An Automated Methodology for Worker Path Generation and Safety Assessment in Construction Projects

Md Mahbubur Rahman, Leonardo Bobadilla, Ali Mostafavi, Triana Carmenate, and Sebastian A. Zanlongo

***Abstract*— Collisions between automated moving equipment and human workers in job sites are one of the main sources of fatalities and accidents during the execution of construction projects. In this paper, we present a methodology to identify and assess project plans in terms of hazards before their execution. Our methodology has the following steps: 1) several potential plans are extracted from an initial activity graph; 2) plans are translated from a high-level activity graph to a discrete-event simulation model; 3) trajectories and safety policies are generated that avoid static and moving obstacles using existing motion planning algorithms; 4) safety scores and risk-based heatmaps are calculated based on the trajectories of moving equipment; and 5) managerial implications are provided to select an acceptable plan with the aid of a sensitivity analysis of different factors (cost, resources, and deadlines) that affect the safety of a plan. Finally, we present illustrative case study examples to demonstrate the usefulness of our model.**

***Note to Practitioners*—Currently, construction project planning does not explicitly consider safety due to a lack of automated tools that can identify a plan's safety level before its execution. This paper proposes an automated construction safety assessment tool which is able to evaluate the alternate construction plans and help to choose considering safety, cost, and deadlines. Our methodology uses discrete-event modeling along with motion planning to simulate the motions of workers and equipment, which account for most of the hazards in construction sites. Our method is capable of generating safe motion trajectories and coordination policies for both humans and machines to minimize the number of collisions. We also provide safety heatmaps as a spatiotemporal visual display of construction site to identify risky zones inside the environment throughout the entire timeline of the project. Additionally, a detailed sensitivity analysis helps to choose among plans in terms of safety, cost, and deadlines.**

Manuscript received May 21, 2016; revised October 9, 2016; accepted November 8, 2016. This paper was recommended for publication by Associate Editor M. Dotoli and Editor S. Sarma upon evaluation of the reviewers' comments. This work was supported by ARO under Grant 67736-CSII.

M. M. Rahman, L. Bobadilla, T. Carmenate, and S. A. Zanlongo are with the School of Computing and Information Sciences, Florida International University, Miami, FL 33199 USA (e-mail: mrahm025@fiu.edubobadilla@cs.fiu.edu; bobadilla@cs.fiu.edu; tcarm002@fiu.edubobadilla@cs.fiu.edu; szanl001@fiu.edu).

A. Mostafavi is with the Department of Civil Engineering, Texas A&M University, College Station, TX 77843 USA (e-mail: amostafavi@civil.tamu.edu).

This paper has supplementary downloadable multimedia material available at http://ieeexplore.ieee.org provided by the authors. The Supplementary Material contains graphical explanations and sample simulations of the proposed methodologies. This material is 7.95 MB in size.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASE.2016.2628898

***Index Terms*— Construction planning, discrete-event simulation, motion planning, motion safety, optimal construction plan.**

I. Introduction

CONSTRUCTION jobsites are a source of potential accidents which include a significant loss of lives every year due to struck-by collisions involving moving machinery and workers [1]. Recent data shows that the percentage of struck-by accidents constituted 17.6% of fatalities and serious injuries among construction workers [3]. During construction planning activities, safety managers and construction engineers might not be aware of the potential hazards on a construction site. Often times, activity sequences are planned to optimize time, available resources, precedence constraints, site congestion, etc. However, the overall safety of a plan is frequently neglected as there is no suitable automated safety estimation tool.

We identify two coupled phenomena that affect the level of safety hazards related to struck-by accidents in construction jobsites: 1) the sequence of activities and jobsite layout and 2) the movement patterns of workers and equipment [7]. The layout of a construction jobsite affects the movement of equipment and workers within the jobsite.

We focus on better understanding construction operations to reduce hazardous conditions created by a selected construction plan. There are high variability and dynamic changes in construction operations that affect the performance and safety of a project. However, the main activities (e.g., excavation and concrete pouring) can be anticipated and the corresponding equipment, such as trucks, cranes, and drill machines, can be modeled using state-space formulations and motion planning algorithms. Therefore, we propose an *ex ante* analysis model of the deterministic aspects of a construction project to identify its risks. We believe that the high-level deterministic aspects dominate the stochastic aspects and if analyzed properly, can help to prevent and reduce risks.

