On the issue of fuzzy timing estimations of the algorithms running at GPU and CPU architectures

Oleg Agibalov[1], *Nikolay* Ventsov[2,*]

[1]Kinoplan LLC, 107/1 Nansena St., Rostov-na-Donu, 344038, Russia
[2]Don State Technical University, Rostov-na-Donu, 344000, Russia

Abstract. We consider the task of comparing fuzzy estimates of the execution parameters of genetic algorithms implemented at GPU (graphics processing unit, GPU) and CPU (central processing unit) architectures. Fuzzy estimates are calculated based on the averaged dependencies of the genetic algorithms running time at GPU and CPU architectures from the number of individuals in the populations processed by the algorithm. The analysis of the averaged dependences of the genetic algorithms running time at GPU and CPU-architectures showed that it is possible to process 10,000 chromosomes at GPU-architecture or 5,000 chromosomes at CPU-architecture by genetic algorithm in approximately 2,500 ms. The following is correct for the cases under consideration: "Genetic algorithms (GA) are performed in approximately 2,500 ms (on average)," and α sections of fuzzy sets, with α = 0.5, correspond to the intervals [2,000.2399] for GA performed at the GPU-architecture, and [1,400.1799] for GA performed at the CPU-architecture. Thereby, it can be said that in this case, the actual execution time of the algorithm at the GPU architecture deviates in a lesser extent from the average value than at the CPU.

Introduction

The stochastic behaviour of bio-inspired algorithms makes it difficult to determine both effective algorithmic structures, and also the choice of hardware architecture with which the search procedure will be implemented. Typical representatives of bio-inspired approaches are genetic algorithms (GA), currently used to solve a wide range of optimization problems [1]. Herein after, the situation under consideration is analyzed in the context of GA using but with some modifications, it can be extended to a significant part of bio-inspired approaches. Modern hardware architectures differ not only by the number of cores and the number of operations performed by a single core per time unit but they also differ by the algorithms for organizing the computing process at the physical level [2]. CPU and GPU architectures are considered as an example of alternative hardware structures, not only

[*] Corresponding author: myvnn@list.ru

because of conceptual differences in the organization of the computing process, but also due to their wide use in solving resource-intensive problems using stochastic methods. In the context of GA execution, the contradictory nature of the work of these architectures is that the CPU starts GA execution almost immediately, in comparison with the GPU processor. The GPU architecture takes some time to initialize the structures associated with ensuring parallelism, while the time complexity of the GPU grows, as a rule, more slowly than in the CPU. Today, to solve practical problems, bio-inspired algorithms are being developed to operate with more than a billion variables [3]. A significant number of research groups are involved in reducing the complexity of multicriteria bio heuristics procedures for multidimensional problems [4]. Thus, with a large number of processed variables, the actual object of research is the issue of evaluating the effectiveness of using various hardware architectures to solve optimization problems using bio-inspired methods.

Scope of research

It is known that there is a certain limiting value of the *np* number of GA individuals, the excess of which will not lead to a significant increase in the quality of the obtained solutions [5]. If it is possible to calculate *np* (not based on experience) for an algorithm that is planning to solve the current issue, then one of the following questions will be the choice of hardware architecture with which the planned algorithm will be most efficiently implemented. In study [6], for a particular case, the average boundaries of the GPU and CPU architectures efficiency were determined, expressed by the number of individuals processed by the genetic algorithm. In study [7], it was shown that the dependences of the GA operating time on GPU and CPU architectures on the number of GA individuals, constructed only on the basis of maximum and minimum values, can significantly differ from similar dependencies constructed on the basis of averaged data. In some sense, the inverse task is the choice of hardware architecture for implementation of the algorithm from which it is required to solve the current optimization task with acceptable accuracy and in the allotted time. In other words, it is necessary to answer the question whether the algorithm running on the relatively slow hardware architecture within the framework of the issue under consideration will generate the number of individuals necessary to obtain an acceptable result. The expression "slow (within the framework of the issue) hardware architecture" indicates the fact that for several hardware architectures, there may be boundaries for their most effective application, expressed, for example, in terms of the number of individuals that need to be processed by GA to obtain an acceptable solution [6].

