Automation and Robotics in Construction XI
D.A. Chamberlain (Editor)

Automating Dynamic Layout Construction

P.P. Zouein[a] and I.D. Tommelein[b]

[a, b] Civil and Environmental Engineering Department,
The University of Michigan, Ann Arbor, MI 48109-2125, USA

Abstract

A critical problem in dynamic environments and especially on construction sites is coordinating the use of limited space to accommodate resources and their changing demand for space over time. This paper presents a heuristic model that allocates site space to resources associated with an activity schedule so that no spatial conflicts would arise and all resources can be functional. It addresses the dynamic layout planning problem, which involves creating a sequence of layouts that span the duration of construction of a project, given the schedule with activity resources and the timing of their presence on site. The MoveSchedule model, presented here, characterizes changing resource space needs over time and uses a heuristic procedure to automatically construct a dynamic layout while allowing space reuse.

1. INTRODUCTION

Laying out temporary facilities on site is often done manually by marking up a site drawing to show different overlapping laydown areas that will be present for consecutive time frames. This typically results in a complicated drawing that can be interpreted only by the one who made it and that is virtually impossible to update as construction progresses [4]. Computer tools with graphical simulation (e.g., walk-through-like systems) can clearly assist in this process, though few algorithms exist for them to automatically create a desirable sequence of layouts.

Most layout construction algorithms pertain to *static layouts*, i.e., layouts that span a single time interval during which all resources require space. They generate solutions that are near-optimal in terms of a heuristic score based on adjacency and location requirements of work groups. Typically, users then custom-tailor those solutions using manual editing tools to make them suit the actual problem requirements.

The process of creating a sequence of layouts to span the duration of construction of a project when given the activity schedule is referred to as *dynamic layout construction*. Each individual layout in the sequence must span a specific time interval and accommodate the resources that are scheduled to be present during this interval.

Few models for dynamic layout construction exist, but research is under way to capture modeling data and some models are under development. For example, [2] argues for improved site materials management and focuses on field data

acquisition. Others (e.g., [3]) use CAD-models to generate and quantify space availability of a process work area, which can serve as input to systems that incorporate space as a constraint in their scheduling procedures. MovePlan [5] is an interactive dynamic layout planning tool that enables users to create layouts for different time intervals, given an activity schedule and associated resources, by providing them with resource templates and a work sheet to graphically manipulate those templates.

The MoveSchedule system, presented here, automates dynamic layout construction. Its underlying model characterizes space requirements of resources and the variation of these requirements over time using different space profiles. The model uses location constraints to construct feasible layouts, and proximity and relocation weights to measure the quality of those layouts. MoveSchedule's heuristic dynamic layout construction procedure allows space reuse by resources over time and accounts for changes in resource space needs over time. An example illustrates the applicability and limitations of the procedure.

2. SPACE-TIME CHARACTERIZATION OF RESOURCES

2.1. Discretizing Time

MoveSchedule represents time in discrete intervals that are based on the arrival and departure of resources which may or may not be associated with activities. Resources not associated with activities are termed *independent;* the timing of their presence on site is user input. Resources associated with activities, termed *dependent*, are on site as defined by the activity schedule. The smallest intervals delimited by the (dis-)appearance of independent resources or the start/finish of activities are termed *Primary Time Frames* (PTFs).

2.2. Resources Space Profiles

All resources in MoveSchedule are modeled as rectangles to be positioned at 0° or 90° orientation. MoveSchedule offers four alternate space profiles (Fig. 1) for the user to choose from while modeling the resources' space requirements and the variations of these requirements over time. These profiles constitute the basic building blocks from which more complex profiles can be built.

