Ivan Brezina
Ján Chabada
Marian Reiff
University of Economics
Faculty of Economic Informatics
Bratislava, Slovakia

Flow Shop Systems as a Part of the Logistics Information System

Summary
The Establishing of an efficient information system to support decision making in a production company is a challenging task, which involves coordination of activities and knowledge of many people in modern technical environment. The organization of each production process into one unit is very demanding and for that reason, production logistics plays an important role in this process. Production logistics is an aggregation of logistics problems and steps essential for preparation and its own development of the production process. Thus logistics information system is a significant part of management information systems. Logistics information system in order to purvey relevant data for decision making should contain methods that objectify decision making, i.e. methods based on quantitative approaches. A Part of the quantitative approaches in the field of production logistics are the FLOW SHOP systems. This article deals with the FlowShop program system (software), which was developed at the Faculty of Economic Informatics, University of Economics in Bratislava in cooperation with Siemens PSE.

Key words
Flow shop system, logistics, information system

1. Logistics information system

The Transformation of economic processes and new direction in the development of information technologies bring a digression on traditional approaches of managing and also changes in the demand on information systems. The distribution of data, processing capacities, control and functions into multiple nodes leads to stronger decentralized organization structures and also brings possibilities of quicker changes and leads also to a broader system. With the transfer of the decision powers into the places of information demand origin, it is possible to reduce the time of decision making significantly, minimize the need of communication and enable to reach some kind of autonomy for decision making, so that only places, which are concerned with the problem, are participating in the decision making process.

A company needs to have appropriate information systems at its disposal, in order to have relevant individual information. It is necessary to gain substantial and authentic data, which are essential for decision making. Spreading out the data to multiple nodes in the systems of data distributed processing, leads to the possibility to run each subsystem of information system of organization on stand alone computers. This causes minimization of data flow, increase of throughput and improves the reliability of the system.

The minimization of production components has significantly decreased prices of all hardware, which is commonly accessible in high quality, but the progress in development of software techniques and tools has a significantly slower rate. For example, in the past the price of hardware was approximately 60 % of the total costs of the information system. In present, in the case of newly build up information systems, these costs represent only approximately 20 %, also basic software costs are approximately the same and more than 50 % are costs associated with the development of software applications.

For that reason, the user shouldn't be bound on contemporary hardware environment and when choosing software, they should decide on the aspect of requirements on an open system and ability to move applications to another environment. In another case, in the future it won't be possible to change old hardware tools to a new one, which will reflect new requirements, or possibly to generate the best proportion between the price and performance, without the additional costs of modification or whole reworking of software applications.

In the real world, there exists a huge number of different information systems, which are made with the aim to fulfill all requests that occur in the organization. There are a big number of usable types of systems for companies. Their selection is often influenced by their price. It's possible to comprehend them according to the type and level of management process, for which they are created.

For responsible decisions making in every area, thus also in the area of logistics activities, every manager has to have necessary data at his disposal. Therefore, it is necessary to build a management

information system (system for support of decision making for mangers). The possible basic scheme of such management information system is represented on the Figure 1. It is impossible to build the mentioned system without building a Base of data (Bank of data), generation of its elements, their storage and also the way of use of this database is important. From the process of building the system for processing the data point of view, the bank of methods has to be built. This Bank of methods contains mainly methods for transformation and analysis of data, and also appropriate software. The Statistical bank, in a similar way as the Bank of data, contains all statistically processed substantial information. The Information bank serves for the exchange of information among the users of the system. A special case of management information system with the emphasis on logistics is the logistics information system. The logistics information system is build as a management information system to support logistics activities.

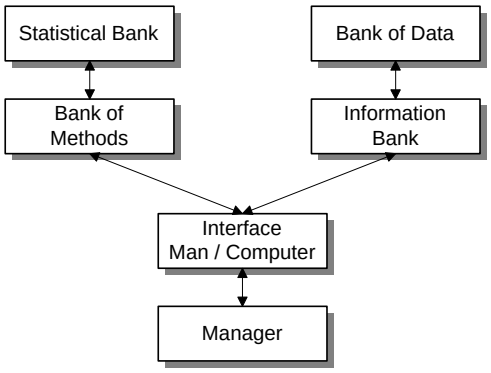

Figure 1. Basic scheme of management information system

It is obvious, that quantitative methods play an unsubstitutable role in the creation of a logistics information system. These methods serve for the optimization of logistics activities.

In a firm, the logistics information system comprises information starting from everyday orders of customers, through information about the logistics network, to supplies for purchasers. The basic source of information for the Bank of Data is the primal collection of data via accounting, material management, financial analyses etc. These data have to be statistically processed, regularly restored and interpreted. All these information are a crucial term for the use of varied quantitative methods, which are used for the optimization of the whole logistic system of the firm.

