Chapter 4
Improvement and Hybridization of Intelligent Optimization Algorithm

Algorithm improvement and hybridization are two important branches in the development of intelligent optimization algorithm. Today there are already hundreds of improvement forms in evolutionary algorithms and neighborhood search algorithms, more than 20 improvements forms in swarm intelligent algorithms and various hybridization structures. We cannot exactly count how many repetitions in these improvement and hybridization for different problems. It's harder for researchers to test all of them in different problems with different environments and compare them one by one. However, with the idea of configuration, we can extract the operators in different intelligent optimization algorithm and their improvement and hybridization forms as independent modules, recombine them and make full use of them in different problems.

In this chapter, from the perspective of algorithm improvement and hybridization, we introduce the improvements in four aspects of intelligent optimization algorithm, i.e., initialization, encoding, operator and evolutionary strategy, and elaborate the hybridizations in three aspects, that are exploration, exploitation and adaptation, as shown in Fig. 4.1. Further, the application of dynamic configuration in algorithm improvement and hybridization are detailed and discussed based on several typical intelligent optimization algorithms commonly used in manufacturing.

4.1 Introduction

Based on genetic algorithm, particle swarm optimization and ant colony optimization and so on, a number of new variations on intelligent optimization algorithm are evolved and developed [1, 2]. Generally, we classify these variations as improvement and hybridization. Improvement includes changing, increasing or deleting part of operators based on original algorithm flow, such as the parameter

F. Tao et al., *Configurable Intelligent Optimization Algorithm*,
Springer Series in Advanced Manufacturing,
DOI: 10.1007/978-3-319-08840-2_4

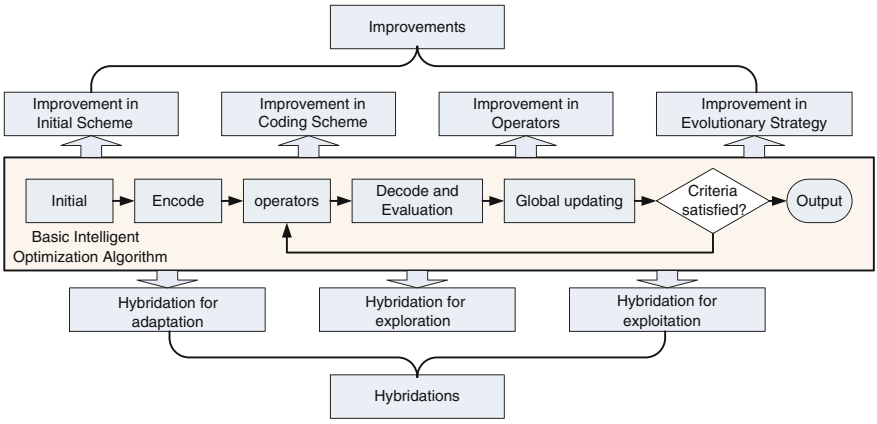

Fig. 4.1 The classification of algorithm improvement and hybridization

adaptive adjustment in crossover and mutation and the addition of niche strategy. Hybridization consists of all kinds of combinations of part of operators in different algorithms, examples are hybridization of genetic algorithm and particle swarm optimization in which the crossover and selection are applied and combined with the learning operators of particle swarm. Taken as a whole, the improvement places emphasis on the modifications in operators, while the hybridization mainly focuses on the recombination of different operators.

From the macroscopic angle, both improvement and hybridization are established for better efficiency. From the angle of specific goals, the improvement and hybridization can be further divided in 4 kinds, the reduction of time consumption, the improvement of solving accuracy, the enhancing of algorithm stability and the handling of searching convergence.

The reduction of time consumption: Many complex problems have the requirement of decision timeliness because of their changing states. Dynamic parameter adjustment in process control and live migration of tasks in production line are typical instances [3, 4]. To make sure the timeliness of the system states, people commonly choose operators with low time complexity and high exploration and simplify the complex variable relationships with fuzzy mapping [5].

The improvement of solving accuracy: As we know, algorithm cannot have the highest searching ability and the lowest time complexity both. Therefore, contrary to algorithm design for time reduction, some designers try to improve the searching ability of algorithm with the sacrifice of searching time for some problems which require higher solving accuracy. With different requirement, designers always apply some exploitation strategies and heuristics to do some local traversal search to improve algorithm searching ability [6, 7].

The enhancing of algorithm stability: For especially large-scale problems, intelligent optimization algorithm generally has some randomness. The fluctuation of solution results in several runs can be represented by algorithm stability. With

different initial population and middle random operators, the solution quality is hard to ensure. Thus the enhancing of algorithm stability becomes quite important. Researchers generally attempt to normalize the initialization and fix the searching direction to narrow the differences of results in various runs [8, 9].

The handling of algorithm convergence: With this target, some of researchers mainly focus on the avoidance of divergence which caused by operators with high exploration, low exploitation and behavior without collaboration [10, 11]. Others generally concentrate on enhancing the searching diversity to avoid the premature convergence which is responsible for low exploration and unbalanced local searching [12, 13].

According to the "no free lunch" law [14], no algorithm can obtain comprehensive performance improvement in the above four aspects simultaneously. With numerous improvements and hybridizations, the accuracy, time consumption, stability and convergence of intelligent optimization algorithms have obtained large progress in different complex manufacturing problems [15–17]. Based on the four objectives, the classification of improvement and hybridization in intelligent optimization algorithm are summarized in the following sections.

4.2 Classification of Improvement

The improvement of intelligent optimization algorithm indicates the modification of part of operators in algorithm flow. According to the uniform searching process, the improvement can fall into four sides, improvement in initial scheme, improvement in coding scheme, improvement in operator and improvement in evolutionary strategy.

4.2.1 Improvement in Initial Scheme

Initialization contains the population generation and parameter assignment. It decides the initial positions of individuals in the solution space. Good initial allocation can help the algorithm obtaining better solutions, while bad initial allocation may result in premature convergence. Besides, uneven distribution of individuals can lowering the algorithm stability to some extent, and too intensive or fixed allocation also can make the whole searching with low diversity. As described in Sect. 1.5.2, the most commonly used initial schemes include random initialization, sequential initialization and rule-based initialization.

