

# **Development of Planning System for Plywood Production Using “Matrix Designer”**

**Shabaev Anton, Arhipov Ivan, Spirichev Maxim,  
Urban Alexander, Torozero Mikhail  
PetrSU**

1. Problem definition
2. Mathematical model
3. Matrix designer
4. User interface
5. Economic benefits, conclusion

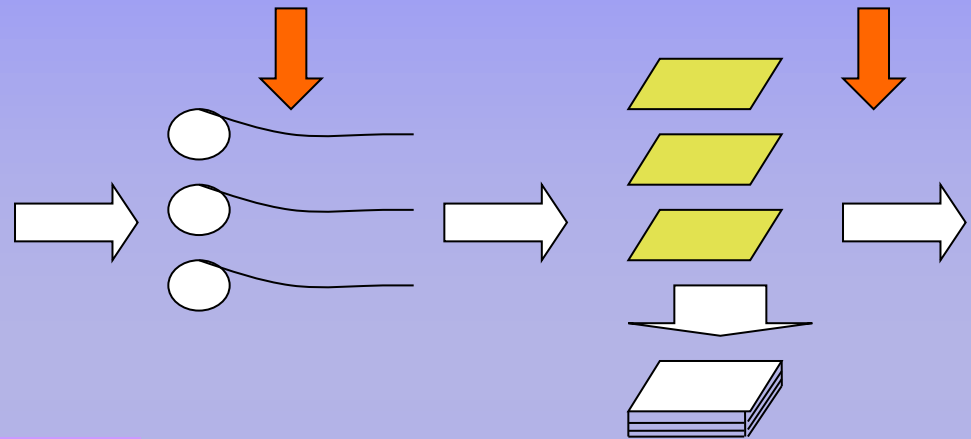
# 1. Problem definition

## Wood delivery

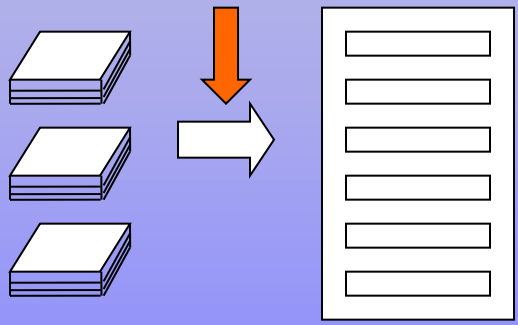


Thickness calculation

Composition calculation



Press calculations



End products or semi-finished for other departments

## 2. Mathematical model

$$\sum_{k=1}^{|C|} \sum_{t=1}^{|T_k|} c_k^p x_{k,t} - \sum_{j=1}^{|Q|} c_j^w y_j - f_1 \left( \sum_{t=1}^{|T|} d_t^p + d^v + d^r \right) + f_2 d^c \rightarrow \max \quad (1)$$

Constraints:

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

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

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

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

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

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

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

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

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

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

*Features of the problem P*

1. Mixed-integer LP problem, several hundreds of variables
2. NP-complete => a special approximate algorithm must be developed.
3. The feasible set of values may be empty, so auxiliary variables are necessary – deviations from the upper and lower constraints: (4),(6)-(8).
4. Several forms of the objective function (1) are possible (by profit, by volume etc.), but this is reflected only in objective function coefficients, and has little effect on the solution algorithm.
5. Dual estimates, obtained during solution of  $\mathbf{P}^*$ , are useful for identification of “bottlenecks” in the production process.