The research issue

If there are several hardware architectures $a_1, a_2, ..., a_n$ and the available time slots $t_1, t_2, .., t_n$ for implementation at these GA architectures, tasks similar to those listed below become relevant:

- the choice of the number of individuals $np_1, np_2, ..., np_n$ for GA, solving tasks $z_1, z_2, ..., z_n$ in such a way that they were solved for time intervals of length $t_1, t_2, .., t_n$;

- decomposition of the original task Z into non-crossing subtasks $z_1, z_2, ..., z_n$ so that they are solved by the GA, acceptable from the point of view of the quality of the solution, with architectures $a_1, a_2, ..., a_n$ for the time $t_1, t_2, .., t_n$.

Therefore, in the case under consideration, the issue of load planning is urgent: performing GA on heterogeneous computing structures, taking into account as follows:

- the number of individuals np necessary to obtain an acceptable solution of the issue;

- the choice of hardware architecture with which the GA will process and solve the task with a given accuracy for an acceptable time;
- estimates of the assumed GA processing time for np individuals with a given architecture;
- estimates of the number of individuals that GA can process with a given hardware architecture in the allotted time.

Estimation of the above factors implies a comparison of the capabilities of hardware architectures in solving the problem for the same time intervals. The purpose of this study is to perform a comparative analysis of search procedures that run for approximately equal time intervals with GPU and CPU architectures based on a computational experiment.

Simulation experiment. The study is partially based on the results of a simulation experiment, the essence of which was that for a given number of individuals (from 1,000 to 10,000, in increments of 1,000), a series of multiple (50–70) GA launches on the GPU and CPU were carried out, determining the points corresponding to optimal values of the Ackley's function [6,7]. The Figure 1 shows diagrams of the averaged dependences of time t, GA execution on CPU and GPU-architectures, on the number of individuals N processed by the algorithm.

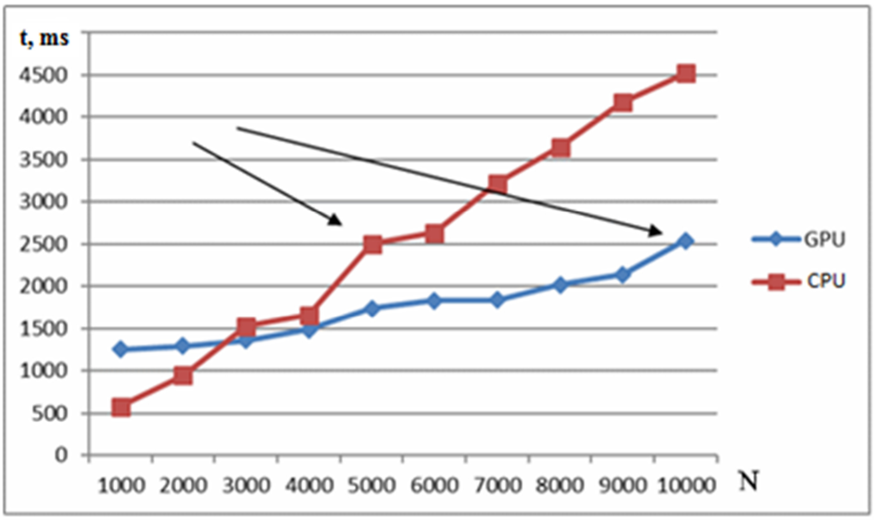

Fig. 1. Diagrams of the averaged dependences of the operating time of the GA, performed on the GPU and CPU architectures, on the number of individuals

There are four points in Fig. 1, pairwise corresponding to approximately equal averaged time intervals of the GA execution time on different hardware architectures with a different number of processed individuals. The fact of GA performance can be described by a triple (a, n, t,) — where a is the processor architecture with which the GA is executed, n is the number of individuals that the GA processed, t is the average GA operating time, the processed populations of which contain n individuals on the processor architecture a, expressed in milliseconds. Then the first pair of points can be denoted by the tuples <CPU, 3,000, 1,524> and <GPU, 4,000, 1,488>, and the second — <CPU, 5,000, 2,501> and <GPU, 10,000, 2,536>. Arrows indicate the second pair of points.

It follows from Fig. 1 that in a relatively small interval [1,000,10,000], characterizing the number of processed individuals with a sufficiently large sampling step (1,000 individuals), in the presence of two hardware architectures, there are alternative, from the point of view of the average operating time, GA implementation options with various hardware architectures, but significantly different in the number of processed individuals. For this reason, the issue to analyze alternatives for the average operating time of GA implementation options on various hardware architectures is urgent.

