第三章 无人机群协同攻击任务分配方法研究

3.1 引言

随着无人机技术的发展以及未来空战中对集群作战的需求，无人机群协同执行任务即将成为未来空中作战的主要形式。无人机群任务分配作为无人机指挥控制系统的顶层决策，该问题的求解也受到了研究者们的广泛关注。无人机群任务分配问题指的是，在作战过程中，综合考虑无人机航程、飞行消耗、任务类型、时间窗口等约束条件，求解得到无人机群任务调度方案，使得无人机群整体所消耗的成本最小，提高无人机群的作战效能。现阶段对于无人机群任务预分配阶段，多采用集中式算法，PSO 及 GA 算法得到了广泛的应用。

值得注意的是，现有的大部分针对攻击场景的任务分配方法研究的目标为单打击任务[17–19,65]，即无人机对目标物单独进行打击，即使考虑到打击目标物所需的武力值和无人机载荷约束时，需要多架多人机对目标物进行协同打击，但也不需要多架无人机同时共同打击，所以现阶段大部分研究的任务分配问题中，无人机序列之间不存在时间耦合和约束关系。因此本章针对无人机群协同打击任务分配问题，优化了原有的 PSO 算法，同时通过结合模拟退火机制与局部邻域搜索等理论，提高了算法的寻优能力。

本章的结构如下：3.2 节描述了本章的研究问题，并对无人机群协同攻击任务分配问题的约束条件、算法评价指标和数学模型进行了详细的介绍；3.3 节介绍了所提出的基于模拟退火机制和局部随机搜索的优化粒子群算法；3.4 节给出了仿真实验结果，并对算法中的参数设置进行了重复性实验；3.5 节总结了本章的研究内容及工作。

3.2 无人机群任务分配问题建模

本文研究了多无人机协同攻击地面目标场景下的任务分配问题，属于多任务分配问题 (Multi-Task Assignment Problem, MTAP) 问题。以无人机协同地对地作战任务为背景，战场环境由多架无人机和目标组成。

3.2.1 问题描述

整个问题中的对象主要由无人机、目标物和任务三部分组成。无人机集合为 $U=\{U_i|i=1,2,...,N_u\}$，其中 N_u 为无人机总数。目标集为 $T=\{T_q|q=1,2,...,N_t\}$，其中 N_t 为目标数量，目标物根据需要攻击的无人机数量 Nu_j 分为 2 种类型。

单攻击目标集是 $T_s = \{T_j \in T | Nu_j = 1\}$，它只需要一架无人机进行攻击。多攻击目标集为 $T_m = \{T_j \in T | Nu_j > 1\}$，需要 Nu_j 架无人机进行攻击。任务集是 $M=\{M_q | q = 1, 2, ..., N_m\}$，其中任务总数是 $N_m=\sum Nu_j, T_j \in T$。综合考虑航行距离、任务总体完成时间、任务效益和目标时间窗，建立了任务分配模型，将决策变量定义为 x_{pq}^i，

$$x_{pq}^i = \{0,1\}, i = 1, 2, \cdots, N_u, p, q = 0, 1, 2, \cdots, N_j \tag{3-1}$$

其中，x_{pq}^i 为 0/1 的决策变量，表示第 i 架无人机 U_i 是否从目标物 T_p 飞到目标物 T_q。p 或 q 为 0 时表示为无人机 U_i 的基地。

$T_j = \{ID_T, L_T, R_T, [t_S^T, t_E^T], CT_T, Nu_T\}$ 是目标的属性。其中，其中，ID_T 是目标物编号集合；L_T 为位置集合；R_T 是目标价值集合；$[t_S^T, t_E^T]$ 表示被攻击目标的时间窗口；CT_T 为执行时间集合；Nu_T 为目标所需无人机攻击数量的集合。

$U_i = \{ID_U, B_U, P_U, M_U^{\max}, v_U\}$ 是无人机的属性。其中，ID_U 为无人机编号集合；B_U 为基地位置集合；P_U 为攻击成功率集合；$M_U^{\max}$ 为无人机执行任务的最大次数；v_U 是无人机的速度集合。

在不失一般性的前提下，假设如下：(1) 每架无人机武器载荷有限；(2) 每架无人机的移动速度是均匀的；(3) 范围简化为欧几里德距离；(4) 无人机攻击单攻击目标和移动目标无需等待时间。

3.2.2 评价指标

协同任务分配的目的是给每个 U_i 分配一个任务序列 $\Delta_i = < T_1^i, T_2^i, \ldots, T_{mi}^i >$，其中任务数量小于 $Q_i^{\max}$。执行任务时，U_i 从 B_i 开始，依次执行任务，最后返回。

对于 Δ_i 中目标 T_p^i，需要根据 U_i 攻击目标的到达时间 $G_i^{T_p^i}$ 和开始执行时间 $E_i^{T_p^i}$ 来计算时间窗偏差代价 TW_i[37]。

$$G_i^{T_p^i} = \begin{cases} \dfrac{w\left(B_i, L_{T_1^i}\right)}{v_i}, p = 1 \\ E_i^{T_{p-1}^i} + CT_{T_{p-1}^i} + \dfrac{w\left(L_{T_{p-1}^i}, L_{T_p^i}\right)}{v_i}, p \in \{2, 3 \ldots, m_i\} \end{cases} \tag{3-2}$$

其中，$w(T_p, T_q)$ 表示 T_p 和 T_q 之间的距离，此时采用欧氏距离。

当 U_i 对多攻击目标进行攻击时，需要等待共同执行的无人机全部到达同一目标后才开始执行任务。$\Pi(T_j) = \{U_s \in U | T_j \in \Delta_s\}$ 是攻击 T_p^i 的无人机集合。执行时间可以如下表示[37]：

$$E_i^{T_p^i}=\begin{cases} G_i^{T_p^i}, T_p^i \in T_s \\ \max\limits_{U_s \in \Pi(T_p^i)} \left\{ G_s^{T_p^s} \right\}, T_p^i \in T_{\mathrm{m}} \end{cases} p \in \{2,3,\ldots,m_i\} \tag{3-3}$$