To the best of our knowledge, our approach is one of the first to consider to use motion planning techniques to evaluate safety scores or determine obstacle-free trajectories for workers and moving equipment. The concrete contributions of this paper are as follows: 1) we generate a number of distinct alternate construction plans that are possible after considering the precedence constraints. Afterward, we select

the plan that would be the best in terms of safety; 2) we develop an activity and event scheduler to simulate all the plans using discrete-event simulation and motion planning; 3) we generate a number of safe trajectories for workers to avoid static obstacles and develop a navigation policy in order to avoid moving equipment; 4) we decompose the layout of a construction site in order to generate heatmaps of the construction layout to identify dangerous hotspots at discrete times; and 5) we develop a model that guides the managers to select one of the Pareto-optimal plans resulting from the sensitivity/tradeoff analysis among the resource, speed, layout modification, duration, etc.

Earlier work from Rahman *et al.* [35], [36] focused on the proof of concept and the preliminary formalization of a motion planning-based safety assessment in construction projects. This paper presents the complete formalization of the methodology and demonstrates its application in case study examples. In particular, the methodology has been extended by incorporating managerial implications, algorithmic analysis, and merging trajectory plans for workers and machines.

II. Related Work

In one stream of research, different studies [33] have developed optimization-based methodologies for the safety assessment of construction site layouts. In another stream of research, discrete-event simulation has been adopted for construction planning [32]. These studies have two main limitations: 1) the lack of consideration of the impact of the layout of construction job sites on the spatiotemporal motion trajectories related to the workers and equipment and 2) lack of consideration related to the dynamic changes in the layout of construction sites at different stages of a project schedule. Our approach for obtaining the safety score is different compared with [32] and [33] since our methodology is based on the motion trajectories of workers and equipment. We convert the construction projects into a state-space model and investigate deeply into the motion planning layers as movement patterns of equipment and workers are the main causes of struck-by hazards. This allows us to generate time indexed dynamic safety scores based on construction events which is an improvement to the static scores found in related literature.

An effective approach requires to be able to translate high-level plans into low-level state trajectories in order to enable better safety assessment. Therefore, our ideas are connected to approaches that use linear temporal logic [5] to create high-level specifications that can be translated to low-level trajectories. Our ideas also share commonalities with STRIPS-like representations [16] that connect with motion planning algorithms [10]. However, in contrast to these approaches, we use *Activity Graphs* and *Discrete-Event System Specification (DEVS)*. The activity graph enables us to efficiently generate a number of alternate plans while the DEVS model helps us to simulate them in detail using low-level motion planning methods.

Some attempts [13] have been made in the construction community to incorporate planning algorithms in the analysis of projects. Motion planning has been used to analyze crane motions and their safe operations. In [43] and [44], a modification of the rapidly exploring random tree (RRT) algorithm for replanning of crane motions was used in real time along with positioning systems for simulation and safety purposes. However, these tools [13], [43], [44] are intended only to capture a small part (e.g., one equipment or a single activity simulation) of the activities in a construction project.

Different models are used to simulate construction activities, such as [4], [8], and [19]. However, these tools are intended only to provide graphical modeling in a virtual construction site without providing any conclusion about the safety level. Moreover, the prior works cannot suggest alternate plans that might be better in terms of safety and other constraints, such as project duration and cost. We developed an automated system that can quantify safety level of a plan, suggest alternate plans, and compare among those in order to reduce the chance of fatalities during a construction project.

Our ideas are also connected to [15] and [34] as these studies propose a hierarchy of task decomposition to accomplish a large task. In contrast, in this paper, the decomposition in subactivities is an input given by the manager as an Activity Graph. We focus on all alternative plans and simulate them to compare them in terms of construction safety, cost, time, and space distribution.

Li *et al.* [28] are concerned about identifying the possible mistakes in a construction plan and repair them by using a virtual simulation. Although we share similar motivations, [28] did not consider the effects of moving machinery and workers, layout, and sequence of activities on the plan's safety level. Another stream of research [30] is mostly concerned with profit maximization by optimizing resources and cash-flow but they ignore safety aspects of a project.

III. Problem Formulation

A. Activity Graph

The *Critical Path Method (CPM)* [31] is widely used in construction projects to determine the minimum amount of time needed to complete a project. An activity graph is a type of CPM with no timing information. The activity graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, is a directed acyclic graph. An edge, $(v, v') \in \mathcal{E}$; $v, v' \in \mathcal{V}$ is formed if and only if an activity denoted by node v is a precondition of another activity represented by a node v'. It is helpful to consider v as a parent of v'. Additionally, $\mathcal{V}_s \subset \mathcal{V}$ is a set of starting nodes with no incoming edges while $\mathcal{V}_f \subset \mathcal{V}$ is the set of finish nodes who have no outgoing edges. Finally, a sequence of all nodes, $\mathcal{P} = (v_1, v_2, \ldots, v_n)$, conserving precedence constraints form a *construction plan*.

