Estimation of computational complexity for sub-optimal swarm control in non-cooperative games

O.M. Kiselev
ok@ufanet.ru

Institute of Mathematics,
Ufa Federal Research Center,
Russian Acad. of Sci.

September 09, 2020
DCNAIR 2020

Short Intro

- In this work formal mathematical rules are mainly based on rules for a championship of computer players for Agar IO
  game on a platform aicups.ru organized by **MailRu Group** *https://github.com/MailRuChamps/miniaicups/tree/master/agario*.
- Hints for a strategy one can see in the paper by A. Dichkovskii, his robot is a champion of these competitions see **A.Dichkovskii**, *https://habr.com/ru/post/420737*.
- Review of problems of swarms control one can find in **L.Bayindir**, *A review of swarm robotics tasks*, **Neurocomputing**, 2015.

Typical obstacles for the swarm control

- A lot of typical objects is located in the phase space.
- The phase space has a lot of dimensions.
- The best strategy for the swarm is not the best for each of the swarm objects.
- One needs a lot of computing in real-time.
- The computational and memory limitations are ordinary for the swarm control.

Objects

- A swarm consists of N uniform objects.
- Any object be determined by a set of *configuration parameters* $\mathsf{X} \in \mathcal{X} \subset \mathbb{F}^n$, where $\mathbb{F}$ is a set of **double float** numbers
- *Internal parameters* $\mathsf{Y} \in \mathcal{Y} \subset \{\mathbb{N}^l \times \mathbb{F}^m\}$.
- *Control vector* $\mathsf{u} \in \mathcal{U}$, where $\mathcal{U}$ is a set of allowed values for the control vector.

Assume that the configuration space has a metric.

Clusters

The objects, which are situated near each other on a distance R during more than k control times, combine into a *cluster*.
The cluster is a consistent object in the space $\{\mathcal{X}, \mathcal{Y}\}$.
The number k of unit objects, joined into the cluster, is one of components of the internal parameters $\mathrm{Y}^{(1)} = k$.
An information about the environment in the configuration space is allowable for the cluster on the distance not more than $\rho = \rho(k)$, where k is a number of the units, which joint into the cluster. For the unit object $\mathrm{Y}^{(1)} = 1$.

