Applying Modified Colonial Competitive Algorithm to Solve Minimal Hit ting Set Problems

Chuanjun Zhu[1], Jing Cao[1], Chaoyong Zhang[2], and Kunlei Lian[2]

[1]Hubei University of Technology, Wuhan, 430068

[2]State Key Laboratory of Digital Manufacturing Equipment & Technology, Huazhong University of Science and Technology, Wuhan, 430074

Abstract

A CCA was developed and modified to solve the problem of minimal candidates set. The minimal candidate set was a minimal hitting set. The modified CCA improved performance in initialization, to the population of countries maintained by CCA by introducing a third type of country, independent country, to the population of countries maintained by CCA. Implementation details of the proposed CCA and modified colonial competitive algorithm (MCCA) were elaborated using an illustrative example. The performance of the algorithms was analyzed, and the results by the MCCA were compared with DMDSE-Tree algorithm. When 90% of the minimal hitting sets are obtained, the MCCA has better efficiency. Finally, the experimental results of certain system verify the effectiveness of the algorithm, which proves that this method can be applied in solving the minimal hitting set of combinatorial optimization problems for selection of equipment effectively.

Key words: business planning; colonial competitive algorithm (CCA); minimal hitting set; equipment selection

1 Introduction

There exist many real-world problems that can be modelled as identifying the minimal set from the non-empty intersection with each set in a given group. The minimal hitting set problem refers to such problems and examples include the creation of model-based diagnosis from the smallest conflict set, the search for gene fragment to expound certain gene characteristics in gene expression analysis and the decision problem when collective uses make book purchases at public expense. There have been lots of efforts in the research community to improve efficiency of solving the minimal hitting set problem, which is NP-hard. Reiter proposed the first method in calculating the minimal hitting set problem in model-based diagnosis using HS-Tree, but it suffers from loss of feasible solutions due to its adoption of pruning and closing strategies in its search process. Greiner proposed an improved HS-Tree algorithm named HS-DAG that uses the idea of acyclic graphs in order to identify all the minimal hitting sets. The minimal hitting set problem has received lots of attentions from research community and many algorithms are proposed to solve this problem more effectively, which can be classified into exact algorithms and heuristic algorithms. Examples of exact algorithms include the algorithm based on BNB-HSSE, hybrid algorithm based on CHS tree and recursive boolean algorithm, hybrid algorithm based on HSSE tree and binary mark, branch reduction algorithm and algorithm based on dynamic maximum (DMDSE-Tree). These exact methods cannot satisfy the time and space requirements of engineering problems in large scale complex systems, and it is not necessary to find the optimal solutions in these systems. Therefore, heuristic algorithms have been proposed to approximate the optimal minimal hitting sets using algorithms like discrete particle swarm algorithm, binary particle swarm optimization, and low-cost heuristic algorithm STACCATO based on heuristic functions.

Atashpaz-Gargari et al. proposed a population-based colonial competitive algorithm (CCA) by simulating the process of colonial competition in human society evolvement. It shows better convergence rate when compared to genetic algorithm and particle swarm optimization and has received more and more applications in recent years. For example, Forouharfard et al. used the colonial competitive algorithm to solve the logistic planning problem in transshipment warehouses. Kaveh et al. applied the colonial competitive algorithm to the structural optimal design problem. Nazari et al. employed the colonial competitive algorithm for the product outsourcing problems in mix-model production. Sarayloo et al. applied the colonial competitive algorithm in solving the dynamic unitised manufacturing problem. In addition, numerous attempts have been made to improve the performance of colonial competitive algorithm and apply it to other combinatorial optimization problems, including mixed-model assembly line balancing problem and the data clustering problem. In this paper, we propose an improved colonial competitive algorithm to solve the minimal hitting set problem and also validate its performance using the enterprise equipment selection problem.

2 Improved colonial competitive algorithm for the minimal hitting set problem

In this paper, we propose an improved colonial competitive algorithm by introducing the concept of independent countries in addition to the existing two types of countries, namely, empires and colonies, in classical CCA. They are defined as follows:

- empire: empire countries strive to assimilate more colonies in the computational process of the algorithm.
- colony: colony countries aim to learn from their corresponding empires in order to become either independent countries or new empire countries.
- independent country: independent countries try to learn from and turn into empire countries.

The main differences between the improved CCA from the classical one exist in the country initialization, assimilation and update stages due to the introduction of independent countries in the algorithm workflow.