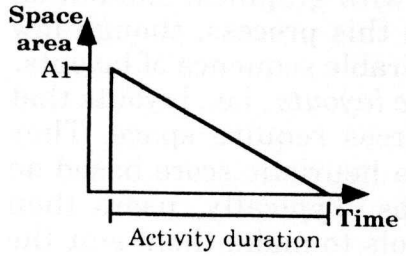


Figure 1.a. Profile-A

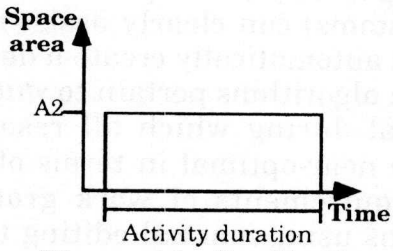


Figure 1.b. Profile-B

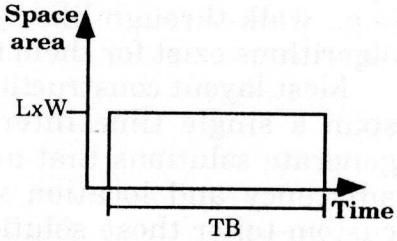


Figure 1.c. Profile-C and Profile-D

The area required by a resource at any one time is inferred from the resource's space profile. For *profile-A* and *profile-B*, the resource is assigned to a single activity but its area varies over the duration of that activity. The dimensions of the resource are computed using the profile at a given point in time and a user-defined length-to-width ratio (L/W) for the resource. For *profile-C* and *profile-D*, the resource dimensions L and W are user defined and fixed over the entire time interval (TB) over which the resource exists. Profile-C models the

space required over TB, computed by MoveSchedule to extend from the start of the earliest activity to the finish of the last activity to which the resource is assigned. Profile-D models the space required by a resource on site for a user-defined value of TB.

Examples of resources that can be modeled with profile-A or profile-B are construction materials that get consumed as the corresponding activity progresses. Profile-C can model construction equipment that is used in multiple activities and that is possibly idle in-between uses. Profile-D typically models Independent resources, such as obstacles on site, construction support facilities (e.g., trailers or parking lots), or resources that exist on site for a given phase of construction (e.g., a tower crane).

2.3. Resource Preferences and Hard Constraints

Two types of constraints drive the construction of the dynamic layout. *Preference constraints* reflect user-defined proximity and relocation weights that gauge the quality of the constructed layouts. A *proximity weight* applies to a pair of resources and measures the level of interaction based on distance between them in a given time interval. Proximity weights have a discrete value set of {0, 25, 75, 100}. A value of 100 means that the two resources have a high level of interaction and the distance between them should be small. The inverse is true for a value of 0. A *relocation weight* applies to an individual resource and measures the penalty for relocating the resource across time intervals. Relocation weights have a discrete value set of {0, 25, 75, 100, + ∞}. The value 0 implies that the resource can easily be relocated, i.e., the cost of relocating it is negligible. In contrast, the value + ∞ implies that the resource should not be relocated across layouts because the cost of doing so is prohibitive; if such a resource has different positions in different layouts, the layout sequence is considered to be infeasible. Resources that have a relocation weight of infinity are termed *non-relocatable*.

Hard constraints are user-defined 2-dimensional geometric constraints between two resources that MoveSchedule uses to prune the *Set of Possible Positions* (SPP) of a resource to possibly one or no position. In the latter case, the layout is infeasible. Examples are non-overlap, in-zone, minimum-distance, maximum-distance, and orientation constraints. Non-overlap constraints apply by default but can be user-overridden by defining in-zone constraints between resources. A minimum (maximum) distance constraint limits the distance between the facing sides of two resources in the X or Y direction, to be greater than (less-) a user-defined value. An orientation constraint may specify that resource i be to the north (south, east, or west) of resource j.

2.4. Constraint Satisfaction Engine

MoveSchedule's hard constraints are built on top of the GS2D constraint engine [1]. GS2D satisfies geometric constraints between rectangles (resources) in a 2-dimensional coordinate system using a bounded interval representation. Each resource's set of possible positions is described by a set of rectangles parallel to the X-Y coordinate axes, representing possible positions of its center point at 0° and 90° orientation. As spatial constraints are applied to resources, GS2D reads their original sets of positions and returns a pair of modified sets, each resource's position set having been pruned to include only those positions consistent with the applied constraint.