Phase and configuration spaces

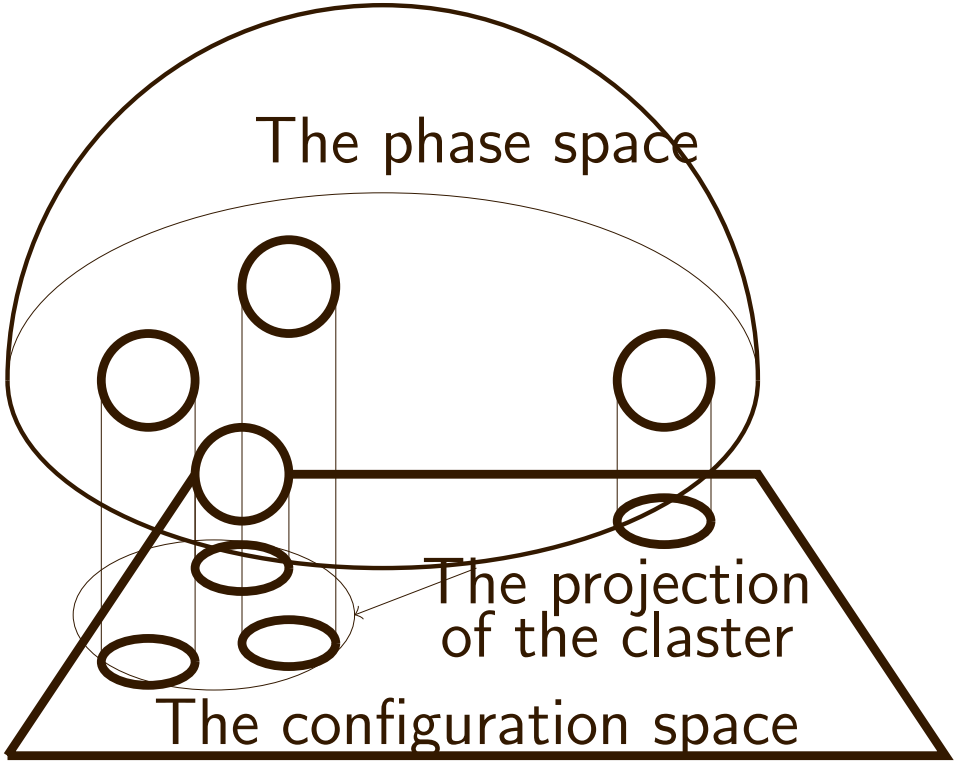

Figure: The phase space is $\mathcal{X} \cup \mathcal{Y}$. The left objects of the swarm can join in a cluster and one object on the right is along.

Swarm control

Let the control be a discrete, then sequence of steps one can consider as a reflection of the set $\mathcal{X}$ into themselves: $\{X_{i+1}, Y_{i+1}\} = F(X_i, Y_i, u_i), \quad i \in \{\overline{1, \ldots, I}\}, \quad I \in \mathbb{N}$. A computational complicity of the reflection will be defined as $N(F)$.

The prize set

The polygon contains a *prize set*.

When any of the swarm unit has arrived to a point of the *prize set* of the configuration space $\mathcal{B} \in \mathcal{X}$. The swarm gets some prize points.

Define an objective function to obtain maximal points over l steps of the game.

The prize points are added for the whole swarm. There fore for any object of the swarm such game is *cooperative with non-zero sum*.

Assume that κ competitive swarms present in the configuration space. The game for such swarms are *non-cooperative*.

The prize set

The polygon contains a *prize set*.
When any of the swarm unit has arrived to a point of the *prize set* of the configuration space $\mathcal{B} \in \mathcal{X}$. The swarm gets some prize points.

Define an objective function to obtain maximal points over l steps of the game.

The prize points are added for the whole swarm. There fore for any object of the swarm such game is *cooperative with non-zero sum*.

Assume that κ competitive swarms present in the configuration space. The game for such swarms are *non-cooperative*.

The prize set

The polygon contains a *prize set*.
When any of the swarm unit has arrived to a point of the *prize set* of the configuration space $\mathcal{B} \in \mathcal{X}$. The swarm gets some prize points.
Define an objective function to obtain maximal points over l steps of the game.
The prize points are added for the whole swarm. There fore for any object of the swarm such game is *cooperative with non-zero sum*.
Assume that κ competitive swarms present in the configuration space. The game for such swarms are *non-cooperative*.

The prize set

The polygon contains a *prize set*.
When any of the swarm unit has arrived to a point of the *prize set* of the configuration space $\mathcal{B} \in \mathcal{X}$. The swarm gets some prize points.
Define an objective function to obtain maximal points over l steps of the game.
The prize points are added for the whole swarm. There fore for any object of the swarm such game is *cooperative with non-zero sum*.
Assume that κ competitive swarms present in the configuration space. The game for such swarms are *non-cooperative*.

The prize set

The polygon contains a *prize set*.
When any of the swarm unit has arrived to a point of the *prize set* of the configuration space $\mathcal{B} \in \mathcal{X}$. The swarm gets some prize points.
Define an objective function to obtain maximal points over l steps of the game.
The prize points are added for the whole swarm. There fore for any object of the swarm such game is *cooperative with non-zero sum*.
Assume that κ competitive swarms present in the configuration space. The game for such swarms are *non-cooperative*.

The penalty set

In case of collision of clusters of competitive swarms the bigger cluster absorbs the smaller one.

Thus a *penalty set* appears in the configuration space like $\mathcal{P}_i \in \mathcal{X}$. This set $\mathcal{P}_i \in \mathcal{X}$ changes locations and properties on different step i. This set contains intervals of the configuration space, where the clusters, bigger than clusters of swarm $\mathbf{1}$ are located, which can hit into this set.

The configuration space contains not only a penalty set, but also an additional prize set $\tilde{\mathcal{B}}_i \in \mathcal{X}$. This set is contained by smaller clusters of competitive swarms, than clusters of swarm $\mathbf{1}$, which can hit into the intervals of $\tilde{\mathcal{B}}_i$.