对于每个目标，都有一个攻击的时间窗口 $[t_S^{T_p^i}, t_E^{T_p^i}]$，如果攻击开始的时间窗早于或晚于该时间窗口，则会产生时间窗口的偏差代价，如式 (3-4) 所示 [37]。

$$TW_i=\sum_{p=1}^{mi}\left[b_1 \max\left(t_S^{T_p^i}-G_i^{T_p^i},0\right)+b_2 \max\left(G_i^{T_p^i}-t_E^{T_p^i},0\right)\right] \tag{3-4}$$

其中，b_1 和 b_2 分别为机会损失因子和延迟惩罚因子。

时间代价 t_i 为 U_i 执行任务后并飞回无人机基地的总时间，可以通过式 (3-5) 求得：

$$t_i = E_i^{T_{mi}^i} + CT_{T_{mi}^i} + \frac{w\left(T_{mi}^i, B_i\right)}{v_i} \tag{3-5}$$

距离成本 D_i 是指目标物和基地之间的距离，计算公式如下：

$$D_i=\sum_{p=0}^{N_T}\sum_{q=0}^{N_T} x_{pq}^i w\left(L_p, L_q\right) \tag{3-6}$$

其中，$w\left(L_p, L_q\right)$ 为目标物 L_p, L_q 之间的距离，L_0 表示 U_i 的基地。

攻击奖励可以通过如下方式求得 [37]：

$$I_i = \sum_{q=0}^{N_t} P_{iq} * x_{pq}^i * R_i \tag{3-7}$$

其中，I_i 表示 U_i 在一定成功概率 P_{iq} 下攻击 T_q 获得的奖励。

3.2.3 数学模型

本论文建立的多无人机协同攻击的数学模型如下:

$$\min J = a_1 \sum_{i=1}^{N_u} D_i + a_2 \max_{i=1,2,\cdots,N_u} t_i + a_3 \sum_{i=1}^{N_u} TW_i - a_4 \sum_{i=1}^{N_u} I_i \tag{3-8}$$

$$\sum_{i=1}^{N_u}\sum_{p=0}^{N_t} x_{pq}^{i} = 1, \forall T_q \in T_s \tag{3-9}$$

$$\sum_{i=1}^{N_u}\sum_{p=0}^{N_t} x_{pq}^{i} = Nu_q, \forall T_q \in T_m \tag{3-10}$$

$$\sum_{q=1}^{N_t} x_{0q}^{i} = \sum_{p=1}^{N_t} x_{p0}^{i} = 1, \forall T_p, T_q \in T \tag{3-11}$$

$$\sum_{i=1}^{N_u}\sum_{p=0}^{N_t} x_{pq}^{i} - \sum_{i=1}^{N_u}\sum_{k=0}^{N_t} x_{qk}^{i} = 0, \forall T_p, T_q, T_k \in T \tag{3-12}$$

$$1 \le \sum_{p=0}^{N_t}\sum_{q=1}^{N_t} x_{pq}^{i} \le M_i^{\max}, \forall T_p, T_q \in T \tag{3-13}$$

其中，式 (3-8) 为目标函数，其值为评价解优劣的重要指标适应值；a_1，a_2，a_3，a_4 分别为距离代价、任务时间代价、时间窗偏差代价和攻击收益的比例系数，反应了各部分代价对目标函数的重要程度，根据任务场景的需求来确定比例系数，取值范围为 [0,1]。式 (3-9)-(3-13) 为约束，式 (3-9) 表示单打击目标只需要一架无人机进行攻击，式 (3-10) 表示多打击任务需要 Nu_q 架无人机协同攻击，式 (3-11) 表示每架无人机从基地起飞后并需要返回同一基地，式 (3-12) 表示每架无人机完成任务后都需要飞离该任务点，式 (3-13) 规定了分配给 U_i 的任务至少有一个，且不能超过最大任务数量 $M_i^{\max}$。

3.3 基于优化粒子群的任务分配算法

本章提出了一种针对协同攻击任务场景的优化粒子群任务分配算法，设计了一种融合目标类型、无人机、任务序列等综合信息的单链粒子编解码、修正算法，针对 PSO 容易陷入局部收敛的缺点，在粒子群算法框架中引入基于模拟退火算法的局部邻域搜索筛选机制和局部邻域搜索算法，提高算法全局寻优能力。

3.3.1 粒子编解码策略

标准粒子群算法中解空间是连续性的，但本文研究的无人机群任务分配问题的解是离散的，需要对粒子的位置和速度进行离散化编解码，形成一个任务分配基本解。

本文采用单链实数编码方式，粒子 k 的位置 X_k 和速度 Y_k 都为 D-维实数向量，

其中 $D=\sum Nu_j+N_u-1, T_j\in T$，位置向量 X_k 中包括所有目标物和无人机的编号信息，考虑在迭代过程中粒子位置元素的可操作性和数据合理性，限制位置向量元素 $X_k[q]$ 范围 $X_k[q]\in[1,D], q=1,2,\cdots,D$，同时考虑到粒子位置元素的范围要求，所以限定粒子速度元素 $Y_k[q]\in\left[\frac{-N_t}{2},\frac{N_t}{2}\right]$。

本文粒子解码依据为粒子位置元素素 $X_k[q]$ 的大小排序、具体任务序号与目标物序号的映射关系以及无人机编号出现的位置，具体任务序号与目标物序号的映射表如下表3-1所示。

表 3-1 任务-目标物序号映射表

粒子向量元素顺序	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
映射对象	T_1	T_2	T_2	T_3	T_4	T_5	T_5	T_6	T_7	T_8	T_8	T_9	T_{10}	U_1	U_2

解码过程中记录 N_u-1 个无人机编号出现的位置 Pos，第 i 个无人机编号出现的位置 $Pos(i)$ 与第 $i-1$ 个的无人机编号出现的位置 $Pos(i-1)$ 之间的元素与原始任务序列按照大小映射之后的目标物序列，则为第 i 个无人机的任务序列 Δ_i。第 N_u 个无人机的任务序列为第 N_u-1 个的无人机编号出现的位置 $Pos(N_u-1)$ 与 $X_k[D]$ 之间的元素 (包括位置向量元素 $X_k[D]$) 与原始任务序列按照大小顺序映射之后的目标物序列。粒子位置向量编码和对应任务分配解如表3-2和表3-3所示。

