ICOT Technical Report: TR-369

TR-369

A Parallel-Inference Problem-Solving Mechanism for Computer Room Layout CAD System

by
Y. Iizuka, T. Watanabe & F. Mori

April, 1988

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191 ~ 5
Telex ICOT J32964

Institute for New Generation Computer Technology

A parallel-inference problem-solving mechanism for computer room layout CAD system

Yumiko Iizuka, Toshinori Watanabe, and Fumihiko Mori

Systems Development Laboratory
Hitachi Ltd.

1099, Ohzenji, Asao-ku, Kawasaki-shi, 215 Japan

Abstract

A parallel layout CAD system for the layout of computer room is developed on the sequential inference machine (SIM). The system is written in ESP, an object-oriented logic programming language developed by ICOT. Solving a layout problem requires treatment of various semantic information conventional programming language are not suited to represent and process. Our objective is to verify the suitability of ESP for solving the layout problem. Especially, in this paper, we discuss methods of constraint definition and evaluation involving multiple processes in parallel inference layout system. A layout problem is divided into several sub-problems, which are then solved in parallel. First, a rough layout is determined, in which equipment groups are assigned to appropriate zones in the computer room. Detail layout follows the rough layout, in which each device is placed appropriately within each zone. A process is activated for layout processing in each zone. Although these sub-problems are solved in parallel, it is necessary to represent and evaluate inter-zone constraints, which define layout constraints among devices belonging to different zones. There are two different approaches to the problem: a constraint can be assigned to one and only one of the two devices belonging to different zones, or it can be assigned to both of them. We implemented the two methods, and evaluated them through layout experiments.

1. Introduction

A parallel layout CAD system for the layout design of computer room is developed on the sequential inference machine (SIM). The system is written in ESP, an object-oriented logic programming language developed by ICOT. The representation and processing of semantic information in ESP, along with its versatile search capability , should make it a powerful tool for solving the layout problem. Dividing a large-scale layout problem into several sub-problems, which are in turn solved in parallel using the

parallel processing function of the ESP, will make search for layout solution highly efficient.

The sample system used here is the one to place various devices, which constitute a computer system, within a computer room in an appropriate manner. A number of constraints must be satisfied. The devices are grouped into several equipment groups. The computer room is divided into several zones, one for each equipment group. Assignment of an equipment group to a zone gives rise to a layout sub-problem, which are to be processed in parallel.

In placing each device, rules are first used to select candidate places for the device. Relevant constraints are evaluated to select candidate places for the device and to select the final placement. There are two types of constraint: intra-zone constraints, which pertain to the relationship among devices in the same zone, and inter-zone constraints, which pertain to the relationship among devices belonging to different zones. Note that, in order to check to see if an inter-zone constraint is satisfied or not, a process should consider the placement situation in other zones which are of course determined by other processes. These two constraint types are implemented in ESP.

2. Approach of the study

Various optimization methods have been used to obtain computer solution of layout problem [1]. However, it has been pointed out that there are qualitative differences between solutions obtained by a computer and those produced by human experts. One reason is that conventional programming languages are not suited to adequately represent various information concerning the layout problem. This has led to partial modelling of the entire problem; that is, only the part which can be expressed in numerical equations is extracted for processing by computer [2]. Partial modelling gives rise to a voluminous set of potential solutions, out of which the final solution must be searched. This tends to demand unpractical computer time.

Recently, knowledge processing approaches, including the use of LISP, are applied to solving layout problems [3,4]. Knowledge based systems can be used to represent the

problem more effectively than conventional OR methods. It can also incorporate expert heuristics for problem solving more powerfully [2]. One such example is the R1 system developed by McDermott [4] which automatically designs parts layout in computer device unit.

The layout system developed by Watanabe et al. [2] uses the knowledge based approach to design computer room layout. This system, originally written in LISP and utilized in practical system engineering task in Hitachi for some time, had been implemented in ESP to demonstrate the efficacy of the language in solving the layout problem [5]. Based on the results reported in [5] we experiment with solving a layout problem by the parallel inference processing function offered by ESP. The technical problems investigated here include division of the original layout problem into several sub-problems, parallel processing of the sub-problems, and information exchange among them. The logic programming language Prolog has a powerful search capability. The slot description and method description implemented in object-oriented programming language are effective in representing semantic relations associated with layout problem, such as description of objects to be placed, constraints to be satisfied, and current situation of the layout in progress which keeps changing as the layout process goes on.

Therefore, we chose ESP as the system description language. Since ESP is a Prolog-based object-oriented logic programming language with parallel inference capability, the language will be effective in implementing parallel solution of a layout problem, the demonstration of which is our chief objective in this study [5].

3. Structure of the system

3.1 Overview of the problem

Basically, the system implemented in ESP automatically develops a layout design of a computer room. Geometrical form of the room and configuration of the computer system are given as input data to the system. The system uses layout rules to determine the placement of each device. Various constraints are taken into account by the inference processing. There are two layers in the overall processing. In the upper layer

processing rough layout is determined. in which the devices are classified into several groups, the room is divided into zones, and each zone is assigned to one of the device groups. In the lower layer processing, detailed layout is determined in each zone. The upper layer is for sub-problem division, and the lower layer is for sub-problem solution. The system takes advantage of the parallel inference function of ESP to solve the sub-problems in parallel.

The processing flow of the system is as follows:

(1) Problem definition data (room data and device data) are read in. Grouping knowledge is used to classify the devices into appropriate groups (DISK group, CPU group, CCP group, etc.).

(2) Zoning knowledge is used to divide the room into zones. The number of zones is the same as that of the device groups. Each device group is assigned to one of the zones. The result of assignment is stored in the memory as layout situation.

(3) The process which performs the processings (1) and (2) becomes the parent-process. The process which designs the detailed layout within each zone is called child-process. The parent-process sequentially activates the child-processes, and performs the synchronization processing until all of the child-processes are ready. When all of them are ready, the parent-process orders them to execute the detail layout processing. The lower layer processing are executed in parallel.