The penalty set

In case of collision of clusters of competitive swarms the bigger cluster absorbs the smaller one.
Thus a *penalty set* appears in the configuration space like $\mathcal{P}_i \in \mathcal{X}$. This set $\mathcal{P}_i \in \mathcal{X}$ changes locations and properties on different step i. This set contains intervals of the configuration space, where the clusters, bigger than clusters of swarm $\mathbf{1}$ are located, which can hit into this set.
The configuration space contains not only a penalty set, but also an additional prize set $\tilde{\mathcal{B}}_i \in \mathcal{X}$. This set is contained by smaller clusters of competitive swarms, than clusters of swarm $\mathbf{1}$, which can hit into the intervals of $\tilde{\mathcal{B}}_i$.

The penalty set

In case of collision of clusters of competitive swarms the bigger cluster absorbs the smaller one.
Thus a *penalty set* appears in the configuration space like $\mathcal{P}_i \in \mathcal{X}$. This set $\mathcal{P}_i \in \mathcal{X}$ changes locations and properties on different step i. This set contains intervals of the configuration space, where the clusters, bigger than clusters of swarm $\mathbf{1}$ are located, which can hit into this set.
The configuration space contains not only a penalty set, but also an additional prize set $\tilde{\mathcal{B}}_i \in \mathcal{X}$. This set is contained by smaller clusters of competitive swarms, than clusters of swarm $\mathbf{1}$, which can hit into the intervals of $\tilde{\mathcal{B}}_i$.

The target function

Let us assume suboptimal algorithm, which maximizes a target function of swarm $\mathbf{1}$ after q steps the control.

Define by b_i a number of the prize points of swarm $\mathbf{1}$ on i step and the penalties define by p_i, then one constructs the target function as follows:

$$H_{i,q} = \sum_{k=i}^{i+q} (b_k - p_k).$$

It is important to say that the target function is not smooth.

The target function

Let us assume suboptimal algorithm, which maximizes a target function of swarm $\mathbf{1}$ after q steps the control.
Define by b_i a number of the prize points of swarm $\mathbf{1}$ on i step and the penalties define by p_i, then one constructs the target function as follows:

$$H_{i,q} = \sum_{k=i}^{i+q} (b_k - p_k).$$

It is important to say that the target function is not smooth.

The target function

Let us assume suboptimal algorithm, which maximizes a target function of swarm $\mathbf{1}$ after q steps the control.
Define by b_i a number of the prize points of swarm $\mathbf{1}$ on i step and the penalties define by p_i, then one constructs the target function as follows:

$$H_{i,q} = \sum_{k=i}^{i+q} (b_k - p_k).$$

It is important to say that the target function is not smooth.

Cluster dynamics

As a control we choose a point in the configuration space, which will be a target point for all objects of swarm **1**. Besides we assume that the movement speed to the target of single cluster depends on its size. During one step smaller clusters move faster than the bigger one.

If the cluster consists of n objects ($n > 1$), this cluster can be divided in two clusters with n_1 and n_2 objects with given proportionality. For example in case of division on two halves one gets $n_1 + n_2 = n$ and $n_1 = [n/2]$. New clusters obtain properties for internal parameters $\mathcal{Y}$ over rules: $\mathcal{Y}_{i+1}|_{\mathcal{Y}^{(1)}=n_1} = G_1(\mathcal{Y}_i), \quad \mathcal{Y}_{i+1}|_{\mathcal{Y}^{(1)}=n_2} = G_2(\mathcal{Y}_i)$. If the target point lies in the linear span of the swarm, the diameter of the linear span decreases step-by-step. If this process lasts l steps and $l > k$ for some given k, then the clusters merge in one bigger cluster.

Cluster dynamics

As a control we choose a point in the configuration space, which will be a target point for all objects of swarm **1**. Besides we assume that the movement speed to the target of single cluster depends on its size. During one step smaller clusters move faster than the bigger one.

If the cluster consists of n objects ($n > 1$), this cluster can be divided in two clusters with n_1 and n_2 objects with given proportionality. For example in case of division on two halves one gets $n_1 + n_2 = n$ and $n_1 = [n/2]$. New clusters obtain properties for internal parameters $\mathcal{Y}$ over rules:

$\mathcal{Y}_{i+1}|_{\mathcal{Y}^{(1)}=n_1} = G_1(\mathcal{Y}_i), \quad \mathcal{Y}_{i+1}|_{\mathcal{Y}^{(1)}=n_2} = G_2(\mathcal{Y}_i)$. If the target point lies in the linear span of the swarm, the diameter of the linear span decreases step-by-step. If this process lasts l steps and $l > k$ for some given k, then the clusters merge in one bigger cluster.