Random initialization: It means to randomly assign values for different individuals in the domain. As the most commonly used scheme, it is independent of the specific problem and easy to implement. In this scheme, uneven allocation is one of its main drawbacks. With different implement environment and different methods of generating pseudorandom numbers, it is quite unstable. Moreover, in

large-scale solution space, a relatively small number of individuals may fall into bad allocations to a great degree. The main solution for the above situations are dividing the solution space and applying more even random scheme such as Monte Carlo or Mersenne Twister in each sub-spaces, or generating more individuals and filter better ones with good allocations [18–20].

Sequential initialization: This method refers to assign regular sequential numbers to individuals successively. During the process, sequential numbers are generated by dividing the variable domain with equal length in each dimension. It makes sure that the individuals evenly distribute in the whole solution space. Yet the fixed positions established by sequential initialization bring some drawbacks to algorithm. Firstly, it may bring about low diversity at the beginning of search. Many corners in the solution spaces are tend to become blind area. Secondly, if the distance of these individuals and the global optimal solutions are far, then better solutions can not be easily found. These may lead to premature convergence in a large degree.

Rule-based initialization: It refers to initialize the individuals according to problem property and environment-based rules. For different problems, people need to design specific rules to guide the initialization. The rules can be defined in accordance with some state forecasting and problem priori-knowledge. For instance, we can firstly divide the solution space, define some optimal positions in the sub-spaces and use these optimal positions as initial allocation. With different problem-based rules, better searching pace and solution quality can be obtained compared to the above methods. The main drawback of the rule-based initialization is that it is problem-dependent with low scalability. Some researchers also use deterministic algorithm beforehand to generate initial solutions and realize the hybridization of deterministic algorithm and intelligent optimization algorithm [21–23].

Beyond that, there are many other initialization schemes in different kinds of intelligent optimization algorithms [24, 25]. On account of the low efficiency of sequential initialization, random initialization is the most widely applied because it is more likely to be implemented. Moreover, although rule-based initialization scheme is hard to design, it is also broadly used in engineering for better searching ability. In some cases, the hybridization of random initialization and rule-based initialization might be a great way to obtain a good initial allocation with high diversity.

4.2.2 Improvement in Coding Scheme

Encoding are defined as the transformation or mapping between individual genes and problem variables. Except direct real number coding scheme, almost all other coding ways need extra time to execute transformation in iteration, which can straightforwardly increase time complexity. The major influences of encoding scheme are increasing algorithm diversity and enhancing the searching ability by

cooperating with operators. Thus efforts on encoding primarily aim at the improvement of solving accuracy and the handling of algorithm convergence. As described in Sect. 1.4.1, most typical coding schemes are designed based on genetic algorithm and gradually generalized to other intelligent optimization algorithms. At present, many newly-developing encoding schemes are established for specific manufacturing optimization problems [26, 27]. Yet the most applied coding scheme in engineering are still real coding, binary coding and matrix coding and so forth [28, 29]. Here we list some representative ones.

Real coding: As introduced above, no matter in continuous or discrete optimization, the encoding and decoding operations in iteration can be avoided. It is suitable for big variables and can effectively reduce the time complexity of algorithm.

Binary coding: In this scheme, variable in each dimension is mapped as a group of binary (or Boolean) genes. It is the eldest coding scheme in intelligent optimization algorithm. When applying it, the length of genes for each variable must be defined previously. When the accuracy requirement is higher, we need very long genes to present big variables. Hence, compared with real coding, it will largely increase the searching time especially in large-scale problems with big variables and high accuracy requirement.

Matrix coding: As the same as binary coding, each variable can be represented by a line of individual genes. The genes can be either Boolean or real number. It is designed especially for discrete combinatorial optimization. Typical time sequence, symbol string and multi-dimension position can all be directly represented by matrix coding. It is more flexible than the above coding scheme, yet will take more memory space and more complex.

Quantum coding: It is a new coding scheme which is used very frequently in recent research. Borrow the dimorphism of dual quantum genes and quantum rotation door, it maps the variables to quantum genes by two steps. This can bring high diversity to population during the whole searching process. However, dimorphic quantum genes will also bring two step operations both in encoding and decoding and take more memory space in the programming, so as to lower the decision efficiency.

With changing environment in production and manufacturing system, researchers and engineers have made great efforts on digging new encoding schemes [30, 31]. No matter in what coding scheme, the main consideration is the encoding and decoding complexity without the loss of searching accuracy. In different coding scheme, especially for combinatorial optimization, we may easily get into a case that multiple individuals mapping as one single solution, such as real coding scheme in job shop scheduling problem. In such situation, repetitive searching and uneven pace become inevitable, which further lower the whole searching quality of algorithm. Therefore, finding solutions for different kinds of many-to-one cases and designing efficient one-to-one mapping coding scheme are quite imperative in more and more complex manufacturing optimization.

4.2.3 Improvement in Operator

Operators are the core of intelligent optimization algorithm, so that improvement in operators attracts more attentions than other aspects. For exploration, random regeneration, chaotic changing, niche strategy [32, 33] and so on gives the algorithm more dynamics, but with less regularity, better solutions are hard to find and the algorithm stability will be very low. For exploitation, heuristics such as priori-knowledge, local search and greedy strategy [6, 34, 35] and so on brings more local searching ability which nevertheless makes the algorithm easy to trap into local convergence and do more repetitive searching. Only if two of them combined together can they establish a balanced searching pace and good solving efficiency. Today, many researchers focus on the balanced searching of intelligent optimization algorithm and design several exploration-oriented, exploitation-oriented and adaptive operators for solving complex problems. From the improvement ways we can divide the operator improvements into the adjustment of control parameter, the modification of operation and the increase of new independent operators.

The adjustment of control parameter: Control parameter in intelligent optimization algorithm, such as crossover and mutation probability in genetic algorithm, decides the strength of operations in iteration. Therefore, for dynamic problems, people are likely to design adaptive parameter control strategy in operators in line with population searching state to guide the operations. Typical examples are the weighted particle swarm optimization, the adaptive genetic algorithm and so forth. In the adaptive genetic algorithm, the crossover and mutate probabilities are decided by the average fitness value and the best fitness value of the whole population. If the average fitness value is close to the best fitness value, the two probabilities will become lower, so as to do more exploitation. If the average fitness value is far from the best fitness value, or the differences between the individuals are large, then the two probabilities will be higher in order to strengthen the crossover and mutation and generate more new diverse individuals. It can be clearly seen that changing weights and parameters can not only balance the exploration and exploitation during search, but also adapt the algorithm for different problem environments. Except the above mentioned parameter-based improvement, fuzzy adaptive theory and knowledge-based adaptive theory can both be introduced to different sorts of operators to enhance the adaptive ability of algorithms for complex problems. Many experiments and applications have verified that such adjustments in iteration can successfully balance most of intelligent optimization algorithms during their searching pace without the increase of time complexity [36, 37]. Thus they are widely applied in engineering.

