INFORMATICA, 2002, Vol. 13, No. 4, 393–404

One Application of the Parallelization Tool of Master–Slave Algorithms *

Milda BARAVYKAITĖ, Rimantas BELEVIČIUS, Raimondas ČIEGIS
Vilnius Gediminas Technical University
Saulėtekio 11, LT-2040 Vilnius, Lithuania
e-mail: {milda.baravykaite, rb, rc}@fm.vtu.lt

Received: May 2002

Abstract. The paper analyzes the performance of parallel global optimization algorithm, which is used to optimize grillage-type foundations. The parallel algorithm is obtained by using the automatic parallelization tool. We describe briefly the layer structure of the Master–Slave Template library and present a detailed mathematical formulation of the application problem. Experiments are done on the homogeneous computer cluster of 7 IBM machines RS6000. The results of experiments are presented.

Key words: mathematical modeling, parallel algorithms, software tools, master–slave algorithm, optimization, grillage-type foundations.

1. Introduction

The parallel programming tools and packages are evolving rapidly. However the complexity of parallel thinking does not allow to implement many algorithms for the end user. Unlike conventional programming, the template programming does not require from the user to know the parallel programming tools to create parallel programs. Instead the user has to recognize his algorithm type and to choose the right template where some code pieces are inserted into predefined places (see, Freeman and Phillips, 1991; Šablinskas and Baravykaitė, 2002; Pflaum, 2001).

Master–Slave (MS) template is used to parallelize algorithms for solving global optimization problems, numerical integration of multidimensional integrals, graph problems. We have developed a special tool, which enables the user to parallelize automatically MS-type algorithms. This tool was used for adaptive multidimensional numerical integration on distributed memory parallel computers (see, Čiegis *et al.*, 1998; 1999). It is interesting to note, that numerical integration and global optimization algorithms have many common features, when their parallelization is considered. The costs of each subproblem are distributed unevenly and a pool of subproblems is generated dynamically during the realization of the algorithm.

*This work has been supported in part by Lithuanian State Science and Studies Foundation grant A-537 and Eureka grant OPTPAPER 2623, E-2002.02.27

The acceptance of numerical algorithms and software tools can benefit from demonstrating that they work robustly and safely over a wide range of practically relevant problems (see, Lang, 2001). In this paper we will use the developed parallelization tool to solve a real-life application in civil engineering. Our goal is to optimize the grillage-type foundations in the construction of buildings. The grillage-type foundations are the most popular and effective scheme of foundations, especially in case of weak grounds. The optimal scheme of grillage should posses the minimum possible number of piles. Theoretically, reactive forces in all piles should approach the limit magnitudes of reactions for those piles. This goal can be achieved by choosing appropriate pile positions. Engineering tests algorithms are useful only in case of simple geometries and loadings. Otherwise computational optimization procedures are evident necessities. Such analysis was done by Belevičius and Valentinavičius (2000), Belevičius *et al.* (2002). The global minimum solution is important in applications, but the proposed techniques can not guarantee that the algorithm will not stop at some local minimum solution. In this paper we apply modifications of these algorithms, which increase the probability to find a better solution.

We note that many problems of engineering, physics, technology, economics are reduced to global minimization of multimodal functions. Such problems are difficult in the sense of the algorithmic complexity theory and global optimization algorithms are computationally very intensive. The global optimization algorithms are reviewed in Törn and Žilinskas (1989), Horst *et al.* (2000). The black box global optimization algorithms and their parallelization is investigated by Madsen and Žilinskas (2000), Žilinskas (2001).

The paper is organized as follows. In Section 2, we describe our tool for automatic parallelization of Master–Slave type algorithms. The problem of optimization of grillage-type foundations is formulated in Section 3. Here we also describe briefly optimization technique and serial algorithms. In Section 4 we use the tool to find optimal positions of piles. Efficiency of the automatically obtained parallel algorithm is investigated and results of computational experiments are presented. Some final conclusions are given in Section 5.

