PERGAMON

Expert Systems with Applications 19 (2000) 1–8

Expert Systems with Applications

www.elsevier.com/locate/eswa

Sufficient search space for spatial expert systems

Kyoung Jun Lee*

School of Business, Korea University, 1, 5Ka, Anam-dong, Sungbuk-ku, Seoul 136-701, South Korea

Abstract

This paper seeks the sufficient search space for the expert systems locating rectangular and arbitrary-shaped objects placed without rotation within a two-dimensional rectangular space. We found that for the layout of rectangular objects, the convex vertex set of feasible allocation space is a sufficient space to determine a feasible layout. We also found that for the layout of arbitrary-shaped objects, the boundary point set of the feasible allocation space is a sufficient space to determine a feasible layout. These two theorems are proved by developing two respective parallel translation algorithms. These theorems show that the search space can be significantly reduced in finding a feasible layout. Since these theorems were discovered while we were developing a spatial scheduling expert system, we have empirically tested the performance of the reduced search space with real world examples. According to the empirical test for the convex polygonal objects, the vertex set of feasible allocation space is satisfactory enough as a search space although the vertex set is not a sufficient space.

Keywords: Spatial layout; Search space; Spatial expert systems; Spatial planning

1. Introduction: search space in spatial layout expert system

Spatial layout problems (Adamowicz & Albano, 1976; Albano & Sapuppo, 1980; Roussel & Maouche, 1993) deal with the allocation of objects in the bounded rectangular space (denoted $\boldsymbol{w}$ in this paper). This paper investigates the sufficient search space required for the spatial layout expert system to sequentially locate objects in a bounded rectangular space $\boldsymbol{w}$, especially without rotation. The spatial layout problem with fixed orientation or a finite number of layout orientations frequently occurs in real world layout problems such as kitchen layout (Baykan & Fox, 1992; Honda, Yokoyama & Kato, 1992), multimedia presentation (Tanimoto, 1992; Tanimoto, 1993), microwave electronic equipment design (du Verdier, 1993), and floor planning (Honda and Mizoguchi, 1995). Since the cases with a small number of layout orientations (0, $\pi/2$, π, $3\pi/2$) can be regarded as the simple extension of the fixed orientation case, we will limit the scope of this paper to the fixed orientation.

In the fixed orientation case, the configuration space approach (Lozano-Perez, 1983) is an effective and efficient representation method. Though the approach is known to have some defects in path-planning problems (Crowley, 1987), it performs well for Find-Space problems with fixed orientation. The shaded region in Fig. 1 shows an example of the configuration space obstacle of a_i due to b_j and the shaded region of Fig. 2 is an example of the configuration space interior of a_i in $\boldsymbol{w}$. The two sorts of configuration space, configuration space obstacle and configuration space interior, correspond to the notion of dilation and erosion in mathematical morphology (Serra, 1986), and Minkowski addition and Minkowski decomposition (Ghosh, 1990), respectively. Using the configuration space approach, we can compute the *Feasible Allocation Space* of a reference point of an object a_i in $\boldsymbol{w}$ which includes the obstacles $\boldsymbol{B} = \{b_j | j = 1, \ldots, \boldsymbol{m},\}$ where objects b_js are already located on $\boldsymbol{w}$ as in Fig. 3.

Since the *Feasible Allocation Space* is a continuous space, as illustrated in the shaded area in Fig. 3, it is computationally impossible to search the infinite number of points within it. To implement an automated spatial layout system, the search space should be reduced by any means. An alternative for reducing the search space may be using the vertex set of the *Feasible Allocation Space*, but it is not certain whether the vertex set is sufficient for finding a feasible layout. Therefore, whether the vertex set of *Feasible Allocation Space* is sufficient as the search space of the spatial layout problem must be confirmed.

In this paper, we specifically attempt to answer the following two questions:

1. Is the vertex set sufficient as the search space of the spatial layout problem?

* Tel.: + 82-2-3290-1952; fax: + 82-2-922-4073.
E-mail address: leekj@kuba.korea.ac.kr (K.J. Lee).

0957-4174/00/$ - see front matter
PII: S0957-4174(00)00015-4

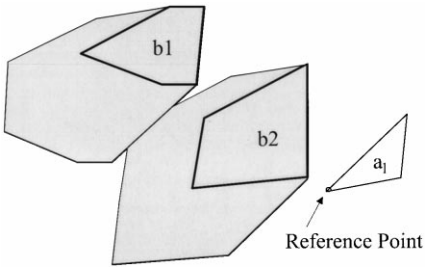

Fig. 1. Configuration space obstacle.

2. If it is not, what is a sufficient search space for the spatial layout problem?

We answer these by categorizing the shape of objects: rectangular or arbitrary.

1. For the rectangular object's layout, we found that the *Convex Vertex Set* (which will be defined in the next section) is a sufficient search space.
2. For the arbitrary-shaped object's layout, we found that the *Boundary Point Set* is a sufficient search space.