The modification of operation: It primarily refers to the structure modification in operators according to different problem requirements and the coding scheme in earlier stage. The most representative ones are multi-point crossover, chaotic mutation, discrete particle swarm learning and record list-based path finding and so on. Some of them are inclined to magnify the operation strength or range to increase algorithm diversity. Some of them are apt to choose part of individuals and add new

operations to enhance the algorithm excavating ability. These modifications noted above are normally independent of specific problem, so that they can be widely used in different environments. Except that, there are also some problem oriented modifications, such as the adding of priori information in ant colony optimization and immune algorithm, which mainly aim at increase the searching accuracy and make the algorithm more suitable for a specific problem. The difference with the above general modification is that the problem-oriented modification can only applied in a particular environment for a particular decision. At the same time, many people additionally focus on modifying the existing operators to adapt the specially designed coding scheme, such as the operator improvement of discrete particle swarm optimization for the particular coding scheme of job shop scheduling and the new record list improvement in ant colony optimization for solving continuous problems. In this point, they may basically change the operational mechanism with some new customized strategies. From all these improvements it can be seen that, when the problem to be solved and its encoding scheme are certain, the operators can all be divided into several sub-modules with population inputs and outputs according to the iterative steps. With the variation of these sub-modules, the searching direction and ability can both be changed.

The increase of new independent operators: Because the existing operators in an algorithm may possess weak capability either in exploration, or in exploitation, researchers generally tend to increase new independent operator to compensate the algorithm in a certain aspect. It is similar to algorithm hybridization, but unlike hybridization, the new operators are newly designed in accordance with the existing ones and do not belong to other algorithms. Characteristic cases are niche strategy, local search strategy and sorting based strategies and so on. For instance, niche strategy are designed before individual optimal selection, which use the hamming distance to get similar individuals and multiple a penalty function to make the weak ones been weeded out. Besides, local search strategy which tries to traverse a small range of space and find the best solution can be widely applied in different algorithms in any step and increase the algorithm exploitation. Most of these augmentations will increase the time complexity of algorithm to some extent, thus they can only be used in the problems which have low requirement on decision timeliness.

In recent times, there are still more and more improved operators springing up for diverse decision scenes and being cross-utilized in a bunch of manufacturing problems [38, 39]. But most of them have only been verified in merely one or two specific problem in theory and tests, not in wider practice.

4.2.4 Improvement in Evolutionary Strategy

Evolutionary strategy are executed after operators in iteration for updating the population and record the best and worst positions during the whole searching process. Since the elite evolutionary strategy is widely applied, it is the least

studied component of intelligent optimization algorithm because its influence on algorithm is lower than the above parts. However, with filtrating of new individuals to replace old ones, unbalanced updating will largely slow down the searching speed and may lower the whole efficiency or even give rise to premature convergence. Thus it is also very important. Improvement research in evolutionary strategy is inclined to consider the enhancing of algorithm stability and the handling of algorithm convergence and handle single-objective and multi-objective problems in different manners.

For single objective optimization, evolutionary strategy is developed from the direct replace manner to the famous elite strategy. In elite strategy, if the best individual obtained by several operators are better than the global best record, then replace the global best one with the new best one, or we should replace the worst individual in current generation by the new best one. Elite strategy becomes almost a uniform basic part in classical intelligent optimization algorithm for simple-objective optimizations. Furthermore, typical improved evolutionary strategies also include the method that combining the new individuals and the old ones and doing filtration after the combination by fitness based sorting or hamming distance based selection. These strategies play an important pre-selection role to prevent population diverge and premature convergence.

For multi-objective optimizations with Pareto scheme, elite strategy becomes not that suitable. Instead, the combination of new and old individuals in generation and screening in frontier Pareto-set is used for population updating. Now the most widely applied strategy is non dominated sorting. It is similar with the fitness based sorting in single-objective optimization. The only difference is that it uses the non-dominating theory to separate the individuals into sorted layers. Additionally, for improving the population diversity, some probability-based elimination mechanisms are also used to delete similar individuals in a specific layer randomly. In this manner, if the randomly generated number is smaller than a threshold, then accept the corresponding individual goes into the next generation, or it will be eliminated. It can increase the searching diversity and ensure the algorithm convergence in actual fact.

In a word, with some certain manners, evolutionary strategy can also be modified and further enhance the algorithm searching ability. As a key assistant part in intelligent optimization algorithm, it is quite independent with specific problem and worth to be researched for improving the efficiency of general intelligent optimization algorithms.

4.3 Classification of Hybridization

Hybridization refers to recombine or rearrange parts of intelligent optimization algorithms to generate a new method. Because a large proportion of initialization and coding schemes can be widely used in intelligent optimization algorithms, the design of hybridization chiefly focus on the recombination or rearrangement

among operators in iteration. With different emphasis, hybridization can also be divided into three kinds, hybridization for exploration, hybridization for exploitation and hybridization for adaptation.

4.3.1 Hybridization for Exploration

Hybridization for exploration mostly designed to search wider solution space in a limited time. In such category, searching step and range-ability of algorithm is quite large, which keeps the population with high diversity. Characteristic operators for exploration are mutation, differential evolution operators and taboo search and so forth [40–42]. The following are brief reviews on these typical operators for hybridization.

Mutation, originate from genetic algorithm, is one of the mostly applied operators in hybridization. It can be independently used in various intelligent optimization algorithms to avoid premature convergence. For example, in particle swarm optimization, after the global and self learning operations finished, mutation can be employed to balance the high learning-oriented operators and generate some new individuals during iteration. Similarly, if we apply the mutation operator to immune clone algorithm, parts of population can break away from the clone rule and do search further. It can be said the mutation operator is a universal operator for improving exploration in intelligent optimization algorithm. Moreover, it should be noted that when using mutation in hybridization, the probability and range of it should be concerned with the environment changing. Because too larger mutation probability and range may bring about searching diverge, while too small operation may not work for improvement.

