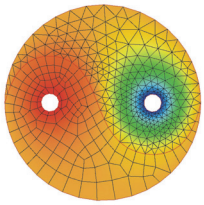
Publishing House
AKAPIT

COMPUTER METHODS IN MATERIALS SCIENCE

Informatyka w Technologii Materiałów

Vol. 9, 2009, No. 2



LIBRARY OF HEURISTIC ALGORITHMS' TEMPLATES

NORBERT SCZYGIOL[1], ANNA WAWSZCZAK[2]

Institute of Computer and Information Sciences
Czestochowa University of Technology
ul. Dąbrowskiego 73, 42-200 Częstochowa
Corresponding Author: norbert.sczygiol@icis.pcz.pl (N. Sczygiol)

Abstract

Heuristic algorithms are used in almost every area of science, including technology, medicine and economics. People engaged in some particular problem often don't have enough knowledge to implement for example a genetic algorithm. In this paper we present a library of heuristic algorithms' templates, which was designed in response to this problem. This library, called MetaHLib, was designed to enable more people to use heuristic algorithms. There is no necessity to know how they works in detail. This article describes fundamental features of the library, components that belong to MetaHLib and a way of using them. Possibilities of using elements of the library in conjunction with external applications were taken into account. Library usage was presented on the example of optimization of input parameters for mesh generator. The mesh generator using elements which are Kalvin's tetradekahedrons was applied for this purpose. This problem was solved by using evolutionary algorithm's template and additional components which belongs to the library discussed in this paper. This optimization enabled us to decrease number of irregular boundary elements.

Key words: metaheuristics, optimization, heuristic algorithms, artificial intelligence, genetic algorithm

1. INTRODUCTION

Dynamic technology's development confront us with new, more difficult problems. We try to take into account more and more factors which have an impact on considered situation, therefore solving problems become more complicated. Disregard of some factors leads to situation, where analysed model differs more and more from reality. On the other hand, taking into account large number of factors causes necessity to use more complicated relations and dependencies between them. It leads to problem's complication and significant rising of computational complexity.

A lot of modern algorithms, which try to give an answer for questions concerning real-world, belong to a class of NP or NP-hard problems. Despite fast computers development, continuous increase in computing power and using parallel computers, these algorithms are still too much time-consuming. Since the number of operations that have to be made rises exponentially, we are able to solve only small problems.

When the size of a problem is so big, that using classical methods is uneconomic or even impossible, it is necessary to use algorithms, which give a solution close to optimal one. Many times it is better to get good, but not optimal solution in reasonable time, then wait a very long time for the best one. Heuristic algorithms deal very well with this kind of problems. Their power consists in the ability of adaptation to concrete problem and taking into account factors specific for particular task, without algorithm complication.

In this paper, we present a library of heuristic algorithms' templates, called MetaHLib. It was im-

 ISSN 1641-8581

plemented in C++. The MetaHLib library provides components that simplify the construction of heuristic algorithms including problem specification.

There are also other similar libraries, which simplify the use of the heuristic algorithms. Some of them, like for example LibGA [3], are designed only for genetic algorithm's construction. Other projects, like HotFrame [4], are big and extended tools, which enable using almost every known heuristic algorithms. A framework called ParadisEO [2] is as big and extended tool as the previous one. It is based on the EOlib library [5], which was dedicated for genetic algorithms design. Thanks to the elasticity and the hybridization mechanisms, its field of use is almost boundless. Z. Michalewicz chose a different conception. He designed GENCOP (GEnetic algorithm for Numerical Optimization of COnstrained Problems) described in [6], as a tool dedicated for solving only one kind of problem, which is finding the global extremum (minimum or maximum) of a function.

One of the groups of issues which require to solve complex problems is numerical modelling of materials processing. Heuristic algorithms can find a lot of applications in this field. One of them, discussed in this paper, is mesh optimisation. Heuristics can be also used always when some set of parameters is to be optimised. Thanks to its generality, they can be really helpful everywhere, not only in materials science.

2. A BRIEF OVERVIEW

The MetaHLib (**MetaH**euristic**Lib**rary) library is a set of templates, which enable fast and comfortable use of heuristic algorithms. This library comprises metaheuristics such as: local search, simulated annealing, tabu search and evolutionary algorithm. Metaheuristics can be defined as sets of rules, on the basis of which algorithms are constructed. The way how generic algorithms works, depends on the problem which is to be solved.