B. Construction Physical State Space

Assume that a construction project takes place in a 2-D world, $\mathcal{W} = \mathbb{R}^2$. Let the construction timeframe be defined as $T = [0, \infty)$. The initial set of static obstacles is $\mathcal{O}(t) \subset \mathcal{W}$, $t \in T$, where the obstacle set, $\mathcal{O}(t)$, is a time variant set, since new obstacles may appear on the jobsite and old obstacles may disappear as the construction project progresses.

We define the system state space as X^v for an individual activity or node, $v \in \mathcal{V}$, in the planning graph. A particular

system state, $x^v \in X^v$, is composed of a number of parameters that describe a subproblem. The parameters in x^v can be configurations, orientations, and velocities of moving bodies (e.g., trucks and cranes) as well as the amount of resources (e.g., soil and concrete) used by an activity. Altogether, the entire system state space is defined as $X = X^1 \times X^2 \times \ldots, X^{|\mathcal{V}|}$.

The *time-varying state space* is the Cartesian product $Z = X \times T$ and a state, $z \in Z$, is denoted as $z = (x, t)$. There is a number of moving equipment in the system, such as trucks, excavators, mixers, and cranes, represented by the set $\mathcal{B}(t) = \{B_1, B_2, \ldots, B_k\}$. Considering both the moving bodies and static obstacles, the obstacle state space is defined as

$$Z_{\text{obs}} = \{(x, t) \in Z | \mathcal{B}(t) \cap \mathcal{O}(t) \neq \emptyset\} \tag{1}$$

and the free space is defined as $Z_{\text{free}} = Z \setminus Z_{\text{obs}}$. An *initial state* is defined as $z_I \in Z_{\text{free}}$ and the set of *goal states* are defined as $Z_G \subset Z_{\text{free}}$

$$Z_G = \{(x, t) \in Z | x \in X_G, t \in T\}. \tag{2}$$

C. Augmented Discrete-Event System Specification

Each node of a high-level construction plan in an *activity graph* is represented as an *Augmented Discrete-Event System Specification (DEVS)* [42] model. This model is used along with geometric information from the construction site to generate obstacle-free paths and policies for moving bodies. Each node in the *activity graph* is associated with an augmented DEVS model.

The $DEVS$ formalism proposed by [42] and detailed in [39] and [40] is used to formalize discrete-event simulation as an extension of finite-state automata. An event scheduling model is a tuple ES^v for the activity $v \in \mathcal{V}$ and is represented as

$$\text{ES}^v = \left(E^v, Z^v, \text{EL}^v, f_\eta^v, f_z^v, z_I\right) \tag{3}$$

where $Z^v = X^v \times T$ is the subset of the states of the system. Any activity in a construction site consists of a set of events. The ith event is denoted by η_i and if there are ξ unique events, and we define the finite event set as $E^v = \{\eta_1, \eta_2, \ldots, \eta_\xi\}$. The event list EL^v is defined by $\text{EL}^v = \{(\eta_1, t_1), (\eta_2, t_2), \ldots\}$.

The system starts at time t_0^v with starting state, z_I. The system state is modified based on the current state and an event of an activity

$$f_z^v : Z^v \times E^v \to Z^v. \tag{4}$$

In some cases, f_z^v is controlled by the availability of resources (for example, the amount of soil that needs to be excavated) and system time. The next event to be scheduled is controlled by f_η^v, based on the current event and system state

$$f_\eta^v : E^v \times Z^v \to E^v. \tag{5}$$

A number of alternate construction plans $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k$ are extracted from the CPM graph. To carry out the simulation for each of the plans, $\mathcal{P}_i$, we need to compute the collision-free trajectories in Z_{free} space for the moving equipment and workers knowing the initial and goal configurations in X space.

Problem 1 (Finding Collision-Free Trajectories for Moving Equipment): Given an initial configuration, x_I, a set of goal states,X_G, and the set of static obstacles, $\mathcal{O}(t)$, find a collision-free trajectory, $\widetilde{Z} : [0, 1] \to Z_{\text{free}}$, such that $\widetilde{Z}(0) = z_I$ and $\widetilde{Z}(1) \in Z_G$.