## 2. Mathematical model: constraints matrix for $P^*$

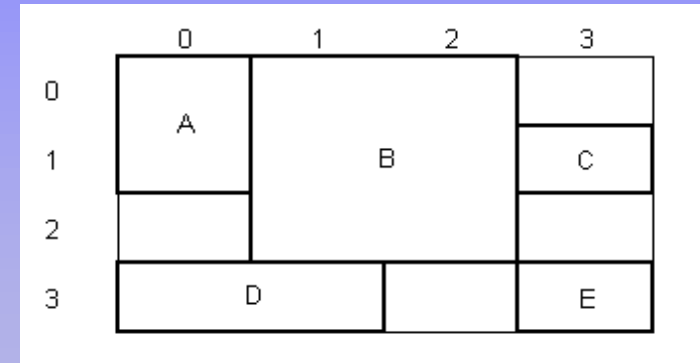
			Q	C	V	T	3		
cost				M (1xS)		f <sub>1</sub> , ..., f <sub>1</sub> , f <sub>2</sub>			
up bound									
W	0 (Wx1)	≤	MM (WxQ)					≤	M (Wx1)
V			MM (VxQ)	MM (VxS)	E (VxV)			=	MM (Vx1)
T	MM (Tx1)	≤		MM (TxS)		E (TxT)		≤	M (Tx1)
P	MM (Px1)	≤		MM (PxS)				≤	MM (Px1)
1	p <sup>m</sup>	≤	M (1xQ)	M (1xS)					
1	v <sup>m</sup>	≤		M (1xS)			-E (3x3)		
1					1 (1xV)			≤	v <sup>M</sup>
				≤					
down bound				MM (1xS)					

## 2. Mathematical model: main solution algorithm for $P$

1. Find vector  $X$  – solution to the problem  $P^*$  (by using multiplicative simplex method).
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  $P$ .

### 3. "Matrix designer". Problems of constructing main matrix.

- Large dimension
- Block structure
- Complicated structure
- Changes with every modification of the mathematical model
- And other



### 3. "Matrix designer" – the special data structure

The special data structure to increase the efficiency of storing and using data with regard to the structure of submatrices, developed in IT-park of PetrSU

It allows to effectively:

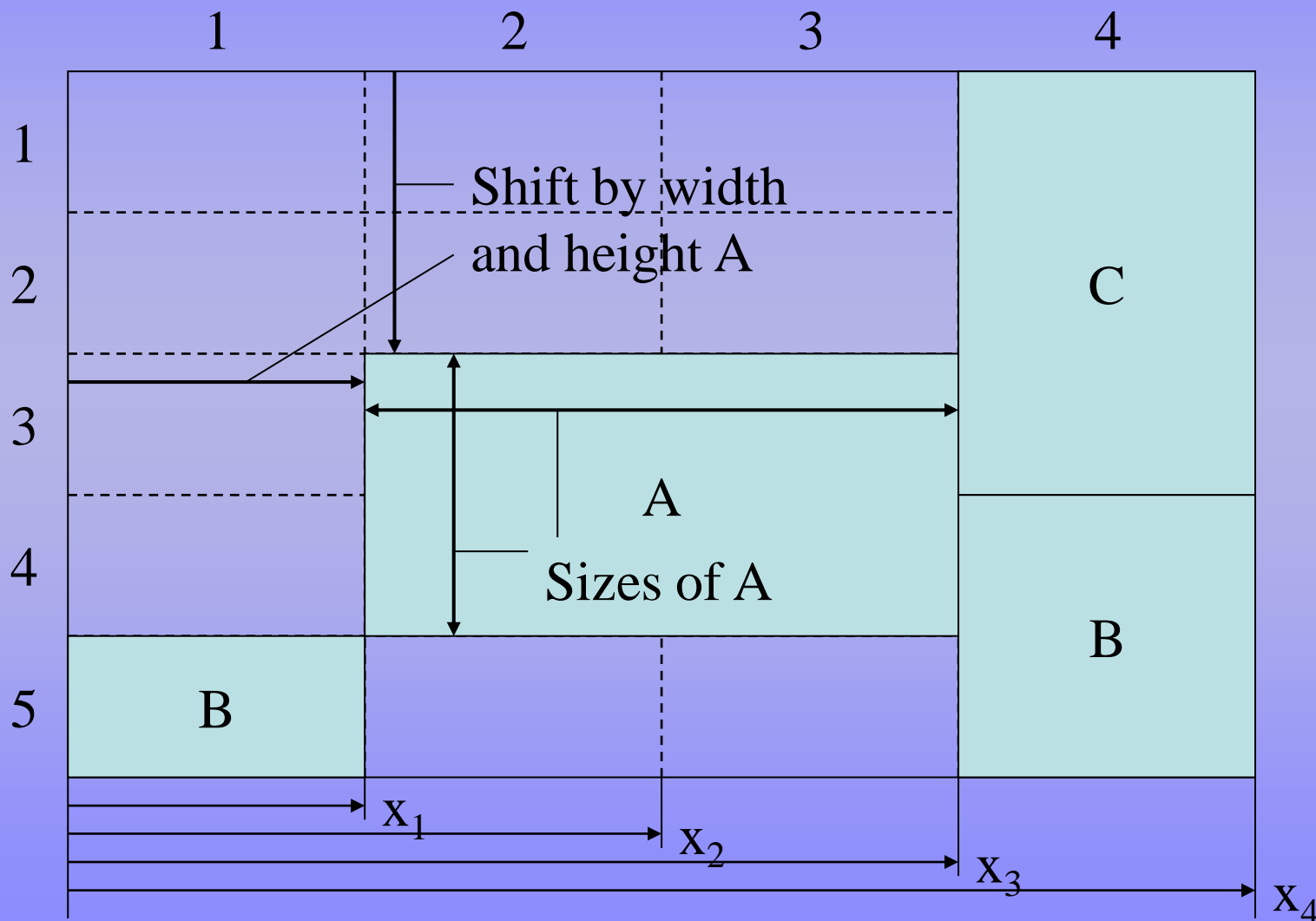
- form a constraints matrix using its block structure
- get data from the matrix
- modify the constraints matrix in case of modifications to the problem statement
- group matrix columns into sets and modify them