(4) In the lower layer processing various constraints, including inter-device constraints, are evaluated to determine appropriate device placements. As noted earlier, the constraints include intra-zone constraints and inter-zone constrains. The former concerns the constraints which can be evaluated using information obtainable within a zone. An example, although somewhat trivial, is the constraint that a device should be placed within the zone to which it is assigned. The latter concerns the constraints the evaluation of which requires information obtainable in other zones, which are being processed in other processes. We identified and implemented two types of inter-zone constraints: inter-zone distance constraints, and inter-zone direction constraints. Inter-zone distance constraints define the constraints on the distance between two devices belonging to different zones. Inter-zone direction constraints define the constraints on

the relative directions to be satisfied by two devices belonging to different zones.

(5) If a device placement satisfying constraints cannot be determined, existing placements are slided in parallel to make room for the new device. In this case, the relative position among existing placements is kept the same.

Fig. 3.1 illustrates the system structure to perform the processing flow described above.

3.2 Structure of the ESP objects

Fig. 3.2 illustrates the ESP objects and their relations in solving the parallel layout processing. Major objects are described in the following:

(1) Problem definition object

The problem definition object represents the layout problem given to the system. It defines the floor plan of the computer room and the devices to be placed in the room. The geometrical form of the room and devices treated by the system are assumed to be rectilinear.

(2)Parallel problem solving object

The parallel problem solving object accepts the request from the user to solve a layout problem. It controls the overall process of problem solving. It requests the upper processing object to generate zones. When zone generation is over, it requests the parallel process control object to control the parallel processings performed by lower processing objects. It also accepts the placements of CD (console display device) and CPU (central processing unit), which are the reference points of the overall layout and specified interactively by the user of the system.

(3) Upper processing object

The upper processing object classifies all the devices in the problem definition into groups, and determines a zone for each group. In order to determine appropriate zones, it requests the upper rule object to generate zones, and the upper constraint evaluation object to evaluate the generated zones to see if they satisfy the constraints.

(4) Upper rule object

The upper rule object determines desired area of zones, and generates zones. The rules

do not have the condition part. The action part generates the zones. It requests the geometrical processing object to compute zone area. The generated zones are stored in the layout situation object.

(5) Upper constraint evaluation object

The upper constraint evaluation object evaluates the zones generated by the upper rule object to see if they satisfy the upper constraints. A couple of examples of the upper constraints are:

- Each zone must have sufficient area to contain all the devices assigned to the zone.
- Each zone must be within N meters from the entrance of the room.

It requests the geometrical processing object to compute zone area and distances between zones and the entrance.

(6) Parallel process control object

The parallel process control object controls the processings performed by lower processing objects which act in parallel. The processing control includes inter-process communication and synchronization. It first selects zones one by one, and activates the process of lower processing objects sequentially. When all of the lower processing objects are ready it requests the start of detail layout processing.

(7) Lower processing objects

Lower processing objects are used to solve the layout sub-problem in each of the zones. Devices in a device group are placed in appropriate positions according to the placement rules and the current layout situation. In order to determine appropriate position it requests the lower rule object to generate candidate positions, and the lower constraint evaluation object to evaluate the candidate positions to see if they satisfy constraints. If no candidate satisfies constraints, it requests the slide adjustment object to slide the existing devices to make room for the new one.

(8) Lower rule object

The lower rule object recognizes the current layout situation and generates candidate positions for a new device to be placed.Current layout situation is recognized in the IF part, and candidate positions are generated in the THEN part. Candidate positions are represented as a range. A sample rule, the rule for floppy disk drive (FD) is as follows:

IF CR (card reader) is already placed in the right hand side of
CD (console display) facing to the left

THEN generate the candidate positions for FD as follows:

1st candidate: the right hand side of CR facing to the left and
within 4 meters from CR

2nd candidate: the left hand side of CR and
within 5 meters from CR

It requests the geometrical processing object to compute distance between devices, and to determine the direction of the devices. The candidate positions are stored in the layout situation object.

(9) Lower constraint evaluation object

The lower constraint evaluation object is used to evaluate the candidate position generated by the lower rule object to see if they satisfy the lower constraints. As noted earlier, the lower constraints include intra-zone constraints and inter-zone constraints.

Examples of intra-zone constraints are:

- The maintenance area for each device should not overlap any fixed objects in the room, such as a pillar.
- The maintenance area for each device should not overlap other device, although it can overlap,albeit undesirable, maintenance area of other device.
- The maintenance area for each device should not cross the border line of the room.

Examples of inter-zone constraints are:

- The front line of device A in a zone must face the same direction as that of device B placed in another zone.
- The distance between device A in a zone and device B in another zone should (or should not) be more than X meters.

It requests the geometrical processing object to compute distance between devices and to check overlap.

(10) Slide adjustment object

When no candidate position satisfies constraints, the adjustment object is used to

slide the devices already placed to make room for the new device. The existing objects are moved within the range of its candidate position.

(11) Layout situation object

The layout situation object represents the current layout situation. Zone position, zone form, candidate position of devices, and placement positions are represented in this object.

(12) Geometrical processing object

The geometrical processing object is used for computation of the area of rectilinear forms, overlap checking, direction determination, and other geometrical processings.

The size of the system is approximately 16K lines. It contains approximately 200 objects.

4. A realization of the parallelism

This system solves the detail layout problem in each of the zones by parallel processing. The parallel processing capability of ESP is utilized. Control method of the parallel processing is discussed in this section.

4.1 Synchronization

The lower processing processes, which perform the detail layout design within each zone, are activated as child-processes by the upper processing process. Since the process activation is performed sequentially by the parent-process, process activation must be synchronized. That is, the parent-process must be sure that all the necessary child-processes are normally activated before the child-process can start the parallel processing. The stream function [6] offered by the SIMPOS operating system is used to realize the synchronization processing.

Fig. 4.1 illustrates the synchronization mechanism of the system. As shown in the figure, the receiving stream is used to receive the messages from the child-processes to the parent-process, and the sending stream is used to send messages from the parent-process to the child-processes. The parallel process control class, which is in the upper layer and responsible for the synchronization, activates child-processes one by one, each

corresponding to a zone. A child-process, when activated, notifies the receiving stream in the message transmission class that it is ready for execution of detail layout processing. The message transmission class waits until the notification is received from all of the child-processes. When all child-processes are ready, the message transmission class posts it to the parallel process control class. After that, the parallel process control class sends a message to the sending stream of the message transmission class requesting the start of execution. Thus all the child-processes start the detail layout processing at the same time.