2. Classification of the points in feasible allocation space

The *Feasible Allocation Space* can be classified into the *Interior Point Set* and the *Boundary Point Set*. The *Boundary Point Set* can be further partitioned into the *Flat Point Set* and the *Vertex Set*. The vertices in the *Vertex Set* can branch to the *Convex Vertex Set* and *Concave Vertex Set*. Refer to the examples in Fig. 3. The classification tree is depicted in Fig. 4. The categories can be formally defined using the *Central Angle of the Feasible Direction Cone* (***CAFDC***(***p***)) notations. A feasible direction cone (Bazaraa & Shetty, 1979) is one with a very small radius from a feasible point ***p***.

Definition (Feasible point categories using the Central Angle of the Feasible Direction Cone).

- *Convex Vertex Set:* $\{\boldsymbol{p}|\boldsymbol{CAFDC}(\boldsymbol{p}) < \pi,\ \boldsymbol{p}$ is in $\boldsymbol{F}(\boldsymbol{a_i}|\boldsymbol{B},\boldsymbol{w})\}$ (e.g. point ***a*** in Fig. 5).
- *Flat Point Set:* $\{\boldsymbol{p}|\boldsymbol{CAFDC}(\boldsymbol{p}) = \pi,\ \boldsymbol{p}$ is in $\boldsymbol{F}(\boldsymbol{a_i}|\boldsymbol{B},\boldsymbol{w})\}$ (e.g. point ***b*** in Fig. 5).
- *Concave Vertex Set:* $\{\boldsymbol{p}|\pi < \boldsymbol{CAFDC}(\boldsymbol{p}) < 2\pi,\ \boldsymbol{p}$ is in $\boldsymbol{F}(\boldsymbol{a_i}|\boldsymbol{B},\boldsymbol{w})\}$ (e.g. point ***c*** in Fig. 5).
- *Interior Point Set:* $\{\boldsymbol{p}|\boldsymbol{CAFDC}(\boldsymbol{p}) = 2\pi,\ \boldsymbol{p}$ is in $\boldsymbol{F}(\boldsymbol{a_i}|\boldsymbol{B},\boldsymbol{w})\}$ (e.g. point ***d*** in Fig. 5).

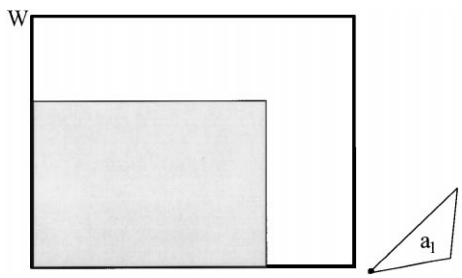

Fig. 2. Configuration space interior.

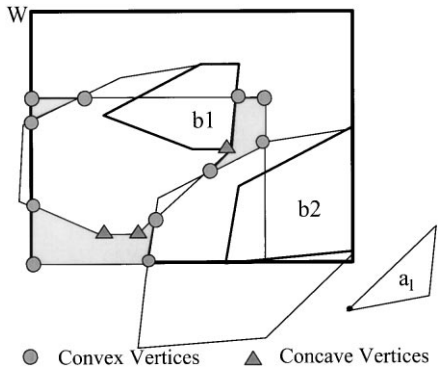

Fig. 3. An illustration of feasible allocation space $\boldsymbol{F}(\boldsymbol{a_i}|\boldsymbol{B},\boldsymbol{w})$ and its vertices.

In addition, using the configuration space concept, we can classify the vertex set as the four categories defined as follows:

Definition (Feasible vertex categories using configuration space).

- $\boldsymbol{FV_I}(\boldsymbol{a_i}|\boldsymbol{B},\boldsymbol{w})$: The set of *Feasible Vertices* of the configuration space *Interior* (e.g. small square in Fig. 6).
- $\boldsymbol{FI_{OI}}(\boldsymbol{a_i}|\boldsymbol{B},\boldsymbol{w})$: The set of *Feasible Intersection* points between the boundary of configuration space *Obstacles* and the boundary of configuration space *Interior* (e.g. small circle in Fig. 6).
- $\boldsymbol{FI_{OO}}(\boldsymbol{a_i}|\boldsymbol{B},\boldsymbol{w})$: The set of the *Feasible Intersection* points between the boundary of a configuration space *Obstacle* and the boundary of another configuration space *Obstacle* (e.g. small diamond in Fig. 6).
- $\boldsymbol{FV_O}(\boldsymbol{a_i}|\boldsymbol{B},\boldsymbol{w})$: The set of the *Feasible Vertices* of configuration space *Obstacles* (e.g. small triangle in Fig. 6).