The integration of technical and economic subsystems is represented by CIM (Computer Integrated Manufacturing). So, it's by computer integrated planning, managing and advocacy. CIM is possible to "understand as a strategic concept, the task of which is to support the main intents of the firm in the area of information and communication technologies."[1] The creation of a CIM system in a concrete firm is especially a time-consuming and financially demanding interdisciplinary process, the aim of which are not short term effects, but formation of long term advantages, mainly in the "flexibility supporting the increase of ability to react on changes on the market, supply promptitude in a sense of quick reaction on the specific demands of the customer, in maturity of the product, namely maturity of higher innovation level, in higher quality of the product."[2]

CIM represents the most complex conception of formation of production systems and their procedural managing. This conception is aimed at the integration of all stages of the production process, beginning with the design of the product, its production, ending with after production services for the customers. According to Scheera, it is possible to depict the continental concept of CIM in Figure 2.

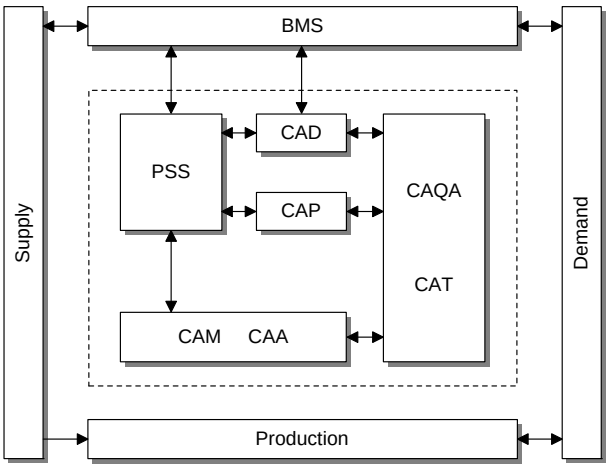

Figure 2. Continental concept of CIM

Meaning of each symbol in the Figure 2:
BMS – Business Management System
CAA – Computer Aided Assembly
CAD – Computer Aided Design
CAM – Computer Aided Manufacturing
CAP – Computer Aided Planning
CAQA – Computer Aided Quality Assurance
CAT – Computer Aided Testing
PPS – Production Planning System

[1] Stern, J.: Logistika a trendy v manažmente výroby. Ekonomický časopis, Vol. 3, 44, 1996 p. 195
[2] Ibid

Recently, also CIL (Computer Integrated Logistics) has become a part of CIM. CIL is logistics supported by computer, mainly warehousing and transportation. In the past, the evolution of integrated technologies supported by computers in the area of production and logistics was run more parallelly. In the past, the liaison between both areas was the output of finished products into physical distribution, identified with logistics. In the present, firms more realize that delimiting leading to integration of logistics and logistics management is in principle everything that is between the customers and their suppliers. Consequently that are also decisions about production content, planning and managing of production, purchasing and supplying (procurement logistics), managing of distribution processes, sale and service for customers. Decisions about research and design of new products are also integrated. For that reason CIL plays an important role in the concept of CIM systems.

2. FLOW SHOP Systems as a part of production logistics.

One part of the logistics, where it is possible to adopt relatively simple, but also complicated quantitative approaches successfully, is the production logistics. The production logistics comprises aggregate logistics tasks and steps essential for preparation and the own development of the production process. It comprises every activity related to material and information flow of raw materials, auxiliary production materials from inbound warehouse to production facilities, from warehouse of semi finished products and purchased parts thought each production and assembly stage to a warehouse of finished products. Possibilities of utilization of existing quantitative approaches in the field of production logistics are depicted in Figure 3.

When optimizing parts of the production process in scope of production logistics, the main goal is to ensure the production process with the in-company transportation of raw materials and semi finished products and their warehousing, insure their optimal allocation to each facility and ensure the organization of production. It is possible to reach a good organization of production by appropriate scheduling of production operations, which are connected to each other in some way. The Scheduling theory (i.e. [Bre]) is aimed at solving time or space scheduling problems of different operations using one or multiple service objects (facilities). The Scheduling models (in literature they are also often named sequence tasks) are connected with determination of the sequence of running diverse operations by one or multiple service objects. These models are mainly used for solving logistics production processes.

The core of scheduling models is to determine the procedure of running diverse operations on one or multiple service objects. The service object (there are used also expressions such as service facility, device, processor) *M* is such a device, which is able to serve one or multiple operations. The set *m* is a set of service objects determined by $\mathrm{M} = \{\mathrm{M}_1, \mathrm{M}_2 \ldots, \mathrm{M}_\mathrm{m}\}$.

