From: IAAI-98 Proceedings. Copyright © 1998, AAAI (www.aaai.org). All rights reserved.

An Expert System for Alarm System Planning

Akira Tsurushima, Kenji Urushima, Daigo Sakata,
Hiroyuki Date, Masatomo Nakata, Yoshinobu Adachi, and Kazuhisa Takahashi
Artificial Intelligence Department SECOM Intelligent Systems Laboratory
Mitaka Tokyo 181, Japan
Email: akira@ai.isl.secom.co.jp

Abstract

This paper discusses the design and implementation of ESSPL, an expert system which generates security plans for alarm systems (Figure 1). Security planning is the task of determining an efficient layout of sensors, alarms, and the associated wiring for a building. ESSPL uses Rule Based System technology to generate a plan and Constraint Based technology to position the alarm equipment. A Constraint Logic Programming Language (CONTA) is developed to solve the positioning problem.

ESSPL proved to be an excellent tool in automatic planning in laboratory tests. Field tests were also carried out and examined whether there is a drawback to use ESSPL in the real world context. This is also discussed in this paper.

Problem Description

SECOM Co., Ltd. is the leading security service company in Japan, providing alarm system services to more than 400,000 subscribers over the country. When a sensor detects an abnormal situation, the alarm system sends a signal to the central monitoring station. A trained response officer is then dispatched to respond to the situation. To provide high quality alarm services, the remote sensors and equipment layout for a building is crucial. This design activity is called security planning. [1]

Security Planning

Security planning is the problem of determining an efficient arrangement of alarm equipment for a building based on an analysis of its potential risks. Poor planning results in an increase of false alarms and failures to alarm, decreasing the overall quality of security services.

Security planning is accomplished by solving three subproblems; 1) alarm zoning, 2) sensor arrangement, and 3) system integration.

Alarm zoning is the process of dividing the building into several security zones depending on the building usage. This allows customers to arm (unarm) all of the alarms in a given zones independently of other zones.

Sensor arrangement determines the best layout of sensors for a building. In this subproblem, there are two important decisions: 1) selection of the proper sensor for the situation and 2) positioning the sensor in the most appropriate location.

System integration connects all of the sensors which are distributed throughout the building and introduces all other necessary equipment to the security plan. This task involves determining security circuits by grouping sensors, selecting proper control equipment and peripherals, assigning addresses to the sensors and connecting all of the equipment.

Traditionally security planning has been undertaken by human engineers. However, security planning is a very complicated design task which requires experience and expertise especially for large scale buildings. Therefore there are only a few engineers who are able to design large security plans. This expert system was developed to overcome these limitations.

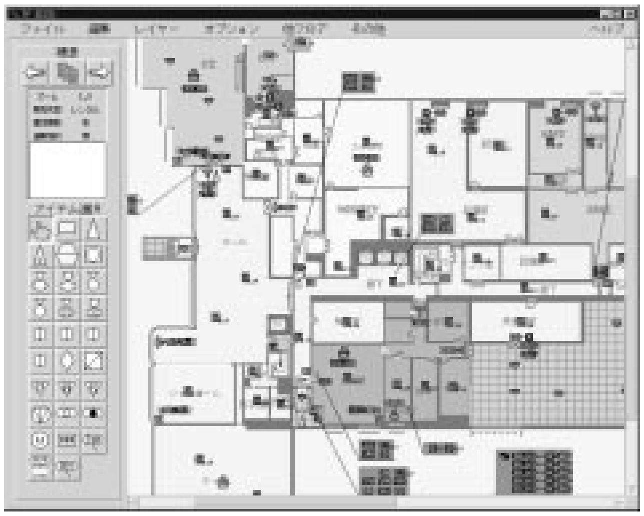

Figure 1: ESSPL: The Main Window

[1] In this paper, the term planning or plan is used in a somewhat different manner than is typically used in the AI community. The definition is close to that of design or configuration.

Copyright 1998, American Association for Artificial Intelligence (www.aaai.org). All right reserved.