In convex-shaped object layout, the *Convex Vertex Set* consists of $\boldsymbol{FV_I}(\boldsymbol{a_i}|\boldsymbol{B},\boldsymbol{w})$, $\boldsymbol{FI_{OI}}(\boldsymbol{a_i}|\boldsymbol{B},\boldsymbol{w})$, and $\boldsymbol{FI_{OO}}(\boldsymbol{a_i}|\boldsymbol{B},\boldsymbol{w})$ while the *Concave Vertex Set* corresponds to $\boldsymbol{FV_O}(\boldsymbol{a_i}|\boldsymbol{B},\boldsymbol{w})$.

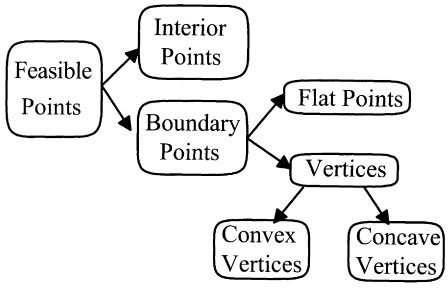

Fig. 4. Classification tree of feasible points.

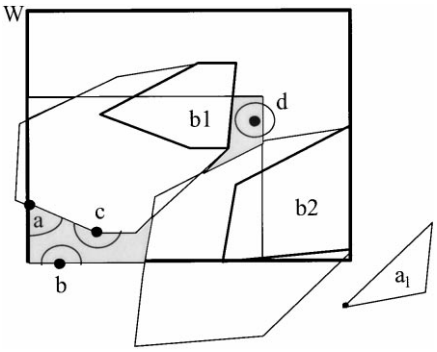

Fig. 5. Classification of feasible points using the feasible direction cone.

Property (Relations between categories in convex-shaped object layout).

Property 1. Convex Vertex Set $= \boldsymbol{FV_I}(\boldsymbol{a_i}|\boldsymbol{B},\boldsymbol{w}) \cup \boldsymbol{FI_{OI}}(\boldsymbol{a_i}|\boldsymbol{B},\boldsymbol{w}) \cup \boldsymbol{FI_{OO}}(\boldsymbol{a_i}|\boldsymbol{B},\boldsymbol{w})$.
Property 2. Concave Vertex Set $= \boldsymbol{FV_O}(\boldsymbol{a_i}|\boldsymbol{B},\boldsymbol{w})$.

3. Sufficient search space for rectangular object layout

Using the classification scheme in Section 2, the sufficient search space for a rectangular object layout will be proven.

3.1. Theorems

Theorem 1 (Sufficient search space for a rectangular object's layout). *If a feasible layout exists for a set of rectangles, at least one feasible layout can be found by a spatial layout procedure using the* Convex Vertex Set *of the* Feasible Allocation Space *as its search space*.

To prove Theorem 1, another theorem has been developed. If the following theorem is proven, the above theorem holds true.

Theorem 1.1. *If a feasible layout exists for a set of rectangles, proper translation of some of the rectangles converts the layout to a layout which can be constructed by a spatial layout procedure using the* Convex Vertex Set *of the* Feasible Allocation Space *as its search space*.

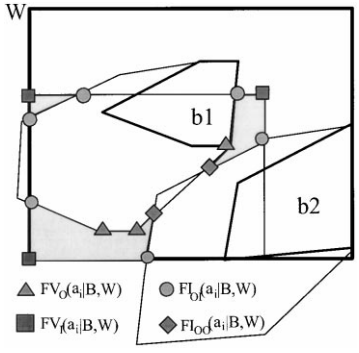

Fig. 6. Classification of feasible vertices into four categories.

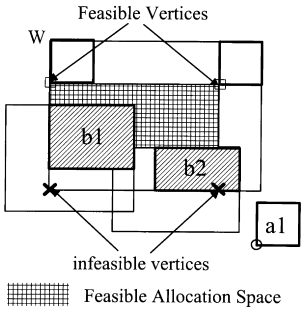

Fig. 7. Two locations in $\boldsymbol{FV_I}(a_1|\{b_1,b_2\},w)$.

We will prove Theorem 1.1 by suggesting the proper parallel translation procedure.

3.2. Characteristics of the convex vertex search and necessary definitions

Before suggesting the translation procedure, we need to clarify the meaning of the spatial layout procedure using the *Convex Vertex Set* of the *Feasible Allocation Space* as its search space in the rectangular object layout. In this paper, we call the spatial layout procedure as *Convex Vertex Search.*

In the rectangular object layout, an object located at a convex vertex must satisfy the following two adjacency conditions simultaneously:

Adjacency condition of an object located at convex vertex in rectangular object layout

1. The upper or lower edge of an object a_i located at a convex vertex is adjacent to an edge of b_j or an edge of $\boldsymbol{w}$.
2. The right or left edge of an object a_i located at a convex vertex is adjacent to an edge of b_j or an edge of $\boldsymbol{w}$.