Under the expression *operation* (often there are used expressions such as batch, activity), marked as *o*, is known the aggregated basic activity and it is not possible to divide that basic operation into smaller parts (from scheduling theory point of view it is not possible to divide the operation, but some approaches allow to interrupt the operation and after some time period

To continue this operation). The operations can have technical character (engineering, chemical, building technologies etc.), or this expression could mean loading goods from the warehouse, receiving goods to the warehouse, processing of products, service for customers, etc.

The order of *n* operation $\{o_1, o_2, \ldots, o_n\}$ is called a *task*, marked as *J*. The core of scheduling models is to assign *n* operations to *m* service objects. The assigning of the *i* operation to the *j* service object o_{ij} is based on the time t_{ij}. It is the time of running the operation *j* on service object *i*. It is assumed, that each operation is realized on a different, not interchangeable service object. If the previous operation is not finished, the next one won't start and waits till the end of the previous one. So it is possible to start running the operation *k* after the previous operation *i* ends (the operation *i* is previous to *k*). It is possible to reach the minimum time needed for the completion of all connected operations, when the operations are arranged appropriately.

In the majority of scheduling models there are applied two basic conditions:

1. It is possible to run only one operation at the same time on the single service object.
2. At the same time, it is impossible to run more than one operation of the same task on multiple service objects.

The core of the simplest scheduling models is scheduling of operation on a single object. More complicated models are for example *scheduling of flow organized systems*. The core of these models is scheduling of operations, the order of which is strongly determined to particular objects (there are elaborated simple heuristics for two, tree and more ob-

jects). In case when we need to schedule operations to service objects which are located according to category and relation, we are talking about scheduling of phase organized models.

In the case of multiple service objects (there is the assumption that the objects are arranged in a row - serial), when the order of the running operations is important (the same order of realization of the particular operations), we are talking about FLOW SHOP systems. When the service objects are in serial order and it is possible to realize an arbitrary order of crossing of each operation (but within complex tasks that order is usually equal), we are talking about JOB SHOP systems. If there is given a set of operations without categorization into joint tasks (the operations don't have a given order in some task) and set of service objects, (each operation is assigned to one service object and the order of realization of operations is arbitrary) we are talking about OPEN SHOP systems.

3. Algorithms for solving the FLOW SHOP systems

FLOW SHOP systems are systems for scheduling operations on serial ordered service objects, with the same order of operations and the goal is to optimize the value of object function *f*. It means, that the operations o_{1j} will be processed on the service object M_1, the operations o_{2j} on the service object M_2, generally the operation j ($j = 1, 2, ..., n$) o_{ij} will be processed on the service object i ($i = 1, 2, ..., m$) with the goal to minimize the value of the chosen optimization criterion.

Generally, for the FLOW SHOP systems it is possible to accept the assumptions below:

- Each task J is aggregation of operations o_{ij}, and these operations are processed on the service objects in the same order.
- The time t_{ij} needed for processing all the operations o_{ij} is known.
- The service objects are to disposal every time they are needed.
- Individual operations are not interrupted.

For the solution of FLOW SHOP systems except other approaches (like Branch and bound method used to solve integer programming problems) different heuristics are used, from which are the best known:

- Johnson's algorithms,
- Palmer's heuristic,
- Grupt's heuristic,
- Campbell's, Dudek's and Smith's heuristic.

Algorithm description of the particular heuristics is placed in the Appendix.

From the character of the above-mentioned problems, it is clear, that these problems are solvable also with the use of combinatory methods. But their use is considerably limited, because their demandingness for solving time is considerably higher (one combinatory algorithm is in the Appendix). Artificial intelligence is a new direction for solving the above-mentioned problems. Genetic algorithms are part of artificial intelligence. In present, their use is possible thanks to progress in IT, because their demandingness for solving time is also much higher compared to an arbitrary heuristics. The computation with the use of genetic algorithms is based on the well known general genetic algorithm.

4. FLOWSHOP program

The program FlowShop is created in a free development environment Eclipse and is written in the Java language. Thanks to the used technology, this program runs not only under MS Windows operation system, but also under an arbitrary operation system with installed Java Runtime Environment (it is recommended to use version 1.4.2 or higher), for example namely Linux or Sun Solaris.

This program enables to create tasks easily (the maximum number of service objects is 100 and the maximum number of operations is also 100, this number should be sufficient for solving real cases), it also enables the editing of data, their saving on a memory medium, repeated loading data to the program and printing of results with additional information. The FlowShop program enables to solve the FLOW SHOP systems problems via heuristics methods (Palmer's heuristic, Grupt's heuristic, Campbell's, Dudek's and Smith's heuristic, heuristic of minimum downtimes sum), via combinatory method (with the possibility of distributed computing, where the computation runs on multiple computers at the same time), this combinatory method gives an optimal solution of the task, and also computation via Genetic algorithms (Figure 4).