4.2 Inter-zone constraints

When detail layout is determined in the lower layout processing two types of constraint should be considered: intra-zone constraint and inter-zone constraint. The latter deserves special attention, since they pertain to the constraints on device layouts in two or more zones which are processed by different processes. In this system we consider inter-zone distance constraints and inter-zone direction constraints. In the case of a constraint concerning two devices within the same zone, the constraint can be applied to the device to be placed later in the order of placement determined by the system without any problem. However, in the case of inter-zone constraints, it is unpredictable which of the two devices is placed first. Therefore, it may well be that, when a constraint is to be evaluated, the reference device has not been placed yet. In order to solve this difficulty, we implemented two methods for constraint definition and evaluation.

As in the case of an operating system, in which synchronization is achieved through a shared variable among processes[7], one of the methods to solve the difficulty is to use an object representing the layout situation as a shared object, such as a 'blackboard', among processes. If a process which tries to place a device A must wait for another process to place device B, the former process can do so looking at the shared object. Synchronization among processes can thus be achieved. This is the uni-directional inter-zone constraint. On the other hand, this method can lead to a chain of waitings, or deadlock. To solve this difficulty, another method, called bi-directional inter-zone

constraint, is also considered.

(1) Uni-directional inter-zone constraint

The uni-directional inter-zone constraint defines constraint on one and the only one of the two devices. For example, if a constraint says that the distance between device A and device B must not be greater than 6 meters, the corresponding uni-directional constraint will be defined for device A as "Device A must be placed within 6 meters from device B." If the device to be used as the reference is not placed in its zone yet, the process is halted until the reference device is placed in its zone by the other process. The former process is re-started when the reference device is placed. That is, if B is not placed yet, the process for A is halted until the process for B places B in some appropriate position.

(2) Bi-directional inter-zone constraint

The bi-directional inter-zone constraint defines essentially the same constraint for both of the two participating devices. Using the same example as the one just considered, the constraint "A should be within 6 meters from B" is given to device A, and the constraint "B should be within 6 meters from A" is given to device B. Therefore, essentially the same constraint is defined twice. When bi-directional constraint is used, and if the reference (the other) device is not placed yet, the constraint is ignored in considering the placement of the device. If device B has not been placed yet when device A is to be placed, the process for A does not evaluate the constraint concerning devices A and B, and at a later time point the process for device B evaluates the other (semantically the same) constraint given to B when it comes to the placement of B.

5. Experiments

Here we cite the result of three experiments.

- Experiment 1

Uni-directional inter-zone constraints are used. The constraints are:

(1) mt03 (magnetic tape) must be placed within 8 meters from tr01 (paper tape reader).

(2) kcp01 (card puncher) must be placed within 3 meters from disk07 (magnetic disk device).

Fig. 5.1 is the layout result satisfying the two constraints cited above. Table 5.1 shows the action chart for experiment 1. The w's in the table means "waiting." It can be seen that mt03 (zone 3) and kcp01 (zone 5) wait until tr01 (zone 5) and disk07 (zone 03) are placed, respectively, and after the reference devices are placed, inter-zone constraints are evaluated to place mt03 and kcp01.

- Experiment 2

Uni-directional inter-zone constraints are used. The constraints are:

(1) lp01 (line printer) must be placed within 5 meters from kpr02 (kanji printer).

(2) lp02(line printer) must be placed within 5 meters from kpr01(kanji printer).

When the two constraints are given to the layout system, the process for zone 3, when it comes to the placement of lp01, waits for the process for zone 4 to place kpr02. On the other hand , the process for zone 4, when it tries to place lp02, waits for the process for zone 3 to place kpr01. However, since kpr01 and kpr02 are to be placed after the placements of lp01 and lp02, deadlock occurs in which the both processes enter into the wait status.

Fig. 5.2 shows the layout results in which the deadlock occurred. Table 5.2 shows the action chart. "dead" in the table indicates that a deadlock occurred when lp01(zone 3) and lp02 (zone 4) are to be placed.

- Experiment 3

Bi-directional inter-zone constraints are used. The constraint is that the distance between cr01 (card reader) and mt03 (magnetic tape) should not exceed 6 meters.

This constraint is given as bi-directional constraints as follows:

(1) cr01 must be placed within 6 meters from mt03.

(2) mt03 must be placed within 6 meters form cr01.

The two constraints are given to cr01 and mt03, respectively. However, actual evaluation of the constraint is performed for the device which is placed later. The constraint for the device placed first is not evaluated.

Fig. 5.3 illustrates the layout output for cr01 in experiment 3. Constraint (1) was considered, but since mt03 (zone 3) had not been placed yet, it was not evaluated, and cr01 was placed with no constraints.

Fig. 5.4 illustrates the layout output for mt03 in experiment 3. Constraint (2) was considered, and since cr01 (zone 5) had been placed already, it was considered effective and evaluated, and mt03 was placed within 6 meters form cr01.

When bi-directional inter-zone constraints are used, essentially the same constraints should be defined for both of the relevant devices. However, since a process will not wait for another process to make placement of reference device, no deadlock phenomena occur as in the case of uni-directional inter-zone constraints.

6. Discussions

The experiments show that the uni-directional and bi-directional constraints have the following problems:

(1) Uni-directional inter-zone constraints

In the case of uni-directional inter-zone constraints, a process enters the wait status if the reference device in other process has not been placed yet. Therefore, processing efficiency is hindered when process P waits for process Q and process Q waits for yet another process R. Moreover, as seen in the experiment 2 cited above, contradiction between the order of waiting and the order of actual placement determined *a priori* by the system leads to the potential deadlock. The deadlock problem is solved in this system by the introduction of deadlock monitoring mechanism. The deadlock monitoring mechanism is activated when a process begins to wait for another process. It checks the inter-zone constraints to see if any other process exist which wait for the process which entered the wait status. If a deadlock is detected, the process in the deadlock are forced to terminate. The deadlock monitor is activated at a constant time interval.

(2) Bi-directional inter-zone constraints

One of the shortcomings of the bi-directional inter-zone constraints is that the same constraint must be defined for each of the relevant devices. Therefore, the volume of constraints description is twice as much as that in the case of uni-directional constraints.