During implementation of the library, we used generic programming, which in C++ language is realised as the mechanism of templates. Besides metaheuristics, MetaHLib provides also classes and class templates, which represent additional elements of heuristic algorithms, such as: neighbour generators, genetic operators, stop criterions, cooling functions, decoders.

The primary design objectivity of the MetaHLib was creating a tool, which will be as easy to use as it is possible. Heuristic algorithms are used in almost every area of science, including technology, medicine and economics. People engaged in some particular problem often don't have enough knowledge to implement for example a genetic algorithm. On the other hand, people engaged in heuristics often do not know enough about the problem which is to be solved. Thanks to the simplification of algorithms, metaheuristics that belongs to the MetaHLib library can be used by people, who have just basic knowledge about an artificial intelligence. Algorithm templates are implemented so as to minimize the amount of code, which has to be written by user.

One of the most important heuristic algorithms' features is a generality. In many cases, it is possible to solve one and the same problem by using two or more different heuristics. There are no strictly specified rules, which tell us how to choose the best algorithm. Sometimes, after program execution, it turns out that the chosen algorithm is ineffective. The MetaHLib library was designed in that way, so as to enable fast change of used algorithm, without redefinition of its components. All of the algorithms require objective function defined in the same way. Most of them require also identical model of neighbour generator and stop criterion.

Sometimes, the algorithm is chosen properly, but one of its components is not selected correctly. Thanks to the solutions used in the library, it is possible to change this component and to find out experimentally, which method is the most efficient. For example, when we are trying to solve some problem using classical genetic algorithm, which works with binary sequences, we can change crossover function from one-point to two-point operator. Only a simply modification of the name of one parameter is required.

3. BASIC ELEMENTS OF HEURISTIC ALGORITHMS

There are three basic concepts that have to be defined for all algorithms mentioned in this paper. They are required during solving all kinds of problems.

- representation – encodes solutions that are used by algorithm,
- target – describes what we want to reach,
- fitness function – is responsible for determining the quality of solution or at least for the comparison of two solutions and arranging them in order.

All algorithms described in this article, are iterative methods. New solutions are generated on a basis

of a solution or a set of solutions from the previous iteration. Component, that is responsible for this operation, is called a neighbour generator or a crossover operator (in case of evolutionary algorithms).

3.1. Representation of solution

Encoded form of a candidate for solution, which is used by algorithm, is called representation of solution. A set of all feasible candidates for solutions, encoded with an established method, is called search space. It is domain of the function to be optimized. The representation of solution depends on the kind of problem that is to be solved.

Components, that belongs to the MetaHLib library, can use any data structure as a representation of solution. There are no restrictions imposed on it, but it is important to use the neighbour generator appropriate for chosen data structure. The MetaHLib library provides classes, which represent solution as bit-strings, real-valued vectors and permutations of integers.

3.2. Evaluation (fitness) function

Evaluation function, also known as fitness or objective function, is a method used to determining solution quality. In most cases, this is a function which transforms a set of candidates for solution into a set of numbers. This transformation assigns value which determines solutions quality to every element of the search space.

The primary rule, which has to be complied with by every fitness function, says that the value of this function for a globally optimal solution has to be better then the value for any other solution. In view of a direct connection between the evaluation function and the problem domain, it is the only one component of described algorithms, which has to be wholly implemented by the user.

3.3. Neighbour generators and crossover operators

A surroundings of the solution, called also a solution neighbourhood, is a small fragment of the search space, a set of solutions, which are situated "close" to the given solution. Meaning of "closeness" is defined by a neighbour generator. This component creates new solutions on the basis of solutions from the previous iteration.

A crossover operator, used in evolutionary algorithms, can be treated as the particular instance of neighbour generator. This function create the new solution on a basis of two (or more) solutions – parents.

The MetaHLib library contains some neighbour generators and crossover operators, that works with standard solution representations that belongs to the library.

3.4. Stop criterion

All algorithms provided by MetaHLib are iterative algorithms. A stop criterion is responsible for an interruption of the iterative process. Usually, it happens after defined number of iterations.

The MetaHLib library provides three classes which represent stop criterions. The first of them interrupts the iterative process after defined total number of iterations. The second criterion stops an optimization process after given number of iterations without an improvement of the best solution's quality. The last one is a combination of two criterions mentioned above.

4. SOLUTIONS DECODERS

Solution decoders enable to use operators dedicated for some type of solution representation while using different type of it. Most often decoders are used to convert different type of solution to a bit-strings representation. Such conversion enables a usage of classical genetic algorithm. Encoded data structures do not represent solution, but give an information how to build this solution. In some cases, it allows to eliminate the problem of constraints.