The second pair of points <CPU, 5,000, 2,501> and <GPU, 10,000, 2,536> describes the options for implementing the GA with a greater difference between the processed individuals and a smaller spread of the average execution time than the first pair. Let's consider it in more detail. Calculation process, the parameters and the results of which are characterized by points <CPU, 5,000, 2,501> and <GPU, 10,000, 2,536> can be described by the phrase "GA execution in approximately 2,500 ms."

If we plan the calculation process in accordance with the averaged dependencies shown in Fig. 1, then if we need to solve the task in about 2,500 milliseconds, we can process 10,000 chromosomes with the GPU-architecture or 5,000 chromosomes with the CPU-architecture with the genetic algorithm. Table 1 shows the minimum, maximum and average values of the operating time of genetic algorithms that process 10,000 chromosomes with the GPU architecture and 5,000 chromosomes with the CPU architecture. Architectures parameters of processes with which the GA are performed, as well as the number of individuals processed by the GA, are considered as parameters for GA performance.

The data presented in Table 1 are taken from the results of a computational experiment, on the basis of which the graphs shown in Fig. 1 were built.

Table 1. Characteristics of genetic algorithms running on average in approximately 2,500 ms

No.	t, ms	GA execution parameters	
		CPU, 5,000 chromosomes	GPU, 10,000 chromosomes
1	t_{min}, ms	1,397	1,764
2	t_{max}, ms	32,625	18,311
3	t_{mean}, ms	2,501	2,536

The data presented in table 1 are the result of the fact that the average execution time of the genetic algorithm, in the cases considered, deviates by less than 2%. Based on the data obtained (Table 1), we can state the following:

- The genetic algorithm executed with the CPU will be able to process 5,000 chromosomes on average in approximately 2,500 ms;
- The genetic algorithm executed with the GPU will be able to process 10,000 chromosomes on average in approximately 2,500 ms.

Let's determine the relative deviation from the average value of the minimum and maximum execution time of the genetic algorithm for the cases presented in Table 1.

The relative deviation Δt_{min} of the minimum execution time t_{min} from the average t_{mean} is calculated by the formula:

$$\Delta t_{min} = |t_{mean} - t_{min}| / t_{mean}. \qquad (1)$$

For the algorithm running with the CPU:

$$\Delta t_{min}CPU = |\, 2{,}501 - 1{,}397 \,|/\, 2{,}501 \approx 0.441. \quad (2)$$

For the algorithm running with the GPU:

$$\Delta t_{min}GPU =|\, 2{,}536 - 1{,}764 \,|/\, 2{,}536 \approx 0.304. \quad (3)$$

The relative deviation Δt_{min} of the maximum execution time t_{max} from the average t_{mean} is calculate by the formula:

$$\Delta t_{max} =|\, t_{mean} - t_{max} \,|/\, t_{mean}. \quad (4)$$

For the algorithm running with the CPU:

$$\Delta t_{max}CPU ==|\, 2{,}501 - 32{,}625 \,|/\, 2{,}501 \approx 12.04. \quad (5)$$

For the algorithm running with the GPU:

$$\Delta t_{max}GPU =|\, 2{,}536 - 18{,}311 \,|/\, 2{,}536 \approx 6.22. \quad (6)$$

Comparing Δt_{min} CPU with Δt_{min} GPU and Δt_{max} CPU with Δt_{max} GPU, it can be argued that in this case, the actual execution time of the algorithm with the GPU deviates to a lesser extent from the average than with the CPU.

In the study [6], for a special case, a membership function of a fuzzy number was constructed which denotes the running time of the GPU algorithm on 3,000 chromosomes. Let's build similarly fuzzy number functions that describe GA processing of 10,000 chromosomes with the GPU-architecture and 5,000 chromosomes with the CPU-architecture.

When conducting a simulation experiment, the minimum runtime of the algorithm on the GPU-architecture that processes 10,000 chromosomes was 1,764 ms, the maximum — 18,311 ms: but, at the same time, in other cases, the running time of the algorithm did not exceed 3,000 ms. Therefore, let's consider the distribution of the duration of the algorithm in ms on the interval [1,600, 3,000] in more detail.

