Development of Planning System for Plywood Production Using "Matrix Designer"

Shabaev Anton, Arhipov Ivan, Spirichev Maxim, Urban Alexander, Torozerov Mikhail
Petrozavodsk State University (PetrSU)
Petrozavodsk, Russia
{ashabaev, iarhipov, mspirichev, aurban, mtorozerov}@petrsu.ru

***Abstract*— The paper describes research and development of a software system for plywood mill operation planning. Various tasks related to planning of plywood operation can be successfully solved with the use of specialized software. The paper describes solution of the problem of calculating monthly production plans, which requires the use of advanced mathematical models and optimization algorithms. Therefore, special attention is devoted to the "matrix designer" – a special data structure for improving the efficiency of storing and using data by taking into consideration the block structure of constraints matrix. The paper also includes description of the system. The efficiency of the system had been confirmed on real operating data of plywood mills. The use of "matrix designer" allowed to speed up and simplify modification of the mathematical model while clarifying technical features of the mill.**

Keywords–Production management, Matrix calculations, Optimization problems.

I. INTRODUCTION

Development of software systems for improving production efficiency of industrial enterprises is a part of long-term and diverse research and development activity carried out at PetrSU. The main focus has been on the forestry industry, which is one of the main sectors of Russian economy [3].

Similarity of production processes on different wood processing enterprises allows developing a common platform for production planning for these enterprises [4]. However, using a large amount of locally installed software on geographically distributed enterprises results in additional time and costs needed for the software configuration and maintenance.

To simplify and speed-up these processes, it is useful to develop a software platform, which includes a wide range of functional possibilities for software implementation, maintenance, and solving process optimization tasks [6].

This paper on the example of plywood mills describes end-to-end research and practical development process of production planning systems from description of industrial problem to its mathematical model, then to software implementation, and approbation on real production data. The end result is working system in industrial use.

The aim of the plywood production process consists in processing round wood veneer sheets into plywood sheets of different thickness and grade [1].

In this respect we can consider a plywood sheet as a complex of veneer sheets, which includes exterior and interior layers. Generally, there can be several types of plywood composition for plywood of the same brand. Production may vary in thickness, brand, sorts and grade, be cut or polished by means of additional profiling of edges and surfaces.

Various tasks related to planning of plywood operation can be successfully solved with the use of specialized software. Such software allows increasing overall mill productivity, output of finished products, and efficiency of construction and correction of planning picking of raw materials.

In Section II we briefly review the plywood production process. In Section III we provide in details the mathematical model of the production planning problem, discuss its features and provide an overview of solution method for the related optimization problem. It must be noticed that the solution method involves using multiplicative simplex method. Therefore, in Section IV we describe a special data structure for increasing the efficiency of storing and using data with regard to the structure of submatrices ("matrix designer"). In Section V we briefly describe the user interface of the software system for plywood operation planning. The system allows users to input initial data, calculate optimal production plan for given time period, as well as to generate various reports. Finally, Section VI concludes the paper and discusses further research directions.

II. PROBLEM DEFINITION

This Section contains a brief review of the plywood production process. The aim of the plywood production is to manufacture plywood sheets from in-stock veneer of different grade, thickness etc. [1]. Plywood is made from

thin sheets of wood veneer (plies), which vary in thickness 2.5-4mm. Usually, 3, 5 or 7 plies of dimension 1.2×2.4m are glued together at right angles to each other to form the plywood panel, which can then be refinished. 4-5 quality grades of panels exist.

With hundreds of production orders received, thousands of plies already available (as well as timber logs for making new plies, if needed), there are thousands of feasible ways to obtain the sheets of required quality. Due to various limitations on the equipment operation, the planning must be made very carefully to meet the production and economical targets. This requires the use of specialized software based on mathematical methods and optimization algorithms.

Initial data for the plan includes available portfolio of orders for plywood production (including limits of allowable production output for each type, timing of orders, prices, currency rates etc.), as well as operational parameters (work schedule and standard productivity of equipment).