# 3. "Matrix designer". Submatrices.

Class name	Memory requirements	Time requirements		
		Get scalar product	Get column	Get element
<i>MatrixSame(m,n)</i>	$O(1)$	$O(m)$	$O(m)$	$O(1)$
<i>MatrixDiagonalSame(m)</i>	$O(1)$	$O(1)$	$O(1)$	$O(1)$
<i>MatrixSimple(m,n)</i>	$O(nm)$	$O(m)$	$O(m)$	$O(1)$
<i>MatrixDiagonalSimple(m)</i>	$O(m)$	$O(1)$	$O(1)$	$O(1)$
<i>MatrixModular(m, n, k)</i> Where $k$ – amount of non-zero elements in column	$O(kn)$	$O(k)$	$O(k)$	$O(k)$

# 3. "Matrix designer". Specify block structure



# 3. "Matrix designer". Example

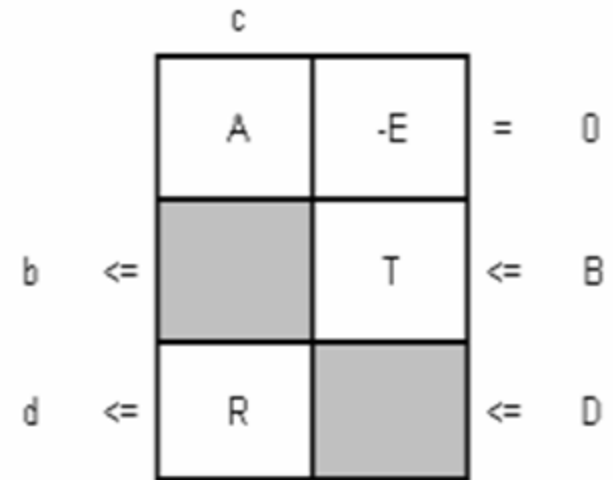
```

MatrixLayout ml = new MatrixLayout();