The operator of random differential evolution is also a typical one in hybridization for exploration. With differential computing of three randomly chosen individuals, it can used to enhance the exploration ability of algorithm. By controlling a differential factor, it can amplify the exploration step to enhance population diversity or shrink the step to realize mutual learning. More flexible than mutation, it can make the new individual evenly distributed in the solution space and realize more balanced exploration. For instance, if we combine the differential evolution operator with the path finding operator of ant colony optimization, after the path finding, the differential computing with three randomly individuals can effectively alleviate the premature convergence brought about by unevenly pheromone. Except that, we can also replace the mutation in genetic algorithm with differential computing to obtain an effective hybridization taking the problem environment into account.

Taboo search are also a most commonly used operator for exploration-oriented hybridization. Different with the above two independent operators, we need to additionally design new taboo list for each hybridization scheme in different problems. If we apply taboo search in genetic algorithm, we need to design the problem-specific taboo list, do taboo judgment after genetic operators (i.e. selection,

crossover and mutation) and update the taboo list with new records. The same principles can apply to other hybridizations such as taboo-based immune algorithm and taboo-based simulated annealing algorithm. Furthermore, we can not only design taboo list for individuals, but also for other searching parameters. It makes sure the diversity during the whole searching process and avoids repetitive iteration to some extent. The main drawbacks of applying taboo strategy in hybridization are, (1) taboo judgment during iteration may bring larger time complexity, (2) long taboo list may increases the space complexity. At present, it is still a good strategy for doing wider exploration in different intelligent optimization algorithms.

4.3.2 Hybridization for Exploitation

Hybridization for exploitation is designed after exploration to make the algorithm searching in local scope more efficiently. In hybridization for exploitation, most heuristics and local search operators are applied for searching in a narrow scope. Characteristic operators are immune operators, learning operators of particle swarm optimization and path finding operators of ant colony optimization [43–45].

Immune operators include two kinds, immune clone and immune vaccination. Immune clone tries to extract features of the global best individuals to guide the operation, while immune vaccination tend to use priori knowledge of the specific problem to change parts of individuals locally. Both of them need some rules to guide and are independent from other operations. Therefore, they can be used in anywhere with some exploration-oriented operators to form a new algorithm with good exploitation ability. However, because of the problem-dependent rules, not only the design of hybridization based on immune operators will become complex, but also the searching with iterative immune vaccination and extraction will take much longer time. In other words, they sacrifices time for quality. Many studies prove that with problem-based guidance, immune operators can truly bring great searching ability and realize combination of deterministic and non-deterministic decision with high accuracy.

Learning operators of particle swarm optimization include global learning and self learning. With the record of self-best and global-best positions, they can only be applied together to balance the searching pace. With single self-learning, individuals without cooperation will become chaos and cannot get evolution any more. In reverse, with single global-learning, the population will converge quickly into a bad position. Thus, only the integration of them can ensure the stable searching. Generally, they can be used with other operators as exploitation-enhanced algorithms to improve the cooperative and stable searching. For example, they can be introduced to chaotic optimization or genetic algorithm. After the original action, the learning operators can reduce the randomness and add some guidance for exploitation. Similar with immune operators, due to the record of self-best in each generation, both the time complexity and space complexity will be increased.

Path finding operator of ant colony optimization tries to generate new population according to the pheromone concentration and do pheromone updating after that. It is also similar to the above learning operators, which attempts to use self and global record to guide the search. With pheromone updating and recording, it also has high complexity in hybridization. The most famous hybridization schemes are the combination of ant colony optimization and genetic algorithm and the simulated annealing ant colony optimization, and so on. Owing to its high complexity, it generally recombined with some simple exploration-based operators in design. For example, if we combine path finding operator with crossover and mutation of genetic algorithm, it can reduce the randomness in individual interaction and improve the local exploitation with group cooperation.

At present, these exploitation-based operators are mostly recombined with crossover, mutation and simulated annealing operators. With lower exploration and higher exploitation in local scope, these hybridizations have performed good exploitation ability in different manufacturing problems. However, with high complexity in both design and execution, how to design a simplified exploitation-based operator without accuracy loss is also an essential problem.

4.3.3 Hybridization for Adaptation

Hybridization for adaptation is mainly designed for adaptively solving problems with dynamic or complex solution space. Take the consideration of the balance of exploration and exploitation, most researchers recombine more than two operators to get high exploration in earlier stage and turn to exploitation in later stage. Generally, many intelligent optimization algorithms lack the ability either in exploitation or in exploration. Thus the above two kinds of hybridizations attempt to design hybridization to fill the gaps. But those strategies may still have the unstable or unbalanced problems. For overcoming the weakness, hybridization for adaptation receive a lot of attentions.

The mostly used operators for adaptation are simulated annealing operator. It particularly refers to the annealing acceptance judgment used in various hybridizations. In early phases, the annealing acceptance probability is quite large so that many degraded individuals can be accepted for high diversity and exploration. With the decreasing temperature, the lowering probability makes lesser and lesser degraded individuals to be accepted, so as to improve the algorithm convergence on the other side. With annealing probability, it can control the whole searching process adaptively with other exploration or exploitation-oriented operators to perform balanced searching. Moreover, it is easy to implement without bring more complexity. Thus, simulated annealing based hybrid intelligent optimization algorithms are more and more designed and used for different problems as well.

Except that, more and more efforts are putting in design adaptive operators for hybridization or recombine adaptive operators for the balance of exploration and

exploitation [46–48]. The aim of hybridization for adaptation can be summarized as getting good ability and balance in exploration and exploitation with less searching time.

4.4 Improvement and Hybridization Based on DC-IA

We have listed the classification of improvement and hybridization of intelligent optimization algorithm and their typical examples in the above sections. For different kinds of manufacturing optimizations, although with the same operations, they perform totally different. People need to do a lot of work to repetitive design various improvements and hybridizations for specific problems. For reducing these repetition and make the existing algorithms more efficiently in diverse environments, we presented a new dynamic configuration strategy mentioned in the previous chapter. With the new strategy, we can separate the solving process as four modules, i.e. initialization, encoding, operators and population updating. The input and output of each module are both population. Moreover, we can also divide and stored these modules according to their features. With two kinds of classifications, improvement and hybridization based on DC-IA can be divided into two styles, (1) module-based improvement and hybridization, (2) process-based improvement and hybridization.

