The recursive method of bottleneck searching in the production systems with uncertain parameters

Pavel Sevastianov * Waldemar Herka

Institute of Computer & Information Science,
Czestochowa University of Technology,Dabrowskiego st.73,
42-201 Czestochowa, Poland, Ph/fax: (+4834) 3250-589

Abstract

The bottleneck problem is a key question of improving performance of production systems e.g. in the production planning (scheduling), in the capacity expansion and in all cases of technological activity systems' optimization. In this paper, the problem of bottleneck in production systems in the aspect of its mathematical formalization is discussed as well as its essential description. Some terms are introduced to describe the features and technological dependence which must be fulfilled by the elements of production system represented by the network. The analysis of the features of presented method is done either on the local level (single production line responsible for one kind of product) or the general one (whole production system). Moreover, the recursive method for bottleneck searching in the production systems with complex structure of technological dependence was elaborated. What is more, algorithms in C++ to facilitate the implementation were presented as well as some aspects of uncertainty in production systems and its impact on bottleneck searching.

Key words: Bottleneck; Discrete bottleneck problem; Production lines

1 Introduction

The losses minimization, profit maximization, aspiration for total utilization of possessed resources are the worldwide problems during last decades. Widely understood optimization concerns almost all areas of our life: beginning with convinient production line organization, and finising on global scale systems (e.g: communication or energetic structures). We mainly concentrate on production systems in this work, however solutions presented here can be successfully extended to different types of real types of problems. First step that precedes optimization work is the exact analysis of the system so as to find the reserves that may be used to enlarge the total throughput of system. However, the presence

* Corresponding author
Email addresses:
sevast@icis.pcz.czest.pl (Pavel Sevastianov), wherka@o2.pl (Waldemar Herka).

of reserves solely does not guarantee such possibility. It is determined mainly by the weakest units of the system so called "bottlenecks". The bottleneck problem is a key question of improving performance of production systems e.g.: in the production planning (scheduling)[9], in the capacity expansion and in all cases of technological activity systems' optimization. The ubiquity of bottleneck problem caused many experts to be interested in it, that includes mathematicians also. It exists quite a lot of different formalized approaches to the solution of this problem nowadays. In the most general mathematical form the bottleneck problem can be formulated as follows: let E = { e_1,e_2..e_m } be the set of stands (machines) and F is the family of subsets of E. Then for $f \in F$ the capacity of the bottleneck machine (minimum capacity $e_i \in f$) is depicted as cap(f,C) where C is the capacity vector which size is m. The capacity of family F is defined as follows:

$$cap(F, C) = \max\{cap(f, C) | \forall f \in F\} \quad (1)$$

For a given capacity vector C system's optimization problem can be formulated as:

$$\max_{f \in F} \min_{e \in f} c(e) \quad (2)$$

Problem (2) is called the discrete bottleneck problem. This problem has been studied extensively by many authors [3]-[7],[10],[11],[13],[14],[18],[21]. A classical method, the threshold algorithm, was established at first by Edmonds and Fulkerson [11]. Since then, many versions of special cases, variants and generalizations have received wide attention in combinatorial optimization. The typical examples are as follows: bottleneck assignment problem [10],[11],[13],[14], the bottleneck spanning tree problem [7],[13],[16], the bottleneck matroid base problem [3] and the bottleneck Travelling Salesman Problem [6],etc. In [16], the problem of searching for minimum spanning tree, which is the bottleneck of the graph, that represents the dependencies of the system was presented. The characteristic feature of this work, is the use of fuzzy random numbers to represent costs in the graph edges. This makes the model close to real system, where uncertainty plays a pivotal role. Adams et al. [1] consider makespan minimization problem for a jobshop. The main idea behind their shifting bottleneck procedure is to consecutively identify the "bottleneck" machine and sequence the operations on this machine by using method developed by Carlier[8]. In [15] as a bottleneck the machine that has maximal loading was determined, which is defined as a sum of processing times assigned to this machine. In recent years, the bottleneck optimization has been generalized to lexicographical type of optimization [5],[21]. The extended form of the bottleneck problem can be stated as follows [28]:

$$\max cap(F, C) \quad (3)$$

$$\mathrm{W}^{\mathrm{T}}(C - \overline{C}) \leq D, \quad (4)$$

$$C \geq \overline{C}, \quad (5)$$

where $W = (w_1, w_2, .., w_m)^T$, w_i is the cost of increasing the capacity of machine (stand) e_i by one unit , and stands for the original capacity vector (before optimization).

This method is attractive, because it includes the optimization of cost. However in the case of real systems, the localization of bottleneck in a given part of network may be a serious problem. The point is that the optimization doesn't allow the bottleneck to be eliminated, because its elimination in one part of the system makes a different machine in a different part of the system to become a bottleneck. For technologists, the position of bottleneck - whether it's at the beginning or at the end of the production line - is of great importance. Suppose that the first stand point of production line is the bottleneck of the system. Then all consecutive machines should work with smaller throughput than it is possible. This causes that the technological possibilities of whole system are wasted. That is why, the optimization based on this approach which doesn't consider the non-equivalence of finding the bottleneck in different parts of the system is rather theoretical task than real-world problem. In practice, there is always a problem to find a location of bottleneck in a view of an image of "ideal" (best possible) system, which reflects the dreams of constructors and technologists dealing with exploitation of the system. For this purpose, in this paper we present the determination of flexible production process, which looks like a comprehensible compromise between technological requirements. The flexible production process may be consider like a model of "ideal" process. Furthermore, the problem of finding the bottleneck is solved by comparing the real system with the "ideal one", which stores the structures and technological requirements of the real one.

The rest of the paper is organized as follows. In section 2, the determination of the continuously working technological chain is introduced. The drawbacks and advantages in its using for the solving real problems are described as well as the method of bottleneck searching in production systems that bases on it. In this section, new terms to facilitate coherent formulation of the described method are introduced. The new terms are as follows: Information granule, Local Factor, Global Factor, Local Vector, Global Vector. Additionally, the method is described in algorithmic way, which enables not very complicated implementation. Section 2 contains the numerical example illustrating the efficiency of the proposed method in conditions of uncertain system parameters represented by intervals(they reflect the real systems where we often meet with parameters are defined in an approximate way). Section 4 contains the summary and the perspectives of the future work.

