problem[J]. Journal of Central South University (Science and Technology)，2008，39(3)：571-576.

[11] 张峰. 基于微粒群算法的民航进港航班排序优化[J]. 中国科技投资，2019(19)：235.
ZHANG F. Ranking optimization of civil aviation incoming flights based on particle swarm optimization[J]. China Venture Capital，2019(19)：235.

[12] 万婧. 求解可重入作业车间调度问题的改进离散微粒群优化算法[J]. 山东科学，2017，30(6)：105-109.
WAN J. Improved discrete particle swarm optimization algorithm to solve thereentrant job-shop scheduling problem[J]. Shandong Science，2017，30(6)：105-109.

[13] NIE Z B，YANG X B，GAO S H，et al. Research on autonomous moving robot path planning based on improved particles warm optimization[C]//2016 IEEE Congresson Evolutionary Computation(CEC). Piscataway，New Jersey：IEEE，2016：2532-2536.

[14] LI N，SONG Z. Modified particles warm optimization and its application in multimodal function optimization[C]//Proceedings 2011 International Conference on Transportation，Mechanical，and Electrical Engineering(TMEE). Piscataway，New Jersey：IEEE，2011：375-378.

[15] SEO J H，IM C H，HEO C G，et al. Multimodal function optimization based on particles warm optimization[J]. IEEE Transactionson Magnetics，2006，42(4)：1095-1098.

[16] 翟文正，胡越黎，冉峰. 异构多核 DAG 任务模型的微粒群优化调度算法[J]. 计算机工程与设计，2016，37(7)：1831-1835.
ZHAI W Z，HU Y L，RAN F. PSO scheduling algorithm for heterogeneous multi-core DAG task model[J]. Computer Engineering and Design，2016，37(7)：1831-1835.

[17] 卜艳萍，俞金寿. 离散微粒群优化算法在网格任务调度中的应用[J]. 计算机仿真，2008，25(4)：175-178.

BU Y P, YU J S. Application of discrete particle swarm optimization algorithm to grid task scheduling[J]. Computer Simulation, 2008, 25(4): 175-178.

[18] 郭研,李南,李兴森. 基于云多目标微粒群算法的多项目调度方法[J]. 计算机工程与应用, 2012, 48(21): 15-20.
GUO Y, LI N, LI X S. Multiple projects scheduling method based on cloud multi-objective particle swarm optimization[J]. Computer Engineering and Applications, 2012, 48(21): 15-20.

[19] GENG N, MENG Q G, GONG D W, et al. How good are distributed allocational gorithms for solving urban search and rescue problems? A comparative study with centralized algorithms[J]. IEEE Transactionson Automation Science and Engineering, 2019, 16(1): 478-485.

[20] LIN S M, GENG N, SUN J, et al. A grouping method based on improved PSO for task allocation in rescue environment[C]// 2019 IEEE Congresson Evolutionary Computation(CEC), Wellingdon, NewZealand. Piscataway, New Jersey: 2019: 619-626.

[21] EBERHART R C, KENNED Y J. A new optimizer using particle swarm theory[C]//MHS'95. Proceedings of the Sixth International Symposiumon Micro Machine and Human Science. Piscataway, New Jersey: IEEE, 1995: 39-43.

[22] SHI Y, EBERHART R C. Empirical study of particles warm optimization[C]//Proceedings of the 1999 congress on evolutionary computation-CEC99(Cat. No. 99TH8406). Piscataway, New Jersey: IEEE, 1999, 3: 1945-1950.

[23] ROPKE S, PISINGER D. An adaptive large neighborhood search heuristic for the pick up and delivery problem with time windows[J]. Transportation Science, 2006, 40(4): 455-472.

Operations Research and Cybernetics

Solving the Task Allocation Problem of Load-Constrained after Disaster Based on PSO Algorithm

GAO Kai[1], GENG Na[2], ZHANG Xinyue[2], QUANG A Nguyen[3]

(1. School of Electrical Engineering and Automation, Jiangsu Normal University, Xuzhou Jiangsu 221116;
2. School of Automation, Northwestern Polytechnical University, Xi'an 710129, China; 3. The Data Driven Research and Innovation Programme, Centre for Intelligent Healthcare, Coventry University, Coventry CV4 7AL, UK)

Abstract: [**Purposes**]Aiming at the problem of rescue after disaster, a rescue task allocation problem of multi-helicopter multi-task with constraints after disaster were studied, and a problem-solving method based on Particle Swarm Optimization (PSO) algorithm was proposed so as to obtain the allocation assignment for each helicopter. [**Methods**]Firstly, taking rescue time and helicopter loads constraints into account, and expecting to rescue the maximum number of trapped people in a limited time, the mathematical model of the problem was established according to the characteristics of the problem; Secondly, the modified PSO algorithm was employed to solve the mathematical model, which mainly included particle decode, particle velocity and position update formulae, g_{best} update method, local search method and so on. [**Findings**]The proposed method is employed to solve the task allocation problem in different rescue scenarios, and compared with the Adaptive Large Neighborhood Search (ANLS) algorithm. [**Conclusions**]The simulation results verify the effectiveness of the proposed method.

Keywords: rescue after disaster; load-constrained; helicopter; task allocation; particle swarm optimization

(责任编辑 黄 颖)