Another problem is that, if the placement of the second device begins before that of

the first device the constraint may be ignored for both of the devices. Placement of a device is a lengthy process, including rule-based selection of candidate positions, and evaluation of intra-zone and inter-zone constraints for each of the candidate positions. A lot of geometrical computation is necessary along with the process. Therefore, it takes time to complete the placement of a device. It is not an instantaneous process. If placement of device B starts while the placement of device A is not completed, the process for B may well consider that A has not been placed yet. This leads to the situation that a constraint is not evaluated at all. A solution of this problem is to use flags to indicate the beginning of placement processing for each device. However, this is not a perfect solution.

Thus, it can be observed that the uni-directional and bi-directional approaches to represent inter-zone constraints have complemental properties. Constraints given in the uni-directional approach are never ignored, but they can lead to a long chain of waiting processes, or even to the deadlock phenomena. Using the bi-directional approach does not lead to the abnormal termination of the process, but given constraints may be totally ignored. That is, uni-directional approach gives higher priority to constraint evaluation than to the progress of layout processing. On the other hand, bi-directional approach considers it more important to make progress in the layout design than to check every constraints given to the system. Therefore, the two approaches should be combined effectively and efficiently, considering the strength that a constraint should have, and its effect on the progress of placement processing.

In fact this observation can be generalized to cover any constraint-oriented, knowledge-based, and parallel-processing planning and/or design systems. In these systems there should be a body of constraints which pertains to two or more of the sub-problems to be solved in parallel. The description of such constraints should either be uni-directional or bi-directional, as in the case reported in this paper. This fact should be reflected in the overall architecture of the system description language, although the detailed discussion of the architectural aspect of knowledge processing language is beyond the scope of this paper.

7. Summary

A parallel layout CAD system for computer room layout is implemented on sequential inference machine and in ESP. Parallel processing is implemented using the parallel process function of ESP. The parallelism requires representation of inter-zone constraints. Two approaches are considered: uni-directional representation and bi-directional representation. The two representation schemes are experimented and their complemental properties are investigated.

Acknowledgements

The authors wish to thank Mr. Nobuyoshi Dohmen and Mr. Koichiro Ishihara of the Systems Develpment Laboratory, Hitachi Ltd for their guidance. Special thanks are due to Mr. Yuichi Fujii of the Institute for New Generation Computer Technology for helpful comments and discussions.

References

[1] Liggett, R. S. : Optimal Space Planning in Practice, CAD, Vol. 13, No. 5, pp. 277-288 (1981)

[2] Watanabe, T. et al. : Design of an Expert System for Computer Room Layout, Transaction of Information Processing Society of Japan, Vol. 26, No. 5, pp. 926-935 (1985)

[3] Pfefferkorn, C. E. : A Heuristic Problem Solving Design System for Equipment of Furniture Layouts, C. ACM, Vol. 18, No. 5, pp. 286-288 (1975)

[4] McDermott, J. : A Rule-Based Configurer of Computer Systems, Artificial Intelligence, Vol. 19, No. 1, pp. 39-88 (1982)

[5] Watanabe, T. et al. : Computer Room Facility Layout in ESP, Proc. of the Third Japanese-Swedish Workshop, pp. 13-14 (1985)

[6] Hattori, T. et al. : SIMPOS : An Operating System for a Personal Prolog Machine PSI, ICOT Technical Report; TR-055 (1984)

[7] Maekawa, M. et al. : Operating Systems, The Benjamin/Cummings Publishing Company, Inc. (1987)