2 Modelling of the ideal system

As the "ideal" we will consider continuously working technological chain, which has only one bottleneck in the final stand (machine). In other words:

*As a **continuously working technological chain** is such an arrangement of stands (machines), which capacities allow to match technological requirements, as well as in which the final stand works with its passport (nominal) capacity - such a system works in the stroke of the final stand.*

Indisputably, technological chain working in this way possesses larger stability than the one, in which bottleneck is located in different place. The main reason of this is the fact that, if the final stand is a bottleneck, this means that every stand of the system possesses the reserves of productive power. It guarantees the flexibility needed for taking into account the uncertainties caused e.g. by human factor. In addition, stands which possess certain excesses of productive power are not so susceptible to wear and breakdowns as these working "at full capacity" - without any reserves. Such a system is also easier in the development - by modernizing the final stand we have a guarantee that the general improvement of throughput of the system will be the final result. The exact qualification of productive possibilities by the possibilities of the final stand is the next essential advantage of this system. It is a separate matter how large the improvement it will be - it is determined by the capacity of stand which is the "closest " to the bottleneck.

2.1 The construction of the continuously working technological chain

The first step is the creation of model of continuously working system on the basis of existing technological dependencies and capacity of the final stand of analyzed subgraph. At this stage, the knowledge of the passport (nominal) capacities of individual stands is not required. The essence of this stage is to find the dependence that determines the number of half-finished products needed for the one production unit of the final stand (finished product), and to refer it to the capacity of the final stand (according to the definition of technological chain working in continuous way). We will describe the algorithm of constructing continuously working technological chain with the help of simple example presented on Fig.1, where: S_{src}, S_{tar}- the stores (source and sink respectively), the numbers near arrows denote technological requirements, which illustrate the proportions of details produced by a given stand needed for one unit of detail produced by final stand (final product). Technological requirements in a traditional form are also presented by the matrix of connections (see Fig.2).

Let us describe the main points of the method elaborated.

Step 1: To define the requirements imposed on individual stands of subgraph in continuously working system let's define the abstract data structure called Local Vector. It will

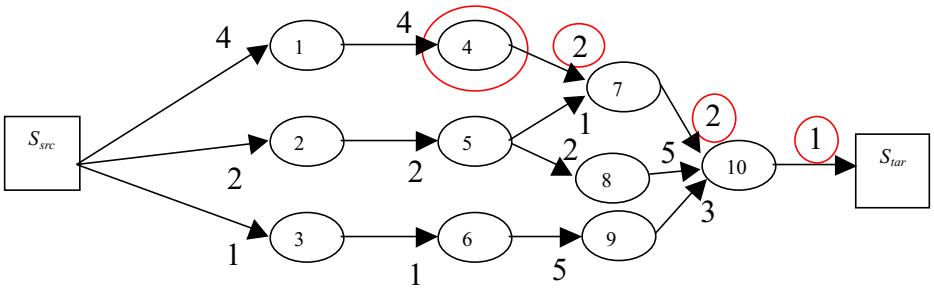


Fig. 1. Subgraph used for the Local Vector construction.

	1	2	3	4	5	6	7	8	9	10	S_{src}	S_{tar}
1				4								
2					2							
3						1						
4							2					
5							1	2				
6									5			
7										2		
8										5		
9										3		
10												1
S_{src}	4	2	1									
S_{tar}												

Fig. 2. Matrix of connections of the system shown in Fig1.

contain information that allows to specify: the stand which concerns concrete entry, its location in subgraph and coefficient that determines requirements imposed by final stand, which determine the continuous flow in subgraph.The local vector will be a vector consisting of the entries, which we call Local Factor. Local Factor, which is assigned to a given stand in subgraph consists of entries of certain structure, which we call Information Granule. Every Local Factor possesses at least one Information Granule in case, when a stand has only one successor (line fragment without forks). The number of Information Granules depends on the number of forks on output of the stand (at this stage only the forks in subgraph are taken into account).

Below, the constructing of the Information Granule and rule of its creating are presented. For example the Information Granule for seventh stand (see Fig. 1) that describes the connections with the tenth stand will look as follows: $\{2, 10, 10\}$ (accordingly for the connection of the fourth stand with the seventh : $\{4, 7, 10\}$). In fact the Information granule is the complex object consist of sequence entries which may be performed by usual numbers, vectors and object as well. The way of interpreting the entries on corresponding positions of the Information Granule is as follows:

- on the first Information Granule position, the product of the coefficients (from matrix of connections) of technological requirements for the considered stand is placed as well as the coefficient taken from first position of the Information Granule of the next stand taking into consideration the direction of the flow and the stand which had been analyzed just before the considered stand. This is very important, particularly when a stand possesses forks and so it has several successors, e.g. fifth stand (several Information Granules fall on one Local Factor of given stand);

- the second position contains the index of the successor incidental with a given Information Granule;
- the third position contains the index of the stand, which is the final stand in the analyzed subgraph. At this stage, this information seems to be redundant because in the subgraph all Information Granules contain information about the same stand on the third position, however it is very useful in the next step of the algorithm that constructs continuously working technological chain, what will be demonstrated below.

Since the procedure of the Information Granule first position determination is based on the recurrence rule, it is obvious to define the data, which will initiate this process. In recurrent processes the order of examination of occurrences to compute, is very important. If we have the element i, we can evaluate the element $i+1$. In a discussed algorithm, the beginning of examination is in the final stand of subgraph, so one should determine his Information Granule in advance. Let's define initially the form of the Information Granule identifying the connection of the stand i with the stand j:

$$GrInfo_{ij}[a_1, a_2, a_3] \tag{6}$$

where: a_1 denotes the coefficient which is calculated on the base of the information taken from previously examinated Local Factors, a_2 - the index of stand that is joined with stand i, which concerns granule except itself (it is specified by j), a_3 - the index of final stand in the given subgraph.

