A Deadlock-free Hybrid Estimation of Distribution Algorithm for Cooperative Multi-UAV Task Assignment with Temporally Coupled Constraints

**Ruipeng Zhang, Yanxiang Feng, Yikang Yang\* Xiaoling Li**

# Basics of Petri nets

A PN is a four-tuple *N* = (*P*, *T*, *F*, *M*), where *P* is the set of places, *T* is the set of transitions, and *P* and *T* are finite and disjoint sets. *F* ⊆ (*P* × *T*) ∪ (*T* × *P*) is the set of directed arcs.

For any node *x*∈ *P*∪*T*, •*x*={*y* ∈ *P* ∪ *T* | (*y*, *x*) ∈ *F*} represents the preset of *x*, and *x*• = {*y* ∈ *P* ∪ *T* | (*x*, *y*) ∈ *F*} means the post-set of *x*. A marking or state of *N* is a mapping *M*: *P* → ℤ+, where ℤ+ = {0,1,2,…}. Given a place *p*∈*P* and a marking *M*, *M*(*p*) denotes the number of tokens in *p* at *M*. A Petri net *N* with an initial marking *M*0 is denoted by (*N*, *M*0).

For a given transition *t*∈*T*, if ∀*p*∈•*t*, *M*(*p*)>0, we say that *t* is *enabled* at marking *M* and is denoted by *M*[*t* >. An enabled transition *t* can be *fired* at *M*. We let the marking *M*′be the new state of Petri net*N* after *t* fired and is denoted by *M*[*t* >*M*′. The change in tokens in place *p* can be divided into three cases: for ∀*p*∈•*t\ t*•, *M*′(*p*) = *M*(*p*) −1; for ∀*p*∈*t*•*\* •*t*, *M*′(*p*) = *M*(*p*) + 1; otherwise, *M*′(*p*) = *M*(*p*). A sequence of transitions α = *t*1*t*2…*tk* is *feasible* at marking *M* if there exists *Mi* [*ti*>*Mi+1*, *i* = 1, 2, …, *k*, where *M*1 = *M.*

## Swap operation

**Algorithm SO**

**Input**: an individual Δ = {λ; μ} and the probability *Pswap*;

**Output**: the new individual Δ′;

1. **if** rand(0, 1) < *Pswap*
2. Choose two different tasks *h* and *j* from λ;
3. Swap *h* and *j* in λ;
4. Perform AlgorithmIAD on Δ, let Δ′ be the resulting individual;
5. **else**
6. Δ′ = Δ;
7. **end**
8. Output Δ′.

## Insert operation

**Algorithm IO**

**Input**: an individual Δ = {λ; μ} and the probability *Pinsert*;

**Output**: the new individual Δ′;

1. **if** rand(0, 1) < *Pinsert*
2. Choose two different tasks *h* and *j* from λ;
3. Insert task *h* into the position of task *j*, while the elements after *j* in λ are

shifted backward.

1. Perform AlgorithmIAD on Δ, let Δ′ be the resulting individual;
2. **else**
3. Δ′ = Δ;
4. **end**
5. Output Δ′.

# Reorganization operation

**Algorithm RO**

**Input**: an individual Δ = {λ; μ}, antibody set Ω*a*, and the probability *Preorg*;

**Output**: the new individual Δ′;

1. **if** rand(0, 1) < *Preorg*
2. Choose a different individual Δ*y* = {λ*y*; μ*y*}from Ω*a*;
3. Set μ = μ*y* in Δ;
4. Perform AlgorithmIAD on Δ, let Δ′ be the resulting individual;
5. **else**
6. Δ′ = Δ;
7. **end**
8. Output Δ′.