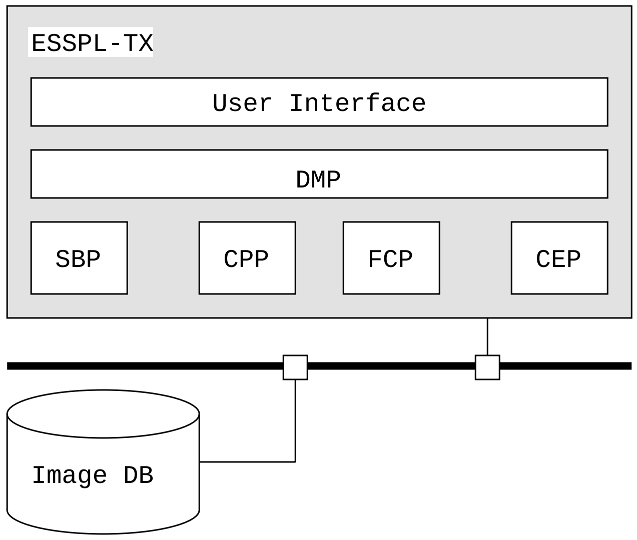

Figure 2: System Architecture

ESSPL : An Expert System for Security PLanning

There is relatively little work that has been published about the application of AI technology to the security field (Ward & Sena 1986; Lamont 1988) except in the field of computer security. Our work is an unique attempt to apply AI technology to the security service industry.

ESSPL (Expert System for Security PLanning)is an expert system which designs a security plan for SECOM ALARM SYSTEM, an alarm system specially designed for large scale office buildings and multi occupancy buildings.

Figure 2 shows the system architecture of ESSPL. First, ESSPL takes a raster image of a building floor plan from the image database. Using the user interface subsystem, the ESSPL user traces the basic architectural elements (e.g. rooms, doors, windows, etc.) of a building on the raster image. This template becomes the input to ESSPL and from this point on, the system creates a security plan without any user intervention. The completed security plan is then output as a printout.

ESSPL consists of the following six subsystems:

1. User Interface (UI)
2. Data Management (DMP)
3. Zone Planning (SBP[2])
4. Sensor Planning (CPP[3] & FCP[4])
5. Control Equipment Planning (CEP)

The knowledge base subsystems (SBP, CPP, FCP and CEP) and the data management subsystem (DMP) are implemented using a combination of CLIPS, SICStus Prolog, and CONTA, a constraint programming language developed for this project. The UI subsystem is implemented in Tcl/Tk.

[2]SBP—Security Block Planning
[3]CPP—Crime Prevention Planning
[4]FCP—Fire Containment Planning

User Interface

As shown in Figure 1 ESSPL has a CAD like user interface with the following functions:

1. Create building data by tracing a floor plan
2. Edit the resulting template
3. Display the result

Since the UI has almost all basic CAD capabilities, an user can design a plan manually without using the automatic planning function. These CAD capabilities are used to create building data and to modify the planning results.

In this project, one of the major design issues was system flexibility due to the following reasons:

- New products are frequently introduced, therefore ESSPL must be updated correspondingly.
- It is not possible to anticipate all necessary types of rooms since there are almost a limitless number of types.

For the first problem, the UI refers to a separate database of alarm system equipment which allows the program to be independent of equipment information. When new equipment is introduced, all the users need to do is to update the database, which does not require program modification.

For the second problem, the UI has a room type attribute table. In ESSPL, all room types are defined with respect to the following nine planning attributes:

1. appropriateness as a main final exit
2. appropriateness as a private zone
3. appropriateness as a entry route
4. appropriateness for general equipment placement
5. appropriateness for control equipment placement
6. appropriateness for monitoring equipment placement
7. level of intrusion risk in a lower floor
8. level of intrusion risk in a higher floor
9. type of fire detector

For example, room type 'office' is defined as 1. BAD, 2. EXCELLENT, 3. BAD, 4. GOOD, 5. FAIR, 6. FAIR, 7. LEVEL 3, 8. LEVEL 2, 9. SMOKE DETECTOR. Since ESSPL does not refer directly to the room types but rather these attributes, users can introduce new room types by setting these attributes. In this way ESSPL can handle new room types correctly.