There are m workers, $\mathcal{A}^1, \mathcal{A}^2, \ldots, \mathcal{A}^m$, present in the workspace, who have to travel from their initial position x_I to a destination region X_G. Accordingly, our next problem is to compute the safe trajectories for the workers that avoid both the static obstacles $\mathcal{O}(t)$ and moving equipment $\mathcal{B}(t)$.

Problem 2 (Finding Safe Trajectories for Workers): Given an initial configuration, x_I, a set of goal regions X_G, the set of static obstacles, $\mathcal{O}(t)$, and the trajectories of the moving equipment, $\mathcal{B}(t)$, find an obstacle-free trajectory, $\tilde{x}_{\text{worker}}$, such that $\tilde{x}_{\text{worker}}(0) = x_I$, $\tilde{x}_{\text{worker}}(1) \in X_G$.

Therefore, by solving Problems 1 and 2, we have a set of trajectories for each feasible plan $\mathcal{P}_i$.

D. Safety Evaluation for Different Plans

We need to calculate safety scores for each of the plans $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k$ in order to choose the best plan. Problem 3 calculates the safety score for the plans based on the trajectories $\tilde{X}_{\text{worker}}$ and $\tilde{Z}$ calculated by solving Problems 1 and 2. To calculate the safety score for individual plans, we evaluate the entire plan by simulating all the nodes. A safety score is defined as a function (detailed definition is provided in Section IV-E)

$$R : \widetilde{Z} \to [0, 1] \tag{6}$$

where 0 is the safest score and 1 is the most dangerous score for a plan. Therefore, we try to minimize the safety score. We calculate the safety score for a plan based on the trajectory paths.

Problem 3 (Safety Score Assessment for Different Plans): Given a set of time variant system trajectories, $\widetilde{Z}$, calculate a safety score for the corresponding plan, $\mathcal{P}$, in the closed interval range [0, 1].

Once the safety score is calculated for the alternate plans, the planning managers need to extract the optimal one which provides minimal completion times and optimal safety scores.

Problem 4 (Managerial Implication): Given a number of safety scores for several plans $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k$, calculate the optimal plan which minimizes the project's finishing time and cost while optimizing the safety score.

IV. Methods

A construction plan starts with a 2-D layout of the construction site as shown in Fig. 1. An example critical path management graph (CPM) for this layout is shown in Fig. 2 where we have two excavation activities (EX) followed by two concrete pouring activities (CP).

The system block diagram of our model used to extract the safest plan is shown in Fig. 3. An activity scheduler subsystem is responsible for generating alternative sequences of activities. It communicates with the event scheduler subsystem to simulate one or a number of activities. The event scheduler then uses motion planning algorithms to generate paths for the

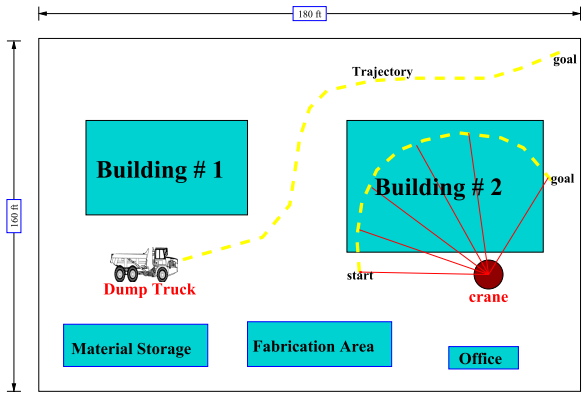


Fig. 1. Example layout of a construction site. Excavation and concrete pouring need to be done in two buildings. Yellow dotted lines are trajectories of a moving truck and a crane's hook.

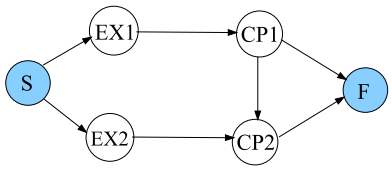


Fig. 2. Example activity graph of a construction site.

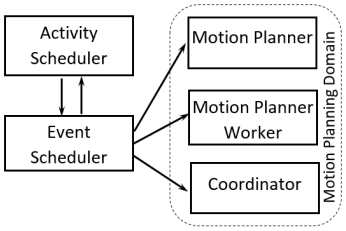


Fig. 3. System framework and subsystem interaction.

moving bodies and a coordinator calculates a way to schedule them without colliding with the bodies of other activities.

