Chapter 1
Brief History and Overview of Intelligent Optimization Algorithms

Up to now, intelligent optimization algorithm has been developed for nearly 40 years. It is one of the main research directions in the field of algorithm and artificial intelligence. No matter for complex continuous problems or discrete NP-hard combinatorial optimizations, people nowadays is more likely to find a feasible solution by using such randomized iterative algorithm within a short period of time instead of traditional deterministic algorithms. In this chapter, the basic principle of algorithms, research classifications, and the development trends of intelligent optimization algorithm are elaborated.

1.1 Introduction

Intelligent optimization algorithm is developed and integrated from a number of relatively independent sub-fields, including the technology of artificial neural networks, genetic algorithms, immune algorithms, simulated annealing, and tabu search and swarm intelligence techniques. As we all know, the current main intelligent optimization algorithms are based on the mode of population based iteration. They operate a population, which represents a group of individuals (or solutions), in each generation to maintain good information in the solution space and find better positions step by step. It is a common method that is independent from specific problems so that it can handle complex optimization problems that are difficult for traditional optimization methods. Their common characteristics are: (1) all operations act on current individuals in each generation; (2) the searching is based on iterative evolution; (3) the optimization can be easily parallelized by multi-population scheme; (4) most of them can give a satisfactory non-inferior solutions close to the optimal solutions, instead of certainly finding the optimal solution; (4) the algorithm is rather random and cannot guarantee the

© Springer International Publishing Switzerland 2015

F. Tao et al., *Configurable Intelligent Optimization Algorithm*,
Springer Series in Advanced Manufacturing,
DOI: 10.1007/978-3-319-08840-2_1

efficiency of finding non-inferior solutions. Therefore we simplify the basic process of intelligent optimization algorithms in Fig. 1.1.

Specifically, the first step in problem solving is the encoding design according to the problem environment, character and constraints. Coding is a mapping of problem variables, and it also directly determines the evolution speed of the algorithm. When an appropriate coding scheme is selected, the algorithm will initialize and generate a certain number of individuals to form a population according to the definition of problem variables. Each variable of an individual is randomly generated in the definition domain, and the fitness value of individual in each generation is calculated according to problem's objective function, i.e. fitness function. Then, according to the coding scheme, the variables are mapped to form a chromosome. After initialization and encoding, the algorithm iteration will be started. In iteration, the combination of operators plays a major role, such as selection, crossover, and mutation operator in genetic algorithm, and path finding and pheromone update in ant colony algorithm. Different operators have different effects in the algorithm, so different composition of operators can usually produce different results in solving problem. Through a few operations, some or the entire individuals in the population are changed. By decoding of the new individuals, a new group of solutions can be obtained. Through the evaluation of population according to fitness function (objective function), the best and the worst individual in the population can be picked out. Then we could update the whole population in generation by using random alternative strategy, elitist replacement strategy or merge-based individual optimal selection. After that, the next iteration will be triggered. When the number of iterations reaches a certain ceiling or the satisfactory feasible or optimal solution has already been reached, we can jump out the iteration loop, and output the global best solutions.

In this evolution mode, problem variables and objectives are reflected by coding and fitness functions, while constraints are embodied in fitness function as penalty function or in coding as a bound. The operators are independent to the problems and can be easily implemented. The unified iterative process can make the design, improvement and hybrid research of intelligent algorithms simpler, more intuitive, and more flexible. With iteration-based sub-optimal searching, intelligent optimization algorithm can effectively avoid the combinatorial explosion when solving NP-hard problem. However, not all operators can be arbitrarily combined to form an effective algorithm. Since the evolutionary process is quite random, and operators have different characteristics and limitations in exploration and exploitation respectively, many intelligent optimization algorithms still have some defects such as premature convergence and large result deviation and so on. Therefore, researchers in various fields still keep looking for bran-new coding schemes, operators, good improvements and hybrid forms of intelligent optimization algorithms. All these works are trying to search better feasible solution with limit iterations and limit size of population.

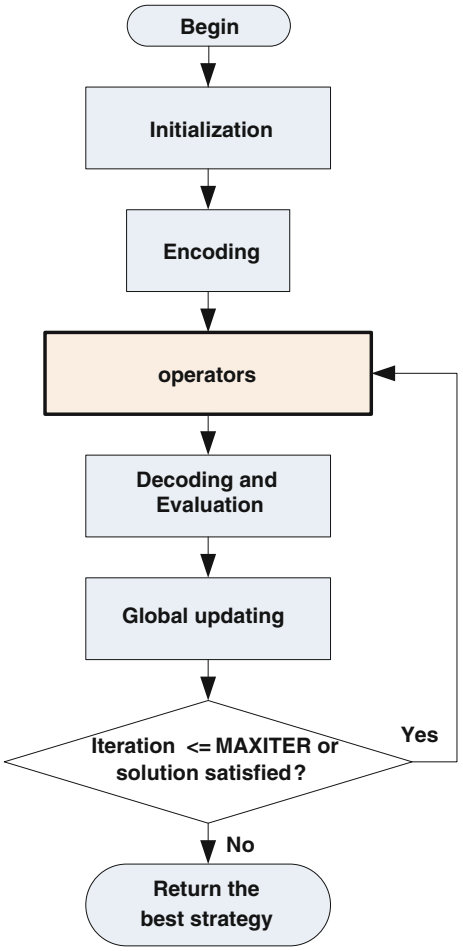

Fig. 1.1 the basic process of intelligent optimization algorithms

1.2 Brief History of Intelligent Optimization Algorithms

With the rapid development of new technology, manufacturing with multi-disciplinary distributed collaboration becomes more and more prevalent in many enterprises. The scale of resources in distribution grows rapidly, and the whole life cycle of manufacturing, including product design, simulation, production, logistics and maintenance, are becoming increasingly complicated. In order to further shorten industry cycle, enhance operation efficiency and improve the utilization of resources and information, many complex problems in macro- and micro-processes should be solved and optimized. From a mathematical standpoint, these problems can also be divided into continuous numerical optimization problems [1, 2] and discrete combinatorial optimization problems [3, 4] according to their variables or the characteristics of the main factors (i.e. the continuity of the solution space), as

well as general optimization problems. For example, the optimization of complex functions, multi linear and non-linear equations in controlling and simulation field, and complex nonlinear programming can be categorized as continuous numerical optimization problems. Typical workflow/task scheduling, service composition optimal selection, collaborative partner selection and resource allocation problems in manufacturing process and system management can be classified as discrete combinatorial optimization problems. Since the quality of these computing and decision-making methods directly determines the efficiency of the whole manufacturing system, experts in all kinds of fields have carried out research to model complex problems under specific conditions and to design of various deterministic algorithm or approximate algorithm for them [5]. In the case of small-scale solution space decisions, various deterministic algorithms can effectively give the optimal solution, and approximation algorithms can also give approximate solution in a shorter time period. However, in reality, most optimization problems are proved to be NP-hard [6–8]. That means no algorithm can find optimal solution in polynomial time. Actually, with the gradually increasing scale of problem, the searching time of traditional deterministic algorithm will grow exponentially, which is quite prone to trigger combinatorial explosion [9]. Likewise, most approximation algorithms have the problem of low approximation ratio, bad versatility and difficulty to meet the requirements of the feasible solutions in the case of solving such large scale problems [10]. Faced with large solution space, both deterministic and approximate algorithms appear to be inadequate.

At this time, the emergence of the intelligent optimization algorithm [11] provides new ideas for this type of large-scale NP-hard optimization problems. Represented by genetic algorithm (GA), simulated annealing algorithm (SAA), ant colony optimization (ACO) and particle swarm optimization (PSO) and so on, this type of algorithm is also known as meta-heuristic. It is formed by simulating biological natural computing skills and the process of evolution. It is also a type of global optimization probabilistic searching algorithm based on the population iterative evolution model. On the other hand, intelligent optimization algorithm can uniformly abstracts problem variables by coding and uses fitness function to represent the objectives. It modifies the population in the solution space by each step iterative operation and evolution with the combination of randomness and guidance to do search. As we have mentioned, it is independent from certain problems and can handle complex optimization problems that are difficult to solve. When solving large-scale NP-hard problem, intelligent optimization algorithm can quickly obtain a set of feasible solutions under the given constrains within a limited time. Instead of finding optimal solution, it uses heuristics to obtain sub-optimal feasible solution and shorten the whole searching process and then improving the efficiency of decision-making. With limited iterations and independent operations, algorithm timeliness would not decrease as the solution space increases and it has good robustness. Therefore, intelligent optimization algorithms have received widespread concern in various fields [12, 13].

