Air-Pressure-Model-Based Fast Algorithms for General Floorplan

Tomonori IZUMI†, Atsushi TAKAHASHI and Yoji KAJITANI

Department of Electrical and Electronic Engineering, Tokyo Institute of Technology

Ookayama, Meguro, Tokyo, 152–8552 Japan

TEL: +81-3-5734-2665   FAX: +81-3-5734-2902

E-mail†: chiron@ss.titech.ac.jp

Abstract— A new approach for the minimum area floorplanning is proposed where the shape of every module can vary under the constraint of area and floorplan topology. Simulating the air-pressure mechanics, the algorithms iterate to improve the layout to decide the shapes and positions of modules. It is proved that the layout approaches the optimal layout each step by the measure of energy which is defined by the current layout. Experimental results showed very fast convergence. An extension to a more practical case with the aspect-ratio constraint is discussed.

I. Introduction

In the field of VLSI designing, the stage known as floorplanning determines the total performance of VLSI circuits. In floorplanning, a *floor-rectangle* is first partitioned into subrectangles, each of which is associated with a module. At this time, the constraint of each subrectangle and the relation of subrectangles are determined. However, the size of the floor-rectangle and the concrete region of each subrectangle within the floor-rectangle are not yet determined. Among the various optimization targets, the primary goal, and the subject of the present paper, is the minimization of the area of the resultant layout.

The floor-rectangle is partitioned by a set of horizontal and vertical line segments which do not intersect each other but which end at the orthogonal segments. These line segments, including the boundaries of the floor-rectangle, are here called the *walls*. Where it is necessary to distinguish among walls, the four boundary walls of the floor-rectangle will be referred to as *outer walls* and the others as *inner walls*. A maximal subregion containing no inner walls forms a rectangle referred to as the *room*. An example is shown in Fig. 1.

A minimum area layout is pursued under the two types of constraints, the topological and physical constraints.

This work has been supported in part by the Research Body of CAD21 at Tokyo Institute of Technology.

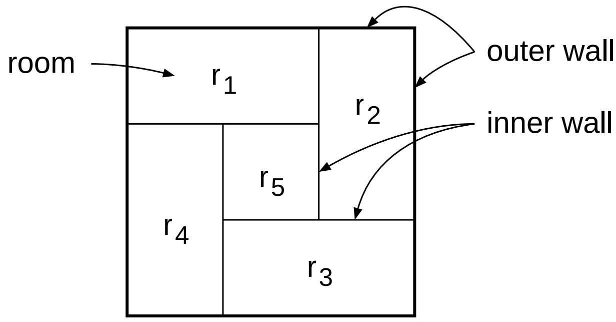

Fig. 1. Definitions of floor-rectangle, rooms, and walls.

The topological constraint is the incidence relation between walls and rooms. A wall and a room are said to be *incident* if the wall is a boundary of the room. The incidence relation is significant for its practical implication: A wall in a layout is considered to represent a straight channel in routing. Ideally, a floorplan will be designed so that the terminals of those modules that are incident to a wall will be easily connectable via the channel. Therefore, we impose the constraint of an invariant incidence relation.

The physical constraint is that a room must be large enough to embed the associated module. This constraint can be variously interpreted according to the style of module. The width and height of modules may be allowed to vary subject to the area or aspect-ratio, or may be defined by the designer. Typical examples are *discrete modules* such that a module is selected from several functionally equivalent candidates ([1, 2, 3, 4, 5] for slicing structures, [6, 7, 8, 9, 10, 11] for hierarchical structures, or [12, 13] for general structures), *continuous modules* such that the width and height of a module are arbitrary subject to the area ([14, 15]), and *bounded continuous modules* such that the width and height of a module have their preassigned lower bounds ([16, 17]). In this paper, we focus on the continuous modules, and thus define the physical constraint such that the area of a room is sufficiently large to embed the associated module area. At the later part of this paper, we also give a discussion on the bounded continuous modules which is equivalent to the modules with the aspect-ratio constraints.

A basic but nontrivial question in the case of the continuous modules is whether or not there exists a layout the area of which is the sum of the areas of the modules. Such a layout is called the *zero-wasted-area layout* [14]. If the floorplan is a slicing structure, the answer is obviously yes. A zero-wasted-area layout is easily obtained by a finite sequence in which linear equations are solved one at a time. As regards the general structures, the most important contribution has been by Wimer, Koren, and Cederbaum [14], who proved the existence of a zero-wasted-area layout through analysis of the solution space. Several subsequent algorithm studies succeeded in realizing the zero-wasted-area layout. Wang and Chen [15], for example, proposed a network analogous approach.