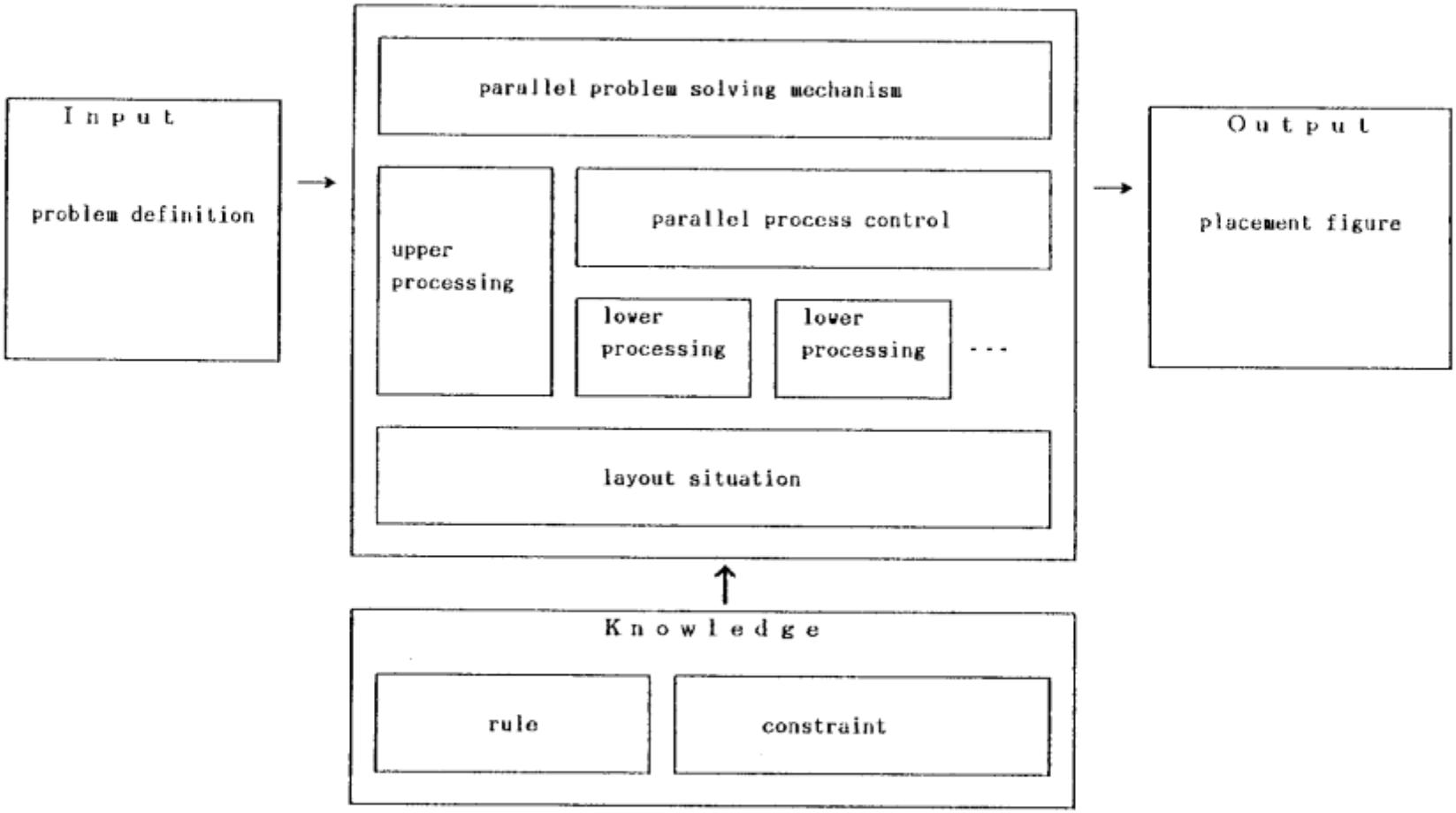


Figure 3.1 Structure of the system

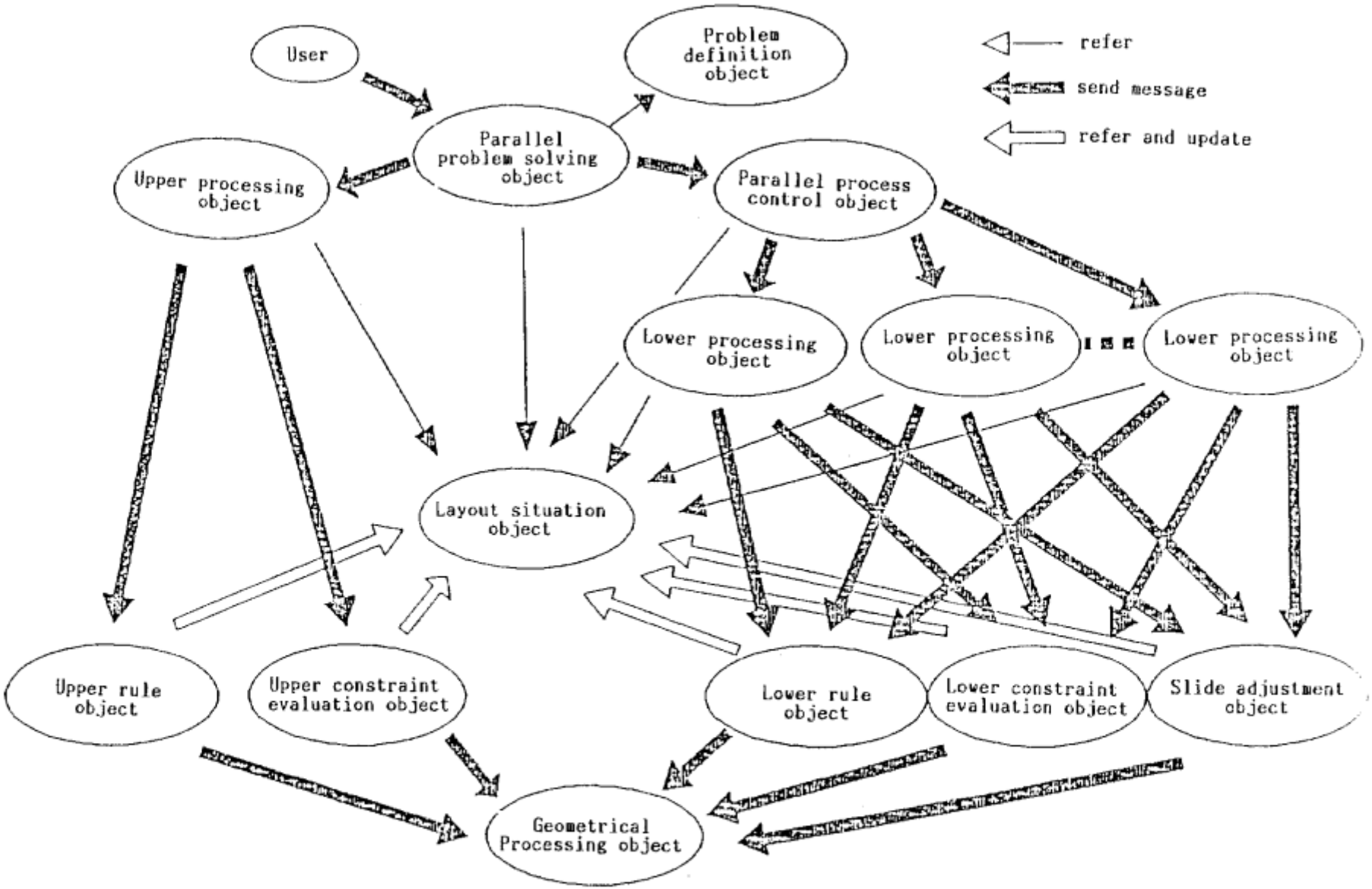


Figure 3.2 ESP objects