$\boldsymbol{FV_I}(\boldsymbol{a_i}|\boldsymbol{B},\boldsymbol{w})$, $\boldsymbol{FI_{OI}}(\boldsymbol{a_i}|\boldsymbol{B},\boldsymbol{w})$, and $\boldsymbol{FI_{OO}}(\boldsymbol{a_i}|\boldsymbol{B},\boldsymbol{w})$, which exclusively and exhaustively compose the convex vertex set (since every rectangle has a convex shape) as explained in Property 1 of Section 2, satisfy these adjacency conditions, respectively:

Adjacency conditions of locations at convex vertex

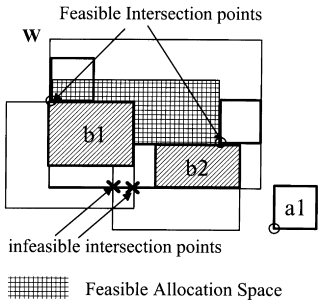

Fig. 8. Two locations in $\boldsymbol{FI_{OI}}(a_1|\{b_1,b_2\},w)$.

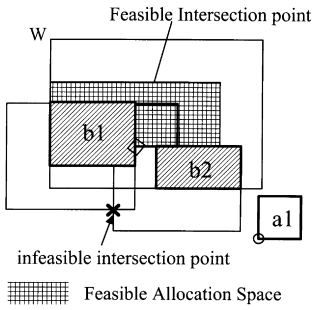

Fig. 9. One location in $\boldsymbol{FI_{OO}}(a_1|\{b_1, b_2\}, w)$.

1. The locations at $\boldsymbol{FV_I}(\boldsymbol{a_i}|\boldsymbol{B}, \boldsymbol{w})$ are adjacent to two edges of $\boldsymbol{w}$ (Fig. 7).
2. The locations at $\boldsymbol{FI_{OI}}(\boldsymbol{a_i}|\boldsymbol{B}, \boldsymbol{w})$ are adjacent to one edge of $\boldsymbol{w}$ and one $\boldsymbol{b_j}$ (Fig. 8).
3. The locations at $\boldsymbol{FI_{OO}}(\boldsymbol{a_i}|\boldsymbol{B}, \boldsymbol{w})$ are adjacent to two $\boldsymbol{b_j}$s (Fig. 9).

In summary, the *Convex Vertex Search* in rectangular object layout means locating the rectangles while maintaining the *Adjacency Condition*. For the convenience of the proof, we regard $\boldsymbol{w}$ as bordered by four virtual exterior rectangles labelled, respectively, as ***nr*** (north rectangles), ***sr*** (south rectangles), ***wr*** (west rectangles), and ***er*** (east rectangles) as shown in Fig. 10. Such a technique is also used in Flemming (1986). In addition, we found that the parallel translation algorithm can be achieved using two directions: the upper and the right direction.

Next, we have defined the following concepts which will be used in the algorithm.

Definition (UR (upper rectangles), RR (right rectangles), and FR (fixed rectangles)).

- $\boldsymbol{UR}(\boldsymbol{r})$: the set of the *Rectangles* adjacent to the *Upper* edge of $\boldsymbol{r}$ (for example, in Fig. 10, $\boldsymbol{UR}(r5) = \{r1, r2\}$, $\boldsymbol{UR}(r1) = \varnothing$).
- $\boldsymbol{RR}(\boldsymbol{r})$: the set of *Rectangles* adjacent to the *Right* edge of $\boldsymbol{r}$ at a spatial status (for example, in Fig. 10, $\boldsymbol{RR}(r5) = \{r3, r4\}$, $\boldsymbol{RR}(r3) = \varnothing$).
- ***FR***: the set of ***nr***, ***wr***, ***sr***, ***er*** and the rectangles satisfying the following two conditions:
  1. The rectangle $\boldsymbol{r}$ has both $\boldsymbol{UR}(\boldsymbol{r})$ and $\boldsymbol{RR}(\boldsymbol{r})$.

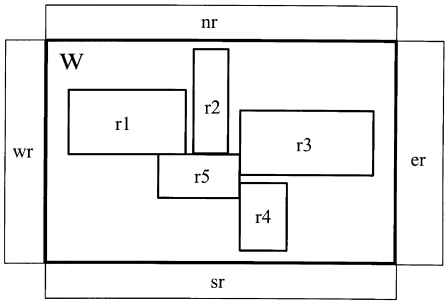

Fig. 10. ***nr***, ***sr***, ***wr***, ***er***, and $\boldsymbol{UR}(r5) = \{r1, r2\}$, and $\boldsymbol{RR}(r5) = \{r3, r4\}$.

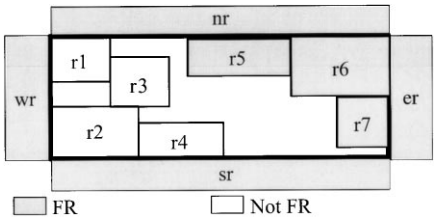

Fig. 11. Explanation of ***FR***.

