Appl. Math. and Comp. Sci., 1993, vol.3, No.2, 357–371

PROBLEM–ORIENTED COMPUTER SYSTEM WITH LOAD CONTROL TO COMPLEX COMPUTATIONAL WORKS†

IWONA POŹNIAK*

Balancing the workload among processors is of fundamental importance to the efficient utilization of a local computer network. In the paper some ideas from control area are applied to improve load–balancing for problem–oriented complex works which are composed of sequences of dependent tasks. The flexible system with load control is recommended. The flexibility gives opportunities to perform the work by using one of the modes of performing available. The designed modes differ by allocation of executors in processors, initial tasks assignment, policies to interprocessor exchange of tasks and tasks scheduling algorithms. The control program is responsible for choosing the mode to maximize processors balancing for given work. An application of the system is to aid the diagnostic process. The results of experiments in that case show a significant increase of efficiency. Moreover, the proposed ideas to design such a system may be adopted to a broader class of computational works which may be characterized by tree–structured parallel processing.

1. Introduction

A computer system with distributed databases in local–area–network provides an opportunity for parallel processing which increases effectiveness of performing complex work. Several ideas how to improve the effectiveness have been proposed in literature, mostly for general–purpose systems.

Balancing the workload among processors may provide higher system throughput, to minimize the average response time and to reduce processor idle time (Hwang, 1982). The general problem of load–balancing to works composed of tasks (program modules) is complex involving tasks allocation, tasks scheduling, tasks assignment, dynamic interprocessor communication, etc. Authors of papers usually concentrate on selected problems from this area. Heuristic model of tasks assignment scheduling in distributed systems has been proposed by (Efe, 1982). The adaptive load–balancing with one–time assignment of works to processors (Stankovic, 1984) and the adaptive load–sharing in homogeneous distributed systems

† This work was partially supported by the State Committee for Scientific Research under grant No.307629101.

* Institute of Control and Systems Engineering, Technical University of Wrocław, Wrocław, Poland

(Eager, 1986) have been considered. For queuing network model a distributed drafting algorithm for load–balancing has been shown in (Ni, 1985). In (Chu, 1987) a method is suggested for optimal module allocation with an objective function that includes the intermodule communication and accumulative execution time of each module. The problem of optimal assigning the modules of single tree–structured parallel program over the processors is considered by Bokhari (1988). The general concept of modelling and analysis of time–cost behavior of parallel computations is presented in (Qin, 1991).

In this paper, the load–balancing problem is considered in a complex way but for problem–oriented computer system. The class of complex computational works under consideration is described by the following facts: work is a set of sequences of dependent tasks, all sequences are ready to be performed at the same time, there are several distinct categories of tasks. Executors to them may be located in different processors, the execution of a given sequence may be broken after finishing one of the tasks belonging to the sequence. In that case, the given work may be characterized by quantitative parameters including the sequences composition and relative execution times to tasks (Pozniak, 1992). The process of performing the work may be modelled as a control plant where the work parameters are measured inputs, an output is the completion time to the whole work and the control variables are parameters of the mode of performing. The mode of performing is defined by allocating the executors in processors (processing structure), initial tasks assignment to processors, policies to interprocessor exchange of information and tasks scheduling algorithm. The number of the structure (processing structure with associated tasks scheduling algorithm) and task allocation ratio (the measure of tasks assignment) are taken as parameters describing the mode of performing. The load–balancing problem may be equivalent to load–control problem i.e. finding parameters of the mode of performing for a given work such that maximizes the processors balancing coefficient defined as the ratio between accumulated active time needed to perform the work and the completion time.

The load control problem is considered in this paper. On the basis of analytical and experimental results obtained an algorithm to load–control is constructed. The recommended computer system contains a control program and possibilities to perform complex works by using several modes what ensures proper efficiency. The class of works under consideration is oriented to aid the diagnostic process. An application shown in the paper is diagnosis to the group of patients being made at the same time by using multistage recognition method (Kurzyński, 1987). This method may be interpreted as step by step narrowing the feasible set of diseases to which any single patient is to be classified. The procedure is similar for each patient in the group. It requires tree–structured parallel computations and leads to execution of different sequences of computer programs (tasks). Moreover, the ideas to design such a system may be adopted to a wider class of practical problems if the sequences of computational dependent tasks may be performed in parallel.