表 3-2 粒子位置向量解码表

X_k	**粒子向量元素顺序**	**映射对象**
1.288	2	T_2
14.686	15	U_2
14.488	13	T_{10}
6.310	5	T_4
14.560	14	U_1
10.951	11	T_8
1.090	1	T_1
9.272	9	T_7
10.076	10	T_8
14.268	12	T_9
7.067	6	T_5
2.653	3	T_2
8.938	7	T_5
9.126	8	T_6
4.404	4	T_3

表 3-3 粒子解码任务分配解

无人机	任务序列
1	T_{10}-T_4
2	T_2
3	T_8-T_1-T_7-T_8-T_9-T_5-T_2-T_5-T_6-T_3

3.3.2 粒子修正策略

由于生成粒子的随机性，解码得到的分配解不一定满足约束条件 (3-9)-(3-13)，对此本研究提出修正算法。修正算法的步骤如算法3-1所示[37]：

算法 3-1 修正算法

input : 任务序列 Δ

output: 修正后任务序列 Δ'

while 存在重复任务 $\left(T_q^k \in T_m \cap T_q^k \in \Delta_i\right) \cup \exists |\Delta_i| > M_i^{\max}$ **do**
  for $1 \le i \le N_u$ **do**
    随机删除 Δ_i 中的 T_q^k;
    随机在 Δ_s 中插入 $M_q^k, s \in [1, N_u], s \ne i$;
  end
  for $1 \le i \le N_u$ **do**
    if $|\Delta_i| > M_i^{\max}$ **then**
      随机删除 Δ_i 中的 T_p^k;
      随机在 Δ_s 中插入 $M_p^k, s \in [1, N_u], s \ne i$;
    end
  end
end

经由修正算法得到的任务分配解可能存在“死锁”现象，即不同无人机执行同时攻击多打击任务时，出现互相等待的现象，导致任务序列陷入僵局，无人机群无法继续执行任务序列。例如无人机群的任务序列 Δ' 如下表3-4所示。其中目标物 T_2 和 T_5 同时需要无人机 U_1 和 U_3 共同执行，U_1 先执行 T_2 再执行 T_5，U_3 执行任务的顺序则与之相反，此时，U_1 与 U_3 则会陷入僵局中。

因此本研究采用有向图 $G = \langle V, E \rangle$，$V = \{j | T_j \in T\}$，$E = \{(i,j) | i,j \in V, i \ne j\}$ 表征死锁，若 G 中存在强联通分量，即存在死锁现象。表3-4中的任务序列的有向图如图3-1所示，图3-1中节点 2 和节点 5 之间存在强联通分量，则存在死锁现象，对

表 3-4 无人机群死锁任务序列

无人机	任务序列
1	T_2-T_5-T_6-T_1-T_3
2	T_4-T_8
3	T_7-T_8-T_9-T_5-T_2-T_{10}

此本文采用去死锁算法，对此本研究采用去死锁算法，步骤如算法3-2所示[37]：

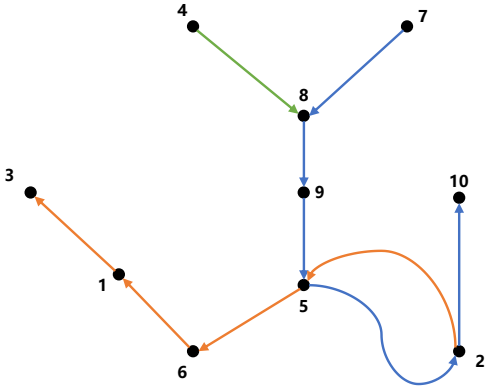


图 3-1 任务序列有向图

算法 3-2 去死锁算法

input ：修正后任务序列 Δ'

output: 去死锁任务序列 Δ''

构造 Δ' 的有向图 G;

while G 中有强联通分量 C **do**

  将强联通分量中的任意两个点之间的连线反向;

end

3.3.3 粒子更新策略

标准粒子更新受个体历代最优解和全局最优解引导，本文重点改进学习因子和粒子速度更新公式[18]，引入社会认知部分，使得粒子更新前期由局部最优引导，后期由全局最优引导，同时每次迭代过程中加入随机因子的影响，保证粒子在探索全局空间的同时，挖掘出更优解。第 t 代粒子 k 的速度位置迭代公式如下所示：

$$Y_k^{t+1} = \omega Y_k^t + c_1 * r_1 * (Xp_k^t - X_k^t) + c_2 * r_2 * \left(X_g^t - X_k^t\right) + \beta * r_3 * \left(X_j^t - X_k^t\right) \quad (3\text{-}14)$$

$$X_k^{t+1} = X_k^t + Y_k^{t+1} \tag{3-15}$$

$$\omega = \omega_{\max} - Iter \times \frac{(\omega_{\max} - \omega_{\min})}{Iter_{\max}} \tag{3-16}$$

$$c_1 = e^{\frac{f_k^t}{f_{pk}^t}} - 1, c_2 = e^{\frac{f_{pk}^t}{f_{gb}}} \tag{3-17}$$

$$\beta = \beta_{\max} + (\beta_{\min} - \beta_{\max}) * \frac{Iter}{Iter_{\max}} \tag{3-18}$$

其中 ω 是惯性权重，r_1, r_2, r_3 为 0 到 1 之间的随机数，X_k^t, Y_k^t 是当前粒子 k 的位置和速度向量，Xp_k^t, Xg^t 为粒子 k 历史最优位置向量和全局最优位置向量，f_k^t, fp_k^t, f_{gb} 分别为粒子 k 第 t 代、粒子 k 的历史最优和全局最优的适应值，β 是随机学习因子，c_1 是历史最优学习因子，c_2 为全局最优学习因子。

c_1, c_2, β 都为学习因子，学习因子反映了粒子间信息交换的程度，c_1 促使粒子在他们搜寻过的最佳位置附近进行搜索，c_2 促使粒子在搜索过程中转移到全局最优位置，此种迭代方式根据粒子的适应值自适应地计算学习因子，使其在算法迭代的早期集中于个体最优值邻域内的搜索，在算法的后期集中于全局极值邻域的搜索。为提高粒子的全局搜索能力，在原有粒子速度更新方程的基础上增加了随机社会认知部分，通过增加种群中某个随机粒子对粒子速度和位置的影响，使粒子跳出历史极值和全局极值的邻域，增加了粒子对搜索空间的挖掘能力。