Standard decoders, which are part of the MetaHLib library, enable to encode a real value (or a real-valued vector) from the defined range to a binary string representation. A range of variability of the encoded variable is divided into subranges. In the next step, the number of a subrange, which contains the encoding value, is tranformed into binary representation. The number of subranges depends on a bit-string length and is one of the factors affecting the precision of solution.

5. METHODS OF USING COMPONENTS OF LIBRARY

Metaheuristics provided by MetaHLib are implemented as templates of classes which represent algorithms. Every template, which represents heuristic algorithm, provides a constructor. This constructor takes parameters, that represent components required to workings of this algorithm, such as:

neighbour generator, crossover operator and so on. Types of these components are defined by the template parameters.

In order to start an optimization process, an operator() has to be called. Initial solution or set of initial solutions (in case of evolutionary algorithms) has to be passed for this operator as the parameter. It returns the best feasible solution found in the optimization process.

All of algorithms' templates, which belongs to the MetaHLib library, provide a set of four methods, which allow to obtain following information about the optimization process:

- the best feasible solution that was found during the optimization process;
- value of fitness function for the best solution;
- total number of iterations;
- number of iterations without improvement of the best solution quality;

These methods should be used after operator() calling, so after ending of the optimization process.

6. POSSIBILITIES OF USING ELEMENTS OF LIBRARY IN CONJUNCTION WITH EXTERNAL APPLICATIONS

The way of implementation of algorithms' templates belonging to the MetaHLib library enables conjunction of them with external applications. One of the most important areas of usage of heuristic algorithms is shape optimization. Most often, some physical values, such as power or frequency, are used as optimization criterion. There are a lot of engineering packages, which enable a simulation of some physical processes and determining these values for the given parameters. Most of them enable to run the simulation by command line interface (without using graphical interface). These applications can be used in conjunction with heuristic algorithms, in particular with the algorithms provided by the library described in this paper. Fitness function is the only one component strictly required by all of heuristic algorithms. After defining a suitable interface, the engineering packages can act as a fitness function.

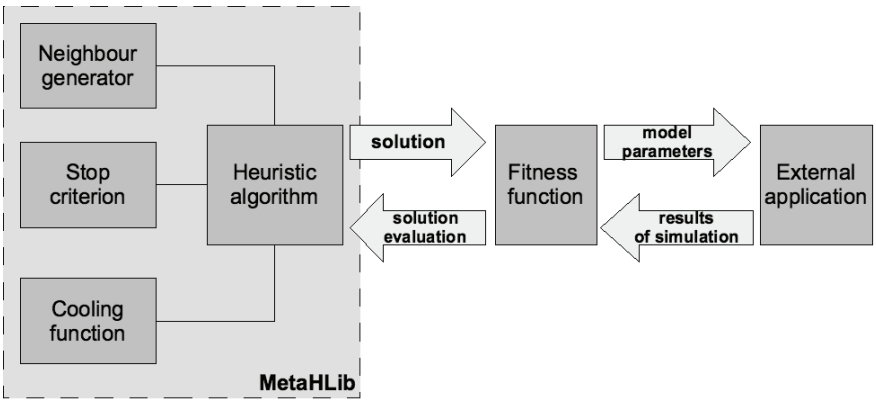


Fig. 1. Using metaheuristics in conjunction with external application.

Parameters which describes model are passed while starting application. They are variables subjected to the optimization or parameters determined on the basis of these variables. After the end of the simulation, the applications returns some results. The value of fitness function is determined on their basis. Conjunction of the engineering packages with the MetaHLib library causes significant reduction in cost of labour and time necessary to implement application. Diagram of the usage of metaheuristics in conjunction with external applications is shown in figure 1.

7. USAGE EXAMPLE

The library usage was presented on the example of optimization of input parameters for a mesh generator. The evolutionary algorithm was used for this purpose.

Finite elements mesh generation is one of the the most important elements of a computer simulation of physical processes. The simulation results depend on it in a significant way. There are a lot of mesh generation techniques. Despite they works in different ways, irregular boundary elements are produces in almost every case. Their shapes are different than the shapes of inner elements. These elements usually cause some problems and affect the precision of calculations.

In this example, we use the mesh generator described in [1]. This generator create mesh of spatial elements, which are Kalvin's tetradekahedrons. Minimization of the irregular boundary elements number is taken as optimization goal. Dimensions of model and size of elements are taken as constant. The parameters subjected to optimization are:

- inner elements orientation,
- coordinates (x,y,z) of reference point, which define offset of the first element from the corner of discretized region.