2. At least one element of the $\boldsymbol{UR}(\boldsymbol{r})$ and at least one element of the $\boldsymbol{RR}(\boldsymbol{r})$ belong to the ***FR***.

Readers should note that the rectangles $r1$ and $r2$ in Fig. 11 do not belong to the ***FR***. Though $r1$ has both the $\boldsymbol{UR}(r1) = \{nr\}$ and the $\boldsymbol{RR}(r1) = \{r3\}$, the only element $r3$ of the $\boldsymbol{RR}(r1)$ does not belong to the ***FR***. Therefore, $r1$ cannot belong to the ***FR***. Neither can $r2$, as seen in Fig. 11.

3.3. The parallel translation algorithm

Using these constructs, we have developed the following algorithm, which translates all rectangles of a layout to those belonging to the ***FR***. To make all the rectangles belong to the ***FR***, this algorithm should first make all rectangles posses, their own ***UR*** and ***RR***. In other words, this algorithm keeps rectangles moving until they cannot further move to the upper or right directions.

Parallel translation algorithm in rectangular object layout

begin
  FR ← *the rectangles satisfying the conditions of FR*
  while *a rectangle r exists s.t.* $UR(r) = \varnothing$ *or* $RR(r) = \varnothing$
    if $UR(r) = \varnothing$ */ * if r can move up * /*
    then *Translate r to the upper direction to be adjacent to an object*
    if $RR(r) = \varnothing$ */ * if r can move right * /*
    then *Translate r to the right direction to be adjacent to an object*
    if $(UR(r) \cap FR) \neq \varnothing$ and $(RR(r) \cap FR) \neq \varnothing$
      */ * r satisfies the condition of the FR * /*
    then $FR \leftarrow FR \cup \{r\}$
end

It is apparent that the above algorithm makes all the

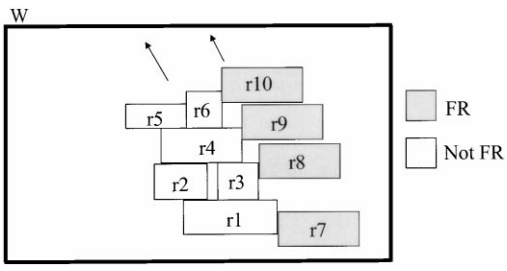

Fig. 12. An illustration for proof by contradiction.

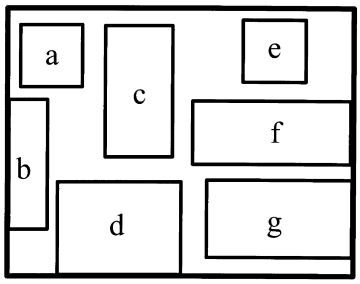

Fig. 13. A final layout.

rectangles have their own ***UR*** and ***RR***. Now, for the completeness of the algorithm, we have only to prove that all rectangles become ***FR*** once they possess their ***UR***s and ***RR***s. The proof is carried out using *proof by contradiction* as follows.

3.4. Proof of the completeness of the algorithm by contradiction

Let us assume that at least one rectangle exists which does not belong to ***FR*** when all rectangles have their ***UR*** and ***RR***. Let us call the rectangle r (for example, $r1$ and Fig. 12, whose right edge is the rightmost among the rectangles). Then all rectangles in ***RR***(***r***) must belong to ***FR***, but at least one ***UR***(***r***) ($\{r2, r3\}$ in Fig. 12) must not belong to ***FR*** (since r is assumed not to belong to ***FR***). At this time, the rectangle whose right edge is the rightmost among those of ***UR***(***r***) is set as the new r (for example, $r3$ in Fig. 12). In this way, we can select new rectangles endlessly ($r1 \rightarrow r3 \rightarrow r4 \rightarrow r6 \rightarrow \cdots$), which contradicts the basic premise that there is a finite number of rectangles in ***w***.

We have shown that the parallel translation algorithm can make all rectangles belong to ***FR***. In addition, if we position the rectangles according to the sequence and position of entering into the list of ***FR***, then the resulting spatial layout becomes one of the solutions that can be obtained by the *Convex Vertex Search*. In other words, if we try to determine all feasible layouts by considering only the convex vertices, then we can find a layout generated by the parallel translation algorithm. Therefore, now, we prove Theorem 1. For example, the layout in Fig. 13 can be translated to the layout in Fig. 14, which the *Convex Vertex Search* can generate by positioning sequentially *e, f, c, a, b, g, and, d*.

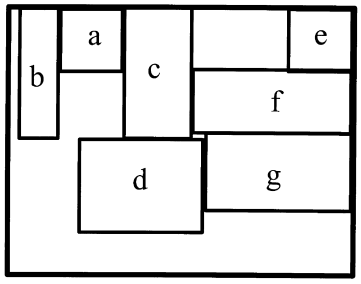

Fig. 14. The translated layout.