In our example, on the first position of Local Vector (the content of first Local Factor) will be the Information Granule that include the following data: $\{1, 10, 10\}$. The characteristic fact is that the Local Factor of the final stand always contains one Information Granule, which is not assigned to any concrete connection. The value of the first position of Information Granule considered is 1 (the point is to establish the dependence that corresponds to the production of the one unit of the product by the final stand), the second and the third ones, have the same values i.e. the index of the stand, which is assigned to this Local Factor - the final stand of the subgraph corresponding to the Local Vector that is being filled. On the basis of this information and entries in matrix of connections we are able to define Information Granules for stands 7, 8, 9, and further on their basis for stands 4, 5, 6 and so on to the stands that are directly connected with initial (source) store S_{src} (1,2,3).On the base of the previous consideration we introduce the formal description of the Information Granule content, i.e. parameters $a_{1,ij}, a_{2,ij}, a_{3,ij}$ for the stand i connected with the stand j :

$$a_{1,ij} = c_{ij} \cdot c_{jk} \cdot c_{kk1} \cdot, ..., \cdot c_{nn+1}, \quad a_{2,ij} = \\ = \ j, \quad a_{3,ij} = S_{tar}, \tag{7}$$

where: c_{ij} - the coefficients from the matrix of stands connections (see Fig. 2) and i,j=1 to n (in our exam-

ple n =10).

Of course, $n+1$ is the index of the final stand of the subgraph - S_{tar}. The Local Vector is a kind of container that includes the Information Granules of individual stands of the subgraph. It doesn't mean that information granules are placed in purely random way: the order of distribution is of great importance for calculations, according to the recurrent algorithm for evaluation of coefficient on the first position of corresponding Information Granule. For this reason the Local Factor was introduced. It groups Information Granules in a packet relates to a given stand. Therefore the Local Factor contains information which is contained in Information Granules as well as the order of its allocation in Local Vector. Information about the order of allocating is particularly important in case of the calculation of the next Information Granule. Without such an arrangement of Information Granules the building of rational algorithm for calculation of the next Information Granules that are necessary to compute continuous flow would be much more difficult. The characteristic feature is that, thanks to the Local Factor there is the possibility of allocating many pieces of the Information Granule in one position of the Local Vector - depending on the number of the forks of given stand in one subgraph. Additionally, this possibility confirms the necessity of the Local Factor introduction. Let's describe the algorithm of constructing the Local Vector of the subgraph:

(1) Allocation of the intermediate Structure Vector:{the index of stand, collection of Information Granules}. Its size corresponds to the number of stands in the subgraph.
(2) Insertion of the Information Granule to the Local Factor of the final stand (always on the first position of the vector).
(3) Filling of the Local Vector on the basis of the Local Factor of the final stand and formula (7) for calculation the Information Granule.

Resulting Local Vector (after steps 1-3) in an examined case, is performed in the Table 1.

The calculation of Local Factor for stands 5 and 2 requires additional explanations, because stand 5 possesses some forks, and stand 2 is calculated on the basis of stand 5. In this case, in order to calculate the Local Factor of stand 5, two steps that precede adequate calculation are necessary. These are: calculation of the Local Factor of stand 7 and calculation of the Local Factor of stand 8. Analogically to stand 5, stand 2 also needs two steps of calculation, because the Local Factor of stand 5, which precedes in calculations stand 2 is changing. This type of situations should be noticed and whenever the Local Factor of stand, on which the calculation of analyzed Local Factor depends is changing, calculations should be repeated. We do not add the next Information Granule to the Local Factor of stand 2, because it possesses only one predecessor (taking into account direction of the flow, it is the successor).

Table 1
Local Vector of the system performed on Fig.1

stand	*10*	*7*	*4*	*1*	*5*	*2*	*8*	*9*	*6*	*3*
Local Factor	*{1,10,10}*	*{2,10,10}*	*{4,7,10}*	*{16,4,10}*	*{2,7,10}, {10,8,10}*	*{12,5,10}*	*{5,10,10}*	*{3,10,10}*	*{15,9,10}*	*{15,6,10}*

Step 2: The next stage of the continuous flow calculation is the aggregation of information obtained in the previous stage on the level of individual subgraphs to structure on the level of the whole system, which we call the Global Vector. This is a structure, to which an access is analogical to vector (index-related), and its size is equal to the number of stands in the system. The indices of the Global Vector correspond to the indices of stands (unlike in Local Vector). The individual units of Global Vector we call the Global Factor. The Global Factor, like the Local Factor, groups Information Granules. The difference is that the individual Local Factors are constructed on the level of the subgraph and that's why the third unit of every Information Granule in the Local Factor is the same over the Local Vector as a whole. It is equal to the index of the final stand of the subgraph, which concerns this Local Vector. In a situation when the stand belongs to several subgraphs, Information Granules can differ in the aspect of the third element. The procedure of the filling the Global Vector isn't complicated - it relies on the review of all Local Vectors and on adding Information Granules contained in Local Factors that belong to stands that have the same indices (in numbering of all the stands of system) to the corresponding Global Factors.

The algorithm in C++ is presented in Table 2.

Step 3: Calculation of the continuous flow system capacity.

On this stage we make the preliminary optimization of the forks in common stands for several subgraphs stands and we calculate the capacity of stands that is required for the definition of the continuously working system. The stand on which depends more than one subgraph possesses various requirements imposed by subgraphs which contain them. The Subgraphs that are autonomically treated, do not keep in Local Vectors, which represent them, any information about requirements of different subgraphs in relation to the common stands. It is possible to observe such information in Global Vector, which represents the whole system. In the aim for matching the optimal fork the portion of the technological requirements of a given subgraph to concrete stand in relation to global requirements (the technological requirement of the common stand are imposed by various subgraphs on the level of the system) should be defined. This can be implemented in the way presented in Table 3.