3. DYNAMIC LAYOUT PROCEDURE

The dynamic layout construction procedure prioritizes: (1) resources that should not be relocated across layouts, and (2) all other resources. It comprises two stages. In the first stage, the sets of possible positions of non-relocatable resources are pruned to ensure that spatial constraints with other resources on site are met at all times. In the second stage, the procedure identifies the points in time based on the activity schedule where demand for site space changes, and uses linear programming to construct individual layouts for the PTFs delimited by these points in time. The solution of this linear program is a refinement of the sets of possible positions of resources in each PTF, while minimizing the objective function that penalizes total travel distance of resources and weighted for the duration of the PTFs. A single position for each resource is then sampled from the refined set of positions.

The value of the sequence of layouts that results from this procedure and that covers the entire duration of the construction schedule, is gauged in terms of the weighted travel distance in each PTF layout plus the cost to relocate resources from one PTF to the next. The second stage of the procedure is described next, and is followed by the first.

3.1. Pruning Positions to Meet Preference Constraints

The objective function (VFL) that assesses the quality of the dynamic layout consists of two components, P and R. P measures the sum of costs or penalties associated with travel distances between interacting resources for all PTF layouts. R measures the sum of costs associated with relocating resources across PTF layouts. The following equations are used to compute VFL = P + R:

$$P = \sum_t P_t = \sum_t \sum_i \sum_j W_{ij}^t \; d_{ij}^{tt} \; \Delta_t \quad \text{and} \quad R = \sum_t R_{t\,(t-1)} = \sum_t \sum_i W'_i \; d_{ii}^{(t-1)t}$$

Where: W_{ij}^t : Proximity weight between resources i and j in layout t

W'_i : Relocation weight of resource i

$d_{ij}^{(t-1)t}$: Rectilinear distance between centroids of resources i in layout (t - 1) and resource j in layout t

Δ_t : Length of time interval over which layout t extends

This objective function is minimized to prune the SPP of a *resource i* to satisfy its preference constraints, but this is done only after all hard constraints on this resource have been met.

Thus, the objective in the second phase of dynamic layout construction is to find the position or reduced set of positions of *resource i* in a *layout t* that minimize(s) VFL. Let *t* be the layout in which a position for resource *i* is sought, SPP_i^t the set of possible positions of *i*, *n* the number of resources in *t* for which a position was selected, and *layout t-1* the layout for the preceding PTF. This problem is formulated as a linear program as shown on the next page.

This problem is separable and can be formulated as two linear programs for each rectangular subset of SPP_i^t, also shown on the next page. Linear programs (I) and (II) are solved independently to get the values of X_i^t and Y_i^t. The solution of (I) lies at one of the points: LB, UB, or h_k; and that of (II) at: LB', UB', or h'_k.

$$\text{Min} \sum_{j=1}^{n} W_{ij}^t \ \Delta_t \ (|X_i^t - X_j^t| + |Y_i^t - Y_j^t|) \ + W'_i \ (|X_i^t - X_i^{t-1}| + |Y_i^t - Y_i^{t-1}|)$$

ST $LB_m \le X_i^t \le UB_m$; m is the number of rectangles in SPP_i^t

$LB'_m \le Y_i^t \le UB'_m$; X_i^t, Y_i^t, UB, UB', LB, LB' ≥ 0

Where: X_i^t, Y_i^t : x and y coordinates of the centroid of resource i.

UB, LB : upper and lower bound on the x-Dim. of a rectangle in SPP_i^t.

UB', LB': upper and lower bound on the y-Dim. of a rectangle in SPP_i^t.

$$(I)\begin{cases} \text{Min} \sum_k C_k \left| X_i^t - h_k \right| \\ \text{ST} \quad LB_m \le X_i^t \le UB_m \end{cases} \qquad (II)\begin{cases} \text{Min} \sum_k C'_k \left| Y_i^t - h'_k \right| \\ \text{ST} \quad LB'_m \le Y_i^t \le UB'_m \end{cases}$$

X_i^t, Y_i^t, UB ,LB, UB', LB', C_k, C'_k , h_k, $h'_k \ge 0$

Where: C_k, h_k, C'_k, and h'_k are positive constants

3.2. Dynamic Layout Construction

The dynamic layout construction procedure starts by generating the activity schedule and identifying PTFs. No changes to the schedule are allowed from this stage on. The procedure goes through the following steps:

1. For each PTF, check if fixed-position independent resources can be positioned on site without overlapping. If for any PTF this check fails, then the problem is declared infeasible and the user is asked to make revisions to the input data.
2. Let **non-relocatables** be the set of non-relocatable resources ordered by the date of the resource appearance on site. If *non-relocatables* is empty then go to 100, else select the first resource in this set (R). Let *R-time-bound* be the time interval over which R is needed on site. Define *Current Time Bound* (CTB) as the smallest time interval that includes the time intervals of resources in *non-relocatables* which overlap with R-time-bound and with each other. Other CTBs are computed by redefining the set *non-relocatables* to exclude R and the non-relocatable resources present in the previous CTB.
3. Find all resources needed in CTB. Position those with fixed position and update the SPP of the remaining resources to reflect this.
4. Satisfy all hard constraints between resources found in (3) and prune their SPPs accordingly. If the SPP of any of these resources is empty, then the problem is declared infeasible and user intervention is required.
5. Check if there exists any PTF preceding CTB for which no layout was constructed yet. If one or more exist, then fire PTF-Layout Construction operator on each. Process the PTFs sequentially.
6. Identify all the PTFs included in CTB and fire the PTF-Layout Construction operator on each. Process the PTFs sequentially.

3.3. PTF-Layout Construction Operator

To construct the layout of a PTF-i-j, the PTF-layout construction operator runs X number of trials (X can be set to any number) to generate at most X sets of compatible sampled positions for the resources present in PTF-i-j. These sets are then evaluated using the equation for VFL and the resources are bound to the positions of the set that minimizes VFL.

7. Set counter = 0 (counter will keep track of the number of sampled sets).
8. Set counter = counter + 1
9. Let *temporary-list* be the list of all resources needed in PTF-i-j.
10. Position resources in temporary-list that have fixed positions, and position the non-relocatable resources that were positioned in previous time frames. Remove these resources from temporary-list.
11. Let *r* be the resource in temporary-list with highest relocation weight.
12. If r has a fixed position, then its SPP will be this single position. If SPP of r is not empty, a single position for r is sampled from its SPP, and the SPP of all resources in temporary-list is updated to reflect this new single position of r.
13. Remove r from temporary-list. If temporary-list is empty go to 17 else choose a resource with relocation weight of $+\infty$, bind it to r, and go to 15. If no such resource exists, go to 14.
14. Choose from temporary-list the resource that has the highest interaction (sum of proximity weights) with the resources for which a position was sampled.
15. Prune the SPP of r as described earlier in section 3.1.
16. If the SPP for r is empty, then go to 17, else go to 12.
17. If counter < X then go to step 8. If counter = X and no one set has sampled positions for all resources, the problem is declared infeasible and user intervention is required. If counter = X and there exists at least one set of feasible positions for all resources, then bind the positions of the resources in PTF-i-j to the positions of the set that minimize VFL.
18. When the layouts for all PTFs in CTB are constructed go to 2.

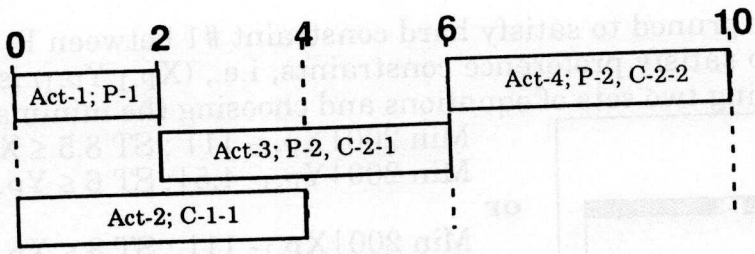
100. Terminate the dynamic layout construction procedure.

4. EXAMPLE

4.1. Input to MoveSchedule

The following example illustrates the use of the dynamic layout construction procedure. The network shown in Fig. 4 describes the process of building a drainage system and foundation walls along the north, south, and west side of a building. In addition to the activity schedule, MoveSchedule input includes the activity resource assignment, the list of independent resources, and the site dimensions (17.5x16). For ease of presentation, the space profiles of all dependent resources are assumed to be of type C. The space profiles of all independent resources are of type D for the duration of construction (in this case 0-10). All resource dimensions have been specified by the user.