However, their approaches are all global; that is, their solutions use the global property. As typically seen in Wang and Chen [15, 16] or in Moh, Chang, and Hakimi [17], the algorithms collect all the constraints in a list which may result in a huge system of equations and solve the system of equations. It is true in general that a global approach involves an enormous number of computations. It is strongly hoped that a local, or greedy, approach with some global optimality will be developed.

This paper proposes such an approach. We begin with a layout, which may not satisfy the area constraint, then simulate the air-pressure mechanics, iterating the movement of one wall at a time to the force-balancing position while maintaining the incidence relation. The distance of a movement is determined by simulated air-pressure of the rooms that are incident to the wall. By this movement, the pressures of these rooms are induced to change. Accordingly, the force of some walls may increase. Thus it appears that a layout is not approaching a zero-wasted-area. But the idea of *energy* is introduced to observe the degree of resemblance of the current layout to the zero-wasted-area layout. It is proved that the movement of a wall monotonically reduces the energy and that the energy is zero if and only if the zero-wasted-area layout is attained.

The algorithm is evolved to methods which move all the walls simultaneously, resulting in a further speed enhancement with a sacrifice of the above mentioned guarantee. The experimental results revealed that our algorithms are too fast to be measured by the instance in the literatures [14, 15, 17] that consists of only twenty modules. A high performance of the algorithm was shown by further experiments for artificially made, complex instances including hundreds of modules.

Since the extremely thin module is hard to be implemented, the aspect-ratio constraint is assigned to each module in practice. Moh, Chang, and Hakimi [17] obtain the optimal layout by such a way that the problem is transformed to the convex programming problem and then solved by the convex optimizing technique. In order to handle the aspect-ratio constraints in the air-pressure model, a prop bar is introduced. A prop bar represents the lower bound of the width or height of a module. Several ideas and algorithms for this extended model are presented.

The rest of this paper is organized as follows. Section II is devoted to a sketch of the principle of the simulated air-pressure model. In Section III, the basic algorithm and the guarantee on it are presented. The algorithm is enhanced in Section IV. The experiments in Section V show the efficiency of the proposed algorithms. In Section VI, we discuss about the extension for our air-pressure model to handle the aspect-ratio constraint. Finally, Section VII concludes this paper.

II. Definitions and Sketch of Principle

A floorplan has rooms $r_1, r_2, \ldots, r_n$. The width and height of room r_ℓ of the layout are x_ℓ and y_ℓ, respectively. The area constraint is $x_\ell y_\ell \geq a_\ell$ $(\ell = 1, \ldots, n)$ where a_ℓ is the area of the associated continuous module. The width and height of the floor-rectangle are X and Y, respectively.

Looking at the floorplan instance shown in Fig. 2, we first determine the positions of the three inner walls w_1, w_2, and w_3 when $XY = a_1 + a_2 + a_3 + a_4$ is satisfied. Accordingly, the width x_ℓ and height y_ℓ of each room r_ℓ are determined. Since the topology is a slicing structure, the solution is easily obtained:

$$x_1 = \frac{a_1}{a_1 + a_2 + a_3} X,\ y_1 = \frac{a_1 + a_2 + a_3}{a_1 + a_2 + a_3 + a_4} Y.$$

The formula is global in the sense that it cannot be calculated without knowing all the physical and topological constraints.

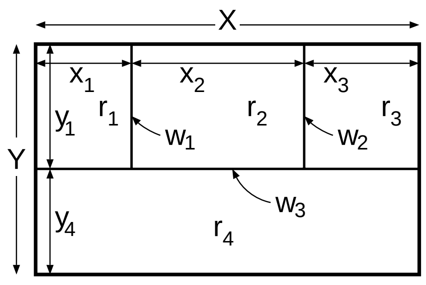

Fig. 2. A sample floorplan instance of a slicing structure.

For the case of a "spiral" as shown in Fig. 1, a global approach requires that a 2nd order equation is solved. For example, x_1 is a positive solution of: $(1 - a_3 - a_4)x_1^2 + (a_1a_3 - a_2a_4 + a_1 - a_2 - a_3 + a_4 - 1)x_1 + (-a_3 - a_4 - 1)a_1 = 0$ assuming without loss of generality that $X = Y = 1$. A more complex floorplan will provide higher order equations. For example, for a floorplan instance with 14 rooms, there are two equations of two variables on a maximum 20th order.