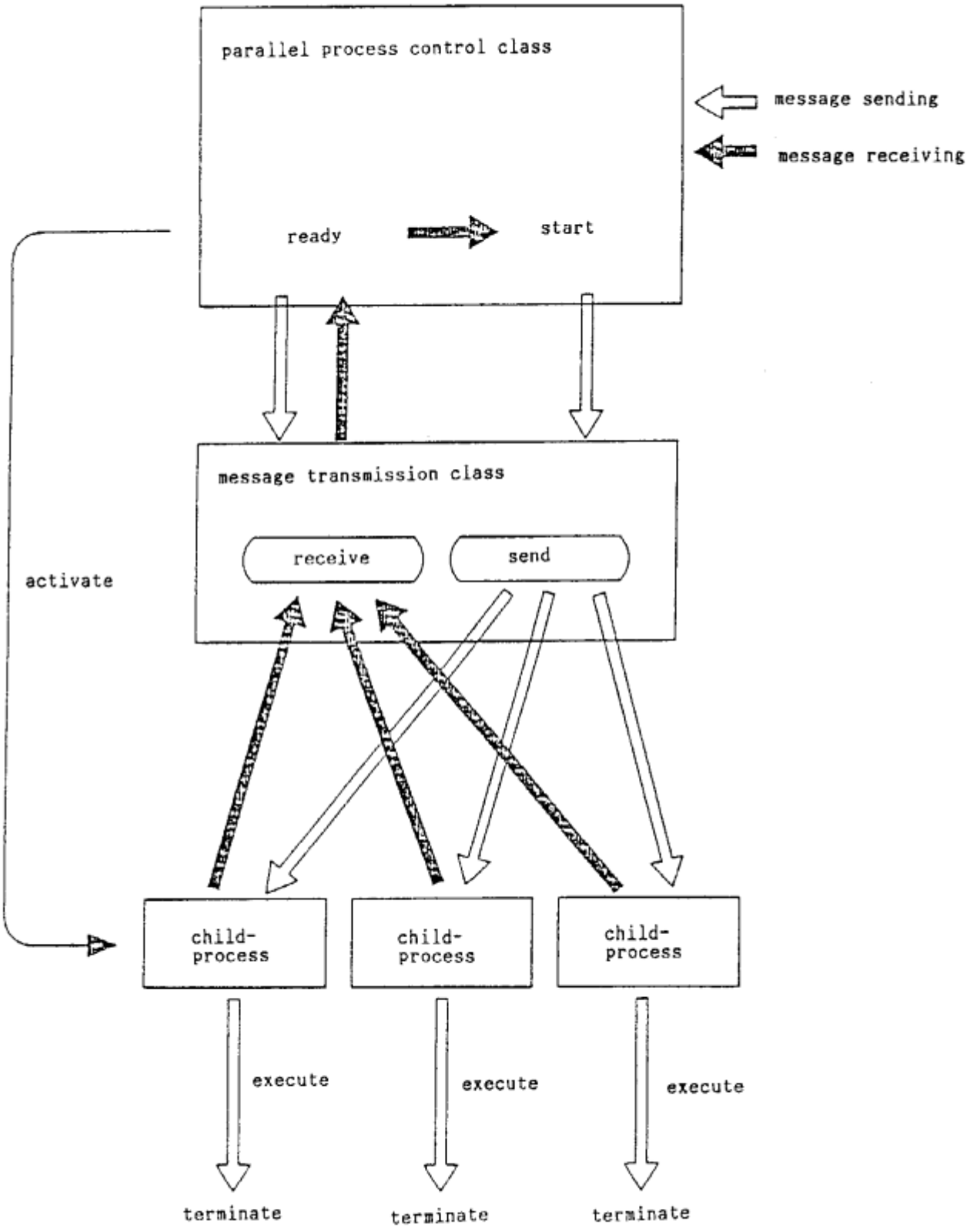


Figure 4.1 Synchronization

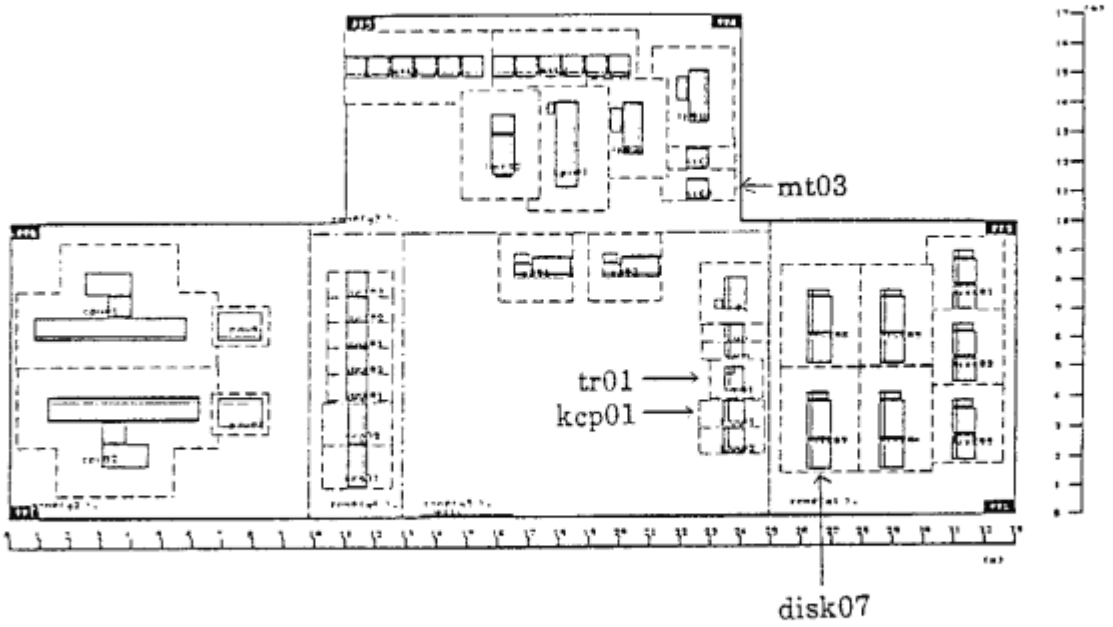
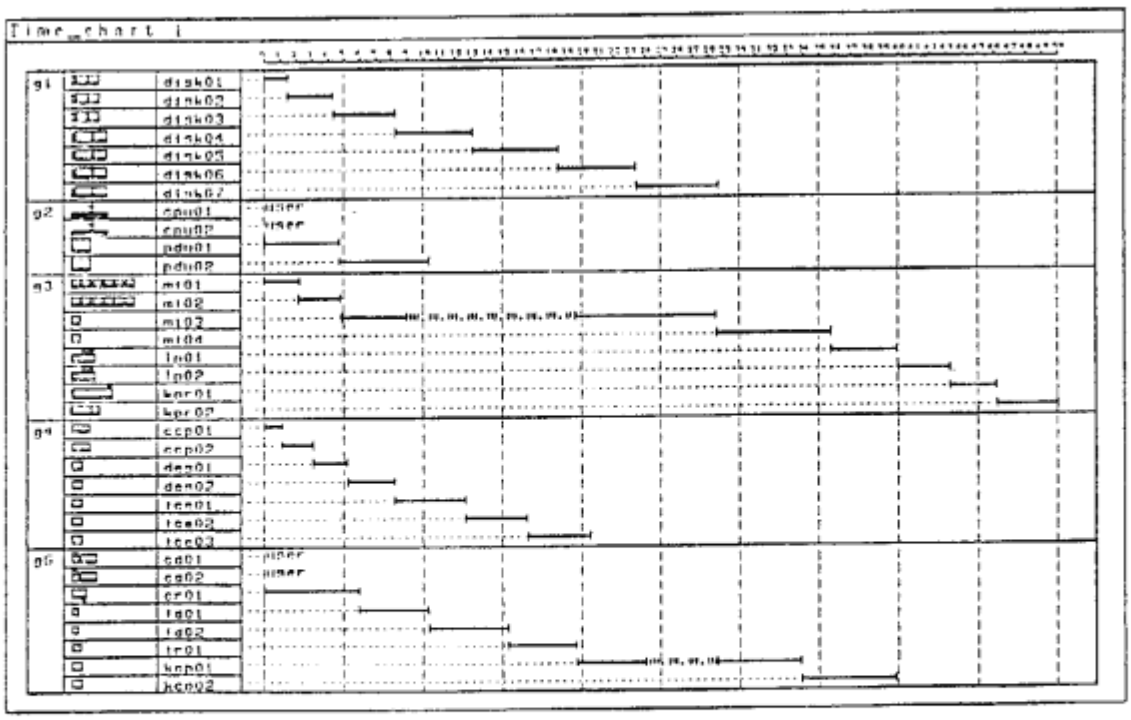