The computed plan contains the list of orders to be fulfilled during given time period (typically, a month), production output for each order, and way of composing plywood sheets of each type (if there are several options). It also contains the operation schedule of every production unit (press and hulling units), and raw materials required for fulfilling the orders, taking into account probability of appearance of veneer sheets of various grade, depending on basic materials and production technology.

Operation can be optimized either by production output (in cubic meters), or by expected profit (in rubles). Other criteria may be implemented, including more complex ones.

The mathematical model of the production planning problem, its features and an overview of solution method for the related optimization problem are provided in the next Section.

III. Mathematical model of the problem

Let's now formulate the mathematical model of the problem of plywood production planning, and solution algorithm for the related optimization problems, which are necessary for development of the software system.

A. Initial data

We shall call a "peeling technology" (or just "technology") the set of timber species, its grade and thickness of processed veneer (for example, birch 1-th grade 1.5mm, birch 3-th grade 1.2mm) [3].

By plywood composition we shall mean the veneer sheets compound, from which the plywood will be glued. Furthermore, it is possible to produce the same type of plywood in different ways. For example, for plywood type "CP 6 mm" (CP is grade of face veneer, 1-st middle is grade of inner veneer)

- 1-st way: veneer type "CP 1.5 mm" – 2 sheets, veneer type "1-st middle 1.2 mm" – 2 sheets, type "2-nd middle 1.5 mm" – 1 sheet;
- 2-nd way: veneer type "CP 1.2 mm" – 2 sheets, type "1-st middle 1.5 mm" – 2 sheets, type "2-nd middle 1.5 mm" – 1 sheet.

Let us denote

J – the set of plywood types
C – the set of ways of composition of all types of plywood
V – the set of veneer types
T – the set of the presses of the mill
W – the set of timber sorts
Q – the set of peeling technologies

Technological parameters:

$w^p_{i,j} = 1$, if peeling technology $j \in Q$ is used for timber $i \in W$, 0, otherwise
$v^p_{l,j}$ – the average number of veneer sheets $l \in V$, produced from 1 cubic meter of timber, conforming with peeling technology $j \in Q$
$v^c_{l,k}$ – the number of veneer sheets $l \in V$ for plywood production in composition $k \in C$
T_k – the set of presses on which plywood in composition $k \in C$ can be processed
$p^c_{q,k} = 1$, if composition $k \in C$ is related to plywood type $q \in J$, 0 otherwise
l^m_t – the minimal utilized capacity of press $t \in T$ in sheets (calculated as minimal number of loads per shift, multiplied by number of shifts in the planning period, multiplied by the number of sheets in press set of shelves)
l^M_t – the maximum utilized capacity of the press $t \in T$ in sheets (calculated in a similar way)
c^t_t – the capacity of press set of shelves $t \in T$ (number of sheets, fixed parameter)

Economic parameters:

c^w_j – the price of 1 m^3 of timber, conforming with peeling technology $j \in Q$

c^p_k – the price of 1 plywood sheet in composition $k \in C$ (price of plywood of one sort does not depend on the composition, but it is more convenient for developing the model)

Storage parameters:

w^r_i – remains of timber sort $i \in W$ in-stock (in cubic meters)
v^r_l – remains of veneer sort $l \in V$ in-stock (in sheets)

Orders parameters:

o^m_q – minimal possible number of plywood sheets $q \in J$ (according to the stock of orders)

o^M_q – the maximum possible number of plywood sheets $q \in J$ (according to the stock of orders)

Other parameters:

p^m – the minimal profit under the plan (in order to search for a plan, which is optimal in terms of total volume, and at the same time with profit not less than specified)

v^m – the minimal number of plywood sheets under the plan (in order to search for a plan, optimal in terms of income, and at the same time with the volume not less than specified)

v^M – the maximum allowed remains of veneer by the end of the period (in sheets)

f_1 – the penalty for deviation from condition of the balanced plan by volume (in sheets)

f_2 – the penalty for deviation from condition of the balanced plan by profit (in rubles)

The unknown variables:

$x_{k,t}$ – the number of plywood sheets, produced in composition $k \in C$, and processed on press $t \in T_k$

y_j – the volume of timber, which is needed to be peeled in accordance with technology $j \in Q$

z_l – the number of veneer sheets $l \in V$, left after fulfilling the plan, including planned veneer production from timber, as well as planned use of veneer for the plywood production

d^p_t – the deviation of press $t \in T$ utilized capacity from the minimal volume, in sheets

d^v – the deviation from the minimal output, in sheets

d^r – the deviation from constraint on maximal volume of veneer in-stock after fulfilling the plan, in veneer sheets

d^c – the deviation from constraint on minimal profit of the plan, in rubles

The mathematical model

$$\sum_{k=1}^{|C|}\sum_{t=1}^{|T_k|} c^p_k x_{k,t} - \sum_{j=1}^{|Q|} c^w_j y_j - f_1\left(\sum_{t=1}^{|T|} d^p_t + d^v + d^r\right) + f_2 d^c \to \max \quad (1)$$

Constraints:

$$0 \le \sum_{j=1}^{|Q|} w^p_{i,j} y_j \le w^r_i, \quad i \in W \quad (2)$$

$$-\sum_{j=1}^{|Q|} v^p_{l,j} y_j + \sum_{k=1}^{|C|} v^c_{l,k} x_k + z_l = v^r_l, \quad l \in V \quad (3)$$

$$l^m_t \le \sum_{k=1}^{|C|} x_{k,t} + d^p_t \le l^M_t, \quad t \in T \quad (4)$$

$$o^m_q \le \sum_{k=1}^{|C|}\sum_{t=1}^{|T_k|} p^c_{q,k} x_{k,t} \le o^M_q, \quad q \in J \quad (5)$$

$$p^m \le -\sum_{j=1}^{|Q|} c^w_j y_j + \sum_{k=1}^{|C|}\sum_{t=1}^{|T_k|} c^p_k x_{k,t} - d^c \quad (6)$$

$$v^m \le \sum_{k=1}^{|C|}\sum_{t=1}^{|T_k|} x_{k,t} - d^v \quad (7)$$

$$\sum_{l=1}^{|V|} z_l - d^r \le v^M \quad (8)$$

$$x_j, y_k, z_l \ge 0 \quad (9)$$

$$x_{k,t} = g_{k,t} c^t_k, \quad k \in C, t \in T_k, g_{k,t} \in Z \quad (10)$$

$$x_{k,t}, z_l \text{ - integers} \quad (11)$$

This model is significantly improved, compared to the one provided in [3], as it contains variables y, set T_k, and constraints (3), (6)-(8) and (10). This improves the adequacy of the model, but complicates the solution algorithm

Let's denote the original problem (with objective function (1) and constraints (2)-(11)) as problem **P**, and linear programming problem with objective function (1) and constraints (2)-(9) – as problem **P***

Let's denote S – the number of ways of plywood composition with consequent processing (i.e. number of pairs (k,t), where $k \in C$, $t \in T_k$). Then structure of constraints matrix of problem **P*** is presented in Fig. 1:

			Q	C	V	T	3		
cost				M (1xS)		$f_1, ..., f_1, f_2$			
up bound									
W	**0** (Wx1)	≤	MM (WxQ)					≤	M (Wx1)
V			MM (VxQ)	MM (VxS)	E (VxV)			=	MM (Vx1)
T	MM (Tx1)	≤		MM (TxS)		E (TxT)		≤	M (Tx1)
P	MM (Px1)	≤		MM (PxS)				≤	MM (Px1)
1	p^m	≤	M (1xQ)	M (1xS)			-E (3x3)		
1	v^m	≤		M (1xS)					
1					**1** (1xV)			≤	v^M
				≤					
down bound				MM (1xS)					

Fig. 1. Structure of constraints matrix of problem P*

Herein

MM – sparse matrix, given by a list of nonzero elements

M – matrix of free structure

B. Features of the problem P

Problem **P** is a mixed-integer linear programming problem [8] of fairly large dimension (for a problem of such class) – with several hundreds of variables. Among its specific features:

1. Problem **P** is NP-complete, so in order to find the solution in reasonable time (no more than several minutes on standard computers, which are typically used in the mills) a special approximate algorithm must be developed.