The Fig, 2 shows a fragment of the distribution of GA operating time with a GPU-architecture processing 10,000 chromosomes over time intervals.

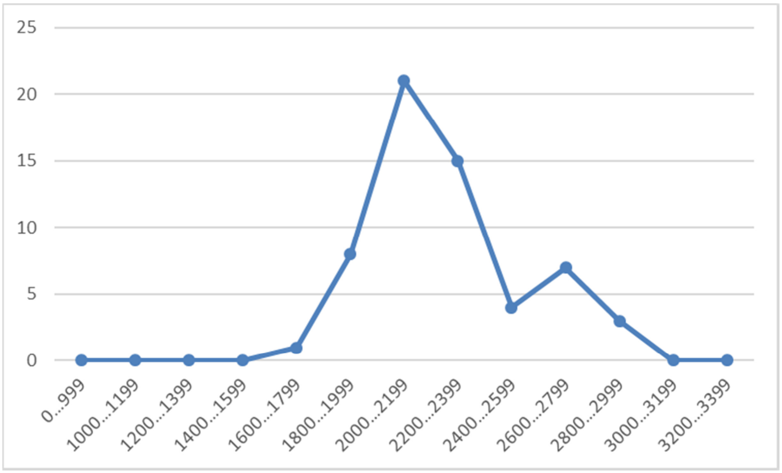

Fig. 2. A fragment of the GA running time distribution with GPU-architecture by time intervals

In Fig. 2 and Fig. 3: along the ordinate axis, the number of GA launches is recorded, the execution time of GA falls into the corresponding segment, indicated on the abscissa axis.

It follows from Fig. 2 that more than half of the GA launches were completed during the period from 2,000 to 2,399 ms.

When conducting a simulation experiment, the minimum operating time of GA with the CPU-architecture processing 5,000 chromosomes was 1,397 ms, and the maximum was 18,311 ms: but, at the same time, in other cases, the running time of the algorithm did not exceed 3,100 ms. Therefore, let's consider the distribution of the duration of the algorithm in ms on the interval [1,200, 3,200] in more detail.

The Fig. 3 shows a fragment of the distribution of GA operating time with a CPU architecture processing 5,000 chromosomes over time intervals.

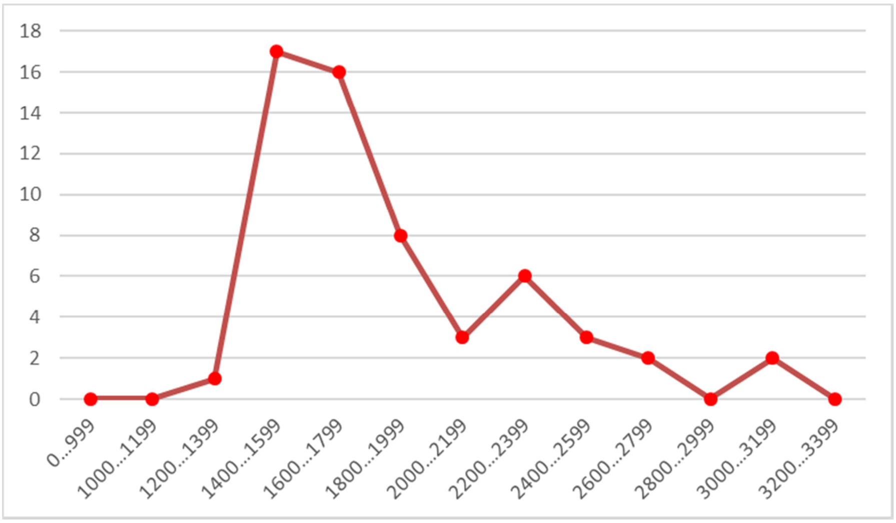

Fig. 3. Fragment of the distribution of the operating time of GA performed with CPU-architecture over time intervals

It follows from Fig. 3 that more than half of the GA launches were completed during the period from 1,400 to 1,799 ms.

In accordance with [6], let's consider the runtime of GA with GPU-architecture that processes 10,000 chromosomes, and 5,000 chromosomes with CPU-architecture as fuzzy numbers.