3.3.4 局部随机搜索策略

本文为防止种群陷入局部极小值，并提高算法的全局搜索能力，采用模拟退火机制[18]，设置启动变异算子的阈值，在每次粒子更新的过程中，判断粒子是否陷入局部最优，同时引入多种变异及交换算子[32]，将多种类型的变异算子组合在一起，以增加种群的多样性，提高 PSO 算法的全局搜索性能，使算法具有跳出局部收敛的能力。

3.3.4.1 模拟退火局部随机搜索概率

模拟退火算法 (SA) 最早由 N. Metropolis 等人于 1953 年提出。模拟退火算法通过温度来计算接收概率。在搜索过程中，温度变化缓慢，在搜索空间中进行随机搜索，导致优化缓慢。在粒子群算法中引入 SA 机制，以纠正粒子群算法的早熟现象。该算法在收敛速度快的同时，还具有全局优化性能。引入 SA 机制后，粒子

k 启动局部随机搜索指标概率 p_k 计算方式如下：

$$p_k=\begin{cases} 1, f_k^t < \alpha * f_g \\ \exp(-\rho), f_k^t \geq \alpha * f_g \end{cases} \tag{3-19}$$

$$\rho = \frac{f_k^t - \alpha * f_{gb}}{T_{SA}} \tag{3-20}$$

$$T_{SA} = T_0 * \gamma^{fix\left(\frac{Iter}{100}\right)} \tag{3-21}$$

其中，α 为比例系数，用来筛选是否启动变异算子的阈值，T_0 为初始温度，γ 为降温速度，粒子群算法每迭代 100 次，降温一次。函数 $fix()$ 表示向 0 取整。当随机概率 p 小于 p_k 时，则对粒子 k 启动变异及交换算子。当 $f_k^t < \alpha * f_g$ 时，判定粒子可能已存在在现全局最优点附近，需要对其进行进一步的变异和交换操作，看能否突破现阶段的全局最优值；当 $f_k^t \geq \alpha * f_g$ 时，判定粒子 k 距离全局最优点较远，以一定概率开启变异和交换操作。通过此概率指标 p_k，在对出现在全局最优附近的粒子进行进一步的随机搜索，同时对性能较差的粒子进行筛选，降低启动局部随机搜索的概率，在挖掘全局最优和计算开销之间达到平衡。

3.3.4.2 局部随机搜索 (Local Random Search,LRS)

LRS 算法主要针对存在于全局最优点附近的粒子，对其进行局部的随机搜索，形成新的粒子，看是否能突破此时的全局最优点。本文主要采用三种变异算子和一种交叉算子[32]，修改粒子位置向量 X_k^t 寻找性能更优的解，粒子 k 的新位置向量为 $X_k^{t\prime}$。

1. 变异算子

(1) 单点交换算子：在 X_k^t 上交换两个随机元素 $X_k^t[i], X_k^t[j], i \neq j, i,j < D+N_u-1$。单点交换算子表达式为 $X_k^{t\prime} = M_1(X_k^t, i, j)$，示意图如下图3-2(a) 所示。

(2) 滑动算子：在 X_k^t 上产生两个随机位置 $i, j, i < j$，将位置向量元素 $X_k^t[i]$ 和 $X_k^t[j]$ 之间的元素随机向左或者向右滑动 n 个位置，滑动算子表达式为 $X_k^{t\prime} = M_2(X_k^t, i, j, Dir, n), n < |i-j|-1$，其中 Dir 为 0 或 1 的随机数，0 表示向左滑动，1 表示向右滑动，示意图如下图3-2(b)、(c) 所示。

(3) 翻转算子：在 X_k^t 上产生两个随机位置 $i, j, i < j$，将元素 $X_k^t[i]$ 和元素 $X_k^t[j]$ 之间的片段进行翻转，翻转算子表达式为 $X_k^{t\prime} = M_3(X_k^t, i, j)$，示意图如下图3-2(d) 所示。

2. 交换算子

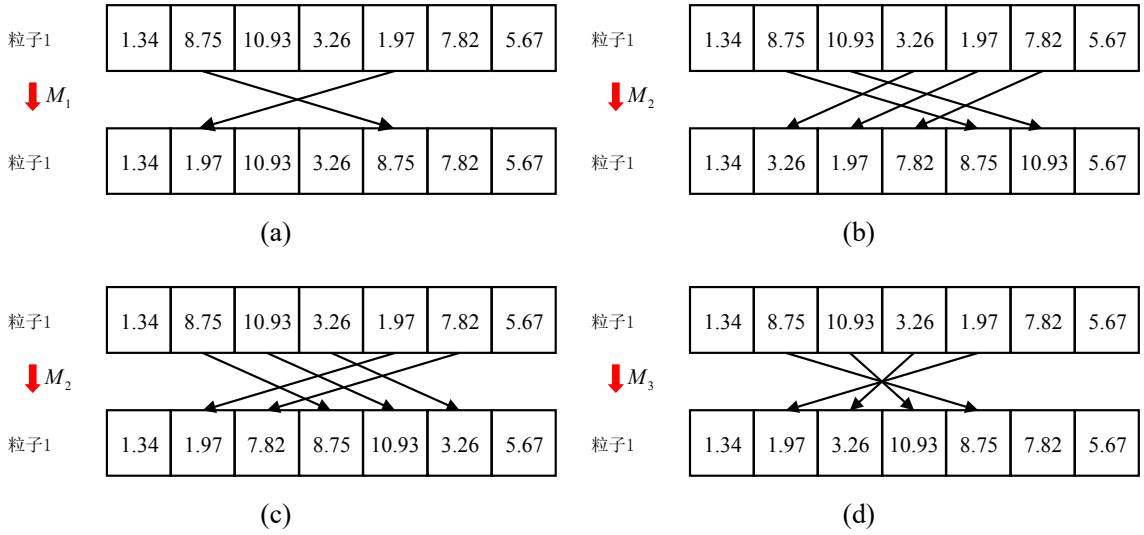


图 3-2 变异算子示意图 (a) 单点交换算子；(b) 滑动算子 (左滑两个位置)；(c) 滑动算子 (右滑两个位置)；(d) 翻转算子

为突破搜寻过程中暂时性的全局最优点，可以通过与其他随机粒子 m 交换位置向量片段，提高种群的多样性，从而提高算法的全局寻优能力。交换算子是产生一个随机位置 i 和随机筛选一个粒子 m 的位置向量 X_m^t，交换 X_k^t 和 X_m^t 位置 i 直至向量末端的片段，交换算子表达式为 $X_k^{t\prime} = M_4\left(X_k^t, X_m^t, i\right), i \le D + N_u - 1$，示意图如下图3-3所示。