2. Taking into consideration that the feasible set of values for problem **P** may be empty, auxiliary variables are necessary – deviations from the upper and lower constraints on each type of plywood, as well as productivity of each press ((4), (6), (7)). In case of absence of feasible values, they allow to provide detailed information on constraints violation to undertake adequate measures by mill management.

3. Several forms of the objective function (1) are possible (by profit, by volume etc.), but this difference as a rule is reflected only in objective function coefficients, and has little effect on the solution algorithm. E.g., if optimization is made by volume, then in objective function (1) the appropriate summand would be $1 \cdot x_{k,t}$, and if by profit, then it would be $c^p_k \cdot x_{k,t}$.

4. Dual estimates [7], obtained during solution of linear programming problem **P***, are useful for analysis of the economic efficiency, structure of orders portfolio and identification of "bottlenecks" in the production process.

C. Solution method for the problem P

For solution of problem **P** the authors offer the following method:

1. By using multiplicative simplex method find vector X – solution to the problem **P*** (this method was chosen because it is the best by speed and precision).

2. Build the initial population for the genetic algorithm, where each specie is a vector obtained by rounding each element of vector X to a number, divisible by the number of sheets in the respective press, either upward or downward in random manner.

3. With the use of genetic algorithm find suboptimal solution to the problem **P**. When using genetic algorithms [5], finding global optimum is not guaranteed, but this method makes possible to find suboptimal solution of problem **P** in a matter of several minutes (when run on PC like Core Duo 2GHz, 4 Gb RAM).

As the solution method for the problem P involves using multiplicative simplex method, in next Section we describe a special data structure for increasing the efficiency of storing and using data with regard to the structure of submatrices ("matrix designer").

IV. Solution of problem P* by using "matrix designer"

Based on experience of IT-park of PetrSU in solving applied optimal planning problems for manufacturing companies [4], it has been noticed that constraints matrix in an applied programming problem as a rule has relatively high dimension and definitive block structure. Therefore, specification of blocks and their positions relative to each other (for example, when adding new variables and constraints) often leads to errors, which are difficult to eliminate. Many problems are not linear, but efficient algorithms, including multiple solution of linear programming problems, may be used for their solution. That rises the requirements to efficiency of the solution algorithms even higher.

Therefore in IT-park of PetrSU we have developed a special data structure in order to increase the efficiency of storing and using data with regard to the structure of submatrices ("matrix designer") which allows to effectively:

- form a constraints matrix using its block structure;
- get data from the matrix;
- modify the constraints matrix in case of modifications to the problem statement;
- group matrix columns into sets and modify them.

Also on the basis of the "matrix designer" the algorithms of effective solution to a number of programming problems have been developed, including linear and nonlinear programming problems, large-scale problems, multi-criteria problems with combined criteria. Methods of linear, convex, dynamic, discrete programming have been implemented, as well as decomposition schemes. This makes possible to effectively find solution to a wide range of programming problems of enterprise planning and management.

From the standpoint of implementation, "matrix designer" is a set of classes and interfaces. Their description is presented below.

A. Main blocks and methods of their use

When developing the matrix designer, the following main types of blocks (submatrices) have been distinguished:

- *MatrixSame* – a matrix of identical elements. The scalar product of i-th column of such matrix on vector v is equal to $\sum_{k=1}^{m} a_{k,i} v_k = p \sum_{k=1}^{m} v_k$.
- *MatrixDiagonalSame* – a diagonal matrix with identical values on the main diagonal. The scalar product of i-th column of such matrix on vector v is equal to $\sum_{k=1}^{m} a_{k,i} v_k = p v_i$.

- *MatrixSimple* – a matrix defined by full enumeration of all its elements.
- *MatrixDiagonalSimple* – a diagonal matrix, with given values of the main diagonal. The scalar product of i-th column of such matrix on vector v is equal to $\sum_{k=1}^{m} a_{k,i} v_k = p_i v_i$.
- *MatrixModular* – a sparse matrix with scarce non-zero elements.