2. Parallelization Tool

In this section we will describe very briefly our tool for automatic parallelization of Master–Slave type algorithms (see, Šablinskas and Baravykaitė, 2002 for more details).

The template programming and other semi-automatic tools for parallel programming hide the fact that the user is doing a parallel programming at all. To achieve this, we organize the Master–Slave Template library into a set of independent layers. The Fig. 1 describes the layer structure of a program. Each layer consists of a set of procedures which either implement the layer logic, or serves as an interface that do the transformation of names and/or parameters.

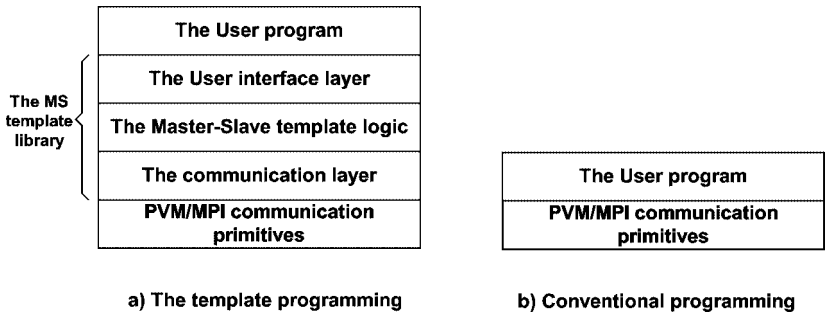

Fig. 1. The layer structure of a program.

We require that the user code comprise of several parts:

1. The data initialization part.
2. The computational process initialization part.
3. The computation loop which uses the following procedures:
   (a) take a non-completed task t_i from the task pool,
   (b) process the task t_i and return the result r_i,
   (c) add up the result r_i to the totals.
4. The result output part.

The user's main program in C will look like:

```
int main()
{
 struct t_init_data id;
 struct t_data      t;
 struct t_result    res;

 M_prepare_job_pool(&id);
  if (! S_initialize(id)) return 0;
  while ( M_take_piece_from_pool(&t))
  {
    S_compute(t, &res);
    M_add_to_result(res);
  }
  M_print_result();
  return 1;
}
```

Testing. After the user adjusts the sequential code to conform the requirements, the testing is made by compiling and running the program in the sequential mode. In case of success, the user simply switches to another directory and compiles/runs the parallel version of the same program.

The purpose of the user interface layer is to transparently connect the user layer procedures to the Master–Slave Template library.

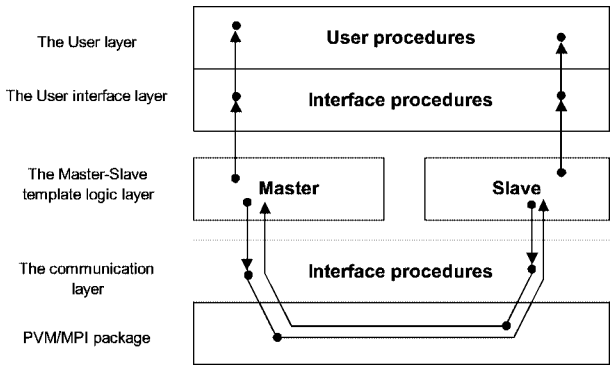

Fig. 2. The Master–Slave template logic layer operations.

The main goals of the MS layer is to create the master and slave processes on the parallel machine, to organize the work flow of the MS algorithm and to exchange the data among the processes. This layer consists of two procedures: `Master`, `Slave`. These procedures invoke the user interface layer procedures to perform the logic of the program and the communication layer procedures to pack and exchange the data (see Fig. 2).

The communication layer is an interface layer, its goal is to hide the communication primitives of the MPI or PVM packages. We define an interface for each MPI or PVM procedure which may have been used in conventional programming. The communication layer procedure list includes data packing, unpacking, message probing and the parallel program finalizing. Each procedure invokes either the PVM or MPI communication primitives depending on the active global setting.