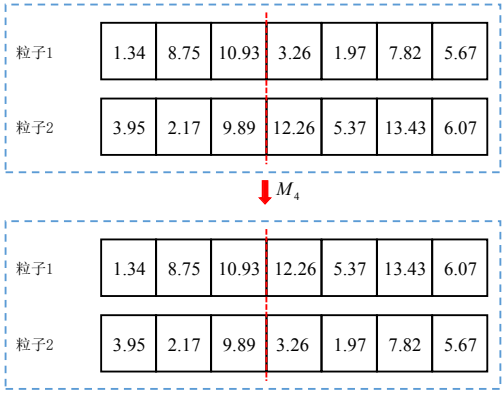


图 3-3 交叉算子示意图

结合上述的三种变异算子和一种交叉算子，提出了一种 LRS 算法，实现突破暂时局部最优点的功能，提高任务分配算法的全局搜索能力。LRS 算法流程如算法3-3所示。

算法 3-3 LRS 算法

input : 粒子位置向量 X_k^t

output: 新的粒子位置向量 $X_k^{t\prime}$

设置搜索次数 $N=0$；最大邻域搜索次数 $N_{\max}=5$；变异算子编号 $K=1$；最大变异算子编号 $K_{\max}=4$；

while $N \le N_{\max}$ **do**

  $K=1$;

  while $K \le K_{\max}$ **do**

    if *rand* $< \frac{1}{K_{\max}}$ **then**

      $X_k^{t\prime}=M_K(X_k^t)$;

      if $f\left(X_k^{t\prime}\right) < f(X_k^t)$ **then**

        $X_k^t=X_k^{t\prime}$;

        $K=1$;

      end

    end

    $K=K+1$;

  end

  $N=N+1$

end

LRS 算法步骤如下：

步骤 1: 设置初始搜索次数 N、最大邻域搜索次数 $N_{\max}$ 和初始变异算子编号 K 和最大变异算子编号 $K_{\max}$。

步骤 2: 判断搜索次数是否小于最大邻域搜索次数，若不是，结束算法。

步骤 3: 判断算子编号是否小于最大变异算子编号，若不是，搜索等级加 1，转至步骤 2。

步骤 4: 根据产生的随机数是否小于进入此种变异算子的阈值，若小于，则根据算子编号进行局部随机搜索，计算新粒子的适应值，变异算子编号加 1。

步骤 5: 判断新粒子适应值是否小于原粒子适应值，若不是，转至步骤 3；若是，新粒子 k' 代替原粒子 k，变异算子编号置为 1。

3.3.5 SA-LRS-PSO 任务分配算法

模拟退火-随机局部搜索-粒子群算法以粒子群算法作为主要算法流程，在粒子更新前加入了模拟退火局部随机搜索启动机制、局部随机搜索，算法流程图如下

图3-4所示。

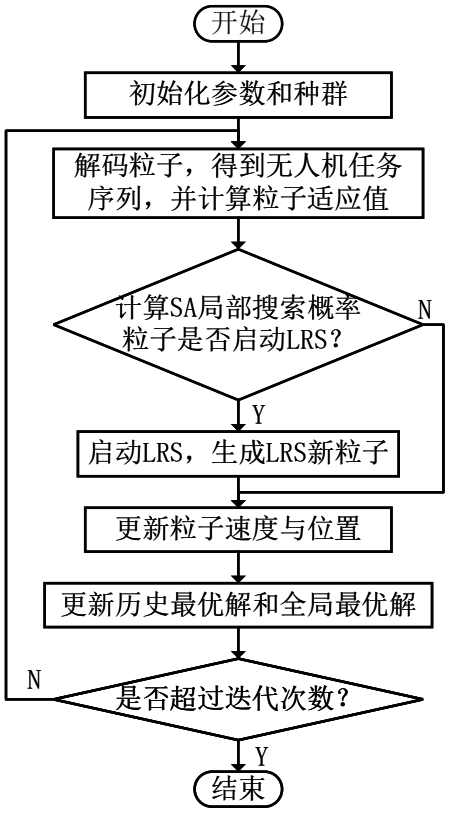


图 3-4 SA-LRS-PSO 算法流程图

从图3-4可以看出，通过 SA 机制和 LRS 算法产生的新粒子都参与了粒子的筛选和更新过程，可以有效地记录全局最优任务分配解。优化后的 PSO 算法步骤如下:

步骤 1: 初始化参数和种群

设置种群规模、无人机即目标物信息、最大迭代次数、粒子维数、速度和位置的上下界、惯性权值的上下界等基本信息。随机生成每个粒子的初始位置和初始速度。

步骤 2: 随机生成粒子，通过解码得到任务序列，计算其适应度值。

根据粒子维数、速度上界和位置下界随机生成粒子。然后使用 3.3.1 中所设计的解码策略生成 N 个任务序列。

步骤 3: 根据粒子的适应度和 SA 局部搜索起始概率准则，采用 LRS 算法进行局部邻域搜索。

计算 SA 局部搜索概率，判断粒子是否需要采用 LRS 算法。如果是，启动 LRS 算法，寻找更好的解突破当时去全局最优。

步骤 4: 更新粒子的位置和速度，寻找历史最优解和全局最优解。

根据式 (3-14)-(3-18) 更新 N 个粒子的位置、速度、历史最优解和全局最优。

步骤 5: 确定结束条件。

确定迭代是否超过最大值。如果是，则保存全局最优值和任务顺序。否则，请转到步骤 2。

3.4 仿真实验与分析

本文任务分配问题以无人机群协同攻击地面目标作为背景，提出了一种 SA-LRS-PSO 算法，为验证其有效性和性能优越性，本文构造了一个无人机群协同攻击仿真实例，验证平台为具有 Inter Core i9-10900K 处理器、128GB 内存的 PC 机，所有算法在 Matlab R2019b 平台编译运行。