2.1 Empire initialization

The improved CCA requires three parameters, namely, N_{imp}, N_{col} and N_{ind}, to indicate the number of empires, colonies and independent countries in the initial population. Therefore, the total number of countries in the initialization step is $N_{pop} = N_{imp} + N_{col} + N_{ind}$. The algorithm first randomly creates N_{pop} solutions (countries) and selects the best N_{imp} countries as empires, the rest countries will be marked as colonies and independent countries based on the parameters N_{col} and N_{ind}. Note that independent countries are not influenced by any empire and therefore not part of any empires. Figure ? depicts the country classification in the initialization stage.

2.2 Solution encoding

Solution encoding scheme is the key step in applying colonial competitive algorithm to optimization problems and will greatly affect its performance thereafter. Binary encoding theme is the most widely used method since it is easy to implement.

For a set group U, an assimilation step is first conducted to remove all the sets that encompass any other sets, which can greatly reduce the number of hitting sets. Only the encompassed sets, denoted by U_i, are left after the assimilation process. Let n indicate the number of sets available in the set group after the assimilation process. Let R represent the union of all sets in the set group, that is, $R = \cup_{i=1,2,\cdots,n} U_i$, $|R| = N$. It is clear that the minimal hitting set is a proper subset of R and every element in R either exists or does not exist in the minimal hitting set. Therefore, a binary value can be used to indicate whether an element in R shows up in the minimal hitting set. In this way, the minimal hitting set can be represented by an array of length N and each element in the array must be a binary value of either 0 or 1.

In order to create good approximations of the minimal hitting set in the initial population, reduce the number of algorithm iterations, and decrease the minimizing calculations, the average number of array elements with value of 1 is set to be close to the number of elements in the minimal hitting set. Let X

denote the average number of elements in the minimal hitting set and β be the probability that an element takes the value of 1, then,

$$\beta = X/N \tag{1}$$

Since the value of X is unknown in general, the value of β must be approximated, which is empirically set to $0.1 \sim 0.5$. Note that the value of β increases when n increases.

Let Y be a chromosome and the gene at the ith ($i < N$) position be x_i, then $Y = \{x_1, x_2, \cdots, x_N\}$. The following method is used to create the initial solution based on the binary encoding scheme: randomly create a number $\xi_i \in (0, 1)$, let $x_i = 1$ if $\xi_i < \beta$; let $x_i = 0$ if $\xi_i \geq \beta$.

2.3 Fitness function

The definition of fitness function plays a key role in an algorithm's search efficiency. The fitness value shows the competitiveness of an solution regarding the objective function and it is also used in the algorithm to decide the solution's capacity to produce offsprings. Fitness function is often called objective function and is normally used to evaluate individual solution's quality. The objective function differs in different problems.

In order to assign an objective value to an individual, its closeness to the optimal solution must be evaluated. For the minimal hitting setting problem, there are two necessary conditions: 1) a solution must be a hitting set; 2) any of the solution's proper subset must not be a hitting set. This paper uses a objective function defined as $f(x) = t$, where t indicates the number of non-empty intersections of the current solution with every set in the set group. Note that the number of elements in a set cannot decide whether any proper subset of a set is a hitting set. The objective function drives a solution to converge to hitting set, not necessarily the minimal hitting set, and the transformation of the resulting hitting set into minimal hitting set is required.

2.4 Minimizing operators

As discussed in the previous section, the hitting set produced by a solution is not necessarily the minimal hitting set, and the purpose of the minimizing operators in this paper is to convert the superset into the corresponding minimal hitting set. To this purpose, all individual gene from an individual with value of 1 are converted into 0 and the fitness value is computed. If the fitness value does not change, the conversion is saved. This conversion is repeated for each gene within a solution until all genes are converted and the individual will produce a minimal hitting set.

Within an individual, the genes with values of 1 are often related, meaning that whether we can convert a gene from 1 to 0 is affected by other genes with values of 1. To decide whether elements are related, we need to check the locations corresponding to the elements after assimilation. If there are two or more elements in the assimilated set, the elements are related; otherwise, they are not related.

If we scan all the genes within a solution chromosome sequentially starting from the first gene, we can find that the probability of converting genes from 1 to 0 is higher for those located in the front of the chromosome than those in the back. In this case, individuals represented by genes in the front part of the chromosome have higher probability to show up in the minimal hitting set than those located in the back. In order to assign equal probability to genes regardless of the relative positions on the chromosome, a random scanning scheme is adopted to convert the superset into minimal hitting set.