Figure 5.1 Layout output of experiment 1

Tabel 5.1 Action chart of experiment 1



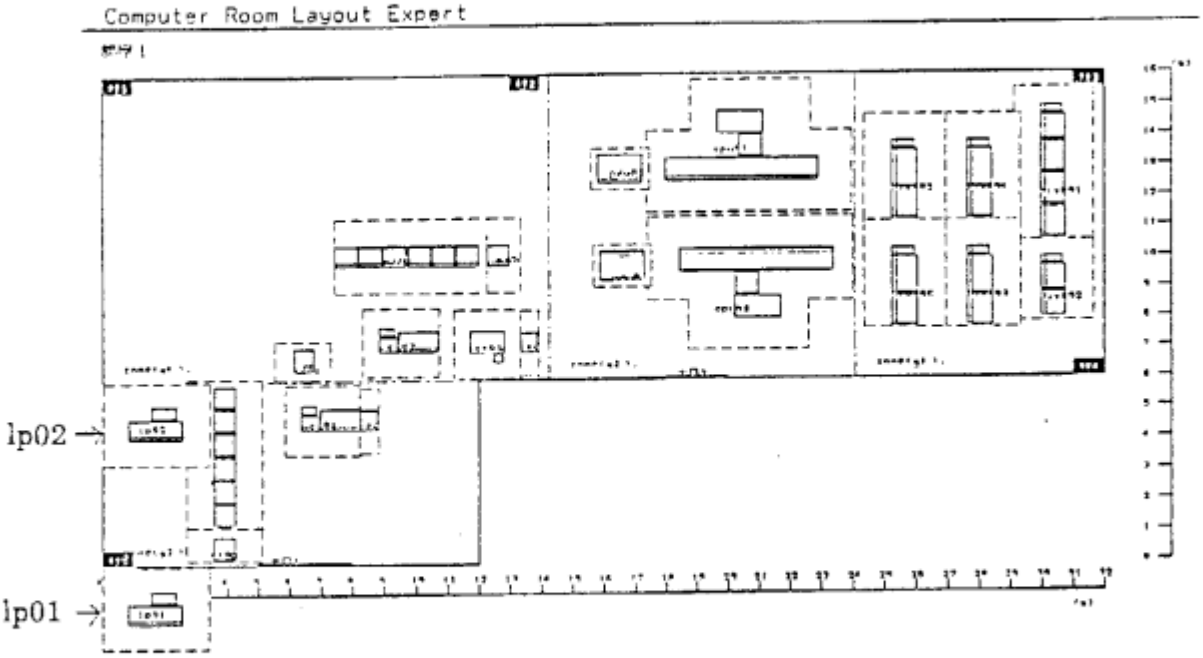
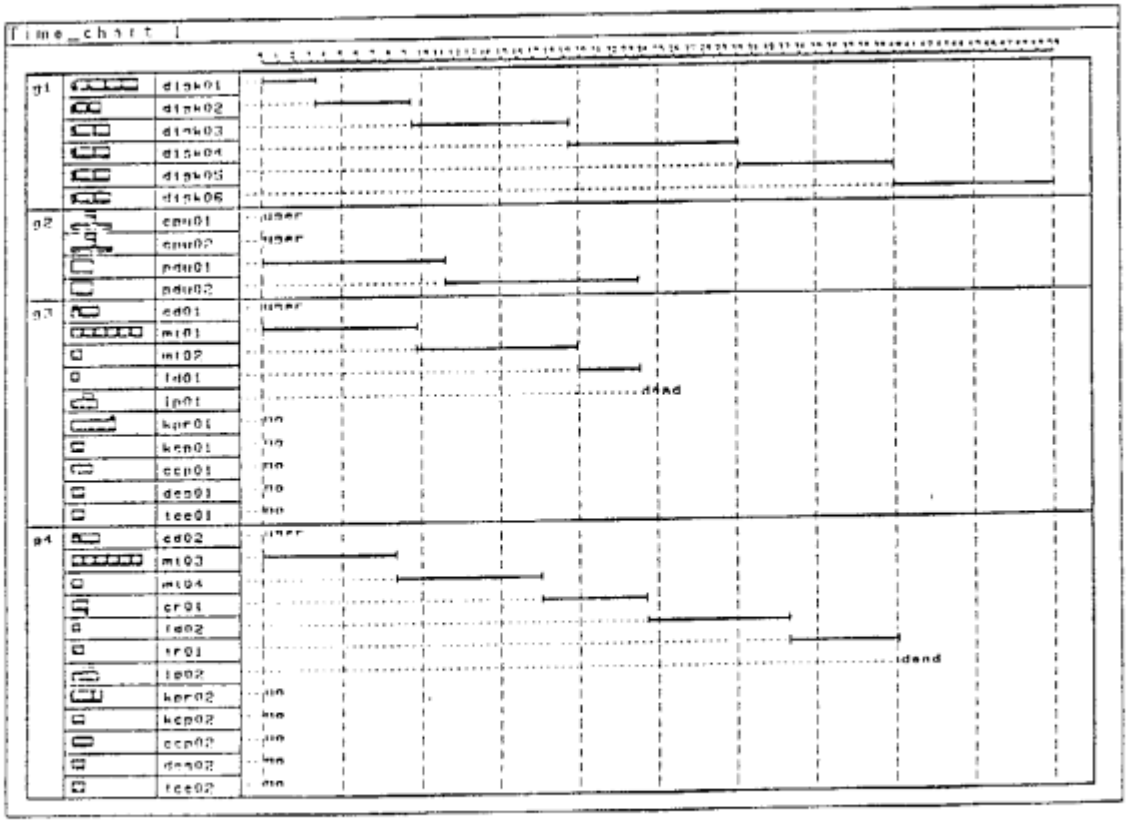