Classes, describing submatrices of each type, implement interface (set of methods) *IMatrixGroupItem*, which includes the following methods:

- *GetScalar* – the function returning the scalar product of the matrix column on any vector $v = (v_1, v_2, \ldots, v_m)$;
- *GetValues* – the function returning all values of the specific matrix column;
- *GetValue* – the function returning the value of the matrix element;
- *Width* – the function returning the width of the matrix (number of columns);
- *Height* – the function returning the height of the matrix (number of rows).

If implementation of submatrixes of other structure is needed for whatever reason, e.g., unimodular, tridiagonal etc, it will suffice to implement the determined functions for the interface *IMatrixGroupItem*.

For construction of compound matrices (consisting of several submatrices) class *MatrixGroup* is used. It implements the same interface, which makes it possible to build matrices recursively.

When calculating the scalar product and taking the matrix values, corresponding functions are called for corresponding submatrices taking into consideration the offset.

For adding a new submatrix in *MatrixGroup* class, function *AddMatrix* has been implemented, to which added matrix and its offset relative to the upper left corner of the matrix is uploaded. In this case the empty matrix cells are considered equal to zero. This procedure is described in further details later.

B. Constructing a matrix from blocks

Construction of constraints matrix from blocks can be simplified in the following manner: the matrix is divided by a rectangular grid in such a way that each cell is either fully occupied by one of the submatrices, or by zeroes (“empty”).

Let us clarify the definition of this grid. For this purpose let us introduce the concept of the *block of rows* as a set of consequent rows and the *block of columns* – set of consequent columns. We will call the *cell* a submatrix, formed by the given block of rows and the block of columns (Fig. 2). Therefore each submatrix of the constraints matrix fills one or multiple adjacent cells vertically and/or horizontally.

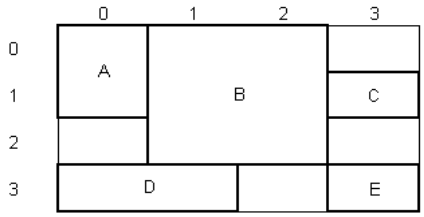

Fig. 2. Example of the block structure of a constraints matrix

Considering that with modification of mathematical model, number of the rows and the columns of the submatrices change, it is significantly more convenient to set size and offset of the submatrices against the given grid.

Let us call the *height of submatrix A’* the number of its rows, and the *width* – the number of columns. *Relative height* is the number of filled cells vertically, and *relative width* – horizontally. *Vertical offset* is the number of the vertical block from the left upper corner of the matrix, *horizontal offset* – the number of the horizontal block. *Absolute offset* (hereafter *offset*) – the number of the row and the column of the upper left element of the submatrix in the constraints matrix.

Therefore, for constructing a matrix out of blocks we need to solve the problem of calculating the absolute offset of submatrices in the matrix according to the given relative offsets.

Given data

Assume that a set of n submatrices is given. Their known parameters are:

h_i, w_i – the height and width respectively
h'_i, w'_i – the relative height and width respectively
t'_i, l'_i – the relative offset vertically and horizontally respectively
W – the width of the main matrix
H – the height of the main matrix
In the context of the problem, h_i, w_i, h'_i, $w'_i \in \{1,2,\ldots\}$ and t'_i, $l'_i \in \{0,1, \ldots\}$.

The unknown

Let us define $y_j \geq 0$, $j = 1,\ldots,H$ and $x_j \geq 0$, $j = 1,\ldots,W$ – as an absolute offset of vertical and horizontal blocks vertically and horizontally respectively.

It is required to find such integers y_j and x_j, for which the following conditions are satisfied:

$$y_{t'_i+h'_i} - y_{t'_i} = h_i,\ x_{l'_i+w'_i} - x_{w'_i} = w_i,\ \forall i = 1..n.$$

$y_1 = 0$, $x_1 = 0$.

Let's note that y_j and x_j are independent of each other.

Then the problem can be split into two identical subproblems, which can be solved separately. Below we present solution to the problem of finding x_j, and the problem of finding y_i is solved in the same manner.

The given set of n submatrices matches the set of n intervals. Their known parameters are:

w_i – width;
w'_i – relative width;
l'_i – relative horizontal offset.