If we have the portion of products of concrete stand directed to a given calculated subgraph , then we are able

Table 2
Algorithm for evaluation of the Global Vector.

```
1 CalculateGlobalVect ()
2 {
3 for( i=0 ; i < numberOfStands() ; i++ )
4         {
5         for( j= 0 ; j < numberOfSubGraphs() ; j++ )
6                 {
7                 GlobalVect[i]+=LocalVect[j]. find(i);
8                 }
9         }
10 }
```

where:
- **numberOfStands() –** function that gives number of stands in the system;
- **numberOfSubGraphs() -** function that counts the number of subgraphs in the system;
- **LocalVect**[j]**. find**(i) – method that allows to find Local Factor of stand *i* in Local Vector *j* (corresponding to Subgraph *j*) – if Local Factor will be not found then returns code of error e.g.NULL and inserting will be ommited;

Table 3
Calculation of the charge(load) sharing.

```
1 double  EvaluateChargeSharing( int i ,int subG)
2 {
3 double FullCharge = 0;
4 double PartialCharge =0;
5 double portion = 0;
6 for( int j = 0 ; j < numberOfGranule( GlobalVect[i] ); j++)
7         {
8         FullCharge+=GlobalVect[i].Granule[j][0] * avg( Pass   Cap(GlobalVect[i]. Granule[j][2]) );
9
10        if( GlobalVec t[i].granule[j][2] == finalStand (subG) )
11                PartialCharge += GlobalVect[i].Granule[j][0] ;
12        }
13  PartialCh arge *= avg( PassCap( finalStand(subG) ) );
14
15  portion = PartialCharge/FullCharge;
16  return portion;
17 }
```

where:
- input parameters: **i** , **subG** – stand and subgraph considered(respectively);
- **GlobalVect[i].gr anule[j][2] –** this call returns third element of Information Granule*j* of Global Factor*i* (this identifies concrete subgraph);
- **numberOfGranule(GlobalVect[i])** - function that counts the number of Information Granules in Global Factor *i*;
- **avg(Pass Cap(GlobalVect[i].granule[j][2]))** - it means avaraging of passport capacity(if we deal with fuzzy intervals this is the defuzzification but when we deal with real numbers we don't need this operation) of the stand described by second element of j-th Information Granule of i-th Global Factor – this is passport capacity of the final stand of the subgraph from which is given Information Granule
- **finalStage(subG) –** function returns index of the final stand of subgraph *subG*.

to calculate the capacity of stand required to treat the flow in the system as a continuous one. It will be the generalized capacity, determining the whole capacity of the concrete stand so as to maintain the continuous flow on the scale of system considered. At a local level, taking into ac-

count the flow in the given subgraph, this capacity may be treated as the global capacity of the stand considered (when subgraphs are disjoint) or may be smaller when we analyze common stand for several subgraphs.

In the calculations we use the following formula:

$$Ccont_i = \sum_{j=0}^{n} \lambda_{ij} * Cfin_j \tag{8}$$

where: $Ccont_i$ - the capacity of stand i in continuous flow; the λ_{ij} - the portion of stand i production directed to subgraph j (calculated by the **EvaluateChargeSharing** procedure - see Table 3); n - the number of subgraphs, which depend on stand i; $Cfin_j$ - the capacity of the final stand of the subgraph j.

2.2 Searching the bottleneck of the system

Firstly, we compare the passport (nominal) parameters (capacities) of individual stands with those that are the result of the continuously working system construction. The operation of comparison may be realized on the global or local level. On the global level we compare summary requirements of the stands in continuous flow system $Ccont_i$ calculated in the way presented above (see eq.8) with passport(nominal) capacity of the stand. In this way we obtain the qualitative information about the relation of requirements of the continuous flow system to passport(nominal) capacities ($Cpass$). If we have ($Ccont < Cpass$) then we can perform this result in the verbal form as "the deficit" and if we get ($Ccont < Cpass$) we will treat this result as "the excess", indeed when we obtain ($Ccont = Cpass$) this result may be described as "the ideal matching". This means that the stands where the above comparison resulted in the deficits, are the potential bottlenecks. However, it finishes here, because it is not possible to state precisely, which stands are the real bottlenecks if several stands with "the deficit" will appear - all are treated identically (Boolean operators return only two values: 0 and 1). Besides, if a stand with "the deficit" is common, it is not even possible to determine a subgraph, in which this stand may be the bottleneck. With respect to the analysis of the system from view point of the summary stands capacities, all subgraphs which contain the given stand are equally suspected of being the bottleneck. The situation is little easier, though insufficiently clear for the disjoint systems. Since there is no problem of sharing the stands, there isn't also a dilemma: which subgraph will be more impacted by "the deficit". It is possible to state that the global approach is useful only in case of purely superficial estimation of the system as a whole (when its weaker sides are pointed out)without quantitative details. The comparison on the local level (the subgraphs level) is a little bit complex and more labour-consuming, of course. However, the results of such a procedure are more complete in quantitative

sense. If we make such a comparison we should use the portion of the production directed to a given subgraph. Then, we don't consider summary capacity, but the capacity required by continuous flow in a given subgraph. This capacity is analogical to passport (nominal) capacity of the stand. The algorithm of searching the bottleneck in the subgraph is presented in table (Table 4).

This algorithm may be divided into two essential sections.

The first section of this algorithm consists of the operations carried out in a loop for all elements of the subgraph:

- the calculation of the capacity required for continuous working system (by use the **EvaluateChargeSharing** (see Table 3) function requirements we limit ourselves to the requirements of the given subgraph) (lines 7 - 10);
- the calculation of stand's passport (normal) capacity portion directed to the given subgraph (line 11);
- the calculation of the difference between capacities (the difference between the passport (nominal) and required capacities is specified by the deficit of the productive power (< 0) or its reserve (> 0) in units of elements produced by the stand (it is called the local difference) (line 12) ;
- the calculation of so-called global difference (the deficit /reserve of productive power related to the total production of the stand) and inserting it to vector containing differences between passport and required capacities at the stand's position in the Local Vector (in the subgraph);

The second section of this algorithm runs according to a standard minimum searching algorithm.

However the results obtained when using the algorithm described above does not contain all elements of the vector of differences, but only those, which demonstrate the deficit of the productive power (< 0). The minimal difference is the maximal deficit of productive power, and the corresponding stand is the bottleneck of the system. It may be happen that all differences will be positive - it is extremely comfortable situation - then we have to deal with a subgraph working in conditions of continuous flow. OIt is easy to see that in extreme case the possible number of the bottlenecks in the whole system should equals the number of the subgraphs in the system. However, there are situations, in which the bottleneck may be common - then the number of bottlenecks may be less (this doesn't concern disjoint systems, where the number of bottlenecks is constant).