Start from the development of early classical optimization algorithms, through careful observation and summary, people have proposed a variety of new

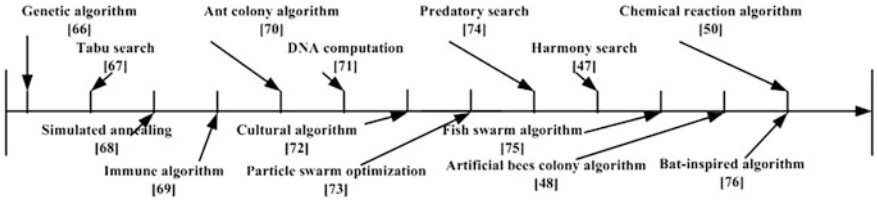

Fig. 1.2 The development of intelligent optimization algorithms [47, 48, 50, 66–76]

intelligent optimization algorithms which are designed to simulate different kinds of nature biological behavior, as is shown in Fig. 1.2. Due to the space limitation, we only list some typical ones, such as Memetic Algorithm (MA) [14], Difference Evolution Algorithm (DEA) and so on [15]. It can be seen that the creation of new intelligent optimization algorithm has not yet stopped and still shows vigorous development trend.

From the results of the application of the very basic intelligent optimization algorithms to many complex numerical problems and standard combinatorial optimization problems (such as the traveling salesman problem and the knapsack problem), a group of researchers have made many improvements from the aspects of algorithm initialization, encoding, operators and evolutionary strategy and produced a series of typical improvement strategy for intelligent optimization algorithm, such as quantum coding improvement [16], adaptive improvement [17], niche strategy improvement [18], prior knowledge-based improvement [19], Pareto search improvement and so on [20]. Considering the problem features, people usually make adjustments on the original basic algorithm in the aspects of exploration, exploitation and the adaptive balance in process respectively and achieved good performance. In addition, after years of study, experts in many fields have found that the hybrid application of many different operators or search mechanisms in different algorithms can greatly enhance the efficiency of problem solving. So hitherto much emphasis has been place on the design and application of hybrid intelligent optimization algorithm. Compared with the design of new operators and improvement schemes, algorithm hybridisation is much simpler, since limited operators stripped from the different algorithms can form a variety of combined intelligent optimization algorithm. After the characteristic analysis of different operators and several tests, a good hybrid intelligent optimization algorithm can be chosen to solve the specific problem.

With a variety of improvements and hybridizations in 40 years' development, the group of intelligent optimization algorithm has been greatly expanded. Therefore, many of the algorithms are applied to engineering practice activities, which have been proved to have good performances in different kinds of benchmark problems. Due to the uniform structure and versatility of intelligent optimization algorithm, users in practice also do some modification and design improvements or hybrid schemes to enhance its performance on specific problem according to particular application environment.

Through the retrieval of intelligent optimization algorithms with several typical keywords during 2001–2012 in "Web of Knowledge" database, we get the number of general literature over the past decade on the intelligent optimization algorithms. According to the following statistical results, a curve with year as abscissa and the annual number of relevant research literatures as ordinate is drawn and shown in Fig. 1.3.

As can be seen from Fig. 1.3, the number of literatures on intelligent optimization algorithm generally has an upward trend in this decade. While in 2009 it reached a peak and had a stable development in the following years, especially in the past 3 years. Actually this research topic is quite popular and abounds great research value, meaning and application prospects because thousands of improvements or hybridizations of intelligent optimization algorithm have been proposed and applied in different fields. However, among the large amount of design and implementation works, whether there exist a large number of duplicate or similar works is still remain unsolved. What's more, with a large number of improved, hybrid and newly proposed algorithm, how to select an appropriate intelligent optimization algorithm to solve the specific problem? These are problems worth studying in depth in the field of intelligent manufacturing.

1.3 Classification of Intelligent Algorithms

As has been explained in the previous section, the number of intelligent optimization algorithms with different evolution mechanisms today has reached to an unparalleled level. Researchers have carried out numerous works on algorithm design for specific problems with different backgrounds and objectives. With different categorizing perspectives, intelligent optimization algorithms can be divided into varying groups. According to the research focuses, the mainstream works on intelligent optimization algorithm can be broadly divided into four categories: (1) algorithm innovation; (2) algorithm improvement; (3) algorithm hybridation; (4) algorithm application. On the basis of literature review in the previous section, we selected 200 of them to conduct research, and draw the approximate percentage of the four categories of institute as Fig. 1.4 shows.

Moreover, according to different algorithm search mechanism, we divide the basic intelligent optimization algorithms into three categories: Evolutionary learning algorithm, neighborhood search algorithm and swarm intelligence algorithm, as is shown in Fig. 1.5.

Evolutionary learning algorithm: Including genetic algorithms, evolutionary programming, artificial immune algorithms, DNA computing and so on, they are formed in accordance with the mechanism of natural learning evolution. Individuals in population are updated by learning from each other in generation according to different heuristics. The most widely ones in this category are genetic algorithm and artificial immune algorithm.

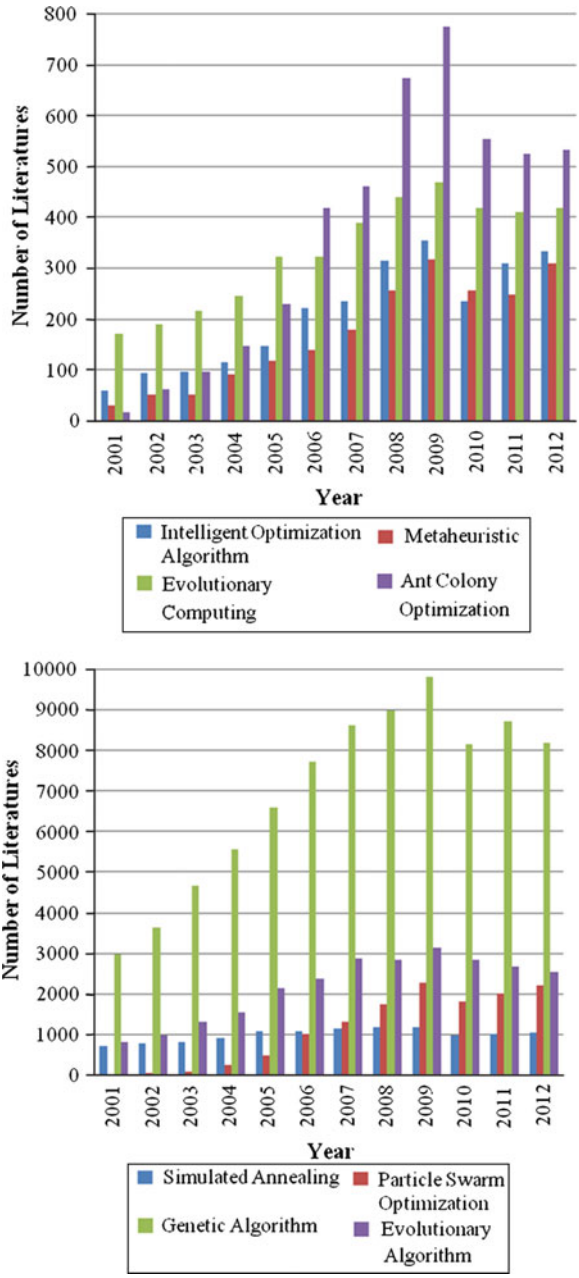

Fig. 1.3 intelligent optimization algorithm research trends in recent years

Neighborhood search algorithm: The most typical ones are simulated annealing algorithm, iterative local search and variable neighbor search. They also belong to local search strategies. Neighborhood search in them are generally implemented by a random or regular changing step and local search step. In searching process,