The Master–Slave parallelization tool written in C is freely distributed at `http://perkunas.vtu.lt/Milda/MST3.html`.

3. Problem Formulation

Grillage consists of separate beams, which may be supported by piles, or may reside on other beams, or a mixed scheme may be the case. The optimal scheme of grillage should possess, depending on given carrying capacities of piles, the minimum possible number of piles. Theoretically, reactive forces in all piles should approach the limit magnitudes of reactions for those piles. Limit pile reactions may differ from beam to beam. Practically, this is difficult to achieve if designer due to some considerations introduces into the grillage scheme the so-called ‘immovable supports’ that have to retain their positions and do not participate in optimization process. These goals can be achieved by choosing appropriate pile positions.

We choose the optimization of a single beam as a basis for the whole optimization process of grillage. All grillage is divided into separate beams, “the upper beams” resting on other “the lower beams”. First, all beams are analyzed and optimized separately. Joints and intersections of the upper and lower beams are idealized as immovable supports for

upper beams and concentrated loads for lower ones. Since the obtained fictitious reactions/concentrated loads depend on pile positions, all calculations are iterated in order to level with proper accuracy forces at joints. The problem has to be solved in static and in linear stage. The problem size is up to 100 supports in separate beam and up to 150 beams in grillage, till 15 000 design parameters in total.

3.1. *Mathematical Model*

The optimization problem is stated as follows (see, also Belevičius *et al.*, 2002)

$$\min_{\text{s.t. } \mathbf{x}\in D} P(\mathbf{x}), \tag{1}$$

where P is the objective function, D is the feasible shape of structure, which is defined by the type of certain supports, the given number and layout of different cross-sections as well as different materials in the structure.

P is defined by the maximum difference between vertical reactive force at a support and allowable reaction for this support, thus allowing us to achieve different reactions at supports on different beams, or even at particular supports on the same beam:

$$P(\mathbf{x}) = \max_{1\leqslant i\leqslant N_s} |R_i - f_i\, R_{allowable}|\,,$$

here N_s denotes the number of supports, $R_{allowable}$ is allowable reaction, f_i are factors to this reaction and R_i are reactive forces in each support.

Further, the minimum-maximum problem is converted to a pure minimum problem with constraints by treating P_{max} as unknown subject to constraints that P_{max} limits the magnitudes of P everywhere in the structure and for all load cases when design changes Δx_i are performed, i.e.,:

$$P(\mathbf{x}) + \sum_{i=1}^{N_s} \frac{\partial P(\mathbf{x})}{\partial x_i}\Delta x_i - P_{max} \leqslant 0, \tag{2}$$

for the total structural space x.

For computational reasons a beam length constraint is also included into formulation of the problem:

$$L(\mathbf{x}) + \sum_{i=1}^{N_s} \frac{\partial L(\mathbf{x})}{\partial x_i}\Delta x_i - L_0 \leqslant 0\,, \tag{3}$$

where L_0 is the initial length of the model.

Several possibilities exist in the choice of design parameters x_i on which the structure shape depends. Our choice is to use the most evident from the engineering point of view parameters: nodal co-ordinates of all (or a chosen set of) supports.

The problem is strongly non-linear, thus it is solved iteratively. In each iteration the current shape is changed to a better neighbouring shape. The solution requires three steps:

- finite element analysis,
- sensitivity analysis with regard to design parameters,
- optimal re-design using linear programming.

3.2. *Finite Element Analysis*

Simple two-node beam element with 4 d.o.f.'s has been implemented in analysis (see, Cook *et al.*, 1989). Nodal d.o.f.'s of element are:

$$\mathbf{u} = \left\{w_i, \theta_i, w_j, \theta_j\right\}^T,$$

w_k and θ_k, $k = i, j$ being deflection and rotation, positive counter-clockwise, respectively. The interpolation functions may be found in most finite element books, see for example (Cook *et al.*, 1989).