3.4.1 参数设置

实验任务分配场景设置为 3 架 UAV 协同攻击 10 个地面目标，仿真场景范围为 100*100 的区域，仿真场景示意图如图3-5所示。

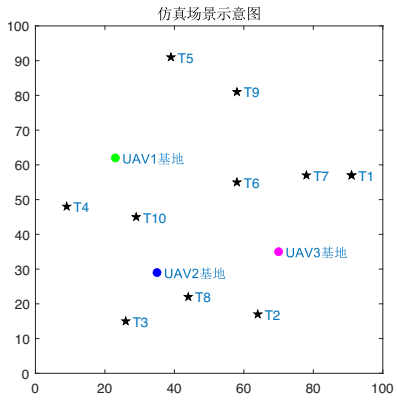


图 3-5 仿真场景示意图

由于无人机自身性能及飞行能力不同，无人机具体参数如下表3-5，表中包括无人机编号，无人机基地位置，无人机打击成功率、最大执行任务数和无人机飞行速度。

表 3-5 无人机参数表

无人机编号	基地位置	攻击成功率	最大执行任务数	飞行速度
1	(23,62)	0.9	5	5
2	(35,29)	0.8	5	4
3	(70,35)	0.85	6	6

仿真区域中包含 10 个目标物，目标点的参数如下表3-6所示，表中包括目标物编号、目标物位置、目标物价值、打击目标物的时间窗口、打击目标物所用时间和所需 UAV 架数。当目标物所需 UAV 架数大于 1 时，表明此目标物需要多架无人机协同打击。

表 3-6 目标物参数表

目标物编号	位置	价值	时间窗口	用时	所需 UAV 架数
1	(91,57)	20	[10,30]	1	1
2	(63,17)	30	[20,30]	2	2
3	(26,15)	25	[15,40]	1	1
4	(9,48)	40	[5,75]	2	1
5	(39,91)	50	[10,60]	1	2
6	(58,55)	30	[20,60]	2	1
7	(78,57)	25	[5,20]	1	1
8	(44,22)	40	[10,40]	2	2
9	(58,81)	25	[20,80]	1	1
10	(29,45)	30	[5,80]	2	1

SA 局部搜索概率中所需参数设置如下：比例系数 α 为 1.6，初始温度 T_0 为 200，降温速度 γ 为 0.98。

3.4.2 有效性分析

针对不同的仿真场景，设置式 (3-8) 中的比例系数，具体设置如下表3-7所示。

表 3-7 场景参数表

场景编号	a_1	a_2	a_3	a_4
1	1	1	1	1
2	1	0.4	1	1
3	0.7	1	1	1
4	1	1	0.6	0.3

所有任务场景都采用本章所提算法进行寻优搜索，迭代次数为 800 代，四个场景的最终平均适应值曲线如下图3-6所示。

由图3-6所示，可以看出不同任务场景下的不同比例系数对最终寻优结果只是数值上的变化，整体搜索过程中的变化趋势是基本一致的，所以比例系数的变化对算法的寻优趋势是没有太大影响的。

根据以上对目标函数中比例系数的讨论结果，同时均衡各个部分对目标函数的影响，设置各比例系数为 1。为了验证本章算法的有效性，采用本章所提算法进

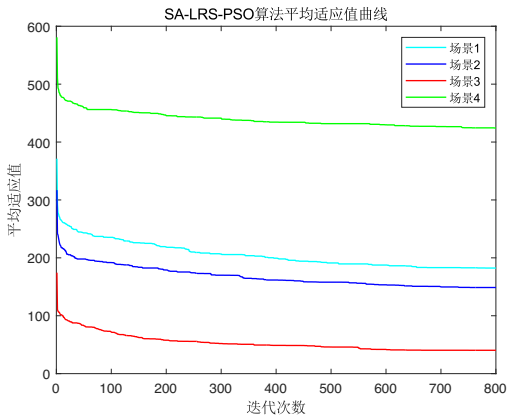


图 3-6 不同比例系数下的平均适应值曲线

行任务分配的结果如表3-8所示，其适应度为所提算法所能搜索到的最小值。

表 3-8 SA-LRS-PSO 任务分配结果

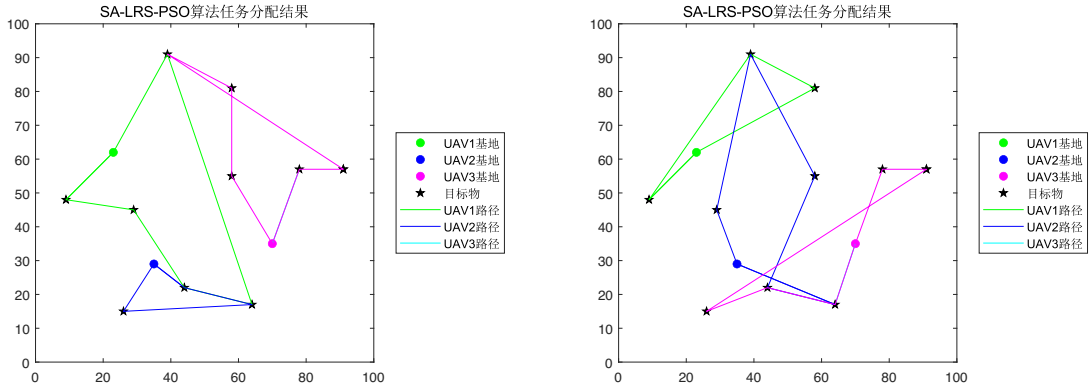
UAV 序号	任务序列	适应值
1	4-10-8-2-5	
2	8-2-3	141.99
3	7-1-5-9-6	

如表3-8所示，观察任务分配结果，可以看出任务序列未超过无人机最大执行任务数，且对于多目标任务，未出现一架无人机重复执行多目标任务和协同攻击时的死锁现象，例如本研究设置三架无人机的最大执行任务数分别为 5、5、6，而任务分配结果中，三架无人机的最大执行任务数为 5、3、5，未超过限制，满足约束条件。而针对多打击目标，T_5 由无人机 U_1 和 U_3 共同打击，T_2 和 T_8 由 U_1 和 U_2 共同执行，且执行顺序都为先执行 T_8，再执行 T_2，此执行顺序不会造成整个无人机群任务序列的“死锁”现象。

为进一步形象化展示任务分配结果，将表3-8任务序列和适应值为 239.05 的任务序列所对应的任务分配结果通过具体地图展示，示意图如图3-7所示。

如图3-7(a) 所示，适应值较低的任务分配结果中，无人机优先完成距离各自无人机基地较近的任务，且任务序列更具有流畅性；而图3-7(b) 所示的任务分配结果分布，无人机被分配的任务距离自己的基地较远且任务序列的流畅性较差，能从无人机基本的路径中看出航线出现交叉的情况较多，这样可能会带来航程上的冗余消耗。