Cluster dynamics

As a control we choose a point in the configuration space, which will be a target point for all objects of swarm **1**. Besides we assume that the movement speed to the target of single cluster depends on its size. During one step smaller clusters move faster than the bigger one.
If the cluster consists of n objects ($n > 1$), this cluster can be divided in two clusters with n_1 and n_2 objects with given proportionality. For example in case of division on two halves one gets $n_1 + n_2 = n$ and $n_1 = [n/2]$. New clusters obtain properties for internal parameters $\mathcal{Y}$ over rules:
$\mathcal{Y}_{i+1}|_{\mathcal{Y}^{(1)}=n_1} = G_1(\mathcal{Y}_i), \quad \mathcal{Y}_{i+1}|_{\mathcal{Y}^{(1)}=n_2} = G_2(\mathcal{Y}_i)$. If the target point lies in the linear span of the swarm, the diameter of the linear span decreases step-by-step. If this process lasts l steps and $l > k$ for some given k, then the clusters merge in one bigger cluster.

Compression and stretching

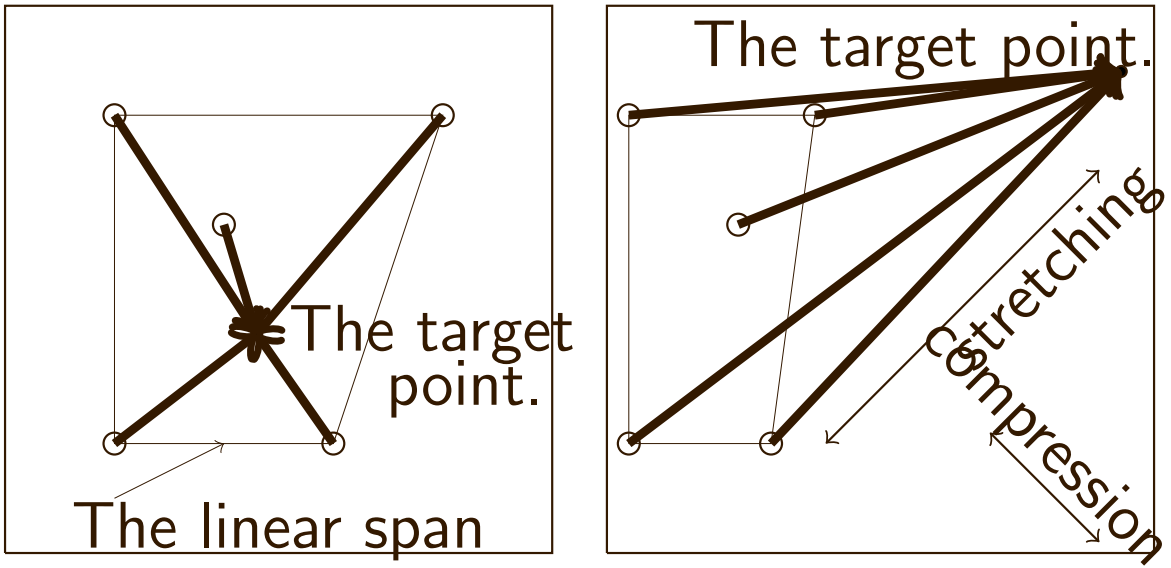

Figure: On the left picture the target point lies in the linear span of the swarm. On the right one the target point lies out of the span.

A sequences of the stretching and compression produce the *mixing* of the swarm in the phase space and growing the perimeter of the linear span.

The computing complexity

Let us consider a case without competitive swarms. Define the measure of the allowable control points by $p = \mathrm{mes}(\mathcal{U})$. The computational complexity of target function for object of the swarm alone:

$$N(H_{pq}) = O(N^q(\mathrm{F})) \equiv O(n^q).$$

Here $n = N(\mathrm{F})$.