The Fig. 4 presents a diagram that can be interpreted as a fuzzy set:

$$\tilde{G} = \{t, \mu_G(t)\}, \tag{7}$$

where $\tilde{G}$ is a fuzzy set constructed on the basis of averaged data corresponding to the expression "GA operating time with GPU-architecture that processes 10,000 chromosomes", t is the GA operating time, $\mu_G(t)$ the membership function of the operating time t to a fuzzy set $\tilde{G}$.

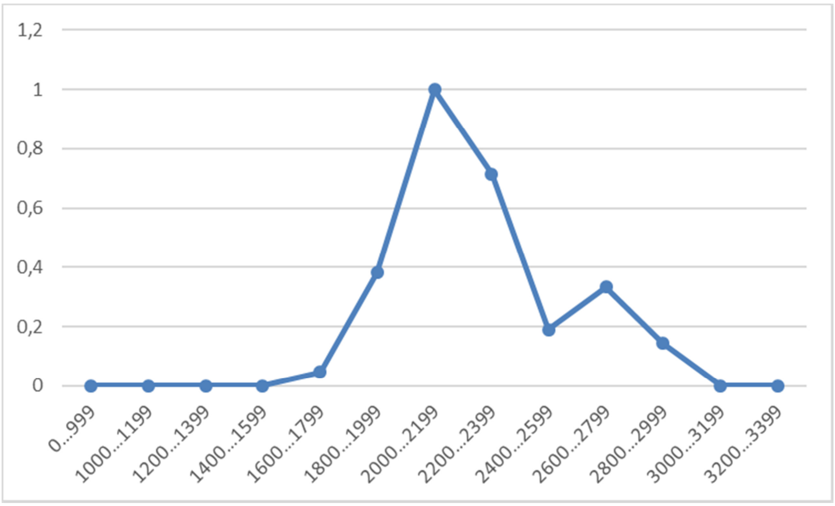

Fig. 4. Fragment of the membership function of a fuzzy number that describes “the time of GA operation with the GPU-architecture processing 10,000 chromosomes”

The Fig. 5 presents a diagram that can be interpreted as a fuzzy set:

$$\tilde{C} = \{t, \mu_C(t)\}, \tag{8}$$

where $\tilde{C}$ is a fuzzy set constructed on the basis of averaged data corresponding to the expression “GA operating time with CPU-architecture that processes 10,000 chromosomes”, t is the GA operating time, $\mu_C(t)$ the membership function of the operating time t to a fuzzy set $\tilde{C}$.

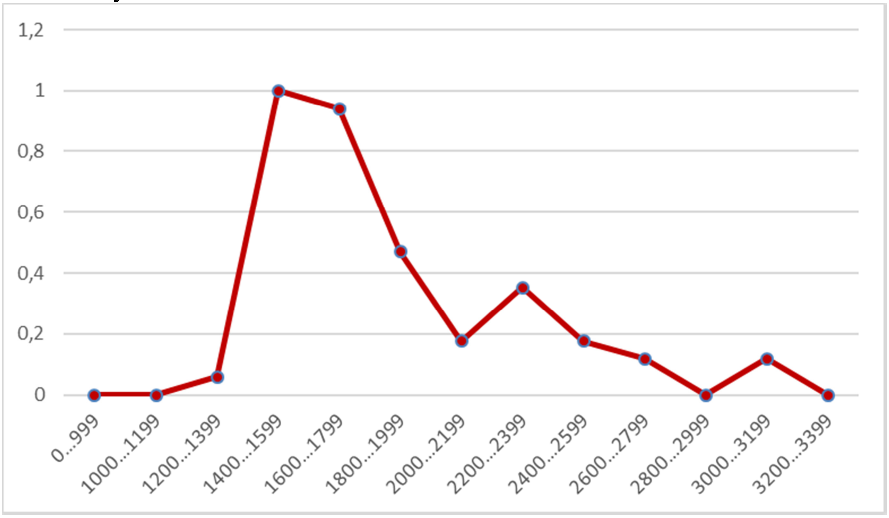

Fig. 5. Fragment of the membership function of a fuzzy number that describes “the time of GA operation with the CPU-architecture processing 10,000 chromosomes”