In contrast, our approach uses a single formula at a time in terms of local values. It starts with an initial layout such that X and Y are arbitrary subject to $XY = \sum_{\ell=1}^{n} a_\ell$ and improves the layout at each step. Topological constraint is maintained in the process, but physical constraint may be violated.

We simulate the natural phenomenon of air-pressure inflicting a force to the walls according to Pascal's principle, each wall thus moving toward the force-balancing position. Area a_ℓ represents the quantity of air sealed in room r_ℓ of volume $x_\ell y_\ell$. The *pressure* of room r_ℓ is given by

$$P_\ell = \frac{a_\ell}{x_\ell y_\ell}.$$

The force from the room r_ℓ to the wall w_i is proportional to the length of the part of w_i shared with r_ℓ. Take a vertical wall w_i, which will receive force from the rooms that are incident to w_i from the right or left. The algebraic sum rightwards is the force that tends to move w_i to the right. Let R_i^- and R_i^+ be a set of rooms adjacent from the left and right of w_i, respectively. The *force* F_i of w_i is defined by

$$F_i = \sum_{r_\ell \in R_i^-} P_\ell y_\ell - \sum_{r_\ell \in R_i^+} P_\ell y_\ell.$$

The force to a horizontal wall is defined similarly.

III. Basic Algorithm with a Guarantee

The operation called the *force-balancing* applied to an inner wall moves it to the position where the force is zero while maintaining the position of the other walls. We have the following lemma on the operation.

Lemma 1 *For any wall, there is a unique force-balancing position preserving the topology.*

The proof of this lemma and those that follow are omitted for the space (See [18]). Furthermore, the force-balancing position can be found efficiently within the prescribed precision, for example, by binary search. The basic version of our algorithm *Simulated-Air-Pressure-Single (SAPS)* is described in Fig. 3.

Although the zero-wasted-area layout is characterized by the condition:

Pressure-Balance $P_\ell = 1$ for every room r_ℓ,

iteration of force-balancing is targeting the layout satisfying the condition:

```
Algorithm SAPS {
  Set initial positions of walls subject to the topological constraint and XY = Σ_{ℓ=1}^{n} a_ℓ.
  Calculate pressures and forces.
  Until the layout has sufficient precision {
    Select an unbalanced inner wall w.
    Do force-balancing of w.
    Update pressures and forces.
  }
  Output the layout.
}
```

Fig. 3. The basic algorithm : SAPS

Force-Balance $F_i = 0$ for every wall w_i.

The following lemma guarantees that the genuine target and the artificial target of the algorithm are consistent.

Lemma 2 *The conditions Force-Balance and Pressure-Balance are equivalent.*

Force-balancing of a wall may cause other force-balanced walls to be unbalanced. Therefore, it may not be expected that the layout approaches a zero-wasted-area layout. This solicitude is solved by introducing a global function called *energy*,

$$E = \sum_{\ell=1}^{n} a_\ell \log \frac{a_\ell}{x_\ell y_\ell}.$$

We have the following lemmas on the energy.

Lemma 3 *If $XY = \sum_{\ell=1}^{n} a_\ell$,*

- *$E \geq 0$ and*
- *The conditions Pressure-Balance and $E = 0$ are equivalent.*

Lemma 3 implies that the energy can be used to measure how close the current layout is to zero-wasted-area layout.

Lemma 4 *The operation of force-balancing of any force unbalancing wall reduces the energy.*

Lemmas 1, 3 and 4 lead the following theorem.

Theorem 1 *The algorithm SAPS improves the floorplan at each step under the energy.*

Note that Theorem 1 does not guarantee that the iteration of force-balancing makes the layout converge to zero-wasted-area floorplan. But, many circumstantial evidences allow us to be optimistic: every example we experimented showed rapid convergency to the zero-wasted-area layout.

Let us apply Algorithm SAPS to the example in Fig. 2. It is trivial that the position of w_3 is determined by one time application of the force-balancing. Therefore, we have only to consider to determine the column vector $\vec{x}^{(k)} = (x_1^{(k)}, x_2^{(k)}, x_3^{(k)})^T$ after the force-balancing for w_1 and w_2 was applied k times. We have

$$\begin{aligned}\vec{x}^{(k)} &= \frac{1}{B}\begin{pmatrix} B & 0 & 0 \\ 0 & a_2 & a_2 \\ 0 & a_3 & a_3 \end{pmatrix} \frac{1}{A}\begin{pmatrix} a_1 & a_1 & 0 \\ a_2 & a_2 & 0 \\ 0 & 0 & A \end{pmatrix} \vec{x}^{(k-1)} \\ &= \frac{X}{C}\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} + \frac{\alpha^k}{C}\begin{pmatrix} B & B & -\frac{AB}{a_3} \\ -a_2 & -a_2 & \frac{a_2 A}{a_3} \\ -a_3 & -a_3 & A \end{pmatrix} \vec{x}^{(0)},\end{aligned}$$