2. Problem Description

Computer system is to be designed to multiple–performance of complex works of the same nature. A single work consists of sequences of tasks. The order of executing tasks Ti $(i = 1,2,3,4)$ from the same sequences is determined: $T1 < T2$, $T1 < T3$, $T3 < T4$. Any sequence belongs to one of the three categories denoted by $C1$, $C2$ and $C3$, where:

$$C1 = (T1, T2), \quad C2 = (T1, T3), \quad C3 = (T1, T3, T4)$$

A single task consists in making an appropriate computational algorithm. To realize any computational algorithm the data stored in database should be used several times. A single data may be used to more than one algorithm. Parallel processing is possible because all sequences of tasks are ready to be performed at the same time and any sequence may be broken after ending any task belonging to the sequence and may be continued in the other processor. It is assumed, that for performing the considered works the following tools are available: the local area network, problem oriented database RDDML (Borzemski *et al.*, 1988) and programs to realize the computational algorithms written in C–language. The important question is how to use these tools in the most effective way. The general idea is to design a flexible computer system capable of several ways of processing complex works.

The proposed approach to design such a system consists in the following steps: (1) determining the set of designed processing structures which differ by allocation of executors to tasks and policies to interprocessor exchange of tasks, (2) finding tasks scheduling algorithms (for each structure separately) which contain rules to tasks ordering and priorities to executing tasks of different categories, (3) defining quantitative variables to describe any work and each mode of performing as well as defining criterion of efficiency that allows to model the process of performing as a control plant and to state the open–loop control problem, (4) comparing modes of performing available on the basis of experimental results taken after preparing elements of the computer system (programs to realize each mode of performing), (5) finding the control algorithm allowing to choose the most efficient mode of performing for a given work, (6) preparing the control program to manage performing the complex work and finally, completing the flexible computer system as a whole. In this section, steps (1)–(3) are described in detail.

As a result of the detailed analysis described by Poźniak (1992) three processing structures have been recommended. These structures denoted by $S1$, $S2$ and $S3$ are shown in Figure 1. In the figure, Oi symbolizes the executor to the task Ti. Processors are denoted by $P1$ and $P2$. Databases are organized in the same manners for all structures. The amount of information which must be sent to the other processor is reduced to the consecutive number of the sequence in the work which makes the time period needed for sending this information much shorter than the time needed to execute any task. Both processors may work in parallel.

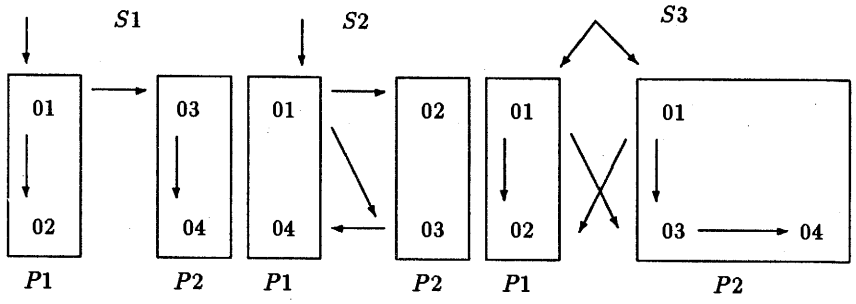

Fig. 1. Three system structures considered.

In structures $S1$ and $S2$ the processors are in Master – Slave regime. Processor Master ($P1$) begins performing any sequence in work and is responsible for executing all tasks $T1$ in the work. Processor Slave ($P2$) is activated by Master to continue parts of all tasks if $S2$ is used, or to continue sequences $C2$ and $C3$ if $S1$ is used. When $S2$ is used, sequences of $C3$ are sent back to $P1$ to execute $T4$. In structure $S3$ both processors are on equal conditions. In the beginning, they perform assigned parts of work, by executing $T1$. Next, they exchange information and continue to execute proper tasks. First, they execute tasks to sequences assigned, then tasks to sequences received from other processor. A proper tasks scheduling algorithm is associated with any structure. Such algorithms were found as a result of theoretical analysis. The objective was to reduce idle times expected in processors.

Algorithm to $S1$:

in $P1$ – $C3$ before $C2$ before $C1$:
– $T2$ after last $T1$;
in $P2$ – $T4$ before next $T3$.