The transition point of a fuzzy set is its element *x* for which μ (x) = 0.5. For $\tilde{G}$ and $\tilde{C}$ we can define fuzzy sets of level α = 0.5 [8]:

$$\tilde{G}_{\alpha=0,5} = \{(1,[2000,2199]),(0{,}71,[2200,2399])\}; \tag{9}$$

$$\tilde{C}_{\alpha=0,5} = \{(1,[1400,1599]),(0{,}94,[1600,1799])\}. \tag{10}$$

Then α is a slice of fuzzy sets $\tilde{G}$ and $\tilde{C}$ for α= 0.5 there will be elements $G_{\alpha=0,5}$ and $C_{\alpha=0,5}$:

$$G_{\alpha=0,5} = \{([2000,2199]),([2200,2399])\}; \tag{11}$$

$$C_{\alpha=0,5} = \{([1400,1599]),([1600,1799])\}. \tag{12}$$

A crisp set, C* closest to the existing fuzzy set $\tilde{C}$, determined in accordance with a known formula:

$$\mu_{C^*}(t) = \begin{cases} 1, \ if\ \mu_C(t) > 0{,}5; \\ 0, \ if\ \mu_C(t) < 0{,}5; \\ 0\ or\ 1, \ if\ \mu_C(t) = 0{,}5. \end{cases} \tag{13}$$

According to the data presented in Fig. 5, it can be argued that the set G * contains two spaces [1,400, 1,599] and [1,600, 1,799].

According to the data presented in Fig. 5, it can be argued that the set C* contains two spaces [2,000, 2,199] and [2,200, 2,399].

Conclusion

1. When planning the calculation process in accordance with the averaged dependencies presented in Fig. 1, if necessary solving the task in approximately 2,500 milliseconds, you can process 10,000 chromosomes with the GPU-architecture or 5,000 chromosomes with the CPU-architecture with the genetic algorithm. The following is correct for the cases under consideration: “GA are performed approximately in 2,500 ms on average.”
2. Comparing Δt_{min} CPU with Δt_{min} GPU and Δt_{max} CPU with Δt_{max} GPU, it can be argued that in this case, the actual execution time of the algorithm with the GPU deviates to a lesser extent from the average than with the CPU.
3. For the statement “genetic algorithms are executed on average in approximately 2,500 ms” (in the context of the case under consideration), it is typical the following:

- fuzzy sets of level α = 0.5 for $\tilde{G}$ and $\tilde{C}$ are equal to:

$$\tilde{G}_{\alpha=0,5} = \{(1,[2000,2199]),(0{,}71,[2200,2399])\};$$

$$\tilde{C}_{\alpha=0,5} = \{(1,[1400,1599]),(0{,}94,[1600,1799])\}.$$

-α slices of fuzzy sets $\tilde{G}$ and $\tilde{C}$ at α = 0.5 correspond to the intervals [2,000, 2,399] for a GA running with GPU-architecture, and [1,400, 1,799] for GA running with CPU-architecture.

4. The results presented in the work correspond to the modern level both from the point of view of mathematical [8] and algorithmic support [9,10].

The reported study was funded by RFBR, projects Nos: 19-01-00357, 18-01-00314

Reference

1. H. Khankhour, J.Abouchabaka, O. Abdoun, Lecture Notes in Networks and Systems, **92**, 145-154 (2019)
2. A.V. Boreskov, Moscow State University publishing office, 336 (2012)
3. K. Deb, C. Myburgh, Breaking the Billion- Proceedings of Genetic and Evolutionary Computation Conference, 653–660 (2016)
4. S. I. Rodzin, Cheboksary: Publishing House "Sreda", 224 (2019)
5. A. N. Saprykin, K. D. Akinina, E. N. Saprykina, Actualscience, **11**, 168–169 (2016)
6. O. I. Agibalov, N. N. Ventsov, Cybernetics and programming, **6**, 1–8 (2017) DOI: 10.25136/2306-4196.2017.6.24509
7. O. I. Agibalov, N. N. Ventsov, Software systems and computational methods, **3**, 12–19 (2019) DOI: 10.7256/2454-0714.2019.3.30502
8. Jiri Mockor. α-Cuts and models of fuzzy logic (2012) DOI: 10.1080/03081079.2012.710438
9. J. Luo, S. Fujimura, D. El Baz, B. Plazolles, Journal of Parallel and Distributed Computing, **133**, 244-257 (2019)
10. Z. Li, G. Xiong, X. Zhang, Z. Shen, C. Luo, X. Shang, X. Dong, G.-B. Bian, X. Wang, F.-Y. Wang, IEEE, 471-478 (2019)