where $A = a_1 + a_2$, $B = a_2 + a_3$, $C = a_1 + a_2 + a_3$, $X = x_1^{(0)} + x_2^{(0)} + x_3^{(0)}$, and $\alpha = \frac{a_1 a_3}{AB}$. Since $0 < \alpha < 1$, $\vec{x}^{(k)}$ converges to the correct value. Although the convergence of x_1, x_2 and x_3 to the correct values is not monotonical as observed in Fig. 4 (above), the energy converges monotonically to zero as shown in Fig. 4 (below).

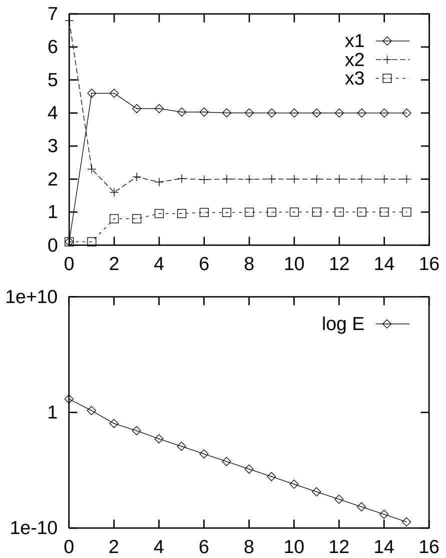

Fig. 4. Transitions of widths (above) and energy (below) by force-balancing for $(a_1, a_2, a_3) = (4, 2, 1)$ and $(x_1^{(0)}, x_2^{(0)}, x_3^{(0)}) = (0.1,\ 6.8,\ 0.1)$.

IV. Enhanced Algorithms

In SAPS, a single wall is selected and moved to force-balancing position at a time. We enhance the algorithm aiming to accelerate the computation by moving all the walls to some appropriate positions simultaneously. This enhanced algorithm is rather considered to simulate more faithly the natural phenomenon than SAPS. We call it *Simulated-Air-Pressure-All (SAPA)* which is described in Fig. 5. This change may lose the guarantee by Theorem 1. A parameter "activity" included in the algorithm will be used to recover this drawback.

Algorithm SAPA {
  Set initial positions of walls subject to the topological constraint and $XY = \sum_{\ell=1}^{n} a_\ell$.
  Set initial values of "activities".
  Calculate pressures and forces.
  Until the layout has sufficient precision {
    Change "activities" δ_i for every inner wall w_i.
    Move w_i by $\delta_i F_i$ for every inner wall w_i.
    Update pressures and forces.
  }
  Output the layout.
}

Fig. 5. The enhanced algorithm : SAPA

At every step, every wall moves by $\delta_i F_i$. We call the coefficient δ_i the *activity* of w_i. If δ_i is too large, the wall will jump over the unknown correct position, and then reciprocate. What is worse, the topological constraint will be violated. If δ_i is too small, the wall moves by a very small distance each time and it takes long time to converge. Intuitively, it is a good strategy that activities are large enough in early stage and decrease as steps proceed. Several ideas for this dynamic activity control are described in the following.

(1) Limit

We introduce the limit on the distance of a movement to avoid radical movement of a wall. By a movement of a vertical wall w with $F > 0$, rooms on the left are expanded and rooms on the right are compressed. Let α and β as defined before be the minimum width among the rooms to be expanded and that to be compressed, respectively

We introduce the parameters $\lambda_1 > 0$ and $0 < \lambda_2 < 0.5$ to limit the distance of the movement of w in order to avoid expanding over $\lambda_1\alpha$ and avoid compressing over $\lambda_2\beta$. The limit by λ_2 guarantees to maintain the topological constraint. Since the movement distance is δF, we should determine the activity δ to satisfy

$$|\delta F| \le \min\{\lambda_1\alpha, \lambda_2\beta\}.$$

SAPA changes the activity δ to the value

$$\delta \leftarrow \min\left\{\delta, \frac{\min\{\lambda_1\alpha, \lambda_2\beta\}}{|F|}\right\}.$$

(2) Inertia

If the repetition of the movement of a wall w in the same direction is observed, the next movement of w will be increased by the inertia of the previous movement. By this idea, SAPA increases the activity δ to $\gamma_1\delta$ where $\gamma_1 > 1$. It may reduce the computation time since the final position of w is expected to be in the same direction. On the other hand, if alternating movements of w is observed, SAPA decreases the activity δ to $\gamma_2\delta$ where $0 < \gamma_2 < 1$.