Algorithm to $S2$:

in $P1$ – $C3$ before other sequences;
– $T3$ of $C3$ before $T3$ of $C2$;
– $T4$ after last $T1$
in $P2$ – $C3$ before other sequences.

Algorithm to $S3$:

in $P1$ – all possible $C1$ are assigned to $P1$;
– $T2$ from $C1$ assigned are executed before $T2$ from $C1$ is received;
in $P2$ – all possible $C3$ are assigned to $P2$;
– $T3$ and $T4$ from $C2$ and $C3$ assigned are executed before $T3$ and $T4$ from $C2$ and $C3$ are received from $P1$.

Following the rules ensures minimizing idle times in processors but unfortunately idle times cannot be eliminated at all. They may occur on $P2$ (if $S1$ is used), on both $P1$ and $P2$ (if $S2$ is used) and on $P1$ or $P2$ (when $S3$ is used). When structure $S3$ is used, the whole work should be divided into parts assigned to both processors. We define the tasks allocation ratio D, where

$$D = N(P1)/N \tag{1}$$

and $N(P1)$ informs us how many sequences from the whole work (which is composed of N sequences) are directed to processor $P1$. For structures $S1$ and $S2$ the ratio D is always equal to 1. The pair Sm $(m = 1,2,3)$ and D describes the mode of performing.

The problem under consideration consists in finding the best mode of performing for a given work. In order to state the problem formally, we introduce some notions which make it possible to describe any work and to define the criterion for comparing different modes of performing. The notions are:

$t(Oi)$ – the execution time to the task Ti,

$t_n(Pr)$ – the accumulated idle time of processor Pr, $(r = 1,2)$,

$t_a(Pr)$ – the accumulated active time of processor Pr,

$t_f(Pr)$ – the time–period needed by Pr to realize its piece of the whole work, where:

$$t_f(Pr) = t_a(Pr) + t_n(Pr) \tag{2}$$

t_c – the completion time to the whole work in the system , where:

$$t_c = \max\{t_f(P1), t_f(P2)\} \tag{3}$$

B – the processors balancing coefficient, where:

$$B = [t_a(P1) + t_a(P2)]/t_c \tag{4}$$

The value of B shows how much faster is two–processors performance over one–processor performance. When the processors are balanced perfectly, the value of B is equal to 2.0.

V – the sequences composition, a vector containing two values:

$$V = [V(C1), V(C3)], \quad V(Cj) = N(Cj)/N \tag{5}$$

The value $V(Cj)$ is a relative number of sequences of the Cj category $(j = 1,2,3)$ in the whole work, $N(Cj)$ is the number of the j–th category of sequences in the work.

K – the execution times relations, where K is a vector containing three values:

$$K = [K2, K3, K4], \quad Ki = t(Oi)/t(O1) \tag{6}$$

The value Ki is the ratio between the execution time to the task Ti $(i = 2,3,4)$ and the execution time to the task $T1$.

Any work may be described by parameters V and K. The coefficient B (4) is taken, as the criterion to compare different modes of performing. The value of B depends on work parameters as well as on the mode of performing:

$$B = F(V, K, Sm, D) \tag{7}$$

The problem may be formulated as follows: *for a given work which is characterized by V and K, find the pair (Sm, D) at which the coefficient B reaches the maximum*:

$$F[(Sm, D)^*, V, K] = \max F[(Sm, D), V, K] \tag{8}$$

The $(Sm, D)^*$ pair determines the best mode of performing. The real problem is to construct a control algorithm solving problem (8).

3. Comparison Between Structures

In order to illustrate the performance of the work by using different structures a simple example is considered. In the example the work is composed of 5 sequences:

$$(z_1, z_2, z_3, z_4, z_5) = (C1, C1, C3, C1, C1) \tag{9}$$

The execution time of task $T2$ is twice as that of $T1$, it is also four times longer than the execution time of $T3$ and two–times less than execution time of $T4$. Using (5) and (6) the work is characterized by the pair V and K, where