例如对于无人机 U_3 来说，当任务分配结果的适应值较低时，U_2 被分配到的



(a) (b)

图 3-7 SA-LRS-PSO 任务分配结果分布图 (a) $f=141.99$；(b) $f=239.05$

任务都距离其基地较近，其执行任务的过程中，航线不会出现交叉情况；而当任务分配结果的适应值较高时，U_2 被分配到的任务 T_5 距离其基地较远，且还被分配了其他多打击任务 T_2 和 T_8，由于多打击任务需要考虑是否出现死锁现象，并且需要多架无人机同时进行打击任务，则在执行任务的过程中会更多的出现其他无人机等待 U_2 的情况，时间窗口损失函数值随之增高，可以看出此种任务分配解的不合理性。

3.4.3 对比分析

为证明 LRS-VNS-PSO 算法的性能优越性，分别与标准 PSO 算法，VNS-PSO 算法，和在解码过程中对每个粒子都进行局部随机搜索的 LRS-PSO 算法进行对比。为了保证比较实验的公平性，比较算法的种群大小为 20，编解码算法一致，初始种群是随机生成的。4 种算法在仿真实验中的单次适应度曲线如下图3-8所示。

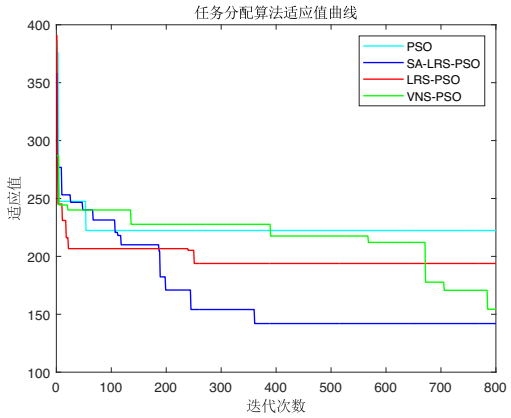


图 3-8 单次实验适应值曲线

从图3-8可以看出 PSO 算法搜索全局最优的能力较差，在迭代到 54 代时就进入了局部收敛，最终搜寻到了全局最优解的适应值为 222.4。VNS-PSO 算法跳出局部最优的能力较强，因为该算法的算法框架中加入了变邻域搜索，从图中可以看出，即使其他算法都进入了较为稳定的收敛状态时，VNS-PSO 算法分别在第 391、569、674、709、787 代跳出了当前局部最优，最后搜寻到的最优解的适应值为 154.3。LRS-PSO 算法的算法框架中对每一个粒子都进行了局部随机搜索，所以该算法前期搜寻局部最优解的能力较强，最后搜寻到的最优解的适应值为 193.9。本章所提的 SA-LRS-PSO 算法在前期迭代的过程中不断跳出局部最优点，迭代至第 361 代时，收敛至最后的全局最优点，该点适应值为 141.99，算法 PSO、VNS-PSO、LRS-PSO 所搜寻出的全局最优点的适应值分别比本章所提算法搜寻到的最低适应值高 23.39%、6.42%、1.64%。

为了避免算法在一个仿真实验解中存在随机性，对 4 种算法进行了 20 次仿真实验，在实验过程中记录每种算法的适应值曲线。4 种算法在仿真实验中的平均适应值曲线如图3-9所示。

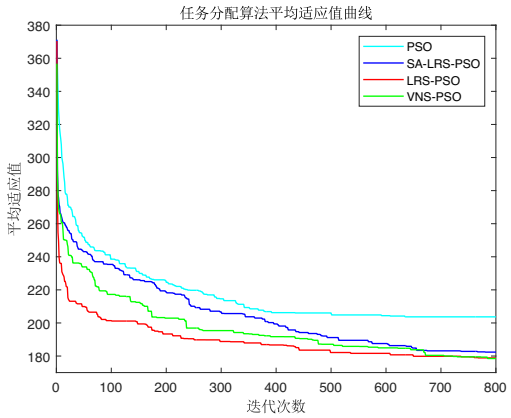


图 3-9 平均适应值曲线 (横坐标为迭代次数)

如图3-9所示，PSO 算法前期的收敛速度较慢，最终搜寻到的平均全局最优点的适应值也较高，其平均适应值为 202.31。LRS-PSO 算法因为其算法框架中对每一个粒子进行了局部随机搜索，所以在前期迭代过程中收俩速度最快，迭代至第 26 代时，平均适应值曲线已收敛至 213.1，最后搜寻的平均最优点的适应值为 178.88。本章所提的 SA-LRS-PSO 算法在前期收敛过程中收敛速度优于 PSO 算法，但次于 LRS-PSO 算法，该算法本身是对粒子进行筛选，判断其是否需要进一步进行局部随机搜索，所以其最后搜寻到的平均最优适应值与 LRS-PSO 算法数值上相差较小，为 182.38。VNS-PSO 算法在前期迭代过程中收敛速度仅次于 LRS-PSO 算法，该算法在迭代至第 175 代时，平均适应值已达到 203.6，其最终搜寻到的最优

点的平均适应值为 178.50，算法 PSO、LRS-PSO、SA-LRS-PSO 所搜寻出的全局最优点的平均适应值分别比 VNS-PSO 算法搜寻到的全局最优点的平均适应值高 13.33%、0.21%、2.17%。

为了验证提出的 SA-LRS-PSO 算法具有平衡跳出局部收敛与计算开销的能力，统计 4 种算法的最优解、平均适应值和每次迭代的平均时间，共进行 20 次实验，并计算 4 种算法所搜寻到的适应值的标准差，结果统计如下表3-9所示。

表 3-9 适应值仿真结果对比表

算法	最优解	平均适应值	平均每代计算时间 (s)	适应值标准差
PSO	175.20	202.31	0.67	16.54
VNS-PSO	151.10	178.50	5.17	18.23
LRS-PSO	144.32	178.88	20.48	11.26
SA-LRS-PSO	141.99	182.38	1.66	18.80