The obstacles, whether moving or static, have a different impact on the safety of the construction plan. Also, different sequences of plans yield different safety scores.

Definition 1: Moving equipment, $\mathcal{B}$, does not affect the safety of two sequential activities. The moving equipment, $\mathcal{B}^u$ and $\mathcal{B}^v$ of two parallel activities, u and v, affects the safety of one another.

Definition 2: Static obstacles, $\mathcal{O}^u$, generated by an activity, u, have a succeeding effect on the safety score of all the successor activities, $v \in \mathcal{V}$, unless the obstacle built earlier is removed by some later activities.

Proposition 1: Different plans yield different safety scores, R.

Proof: Suppose we have two alternate plans, $\mathcal{P}_1$ and $\mathcal{P}_2$, from a graph, $\mathcal{G}$. We choose two activities, u and v, where in plan $\mathcal{P}_1$, u is scheduled before v, and in plan $\mathcal{P}_2$, v is scheduled before u. By Definition 1, their safety score is the same. However, by Definition 2, if the static obstacles generated by u and v are not the same, then the plans yield different safety scores. □

Algorithm 1: ActivityScheduler $(\mathcal{P}, \mathcal{G})$

1: $t \leftarrow 0$
2: $Q_t \leftarrow \emptyset$
3: **for** $i = 1$ *to* $|\mathcal{P}|$ **do**
4: $u \leftarrow \mathcal{P}[i]$
5: **if** $\neg u.ParentsVisited()$ **then**
6: $t \leftarrow t + EventScheduler(Q_t)$
7: **for all** $v \in Q_t$ **do**
8: $v.Visited \leftarrow true$
9: **end for**
10: **end if**
11: $Q_t.Insert(u)$
12: **end for**

A. Plan Extraction From an Activity Graph

A *topological sorting* algorithm is used to extract all possible valid plans. Given n vertices, and a set of integer index pairs, (i, j), of the nodes of the graph, $\mathcal{G}$, where $1 \leq i, \ j \leq n$, the problem of topological sorting is to find a permutation $v_1, v_2, \ldots, v_n$ such that i appears to the left of j for all pairs (i, j) [23].

There might be more than one start and finish activities in CPM graph. Accordingly, two dummy activities, v_s and v_f, are added to the graph as starting and final activities with a duration zero in order to create single starting and finishing points (nodes S and F in Fig. 2). Also, the floating activity nodes (without precedence constraint) are added to $\mathcal{G}$ by making v_s as parent and v_f as their child node. By default, v_s is labeled as $Visited$ and is the *parent* of all initial nodes, $\mathcal{V}_s \subset \mathcal{V}$, while v_f is the *child* of all the finishing activities, $\mathcal{V}_f \subset \mathcal{V}$. Given a plan $\mathcal{P}$, produced by the topological sorting algorithm, Algorithm 1 is used for scheduling the activities inside $\mathcal{P}$. A queue, Q_t at time t, is initialized to hold the active (not yet scheduled/visited) activities in Line 2. Line 3 starts a for loop to go over all the activities $u \in \mathcal{P}$ starting from index 1 (remember activity 0 is the dummy starting activity). Line 5 uses the $ParentVisited$ function to check whether all the parents of the current activity have been scheduled. If not, the activities in Q_t are scheduled by calling the $EventSchedule(Q_t)$ routine and the time is updated. The corresponding activity nodes are all set as $Visited$ from Lines 7–9. At Line 11, the current activity node is enqueued into partition Q_t at current time t, whose parent nodes have already been scheduled.

A notable property of this algorithm is that it tries to schedule activities in parallel using the activity queue Q_t whenever possible to reduce project completion time. Accordingly, Q_t continues to hold the activities for which the dependence has been met in the CPM graph at time t. We must schedule the activities in Q_t once the parent of a new activity has not been scheduled as it implies that the parent is in Q_t.