After the nodal displacements are obtained, the reactive forces are available according to:

$$R_i = \sum_j K_{ij}\, u_j^a\,,$$

where $\mathbf{K}$ is the stiffness matrix of the finite element ensamble.

3.3. *Sensitivity Analysis*

As seen from (2), the sensitivity (i.e., derivatives with regard to nodal co-ordinates) of reactive forces is the must for optimization:

$$\frac{\partial \mathbf{R}}{\partial x_k} = \frac{\partial \mathbf{K}^a}{\partial x_k}\mathbf{u}^a + \mathbf{K}^a \frac{\partial \mathbf{u}^a}{\partial x_k},$$

where superscript a is standing for ensemble. The derivatives of nodal displacements are obtained by solution of the following equation

$$\mathbf{K}^a\, \frac{\partial \mathbf{u}^a}{\partial x_k} = \overline{\mathbf{P}}^a$$

with pseudo-load vector

$$\overline{\mathbf{P}}^a = \frac{\partial \mathbf{P}^a}{\partial x_k} - \frac{\partial \mathbf{K}^a}{\partial x_k}\mathbf{u}^a,$$

here P^a is the loading vector for the whole structure.

The procedure for derivative of element stiffness matrix from which matrix of ensemble $\frac{\partial \mathbf{K}^a}{\partial x_k}$ is composed is as follows: replace the finite element length L^e with $x_j - x_i$, detect whether k is ith or jth node of an element, and obtain $\frac{\partial \mathbf{K}^e}{\partial x_i}$ or $\frac{\partial \mathbf{K}^e}{\partial x_j}$, respectively. Thus only the element possessing node k renders non-zero stiffness derivatives. Similar procedures are valid for derivatives of forces and reactions.

3.4. *Optimization Algorithm*

Let us shortly describe the optimization procedures, first for a single beam and then for the whole grillage.

Optimization procedure for a single beam. Two absolute limits sets (maximum and minimum) on all design co-ordinates status $\mathbf{T}$ are introduced according to existing design restrictions or other considerations. In any case the design variable cannot exceed these limits. For the first solution step, current design variables status $\mathbf{T} = 0$. The absolute limits may differ from one design variable to another, however the maximum absolute move limits must be non-negative, and the minimum ones non-positive:

$$\mathbf{T}^{\min} \leqslant \mathbf{T} \leqslant \mathbf{T}^{\max}, \quad \mathbf{T}^{\min} \leqslant 0, \quad \mathbf{T}^{\max} \geqslant 0\,. \tag{4}$$

Further, the move limits on the design variables alterations per one iteration are introduced, again maximum and minimum. These move limits may vary from one design variable to another and have to be adjusted to the extent of non-linearity of problem so that *Simplex* predictions on the future behaviour of the structure do not differ considerably from finite element solution. In general, move limits should be gradually shrunk as the design approaches the optimum. The accuracy of the approximation is required to be higher when we get close to the optimum because the gains are small and can be swamped by approximation errors. Adjustment of iteration move limits can be done easily by special programmed procedures. Thus,

$$\Delta\mathbf{T}^{\min} \leqslant \Delta\mathbf{T} \leqslant \Delta\mathbf{T}^{\max}\,. \tag{5}$$

Two sets of intermediate always positive variables $\Delta\mathbf{T}^{+}$ and $\Delta\widetilde{\mathbf{T}}$ are introduced in order to satisfy requirements of *Simplex* procedure. Omitting details (see, Pedersen, 1989) we set

$$\Delta\mathbf{T}^{+} + \Delta\widetilde{\mathbf{T}} = \Delta\mathbf{T}^{\max} - \Delta\mathbf{T}^{\min}\,.$$