Module-based improvement and hybridization means to design uniform iterative operations with new or recombined modules. It comes down to operator-based configuration. In detail, it can be classified as cascade configuration, parallel configuration and mixed configuration, as shown in Figs. 4.2, 4.3 and 4.4.

Similar with traditional hybridization, cascade configuration means to simply select multiple modules with different functions and assign operational order for them. In this scheme, one single hybrid algorithm is generated without generation division and population division. Based on the method of DC-IOA, this way can largely improve the reutilization rate of operators and produce new hybridizations directly.

Different with cascade configuration, parallel configuration refers to divide population into several sub-populations and select groups of operators for different sub-populations. In such scheme, more operators can be combined to do diverse search. Specifically, with uniform initialization and encoding scheme, population is randomly divided into sub-populations. During iteration, each sub-population is modified by different group of operators. After all operators finished, the sub-populations are combined and evolved together with uniform evolutionary strategy and randomly divided again to sub-populations. Diverse groups of operators guarantee the diversity of the whole population, while the random division of sub-populations ensures each individual can be evenly updated by different groups of operators. The whole process is balanced and more flexible than cascade configuration. Moreover, more operators will not increase the whole time complexity of algorithm as a result of the multi sub-population scheme. The main drawback is

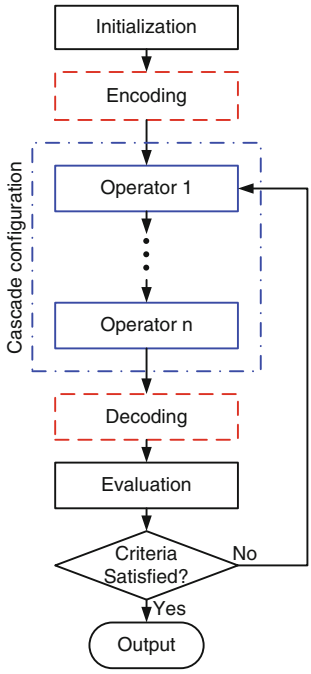

Fig. 4.2 Cascade configuration in module-based improvement and hybridization

that, it is hard to select suitable group number and design operators for several groups with high the coordination.

Further, based on cascade and parallel configuration, we can quickly implement mixed configuration. That is to say, we can perform cascade configuration with ordered simple operators and divide population into several sub-populations for parallel groups of operators. Sometimes, single cascade configuration is simple, stable but with low flexibility and diversity, while single parallel configuration ensures high diversity but inversely with low stability and interoperability. In such cases, we can combine cascade and parallel configuration, use cascade operators to enhance the interoperation and stability among sub-populations and perform parallel groups of operators to improve searching flexibility and diversity. However, it is harder to design than parallel configuration and the complexity is much higher.

Except that, we can perform dynamic configuration from the perspective of algorithm process, i.e. process-based improvement and hybridization. Different with the above module-based configuration, we can divide the generation process into several parts. Each part, containing multiple generations, performs different group of operators. It can be boiled down to algorithm-based configuration. Based on this frame, the process-based improvement and hybridization can be further divided as (1) process-based design with homogeneous coding and (2) process-based design with heterogeneous coding, as shown in Figs. 4.5 and 4.6.

In the first kind, the coding way in the whole process is unchanged. With uniform coding style, the process can be separated to two or more phrases. Each

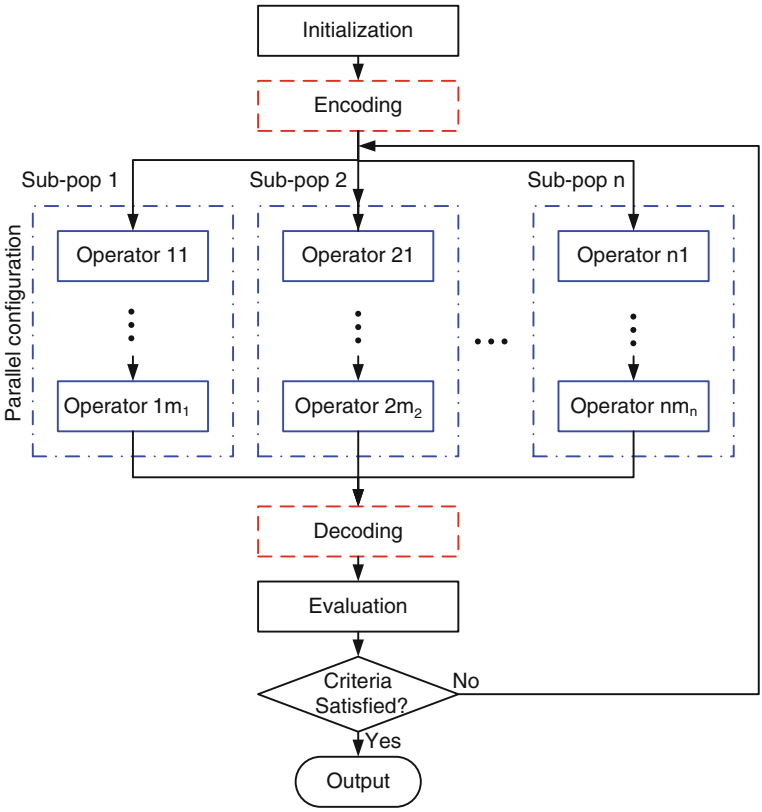

Fig. 4.3 Parallel configuration in module-based improvement and hybridization

phrase applies different group of operators. For example, with uniform binary coding scheme, we can apply genetic algorithm in generation 1–100, employ particle swarm optimization in generation 101–200 and perform ant colony optimization in generation 201–500, just as we described in Chap. 3. This configuration way is more flexible and simple than the module-based way, because users can easily obtain exploration in early stage, balance searching in middle stage and exploitation in later stage by using operators with corresponding functions, so as to make the process easier to control. In addition, with partition of searching process, the total time complexity will not be increased as well. For users with existing operators, this configuration way is much easier to design an efficient improved or hybrid algorithm than others.

Based on the process-based strategy, we can also use heterogeneous coding schemes in different parts of process. The only different is that it needs several trans-coding steps among different phrases. This allows us to applied more types of operators to further enhance diversity of operations. However, this will also largely increase the time complexity and decrease the searching efficiency.