Zone Planning (SBP)

Zone planning arranges the rooms in the building into several spatial units (zones) and determines each of their final exits. When a customer arms a zone (i.e., all sensors in the zone are switched on), they must leave the zone from the final exit, otherwise the alarms will activate. Zones are classified into three categories: private, public, and independent. Zones are determined by the following factors:

- room type attributes and door types
- building topology (how the rooms are connected by doors)
- customer request

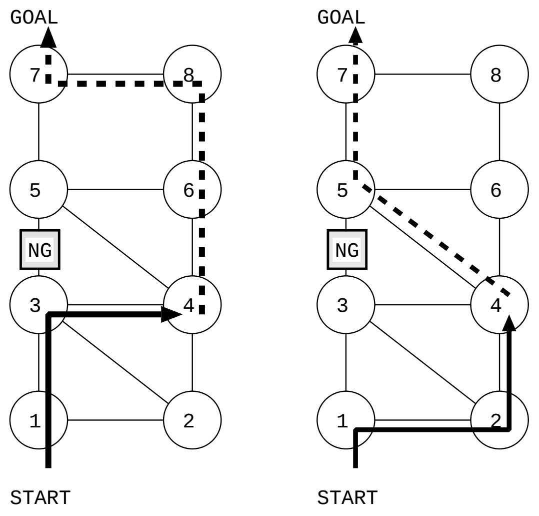

Figure 3: An example of the two paths through the same node (node 4)

The key element in zone planning is in determining the public zone. A public zone is a section of a building that anyone in the building is approved to enter. A public zone can only be armed after all other private zones are armed. Only one public zone is allowed per building owing to the system specification, and the final exit of this zone is called the main final exit of the building.

When determining a public zone, it is important to ensure that a customer can leave the building without entering any other private zones. Once a public zone is properly determined, arranging private zones may be trivial. The SBP subsystem searches for a route which connects the final exit of each private zone to the main final exit of the building. This is accomplished by a modified branch and bound algorithm with an evaluation function

$$f(n_i) = f(n_{i-1}) + C(n_i)$$

where n_i is the i-th room on the path. $C(n_i)$ is determined by the value of the 'appropriateness as a entry route' room type attribute of n_i as follows:

$$C(n_i) = \begin{cases} 20 & \text{if value} = \text{BAD} \\ 5 & \text{if value} = \text{FAIR} \\ 2 & \text{if value} = \text{GOOD} \\ 1 & \text{if value} = \text{EXCELLENT} \end{cases}$$

In ESSPL, an user can select any door as the final exit of a private zone. If a door is selected to be the final exit, the two rooms connected by the door cannot be members of the same zone. Therefore this condition must be treated as a constraint on the search procedure. This constraint is called **FE-constraint** and the set of nodes which are affected by FE-constraint the **prohibition node set** of the path.

For example, the graph in Figure 3 shows the room topology in a building, where the nodes and vertices represents rooms and doors, respectively. The box containing characters 'NG' represents the FE-constraint. In this example, nodes 3 and 5 are affected by this; therefore they both cannot be members of the same path. Suppose a search proceeds through nodes 1, 3 and 4. Node 5 is now a member of the prohibition node set because node 3 is on the path. Therefore, the path must proceed through nodes 6, 8 and 7 (as shown in the left hand side).

Suppose a search proceeds through nodes 1, 2 and 4. In this case, since node 3 is not on the path, the prohibition node set is empty, allowing node 5 to be placed on the path. Therefore the rest of the path can proceed through nodes 5 and 7 (as shown in the right hand side).

This example shows that the prohibited nodes are dynamically changes as the search proceeds. The whole search procedure is as follows.

```
Search (start goal)

1. Form a queue of partial paths. Let the initial queue
   consists of the zero-length path from the start node
2. While the queue is not empty repeat the following procedure
   (a) Remove the first path from the queue
   (b) Form new paths from the removed path by extending
       one step
   (c) Return the path if the goal is reached. Stop the procedure
   (d) Remove the paths that violate the FE-constraints
   (e) For each remaining path, do one of the following:
       i. Do nothing if there is a path which has the same last
          extended node, the same prohibition node set and a
          higher $f(n_i)$ value in the queue
       ii. Else, remove all the paths which have the same last
          extended node, the same prohibition node set and a
          lower $f(n_i)$ value from the queue. Insert the path
          into the queue and sort it by the evaluation function
          in ascending order
3. Report failure and stop the procedure
```

The FE-constraint is applied in step 2-(d). Since two paths having the same last extended node may results in two different solutions due to the FE-constraint, the queue must hold all paths rather than just the last extended nodes. Furthermore, the optimal path cannot be determined until the goal is reached. This makes the search procedure inefficient, especially for large graphs. Step 2-(e) handles this problem by pruning non-optimal paths from the queue.