Thus we obtained the following problem of mathematical programming:

$$\min_{\text{s.t. } \mathbf{x}\in D} P_{\max}\,.$$

Here D is defined by the following constraints

$$\mathbf{P} + \frac{\partial \mathbf{P}}{\partial \mathbf{T}}\Delta\mathbf{T} - \mathbf{I}P_{\max} \leqslant 0\,,$$

$$\Delta\mathbf{T}^{+} + \Delta\widetilde{\mathbf{T}} = \Delta\mathbf{T}^{\max} - \Delta\mathbf{T}^{\min},$$
$$L + \frac{\partial\mathbf{L}^{e}}{\partial\mathbf{T}}\Delta\mathbf{T} = L_0.$$

Optimization procedure for the grillage. First of all, "the upper" and "lower" beams are distinguished, either automatically or by designer. Optimization of separate beams then has to be embraced by accuracy loop because pile placement scheme of one beam influences reaction distributions in remaining beams. The following algorithm is employed:

Algorithm

Initialization:
Set stiffnesses at fictitious immovable supports of upper beams simulating joints with lower beams and accuracy tolerance.
Set *stop* ← *false*.

while *stop* = *false* **do**
1.1. Optimize the upper beams using defined in the last iteration stiffnesses of fictitious immovable supports.

1.2. Optimize the lower beams in addition to specified loadings taking into account concentrated loads coming from the upper beams.

1.3. **if** stiffnesses of the upper and lower beams at joints match (with specified accuracy) **do**
Set *stop* ← *true*.
end if
end while

1.4. Filtering results to exclude matching supports at joints of beams.

3.5. *Global Optimization*

We use the random search method: a number of points are generated in the feasible region. Then the previous serial algorithm is applied with different initial conditions. Since all problems can be solved independently, the MS parallel algorithm is suited for parallelization of the whole job.

We note that computational costs of each subproblem can be very different, thus a priori distribution of these subproblems among processors will lead to a non-efficient parallel algorithm. The parallel MS algorithm distributes jobs dynamically and good efficiency can be obtained in many cases. Thus optimization of grillages is a good test for our MS parallel tool.

This method of global optimization is very simple, but we can not guarantee that the obtained solution is close to the global minimum. The probability of finding the global

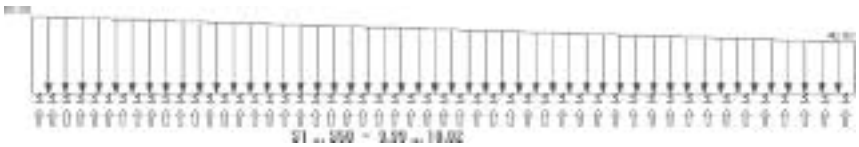

Fig. 3. The obtained optimized pile placement scheme.

minimum approaches one when the number of test points approaches infinity. The efficiency of the algorithm depends on adaptivity of selected points, we attempt to distribute the trial points nonuniformly in the feasible region with greater density in most promising subregions.

4. Numerical Results

The following numerical examples demonstrate the capabilities of the proposed technique. We restrict ourselves with examples of reaction minimization of academic character in order to be able to compare the obtained results with the optimal shapes of grillages known in advance. Our primary goal is to test the efficiency of the parallelization tool, but we also are interested to illustrate the efficiency of the global minimization algorithm, when it is applied for optimization of grillages.

Example
Single beam is loaded with trapezoidal distributed loading of intensity varying from 60 on the leftmost node of beam to the 40 on the rightmost one. All the supports should be generated by program; supports with specified vertical stiffness $k_{spring} = 10^5$ were given for program, while $R_{allowable} = 10.20$. The program generates the computational scheme with 101 node and 100 finite elements, which yields 50 spring-supports. Note, that the reactions in supports are not equal changing from 9.9 to 10.2 because the program stops optimization loops after the allowable reaction is not exceeded in any support.