$$V = [0.8, 0.2] \quad \text{and} \quad K = [2.0, 0.5, 4.0].$$

The results of experiments are shown in Figure 2. The notation $t_p(Oi)$ stands for the execution time needed to execute Ti for the p–th sequence in work $(p = 1,2,3,4,5)$ where p is the index associated with z in (9). Idle times are symbolized by empty time–intervals. Four cases are presented: (a) structure $S1$, (b) structure $S2$, (c) structure $S3$ with $D = 0.4$, (d) structure $S3$ with $D = 0.2$. The exchange of information is shown by using arrows directed to processors. The number in brackets listed over the arrow is equal to the p–th number of sequence in the work. The values of parameters (2), (3) and (4) introduced in the previous section are shown in Table 1. The greatest value of B equal to 1.94 has been reached for case (b). Hence, we may conclude that for the example considered here, the best structure is $S2$. In the example, cases (c) and (d) are characterized by the same values of B. It is an unusual situation, usually for different D we reach different processors balancing coefficients. Another observation which may

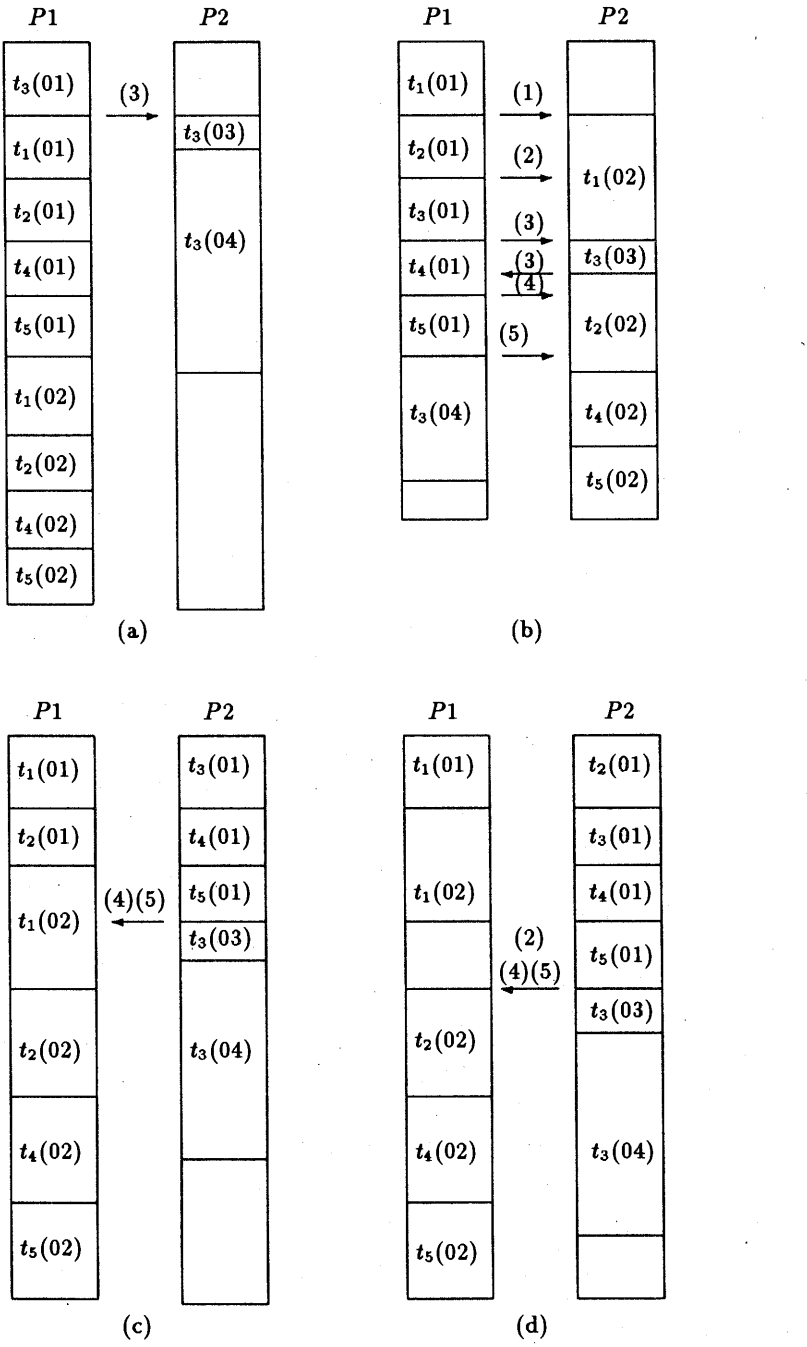

Fig. 2. Time characteristics of work. Performing using different structures.

be mode is that structure $S3$ is not the best, even when we use the best tasks allocation ratio D equal to 0.2 because such a way of dividing caused idle time in $P1$. In general, it may be concluded that using different structures gives different results. As in the example, the results may differ significantly. Comparing (a) with (b) we observe almost 50% difference in efficiency.