Besides the final result, the program enables the user to get also additional information in tables, for example information about the timetable or downtimes on particular devices. It also offers a graphical description of the solution via Gantt diagrams (Figure 5). Particular information on the computed solutions is logically ordered and depicted in the bookmarks. It is important to

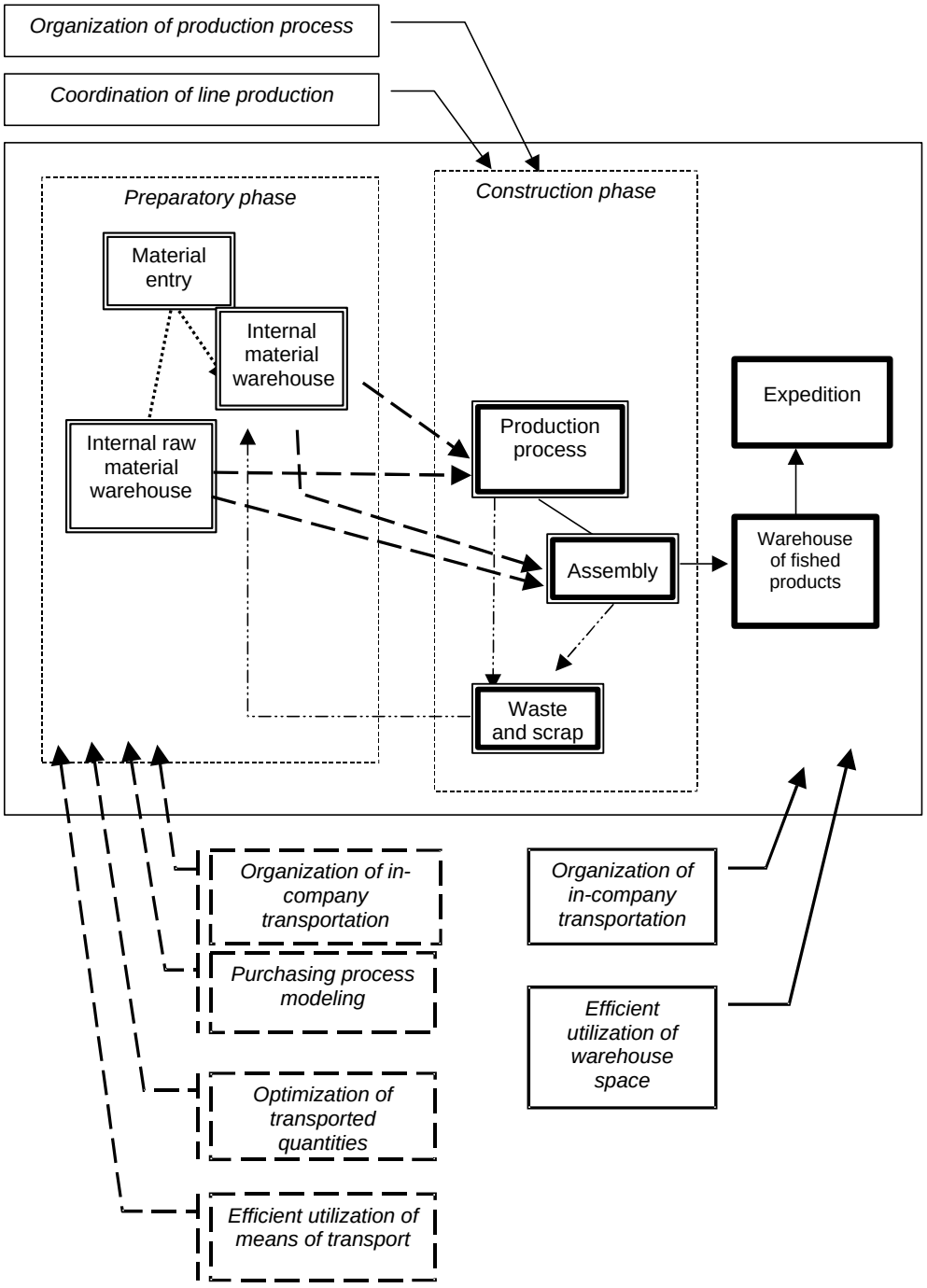

Figure 3. Utilization of quantitative approaches in production logistics

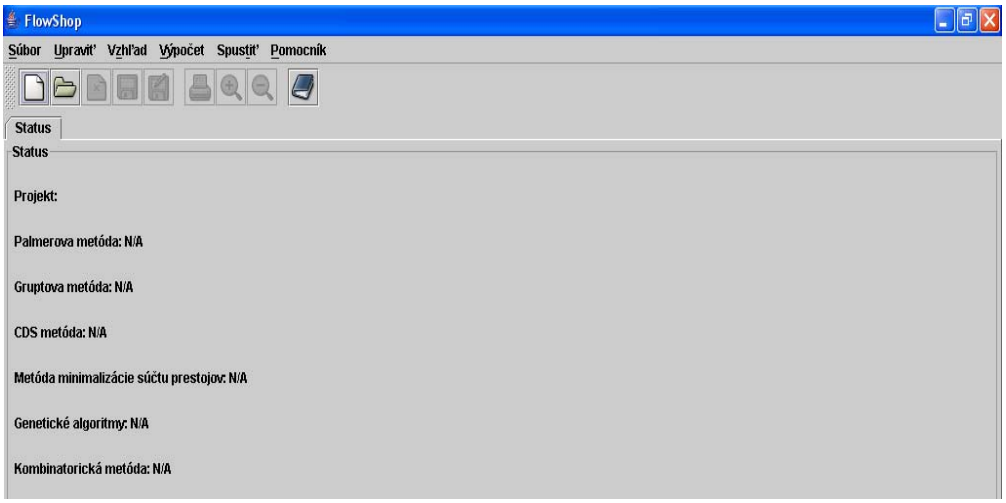

Figure 4. Initial screen of the FlowShop program

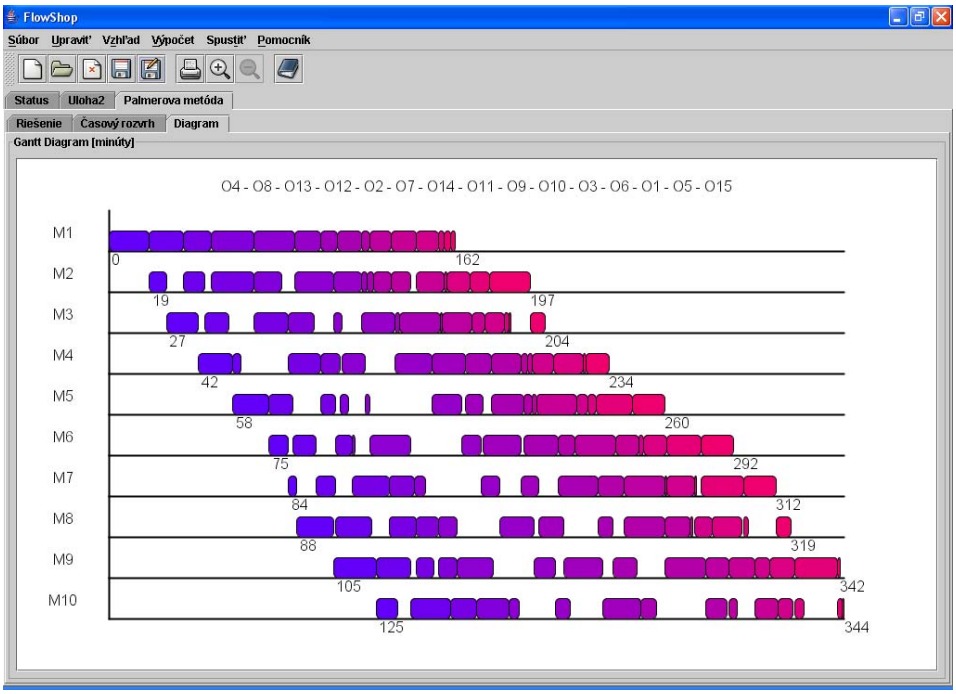

Figure 5. Output depicted via Gantt diagram

remind, that the final solution doesn't have to be optimal, when the solution is computed via heuristics (in most of the cases it is not).

Because the computation of optimal solution via combinatory method is very demanding, in the program there is used distributed computing, based on the architecture client/server. The main technology feature of the created program solution is the usage of the technology Java RMI (Remote Method Invocation), which ensures distant communication between server and client via network.

The Main idea of using distributed computing, when computing FLOW SHOP systems via combinatory method, is to spread computation between multiple computers in the network in order to speed up computation, thus to distribute the performance. The reason for the implementation of distributed computing when the combinatory method is used, is the fact, that during the computation process it is necessary to solve a huge amount of mathematical operations. As the demandingness of computation rises exponentially with the number of operations (the number of every possible solution is $n!$, where n is the number of operations), it was necessary to find a way, for the computation of an optimal solution also for bigger problems, which is the distributed computing among multiple computers in the network.

The architecture client/server is based on dividing the work between the client and the server. The client is a software process, which requires service from another software process. In reverse, the server is a software process, which serves to the client as a response to placed requests of the client, independently of the hardware platform. It is rather common to call the computer, on which client functions run, client and in reverse, the computer running server software application, server (Figure 6). It is possible to talk about a concrete computer, which works as folder or printing server, because it has got hardware or software equipment, but from the application's point of view, this shouldn't be relevant. The processor of an arbitrary computer can meet both client or server functions (Figure 6). From above it is clear, that every computer can work as a server or as a client, it can even work as a server and as a client at the same time.