The minimizing operator works as follows:

- check whether the current object is hitting set of not, if so, go to step (2), other make no changes to the object.
- starting scanning the chromosome at position k where k is a random number and $k < N$, convert a gene value from 1 to 0 if applicable and compute its objective value. Keep the change if the resulting solution is a hitting set; otherwise, reverse the change. Let $k = k + 1$. Let the scanning position be $k - N$ if $k \geq N$. After all the chromosome is scanned, the resulting objective represents a minimal hitting set and copy the distinct components into the final solution.

2.5 Competition

Similar to classical CCA, the improved CCA needs to calculate the total energy for each empire, which is determined by all the energies of its colonies:

$$E_{C_n} = J_n + \alpha \times \text{mean}\{J_m\} \tag{2}$$

where E_{C_n} is the total energy of the nth empire; $\alpha \in (0, 1)$ is used to control the weight of the all the colonies' energies in the final total energy of the empire. Note that a bigger α value indicates a smaller weight, whereas a smaller α value indicates a bigger weight. Normally, α is set to 0.1 In addition, J_n represents all the colonies' energy and J_m is the energy of the mth colony of the nth empire.

Another similarity between classical CCA and the improved CCA is that all the empires try to possession more colonies to increase their total energies. Some empires will have more and more colonies in the competing process, which will result in some empires losing their colonies gradually. The competition works as follows: 1) identify the weakest colony from the empire with the smallest total energy and set the colony as free; 2) all the empires compete for the free colony. The competition result is decided using a probability value that is associated with the empire's strength, in other words, more powerful empire has higher probability of getting the free colony.

The probability computation requires the normalised total energy of an empire E_{CN_n}:

$$E_{CN_n} = E_{C_n} - \max_{n=1}^{N_{imp}}\{E_{C_n}\} \tag{3}$$

where E_{C_n} and E_{CN_n} represents the empire's total energy and normalised total energy, respectively. The competition probability can therefore be computed as P_n:

$$P_n = |E_{CN_n}/\max_{n=1}^{N_{imp}}\{E_{C_n}\}| \tag{4}$$

In order to assign the free colony to an empire, construct the following vector:

$$P = (p_1, p_2, \cdots, p_{N_{imp}}) \tag{5}$$

and another evenly distributed vector R of the same size:

$$R = (r_1, r_2, \cdot, r_{N_{imp}}), \quad r_1, r_2, \cdots, r_{N_{imp}} \in U(0,1) \tag{6}$$

then we get the following vector D:

$$D = P - R = (D_1, D_2, \cdots, D_{N_{imp}}) = (p_1 - r_1, p_2 - r_2, \cdots, p_{N_{imp}} - r_{N_{imp}}) \tag{7}$$

Based on the above computations, the free colony will be assigned to the empire with the biggest D value. The competition process is shown in figure ?.

2.6 Assimilation

There exist two types of assimilation in the improved CCA, namely, assimilation within an empire and assimilation between an empire and an independent country. The former assimilation process involves on the empire and its colonies and is the same with the classical CCA. The latter assimilation process is described as follows:

- an independent country will move to all empires and the movement process is the same with that of colony to its empire. The movement process is achieved by the crossover and mutation operators. This paper uses the single-point crossover operator and the mutation operator randomly selects a gene to reverse its current value in order to generate a new solution.

- separate new independent country is generated after the movement toward every empire and the best one is used as the new independent country.

The assimilation processes are depicted in figure ?. It can be seen from the figure that empire assimilation only involves the empire and its associated colonies, which is different from the assimilation between an independent country and an empire. The attempts to move to every empires made by an independent country can speed up the algorithm's convergence rate by only selecting the best move; on the other hand, more computational times are required.

2.7 Update

The updating processes in the improved CCA include three types depicted in figure ?:

- Updating process between an empire and its colonies. If the new colony created after the assimilation process has better objective value than that of the empire, it will become the new empire.

- Updating process between an independent country and an empire. If the best independent country has better objective value than that of the weakest empire, it will replace the empire and the replaced empire will become an independent country.

- Updating process between colonies and independent countries. If the best colony after assimilation has better objective value than that of the weakest independent country, it will replace the independent country, which will become a colony.