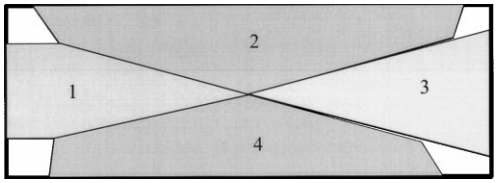

Fig. 15. In order to locate all four objects, the first one should be located at a flat point.

4. Sufficient search space in arbitrary-shaped object layout

4.1. A counter-example and expansion of the search space

With a simple counter-example, we are able to see that the *Convex Vertex Set* of the *Feasible Allocation Space* is not the sufficient search space in an arbitrary-shaped object layout. For example, when locating the four objects in Fig. 15, it is impossible to locate all the objects by considering only the *Convex Vertex Set* as its search space, because in order to do so, the first object should be located at a *Flat Point* of its *Feasible Allocation Space*.

Now the scope of the sufficient search space should be expanded to the space including *Flat Points*. So, we suggest a theorem with the *Boundary Point Set* as the search space.

Theorem 2 (Sufficient search space in arbitrary-shaped object layout). *If a feasible layout exists for a set of objects, then a feasible layout can be found by the spatial layout procedure using the* Boundary Point Set *as its search space*.

Like in Section 3, we develop another theorem to prove the above theorem. If we prove the following theorem, the above theorem holds true.

Theorem 2.1. *If exists a feasible layout for a set of objects, then, by properly translating some of the objects in the layout, it can be converted to a layout that can be constructed by a spatial layout procedure using the* Boundary Point Set *as its search space*.

4.2. Characteristics of the boundary point search and necessary definitions

We will prove Theorem 2.1 by suggesting a simple and intuitive translation procedure. Locating an object at its

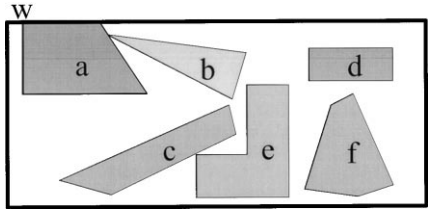

Fig. 16. A layout.

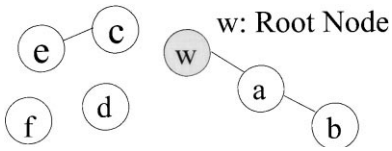

Fig. 17. The adjacency graph of Fig. 16.

feasible boundary point means locating it adjacent to another object (including the object representing the edges of ***w***). We call such a search as the *Boundary Point Search*. We classify the objects at a spatial status according to its adjacency to other objects.

Definitions of two object sets at a layout status

1. ***IO*** (*Isolated Objects*): The set of objects isolated from (i.e. not adjacent to) any other object (for example, {*d*, *f*} in Fig. 16).
2. ***AO*** (*Adjacent Objects*): The set with ***w*** and objects adjacent to another object (for example, {*a*, *b*, *c*, *e*, ***w***} in Fig. 16).

The adjacency status of objects in a layout can be represented with the adjacency graph. Fig. 17 is the graph representation of the spatial layout in Fig. 16.

As seen in Fig. 17, the objects in ***AO*** compose connected graphs. Among them, only one connected graph has the node representing ***w*** (called as *Root Node*) and we name it ***RCG*** (*Root Connected Graph*). The other connected graphs which do not include the *Root Node* can be called ***RIG*** (*Root Isolated Graph*).

Definition (Root Connected Graph, Root Isolated Graph, and Root Isolated Graph Set).

1. ***RCG*** (*Root Connected Graph*): The connected graph which has the *Root Node*. In other words, at least one object of which is adjacent to an edge of ***w***, and each object of which is adjacent to another object in the graph (e.g. {***w***, *a*, *b*} in Fig. 16).
2. ***RIG*** (*Root Isolated Graph*): The connected graph, any object of which is not adjacent to the edges of ***w***, but to another object in the graph (e.g. {*e*, *c*} in Fig. 16). The set of ***RIG*** is called ***RIGS*** (*Root Isolated Graph Set*) and is represented as {{*e*, *c*}}.

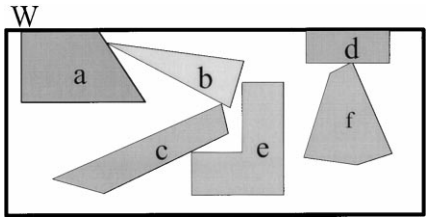

Fig. 18. A translated layout.

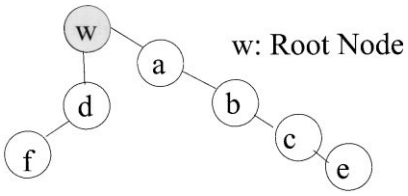

Fig. 19. The adjacency graph of Fig. 18.