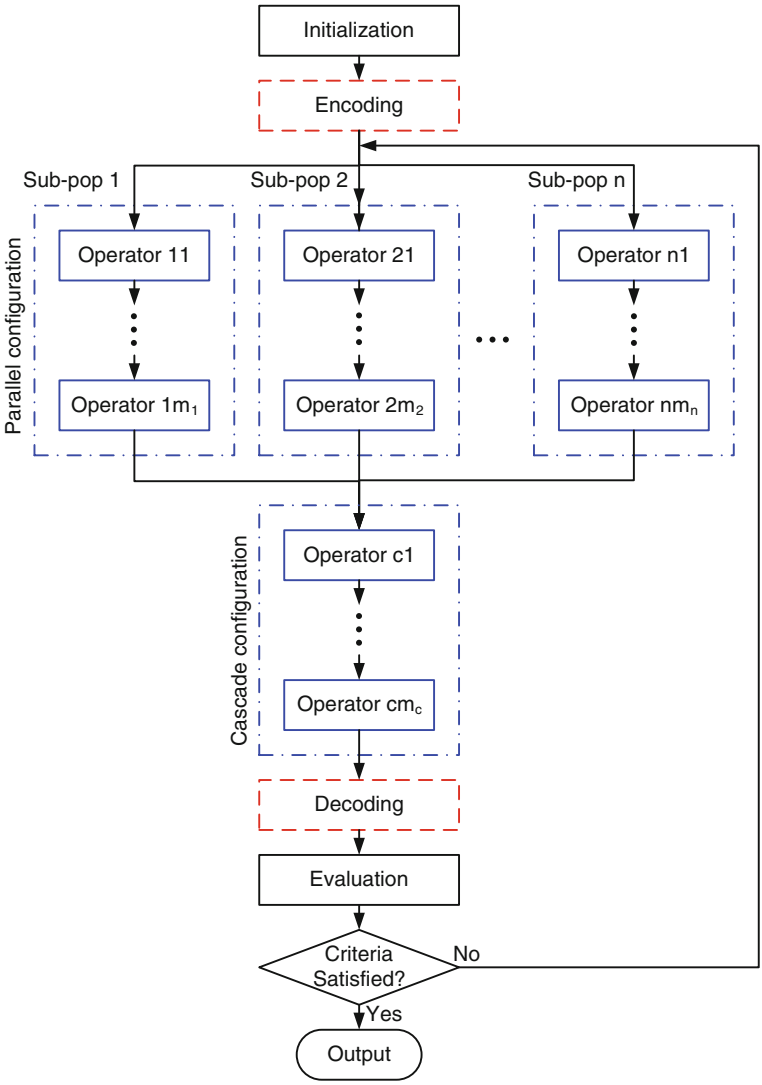

Fig. 4.4 Mixed configuration in module-based improvement and hybridization

It is worth noting that, for large-scale complex manufacturing optimization, we can combine both module-based configuration and process-based configuration to design improvement and hybridization. The design and test process in this case will become much harder. But with population and process divisions, the time complexity will not be increased much.

It can be seen, based on the concept of DC-IOA, with module-based and process-based improvement and hybridization, limited operators with uniform

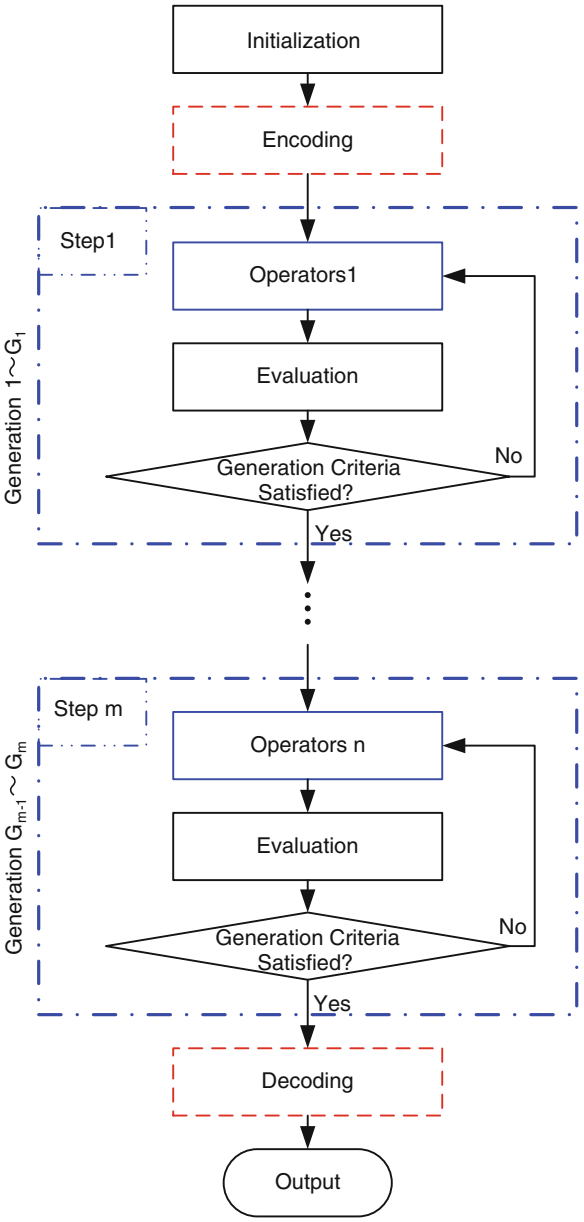

Fig. 4.5 Process-based improvement and hybridization with homogeneous coding

input and output can form hundreds of intelligent optimization algorithms with various types of structures. Not only the reutilization of existing operators is increased, but also a bunch of new dynamic improvement and hybridization schemes can be effectively applied for different kinds of problems.