A brute-force search of strategies on one step of control is $O(np)$. The computational complexity grows polynomially for the search of strategies for q steps of the control:

$$C = O((np)^q).$$

The computing complexity

Let us consider a case without competitive swarms. Define the measure of the allowable control points by $p = \mathrm{mes}(\mathcal{U})$. The computational complexity of target function for object of the swarm alone:

$$N(H_{pq}) = O(N^q(\mathrm{F})) \equiv O(n^q).$$

Here $n = N(\mathrm{F})$.
A brute-force search of strategies on one step of control is $O(np)$. The computational complexity grows polynomially for the search of strategies for q steps of the control:

$$C = O((np)^q).$$

The computing complexity

Let us consider a case without competitive swarms. Define the measure of the allowable control points by $p = \mathrm{mes}(\mathcal{U})$. The computational complexity of target function for object of the swarm alone:

$$N(H_{pq}) = O(N^q(\mathrm{F})) \equiv O(n^q).$$

Here $n = N(\mathrm{F})$.
A brute-force search of strategies on one step of control is $O(np)$. The computational complexity grows polynomially for the search of strategies for q steps of the control:

$$C = O((np)^q).$$

The partial searching on the grid

- A part of components for the control vector u_k belongs to set $\mathcal{U}$. Really, it is too large for brute-force searching over all $\mathcal{U}$.
- We consider partial searching on the grid in the phase space for the swarm objects and for the control points. On such grid we search a maxima for the target function. Then in the ball with a center in this conditional maxima we take smaller grid and search the maxima again.
- Let us define the quantities of the points in the grid of k level by p_k. Then the iterative process has the computational complexity as follows:

$$C_{kr} = O((np_k)^{rq}),$$

where r is the number of iterations of localisation.

The partial searching on the grid

- A part of components for the control vector u_k belongs to set $\mathcal{U}$. Really, it is too large for brute-force searching over all $\mathcal{U}$.
- We consider partial searching on the grid in the phase space for the swarm objects and for the control points. On such grid we search a maxima for the target function. Then in the ball with a center in this conditional maxima we take smaller grid and search the maxima again.
- Let us define the quantities of the points in the grid of k level by p_k. Then the iterative process has the computational complexity as follows:

$$C_{kr} = O((np_k)^{rq}),$$

where r is the number of iterations of localisation.

The partial searching on the grid

- A part of components for the control vector u_k belongs to set $\mathcal{U}$. Really, it is too large for brute-force searching over all $\mathcal{U}$.
- We consider partial searching on the grid in the phase space for the swarm objects and for the control points. On such grid we search a maxima for the target function. Then in the ball with a center in this conditional maxima we take smaller grid and search the maxima again.
- Let us define the quantities of the points in the grid of k level by p_k. Then the iterative process has the computational complexity as follows:

$$C_{kr} = O((np_k)^{rq}),$$

where r is the number of iterations of localisation.