The task list is obtained by selecting 50 different initial distributions of spring-supports. The computed pile placement scheme is shown in Fig. 3.

Computation Results
In our experiments we have used the homogeneous computer cluster of 7 IBM machines RS6000 running AIX operating system. The computational results are presented in Table 1. Here S_p is the speed-up of the parallel algorithm

$$S_p = \frac{T_1}{T_p},$$

and E_p is the efficiency

$$E_p = \frac{S_p}{p},$$

T_p is the time taken by p processors to solve the problem.

Table 1
Application of the global optimization algorithm

Processors, p	Time, T_p	Speedup, S_p	Efficiency, E_p
1	365.1	1.00	1.000
2	188.8	1.93	0.967
3	1266	2.88	0.961
4	96.5	3.78	0.946
5	78.5	4.65	0.930
6	65.4	5.58	0.930
7	56.3	6.48	0.926

From these results it is clear that the parallel algorithm, which is generated by MS tool, uses efficiently the computer resources and the speedup increases linearly with increasing number of processors.

5. Conclusions

In this paper we have studied the performance of parallel global optimization algorithm, which was applied to optimize grillage-type foundations. This algorithm was generated by the Master–Slave algorithm parallelization tool. It is shown that speed-ups proportional to the number of processors can be achieved on distributed memory systems, like clusters of RS6000 computers.

The global optimization algorithm, which is used in this article, is very simple and convenient for parallel realization. But the efficiency of such algorithm is not good for many problems. Thus it is important to analyze the possibility to use more sophisticated algorithms of global optimization, such as branch and bound algorithms. Black box global optimization methods are the most promising for solving civil engineering problems (see, Madsen and Žilinskas, 2000).

Considering the results of computational example, it should be stated that we did not obtained better solution than in (Belevičius and Valentinavičius, 2000), where the optimization was started from an initial supports placement scheme prepared by a special expert–program. However, the proposed algorithm enables us to explore arbitrary number of initial distributions thus increasing the probability of finding a better solution. The parallel computations are obvious recipe for success in such real-life applications as optimization of the foundation scheme, where the number of design parameters may reach thousands. The simple engineering algorithms used in civil engineering design packages for problem of such size require days of computer time. On the other hand, now widespread computer cluster technologies enable practical use of parallel computations in engineering practice.

References

Baravykaitė, M., and R. Šablinskas (2002). The template programming of parallel algorithms. *Mathematical Modelling and Analysis*, **7**, 11–20.

Belevičius, R., and S. Valentinavičius (2000). Optimization problems for foundations of erections. *Lith. Math. Journal*, **40**, 365–373.

Belevičius, R., S. Valentinavičius and E. Michnevič (2002). Multilevel optimization of grillages. *Journal of Civil Engineering and Management*, **8**, 98–103.

Cook, R.D., D.S. Malkus and M.E. Plesha (1989). *Concepts and Applications of Finite Element Method.* John Wiley & Sons, New York, London, Toronto, Sydney, Tokyo.

Čiegis, R., R. Šablinskas and J.Wasniewski (1998). Numerical integration on distributed-memory parallel system. *Informatica*, **9**(2), 123–140.

Čiegis, R., R. Šablinskas and J.Wasniewski (1999). Hyper-rectangle distribution algorithm for parallel multidimensional numerical integration. In J. Dongarra, E. Luque and T. Margalef (Eds.), *Lecture Notes in Computer Science, Recent Advances in PVM and MPI, 6th European PVM/MPI User's Group Meeting*, Vol. 1697, Barselona, Spain, pp. 275–282.

Freeman, T.L., and C. Phillips (1991). *Parallel Numerical Algorithms.* Prentice Hall, New York, London, Toronto, Sydney, Tokyo, Singapore.

Horst, R., P.M. Pardalos and Nguyen V. Thoai (2000). *Introduction to Global Optimization.* Kluwer Academic Publishers, Dordrecht, Boston, London.