(3) Temperature

If increase of the energy is observed, we guess that some walls move over some unknown correct position. Thus we should suppress the distance of movements. By this idea, SAPA decreases the activity δ of every wall to $\gamma_3\delta$ where $0 < \gamma_3 < 1$. This idea is understood to simulate a cooling the temperature.

V. Experiments

Our Simulated-Air-Pressure Algorithms were implemented in C language on a SUN SPARC-Station10 workstation. Algorithms tested are as follows.

SAPS(SEQ) : Move all inner walls sequentially, in a fixed order.

SAPS(MAX) : Select an inner wall with maximum force and move it.

SAPA(LT) : $P = (0.50, 0.25, 1.00, 1.00, 0.80)$. ("Limit" and "Temperature")

SAPA(LI) : $P = (0.50, 0.25, 1.05, 0.95, 1.00)$. ("Limit" and "Inertia")

SAPA(LIT) : $P = (0.50, 0.25, 1.05, 0.95, 0.80)$. ("Limit", "Inertia" and "Temperature")

SAPA algorithms are characterized by parameter $P = (\lambda_1, \lambda_2, \gamma_1, \gamma_2, \gamma_3)$. Every SAPA algorithm is equipped with "Limit" or the topological constraint may be violated. We exclude SAPA algorithm only with "Limit" since it does not converge in most cases. Every algorithm stops when the normalized energy $E/\sum_{\ell=1}^{n} a_\ell$ becomes less than 1.0×10^{-5}. In SAPA algorithms, the energy is evaluated after every $n-1$ movements.

The example named "F20" in the literatures [14, 15, 17] shown in Fig. 6 and complex but regular floorplans named "poly-spiral" structure were used for comparison. "Poly-spiral" structure is the extension of the spiral structure as shown in Fig. 7. The significant feature of poly-spiral structure is that it has no subfloorplan in it, making the problem difficult. Floorplans "Sn" of poly-spiral structure with n rooms were examined for $n = 13, 41, 85, 145, 221, 313$. For "F20", the algorithms start with 100 randomly generated initial layouts. For each "Sn", 100 sets of areas of rooms were generated where each area was randomly ranging from 10 to 100, and for each set the algorithms start with a randomly generated initial layout. Note that the initial layout affects slightly the time to converge but does not affect the level of quality of the final layout since the precision is the condition to end the process.

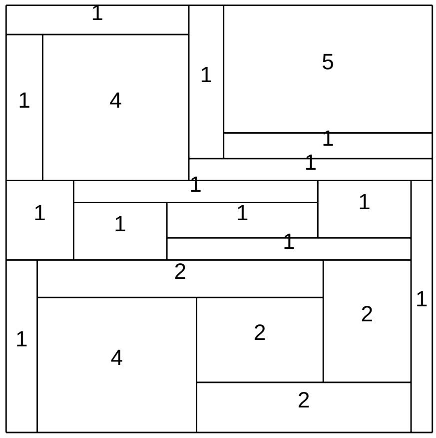

Fig. 6. The example in the literatures consisting of 20 modules. Numbers represent the areas of each module.

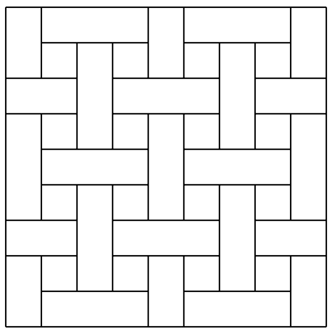

Fig. 7. Poly-spiral structure with 41 rooms.

TABLE I
Average cpu-times (sec.) of our algorithms.

Algorithm	F20	S13	S41	S85	S145	S221	S313
SAPS(SEQ)	0.04	0.01	0.13	0.49	1.50	3.09	6.06
SAPS(MAX)	0.03	0.00	0.08	0.38	1.72	4.91	11.21
SAPA(LT)	0.04	0.01	0.13	0.45	1.46	3.22	6.24
SAPA(LI)	0.03	0.00	0.06	0.23	0.65	1.37	2.58
SAPA(LIT)	0.02	0.00	0.05	0.19	0.56	1.22	2.34

Results are summarized in Table I. The column "Algorithm" lists the algorithms tested. The remaining columns give the average cpu-time (user time + system time) consumed for "F20" and the poly-spiral floorplans, respectively. We observe that every algorithm converges very quickly for each instance. SAPA(LIT) is the best and comparable one is SAPA(LI). The difference of these two is in "Temperature". "Temperature" is introduced to suppress violent movements of the walls when the energy increases. Since "Inertia" also has similar effect, "Temperature" gives a few effect in case that "Inertia" is equipped.