Sensor Planning (CPP & FCP)

SECOM ALARM SYSTEM can detect two types of emergencies with its sensors: fire and intrusion. Security planning for both are carried out in a similar manner, therefore this section focuses on intrusion only. Sensors for detecting intrusion are classified into two categories:

Direct Detection Sensors Sensors in this category usually monitor a specific item, for example a door, shutter, window, etc. These sensors are mounted directly onto the target. This category includes magnetic reed switches, glass break sensors, etc.

Indirect Detection Sensors Sensors in this category detect intrusions by monitoring a predetermined area. This type of sensor is mounted away from the target. This category includes photoelectric beams, passive infrared sensors, etc.

These sensors are used in conjunction with the following detection methods:

Direct Outer Detection Intrusion detection at the boundaries of a building by a direct detection sensor

Indirect Outer Detection Intrusion detection just inside building boundaries by an indirect detection sensor

Inner Detection Intrusion detection inside of a building.

Target Detection Intrusion detection by monitoring specific items.

The sensor planning procedure consists of the following four steps: 1) Target Identification, 2) Threat Analysis, 3) Sensor Selection, and 4) Sensor Positioning.

Target Identification

Rooms and doors which are especially vulnerable from the viewpoint of intrusion risks are identified in this task. For example all entrances to the building are potentially high risk. All targets identified here are candidates for monitoring by alarm equipment.

Threat Analysis

Depending on the room and door type and the location of the room, security levels are assigned to each room; these levels designate which detection method is required, as follows:

Level 0 No detection is necessary

Level 1 Direct outer detection should be applied

Level 2 Indirect outer detection should be applied

Level 3 Direct or indirect outer detection should be applied

Level 4 Inner detection and either direct or indirect outer detection should be applied

Threat analysis determines detection methods for each target depending on the security level of the adjacent rooms.

Due to physical constraints, it may not be possible to install the proper sensors for the detection method which the threat analysis determines as necessary. In such a case, the detection method assigned to the target will be changed to another method or shifted to adjacent rooms depending on the situation.

On the basis of these detection methods, specific sensing techniques, e.g. closure-confirmation, shutter-control, etc., are assigned to each target.

Sensor Selection and Positioning

Sensor selection determines the type of sensor which can accomplish the designated sensing technique. The possible selections are:

- Direct Detection Sensors
- Indirect Detection Sensors
- A contact embedded in electric locks
- A contact embedded in elevator doors

The CPP subsystem has declarative knowledge about sensing techniques for different types of sensors. For example, to correlate the sensor 'SHTxxx' with targets which require closure confirmation and shutter control sensing techniques,

```
    (selection-db DIRECT ENTRY-OBJECT ANY SHTxxx
    closure-confirmation shutter-control)
```

knowledge entry is declared.

The sensor positioning task determines the number of sensors required to monitor the targets and their coordinates within the building. The details are discussed in a later section.

Control Equipment Planning (CEP)

Equipment Selection

After sensor planning, the equipment required to integrate the sensors into an alarm system is introduced into the security plan, according to the following procedure:

1. Identify locations for equipment installation
2. Assign functional roles to each location to achieve desired system operation
3. Introduce system equipment which fulfills the functional role

Locations identified in the first step are as follows:

- inside and outside of the final exits of each zone
- building manager rooms
- rooms or spaces on each floor to keep the equipment out of view

Each location plays an important part in achieving alarm system operation. For example, a card reader is necessary near the exterior of a final exit to unarm the alarms in the zone and unlock the electric locking mechanism. The functional roles UNARMING and UNLOCKING can be assigned to the location. A device, in this case a card reader, can be placed at the location because it has the matching functional roles of UNARMING and UNLOCKING.