Lang, J. (2001). *Multilevel Solution of Nonlinear Parabolic PDE Systems*. Springer, Berlin, Heidelberg, Tokyo.

Madsen, K., and J. Žilinskas (2000). Testing real and interval methods for global optimization. *Technical Report IMM-Report-2000-05, Department of Mathematical Modelling*, DTU, Denmark.

Pedersen, P. (1989). Design for minimum stress concentration – some practical aspects. In: *Structural Optimization*, Kluwer Academic, 225–232.

Pflaum, C. (2001). Expression templates for partial differential equations. *Comput. Visual. Sci.*, **4**, 1–8.

Törn, A., and A. Žilinskas (1989). Global optimization. *Lecture Notes in Computer Science*, **355**.

Žilinskas, J. (2001). Black box global optimization inspired by interval methods. *Information Technology and Control*, **21**, 53–60.

M. Baravykaitė has graduated from Vilnius Gediminas Technical University (VGTU) Faculty of Fundamental Sciences in 2002. She received Master degree in engineering informatics. Now she is a PhD student in Vilnius Gediminas Technical University. Her research interests include mathematical modeling, parallel computing and development of software tools.

R. Belevičius has graduated from Vilnius Civil Engineering Institute (VISI, now VGTU) Faculty of Civil Engineering. He received the PhD degree from the VISI in 1986 and the degree of Habil. Doctor of Mechanics from VGTU in 1994. Now he is professor and the head of Engineering Mechanics Department of VGTU. His research interests include finite element methods and optimization methods for solution of mechanical problems.

R. Čiegis has graduated from Vilnius University Faculty of Mathematics in 1982. He received the PhD degree from the Institute of Mathematics of Byelorussian Academy of Science in 1985 and the degree of Habil. Doctor of Mathematics from the Institute of Mathematics and Informatics, Vilnius in 1993. He is a professor and the head of Mathematical Modeling Department of Vilnius Gediminas Technical University. His research interests include numerical methods for solving nonlinear PDE, parallel numerical methods and mathematical modeling in nonlinear optics, biophysics, technology.

Lygiagretinimo įrankio algoritmui "Šeimininkas–darbininkai" taikymo pavyzdys

Milda BARAVYKAITĖ, Rimantas BELEVIČIUS, Raimondas ČIEGIS

Nagrinėjama lygiagretaus globaliosios optimizacijos algoritmo, taikyto polinių pamatų sijyno optimizavimui, kokybė. Lygiagretieji skaičiavimai yra viena iš galimų išeičių tokiose realiose aplikacijose kaip ši, kur planavimo kintamųjų kiekis gali siekti tūkstančius. Inžineriniai perrinkimo algoritmai, naudojami kai kuriose konstrukcijų projektavimo programose, tokiems uždaviniams reikalauja 20–50 valandų kompiuterio laiko. Tuo tarpu plintančios kompiuterių klasterių technologijos jau šiandien gali būti naudojamos inžinerinėje praktikoje. Lygiagretusis algoritmas gaunamas naudojant automatinį lygiagretinimo įrankį. Aprašoma sluoksninė "Šeimininko–darbininkų" šablono bibliotekos struktūra. Pateikiamas detalus sijyno optimizavimo matematinis modelis. Skaitiniai eksperimentai, atlikti homogeniniame kompiuterių klasteryje iš 7 IBM RS6000 kompiuterių, demonstruoja proporcingą procesorių skaičiui pagreitėjimą. Šio konkretaus taikymo pavyzdžio atveju nėra gauti geresni inžineriniai rezultatai. Tokia pati išvada teisinga ir skaičiuojant kitais inžineriniais lokaliosios optimizacijos algoritmais. Tačiau siūlomi algoritmai leidžia ištirti bet kokį kiekį pradinių sijyno schemos artinių, tuo būdu didindami tikimybę surasti geresnį sprendinį.