SAPA algorithms are faster than SAPS algorithms as expected. The reason may be as follows. In a force-balancing operation, the force is evaluated many times to search the force-balancing position. Thus it seems that the energy reduction per force evaluation in SAPS is less than that in SAPA in which the distance of a movement is determined by evaluating the force once.

The worst is SAPS(MAX) though the idea of moving a maximally unbalanced wall looks very reasonable compared with SAPS(SEQ) which moves walls independent of the current layout. It may be by the reason that the operation of force-balancing is so fast that a thoughtless selection following the data order is better than spending time to search a wall of the maximum force Note that SAPS(MAX) is equipped with linear search to find the wall in this implementation. It might be faster than SAPS(SEQ) if with a more efficient method to search.

VI. Extension for Aspect-Ratio Constraints

Since the extremely thin module is hard to be implemented, the aspect-ratio constraint is assigned to each module in practice. In this section, we extend the air-pressure model to handle the aspect-ratio constraints on continuous modules. The aspect-ratio constraints can be interpreted as the bounded continuous modules. Every room r_ℓ has the lower bounds $x_\ell^{\min}$ and $y_\ell^{\min}$ of width and height, respectively, which are derived from the associated module.

For the problem on continuous modules, we can fix the positions of outer walls, since the zero-wasted-area layout is attained for any aspect-ratio of floor-rectangle. However, for the problem on bounded continuous modules, zero-wasted-area layout may not be attained. Furthermore, the feasibility of a layout with a given area may depend on the aspect-ratio of the floor-rectangle. Thus, we move the outer walls as well as inner walls. We begin with a initial layout and iterates movements of walls maintaining the incidence relation and the lower bounds of width and height of rooms. Note that Theorem 1 and Lemma lemm:Balance do not hold for the problem on bounded continuous modules.

We introduce a *prop* bar to represent the lower bound of the width or height of a room. A horizontal prop of length $x_\ell^{\min}$ and a vertical prop of length $y_\ell^{\min}$ are put into every room r_ℓ. If the width of a room is larger than the prop length, the prop plays no role in the model. A prop prevents a wall from moving to make the room narrower than the bound. If a prop prevents a wall from moving but the force of the wall directs the wall to move, the prop transmits the force to the wall on the opposite side and these walls move together. We introduce a *composite wall* that is a set of such walls moving together as follows: A wall forms a composite wall; If there are two composite walls incident to a prop and either composite wall pushes the prop, then the union of the walls in two composite walls forms a composite wall. Here, the force of a composite wall is defined by the sum of the forces of walls in the composite wall. An example of a prop and a composite wall is given in Fig. 8. We move the walls in a maximal composite wall as one wall. In the following, we abbreviate a maximal composite wall as a composite wall for simplicity.

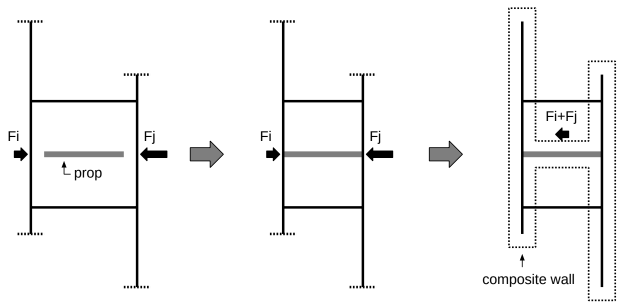

Fig. 8. A prop and a composite wall.

Our algorithm *Simulated-Air-Pressure-Single/prop (SAPS/prop)* iterates the movement of one composite wall at a time. Basically, the composite wall is moved to the force-balancing position. However, a prop in a room incident to the composite wall may prevent such movement. If so, the composite wall stops at the point where it touches the prop. In the next step, walls in the composite wall and the wall on the opposite side of the prop will form a composite wall. Although a composite wall may be no longer a composite wall on the way of a movement because the forces of walls in the composite wall will change according to the movement, we assume these walls still form a composite wall within the movement. The outline of *Simulated-Air-Pressure-Single/prop (SAPS/prop)* is described in Fig. 9.