// Adding submatrices
ml.AddMatrix(new MatrixSimple(A), 0, 0);
ml.AddMatrix(new MatrixDiagonalSame(nTV, -1), 0, 1);
ml.AddMatrix(new MatrixModular(nSV, T), 1, 1);
ml.AddMatrix(new MatrixModular(nP, R), 2, 0);

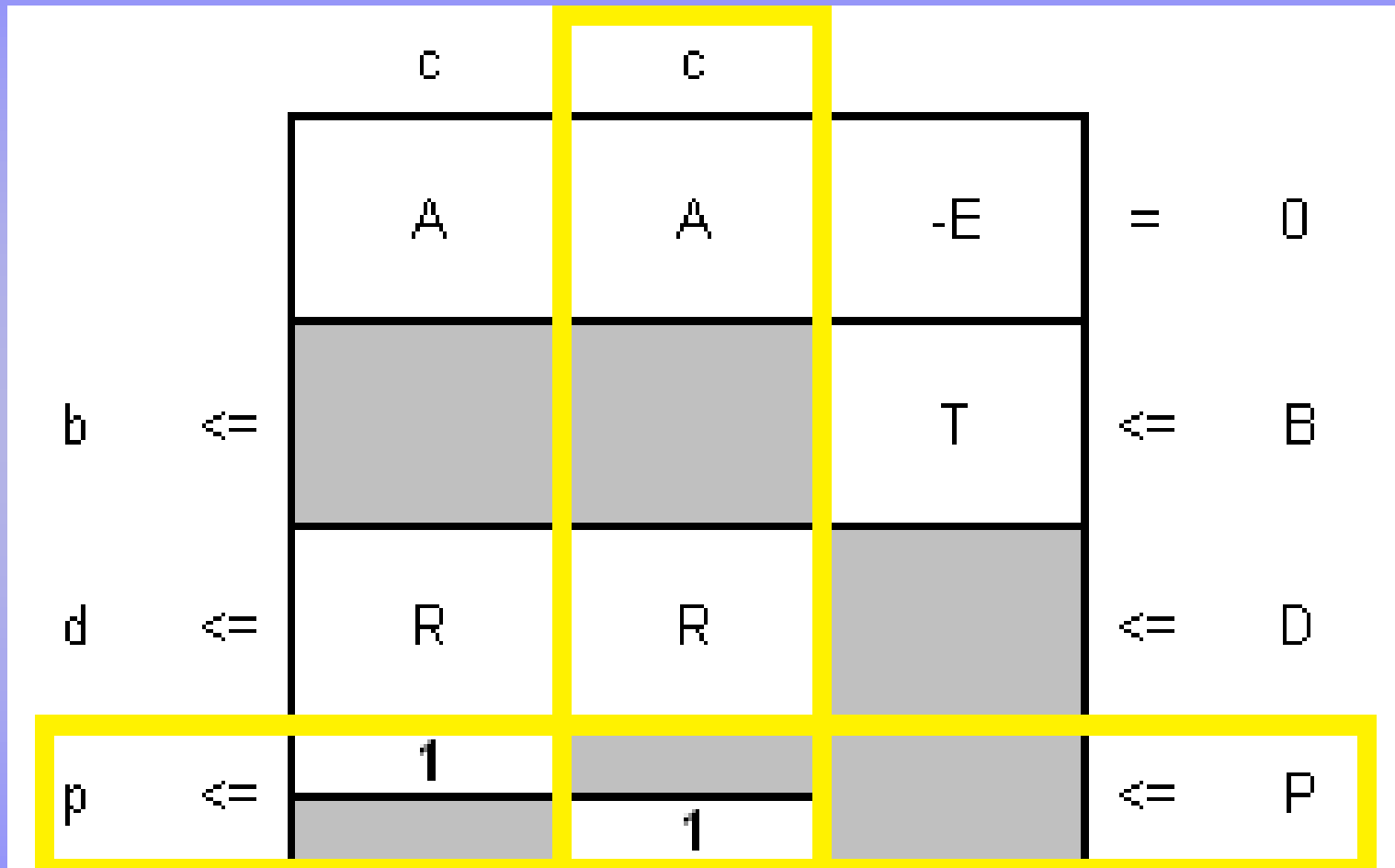
// Adding constraints
ml.AddEqualsRowBound(new MatrixSame(nTV, 1, 0), 0, 1);
ml.AddDownRowBound(new MatrixSimple(b, false), 1, 1);
ml.AddUpRowBound(new MatrixSimple(B, false), 1, 1);
ml.AddDownRowBound(new MatrixSimple(d, false), 2, 1);
ml.AddUpRowBound(new MatrixSimple(D, false), 2, 1);

// Adding cost vector
ml.AddCost(new MatrixSimple(c, true), 0, 1);