Equipment instances are created by the following procedures:

1. Determine the equipment model
2. Determine how much equipment is required for the situation
3. Determine installation location
4. Create equipment instances

Equipment Connections

System equipment introduced to the plan must be connected together for the system to function properly. SECOM ALARM SYSTEM incorporates a LAN for several devices, thus interconnections between those devices is relatively straightforward. However for other devices, the connections must be made in an arbitrary manner. In addition to the sensors, SECOM ALARM SYSTEM is able to control a portion of the building facilities (e.g. electric locks, automatic doors, shutters, elevators, etc.). For example, when a sensor detects a fire, all the electric locks in the building should be unlocked automatically to allow the occupants to evacuate. Since there are many variations in this kind of facility control, they are carried out through ad hoc connections. If all of these ad hoc connections are put into rules, the resulting program would be cumbersome. To avoid hard-coding the connection knowledge, simple case based reasoning (CBR) techniques are used to implement equipment connections. Therefore, we can maintain the connection knowledge separately from the program. The following is an example of a case.

```
(defcase case1
  "air-conditioning and light control"
  (deftype INTERLOCKING)

  (defcontext ZONE ANY-ZONE)
  (defcontext GOAL CONTROL-LIGHT)
  (defcontext GOAL CONTROL-AIR-CONDITIONING)

  (defactor *POT PO-T0210)
  (defactor *RLY RL-Y5700)
  (defactor *LIGHT MT-R0300)
  (defactor *AIR MT-R0310)

  (defrelation BLAN *POT)
  (imply (defrelation BINARY *POT *RLY)
         "set-signal")
  (defrelation BINARY *RLY *LIGHT)
  (defrelation BINARY *RLY *AIR)
  )
```

In this example, equipment to be connected is represented by the *defactor* statements and the connections are represented by the *defrelation* statements. Cases are transformed to instances and stored in working memory prior to reasoning. For equipment connections, the closest case is selected by case retrieval rules and a similarity function. Then the connection rules compare the selected case with the equipment instances introduced by the equipment selection rules and then establish the necessary connections. The resulting connections are presented in the system diagram (Figure 4).

In ESSPL, CBR is not a major inference mechanism, using only the case retrieval feature in the CBR paradigm. Thus some advanced features such as learning, case adaptation or maintenance are not employed.

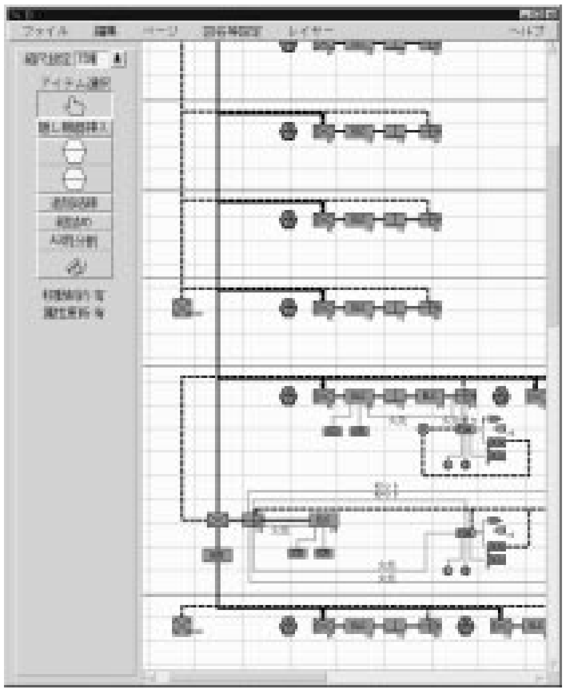

Figure 4: System Diagram Editor

Replanning Mechanism

Designing a building requires many changes, therefore security planning is a long term project. Whenever the design of a building is modified, the security plan must be updated to reflect the modification. Therefore, the replanning mechanism is an essential ESSPL feature. This mechanism must be invoked in the following three cases:

1. The building design is modified
2. Parts of a completed security plan is modified by the user
3. Part of a security plan is predefined by the user

For the first case, the plan must be updated as the building design changes. ESSPL deletes the portion of the plan which was affected by the modification and replans the location so that it is consistent with the rest of the plan.

In the second case, ESSPL removes the equipment which became unnecessary due to the user action. In this case, ESSPL does not introduce any new equipment to the plan, even if it is determined that it is required. This is sometimes required due to the terms of the customer contract. ESSPL is tolerant of some inconsistency arising from the user editing.