SAPS/prop may produce an infeasible layout where the area of some room is less than the area of module assigned to it. An example of such an infeasible layout is shown in Fig. 10 where the forces of composite walls are balanced. Walls w_1 and w_B form a composite wall since they are received forces to shrunk r_1. The pressures of r_1, r_3, and r_5 push the composite wall downward and the pressure of the outer room pushes it upward. Since the pressures of r_1 and r_3 are lower than that of the outer room ($= 1$), the composite wall balances when the pressure of r_5 is

Algorithm SAPS/prop {
  Set initial positions of walls subject to the topological constraint.
  Calculate pressures and forces.
  Compose composite walls.
  Until the system saturates {
    Select an unbalanced composite wall w.
    Do force-balancing of w.
    Update pressures, forces and composite walls.
  }
  Adjust the positions of walls to meet the physical constraints.
  Output the layout.
}

Fig. 9. The algorithm for prop model : SAPS/prop

higher than that of the outer room. Same discussions for the other composite walls hold. Thus, the system are balanced though the pressure of r_5 is higher than 1.

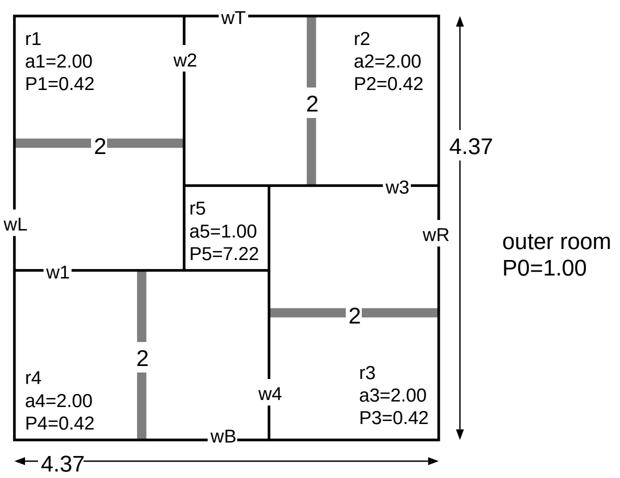

Fig. 10. An infeasible but balanced layout produced by SAPS/prop.

There are several approaches to get a feasible layout by modifying SAPS/prop. In the following, we show two approaches for example, although they are not outstanding in the sense that a layout obtained may not be optimal.

One approach is postprocessing which adjusts the positions of walls produced by the original SAPS/prop. First, the aspect-ratio of each module is determined as the nearest value to the aspect-ratio of its room within the constraint. Next, the positions of walls that minimize the area of floor-rectangle are determined by a conventional way using the longest path algorithm since the shapes of modules are fixed. Then, a feasible layout is obtained. Fig. 11 shows the resultant layout for the example in Fig. 6 with aspect-ratio constraint of every module being $[0.5, 2.0]$. This attains the figures that the area of floor-rectangle is $5.29 \times 9.32 = 49.30$. This quality is comparable to the optimal layout shown in Fig. 8(c) in [17].

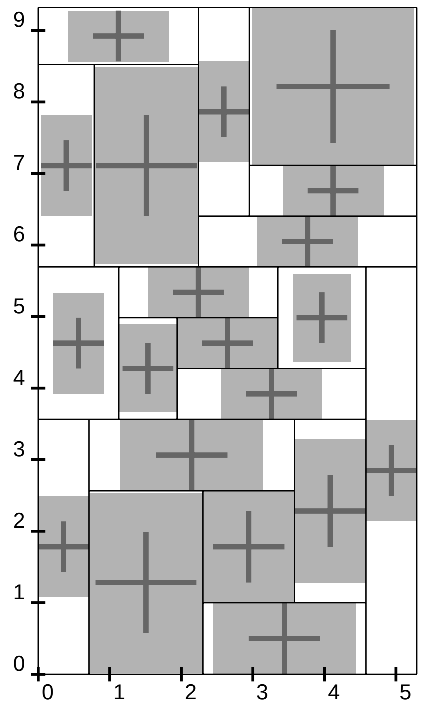

Fig. 11. A resultant layout with aspect-ratio constraints.

Another approach is to try to cancel the undesirable effects due to the prop. As shown in Fig. 10, it seems that the pressure of r_5 is high because of the low pressures of the other rooms where the props touches the walls. Thus, we calculate the forces of the walls incident to such room, that is, the prop touches the walls and the pressure is lower than 1, as if the pressure of the room is 1. SAPS/prop will produce feasible layouts by this approach.