如表3-9所示，PSO 算法因为其算法流程简单，所以其搜寻到的最优解的平均适应值最大，为 202.31，但其平均每代计算时间最短，为 0.67s。VNS-PSO 算法的平均适应值是四个算法中最低，但其能搜寻到的最优解不是最低的，而且由于设置的开启 VNS 的机制，导致其平均每代迭代时间为 5.17s，为 PSO 算法的 7.71 倍。而由于 LRS-PSO 算法对每个粒子都进行了局部随机搜索，导致其平均每代计算最高，为 20.48s，是 PSO 算法的 30.57 倍。本章所提的 SA-LRS-PSO 算法其搜寻的最优解为 141.99，为四个算法中最低的，且其平均每代计算时间仅次于 PSO 算法，数值为 1.66s，仅为 PSO 算法的 2.48 倍，是优化 PSO 算法中增加计算消耗最小的。

其次，如表3-9所示，LRS-PSO 算法的标准差是最小的，由于其对每个粒子都进行了邻域搜索，所以算法的稳定性较好；其余四种算法适应值相差不大，其中 PSO 算法跳出局部最优的能力较差，最终搜寻结果趋于局部最优，所以其标准差位列第二；而加入了邻域搜索的 VNS-PSO 和 SA-LRS-PSO 算法的标准差稍逊于其余两种算法，且这两种算法的标准差之间相差不大。

为更好的直观展示四种算法的计算消耗，在实验过程中记录每种算法搜寻全局最优值的时间消耗，4 种算法在仿真实验中以时间为横坐标的平均适应值曲线如图3-10所示。

从图3-10可以看出，PSO 算法由于算法流程简单，在 533.3s 时已结束算法。LRS-PSO 算法的计算消耗是最大的，由于图中空间有限，仅展示 0-4000s 的适应值曲线变化，但是 LRS-PSO 算法在 16405s 时收敛至 178.88，即使从图3-9得出该算法前期收敛较快的结论，但以时间为横坐标时，可以明显看出其收敛速度不是最快的。SA-LRS-PSO 算法的平均适应值曲线收敛至 182.4 时需要 1331s，而 VNS-PSO

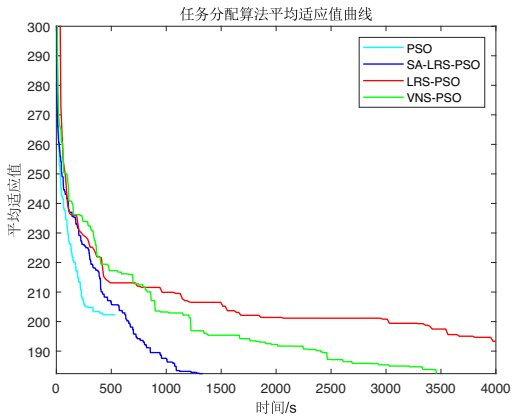


图 3-10 平均适应值曲线 (横坐标为时间)

算法收敛至同等平均适应值则需要 3463s，比 SA-LRS-PSO 算法高出 160.18%。由此可以明显看出，虽然 VNS-PSO 的全局寻优能力略微高于 SA-LRS-PSO 算法，但是以明显增加计算消耗为代价的。

最后为分析 4 种算法的稳定性，绘制 4 种算法重复实验搜寻到的全局最优点的适应值的分布盒图，如图3-11所示。

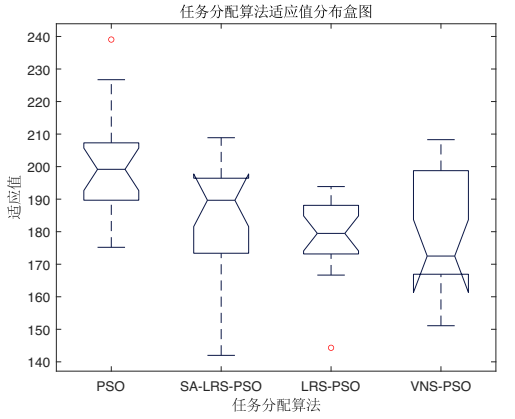


图 3-11 任务分配算法适应值分布盒图

从图3-11可以看出，PSO 算法所搜索到的全局最优点的适应值较大，而且上限较高；因为 LRS-PSO 算法对每个粒子进行了局部随机搜索，所以其最终适应值的分布比较统一，算法稳定性较好；SA-LRS-PSO 算法和 VNS-PSO 算法所搜寻的全局最优点的适应值分布较为类似，VNS-PSO 算法的平均适应值较低，而 SA-LRS-PSO 算法所搜寻的最低适应值较小。

通过仿真实验综合各算法的仿真曲线，并对多次仿真的最终结果进行分析，各种算法可以在规定的迭代次数内达到收敛。进一步分析可以得到:

PSO 算法为解决组合优化问题的通用方法，但是全局搜索能力较差，非常容易

陷入局部最优；VNS-PSO 算法跳出局部最优的能力是所讨论的四种算法中最好的，但所增加的计算消耗十分明显，每次迭代所需时间也较长；LRS-PSO 算法由于对每个粒子都进行了局部邻域搜索，所以算法稳定性是最佳的，但所需的计算消耗太大，不利于实际应用；本文所提的 SA-LRS-PSO 算法虽然全局寻优能力略微逊于 VNS-PSO 算法，但也可以不断跳出局部最优，其计算消耗明显小于 VNS-PSO 和 LRS-PSO 算法。

基于上述实验结果和分析，SA-LRS-PSO 算法能够在保持较快迭代速度的同时专注于全局搜索空间的挖掘，使其在处理针对协同打击任务场景的无人机群任务分配问题时获得良好的结果。

3.5 本章小结

本章解决了无人机群协同攻击任务分配问题，本章建立了综合考虑飞行航程、任务总体完成时间、时间窗口和攻击收益的多任务分配模型，并提出了 SA-LRS-PSO 任务分配算法。首先设计了一种同时融合无人机、目标类型、任务顺序等信息的单链粒子编解码算法，实现了粒子离散化。其次避免算法陷入局部收敛，在 PSO 粒子更新公式部分，引入自适应学习因子和社会学习因子，同时设计了基于模拟退火机制的局部随机搜索启动机制，并采用 3 种变异算子和 1 种交叉算子对粒子进行局部随机搜索，提高算法全局搜索能力。通过数值实验验证了所提算法能够在保持较快迭代速度的同时专注于全局搜索空间的挖掘。