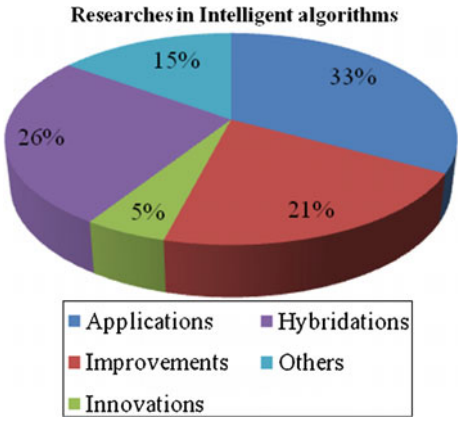

Fig. 1.4 The percentages of the studies on algorithm innovation, improvement, hybridation and application respectively

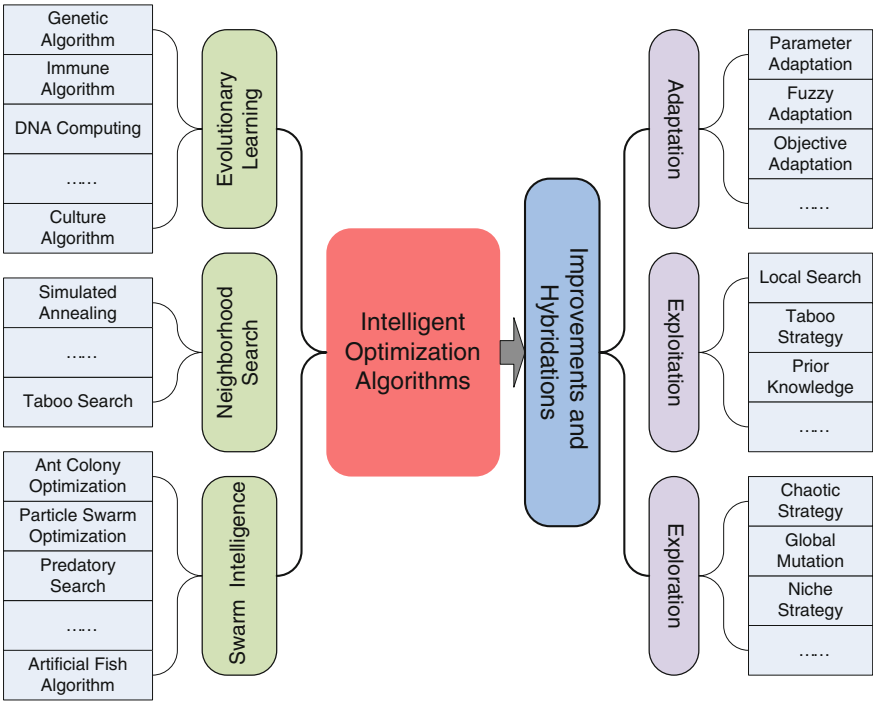

Fig. 5 The classification of intelligent optimization algorithm

additional control parameters or individual acceptance rules in these algorithms are gradually changed to achieve convergence. Thus the main characteristic of this category is the independent local searching of individual. Now most of them are commonly used as an improvement strategy for other algorithms.

Swarm intelligence algorithm: Including ant colony algorithm, particle swarm optimization and artificial fish algorithm and so on, they are designed by simulating self-organizing behavior of social animals (ant colony, bees, flocks of birds, etc.). Normally, they use the broadcasting or transmission of social information extracted by individuals to achieve organizational optimization. At present, the most popular studied algorithms are ant colony algorithm and particle swarm algorithm.

Additionally, with the proposed algorithm shown in Fig. 1.5, a number of improved strategies with different characteristics have been proposed. Researchers conduct their studies from the perspectives of algorithm convergence, exploration, exploitation and stability, aiming to guide the algorithm searching for better near-optimal solutions. From the strategy efficacy [21], we divide these strategies also into three categories: adaptive improvement, exploitation improvement, and exploration improvement.

- Adaptive improvement: Adaptive improvement aims at balancing the search breadth at the prophase and the excavation depth at the late stage. Parameter adaptive improvement, fuzzy adaptive improvement and objective-based adaptive improvement are typical ones.
- Exploitation improvement: Exploitation improvement is primarily to improve the excavation performance of algorithms, mostly manifested by enhancing searching orientation and small-scale traversing.
- Exploration improvement: Global search improvement is designed mainly to enhance the diversity of the population in the algorithmic search, preventing the algorithm from falling into local optimum. Niche strategy, chaos strategy and various mutations are typical ones.

Apart from the above two study branches, there is one more important research focus, i.e. algorithm application. No matter for communication, electronic engineering and mechanical analysis in manufacturing system, or for control and management in manufacturing process, different application object have different requirements on algorithm. In terms of the purpose, the algorithm application object falls into three areas: pedagogical, application and research, as shown in Fig. 1.6.

- Pedagogical: For this purpose, what users want is to quickly understand the basic classification and characteristics of different algorithms. After that, they might move on to their respective principles, mainframes, and performances.
- Application: For application, users care more about selecting appropriate algorithms for addressing the given problem. Theoretical study might be unimportant for them.
- Research: For senior researchers, merely providing a few fixed algorithms is not enough. Beyond that, they need to test and compare more different algorithms under some standard circumstances.

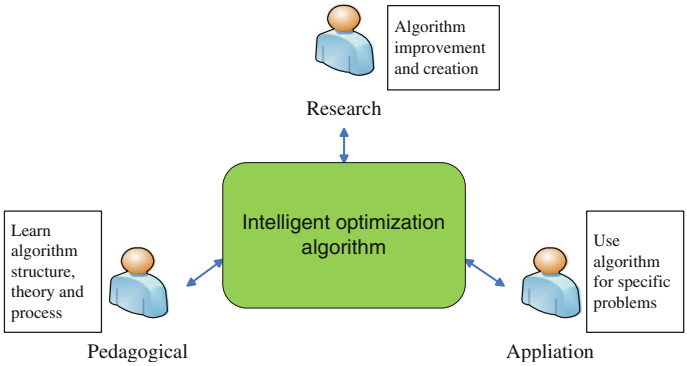

Fig. 1.6 Different design purpose in intelligent optimization algorithm

For different users, the focus is different, and there will be different requirements for algorithm design and application. So how to fully develop the potential and efficiency of the existing intelligent optimization algorithms in different kinds of applications is a problem worth studying.

1.4 Brief Review of Typical Intelligent Optimization Algorithms

Based on different categorizations of intelligent optimization algorithms, this section hence gives brief introduction for several typical algorithms from the perspective of the searching mechanism classification.

1.4.1 Review of Evolutionary Learning Algorithms

The most typical evolutionary learning algorithms include genetic algorithm and immune algorithm and so on. Genetic algorithm simulates biological evolution rules, and uses "survival of the fittest" rules to retain good search information, applies crossover operator to implement information transmission during individuals and employs mutation operator to preserve the diversity of the population. The immune algorithm simulates human body's immune mechanism, using prior knowledge as antigen to guide the whole population search with heuristic directions. Combining with immune selection, individuals with excellent antibody can be obtained and evolution can then be kept. Both of them have shown good performances in representative numerical and combinatorial optimization problems.

1.4.1.1 Genetic Algorithm

Genetic algorithm (GA) is one of the earliest intelligent optimization algorithms, which is relatively mature. The pseudo-code of the most commonly used genetic algorithm can be shown in the following

```
Begin
    t := 0;
    P(t) := InitPopulation();
    Evaluate(P(t));
    While (stop criteria unsatisfied)
        P'(t) = Select(P(t));
        P'(t) = Crossover(P'(t));
        P'(t) = Mutate(P'(t));
        Evaluate(P'(t));
        P(t + 1) = UpdateNewPop(P(t), P'(t));
        t = t + 1;
    End
End
```

where t represents the iteration number, $\mathbf{P(t)}$ represents the population in generation t, and $\mathbf{P'(t)}$ represents the population after one time algorithm operations. Normally, genetic algorithm consists of three basic operators, i.e. selection, crossover and mutation. Since 1975 Holland proposed its complete structure and theory, the improvement of operators for exploration and exploitation and the avoidance of premature convergence are also hot topics in this field.