Table 1 summarizes the input of hard and preference constraints. Resources I-1, I-2, I-3, and I-4 have user-assigned positions, which are as shown on the layouts in Fig. 5. In particular, the centroid of I-1 is at (11, 8) from the origin (the origin is the lower left hand corner of the figure).



Activity Description	Independent Resources	Dependent Resources
1. Compacting soil S. wall	I-1: Excavated Bldg. (10x8)	P-1: Soil compactor (3x2)
2. Formwork W. wall	I-2: S. wall trench (10x1)	P-2: Crane (2x2)
3. Pipelaying S. wall	I-3: N. wall trench (10x1)	C-1-1: Lumber (2x5)
4. Pipelaying N. wall	I-4: W. wall trench (1x6)	C-2-1: S. wall pipes (5x3)
		C-2-2: N. wall pipes (5x3)

Figure 4. Example Schedule Showing Activities and Associated Resources

Table 1. Example Input of Hard and Preference Constraints

Hard Constraints		Proximity Weights	Relocation Weights
P-1 in-zone I-1	(#1)	$W^{0\text{-}2}_{(\text{I-2 P-1})} = 100$	$W'_{\text{P-2}} = \infty$
d_x (P-2 I-3) = 0	(#2)	$W^{0\text{-}2}_{(\text{I-4 C-1-1})} = 50$	$W'_{\text{C-1-1}} = 0$
d_x (P-2 I-2) = 0	(#3)	$W^{2\text{-}4}_{(\text{I-4 C-1-1})} = 50$	$W'_{\text{C-2-1}} = 100$
d_y (P-2 I-3) ≤ 2	(#4)	$W^{2\text{-}4}_{(\text{I-2 C-2-1})} = 100$	$W'_{\text{C-2-2}} = 100$
d_y (P-2 I-2) ≤ 2	(#5)	$W^{6\text{-}10}_{(\text{I-3 C-2-2})} = 100$	

4.2. Generating the Dynamic Layout

Given the aforementioned input, MoveSchedule identifies the PTFs. First, it checks independent resources with fixed positions to see whether they fit on site without violating non-overlap constraints (Fig. 5.a). Second, it identifies the set of non-relocatable resources (in this case the singleton {P-2}) and computes the resulting CTB, 2-10. The SPP of P-2 ($SPP_{P\text{-}2}$) is then computed given the fixed position resources and the default non-overlap constraints. $SPP_{P\text{-}2}$ is further pruned by applying the hard constraints that exist in CTB between P-2, I-2, and I-3. In Fig. 5.a, an arrow labeled with the number of each hard constraint, points to the set of positions (shaded area) where P-2 satisfies that constraint. In this particular case, the intersection of these rectangles results in the single position for P-2, as shown in Fig. 5.a.

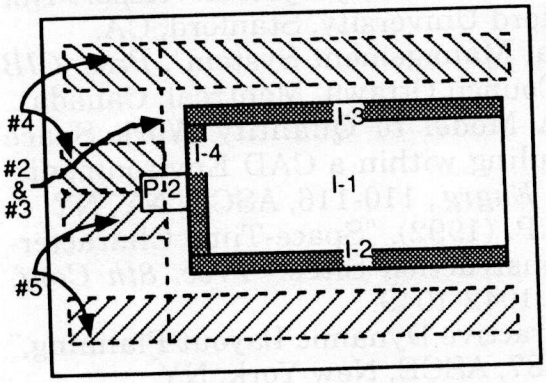


Figure 5.a. Layout-2-10

The following steps detail the sequential construction of individual PTFs that are either subsumed by CTB or in-between CTBs.

PTF-0-2 precedes CTB and is selected first. MoveSchedule selects the non-relocatable resources for which no position was selected yet; none exist for this PTF. The remaining resources in PTF-0-2 are sorted in descending order of the sum of their proximity weights with positioned resources on site: P-1 is selected first, followed by C-1-1.