In the context of the problem, $w_i, w'_I \in \{1, 2, \ldots\}$ and $l'_i \in \{0, 1, \ldots\}$.

It is required to find such non-negative integers x_j, $j = 1,\ldots,W$, for which the following conditions are fulfilled:

$$x_{l'_i + w'_i} - x_{w'_i} = w_i,\ \forall i = 1..n. \quad x_1 = 0, \quad x_j \geq 0,\ \forall j = 1..W \quad (12)$$

The 1st approach. Reduction to a linear programming problem

This problem can be reduced to linear programming problem **Q** with constraints (12) and objective function

$$\sum_{j=1}^{W} j x_j \to \min \quad (13)$$

Let us prove that if problem **Q** has a solution, then x_j will take non-negative integer values. For this reason let us remind that $\{\alpha\}$ – fractional part of a number α.

Theorem 1: If $x = (x_1,\ldots,x_W)$ – is the solution of the problem **Q**, then x_j – are non-negative integer numbers.

Proof.

Let x_p and x_q be bound by constraint (12): $x_p - x_q = w_i$

As w_i is a non-negative integral number, then $\{w_i\} = 0$, $x_q = x_p - w_i$

$\{x_q\} = \{x_p - w_i\} = \{\{x_p\} - \{w_i\}\} = \{x_p\}$

Proof for the case $x_q - x_p = w_i$ – is the same.

So, if x_p and x_q are bound by constraint (12), then $\{x_p\} = \{x_q\}$.

Assume there is such p, that $\{x_p\} = d > 0$. Let A be the set of all such p. Let's extend the set A in the following manner: if $i \in A$, x_i and x_j are bound by constraint (12) $(i \neq j)$, then add j to A.

All of x_p in set A are bound by constraint (12) and the previous statement implies that $\forall j \in A \Rightarrow \{x_j\} = d > 0$.

It means that there is x' – another solution of problem **Q**:

$$x'_j = \begin{cases} x_j - d, & for\ j \in A, \\ x_j, & otherwise. \end{cases}$$

In this process all constraints (12) will be satisfied, and the objective function (13) value will be $|A|\ d > 0$ less, i.e. x – is not a solution to problem **Q**, which contradicts the conditions of the theorem.

Therefore the initial assumption is wrong, which was to be proven. □

This approach to finding offsets for submatrices is implemented in class *Matrix Layout Simplex*. Let's note that problem **Q** does not have a solution only in the case, when location of submatrices relative to each other is given incorrectly (one of the cells belongs to more than one submatrix). This allows to find incorrectly constructed matrices prior to starting the simplex-method and thus save the time on error elimination.

Let us note that using simplex-method for solving problem **Q** takes significant memory and time requirements, which is not always worth. In some cases a faster algorithm can be used as follows.

The 2nd approach. "Greedy" algorithm of finding offsets of submatrices.

Let us denote $v_{ij} = x_j - x_i$, $i,j \in 1, \ldots, W$. We proceed as follows:

Step 0. Let $D = \{(i, j): x_i, x_j$ – are bound with constraint (12)$\}$.

Step 1. While there is $k \in 1\ldots,W,$ such that x_i, x_k – are bound by constraint (12), and x_k, x_j – are bound by constraint (12), then add pair (i, j) to the set D

Step 2. If there is a pair $(i, i) \notin D$, $i \in 1\ldots,W$, then denote $v_{ii} = 0$, and proceed to Step 1.

Then $x_j = v_{jj}$. If solution to the problem exists, then the constraint (12) will be met. But this method does not guarantee that in general case $v_{jj} \geq 0$.

Theorem 2: If $w'_i = 1,\ i = 1,\ldots,n,$ then $x_j \geq 0$.

Proof.