Tabl. 1. Characteristics of different structures.

	(a)	(b)	(c)	(d)
$t_f(P1)$	13.0	9.0	10.0	10.0
$t_f(P2)$	5.5	9.5	7.5	8.5
$t_n(P1)$	0.0	0.5	0.0	1.0
$t_n(P2)$	8.5	1.0	2.5	1.5
t_c	13.0	9.5	10.0	10.0
B	1.35	1.84	1.75	1.75

4. Properties of the Modes of Performance

In order to observe the properties of the structures, several experiments have been carried out for the following situation:

I. Any sequence in work is the realization of a three–stage recognition process to single object (patient) which is characterized by values of 17 features creating a set of features. Different subsets of features are taken into consideration at different stages. At the first stage the tasks of $T1$ are executed, at the second stage the tasks of $T2$ or $T3$ and at the third stage the tasks of $T4$.
For example, in case 3: the task of $T1$ means the realization of recognition algorithm (NN) based on 11 features, task of $T2$ is the realization of (NN) with 6 features, task of $T3$ is (NN) with 5 features and task of $T4$ with one feature. The different subsets of features create different execution times to tasks. At all stages the learning sequences are of the same length. The values of features to objects to be classified and the values of features in learning sequences are data from the proper tables located in databases (Poźniak, 1990).

II. Each single experiment consists in performing the work composed of 8 sequences (the recognition to the set of 8 single objects) with given V and K by using each available mode of performing. In the situation considered here we have 11 different modes (Sm, D) including: $(S1, 1)$, $(S2, 1)$, $(S3, 0)$, $(S3, 0.125)$, $(S3, 0.25)$, $(S3, 0.375)$, $(S3, 0.5)$, $(S3, 0.625)$, $(S3, 0.75)$, $(S3, 0.875)$, $(S3, 1)$.

III. The single experiments are repeated 45 times for each different work composition possible, when $N = 8$, including $V = (0.0, 0.0)$, $V = (0.0, 0.125)$, ..., $V = (0.125, 0.875)$, $V = (0.125, 0.750)$, ..., etc. The values of coefficients B for all different modes, all different V and fixed K are obtained as the results of the series of experiments.

IV. The series of experiments are repeated for several different K.

In Tables 2,3,4 the results of experiments are shown in a shortened form. Three different sets of K are considered (cases 1, 2, and 3). In each case values of processors balancing coefficients are given for three modes $((S1,1)$, $(S2,1)$ and $(S3, D^*)$ where D^* is the value of D at which $S3$ can have the greatest B. The values of processors balancing coefficients $\overline{B}$ in Tables are the average values denoted by $\overline{B}_q$, where

$$\overline{B}_q = \sum_{k=0}^{N-q} B_k \tag{10}$$

B_k is the processors balancing coefficient for $V = [q/N, k/N]$ and q is the number of sequences of $C1$ category in the work. In the Tables, q is expressed in [%] and denoted as

$$\overline{V} = (q/N)100\% \tag{11}$$

and can be interpreted as sequences composition with respect to $C1$ category.

Tabl. 2. Coefficient $\overline{B}$ at $K = [1.0, 1.0, 1.0]$ for $\overline{V}$ and modes (case 1).

$\overline{V}$	$\overline{B}(S1,1)$	$\overline{B}(S2,1)$	$\overline{B}(S3, D^*)$	D^* in $S3$
0.0	1.56	1.68*	1.41	0.75
12.5	1.70*	1.70*	1.54	0.75
25.0	1.75*	1.72	1.69	0.75
37.5	1.68	1.75	1.89*	0.75
50.0	1.49	1.77	1.95*	0.625
62.5	1.35	1.80	1.96*	0.50
75.0	1.21	1.82	1.89*	0.375
87.5	1.10	1.84*	1.65	0.375
100.0	1.00	1.78*	1.45	0.375

Tabl. 3. Coefficient $\overline{B}$ at $K = [2.0, 2.0, 1.0]$ for $\overline{V}$ and modes (case 2).