VII. Conclusion and Remarks

We proposed a new approach based on a simulated air-pressure model for the zero-wasted-area layout problem. The notion of energy was introduced to measure how close the current layout is to the zero-wasted-area layout. It was proved that the energy is reduced monotonically to zero each time the operation force-balancing is applied and that it is zero if and only if the zero-wasted-area layout is attained. It is a future work of theoretical interest to complete the theorem which guarantees the layout certainly converges to a zero-wasted-area by the repetition of the force-balancing.

Further ideas obtained from observation of natural phenomenon were implemented. Experimental results showed that the proposed method is very promising in computation time and precision. It will be useful as an evaluation tool in such layout methodologies that repeat generation and evaluation of layout topologies.

In order to handle the aspect-ratio constraints which arise from practical circumstances, a prop bar which represents the lower bound of the width or height of a module was introduced in the model. Several ideas and algorithms for this extended model were presented. Further enhancements are needed to fit real cases including a module the area of which changes according to its aspect ratio, the area for routing, and so on.

There is yet another future work as follows. In practice, local changes of the topology of a floorplan are often acceptable expecting a better layout. Our proposed model will be appropriate for the purpose since the pressures and forces which reflect some local state will give us reasonable hints how to change the topology.

References

[1] R. H. J. M. Otten. Automatic floorplan design. In *Proc. 19th Design Automation Conf.*, pages 261–267, 1982.

[2] R. H. J. M. Otten. Efficient floorplan optimization. In *Proc. ICCD*, pages 499–503, 1983.

[3] L. Stockmeyer. Optimal orientations of cells in slicing floorplan designs. *Information and Control*, 57:91–101, 1983.

[4] K.-H. Yeap and M. Sarrafzadeh. An integrated algorithm for optimal floorplan sizing and enumeration. In *Proc. European Design Automation Conf.*, pages 29–33, 1993.

[5] W. Shi. A fast algorithm for area minimization of slicing floorplans. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 15(12):1525–1532, Dec 1996.

[6] D. F. Wong and P. S. Sakhamuri. Efficient floorplan area optimization. In *Proc. 26th Design Automation Conf.*, pages 586–589, 1989.

[7] T.-C. Wang and D. F. Wong. Optimal floorplan area optimization. *IEEE Trans. on Computer-Aided Design*, 11(8):992–1002, Aug 1992.

[8] K. Chong and S. Sahni. Optimal realizations of floorplans. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 12(6):793–801, Jun 1993.

[9] T.-C. Wang and D. F. Wong. A note on the complexity of stockmeyer's floorplan optimization technique. In M. Sarrafzadeh and D. T. Lee, editors, *Algorithmic Aspects of VLSI Layout*, pages 309–320. World Scientific, Singapore, 1992.

[10] C.-H. Chen and I. G. Tollis. Area optimization of spiral floorplans. *Journal of Circuits, Systems, and Computers*, 3(4):833–857, 1993.

[11] P. Pan, W. Shi, and C. L. Liu. Area minimization for hierarchical floorplans. *Algorithmica*, 15:550–571, 1996.

[12] S. Wimer, I. Koren, and I. Cederbaum. Optimal aspect ratios of building blocks in VLSI. *IEEE Trans. on Computer-Aided Design*, 8(2):139–145, Feb 1989.

[13] P. Pan and C. L. Liu. Area minimization for floorplans. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 14(1):123–132, Jan 1995.

[14] S. Wimer, I. Koren, and I. Cederbaum. Floorplans, planar graphs, and layouts. *IEEE Trans. on Circuit and Systems*, 35(3):267–278, Mar 1988.

[15] K. Wang and W.-K. Chen. A class of zero wasted area floorplan for VLSI design. In *Proc. IEEE International Symposium on Circuits and Systems*, pages 1762–1765, 1993.

[16] K. Wang and W.-K. Chen. Floorplan area optimization using network analogous approach. In *Proc. IEEE International Symposium on Circuits and Systems*, pages 167–170, 1995.

[17] T.-S. Moh, T.-S. Chang, and S. L. Hakimi. Globally optimal floorplanning for a layout problem. *IEEE Trans. on Circuit and Systems – I: Fundamental Theory and Applications*, 43(9):713–720, Sep 1996.

[18] T. Izumi, A. Takahashi, and Y. Kajitani. Air-pressure model and fast algorithms for zero-wasted-area layout of general floorplan. *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Science*, E81-A(5), May 1998. to appear.

[19] N. Yonezawa, N. Nishiguchi, A. Etani, F. Tsukuda, and R. Hashishita. A VLSI floorplanner based on "balloon" expansion. In *Proc. European Design Automation Conf.*, pages 257–261, 1990.