3 Numerical Example

In this section, numerical example that illustrates the efficiency of the bottleneck searching method in complex production systems is presented. The parameters (capacities) of the machines (stands) are introduced in an interval form to take into acco-

Table 4
The algorithm of searching the bottleneck in the subgraph.

```
1 int  FindBottleneck( i )
2 {
3 vector<CapType> diff(sizeOf(i));
4 for( int j = 0 ; j < sizeOf(i); j++)
5       {
6       CapType SumCapCont = 0; CapType  LocalDiff = 0;
7       for( int k = 0 ;  k < numberOfConsumers(j) ; k++)
8               {
9               SumCapCont += EvaluateChargeSharing(j,k) *CCont(j)* PassCap( finalStand( k ) ) ;
10              }
11      CapType CapPass = EvaluateChargeSharing(j,k) * PassCap(j) ;
12      LocalDiff = CapPass – SumCapCont;
13      diff[j] = LocalDiff / ( EvaluateChargeSharing(j,k) * PassCap( finalStand( k ) ) ;
14      }
15 CapType minval = diff[0];
16 int stand_index = 0;
17 for( int l = 0 ; l < sizeOf(i); l++)
18      {
19      if(diff[l]<CapType_cast(0))
20              if(diff[l]<minval)
21                      {
22                      minval = diff[l];
23                      stage_index = l;
24                      }
25      }
26 return globalIndex(stand_index,i);
27 }
```

where:

- input parameter **i** – determines subgraph in which a bottleneck is finding;
- returned result: index of the stand (in global numbering) which is a bottleneck of the subgraph;
- **vector<CapType> diff(sizeOf(i))** – allocation of the of the vector with elements of type CapType (it is abstract type, in reality capacity may be represented by e.g. interval);
- **numberOfConsumers(j)** – returns number of stand consumers – subgraphs depended on stand *j*;
- **PassCap(finalStand(k))** – returns passport capacity of the subgraph k final stand ;
- **CCont(j)** – function calculates sum of technological requirements (by one final product) on the basis of information from Information Granule of the Global Factor *j*;
- **globalIndex(stand_index,i)** – global index evaluating on the basiss of local index as well as subgraph index1;

unt an uncertainty, which is tightly connected with the reality. Some stages of the algorithm are the same for all data type (the division on the subgraphs and determination of the technological requirements). The other stages are similar in relation to the structure of the calculations (algorithms). The feature, which makes the stages different, is the different rules used in traditional and interval arithmetic.

3.1 The clustering the system to the subgraphs

The first preliminary step of describing method is the clustering the system to the subgraphs. It allows to analyze the system on the local level and to find the stands which may be the bottlenecks in concrete technological chain(subgraph). Table 5 shows the results of this step for the example presented in Fig. 3.

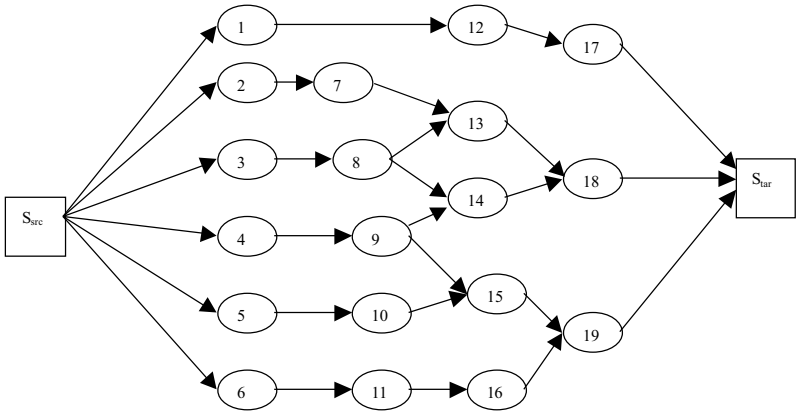


Fig. 3. Complex production network.

Table 5
Clustering of the system.

Subgraph	Set of stands
1	17,12,1
2	18,13,7,2,8,3,14,9,4
3	19,15,9,4,10,5,16,11,6

3.2 Technological requirements

The technological requirements, illustrating the proportions of half-products included in unit of the final product are defined as a matrix of connections.

The entries in Table 6 should be interpreted as follows: if in a given field is a non-void value, then it is the technological requirement imposed by the stand indicated by the column in relation to stand indicated by the row.

3.3 The construction of the continuously working system

The passport (nominal) capacities of individual stands we introduce as intervals to show the advantages of the method in uncertain environment. For the example presented in Fig.3 they are shown in Table 7.

3.3.1 Calculating the Local Factors of the individual subgraphs

The Local Vectors obtained as the result of calculating the Local Factors for the clustered subgroup are shown in Tables 8,9,10.

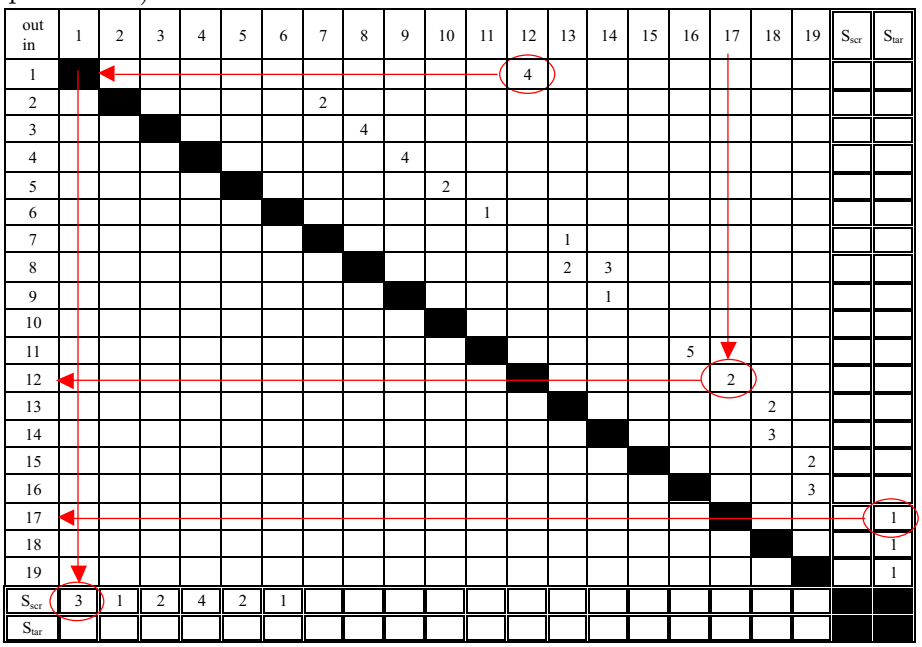
Table 8
The Local Vector of the first subgraph.

stand	**17**	**12**	**1**
Local Factor	{1,17,17}	{2,17,17}	{8,12,17}

3.3.2 The composition of the Global Vector with the Local Vectors

As the result of executing the **Calculate-GlobalVect** procedure (see Table 2) we get the Global Vector shown in Table 11.