Brute force and partial searching

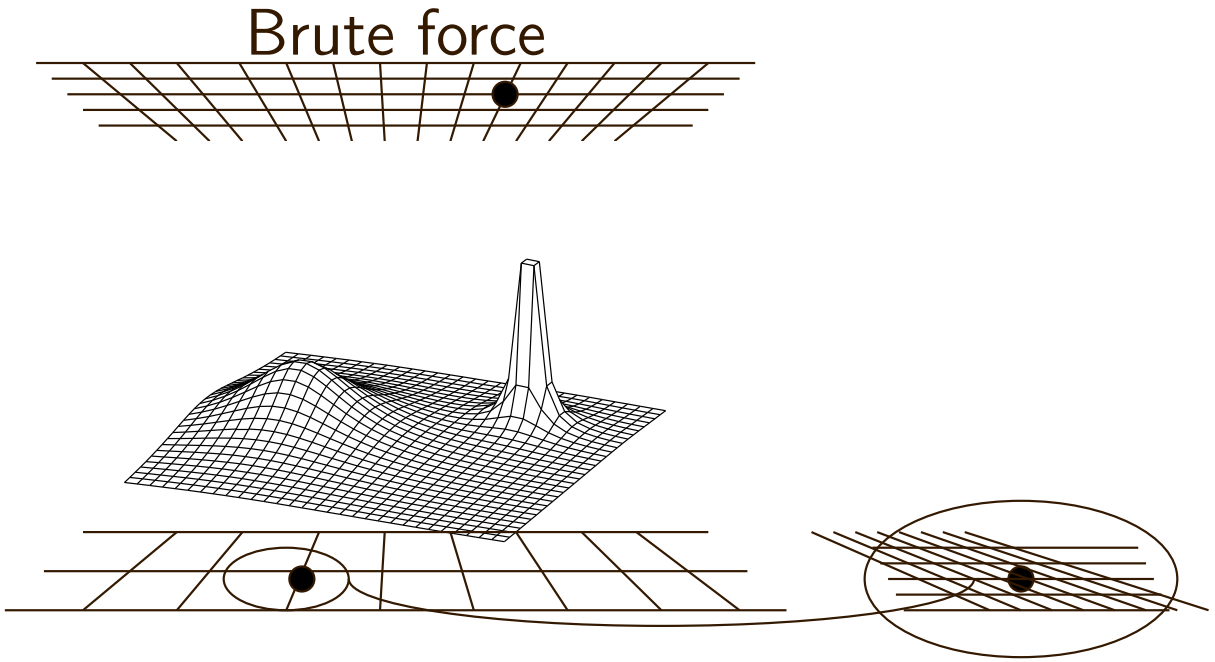

Figure: The projection on the brute force grid is shown on the top. The partial search grid is shown on the bottom. Such approach leads to sub optimal position.

The complexity for the competitive swarms

If there exists the competitive swarms, then the computational complexity of target function for an object of swarm $\mathbf{1}$ grows due to knowing object parameters of competitive swarms, for which the linear span contains given object of swarm $\mathbf{1}$.

A construction of the linear span using grids and following localization looks as a process stated before. The computational complexity can be estimated as follows:

$$c_{kr} = O((snp_k)^{rq}),$$

where s is a number of competitive swarms. The computational complexity for non-cooperative games with s competitive swarms for searching the suboptimal control on grid decreases in r times with p_k points and for q steps of the control:

$$C^s_{kr} = O((sn^2p_k^2)^{rq}).$$

The complexity for the competitive swarms

If there exists the competitive swarms, then the computational complexity of target function for an object of swarm $\mathbf{1}$ grows due to knowing object parameters of competitive swarms, for which the linear span contains given object of swarm $\mathbf{1}$.
A construction of the linear span using grids and following localization looks as a process stated before. The computational complexity can be estimated as follows:

$$c_{kr} = O((snp_k)^{rq}),$$

where s is a number of competitive swarms. The computational complexity for non-cooperative games with s competitive swarms for searching the suboptimal control on grid decreases in r times with p_k points and for q steps of the control:

$$C^s_{kr} = O((sn^2p_k^2)^{rq}).$$

The complexity for the competitive swarms

If there exists the competitive swarms, then the computational complexity of target function for an object of swarm $\mathbf{1}$ grows due to knowing object parameters of competitive swarms, for which the linear span contains given object of swarm $\mathbf{1}$.
A construction of the linear span using grids and following localization looks as a process stated before. The computational complexity can be estimated as follows:

$$c_{kr} = O((snp_k)^{rq}),$$

where s is a number of competitive swarms. The computational complexity for non-cooperative games with s competitive swarms for searching the suboptimal control on grid decreases in r times with p_k points and for q steps of the control:

$$C^s_{kr} = O((sn^2p_k^2)^{rq}).$$

Simplest conclusions

The complexity grows polynomially with respect to numbers of the swarm objects and the controlled parameters.

The complexity grows exponentially with respect to the numbers of the computed steps of the game.

The partial searching on the grid may be effective approach to decrease the computational complexity.

Simplest conclusions

The complexity grows polynomially with respect to numbers of the swarm objects and the controlled parameters.
The complexity grows exponentially with respect to the numbers of the computed steps of the game.
The partial searching on the grid may be effective approach to decrease the computational complexity.

Simplest conclusions

The complexity grows polynomially with respect to numbers of the swarm objects and the controlled parameters.
The complexity grows exponentially with respect to the numbers of the computed steps of the game.
The partial searching on the grid may be effective approach to decrease the computational complexity.