In the third case, ESSPL simply finishes the partially completed plan. The user input is considered as a constraint in this case.

In ESSPL, the replanning is handled by the DMP subsystem, shown in Figure 5. The user may stop the system during the execution, display and modify the partial solution, and resume the execution. Most likely the user intervention will conflicts with the partial plan that was being computed, therefore the DMP subsystem resolves the

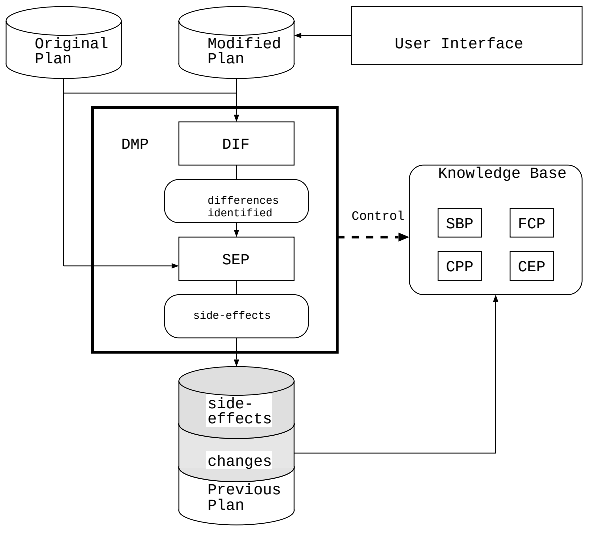

Figure 5: Replanning Mechanism

conflict automatically and attempts to keep the reasoning sound.

The DMP subsystem mediates between the user interface and the planning knowledge bases (SBP, CPP, FCP and CEP). It has two important roles:

1. Control the execution of planning knowledge bases
2. Prepare the proper input files for planning knowledge bases

The DMP subsystem consists of two parts, the DIF and SEP. The DIF compares the original plan to the user modified plan and identifies the changes which were made. These changes are translated into operators such as ADD, DELETE, CHANGE and HIDE.

The SEP identifies any side effects as a result of applying the operators to the original plan. The side effects are then translated into operators. This process is repeated until all side effects have been identified.

Finally, the DMP modifies and recreates the instances of the original plan on the basis of these changes and side effects. These new instances then become the input for the next execution of the knowledge bases.

Uses of AI Technology

Sensor Positioning by Geometric Constraints

In order to place the sensors, ESSPL must calculate the exact placement of the sensor in the security plan. In ESSPL, the Sensor Placement Subsystem (SPS) deals with this task. Since CLIPS does not have the capability to handle geometric or numeric constraints, a constraint logic programming language CONTA (Urushima 1993c; 1993a; 1993b) was developed to build the SPS.

In the case of indirect detection sensors, calculating their coordinates is difficult due to the number of geometric constraints to consider. For passive infrared sensors, we must consider the following constraints to find a suitable position:

1. Sensor must be placed inside of a room
2. All targets must be in its coverage area
3. Maintenance area is required
4. Coverage area should exclude windows
5. Placement near a corner of a room

To calculate the coordinates, all of these constraint must be considered and solved. The first and the second constraints are strong constraints, thus a solution must satisfy these two. On the other hand, the rest are weak constraints, thus a solution should satisfy them only if possible.

The SPS generates a spatial solution (coordinates) and checks whether the solution satisfies the above constraints. If the solution satisfies all of the constraints, it is adopted. Otherwise the geometric constraints are transformed to a set of numerical equations and solved by CONTA's constraint satisfaction capability.

CONTA : A Constraint Programming Language

CONTA is a general purpose constraint logic programming language designed to solve the sensor layout problem, developed on top of Prolog. CONTA can handle numeric equalities and inequalities as constraints like other Prolog clauses. A programmer needs only to state numerical equations with Prolog clauses in their program. Without giving the algorithm to solve these equations, CONTA automatically calculates a solution which satisfies the constraints by using the constraint solver (CS-Solver). CONTA has the following features:

- CONTA can handle linear equalities and inequalities
- CONTA has the capability to resolve conflicts between constraints
- CONTA looks for a solution which satisfies as many constraints as possible if no solution satisfies all constraints
- The programmer can set the strength of each constraint
- The programmer can state their preferences of feasible solutions with respect to the constraints.