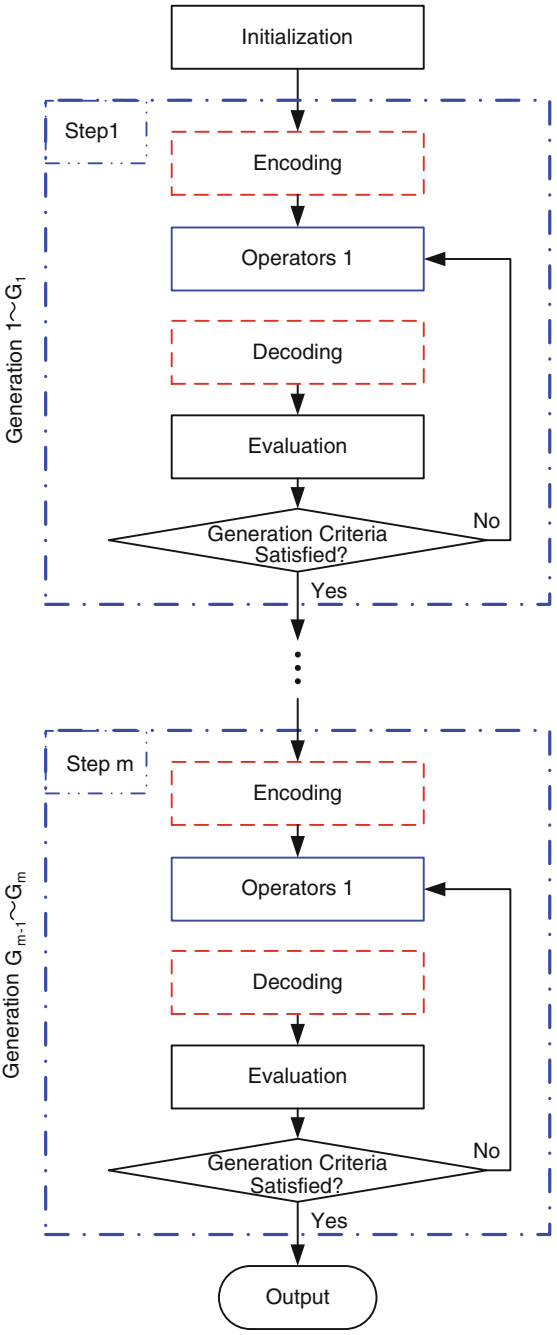

Fig. 4.6 Process-based improvement and hybridization with heterogeneous coding

4.5 Summary

Improvement and hybridization of intelligent optimization algorithm are always a focal point in the optimization of the whole life-cycle manufacturing. More than half of researches employs different sorts of improved or hybrid intelligent optimization algorithms in manufacturing optimizations, such as [49–53] and so on. This chapter introduced the improvement and hybridization of intelligent optimization algorithm, classified the improvement into four categories, i.e. improvement in initialization, improvement in coding scheme, improvement in operators and improvement in evolutionary strategy, and divided the hybridization into three kinds, i.e. hybridization for exploration, hybridization for exploitation and hybridization for adaptation. Then the detailed characteristics of the above categories are given with typical examples respectively.

Based on the existing manners and the new DC-IOA concept, we further present two kinds of new strategies for improvement and hybridization, (1) module-based improvement and hybridization and (2) process-based improvement and hybridization. The design process and features of the two kinds are elaborated. They are very practical and flexible and can play a guiding role in design and optimization in manufacturing. With the new design strategies, the flexibility and reusability of existing operators for different sorts of complex problems can be fundamentally enhanced.

References

1. Raidl GR (2006) A unified view on hybrid metaheuristics, hybrid metaheuristics. Lect Notes Comput Sci 4030:1–12
2. Parejo JA, Ruiz-Cortes A, Lozano S, Fernandez P (2012) Metaheuristic optimization frameworks: a survey and benchmarking. Soft Comput 16(3):527–561
3. Trappey AJC, Trappey CV, Wu CR (2010) Genetic algorithm dynamic performance evaluation for RFID reverse logistic management. Expert Syst Appl 37(11):7329–7335
4. Rao RV, Pawar PJ (2010) Parameter optimization of a multi-pass milling process using non-traditional optimization algorithms. Appl Soft Comput 10(2):445–456
5. Shen C, Wang L, Li Q (2007) Optimization of injection molding process parameters using combination of artificial neural network and genetic algorithm method. J Mater Process Technol 183(2–3):412–418
6. Moslehi G, Mahnam M (2011) A pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search. Int J Prod Econ 129(1):14–22
7. Yildiz AR (2013) Hybrid taguchi-differential evolution algorithm for optimization of multi-pass turning operations. Appl Soft Comput 13(3):1433–1439
8. Burnwal S, Deb S (2013) Scheduling optimization of flexible manufacturing system using cuckoo search-based approach. Int J Adv Manuf Technol 64:951–959
9. Yildiz AR (2009) An effective hybrid immune-hill climbing optimization approach for solving design and manufacturing optimization in industry. J Mater Process Technol 209(6):2773–2780
10. Duran N Rodriguez, Consalter LA (2010) Collaborative particle swarm optimization with a data mining technique for manufacturing cell design. Expert Syst Appl 37(2):1563–1567