In terms of encoding, the original method used by Holland is binary coding. But for many applications of genetic algorithms, especially in industrial engineering, this simple coding way is difficult to describe some key properties of problem directly. Thus in recent years, many typical encoding methods are proposed intended to some special or benchmark problems. For instance:

Real number coding [22]: The method is most commonly used coding way in addition to binary coding. It can intuitively express a large amount of continuous and discrete variables, especially suitable to represent variables with large range in genetic algorithm and avoid the process of encoding/decoding. As a result, it can

improve the efficiency and accuracy of genetic algorithm to some extent and reduce the algorithm complexity.

Matrix coding [23]: Matrix coding, in which every variable is represented by multiple gene-bits, is mainly applied to combinatorial optimization problems. It can be used to represent time scheduling sequences, symbol strings, positions in multi-dimensional space and so on clearly and intuitively. In some cases, it can also enhance the searching performance with proper information exchange and diversity keeping. The disadvantage is mainly that it occupies larger memory space and needs encoding/decoding steps which will increase the algorithm complexity.

Quantum coding [24]: This coding method is not only preferred by genetic algorithm, but also widely used in other algorithms. Due to its double-bit coding style, individuals have the characteristics of dimorphism which can largely enhance the diversity of the population in the process of encoding and decoding. It makes individuals randomly cover a wider range of solution space, and obtain improvement of algorithm search performance at the expense of the increase of time complexity.

More than we mentioned, more coding methods are designed and presented as the development of intelligent optimization algorithms, such as dynamic coding, symbol coding, and structure coding and so on.

In the prospect of operators, selection operator plays an evolution guiding role with the rule "survival of the fittest". A variety of improved operators are proposed during research and development, for instance:

- roulette selection strategy
- tournament selection strategy
- sort selection strategy
- random selection strategy.

Crossover operator, as the core operator, is used to recombine gene information, generate offspring, and spread good information. So far, except the single or multi-points crossover, the commonly used methods also include:

- uniform crossover
- shuffle crossover
- crossover with reduced surrogate.

Besides, mutation operator allows a few individuals to jump out of the current states and explore the new area. It plays a role to avoid the premature convergence and enhance the exploration ability of algorithm. In addition to single/multi-point mutation, commonly used methods are:

- uniform mutation
- chaotic mutation
- Gaussian mutation.

By and large, GA has a deep theoretical foundation in both theory and application aspects [25–27]. Other than these improvements, there are also many artificial intelligence strategy-based hybrid genetic algorithms with different forms. Currently, most classic coding methods, typical operators, improvements

and evolution strategies are researched and obtained based on genetic algorithm, and gradually applied to other typical intelligent optimization algorithms. Of course, for different problems, the searching performances of these algorithms are all different and more or less have some drawbacks on searching ability or process time. Hence the researches on genetic algorithm and its improvements for different specific problems are never stopped.

1.4.1.2 Immune Algorithm

The diversity of identification of the immune system has brought a lot of inspiration in the researches of intelligent optimization algorithm. According to immune system and its learning mechanism of diversity recognition, various immune algorithms are designed. The pseudo-code of the typical immune algorithm [28] is as follows:

```
Begin
    t := 0;
    P(t) := InitPopulation();
    Evaluate(P(t));
    While (stop criteria unsatisfied)
        P'(t) = Crossover/Mutate(P(t));
        P'(t) = Vaccination(P'(t) );
        P'(t) = ImmuneSelection(P'(t) );
        Evaluate(P'(t));
        P(t + 1) = UpdateNewPop(P(t), P'(t));
        t = t + 1;
    End
End
```

where t represents the iteration number, **P(t)** represents the population of generation t, **P'(t)** represents the population after one time operations. The immune algorithm contains three basic operators: vaccination, crossover/mutation and immune selection. Priori knowledge of problem is generally extracted as antigen to guide the individual changing. The priori knowledge-based individual changing is vaccination. After that, typical or special designed crossover or mutation operators are employed to make further variation. Finally the immune selection operator tries to

select good individuals with high fitness to realize population updating. The whole process simulates the adaptive production of antibodies in human body's immune system to make the algorithm searching with adaptive convergence and breadth-searching ability. Its main drawback is that the selection of vaccines and vaccination requires deep analysis of the properties and priori information of specific problems, which largely increases the design process and decreases the versatility of algorithm.

In addition, inspired by artificial immune system, there are several other immune algorithms:

- Immune programming (IP) [29].
- Clonal selection algorithm [30].
- Negative selection algorithm [31].

In which immune programming is similar to the immune algorithm, using the diversity and maintaining mechanism of immune system to maintain the diversity of population. In addition, clonal selection algorithm uses the characteristics of adaptive immune response to construct evolutionary selection based on the cloning reaction affinity maturation process, which gets a lot of attention researches. Other than clonal selection algorithm, negative selection algorithm conducts changing and monitoring of the individual alternating probability based on the principle of self and non-self recognition in immune system to realize evolution and optimal selection. The three important principles in negative selection algorithm are: [1] each monitoring algorithm is unique, [2] the monitoring process is probabilistic and [3] the system with robustness should be able to randomly monitor external events rather than search for a known mode. Of course, since the artificial immune systems and its applications still have a long way to go, the design of different sorts of immune algorithms concerning parameter selection and theoretical discussion still has much to be improved.

1.4.2 Review of Neighborhood Search Algorithms

The most typical examples of neighborhood search algorithm are Simulated Annealing (SA) algorithm and Iterative Local Search (ILS) and Variable Neighborhood Search (VNS). We only introduce the first two methods in this chapter. Simulated annealing imitates the annealing process in thermodynamic, achieving optimal selection and convergence process based on random neighborhood search technique. Iterative local search is a combination of local search or hill climbing strategy and general random operation. The evolutionary process of these two methods can be either based on single-individual or on population, and the search strategy can be flexibly altered. Therefore, now they usually play a significant role in the hybridization with other algorithms to improve the overall performance of problem solving.

1.4.2.1 Simulated Annealing Algorithm

In the year of 1982, Kirkpatrick et al. brought the annealing theory into the field of combinatorial optimization, proposing simulated annealing algorithm to solve large-scale combinatorial optimization problems. The essence of this algorithm lies in the simulation of liquid freezing and crystallization process or metal solution cooling and annealing process. If the simulating process is sufficient enough, the algorithm can converge to the global optimal solution with the probability of 1. To maximum problems, the pseudo code is as follows

```
Begin
    t := 0;
    P(t) := InitPopulation();
    Evaluate(P(t));
    While (stop criteria unsatisfied OR T > T_min)
        P' (t) = NeighborSearch(P(t));
        For i = 1 to N
            e := I_i'(t) – I_i(t);
            If (e > 0)
                I_i(t + 1) = I_i'(t);
            Else if (exp(e / T) > random(0, 1))
                I_i(t + 1) = I_i'(t);
            Else
                I_i(t + 1) = I_i(t);
            End
        End
        T = r * T;
        t = t + 1;
    End
End
```

where t is the number of iterations, and N represents the population size. **P(t)** is the tth generation, and **P'(t)** means the new population generated by the neighborhood search operators. $I_i(t)$ and $I_i'(t)$ are two individuals of **P(t)** and **P'(t)**, respectively. T is annealing temperature, $0 < r < 1$ is the cooling rate, T_{min} is the lowest temperature. As can be seen, simulated annealing contains two fundamental operators, neighborhood search and annealing acceptance judgment. It can not only accept optimized solutions, but also accept limited deteriorated solutions with Metropolis principle.

Even though the convergence of the original form of simulated annealing can be proved from a probability perspective, its convergence speed is quite slow. So that many improvements have appeared to improve its performance, including modified cooling schedule [32], heating and annealing with memory [33], two-stage simulated annealing [34], etc. Moreover, due to the randomness and flexibility of the neighborhood search, the annealing acceptance rule is often stripped out to be the improvement of other algorithms, and hence getting a better suitability and search diversity to different problems. Actually, different acceptance functions will bring different influences to intelligent optimization algorithm.