These features allow CONTA to be useful for many engineering applications such as scheduling and layout problems. CONTA can handle the following three types of constraints.

1. **required constraint**

   All feasible solution must satisfy this constraint
2. **desired constraint**

   A feasible solution can may satisfy this constraint if possible. A solution which violates this constraints is still feasible.
3. **preferred constraint**

   If there is more than one feasible solution, the solutions are returned in the order preference as determined by this constraint.

These constraints are expressed in equalities and inequalities by using the following expressions:

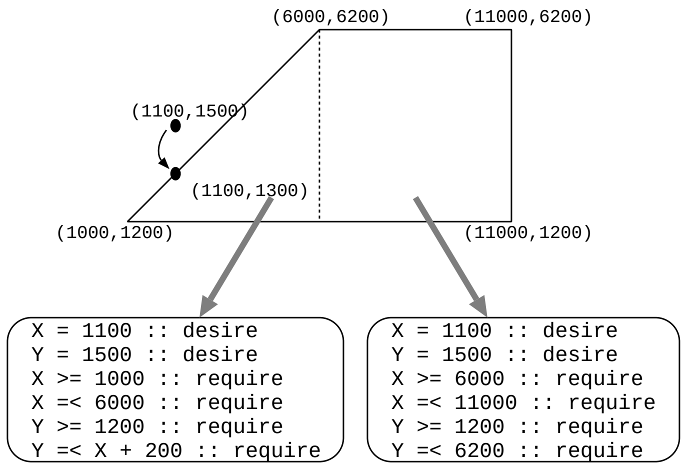

Figure 6: Positioning by CONTA

1. **<< : diverging inequality sign**

   Inequality with preference for a larger difference between the right hand side and left hand side

2. **~< : converging inequality sign**

   Inequality with preference for a smaller difference between the right hand side and left hand side

3. **=< : unrestricted inequality sign**

   Inequality with no preference

4. **= : equality sign**

   Equality

These constraints can also have weights. The programmer can declare the strength of the preference by setting a weight for each constraint. For example, if there are two conflicting constraints with the same weight, CONTA will generate a solution "in the middle" of the constraints. On the other hand, if one constraint has a higher weight than the other, the solution will be closer (in the case of converging inequality) to the constraint with the larger weight. The typical constraint

```
    X >> 30 :: desire 5
```

implies that the variable X should be greater than 30 if possible. The greater it is, the better.

The CS-Solver transforms these numerical constraints into a linear programming standard form and solves them using the Simplex algorithm. (Urushima 1993b)

An example of positioning a passive infrared sensor in a room is shown in Figure 6. The polygon represents the desired area to position the sensor considering the required maintenance area. Supposing the spatial solution (1100, 1500) failed to satisfy the inside-of-a-room constraint, CONTA generates the positioning constraints by considering the following:

1. The solution should be inside of the polygon
2. The solution should be close to the spatial solution

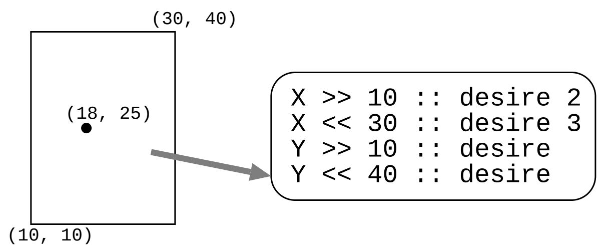

Figure 7: Example of diverging constraints

The polygon is divided into two simpler polygons and procedure is applied to each. The above constraints are transformed into numeric constraints as shown in the boxes. The CS-Solver solves these constraints and returns the best solution (1100, 1200).