$\overline{V}$	$\overline{B}(S1,1)$	$\overline{B}(S2,1)$	$\overline{B}(S3,D^*)$	D^* in $S3$
0.0	1.33	1.65*	1.28	0.75
12.5	1.50	1.62*	1.43	0.75
25.0	1.70*	1.59	1.60	0.75
37.5	1.85*	1.56	1.84	0.75
50.0	1.62	1.53	1.97*	0.75 ÷ 0.50
62.5	1.42	1.50	1.96*	0.37 ÷ 0.25
75.0	1.25	1.47	1.79*	0.25
87.5	1.12	1.44	1.53*	0.25
100.0	1.00	1.41*	1.33	0.25

Tabl. 4. Coefficient $\overline{B}$ at $K = [0.6, 0.5, 0.1]$ for $\overline{V}$ and modes (case 3).

$\overline{V}$	$\overline{B}(S1,1)$	$\overline{B}(S2,1)$	$\overline{B}(S3,D^*)$	D^* in $S3$
0.0	1.55	1.47	1.60*	0.625
12.5	1.45	1.49	1.75*	0.625
25.0	1.36	1.51	1.90*	0.625
37.5	1.29	1.52	1.88*	0.62 ÷ 0.50
50.0	1.21	1.54	1.97*	0.50
62.5	1.15	1.55	1.91*	0.375
75.0	1.09	1.57	1.87*	0.375
87.5	1.05	1.59	1.73*	0.375
100.0	1.00	1.60*	1.60*	0.375

In the Tables the best values $\overline{B}$ for each $\overline{V}$ are denoted by stars. Analyzing the results shown in the Tables we may come to the following conclusions:

1. The ranges of sequences compositions at which different modes of performing are the most effective. For example, in case 2 mode $(S1,1)$ is the best, when $\overline{V}$ is between 20% and 40%, mode $(S2,1)$ is the best at $\overline{V}$ ranging from 0% to 20% as well as between 90% and 100%, mode $(S3,D^*)$ has the best properties at the range from 40% to 90% of $\overline{V}$. For some values of $\overline{V}$ more than one mode can be the best (see $\overline{V} = 12.5$ in case 1).

2. The ranges of *goodness* for different modes vary for different execution times relations K. For each K dependencies $\overline{B}$ of $\overline{V}$ have some regularities. The relationships between $\overline{B}$ and $\overline{V}$ may be modelled by quadratic functions for modes $(S1,1)$ and $(S3,D^*)$; but by linear function for mode $(S2,1)$.
3. It is easy to find an analytical expression

$$|V(C1)K2 - [1 - V(C1)]K3 - V(C3)K4| \leq 1 - 2/N \qquad (12)$$

When relation (12) is satisfied, then mode $(S3,D^*)$ is almost always better than using structures $S1$ and $S2$. This situation is observed in case 3.
4. The best tasks allocation ratios D^* are strongly dependent on K and V. The intuitive division of the work into processors in proportion fifty–fifty (if $S3$ is used) is good only for narrow range of $\overline{V}$. If $D \neq D^*$, then usually other structures: $S1$ or $S2$ are more effective than $S3$ with D.
5. If for a given work described by K and V the choice of mode of performing is arbitrary or random , then we may encounter a significant loss of efficiency. The results shown in the tables may justify that statement. Obviously, it is true provided that the computer system is equipped with a control program which makes it possible to find an optimal mode of performing for a given work.

5. Control Algorithm

One of the general ideas proposed by Poźniak (1992) to solve problem (8) is recognition approach. According to this concept the choice of the most efficient mode of performing is made as a solution to a new recognition problem. This problem has no direct connection with the recognition problem considered in previous sections. It may be observed that we consider the recognition problems on two levels. On the first level the three stage recognition process is aided by controlling the realization of this process. On the second level the control algorithm is based on the solution of another recognition problem. The proposed concept is based on:

1. The choice of the best mode of performing for a given work is made by using the knowledge taken from past processes of performing the works.
2. The learning sequence is constructed from results of previous processes of performing works of the same nature. The works of the same nature are the works with the same amount of different categories to the sequences of tasks and the same set of different types of tasks but different K and V.
3. In a learning sequence we distinguish five parameters describing the work, interpreted as *features*. These are $V(C1)$, $V(C2)$, $K2$, $K3$, $K4$. The modes of performing (with values which were found as most efficient in previous experiments) are taken *as classes*. An *object* to be classified is the process of performing a given work. This work is characterized by its own set of features.
4. A two–stage recognition algorithm is suggested. At the first stage the set of features $K = [K2, K3, K4]$ is considered. Using a proper recognition algorithm L_1 (for example NN – algorithm) we may find the sector $u1$ in a learning

sequence (the set of rows associated with K found as the result of recognition at the first stage). At the second stage the sequences composition closest to the given work is chosen by using (NN) recognition algorithm. The mode of performing (*class*) belonging to the proper row of the learning sequence (chosen at the second stage) denoted by $u2$ is the solution of the recognition problem and is taken as the solution of the control problem (8). The block diagram of the control algorithm is shown in Figure 3.

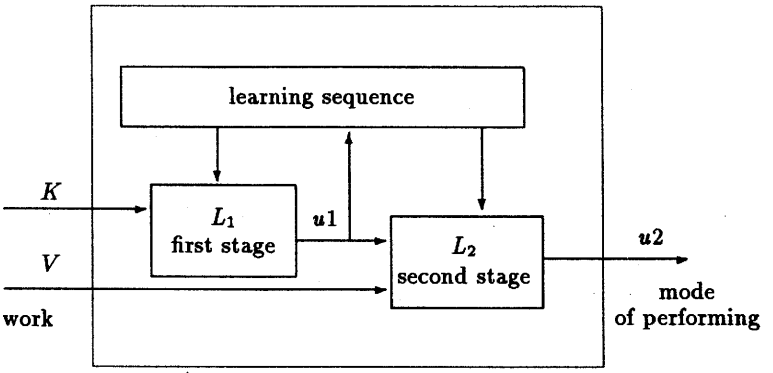

Fig. 3. Block–diagram of control algorithm (L denotes recognition algorithm).

6. Example

An example is given in order to illustrate the idea described above. To simplify the presentation, a shortened version of the control algorithm is shown. In this version only two accumulated features instead of five are considered. Those features are: the value $\overline{V}$ defined by (11) (which is given by ranges) and the parameter $\overline{k}$, where

$$\overline{k} = K2/[K3 + K4] \tag{13}$$

The data listed in Tables 2,3,4 are taken as the results of previous processes of performing (experiments). The shortened learning sequence to shorten the version of the algorithm is constructed on the basis of these results and is shown in Table 5.

In the example, the given work is characterized by the following times of executions to tasks:

$$t(01) = 8sec., \quad t(02) = 36sec., \quad t(03) = 24sec., \quad t(04) = 16sec.$$

Tabl. 5. Learning sequence.

features		class		sector	row
$\overline{k}$	$\overline{V}$	Sm	D^*	$u1$	$u2$
0.5	0 ÷ 15	$S2$	1	1	1
	10 ÷ 30	$S1$	1	1	2
	30 ÷ 45	$S3$	0.75	1	3
	45 ÷ 60	$S3$	0.625	1	4
	60 ÷ 70	$S3$	0.50	1	5
	70 ÷ 80	$S3$	0.375	1	6
	80 ÷ 100	$S2$	1	1	7
0.66	0 ÷ 20	$S2$	1	2	1
	20 ÷ 40	$S1$	1	2	2
	40 ÷ 50	$S3$	0.75	2	3
	50 ÷ 60	$S3$	0.50	2	4
	60 ÷ 70	$S3$	0.375	2	5
	70 ÷ 90	$S3$	0.25	2	6
	90 ÷ 100	$S2$	1	2	7
1.0	0 ÷ 40	$S3$	0.625	3	1
	40 ÷ 60	$S3$	0.50	3	2
	60 ÷ 100	$S3$	0.375	3	3*
	90 ÷ 100	$S2$	1	3	4

The work is composed of 35 sequences of $C1$, 5 sequences of $C2$ and 10 sequences of $C3$. The problem is to find the mode of performing at which the process of performing to the given work is expected to be the most efficient. Using (5) and (6) it may be found

$$K2 = 4.5, \quad K3 = 3.0, \quad K4 = 2.0 \quad \text{and} \quad V(C1) = 0.7, \quad V(C3) = 0.2.$$