There are also some con-

straints imposed on coordinates. They result from how the generator works. These constraints are shown in equations (1),(2) and (3).

$$0 \le x \le elemsize \tag{1}$$

$$0 \le y \le \sqrt{3} \cdot elemsize \tag{2}$$

$$0 \le z \le 2\sqrt{2} \cdot elemsize \tag{3}$$

where: *elemsize* – size of element (length of figures' sides that construct tetradekahedrons).

7.1. Representation and fitness function

Solution of this problem consists of one boolean value (orientation of elements) and three real values (offset point coordinates). Bit-string was chosen as the solution representation. It was necessary to use decoders in order to convert a real solution of considered problem to chosen solution representation.

Direct usage of the decoders provided by The MetaHLib library was impossible, because real solution consists of two different value types. For the purposes of analysed problem, a new decoder was created. It converts real solution in the following way:

- first bit of the solution represents elements direction
- the next 48 bits of solution are dedicated for encoding of three offset point coordinates – 16 bits for each coordinate

Decoder FloatVecToBinary, provided by the MetaHLib library, was used to encode real values to bit-string form (and vice versa).

The decoder, which works in the way described above, was used inside the fitness function. Binary vector, that is to be evaluated, is passed to the decoder. The solution is returned as a structure that consists of one boolean value and three real values. After solutions decoding, the members of the received object are passed to mesh generator, which returns the number of irregular boundary elements.

7.2. Genetic operators

Using bit-string representation enabled to use classical genetic operators. There where used classical genetic operators provided by the MetaHLib library:

- OnePoint – operator that realises one point crossover;
- BinaryMutation – operator that realises binary mutation operation;

An operation of selection was made by using standard operator, that belongs to the MetaHLib library. The operator, which realises roulette wheel selection method was chosen for this purpose.

7.3. Process of optimisation and its results

As a stop criterion, there was used a standard criterion provided by the MetaHLib library. It interrupts optimisation process after 30 iterations, which is equivalent to 30 generations. Population consisted of 30 members.

In the considered example, generator had a task of creating mesh for region, whose randomly chosen dimensions are equal (121.0; 111.0; 87.0). The optimisation process was carried out for two different element sizes: 5.0 and 16.0.

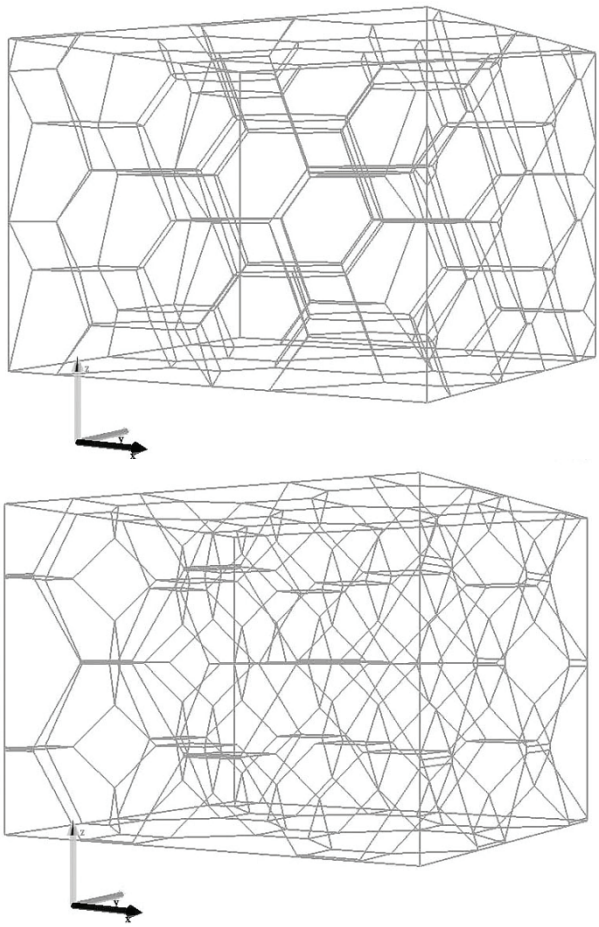


Fig. 2. *a) Mesh before optimization (55 irregular boundary elements), b) Mesh after optimization (46 irregular boundary elements).*

In the first case (element size equals 5.0), mesh generated for the first type of element orientation (defined as 0) and offset point coordinates established as $x = 0.0$, $y = 0.0$, $z = 0.0$, consists of 471 irregular boundary elements. Optimisation process allow to decrease this number to 441. It is about

13% improvement. This result was reached for the first type of element orientation and offset point coordinates equal $x = 0.0$, $y = 0.0$, $z = 10.6$.