1.4.2.2 Iterative Local Search

Iterative local search [35] is a representative neighborhood search algorithm which was firstly proposed by Lourenco in 2003. It is an extension of local or neighborhood search. In each generation, each individual tries to do local search within a small scope and update itself with better position. Through introducing a perturbation operator, the history information of the previous iteration can be used for guiding and the whole process local traverse scope can be fully enlarged. It is more flexible than the traditional local search. For different problems, the implementation of perturbation and local search operators needed to be redesigned. Its pseudo code is shown below:

```
Begin
    t := 0;
    P(t) := InitPopulation();
    Evaluate(P(t));
    P(t) := LocalSearch(P(t));
    B_est := P(t);
    While (stop criteria unsatisfied)
        P'(t)  = Pertubation(P(t));
        P*'(t) = LocalSearch(P'(t));
        P(t + 1) = AcceptanceCriterion(P*'(t), P'(t));
        t = t + 1;
    End
End
```

where t is the number of iterations, and $P(t)$ represents the individuals in the tth generation. **P'(t)** is the new population generated by perturbation and **P*'(t)** is the new population found by local search. Similar to simulated annealing, Iterative local search also contains acceptance criterion and local search operator. More than that, the perturbation based on current population **P(t)** is introduced to increase the diversity and guide the whole population do neighborhood search in different position.

The standalone use of iterative local search [36] has gotten great success in the fields of combinatorial optimization, such as travelling salesman problem and job-shop scheduling problems. Following that, many hybrid version of iterative local search are presented for scheduling in different environments, such as the hybridization with greedy strategy for unrelated parallel machine scheduling [37] and the hybridization with genetic algorithm for location-routing problem [38] and so on.

Though these neighborhood search methods were proposed relatively early, their ability for solving complex optimization problems is still strong. With the above neighborhood search structures, many deterministic searching algorithms can be partly introduced into the iterative process to enhance the exploitation and find better solutions. It can also be designed as a bridge between traditional deterministic algorithms and the new advanced intelligent algorithms.

1.4.3 Review of Swarm Intelligence Algorithm

Swarm intelligence algorithms search for optimal solution to the problem by individual collaboration and competition in population. The most representative ones are ant colony optimization and particle swarm optimization. Anty colony optimization simulates the pheromone secretion in ants foraging behavior, positioning population with pheromone and taking advantage of pheromone dissipation to avoid population premature convergence. Particle swarm optimization imitates birds' global and individual optimal learning mechanisms, integrating excellent global and individual information and hence implement directed cooperation. To sum up, ant colony optimization is suitable for combinatorial optimization problems such as path finding and scheduling, while particle swarm optimization is more applicable to complex continuous numerical problems.

1.4.3.1 Ant Colony Optimization

In 1990s, Dorigo proposed the first ant colony algorithm—ant system, and meanwhile applied them to the unsolved travelling salesman problem (TSP). From then on, basic ant colony optimization gets continuous development and improvement. Today, these different versions share a same characteristic, i.e., the improved ant detection ability in searching process. The pseudo code of commonly used ant colony optimization is:

Begin

t := 0;

P(t) := InitPopulation();

Evaluate(**P(t)**);

$\mathbf{P_{heromone}(t)}$:= InitPheromone(**P(t)**);

$\mathbf{P_{rior}}$:= InitPriorKnowledge();

While (stop criteria unsatisfied)

$P(t + 1) = FindPath(P(t), P_{heromone}(t), P_{rior})$;

Evaluate(**P(t + 1)**);

$P_{heromone}(t + 1) = UpdatePheromone(P(t + 1), P_{heromone}(t), P_{rior})$;

$t = t + 1$;

End

End

where t is the number of iterations, and $\mathbf{P(t)}$ represents the tth generation. $\mathbf{P_{heromone}(t)}$ and $\mathbf{P_{rior}}$ are the pheromone matrix of the tth generation and priori knowledge information matrix, respectively. They are used to guide ants' path finding behavior. Therefore, it contains two fundamental operators, path finding and pheromone updating, which aim at guiding the population searching by the integration of static priori knowledge and dynamic pheromones which are formed by every individuals' step.

As is shown, ant colony optimization is specifically designed to solve combinatorial optimization problems, and it has concurrency, robustness and positive feedback characteristics. Current mainstream improvements to ant colony optimization include:

- Improvements on foraging behavior (path-finding method)
- Improvements on pheromone updating
- Parameter self-adjustment.

An early representative example of improved method is elitist strategy [39]. Through elite screening, better paths are more likely to be chosen. But if there are too many elites, the algorithm will fall into a local optimum, resulting in the search premature stagnation. In order to overcome these exposed drawbacks, literature [40] proposed an ant colony system, which improved algorithm behavior selection rules and enhanced path pheromone of the global optimum. In the direct improvement on ant colony optimization, MAX-MIN ant system is a typical representative, who modified the way of pheromone updating, and helped to increase the search capabilities of algorithm in the initial stage [41]. Besides, researchers further analyzed and interpreted the invariance and classification of the algorithm in different point of views [42, 43]. Its applications ranged from TSP to quadratic assignment problem (QAP), vehicle routing problem (VRP) and robot path planning. However, although the applications and improvements are relatively diverse, its parameter setting, convergence, effectiveness all come from a large number of experimental results, which not only lack theoretical research like genetic algorithm, but also lack mature methods to guide the process analysis.

1.4.3.2 Particle Swarm Optimization

Kennedy et al. firstly designed particle swarm optimization in 1995. This algorithm conducts searching according to the pursuit of particles to the best individual in solution space, whose process is simple and easy to implement. PSO has simple parameters without complex adjustments, and its implementation is shown in the following pseudo code:

Begin

t := 0;

P(t) := InitPopulation();

Evaluate(**P(t)**);

$\mathbf{p_{best}}$:= the best solution founded by each individual

g_{best} := the best solution founded by the whole population

V(t) := InitVelocity();

While(stop criteria unsatisfied)

V(t + 1) = UpdateVelocity(**V(t)**, $\mathbf{p_{best}}$, g_{best});

P(t + 1) = UpdateLocation(**P(t)**, **V(t + 1)**;

Evaluate(**P(t + 1)**);

Update $\mathbf{p_{best}}$, g_{best} according to **P(t + 1)**;

$t = t + 1$;

End

End

where t denotes the number of iterations, **P(t)** is the tth-generation population, $\mathbf{p_{best}}$ and g_{best} represent the individuals' current best solution matrix and the global best solution, respectively. **V(t)** is called as velocity matrix which stores the updated incremental positions for all dimensions in each generation. Even if particle swarm optimization is divided into two coupled steps, the updating of particles' velocity and the changing of particles' position, it can still be considered as two procedures, which are self learning and global learning according to functions. Updating rate and position according to self optimization and global optimization share the same effects.

From its emergence to the present, particle swarm optimization is rapidly used in function optimization, neural networks, fuzzy systems control and other fields. Most researchers have focused mainly on the aspects of the algorithm structure and performance improvement, which can be classified as:

- Parameter setting
- Algorithm convergence
- Topology architecture
- Particle diversity.

In addition, there are also researches emphasize on population structure and hybridation with other algorithms. In parameter setting, typical examples are the introduction of inertia weight and nonlinear variation of inertia weight proposed [44], etc. From algorithm convergence, M Clerc introduced compression factor to maintain the convergence of algorithms [45]. Besides, J Kennedy's classical research about topology structure significantly affected the system performance [46]. As a new evolutionary algorithm, it is easy to implement and suitable to continuous problems. Yet most of the current researches focus on the applications. The study on algorithm internal mechanism is relatively rare. The immature model parameter settings, the position and velocity structure design rely too much on the experience, whose own structure is overlooked and yet to be perfected.

1.5 The Classification of Current Studies on Intelligent Optimization Algorithm

A variety of intelligent optimization algorithm has appeared one after another, and according to the research focus, existing research can be divided into: algorithm innovation, algorithm improvement, algorithm hybridization, algorithm parallelization and algorithm application. Here's a brief summary of these five aspects.

1.5.1 Algorithm Innovation

The algorithms mentioned above still have a poor diversity and limited searching ability, despite they have brought different improvements to the solution of many complex optimization problems. Inspired by different problems, many researchers are discovering brand new intelligent optimization algorithms to bring new operators to this field.