From the FlowShop program point of view, the server is considered as a process, which registers the task on the Name server and waits for requests of the clients (Figure 7). The main task of the server (Project server) is the distribution of the task between particular computers on the network and the managing of the computation. The Project server divides the task according to its complication into $x = (n! / 10!)$ independent parts (batches), where n is the number of operations in the task. If $n <= 10$, than $x = 1$, so the task consists only of one single part. So the maximum number of clients, participating in distributed computing of the concrete task is equal to the total number of independent parts x, to which this task is divided.

After accepting a request from the client, as a response the project server is sending a partial task for computation to the client and waits for confirmation after finishing computation together with the partial result. This is the fundamental service of the project server and is served according to the client's requests. The project server ensures that each partial computation task is sent to the client only once. When the connection between the client and the server falls down (for example due to problems with the network), the project server ensures the consistence of the distributed computation through repeated sending of the not computed (lost) partial computation task to another client. On the side of server also runs the optimization of the computation process. For each separate part of the task, before it is send to the client, the project server runs a recount about the best possible reached result for a given part of the task, with the assumption, that the sum of downtimes in that part of the task is equal to zero. In case, when the reached partial result is worse than the best reached total result for the moment, this sequence is skipped. In the reverse situation, the sequence is sent to the client to recount. The server then continues similarly with other independent tasks. This way, the computation process is dynamically optimized and is also faster in time.

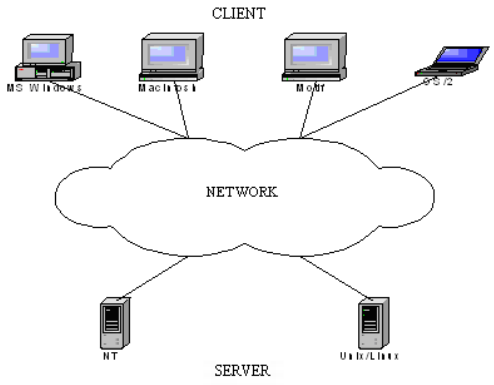

Figure 6. Concept of architecture client/server

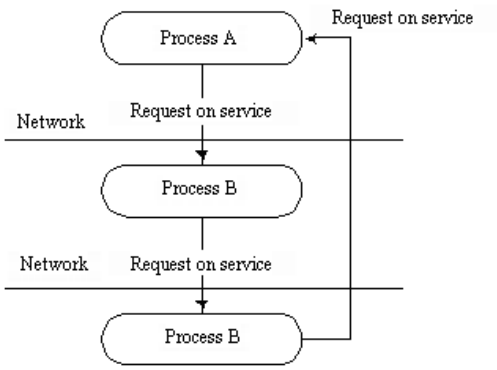

Figure 7. Architecture client/server

From FlowShop point of view the client is a process, the goal of which is to look for a particular task on the name server. This server will return a ref on the process of the project server, which ensures the managing of the computation of the selected task (Figure 8). Consecutively, the client will create thanks to the received ref from the project server via dialing Java RMI a distant object, on which it can call the project server methods. So the client can use the service of the project server, which is offered to it by that server. The basic and most important service offered by the project server according to the requests of the clients is provision of a subsequent not yet computed stand alone part of task for computation. After consequent computing, the client informs the project server about the end of computation together with the reached partial result and asks for the next stand alone task for computation. This process is repeated, till every independent part of task is computed.

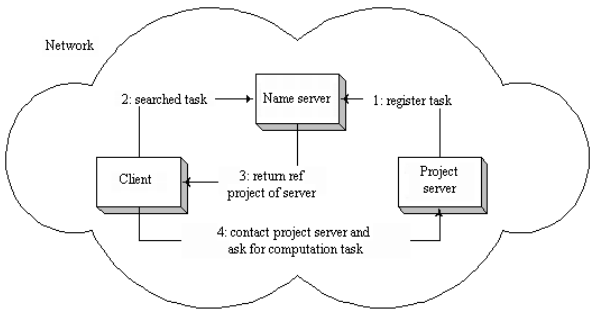

Figure 8. Communication process between client and server when computing task

According to the above-mentioned, it is clear, that when computing task via combinatory method, it is necessary to create a project server first, which will subsequently register the task on the name server.