The conditions imply that variables are bound by constraints (12). Then under conditions of the theorem:

$x_{j+1} - x_j = w_j,\ j \in B \subset \{1, \ldots, n - 1\}.$ (14)

Further, as all constraints are fulfilled, then (14),

$$v_{1,j+1} = \begin{cases} v_{1j} + w'_j, & if\ j \in B, \\ v_{1j}, & otherwise. \end{cases}$$

As $v_{11} = x_1 = 0$ and $w'_j \geq 0$, then $x_j = v_{1j} \geq 0$. □

This approach to finding offsets for submatrices is implemented in class *Matrix Layout BFS*. It can be used, if each submatrix has relative height and width equal to 1, i.e. fills only one cell.

Let's estimate the complexity of the algorithm. In total W^2 pairs are added. For each pair on Step 1 it is needed to test all constraints binding it, i.e. $O(W)$. Then the complexity of the algorithm is $O(W^3)$. If condition of Theorem 2 is met, then the number of constraints will not exceed 2, which means that in this case the complexity is $O(W^2)$, while simplex-method has complexity $O(W^3)$ at best.

Conclusion: in some cases the "greedy" algorithm cannot be used, and the problem can be solved only using simplex-method. At the same time, the "greedy" algorithm works significantly faster, so both classes are feasible.

C. Reducing linear programming problem to canonical form

It should be noted that in classes *MatrixLayoutSimplex* and *MatrixLayoutBFS* reduction of a linear programming problem, specified in arbitrary way, to canonical form is implemented automatically. It is very convenient, because a user can define a problem in the most convenient way, as during the reduction to canonical form the physical meaning of variables is often distorted.

The "matrix designer" has been practically implemented and integrated with the software system for plywood operation planning, briefly described in next Section. The main modules of the "matrix designer" are shown on Fig.3

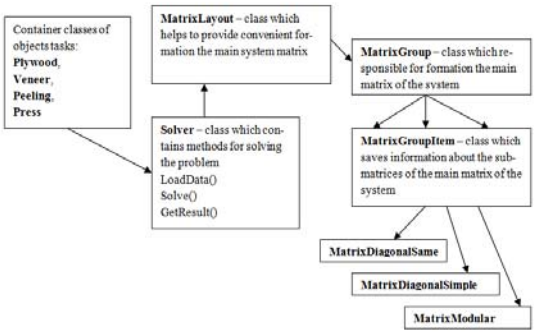

Fig. 3. The main modules of the "matrix designer"

V. User Interface

The software system for plywood operation planning allows users to input initial data, calculate optimal production plan for given time period, as well as to generate various reports (tables and diagrams). These functions are accessible from the menu. Access to certain menu items can be restricted for certain users and user roles. For example, some users can be allowed only to enter data, but not to calculate plans, etc.

The software system was implemented using a specialized versatile library [6] developed in IT-park of PetrSU based on 25 years of experience in research and development of software systems for customers in various branches of industry [2]. The library includes a large number of closely integrated components and allows to unify and speed up the software development processes, to simplify the description of data models and to reduce the number of errors. All components of the library are integrated into MS Visual Studio.NET environment and can therefore be freely and uniformly used together with standard methods and components of MS Visual Studio.

Initial data for planning is entered and stored in tables. There is a special tool box for adding, editing, deleting, sorting, copying and finding elements. Also visibility of fields can be configured.

Total of 6 tables are implemented: “Timber” (including statistical expected veneer production from peeling 1 m^3 of timber for each thickness parameter), “Veneer”, “Veneer thickness”, “Plywood” (including reference table of plywood composition), “Presses” (including a list of allowable for plywood production) and “Clients”. New tables can be added, if needed.

The user interface for input of initial data and calculation of plans is given on Fig.4.

The calculation having been completed, for each type of plywood the specified limits and computed volume are displayed.