4.3. The parallel translation algorithm

The following simple algorithm converts a layout to one in which only the ***RCG*** exists (Fig. 18). In this algorithm, we restrict the direction of the parallel translation to only one direction: the upper direction. In step 2, all isolated objects are moved adjacent to another object. Then IO becomes a null set. In step 4, all root isolated graphs are moved adjacent to the root node.

The parallel translation algorithm in arbitrary-shaped object layout

Begin
Step 0. Initialize IO, RCG, and RIGS by definition
Step 1. **if** $IO = \varnothing$ **then** *go to step 3*;
Step 2. **for** OBJ_i *in IO* **do**
  Translate OBJ_i *to the upper direction to be adjacent to any object*;
  $IO \leftarrow IO - \{OBJ_i\}$;
  $OBJ_j \leftarrow$ *one of the objects adjacent to* OBJ_i;
  if $OBJ_j \in IO$
  then $RIGS \leftarrow RIGS \cup \{\{OBJ_i, OBJ_j\}\}$;
    $IO \leftarrow IO - \{OBJ_j\}$;
  else if $OBJ_j \in$ *a graph in RIGS*
    then $GRP1 \leftarrow$ *the graph which* OBJ_j *belongs to*;
      $GRP1 \leftarrow GRP1 \cup \{OBJ_i\}$;
    else / * OBJ_j *belong to RCG* * /
      $RCG \leftarrow RCG \cup \{OBJ_i\}$;
Step 3. **if** $RIGS = \varnothing$ **then** *go to step 5*;
Step 4. **for** GRP_i *in RIGS* **do**
  Translate GRP_i *to the upper direction to be adjacent to any graph*;
  $RIGS \leftarrow RIGS - \{GRP_i\}$;
  $GRP_j \leftarrow$ *one of the graphs adjacent to* GRP_i;
  if $GRP_j \in RIGS$
  then $RIGS \leftarrow RIGS - \{GRP_j\}$;
    $RIGS \leftarrow RIGS \cup \{GRP_i \cup GRP_j\}$;
  else / * GRP_j *belongs to RCG* * /

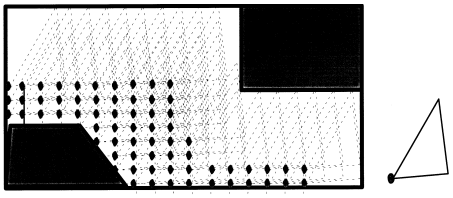

Fig. 20. The grid points.

Table 1
Comparison of the Vertex Search and Grid Point Search

Experiment no.	Search space	Average run time (h:min)	Average tardy activities	Backtracking frequency	Dominated by
1	Grid points	7:47	48	1052	nos. 2, 3
2	Vertices	3:18	30	605	
3	Grid + Vertices	4:27	28	448	

$RCG \leftarrow RCG \cup GRP_i$;
Step 5. Terminates
End

The adjacency graph of objects in the resulting ***RCG*** shows the adjacency between objects and the edges of ***w***. If we sort the nodes of the graph by ascending order of the distance from ***w***, then the resulting sequence becomes one of the spatial layout sequences by the *Boundary Point Search*. In Fig. 19, the sequence is *a–d–b–f–c–e*.

5. Practical usefulness of the vertex set in real application

In Section 4, we show that the *Convex Vertex Set* of the *Feasible Allocation Space* is not the sufficient search space in arbitrary-shaped object layout. However, it is still impossible to search all feasible boundary points in real world applications. An alternative to the boundary points is using the vertex set of the *Feasible Allocation Space*, since the vertices can be thought of as the representative and distinctive points of the feasible space. This section explains our experience in experimental results using the vertices in the spatial scheduling system (Lee, Lee & Choi, 1996) for a shipbuilding plant.

We compared the performance of the search using the vertex set with that of a pseudo-full search. To imitate a full search, we devised a grid search method whose search space consists of hundreds of discrete grid points in the *Feasible Allocation Space*, as shown in Fig. 20. We compared the three search-space generation methods: grid points, feasible vertices, and combination of grid points and vertices. The combined point search has the largest search space among the three alternatives, because it includes both the grid points and distinctive points. The empirical tests were carried out using the real data for a four-month spatial schedule (spatial scheduling is a scheduling problem that pursues the optimal dynamic spatial layout schedule that can also satisfy traditional scheduling constraints). There were 2601 objects in the data set, which were all convex polygons. Table 1 summarizes the results of the experiments. The vertex search outperformed the grid point search with respect to all criteria of computation time, number of tardy activities, and backtracking frequency. Between the vertex search method and the combined grid-and-vertex search method, there were trade-offs between time and performance. With the results, we conclude that the vertex set is useful in practical applications in spite of it not theoretically being sufficient search space. For the choice of a good position, given each search space, we employed five heuristic spatio-temporal allocation strategies devised for the shipbuilding domain (Lee et al., 1996) and the average space utilization ratio of the spatial scheduling system was about 75%.