11. Wang JQ, Sun SD, Si SB, Yang HA (2009) Theory of constraints product mix optimization based on immune algorithm. Int J Prod Res 47(16):4521–4543
12. Caponio A, Cascella GL, Neri F, Salvatore N, Sumner M (2007) A fast adaptive memetic algorithm for online and offline control design of PMSM drives. IEEE Trans Sys Man Cybern B Cybern 37(1):28–41
13. Yang WA, Guo Y, Liao WH (2011) Multi-objective optimization of multi-pass face milling using particle swarm intelligence. Int J Adv Manuf Technol 56(5–8):429–443
14. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82
15. Chandrasekaran M, Muralidhar M, Krishna CM, Dixit US (2010) Application of soft computing techniques in machining performance prediction and optimization: a literature review. Int J Adv Manuf Technol 46(5–8):445–464
16. Tiwari MK, Raghavendra N, Agrawal S, Goyal SK (2010) A hybrid taguchi-immune approach to optimize an integrated supply chain design problem with multiple shipping. Eur J Oper Res 201(1):95–106
17. Chan KY, Dillon TS, Kwong CK (2011) Modeling of a liquid epoxy molding process using a particle swarm optimization-based fuzzy reguression approach. IEEE Trans Industr Inf 7(1):148–158
18. Goicoechea HC, Olivieri AC (2002) Wavelength selection for multivariate calibration using a genetic algorithm: a novel initialization strategy. J Chem Inf Model 42(5):1146–1153
19. Zainuddin N, Yassin IM, Zabidi A, Hassan HA (2010) Optimizing filter parameters using particle swarm optimization. In: The 6th international colloquium on signal processing and its applications (CSPA) pp 21–23, May 1–6
20. Wang CM, Huang YF (2010) Self-adaptive harmony search algorithm for optimization. Expert Syst Appl 37(4):2826–2837
21. Zhang Y, Li X, Wang Q (2009) Hybrid genetic algorithm for permutation flowshop scheduling problems with total flowtime minimization. Eur J Oper Res 196(3):869–876
22. Hong SS, Yun J, Choi B, Kong J, Han MM (2012) Improved WTA problem solving method using a parallel genetic algorithm which applied the RMI initialization method. In: The 6th international conference on soft computing and intelligent systems, vol 20–24, pp 2189–2193
23. Yao HM, Cai MD, Wang JK, Hu RK, Liang Y (2013) A novel evolutionary algorithm with improved genetic operator and crossover strategy. Appl Mech Mater 411–414:1956–1965
24. Kazimipour B, Li X, Qin AK (2013) Initialization methods for large scale global optimization. IEEE Congr Evol Comput 20–23:2750–2757
25. Dimopoulos C, Zalzala AMS (2000) Recent developments in evolutionary computation for manufacturing optimization: problems, solutions, and comparisons. IEEE Trans Evol Comput 4(2):93–113
26. Fumi A, Scarabotti L, Schiraldi MM (2013) The effect of slot-code optimization in warehouse order picking. Int J Eng Bus Manag 5(20):1–10
27. 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 Manuf Technol 59(1):485–488
28. Oysu C, Bingul Z (2009) Application of heuristic and hybrid-GASA algorithms to tool-path optimization problem for minimizing airtime during machining. Eng Appl Artif Intell 22(3):389–396
29. Lv HG, Lu C (2010) An assembly sequence planning approach with a discrete particle swarm optimization algorithm. Int J Adv Manuf Technol 50(5–8):761–770
30. Kuo CC (2008) A novel coding scheme for practical economic dispatch by modified particle swarm approach. IEEE Trans Power Syst 23(4):1825–1835
31. Bhattacharya A, Kumar P (2010) Biogeography-based optimization for different economic load dispatch problems. IEEE Trans Power Syst 25(2):1064–1077
32. Laili YJ, Tao F, Zhang L, Cheng Y, Luo YL, Sarker BR (2013) A ranking chaos algorithm for dual scheduling of cloud service and computing resource in private cloud. Comput Ind 64(4):448–463

33. Perez E, Posada M, Herrera F (2012) Analysis of new niching genetic algorithms for finding multiple solutions in the job shop scheduling. J Intell Manuf 23(3):341–356
34. Prakash A, Chan FTS, Deshmukh SG (2011) FMS scheduling with knowledge based genetic algorithm approach. Expert Syst Appl 38(4):3161–3171
35. Tasgetiren MF, Pan QK, Suganthan PN, Buyukdagli Q (2013) A variable iterated greedy algorithm with differential evolution for the no-idle permutation flow shop scheduling problem. Comput Oper Res 40(7):1729–1743
36. Valente A, Carpanzano E (2011) Development of multi-level adaptive control and scheduling solutions for shop-floor automation in reconfigurable manufacturing systems. CIRP Ann Manuf Technol 60(1):449–452
37. Ye A, Li Z, Xie M (2010) Some improvements on adaptive genetic algorithms for reliability-related applications. Reliab Eng Syst Saf 95(2):120–126
38. Tao F, Qiao K, Zhang L, Li Z, Nee AYC (2012) GA-BHTR: an improved genetic algorithm for partner selection in virtual manufacturing. Int J Prod Res 50(8):2079–2100
39. Azadeh A, Miri-Nargesi SS, Goldansaz SM, Zoraghi N (2012) Design and implementation of an integrated taguchi method for continuous assessment and improvement of manufacturing systems. Int J Adv Manuf Technol 59(9–12):1073–1089
40. Wu TH, Chang CC, Yeh JY (2009) A hybrid heuristic algorithm adopting both boltzmann function and mufation operator for manufacturing cell formation problems. Int J Prod Econ 120(2):669–688
41. Wang L, Pan QK, Suganthan PN, Wang WH, Wang YM (2010) A novel hybrid discrete differential evolution a algorithm for blocking flow shop scheduling problems. Comput Oper Res 37(3):509–520
42. Li JQ, Pan QK, Liang YC (2010) An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems. Comput Ind Eng 59(4):647–662
43. Wang XJ, Gao L, Zhang CY, Shao XY (2010) A multi-objective genetic algorithm based on immune and entropy principle for flexible job-shop scheduling problem. Int J Adv Manuf Technol 51(5–8):757–767
44. Zhao F, Hong Y, Yu D, Yang Y (2013) A hybrid particle swarm optimization algorithm and fuzzy logic for processing planning and production scheduling integration in holonic manufacturing systems. Int J Comput Integr Manuf 23(1):20–39
45. Akpinar S, Bayhan GM, Baykasoglu A (2013) Hybridizing ant colony optimization via genetic algorithm for mixed-model assembly line balancing problem with sequence dependent setup times between tasks. Appl Soft Comput 13(1):574–589
46. Muller LF, Spoorendonk S, Pisinger D (2012) A hybrid adaptive large neighborhood search heuristic for lot-sizing with setup times. Eur J Oper Res 218(3):614–623
47. Moradinasab N, Shafaei R, Rabiee M, Ramezani P (2013) No-wait two stage hybrid flow shop scheduling with genetic and adaptive imperialist competitive algorithms. J Exp Theor Artif Intell 25(2):207–225
48. Yun YS, Moon C, Kim D (2009) Hybrid genetic algorithm with adaptive local search scheme for solving multistage-based supply chain problems. Comput Ind Eng 56(3):821–838
49. Yildiz AR (2009) Hybrid immune-simulated annealing algorithm for optimal design and manufacturing. Int J Mater Prod Technol 34(3):217–226
50. Noktehdan A, Karimi B, Kashan AH (2010) A differential evolution algorithm for the manufacturing cell formation problem using group based operators. Expert Syst Appl 37(7):4822–4829
51. Ho WH, Tsai JT, Lin BT, Chou JH (2009) Adaptive network-based fuzzy inference system for prediction of surface roughness in end milling process using hybrid taguchi-genetic learning algorithm. Expert Syst Appl 36(2):3216–3222
52. Zhang H, Zhu Y, Zou W, Yan X (2012) A hybrid multi-objective artificial bee colony algorithm for burdening optimization of copper strip production. Appl Math Model 36(6):2578–2591
53. Yildiz AR (2013) Optimization of cutting parameters in multi-pass turning using artificial bee colony-based approach. Inf Sci 220(20):399–407