The values of parameters defined by (11) and (13) may be obtained:

$$\overline{k} = 0.9 \quad \text{and} \quad V = 70\%.$$

Employing algorithm (NN) at the first stage, we choose the sector $u1 = 3$ at which $\overline{k} = 1.0$. The sector contains 4 rows. Next, at the second stage by using (NN) algorithm we find $u2 = 3$, as the row number 3 in a considered sector. Finally, we get the best mode of performing $(S3, 0.375)$, i.e. structure $S3$, and the tasks allocation ratio $D = 0.375$. The solution of the example is denoted by a star in Table 5. In practice, we should assign the number of $N(P1) = \text{ent}\,[ND^*]$ sequences to processor $P1$ (all of them of category $C1$ if possible). In the

example $N(P1) = 18$. Because of the fact that only a shortened algorithm is applied to solve the problem, the solution obtained is a suboptimal solution to the formal problem (8).

7. Conclusions

The proposed computer system to performe problem–oriented complex computational works is characterized by:

- opportunities of flexible processing by three system structures (described in Section 2); each of them is associated with effective tasks scheduling algorithm (presented in Section 2). It offers the possibilities to use one of $(N+3)$ modes of performing.
- control program (which is based on a two–stage recognition) to choose the best mode of performing (described in Section 5),
- distributed databases organized in the way described in (Pozniak,1990),
- programs to execute computational tasks (corresponding to user's procedures).

The results of experiments show that using a flexible system with the control program proposed may ensure the efficient load balancing and eventually may significantly improve the effectiveness of performing complex works.

The system called RWS (recognition-in-network) may be easily implemented in Novell net and may be used to aid the diagnostic process as well as to aid research and teaching in university laboratories. Moreover, the proposed methodology to design such a system (described in section 2) may be adopted to a broader class of complex computational works which are characterized by possibilities of tree–structured parallel processing.

References

Bokhari S.H. (1988): *Partitioning problems in parallel, pipelined and distributed computing.* — IEEE Trans. Comput., v.37, No.1, pp.48–57.

Borzemski L. (1988): *RDDML – Database server for LAN/KASK.* — Inst. of Contr. and Syst. Eng., Techn. Univ. Wrocław, Scientific Report, No.24.

Borzemski L. and Poźniak I. (1990): *Concept of distributed decision making system.* — Proc. Conf. *Knowl. Eng. and Expert System*, Wrocław, v.1, pp.280–286.

Casavant T.L. and Kuhl J.G. (1988): *A taxonomy of scheduling in general purpose distributed computing systems.* — IEEE Trans. Software Eng. v.14, pp.141–154.

Chu W.W. and Lan L. (1987): *Task allocation and precedence relations for distributed real–time systems.* — IEEE Trans. Comput., v.36, No.6, pp.667–679.

Eager D.L. and Lazowska E.D. (1986): *Adaptive load sharing in homogeneous distributed systems.* — IEEE Trans. Soft. Eng., v.12, No.5, pp.669–675.

Efe K. (1982): *Heuristic models of task assignment scheduling distributed systems.* — Computer, v.15, No.6, pp.50–56.

Hwang K. *et al.* (1982): *A Unix–based local computer network with load balancing.* — Computer, v.15, No.4, pp.55–66.

Kurzynski M. (1987): *Diagnosis of acute abdominal pain using three–stage classifier.* — Computers in Biology and Medicine, v.17, pp.19–27.

Ni L.M. and Xu C.W. (1985): *A distributed drafting algorithm for load balancing.* — IEEE Trans. Soft. Eng., v.11, No.10, pp.1153–1161.

Poźniak I. (1992): *Load control to complex work in two computers systems.* — Proc. Conf. *CIR'92*, Warszawa–Siedlce, pp.291–295, (in Polish).

Poźniak I. (1992): *The control of performing the complex work with application to diagnostic in medicine.* — Proc. Conf. *SAS'92*, Berlin, Elsevier Sc. Publ. B.V. 1992, pp.155–160.

Qin B. and Sholl H.A. (1991): *Micro time cost analysis of parallel computations.* — IEEE Trans. Comput., v.40, No.5, pp.613–628.

Stankovic J.A. (1984): *Simulations of three adaptive, decentralized controlled, job scheduling algorithms.* — Comput. Networks, v.8, No.3, pp.199–217.

Received February, 1993
Revised May 10, 1993