5. Comparison of effectiveness of particular methods

When solving FLOW SHOP systems problems, for their computation it is possible to choose among different methods. These methods can be differentiated according to the time spent for their computation and the reached solution. Generally, heuristics methods are giving good solution in relatively short time and when computing large scale problems, it's possible to reach a solution in real time. On the other hand, combinatory methods are giving an optimal solution, but the time demandingness for computation is considerably higher compared to heuristics methods. The disadvantage is that when solving large scale problems, it is nearly impossible to get an optimal solution in real time, not even using distibuted computing with many computers. The next possibility of solving computation problems via FlowShop program is via genetic algorithms. With use of a genetic algorithm, it is possible to get a very good solution in relatively short time. As early as some hundreds of genetic populations of chromosomes are generated, in most of the instances, this method gets a better solution then an arbitrary heuristic and during the whole time of computation it is reaching an optimal solution. The quality of the solution reached in the end via a genetic algorithm, mostly depends on the time of computation, ruled by the user's decision.

A computer with processor Intel Pentium 4 630, 3.0 GHz with standard options of Java Runtime Environment version 1.4.2 was used for computation. Due to better comparison of the abovementioned methods, distributed computing wasn't used in case of combinatory method. The corresponding computational statistics for this problem are summarized in Tables 1 – 5.

According to the figures in the tables above it is possible to say, that when solving FLOW SHOP systems via FlowShop program it is suitable to use combinatory method for solution, because it gives an optimal solution. This statement is valid only in instances with a small number of objects and operations. According to the performance of the computer and the number of computers associated in distributed computing it is reasonable to use combinatory method till the number of operation is not larger than 15. Otherwise the computation time is too large. In these instances it is better to use a genetic algorithm or one of the heuristic for computation, where the computation time is considerably smaller compared to a combinatory method. From these methods the best solution is

Table 1.

2 objects, 3 operations	Reached solution (in time units)	Time needed for solution (in seconds)
Palmer's heuristic	24	< 1
Grupt's heuristic	24	< 1
Campbell's, Dudek's Smith's heuristic	23	< 1
Heuristic of minimum downtimes sum	23	< 1
Combinatory method	23	< 1
Genetic algorithm	23	< 1

Table 2.

(4 objects, 6 operations)	Reached solution (in time units)	Time needed for solution (in seconds)
Palmer's heuristic	97	< 1
Grupt's heuristic	86	< 1
Campbell's, Dudek's Smith's heuristic	86	< 1
Heuristic of minimum downtimes sum	86	< 1
Combinatory method	86	< 1
Genetic algorithm	86	< 1

Table 3.

3 objects, 11 operations	Reached solution (in time units)	Time needed for solution (in seconds)
Palmer's heuristic	121	< 1
Grupt's heuristic	110	< 1
Campbell's, Dudek's Smith's heuristic	110	< 1
Heuristic of minimum downtimes sum	116	< 1
Combinatory method	110	~ 74
Genetic algorithm	110	< 1

Table 4.

6 objects, 14 operations	Reached solution (in time units)	Time needed for solution (in seconds)
Palmer's heuristic	218	< 1
Grupt's heuristic	211	< 1
Campbell's, Dudek's Smith's heuristic	197	< 1
Heuristic of minimum downtimes sum	203	< 1
Combinatory method	-	estimation ~ 5 days
Genetic algorithm	188	~ 2

Table 5.

10 objects, 15 operations	Reached solution (in time units)	Time needed for solution (in seconds)
Palmer's heuristic	344	< 1
Grupt's heuristic	305	< 1
Campbell's, Dudek's Smith's heuristic	288	< 1
Heuristic of minimum downtimes sum	279	< 1
Combinatory method	-	estimation ~ 100 days
Genetic algorithm	264	~ 10

given by a genetic algorithm, and from heuristics methods in most instances by Campbell's, Dudek's and Smith's heuristic, then heuristic of minimum downtimes sum, followed by Grupt's heuristic and Palmer's heuristic. This affirmation is confirmed also by the above-mentioned results in the tables.

Appendix

1. Algorithm of Palmer's heuristic:

1. For $j = 1, 2, ..., n$ are computed coefficients:

$$s_j = |m-1|.t_{mj} + |m-3|.t_{m-1j} + |m-5|.t_{m-2j} + ... + |m-(2m-1)|.t_{1j}$$

2. Then, as a suboptimal schedule is picked that schedule, for which stands:

$$s_1 \geq s_2 \geq s_3 \geq ... \geq s_n.$$

2. Algorithm of Grupt's heuristic:

1. For $j = 1, 2, ..., n$ are computed coefficients:

$$s_j = \frac{e_j}{\min\limits_{1\leq k\leq m-1}\{t_{kj} + t_{k+1j}\}}, \text{ where } e_j = 1,$$

if $t_{1j} < t_{mj}$, otherwise $e_j = -1$

2. Then, as a suboptimal schedule is picked that schedule, for which stands:

$$s_1 \geq s_2 \geq s_3 \geq ... \geq s_n.$$

3. Algorithm of Campbell's, Dudek's and Smith's heuristic:

1. For $k = 1, 2, ..., m-1$ are computed:

$$t^k_{1j} = \sum_{i=1}^{k} t_{ij}, \qquad t^k_{2j} = \sum_{i=m-k+1}^{m} t_{ij}$$

2. With Johnson's algorithm for two to each other following up objects are computed $m-1$ permuting schedules entirely, based on t^k_{1j} a t^k_{2j}.
3. Then, as a suboptimal schedule (from $m-1$ schedules) is picked that schedule, which minimizes the time needed for completion of each operation on the last service object.