Table 6
The matrix of connections of the graph(network) from Fig.3 (filled with technological requirements)



out / in	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	S_{scr}	S_{tar}
1												4									
2							2														
3								4													
4									4												
5										2											
6											1										
7													1								
8													2	3							
9														1							
10																					
11																5					
12																	2				
13																		2			
14																		3			
15																			2		
16																			3		
17																					1
18																					1
19																					1
S_{scr}	3	1	2	4	2	1															
S_{tar}																					

Table 7
Passport capacities.

stand	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
passport capacity	[390,400]	[82,86]	[1100,1300]	[340,360]	[51,53]	[83,89]	[47,49]	[270,300]	[73,77]

10	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**
[24,26]	[80,86]	[63,71]	[40,44]	[62,70]	[13,15]	[16,18]	[42,48]	[19,21]	[4,6]

Table 9
The Local Vector of the second subgraph.

stand	**18**	**13**	**7**	**2**	**8**	**3**	**14**	**9**	**4**
Local Factor	{1,18,18}	{2,18,18}	{2,13,18}	{4,7,18}	{4,13,18}, {9,14,18}	{52,8,18}	{3,18,18}	{3,14,18}	{12,9,18}

Table 10
The Local Vector of the third subgraph.

stand	**19**	**15**	**9**	**4**	**10**	**5**	**16**	**11**	**6**
Local Factor	{1,19,19}	{2,19,19}	{4,15,19}	{16,9,19}	{4,15,19}	{8,10,19}	{3,19,19}	{15,16,19}	{15,11,19}

3.3.3 The calculation of the continuous working system capacity

The results obtained in the previous point allow to calculate capacities of the continuously working system, which are the key to find the bottlenecks of the system. The calculations are carried out using formula (8). For this, an earlier choosing of the optimized fork is required (the **EvaluateChargeSharing** procedure, see Table 3) with respect to the

Table 11
Global Vector

stand	1	2	3	4	5	6
Global Factor	{8,12,17}	{4,7,18}	{52,8,18}	{12,9,18}, {16,9,19}	{8,10,19}	{15,11,19}

7	8	9	10	11	12	13
{2,13,18}	{4,13,18}, {9,14,18}	{3,14,18}, {4,15,19}	{4,15,19}	{15,16,19}	{2,17,17}	{2,18,18}

14	15	16	17	18	19
{3,18,18}	{2,19,19}	{3,19,19}	{1,17,17}	{1,18,18}	{1,19,19}

possibility of the "false bottlenecks" appearance . The user's intervention through "choosing by hand" of the fork is also possible. Instead of the calculation of the fork coefficient λ_{ij} by the **EvaluateChargeSharing** procedure, it may be chosen in an experimental way. Furthermore, we can state that the results will not be the optimal ones. The more complicated the system is, the harder is to choose the appropriate coefficients in this way. The capacities required by the continuously working system (global requirements imposed by all subgraphs of the system) are presented in Table 12.

3.4 Searching the bottlenecks of the system

At first we have get the the local differences by use of **FindBottleneck** procedure executed for every subgraph individually (line 13 in Table 4). They are presented in Table 13.

However, the differences calculated in this way can not be compared directly. These values describe only a concrete quantity of the elements, which are produced above the requirements of the continuously working system (positive volumes) or below them (negative). With respect to different technological requirements imposed by the final stand on the individual stands, one should introduce a transformation that allow to standardize the treatment of differences. So, they should be related to the complete sets of requirements imposed by the final stand. In the line 13 of **FindBottleneck** the procedure (see Table 4) realizing this transformation is shown.

The global differences calculated in such a way allow to find the bottlenecks in individual subgraphs by searching the maximal difference between the requirements of the continuously working system and the passport (nominal) capacities.

Table 12
Capacities of the continuously working system on the basis of the system shown in Fig. 3

stand	1	2	3	4	5	6
capacity	$8*PassCap_{17}$ [336,384]	$4*PassCap_{18}$ [76,84]	$52*PassCap_{18}$ [988,1092]	$12*PassCap_{18}$ $+16*PassCap_{19}$ [228,252]+[64,96] =[292,348]	$8*PassCap_{19}$ [32,48]	$15*PassCap_{19}$ [60,90]

7	8	9	10	11	12	8
$2*PassCap_{18}$ [38,42]	$4*PassCap_{18}+9*PassCap_{18}$ [76,84]+[171,189] =[247,273]	$3*PassCap_{18}$ $+4*PassCap_{19}$ [57,63]+[16,24] =[73,87]	$4*PassCap_{19}$ [16,24]	$15*PassCap_{19}$ [60,90]	$2*PassCap_{17}$ [84,96]	$4*PassCap_{18}+$ $9*PassCap_{18}$ [76,84]+[171,189] =[247,273]

13	14	15	16	17	18	19
$2*PassCap_{18}$ [38,42]	$3*PassCap_{18}$ [57,63]	$2*PassCap_{19}$ [8,12]	$3*PassCap_{19}$ [12,18]	$1*PassCap_{17}$ [42,48]	$1*PassCap_{18}$ [19,21]	$1*PassCap_{19}$ [4,6]