The following is an example of CONTA code to achieve this. The coordinates are found by proving the CONTA goal conta_inside_room.

```
conta_inside_room
        (Point, [[XAREA, YFOOT, YTOP]|T], [X,Y]) :-
        conta_near(Point, X, Y),
        conta_inside_x(XAREA, X),
        conta_larger_y(YFOOT, X, Y),
        conta_smaller_Y(YTOP, X, Y)
        ;
        conta_inside_room(Point, T, [X, Y]).
conta_near([Xp, Yp], X, Y) :-
        X = Xp :: desire,
        Y = Yp :: desire.
conta_inside_x([Xmin, Xmax], X) :-
        Xmin =< X :: require,
        X =< Xmax :: require.
conta_larger_y([X1, Y1, X2, Y2], X, Y) :-
        A is (Y1 - Y2)/(X1 - X2),
        Y >= A * (X - X1) + Y1 :: require.
conta_smaller_y([X1, Y1, X2, Y2], X, Y) :-
        A is (Y1 - Y2)/(X1 - X2),
        Y =< A * (X - X1) + Y1 :: require.
```

The use of diverging inequalities can be seen when placing a smoke detector. A smoke detector must be placed away from any walls. Figure 7 shows how the weights attached to the constraints affect the solution (18, 25).

Evaluation

Tests on Real Data

The ESSPL execution has been evaluated using customer data. ESSPL has shown to be of significant advantage when generating security plans from scratch. On average, for a small sized, 7 story office building, a human engineer took approximately 8 hours to complete the plan. A plan with equivalent quality was achieved in 1.5 hours with ESSPL. It is worth noting that 95% of the time was spent constructing the input data. The execution time to actually generate the plan was less than five minutes. In a large sized 15 story building, ESSPL took 2.5 hours while a human engineer took 2 days. The actual execution time was less than 15 minutes.

The Experiences of Field Tests

Field tests were carried out in a security planning department which deals with large-scale security plans. In field tests, ESSPL proved advantageous for automatic planning. However few drawbacks were found in using ESSPL in the real world context.

1. A large effort was required to construct the input data by tracing the floor plans. Most of the engineers in the department felt that it was difficult to use and cumbersome.
2. The side effects of the replanning mechanism when the user edited the plan were unpredictable, thus the user often found undesirable side effects at the end of the replanning process.

Even though few drawbacks are exist, we can conclude that ESSPL is very useful for security planning. We are expecting to utilize it in the department after a few modifications.

Future Directions

After examination of the field test results, the following policies were devised:

1. Automatic conflict resolution of the user input is not useful in most cases, since the users tend to be puzzled by the unpredictable side effects. Instead of one function which does everything, smaller functions, e.g. addressing, zone planning, etc., which give the users finer control over the planning process will be provided.
2. Currently, there is no way to reduce the effort required to construct the input data necessary for automatic zone and sensor planning. Although these two functions are the primary tasks in security planning, we have abandoned the use of these functions at this time.

Recently, the International Alliance for Interoperability (IAI) has proposed the Industrial Foundation Class (IFC), which is a high-level common language for building complexes. We are expecting to receive most building data in the IFC format in the near future, at which point, the problem of constructing the input data will become negligible and the automatic planning function of ESSPL will have full play of its ability.

Acknowledgements

The authors wish to express our appreciation to Yasuyuki Tauchi, Muneaki Saitoh and Dr. ShinIchiro Hashimoto of SECOM Intelligent Systems Laboratory, for their support and encouragement. In particular, we would like to thank Janelle Jurek Kozak, Joshua Hornik, Linda Chen and Dr. Taesik Jeong for their work and dedication to the project. We are grateful to Yoshikazu Shinoda for reading the paper and providing many helpful suggestions.

References

Lamont, A. 1988. A computer model for identifying security system upgrades. In *Institute of Nuclear Materials Managment 29th Annual Meeting.*

Urushima, K. 1993a. A Constraint Logic Programming Language CONTA Ver 1.3 : Detailed Specification. Technical Report AI-93-014, SECOM Intelligent Systems Laboratory. in Japanese (unpublished).

Urushima, K. 1993b. A Constraint Logic Programming Language CONTA Ver 1.3 : Functional Specification. Technical Report AI-93-016, SECOM Intelligent Systems Laboratory. in Japanese (unpublished).

Urushima, K. 1993c. A Constraint Logic Programming Language CONTA Ver 1.3 : Outer Specification. Technical Report AI-93-013, SECOM Intelligent Systems Laboratory. in Japanese (unpublished).

Ward, J. D., and Sena, K. J. 1986. Senlex : Sensor Layout Expert System. In *Proceedings 1986 International Carnahan Conference on Security Technology: Electric Crime Countermeasures.*