As is shown in Fig. 1.2, eye-catching algorithms include:

- Memetic Algorithm [14]: from the operators point of view, it is similar to a kind of genetic algorithm with local search, including selection, crossover and mutation, and local search operators;
- Harmony Search [47]: It simulates the timbre reconciliation of musicians, transforming and accepting individual gene-bits with a certain probability. The operating mechanism approximates simulated annealing.
- Artificial bee Colony Algorithm [48]: It selects multiple investigation bees to conduct local search combined with a number of preferred sites, and meanwhile to use the remaining individual to carry on cooperative random search. Bee colony algorithm is a rising star in the field of intelligent optimization algorithm;

- Quantum Evolutionary algorithm [49]: Using quantum coding and quantum-bit operations to achieve a probabilistic search, the coding method of this algorithm is wide used while the operators' application is relatively rare;
- Chemical Reaction Algorithm [50]: Chemical Reactions algorithm takes use of molecular collisions, combinations, split and cross to implement evolutionary change. Although the process of cross and molecular collisions are similar to that in genetic algorithm, this algorithm adds combination and split operators, expressing great potential in some combinatorial optimization problems;
- Plant Growth Algorithm [51]: Similar to Taboo search, it bases on single-individual to select growth point, generating a series of new individuals and selecting the best one to be the substitute of the original one. With a high complexity, it is only proper to continuous optimization problems;
- Group Leadership Algorithm [52]: Similar to bee colony algorithm, it selects leaders according to fitness values, and makes use of cross to achieve in-group local search and inter-group interaction.

What's more, there are also new heuristic evolutionary methods such as Invasive Weed Algorithm [53] and Firefly Algorithm [54], but we would not repeat here. To sum up, it can be found by the algorithms above that most of the operators in new optimization algorithm design are similar or even the same as that in classical genetic algorithm, ant colony algorithm and simulated annealing operators. Actually, only a few of them are brand new operators, and the effectiveness is yet to be further tested by experimental and theoretical proof. The performance of these new algorithms cannot and has never been compared in a standard platform, but only the rule pattern of them has gotten an innovative progress. Therefore, it is well worth of investigating that how to further find out the characteristics and potentials of these new algorithms to achieve higher level efficient search rather than just comparing them with conventional methods.

1.5.2 Algorithm Improvement

Another development branch of new intelligent optimization algorithms is the algorithm improvement. Based on fundamental principles, algorithm initialization, encoding method, operators and evolution strategy have different effects on the process of solution search in most problems. Thus, many researchers draw on the above four aspects to improve existing algorithms.

Algorithm Initialization: The method and application of algorithm initialization can be summarized as random initialization, sequential initialization, initialization with same starting point and priori information-based initialization. The most common used is random initialization, which has relatively less research currently.

Encoding Method: Encoding method had been illuminated in Sect. 1.4.1.1, i.e., the improvement of genetic algorithm. Existing researches concerning new

encoding method are increasingly less, and most mainly focus on typical encoding way according to specific problems.

Operators: The typical ways to improve an algorithm are parameter adjustment and operator adjustment. Since operators usually need different parameters in different problems or in different searching period of one problem, many researches concentrate on the design of population-state based parameter adaptive modification method. In addition, due to the blindness of search, most studies tend to balance the exploration and exploitation of algorithms according to their overall characteristics. Niche strategy, orthogonal strategy and priori knowledge-based strategy are examples in this case. By this mean, many improved operators come into being and can be widely used in different algorithms. By and large, operators are the core of algorithm, thus researches in this field are the most flourished point and have brought many improvements in intelligent optimization.

Evolutionary Strategy: Evolutionary strategy contains Elitist Strategy (i.e. optimal instead of the worst), the Sort Preferred strategy, Competition Strategy, etc. Elite Strategy is the most commonly used one. Although there is not much research regarding evolutionary strategies, the screening of new and old individuals is also an important part, which needs to be further studied.

1.5.3 Algorithm Hybridization

The iterative evolution modes of intelligent optimization algorithms are comparatively unified. Differences only lie in operators, which provide great space for development. Hitherto, according to the overall efficacy of different operators, people can arbitrarily amalgamate the algorithms to make improvements for solving different problems. Consequently, to algorithms that emphasize exploitation, we can introduce operators with more diversity from other algorithms, while to those focus on exploration, we can also add operators with more local mining ability. Moreover, self-adaptive search strategies can be added to improve the performance of hybrid algorithms as well. Given that operators are mainly implemented based on population, when comparing with the improvement of algorithms, the arbitrary combinations of operators in different algorithms are relatively easy and convenient. Therefore, both algorithm beginners and engineering employers can freely select different operators to design various hybrid algorithms and test their efficacy after simply understanding of algorithm iterative modes. Furthermore, how to select suitable algorithms from so many valuable improvements and hybridizations and to practically apply them is a problem that needs insightful discussion.

1.5.4 Algorithm Parallelization

Parallel intelligent optimization algorithms combine the high performance of super computers and the natural parallelism of the algorithms. It can greatly enhance algorithms' processing speed and expansion space. The only parallel strategy of optimization algorithm is population-based parallelism, which means dividing population into sub-group to implement concurrent searching. The earliest studies of parallelization are mainly based on genetic algorithm. From then on, people have proposed many schemes for it. All these works can be classified into three main frameworks, i.e., master-slave parallel model, coarse-grained parallel model and fine-grained parallel model [55]. These three frameworks made possible a variety of parallel intelligent optimization algorithms in recent decades, and the most typical examples are parallel genetic algorithms, parallel particle swarm optimizations and parallel ant colony optimizations.

At present, studies on the parallelism of algorithms mainly focus on three aspects that can largely influence the parallel performance: parallel topology, communicational migration cycle, and the number of communicational migration. Especially in topology structures, there exist single and bi-directional ring topology [56], grid topology [57] and hypercube topology [58] and so on. Presently, even though some research have compared different topology structures and concluded that single bi-directional ring topology was better [59], the role played by different topologies in parallelization is also different. According to the specific circumstances and problems, the performances of different topologies are still to be analyzed with migration cycle and migration number. Overview, there are two issues in parallel studies, which are:

Whether it is possible to design a new parallel mode, expect for population-based parallelism;

Densely connected parallel topology has a large information exchange, strong collaborations, but slow speed, while loosely connected one has smaller information exchange and collaboration, but faster speed.

So the problem of how to embody integrated topology structure to get best search performance urges researchers to conduct deeper analysis and discussion in different application background.

1.5.5 Algorithm Application

Except for the two research branches above, another important point is the application of algorithms. No matter in communication [12], project construction management [13], electronic engineering [60] or manufacturing related fields [61–64], intelligent optimization algorithms have brought varying degrees of efficiency to problem solving. From the standpoint of problem type, algorithm application can be divided into two aspects:

Functions optimization application: The optimization of some typical complex functions is often regarded as the algorithm performance evaluation criteria. Researchers construct many static and dynamic multimodal function such as concave/convex functions and high/low-dimensional functions and consider them as Benchmark to provide comparison for algorithms. And a variety of optimization algorithms aiming at functions optimization are applied to automatic control and power calculation area. In terms of automatic control, intelligent optimization algorithms are commonly used for continuous optimization problems like controller parameter adjustment, simulated control system and control rules learning and control system simulation. It makes possible further out-of-hand regulation and achieves systematic automation and intelligence with higher control accuracy. As to power calculation, intelligent optimization algorithms usually play important roles in the optimization of large-scale multidimensional energy consumption calculation function and circuit loss. It can improve decision-making qualities within a short time, largely lowering electricity power waste.

Combinational optimization application: With the more and more complex multi-domain system, the sizes of a variety of combinatorial optimization problems increase sharply. Thus, many NP-hard problems appear in real situations. The emergences of intelligent optimization algorithms give a way to these problems. For typical combinatorial optimization problems such as the traveling salesman problem, knapsack problem and packing problem,researchers have proposed many different algorithms to try to achieve decision-making efficiency and improvement. Moreover, in real engineering projects, intelligent optimization algorithm is applied to machinery manufacturing, image processing, machine learning and data mining. Among them, the most typical one is resource scheduling in manufacturing management. In fact, the proposal and development of intelligent optimization algorithms can effectively solve the problem of job shop scheduling, production planning and task allocation; In image processing area, intelligent optimization algorithms are often used for pattern recognition and image feature extraction to achieve precise identification and display of the image. While in machine learning, intelligent optimization algorithms often appear in the fuzzy control rule learning, classifier dynamic tuning and the screening and combination of learning rules to improve the intelligence of machine learning. When it comes to data mining, intelligent optimization algorithms can achieve the evolution of search rules and direct search in order to improve the accuracy.