6. Related and future research

Our findings in a Find-Space problem are related with the result from the motion planning field, which is a Find-Path problem, having started with the seminal paper of Reif (1979) on the piano mover problem. In motion planning, it has been known that if there is a path for moving an object through space, there is also a path along the boundary of the space. In this paper, we could confirm that the same principle holds true for placing an object, as well as for finding a path.

In this paper, we address only the existence problem without considering solution quality, which may be critical to real world applications. However, since the evaluation of solution quality is quite domain dependent, an investigation on solution quality should consider the domain-specific features. For example, in a dynamic spatial layout problem such as the spatial scheduling problem in shipbuilding, we have developed heuristic strategies, which were very successful in the field (Lee et al., 1995).

In relation to the solution quality problem, a future research direction we consider is applying the present result to the research on domain specific spatial constraint representation and propagation techniques such as du Verdier (1993), Honda and Mizoguchi (1995), and Tanimoto (1993) for improving the efficiency of layout generation.

Acknowledgements

The work upon which this paper is based was supported in part by the Korea Science and Engineering Foundation.

References

Adamowicz, M., & Albano, A. (1976). Nesting two-dimensional shapes in rectangular modules. *Computer Aided Design, 8* (1).
Albano, A., & Sapuppo, G. (1980). Optimal allocation of two-dimensional

irregular shapes using heuristic search methods. *IEEE Transactions of System, Man, and Cybernetics*, *10* (5).

Baykan, C., & Fox, M. (1992). *WRIGHT: a constraint based spatial layout system, Artificial Intelligence in Engineering Design*. New York: Academic Press.

Bazaraa, M., & Shetty, C. (1979). *Nonlinear programming*. New York: Wiley.

Crowley, J. (1987). Path planning and obstacle avoidance. In S. Shapiro, *Encyclopedia of Artificial Intelligence*, vol. 2. New York: Wiley-Interscience.

du Verdier, F. (1993). Solving geometric constraint satisfaction problems for spatial planning. In: Proceedings of 93' International Joint Conference on Artificial Intelligence.

Flemming, U. (1986). On the representation and generation of loosely packed arrangements of rectangles. *Environment and Planning B: Planning and Design*, *13*.

Ghosh, P. (1990). A solution of polygon containment, spatial planning, and other related problems using Minkowski operations. *CVGIP (Computer Vision Graphics and Image Processing)*, *49*.

Honda, K., Mizoguchi, F. (1995). Constraint-based approach for automatic spatial layout. In: Proceedings of the 11th Conference on Artificial Intelligence for Applications.

Honda, K., Yokoyama, T., Kato, C. (1992). A knowledge-based design system for the restaurant kitchen layout. In: Proceedings of '92 The Second Pacific Rim Conference on Artificial Intelligence, Seoul, South Korea.

Lee, J., Lee, K., Hong, J., Kim, W., Kim, E., Choi, S., Kim, H., Yang, O., & Choi, H. (1995). DAS: intelligent scheduling systems for shipbuilding. *AI Magazine*, *16* (4).

Lee, K., Lee, J., & Choi, S. (1996). A spatial scheduling system and its application to shipbuilding: DAS-CURVE. *Expert Systems with Applications*, *10* (3/4).

Lozano-Perez, T. (1983). Spatial planning: a configuration space approach. *IEEE Transactions on Computers*, *32* (2).

Reif, J. (1979). The complexity of the movers problem and generalizations. In: Proceedings of IEEE Symposium on Foundations of Computer Science.

Roussel, G., Maouche, S. (1993). Improvements about automatic lay-planning for irregular shapes on plain fabric. In: Proceedings of IEEE Conference on Systems, Men, and Cybernetics.

Serra, J. (1986). Introduction to mathematical morphology. *CVGIP (Computer Vision Graphics and Image Processing)*, *35*.

Tanimoto, T. (1992). Configuring multimedia presentations using default constraints. In: Proceedings of the Second Pacific Rim International Conference on Artificial Intelligence.

Tanimoto, T. (1993). A constraint decomposition method for spatio-temporal configuration problems. In: Proceedings of National Conference on Artificial Intelligence.

Kyoung Jun Lee is an assistant professor of the School of Business in Korea University. He was a senior researcher at the International Center for Electronic Commerce (ICEC). He received his BS (1990), MS (1992), and PhD (1995) in management science from KAIST. From 1996 to 1997, he worked as a visiting scientist of the Robotics Institute of the Carnegie Mellon University. He has won the Innovative Applications of Artificial Intelligence (IAAI) Award twice, in 1995 and 1997. He has published papers in AI Magazine, Expert Systems with Applications, and European Journal of Operational Researches. His current research interests are on developing intelligent information systems for electronic commerce, supply chain management, and planning/scheduling problems, etc.