2.8 Empire removal

The improved CCA uses the same empire removal step as in the classical CCA.

3 Experiments

In this paper, we randomly generated 50 arrays of size 10 with values no greater than 20. All the arrays are sorted non-decreasingly and all the equal numbers are set to 0. Put the 50 arrays into five groups: 1) the first group encompasses arrays $1 \sim 10$; 2) the second group encompasses arrays $1 \sim 20$; 3) the third group encompasses arrays $1 \sim 30$; 4) the first group encompasses arrays $1 \sim 40$; 5) the first group encompasses arrays $1 \sim 50$; These generated instances are solved using the modified CCA (MCCA) and the minimal hitting set algorithm based on dynamic maximum (DMDSE-Tree). For each group, the minimal hitting set and average computation times from five runs are recorded to make further comparisons.

The experiments are run on a Windows XP computer with Intel Pentium G630 2.70 GHz CPU and 2GB memory. All the algorithms are implemented using Visual C++ 6.0. The parameters of MCCA are set as follows: $N_{pop} = 100$, $N_{imp} = 7$, $N_{ind} = 5$, the maximum number of iterations $N_{max} = 100$, the weight is set as $0.2, 0.3, \cdots, 0.8$ and $\alpha = 0.8$. The minimal hitting set and computational time comparisons between MCCA and DMDSE-Tree are given table ? and table ?, respectively.

The experiments are run fives times and the average computation time and minimal hitting sets are obtained. Table ? shows the comparisons between computation time percentage and number of minimal hitting sets.

It can be seen from table ? that MCCA is able to solve 90% of instances with about 70% of computational times when compared to DMDSE-Tree, which shows the superior performance of MCCA in solving minimal hitting set problem. In addition, MCCA shows better robustness in solving problems with various scales.

4 Application in equipment selection

In this section, we report an application of using modified CCA to solve the minimal hitting set problem encountered in a manufacturing company. There are 12 manufacturing units in the module-based production system and are denoted as $A, B, \cdots, L$, respectively. Every manufacturing unit consists of multiple general-purpose machines. The studied company plans to manufacture a new product $P1$, which requires 23 self-made parts. Multiple parts can be produced within the same units due to the small volume and extra processing capacity. The problem is to use minimal number of units to produce the new product in order to reduce the impact on existing production plan. Table ? shows all the required parts and their corresponding unit.

The steps of using modified CCA to solve the minimal hitting set problem to get the minimal number of production units are as follows:

- Assign numbers to each manufacturing unit. In this case, 1 is assigned to A, 2 is assigned to B. Repeat this assignment until L is assigned 12. Create an input file by setting 5 as the number of manufacturing units corresponding to the parts, all units are set to 0.

- Identify the minimal hitting set using the implemented C++ code using parameters $N_{pop} = 100$, $N_{imp} = 7$, $N_{ind} = 5$, $N_{max} = 100$ and weight set as 0.6. Table ? shows the computational results. The minimal hitting sets that has the minimum number of element are: $\{7, 8, 9, 11, 12\}$, $\{1, 7, 9, 11, 12\}$, $\{1, 2, 7, 8, 12\}$, $\{2, 7, 8, 10, 12\}$, $\{2, 7, 8, 11, 12\}$, $\{6, 7, 9, 11, 12\}$, $\{7, 8, 9, 10, 12\}$.

- Convert the hitting set with the minimal number of elements into manufacturing units: $\{G, H, I, K, L\}$, $\{A, G, I, K, L\}$, $\{A, B, G, H, L\}$, $\{B, G, H, J, L\}$, $\{B, G, H, K, L\}$, $\{F, G, I, K, L\}$, $\{G, H, I, J, L\}$.

Using the obtained manufacturing units to produce product $P1$, there is no need to change the production plans for other manufacturing units. It will incur less adjustment fees when product volume changes in the future.

5 Conclusions

The minimal hitting set problem has many real-world applications but is computationally hard to solve. This paper proposed an modified colonial competitive algorithm to solve this combinatorial optimization problem and elaborated the detailed steps. Performance of the proposed algorithm is validated on testing instances and minimal hitting sets are obtained for most instances with reasonable times. Lots of comparisons are made during the computation process in order to get distinct minimal hitting sets, which slows down the solution process. Finally, the minimal hitting set problem is applied to the equipment selection problem in a manufacturing system.

References