Table 13
Local Differences to the example from Fig.3

stand	1	2	3	4	5	6	7	8	9
LocDiff[1]	[6,64]	-	-	-	-	-	-	-	-
LocDiff[2]	-	[-2,10]	[8,312]	[-11,26]	-	-	[5,11]	[-3,53]	[-8.25,0.75]
LocDiff[3]	-	-	-	[3,42]	[3,21]	[-7,29]	-	-	[-5.75,3.25]

10	11	12	13	14	15	16	17	18	19
-	-	[-33,-15]	-	-	-	-	[0,0]	-	-
-	-	-	[-2,6]	[-1,13]	-	-	-	[0,0]	-
[0,10]	[-10,26]	-	-	-	[1,7]	[-2,6]	-	-	[0,0]

Table 14
The global differences obtained from the **FindBottleneck** procedure.

stand	1	2	3	4	5	6
GlobDiff[1]	[0.16,0.19]	-	-	-	-	-
GlobDiff[2]	-	[-0.02,0.13]	[0.007,0.31]	[-0.05,0.11]	-	-
GlobDiff[3]	-	-	-	[0.03,0.65]	[0.065,0.656]	[-0.11,0.48]

7	8	9	10	11	12	13
-	-	-	-	-	[-0.39,-0.15]	-
[0.12,0.29]	[-0.012,0.214]	[-0.14,0.013]	-	-	-	*[-0.05,0.15]*
-	-	[-0.36,0.20]	[0,0.625]	[-0.167,0.43]	-	-

14	15	16	17	18	19
-	-	-	[0,0]	-	-
[-0.017,0.23]	-	-	-	[0,0]	-
-	[0.08,0.87]	[-0.17,0.5]	-	-	*[0,0]*

While analyzing the obtained results, an interesting question arises: how should be interpreted the position of global difference in relation to zero point. The answer relies on the type of data on which we operate:

- for the real type data, it is possible to state intuitively that if difference is equal to 0 then we have an ideal matching of our capacities to the system (the stand exactly fulfils the requirement with no reserve); if the difference is greater than 0 - then stand fulfils the requirements of the continuously working system and it additionally possesses some productive power reserve,

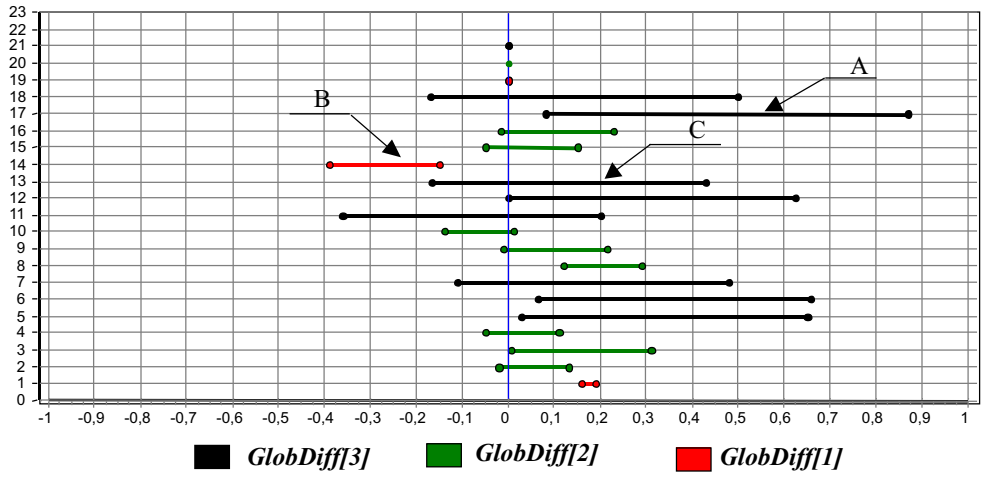


Fig. 4. The graphic representation of global differences.

which value is proportional to the difference; if difference is less than 0 then stand works below the requirements imposed by continuous flow and is potentially subject to the bottleneck effect;

- for the data performed by intervals, the situation is more complicated: if the whole interval is on the positive side (interval A on Fig. 4.), the capacity of the stand is greater than expected by continuous flow; if the whole interval of the difference is on the negative side (interval B), then the capacity is below the requirements of the continuous flow and obviously the stand is the potential bottleneck (the comparison with other differences corresponding to the remaining stands of the system will show, if it is a really bottleneck). The most interesting case is the interval C (Fig. 4.) which has negative and positive parts. In such a case the probability may be considered, with which the stand can work according to the requirements imposed by the continuous flow. This probability can be calculated using geometrical reasons taking into account the positions of the beginning and the end of the interval in relation to zero. If the interval is symmetric in relation to zero, the probability that stand fulfils the requirement of the continuously working system is equal to the probability that it does not fulfil such requirements and it is equal to 0.5. The change of proportion which increases the negative part results in the decreasing the probability that the system works in the continuous flows way and the opposite change results in the increasing this probability. However, it should be emphasized that in a situation when several symmetric intervals appear during the searching of the bottlenecks on the stage of comparing the global differences it doesn't means that the identical impacts of the stands with such differences on the appearance of the effect of the bottleneck in the system takes place. In such case, we consider the measure

Table 15
The possibilities of stands to the be bottlenecks for subgraph 1

stand	1	12	17
P(min)	0	1	0

of uncertainty of the parameters, i.e. the width of the capacity intervals. It can be intuitively stated that the tendency of the appearance of the bottleneck is stronger for the stands, which intervals of differences are wider. The method for the probabilistic estimation of the intervals and fuzzy values equality and inequality degrees is presented in [19]. But in our case it make a sense to consider an additional parameters characterizing such relation depending on relative widths of interval compared. Thus, we have the two-criterial interval comparison problem which may be solved using the method presented in [20].

This method give us the real index P which change from 0 to 1 with rising of inequality degree. It is easy to see that in our case this index is the measure of possibility that the considered stand is a bottleneck. The results of our calculation with using this method are presented below (see Tables 15,16,17).

As we can see in Fig.4 and in the Table 15, there is only one completely negative interval of difference in the subgraph 1.

In the subgraph 2, the probability estimation *P(min)*=0.85 for the ninth stand may be considered as a strong evidence for this stand to be a bottleneck. It is caused by the fact that the negative part of corresponding difference interval is definitely wider than its positive part. The other intervals of global differences in this subgraph demonstrate the greater symmetry in relation to zero .