3b. Johnson's algorithm for two to each other following up objects:

1. Let I be a set of not realized operations on two each other following up objects, L^1 a set of placed operations on the service object M_1 and L^2 a set of placed objects on the service object M_2.
2. Then the shortest time needed for processing of the particular operations, i.e. $\min\limits_{i\in I}(t_{1j}, t_{2j})$ is found. If several operations have the same shortest time, an arbitrary one of them is picked. Let the minimum for k-th operation be o_k.
3. If the minimum time of processing operation o_k requires service object M_1 (the minimum value is t_{1k}), the operation o_k is placed to the set L^1. If it requires service object M_2 (the minimum value is t_{2k}), the operation o_k is placed to the set L^2.
4. Let $I = I - \{o_k\}$. If $I = \emptyset$, step 5 follows, otherwise, return to step 2 (the whole computing process is repeated for n -1 working operations).
5. The schedule is made, which comprises ordered operations from the set L^1 in a given order and from the set L^2 in an inverse order, so the first operation from L^1 is realized as the first one and the first operation from L^2 is realized as the last one (the second operation from the set L^1 is realized as the second one, the second operation from L^2 is realized as the last but one etc.).

4. Algorithm for heuristic of minimum downtimes sum:

1. For $k = 1, 2, ..., n$ permuting schedules P_k, let as first be the operation of the given permuting schedule operation o_{ik}.
2. Let $O_k = \{o_{i1}, o_{i2}, o_{i3}, ..., o_{in}\} - \{o_{ik}\}$ be the set of not placed operations for each permuting schedule P_k.
3. As first are computed the soonest possible time of starting, the time of finishing and the value of downtimes for each not placed operation from the set O_k on every service object for each one permuting schedule P_k.
4. For each not placed operation from the set O_k is computed the sum of downtimes from every service object for each one permuting schedule P_k.
5. For the following operation of the given permuting schedule Pk that operation from the set O_k is selected, which has reached the smallest sum of downtimes for the given service object.
6. From the set of not placed operations O_k, that operation is picked, which is placed into the permuting schedule P_k.
7. Follow step 3. until the set of not placed operations is not empty.
8. Then as suboptimal schedule (from n permuting schedules) that schedule is selected, which minimizes the time needed for the

completion of all operations on the last service object.

5. Algorithm for the combinatory method of solution:

1. Let I be the set of operations, representing the given permuting schedule, let i be the index of the set I ($i = 1, 2, ..., n$) and let s be the variable, which represents such index i of the permuting schedule, on the position of which there are changed operations. Into the set I the initial permuting schedule is inserted, consisting of operations $o_1, o_2, o_3, ..., o_n$ ordered according to the index from the lowest to the highest; I = $\{o_1, o_2, o_3, ..., o_n\}$. Let the initial value of s be $s = 0$.
2. The element from the set I is found, for which stands $i = s$. In case, that variable s is not equal to any index from the set I (if $s = 0$), step 4 will follow. Otherwise the operation on the given place s will be changed with the operation with the nearest higher index from the set I, for which stands $i > s$.
3. Subsequently every operation in the set I, for which stands $i > s$ is ordered in the set, so that the indexes of operations are ordered from the lowest to the highest.
4. The set I represents one of the possible permuting schedules in the given condition.
5. Let variable s have the maximum value i, for which stands, that the operation, which is situated on the i-th position of the set I has got lower index than an arbitrary operation situated on the position in the set I with index higher than i. If such a situation occurs, that no such i exists, which stands the above-mentioned condition, than step 6. will follow, otherwise the next possible permuting schedule is searched and the computing procedure is returned to the step 2.
6. Finishing the algorithm when every possible permuting schedule is taken, and the number of possible permuting schedules is $n!$.

Doc. Ing. **Ivan Brezina**, CSc. is employed on Faculty of Business Informatics, University of Economics at the Department of operations research and econometrics since 1985. He is interested in quantitative methods used in the field of logistics information systems construction, operations research theory development, and methodology of officiating business sectors density creation.

Ing. **Marian Reiff** is at the Department of operations research and econometrics since 2002, he is interested in supply chain models.

Ing. **Ján Chabada** is graduate in Faculty of Business Informatics and nowadays works at Siemens Company.