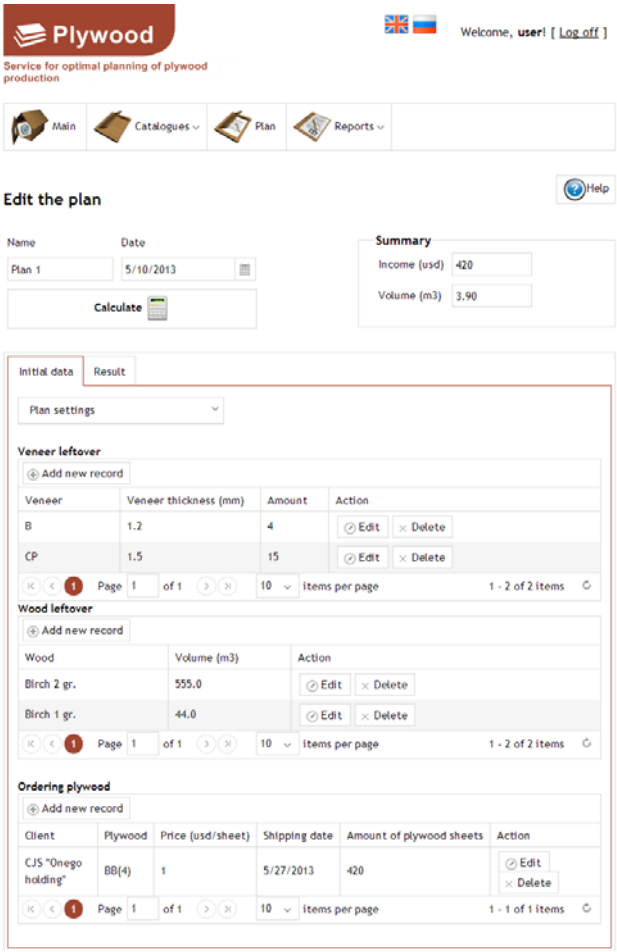

Fig. 4. The user form of creating/editing the plan

Calculated composition plans (e.g., per month) can be stored in a database for later use.

The implemented reports include information on timber balance, plywood production, efficiency and load of the presses, economic indicators of sawmill performance. New reports can be created according to customer demand.

VI. Conclusion

The article describes work on development and practical implementation of software system for optimal planning of plywood production.

The system has been tested on the actual data of plywood mill “Bumex Ltd.” In Karelia (Russia). The calculations show that increase in profit may be up to 1.3% per month.

While clarifying with the Customer technical features of the mill, major modifications of the mathematical model have been made several times, but with the help of the "matrix designer" they has been done quickly and correctly. According to expert estimate, the use of the "matrix designer" allowed to save more than 50% of time on modifications of the mathematical model and related changes to the source code of the System.

A possible direction of the system improvement is consideration of additional constraints, including evenness of press loads, using plywood composition not listed in reference tables, etc. All the components required for that are available, which allows to easily add new functions to the system in order to solve the above mentioned and other optimization problems facing the mills and the industry as whole.

Acknowledgment

The article was published with financial support from the Strategic Development Program of Petrozavodsk State University.

References

[1] Veselov A.A., Galyk L.G., Doronin Y.G. and others. *Reference book of plywood production*, edited by PhD Kachalin N.V. M/: Forest industry, 1984. -432 p. (in Russian)

[2] Voronin A.V., Kuznetsov V.A., Shabaev A.I., Arkhipov I.V. Software system for sawmill operation planning. SPb.: *Proceedings of 12th conference of Open Innovations Association FRUCT*, 2012. Pp. 165-171.

[3] Voronin A.V., Kuznetsov V.A. *Mathematical models and methods of planning and managing pulp and paper factory* / Petrozavodsk: PetrSU publishing house, 2000. – 256p (in Russian)

[4] Voronin A.V., Shabaev A.I., Pechnikov A.A. Pipelined technology of software development for the management of production resources and processes // *Prospects of science*. 2010. V. 4. P. 95-99. (in Russian)

[5] Gladkov L.A., Kureichik V.V., Kureichik V.M. *Genetic algorithms*: Text book. – 2-nd edition – M: Physics and Mathematics Literature, 2006. – 320 p. (in Russian)

[6] Kositsyn D.P., Kuznetsov V.A., Shabaev A.I. Optimal planning software platform development with cloud computing technology. SPb.: *Proceedings of 12th conference of Open Innovations Association FRUCT*, 2012. Pp. 129-135

[7] Dantzig G.B., *Linear programming and extensions*. / Princeton, N.J., Princeton University Press, 1963

[8] Taha H.A. *Operations Research: an Introduction* / Prentice Hall, 2010. – 832 p.