When analyzing subgraph 3, we can see that probability *P(min)* for the ninth stand is a greatest one. The results we got for the third subgraph are more smooth then those obtained for the subgraph 1 and 2. This may be explained by the effect that in the subgraph 3 there are no complete negative difference intervals.

The final results are shown in Table 18.

Table 18
Bottlenecks of the our example system.

Subgraph	1	2	3
The number of stands which is the bottleneck	12	9	9

The proposed method points at production stands, which are the bottlenecks of individual subgraphs in a given moment, but it also brings additional information about the probability of the minimality of the global differences which can be used to stand ordering with respect to the necessity of correction of their parameters beginning from the bottleneck. Furthermore the results of searching contain information about probability of breakdowns of the particular stand, so the more loaded stand is, the greater risk of breakdown it has. The proposed method may be used in a relatively simple way, when the parameters of the stands are fuzzy num-

Table 16
The possibilities of stands to the be bottlenecks for subgraph 3

stand	2	3	4	7	8	9	13	14	18
P(min)	0.0134	0	0.07	0	0.0042	0.85	0.055	0.006	0

Table 17
The possibilities of stands to the be bottlenecks for subgraph 3

stand	4	5	6	9	10	11	15	16	19
P(min)	0	0	0.06	0.56	0	0.12	0	0.11	0

bers presented by set of $\alpha - cuts$.

4 Summary

The analysis of the bottleneck problem cannot be trivialized. Every optimization of the system should start from the analyzing these components. We have a guarantee then, that it will be really effective. The development of the real system without controlling the bottlenecks may not bring any rational results. Every optimization of the production system should assume the existence of the bottlenecks and consider there impact on the system. The method elaborated allows to control the system with respect to the location of bottleneck (potential as well as real one) when the system parameters are presented by intervals. The method is realized as a software in $C++$. When elaborating the proposed method some new definitions needed for the formalization of algorithm elaborated were introduced. The further work will be oriented on extending the method proposed for modelling of system and for planning the production(the scheduling) in conditions of crisp and fuzzy interval uncertainties.

References

[1] Adams J, Balas E and Zawack D. The Shifting Bottleneck Procedure for Job Shop Scheduling, Management Science 1988;34:391-401.

[2] Ahuja RK, Magnanti TL, Orlin JB. Network flows, Englewood Cliffs, NJ: Prentice-Hall, 1993.

[3] Averbakh I, Berman O, Punnen AP. Constrained matroidal bottleneck problems, Discrete Applied Mathematics 1995;63:201-14.

[4] Berman O,Einav D, Handler G. The constrained bottleneck problem in networks, Operations Research 1990;38:178-81.

[5] Burkard RE, Rendl F. Lexicographic bottleneck problems, Operations Research Letters 1991;10:303-318.

[6] Burkard RE, Sandholzer W. Efficiently solvable special cases of bottleneck traveling salesman problem, Discrete Applied Mathematics 1991;32:61-76.

[7] Camereni PM. The minimax spanning tree problem and some

extensions, Information Processing Letters 1978;7:10-24.

[8] Carlier J. The one-machine sequencing problem, European Journal of Operations Research 1982;11:42-47.

[9] Cristofari M, Caron F, Tronci M. Dynamic scheduling approach in case of machine breakdown and preventive maintenance - 2000 Summer Computer Simulation Conf.

[10] Derigs U, Zimmermann U. An augmenting path method for solving linear bottleneck assignment problems, Computing 1978;19:285-95.

[11] Edmonds J, Fulkerson DR. Bottleneck extrema, Journal of Combinatorial Theory 1970;8:299-306.

[12] Fulkerson DR, Harding GC. Maximizing the minimum source-sink path subject to a budget constraint, Mathematical Programming 1977;13:116-128.

[13] Gabow HN, Tarjan RE. Algorithms for two bottleneck optimization problems, Journal of Algorithms 1988;9:411-7.

[14] Garfinkel R. An improved algorithm for the bottleneck assignment problem, Operations Research 1971;19:1747-1751.

[15] Huh WT, Janakiraman G, Jackson PL, Sawhney N. Minimizing flow time in cyclic schedules for identical jobs with acyclic precedence: the bottleneck lower bound, Operation Research Letters 2003;31:366-374.

[16] Katagiri H, Sakawa M, Ishii H. Fuzzy random bottleneck spanning tree problems using possibility and necessity measures, European Journal of Operations Research 2004;154:88-95.

[17] Papadimitriou CH, Steiglitz K. Combinatorial optimization: algorithms and complexity, Englewood Cliffs, NJ: Prentice-Hall, 1982.

[18] Punnen AP, Nair KPK, Aneja YP. Generalized bottleneck problems, Optimization 1995;35:159-169.

[19] Sewastianow P, Róg P. A Probability Approach to Fuzzy and Crisp Intervals Ordering, Task Quarterly 2003;7(1):147-156.

[20] Sewastianow P, Róg P. Fuzzy Optimization Using Direct Crisp and Fuzzy Interval Comparison, Six International Conference on Neural Networks and Soft Computing, Zakopane, Poland, 2002, p. 43-44.

[21] Sokkalingam PT, Aneja YP. Lexicographic bottleneck combinatorial problems, Operations Research Letters 1998;23:27-33.

[22] Yang C, Zhang J. Inverse maximum capacity problems, OR Spektrum 1998;20:97-100.

[23] Yang C, Zhang J. Two general methods for inverse optimization problems, Applied Mathematics Letters 1999;12:69-72.

[24] Yong-Heng J, Ling W, Yi-Hui J. Bottleneck analysis for network flow model, Advances in Engineering Software 2003;34:641-651.

[25] Zhang J, Ma Z. A network flow method for solving some inverse combinatorial optimization problems, Optimization 1996;37:59-72.

[26] Zhang J, Liu Z. A further study on inverse linear programming problems, Journal of Computational and Applied Mathematics 1999; 106:345-359.

[27] Zhang J, Liu Z, Ma Z. Some reverse location problems, European Journal of Operations Research 2000;124:77-88.

[28] Zhang J, Yang C, Lin Y. A class of bottleneck expansion problems, Computers & Operations Research 2001;505:519-528.