SPP_{P-1} is first pruned to satisfy hard constraint #1 between P-1 and I-1. It is further pruned to satisfy preference constraints, i.e., (X_{P-1}, Y_{P-1}) is computed by solving the following two sets of equations and choosing the minimal set:

Min $200 | X_{P-1} - 11 |$; ST $8.5 \leq X_{P-1} \leq 14.5$
Min $200 | Y_{P-1} - 4.5 |$; ST $6 \leq Y_{P-1} \leq 10$

or

Min $200 | X_{P-1} - 11 |$; ST $8 \leq X_{P-1} \leq 15$
Min $200 | Y_{P-1} - 4.5 |$; ST $6.5 \leq Y_{P-1} \leq 9.5$

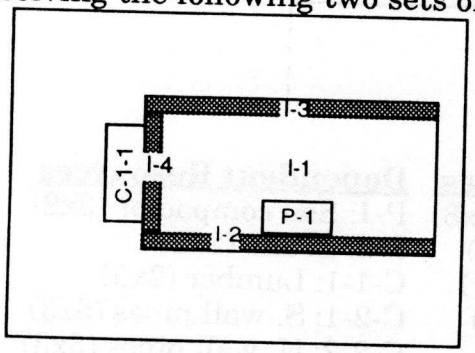


Figure 5.b. Layout-0-2

The same method is used to derive SPP_{C-1-1}. Positions for P-1 and C-1-1 are shown in Fig. 5.b. Similarly, layout PTF-2-4 is constructed (Fig. 5.c). Note the relocation of C-1-1 to accommodate the non-relocatable resource P-2. The position of P-2 was determined earlier to be the only one feasible after satisfying P-2's hard constraints.

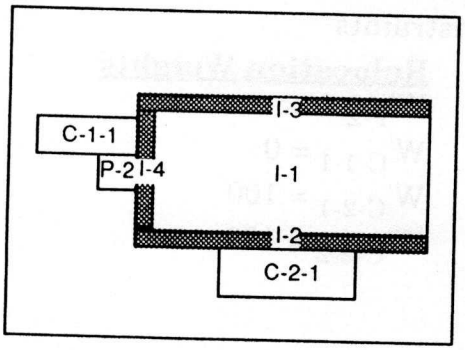


Figure 5.c. Layout-2-4

5. RESEARCH STATUS

MoveSchedule is currently being implemented and its heuristics remain to be tried. Its success will depend on the amount of work involved in specifying all constraints, and on the adequacy of the heuristic values used.

To our knowledge, this model is unique in that it addresses the reuse of space by resources whose presence on site depends on a schedule. The MoveSchedule model for dynamic layout construction as presented here is part of ongoing work to automate the *space scheduling* task [6] which refers to the bi-directional interaction between activity scheduling and space allocation. Automating this task includes implementing strategies for performing time-space trade-offs to solve spatial conflicts or infeasible layouts as they are arise in the process of developing the dynamic layout.

REFERENCES

1. Confrey, T., Daube, F. (1988). "GS2D: A 2D Geometry System." *Report No. KSL 88-15*, Computer Science Dept., Stanford University, Stanford, CA.
2. Riley D., Sanvido V. (1992). "Site Material Management System". *Proc. CIB 92 World Bldg. Congress*, Natl. Research Council Ottawa, Montréal, Canada.
3. Thabet, W.Y., Beliveau, Y.J. (1993). "A Model to Quantify Work Space Availability for Space Constrained Scheduling within a CAD Environment." *Proc. 5th Intl. Conf. Comp. Civil and Bldg. Engrg.*, 110-116, ASCE, NY, NY.
4. Tommelein, I.D., Castillo, J.G., Zouein, P.P. (1992). "Space-Time Characterization for Resource Management on Construction Sites." *Proc. 8th Conf. Comp. Civil Engrg.*, ASCE, New York, NY, 1042-1049.
5. Tommelein, I.D., Zouein, P.P. (1993). "Interactive Dynamic Layout Planning." *J. Constr. Engrg. and Mgmt.*, 119(2), 266-287, ASCE, New York, NY.
6. Zouein, P.P., Tommelein, I.D. (1993). "Space Schedule Construction." *Proc. 5th Intl. Conf. Comp. Civil & Bldg. Engrg.*, 1770-1777, ASCE, NY, NY.