Figure 5.2 Layout output of experiment 2

Tabel 5.2 Action chart of experiment 2



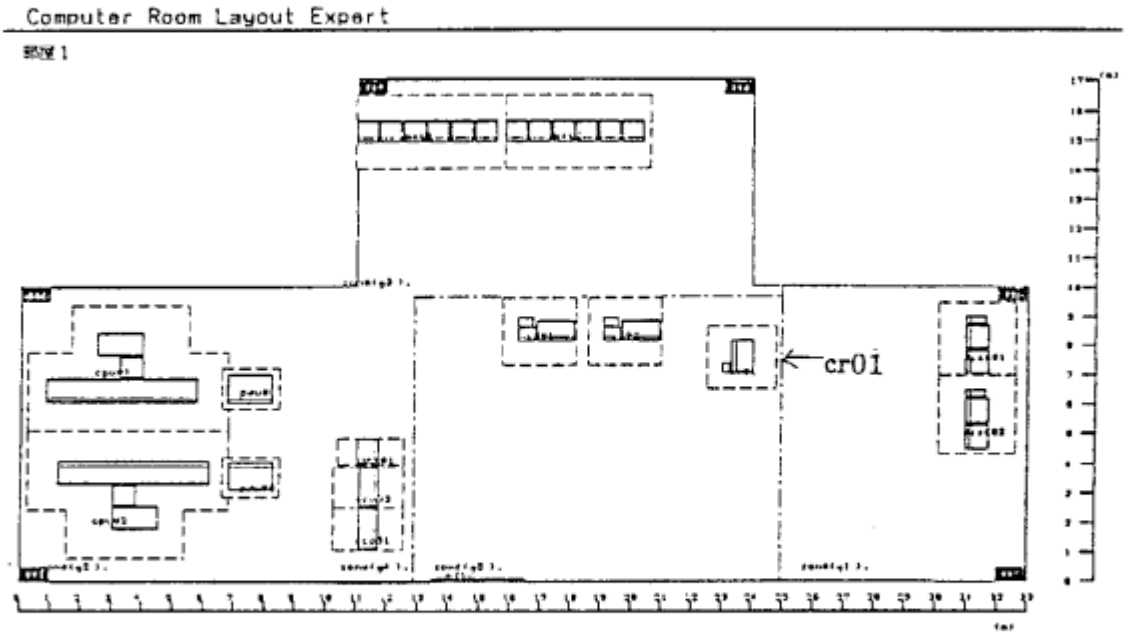


Figure 5.3 Layout output of experiment 3 (1)

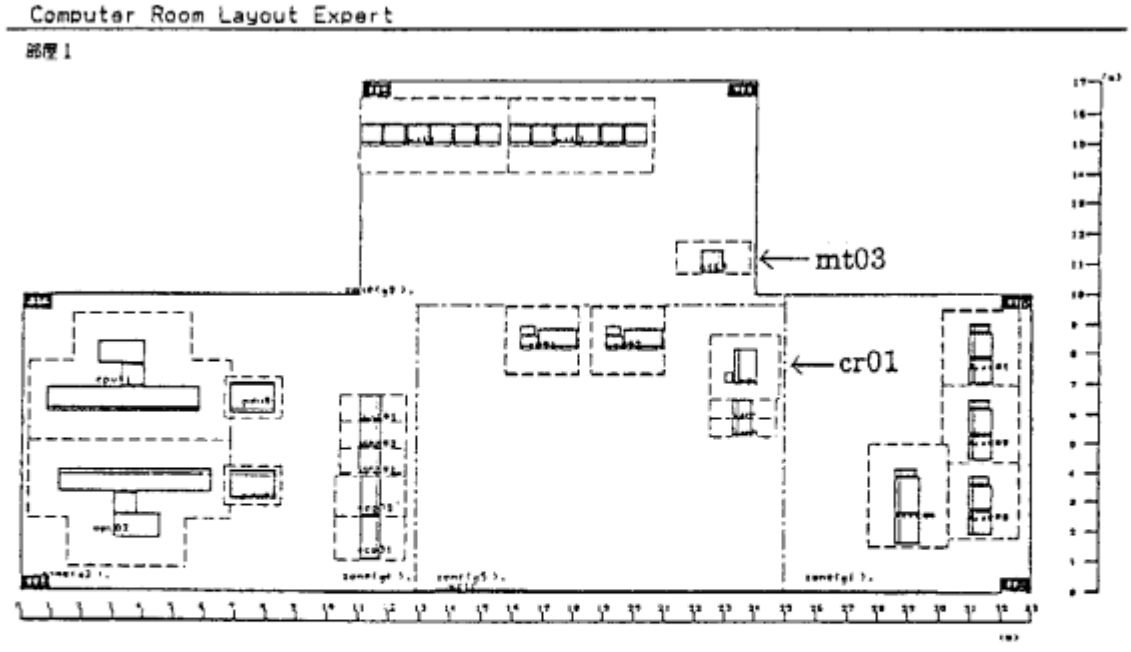


Figure 5.4 Layout output of experiment 3 (2)