```







# 3. "Matrix designer". Modifying problem



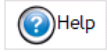
# 3. "Matrix designer". Modifying source code.


```
MatrixLayout ml = new MatrixLayout();  
// Adding submatrices  
ml.AddMatrix(new MatrixSimple(A), 0, 0);  
ml.AddMatrix(new MatrixSimple(A), 0, 1);  
ml.AddMatrix(new MatrixDiagonalSame(nTV, -1), 0, 2);  
ml.AddMatrix(new MatrixModular(nSV, T), 1, 2);  
ml.AddMatrix(new MatrixModular(nP, R), 2, 0);  
ml.AddMatrix(new MatrixModular(nP, R), 2, 1);  
ml.AddMatrix(new MatrixSame(1, nC, 1), 3, 0);  
ml.AddMatrix(new MatrixSame(1, nC, 1), 4, 1);  
  
// Adding constraints  
ml.AddEqualsRowBound(new MatrixSame(nTV, 1, 0), 0, 1);  
  
ml.AddDownRowBound(new MatrixSimple(b, false), 1, 1);  
ml.AddUpRowBound(new MatrixSimple(B, false), 1, 1);  
ml.AddDownRowBound(new MatrixSimple(d, false), 2, 1);  
ml.AddUpRowBound(new MatrixSimple(D, false), 2, 1);  
ml.AddDownRowBound(new MatrixSimple(p, false), 3, 2);  
ml.AddUpRowBound(new MatrixSimple(P, false), 3, 2);  
  
// Adding cost vector  
ml.AddCost(new MatrixSimple(c, true), 0, 1);  
ml.AddCost(new MatrixSimple(c, true), 1, 1);
```


Service for optimal planning of plywood production

-  Main
-  Catalogues ▾
-  Plan
-  Reports ▾

### Edit the plan



Name:  Date:  

**Calculate** 

**Summary**


Income (usd)





Volume (m3)


- Initial data
- Result

Plan settings ▾


#### Veneer leftover





 Add new record




Veneer	Veneer thickness (mm)	Amount	Action
B	1.2	4	 Edit  Delete
CP	1.5	15	 Edit  Delete

 Page 1 of 1  10 items per page 1 - 2 of 2 items 

#### Wood leftover

 Add new record




Wood	Volume (m3)	Action
Birch 2 gr.	555.0	 Edit  Delete
Birch 1 gr.	44.0	 Edit  Delete

 Page 1 of 1  10 items per page 1 - 2 of 2 items 

#### Ordering plywood

 Add new record

Client	Plywood	Price (usd/sheet)	Shipping date	Amount of plywood sheets	Action
CJS "Onego holding"	BB(4)	1	5/27/2013	420	 Edit  Delete

 Page 1 of 1  10 items per page 1 - 1 of 1 items 

# 4. User interface

## 5. Economic benefits, conclusion

The use of the services enables the Customers to

- increase profitability («effect») 1-1.5%
- reduce material losses by 1-3%,
- increase the equipment uptime by 1-3%,
- decrease working assets (raw materials, and semi-finished products in production process)
- reduce the production planning calculations time by 1.5-2 times due to very fast planning and replanning calculations (just few seconds)

The annual effect of  $\approx 80\text{k€}$  at each mill