In the second case (element size equals 16.0), mesh generated for the first type of element orientation (defined as 0) and offset point coordinates established as $x = 0.0$, $y = 0.0$, $z = 0.0$, consists of 55 irregular boundary elements (figure 2a). Optimization allow to decrease this number by 16.4% (46 irregular boundary elements). This result was reached for the second type of element orientation and offset point coordinates equal $x = 6.41$, $y = 1.30$, $z = 0.0$ (figure 2b).

8. SUMMARY

In this article, we presented the design and application of the library of heuristic algorithms' templates called MetaHLib. This library was implemented in C++ using the mechanism of templates. The MetaHLib library consist of heuristic algorithms' templates and additional components that are helpful while using them.

The primary design objectivity was making it possible to use this kind of algorithms by people who do not have specialistic knowledge about them. The example, which was presented in this paper, shows how to use this library to solve real engineering problems. Thanks to applying standard components provided by MetaHLib, the amount of code that has to be written was reduced to the essential minimum. Since presented library is general and can be used to solve problems of different kinds, it was impossible to avoid necessity for fittness function implementation. It is essential every single time when described algorithms are used. This component is very strictly connected with the problem domain. Using standard fitness function defined inside the library would limit set of problems that can be solved to one small group of problems.

The example presented in this library shows great efficiency of heuristic algorithms. At the same time, thanks to libraries such as MetaHLib, implementation of programs using heuristics is fast and easy. Even people who do not know precisely the rules of their working, can use it. Conjunction of this two elements: efficiency and simplicity of use makes heuristic algorithms great and powerful tools, which can be used while solving a lot of problems from different areas of science and engineering.

REFERENCES

1. Bieda, R., Projekt i wykonanie generatora strukturalnej siatki elementów przestrzennych, MSc thesis, Czestochowa University of Technology, 2008 (in Polish).
2. Cahon, S., Talbi, S., Melab, N., ParadisEO: a Framework for Parallel and Distributed Biologically Inspired Heuristics, Proc. Int. Parallel and Distributed Processing Symposium, 2003.
3. Corcoran, A.L., Wainwright, R.L., Using LibGA to Develop Genetic Alghorithms for Solving Combinatorial Optimization Problems, CRC Press, 1995.
4. Fink, A., Voss, S., HotFrame: A Heuristic Optimization Framework, Optimization Software Class Libraries, Kluwer, Boston, 2002.
5. Keijzer, M., Morelo, J.J., Romero, G., Schoenauer, M., Evolving Objects: A General Purpose Evolutionary Computation Library, Proc. 5th Int. Conf. on Artificial Evolution, France, 2001.
6. Michalewicz, Z., Fogel, D. B., How to Solve It: Modern Heuristics, Springer-Verlag, Berlin, Heidelberg, 2000.

BIBLIOTEKA SZABLONÓW ALGORYTMÓW HEURYSTYCZNYCH

Streszczenie

Algorytmy heurystyczne są wykorzystywane praktycznie w każdej dziedzinie nauki, między innymi w technice, medycynie, ekonomii. Często jednak osoby zajmujące się konkretnym problemem nie posiadają wiedzy wystarczającej do samodzielnego zaimplementowania na przykład algorytmu genetycznego. W niniejszej pracy zaprezentowana została biblioteka szablonów algorytmów heurystycznych MetaHLib będąca odpowiedzią na ten problem. Biblioteka zaprojektowana została tak, aby umożliwić korzystanie z algorytmów heurystycznych bez konieczności posiadania szczegółowej wiedzy na temat ich działania. Artykuł opisuje podstawowe cechy biblioteki, wchodzące w jej skład komponenty oraz sposób ich wykorzystania. Uwzględniono również możliwość wykorzystania elementów biblioteki w połączeniu z aplikacjami zewnętrznymi. Sposób wykorzystania biblioteki został zaprezentowany na przykładzie optymalizacji parametrów wejściowych dla generatora siatek elementów przestrzennych. Do tego celu zastosowany został generator wykorzystujący elementy będące czternastościanami Kelvina. Do rozwiązania tego problemu zastosowany został wzorzec algorytmu ewolucyjnego oraz dodatkowe komponenty należące do omawianej biblioteki. Przeprowadzona optymalizacja pozwoliła na zmniejszenie liczby nieregularnych elementów brzegowych.

Submitted: October 14, 2008
Submitted in a revised form: November 15, 2008
Accepted: December 15, 2008