Besides, in their respective fields, researchers have successfully developed various forms of decision-making system based on typical intelligent optimization algorithms such as the genetic algorithm and ant colony algorithm, and have brought much efficiency to field decision-making. From the perspective of optimization algorithm development and application, in studies based on most specific problems, most researchers tend to re-design algorithms or directly use one or several traditional algorithms and improvement strategies according to problem features. In experiments, they usually compare their methods with conventional certain algorithms and the early non-improved algorithms rather than with existing improved studies on intelligent optimization algorithms. As can be seen, there are

fewer relationships between the research focus on algorithm mechanisms and on applications. Therefore, it seems an urgent problem that how to combine the two together to accelerate algorithm development and make it more efficient in engineering area.

1.6 Development Trends

Intelligent optimization algorithms now remain a research focus in the field of artificial intelligence, whose main development direction is towards high efficiency and high intellectualization. However, despite this, when dealing with more and more large-scale complex optimization problems, they are still faced with challenges such as how to take care of multi-users and multi-applications at the same time. Therefore, throughout the development needs of the networked integrated manufacturing system for complex productions, we can summarize the development trend of intelligent optimization algorithms as follows: intelligent, service-oriented, application-oriented and user-centric.

1.6.1 Intellectualization

Intellectualization can be embodied from two aspects. One is that algorithms should have certain functions like parameter self-adjustment and dynamic self-adaptive operations when solving different problems. This kind of quality, which can be defined as function intellectualization, makes possible good performances for different types of problems. The other is that algorithms must have excellent recognition performance during the process of designing, implementing and applying. This quality is defined as application intellectualization.

Currently, many researchers have devoted to the function intellectualization of intelligent optimization algorithms, while the application intellectualization is rarely studied. For application, they tend to simply mix multiple operators, with the help of parameter adjustment and parallelism to adapt varying problems. However, "there is no free lunch" [65]. D.H Wolpert's theory had proven that enforcement of any algorithm to specific area in one aspect always needs additional cost in other aspects, not mention to a multi-suitable algorithm. In summary, great development potential lies in the application intellectualization field, and the problem of how to intelligently integrate different algorithms or operators to cater varying problems is a field that deserves deep investigation.

1.6.2 Service-Orientation

The same as services in networked manufacturing, service-orientation is designed to encapsulate intelligent optimization algorithms in the form of services and provide to clients in different areas. This requires universal interface design for different algorithms and simplified problem interfaces. However, complex issues abound, and their respective attributes are ever changing. So it is difficult to offer universal, service-oriented and efficient algorithms.

Of course, there are very few researches about the servitization of intelligent optimization algorithms, and most researchers only envisaged framework on it, but did not actually begin. Nevertheless, seeing from the rich concepts of networking and cloud and the convenience of service-oriented software and hardware, we could safely draw the conclusion that the service-oriented optimization algorithms are well worthy of studying in future.

1.6.3 Application-Oriented

In the development of intelligent optimization algorithm, application-oriented area can be said is the most important and attractive one in the direction trends. Many researchers have focused on the designing of algorithm based on Benchmark without tests and applications on specific practical problems. Moreover, there are many large-scale problems in reality, and most domain experts are not equipped with optimization algorithm-related knowledge. Among the existing algorithm improvements, only the selecting of suitable intelligent optimization algorithms has made many practitioners overwhelmed. Most applications are more likely to select common algorithms to improve and amalgamate, and only few can devote to design new algorithms to solve certain problems. Therefore, the repetition of many works and the waste of many excellent research studies on intelligent optimization algorithms are unavoidable.

Due to this situation, many researchers are considering to build a uniform platform so that many high-efficient algorithms and their improvements can be categorized and collected to provide reference to users in different industry. But how to build such a flexible, convenient and application-oriented platform is exactly both a research focus and a research difficulty.

1.6.4 User-Centric

We have mentioned in the analysis of algorithm classification that intelligent optimization algorithms have three kinds of users: algorithm beginners, algorithm

researchers and algorithm users. From the initial goal of algorithm design, algorithms should be user-centric, and different users' demands vary.

To algorithm beginners, they hope to understand the classification and features of many basic algorithms, and to learn their principles, structures and processes.

To algorithm researchers, they not only need to study existing algorithms and their improvements, but also be able to conduct tests and theoretic analysis in a standard platform according to different problems.

To algorithm users, they may have no deep understanding to algorithms, and in real applications, they just need to find out corresponding optimization algorithms and one or two improvements to encode and design. The study and improvement of algorithms are too complicated to them.

Intelligent optimization algorithms' research are rarely concerned about user-centric problem. But to the expansion of intelligent optimization algorithms, this problem can never be underestimated. Therefore, user-centric algorithm design and improvement is another potential mainstream trend in future.

1.7 Summary

Intelligent optimization algorithms are beneficial to the following problems.

Large-scale optimization problems, which are difficult to get solution with deterministic algorithm on limited computing resources.

Real time decision problems, which require the algorithm with high time efficiency. Suboptimal solutions are generally acceptable in such problems. When dealing with such problems, intelligent optimization algorithms could bring very high solving efficiency.

In this chapter, referring their functions and features, we divided intelligent optimization algorithms into evolutionary study, neighborhood Search and population intelligence and also respectively elaborated their research outlines and basic principles. This classification is preliminarily achieved based on main functions, and although operation differences lie between different kinds of algorithms, their mechanisms are probably the same. Moreover, many improvements are actually the hybridization of different algorithms, so there is no significant boundary between them. The research of algorithms has been abundant, but the pace of development has not stopped. We briefly summarized the development trend of algorithms in this chapter. In prospect, the theories, features and applications of intelligent optimization algorithms are still the research focus in many fields of human endeavor, which belong to a key technology in interdisciplinary decision optimization.

References

1. Nocedal J, Wright SJ (2006) Numerical optimization. Springer, Berlin
2. Bonnans JF, Gilbert JC, Lemarechal C, Sagastizabal CA (2006) Numerical optimization: theoretical and practical aspects. Springer, Berlin
3. Papadimitriou CH, Steiglitz K (1998) Combinatorial optimization: algorithms and complexity. Dover Publications, Mineola
4. Schrijver A (2003) Combinatorial optimization. Springer, Berlin
5. Blum C, Roli A (2003) Metaheuristics in combinatorial optimization: overview and conceptual comparison. J ACM Comput Surv (CSUR) 35(3)68–308
6. Garey MR, Johnson DS (1990) Computers and intractability: a guide to the theory of NP-completeness. W. H Freeman and Co, San Francisco
7. Ullman JD (1975) NP-complete scheduling problems. J Comput Syst Sci 10(3):384–393
8. Gawiejnowics S (2008) Time-dependent scheduling. Springer, Berlin
9. Karp RM (1986) Combinatorics, complexity, and randomness. Commun ACM 29(2):98–109
10. Kann V (1992) On the approximability of NP-complete optimization problems. Royal Institute of Technology, Sweden
11. Talbi EG (2009) Metaheuristics: from design to implementation. Wiley, New york
12. Ribeiro CC, Martins SL, Rosseti I (2007) Metaheuristics for optimization problems in computer communications. Comput Commun 30(4):656–669
13. Liao TW, Egbelu PJ, Sarker BR, Leu SS (2011) Metaheuristics for project and construction management—a state-of-the-art review. Autom Constr 20(5):491–505
14. Moscato P (1989) On evolution, Search, optimization, genetic algorithms and martial arts: towards memetic algorithms. Caltech Concurrent Computation Program
15. Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11(4):341–359
16. Tao F, Zhang L, Zhang ZH, Nee AYC (2010) A quantum multi-agent evolutionary algorithm for selection of partners in a virtual enterprise. CIRP Ann Manufact Technol 59(1):485–488
17. Shi Y, Eberhart RC (2001) Fuzzy adaptive particle swarm optimization. In: Proceedings of the 2001 congress on evolutionary computation, vol 1, pp 101–106
18. Horn J, Nafpliotis N, Goldberg DE (1994) A niched pareto genetic algorithm for multiobjective optimization. In: Proceedings of the 1st IEEE congress on evolutionary computation, vol 1, pp 82–87
19. Wang DW, Yung KL, Lp WH (2001) A heuristic genetic algorithm for subcontractor selection in a global manufacturing environment. IEEE Trans Syst Man Cybern Part C 31(2):189–198
20. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197
21. March JG (1991) Exploration and exploitation in organizational learning. Organ Sci v2(1):71–87
22. Tsoulos IG (2008) Modifications of real code genetic algorithm for global optimization. Appl Math Comput 203(2):598–607
23. Zhang G, Gao L, Shi Y (2011) An effective genetic algorithm for the flexible job-shop scheduling problem. Expert Syst Appl 38(4):3563–3573
24. Zhang G (2011) Quantum-inspired evolutionary algorithms: a survey and empirical study. J Heuristics 17(3):303–351
25. Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Kluwer Academic Publishers, Boston
26. Whitley D (1994) A genetic algorithm tutorial. Stat Comput 4(2):65–85
27. Schmitt LM (2001) Theory of genetic algorithms. Theoret Comput Sci 259(1–2):1–61
28. Wang L, Pan J, Jiao LC (2000) The immune algorithm. ACTA Electronica Sinica 28(7):74–78
29. Wang L, Pan J, Jiao LC (2000) The immune programming. Chin J Comput 23(8):806–812

30. de Castro LN, Von Zuben FJ (2002) Learning and optimization using the clonal selection principle. IEEE Trans Evol Comput 6(3):239–251
31. Hofmeyr SA, Forrest S (2000) Architecture for an artificial immune system. Evol Comput 8(4):443–473
32. Noutani Y, Andresen B (1998) A comparison of simulated annealing cooling strategies. J Phys A: Math Gen 41(31):8373–8385
33. Ali MM, Torn A, Viitanen S (2002) A direct search variant of the simulated annealing algorithm for optimization involving continuous variables. Comput Oper Res 29(1):87–102
34. Varanelli JM (1996) On the acceleration of simulated annealing. University of Virginia, USA
35. Lourenco HR, Martin O, Stutzle T (2003) Iterated local search. Int Ser Oper Res Manag Sci 57:321–353 (Handbook of Metaheuristics. Kluwer Academic Publishers)
36. Lourenco HR, Martin O, Stutzle T (2010) Iterated local search: framework and applications. Int Ser Oper Res Manag Sci 146:363–397 (Handbook of Metaheuristics, 2nd edn. Kluwer Academic Publishers)
37. Fanjul-Peyro L, Ruiz R (2010) Iterated greedy local search methods for unrelated parallel machine scheduling. Eur J Oper Res 207(1):55–69
38. Derbel H, Jarboui B, Hanafi S, Chabchoub H (2012) Genetic algorithm with iterated local search for solving a location-routing problem. Expert Syst Appl 39(3):2865–2871
39. Dorigo M, Maniezzo V, Colorn A (1996) The ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern 26(1):29–42
40. Dorigo M, Gambardella M (1997) Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans Evol Comput 1(1):53–66
41. Stutzle T, Hoos HH (2000) MAX-MIN ant system. Future Gener Comput Syst 16(8):889–914
42. Birattari M, Pellegrini P, Dorigo M (2007) On the invariance of ant colony optimization. IEEE Trans Evol Comput 11(6):732–742
43. Martens D, De Backer M, Haesen R, Vanthienen J, Snoeck M, Baesens B (2007) Classification with ant colony optimization. IEEE Trans Evol Comput 11(5):651–665
44. Chatterjee A, Siarry P (2006) Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization. Comput Oper Res 33(3):859–871
45. Clerc M, Kennedy J (2002) The particle swarm-explosion, stability and convergence in a multidimensional complex space. IEEE Trans Evol Comput 6(2):58–73
46. Kennedy J, Mendes R (2002) Population structure and particle swarm performance. In: IEEE 2th proceedings of evolutionary computation, pp 1671–1676
47. Geem ZW, Kim JH, Loganathan GV (2001) A new heuristic optimization algorithm: harmony search. Simulation
48. Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Global Optim 39(3):459–471
49. Platel MD, Schliebs S, Kasabov N (2009) Quantum-inspired evolutionary algorithm: a multimodel EDA. IEEE Trans Evol Comput 13(6):1218–1232
50. Lam AYS, Li VOK (2010) Chemical-reaction-inspired metaheuristic for optimization. IEEE Trans Evol Comput 14(3):381–399
51. Wang C, Cheng HZ (2008) Optimization of network configuration in large distribution systems using plant growth simulation algorithm. IEEE Trans Power Syst 23(1):119–126
52. Daskin A, Kais S (2011) Group leaders optimization algorithm. Mol Pheys 109(5):761–772
53. Mehrabian AR, Lucas C (2006) A novel numerical optimization algorithm inspired from weed colonization. Ecol Inform 1(4):355–366
54. Yang XS(2008) Nature-inspired metaheuristic algorithms. Luniver Press
55. Muhlenbein H, Schomisch M, Born J (1991) The parallel genetic algorithm as function optimizer[J]. Parallel Comput 17(6–7):619–632
56. Yang HT, Yang PC, Huang CL (1997) A parallel genetic algorithm approach to solving the unit commitment problem: implementation on the transputer networks. IEEE Trans Power Syst 12(2):661–668
57. Fukuyama Y, Chiang HD (1996) A parallel genetic algorithm for generation expansion planning. IEEE Trans Power Syst 11(2):955–961

58. Xu DJ, Daley ML (1995) Design of optimal digital-filter using a parallel genetic algorithm. IEEE Trans Circ Syst 42(10):673–675
59. Matsumura T, Nakamura M, Okech J, Onaga K (1998) A parallel and distributed genetic algorithm on loosely-coupled multiprocessor system. IEICE Trans Fundam Elect Commun Comput Sci 81(4):540–546
60. Yeung SH, Chan WS, Ng KT, Man KF (2012) Computational optimization algorithms for antennas and RF/microwave circuit designs: an overview. IEEE Trans Industr Inf 8(2):216–227
61. Tao F, Zhao DM, Hu YF, Zhou ZD (2008) Resource service composition and its optimal-selection based on particle swarm optimization in manufacturing grid system. IEEE Trans Industr Inf 4(4):315–327
62. Tang KS, Yin RJ, Kwong S, Ng KT, Man KF (2011) A theoretical development and analysis of jumping gene genetic algorithm. IEEE Trans Industr Inf 7(3):408–418
63. Lo CH, Fung EHK, Wong YK (2009) Intelligent automatic fault detection for actuator failures in aircraft. IEEE Trans Industr Inf 5(1):50–55
64. Hur SH, Katebi R, Taylor A (2011) Modeling and control of a plastic film manufacturing web process. IEEE Trans Industr Inf 7(2):171–178
65. Wolpert DH (1997) W G Macready (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82
66. Holland J (1975) Adaptation in natural and artificial systems. The University of Michigan Press
67. Glover F (1989) Tabu search. ORSA J Comput 1(3):190–206
68. Kirkpatrick S, Gelatt CD, Vechi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680
69. Farmer JD, Packard NH, Perelson AS (1986) The immune system, adaptation, and machine learning. Physica D 22(1–3):187–204
70. Dorigo M (1992) Optimization, learning and natural algorithms. Ph.D. Thesis, Politecnico di Milanno
71. Adleman LM (1994) Molecular computation of solutions to combinatorial problem. Science 266(5187):1021–1024
72. Reynolds RG (1994) An introduction to cultural algorithms. In: The 3rd annual conference on evolution programming
73. Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: IEEE international conference on neural networks
74. Linhares A (1998) State-space search strategies gleaned from animal behavior: a traveling salesman experiment. Biol Cybern 87(3):167–173
75. Li XL (2003) A new intelligent optimization algorithm—artificial fish school algorithm. Ph.D. Thesis, Zhejiang University, China
76. Yang XS (2010) A new metaheuristic bat-inspired algorithm. Nature inspired cooperative strategies for optimization (NICSO 2010), Springer, Berlin, p 65–74