The running time of Algorithm 1 depends on various submethods. The for loop at Line 3 runs in $O(|\mathcal{P}|)$ time. Each activity is scheduled and simulated exactly once either individually or simultaneously with other activities. Therefore, the total aggregated calls of Line 8 are $O(|\mathcal{P}|)$ and

Algorithm 2: EventScheduler (Q)

1: **for all** $v \in Q$ **do**
2: $ES^v \leftarrow CreateDEVS(v)$
3: $z_v \leftarrow initial\ state \in ES^v$
4: $EL_v \leftarrow ((\eta_1^v, 0) : \eta_1^v \in E^v)$
5: **end for**
6: **while** $[(EL_1 \neq \emptyset) \vee \cdots \vee (EL_{|Q|} \neq \emptyset) \wedge t < t_{th}]$ **do**
7: **for all** $v \in Q$ **do**
8: $t^v \leftarrow min\{t : (\eta, t) \in \mathrm{EL}^v\}$
9: $\eta^v \leftarrow \{\eta : t^v \in (\eta, t)\}$
10: $\tilde{x}_v \leftarrow MotionPlanner(\eta^v, z_v)$
11: **end for**
12: $\tilde{x}_{worker} \leftarrow MotionPlannerWorker(z_v)$
13: $(\tilde{z}_1, \tilde{z}_2, \ldots, \tilde{z}_{|Q|}, \tilde{z}_{worker}) \leftarrow Coordination(\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_{|Q|}, \tilde{x}_{worker})$
14: **for all** $v \in Q$ **do**
15: $z_v \leftarrow f_z^v(\eta, z_v)$
16: $\eta_{new}^* \leftarrow f_\eta^v(\eta, z_v)$
17: $EL_v \leftarrow (EL_v \setminus (\eta, t)) \cup (\eta_{new}^*, t + \tilde{z}_v.t)$
18: **end for**
19: **end while**
20: $return\ t$

together the loops in Lines 3 and 7 run $O(2|\mathcal{P}|)$ instead of $O(|\mathcal{P}|^2)$. The only factor that dominates the running time of the algorithm is $EventScheduler$ submethod in Line 6. Accordingly, the running time of Algorithm 1 is $O(|\mathcal{P}|.O(EventScheduler))$.

B. Event Scheduling Using Augmented DEVS

Each node in a plan needs to be evaluated through our event simulation method. A queue of activities Q is received from the $ActivityScheduler$ routine. Each activity is a collection of events, $\eta \in E$. All the events in E are motion planning problems which have to be solved before going on to the next event.

Algorithm 2 is used to simulate a number of nodes in the activity graph using our *augmented DEVS* model. In order to carry out the simulation in Line 2, we first create an event scheduling model, ES, as defined earlier, for each node. Line 3 extracts the initial state, $z_v \in Z^v$ and Line 4 takes the first event from the event set to populate the empty event list. The $while$ loop in Line 6 is used for scheduling all the events from multiple activities. The min method in Line 8 helps to extract the immediate event's time from the event list to be scheduled if more than one event is in the list. Consequently, Line 9 provides the next event. The $MotionPlanner$ routine in Line 10 generates a number of trajectories, $(\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_{|Q|})$, each of which contains a sequence of configurations. Any state of the system involves some construction workers moving in the workspace. We generate the trajectories of the moving workers, $\tilde{x}_{\text{worker}}$ based on the system state z_v in Line 12 using the $MotionPlannerWorker$ subroutine. We will describe their details shortly.

Line 13 calls the $Coordination$ routine to generate a set of collision-free-time-variant trajectories, $(\tilde{z}_1, \tilde{z}_2, \ldots, \tilde{z}_{|Q|})$, for each activity. Lines 14–17 are the updating steps of the system states. On Line 15, a new system state, z_v, is calculated based on a current state and event. State z_v keeps track of the *resources, configurations*, and other attributes of moving bodies for each of the events along with other information. If $z_v \in Z_G$, then the function f_η in Line 16 will generate a $Null$ event. The routine stops when no activity generates any event other than $Null$.

As a *DEVS* simulation system, the running time of Algorithm 2 does not depend on input size. The necessary end conditions for the while loop in Line 6 are self-generating. To terminate the simulation, we put a maximum time t_{th}. This threshold t_{th} (defined by the planning manager) forces the simulation to terminate if all the event lists EL_i from different activities do not finish in time. Once termination is guaranteed, the running time of Lines 6–19 is polynomial as the methods $MotionPlannerWorker$ and $Coordination$ take polynomial time which we will show in Sections IV-C2 and IV-D. $MotionPlanner$ in Line 10 is resolution complete but it takes polynomial time as we have finite search space and specific goal regions.

C. Motion Planner

We need two motion planners: one for the moving equipment and another for the workers.

1) Planning Under Differential Constraints: The $MotionPlanner$ routine called in Line 10 of the $EventScheduler$ routine works based on existing motion planning algorithms. Sampling-based algorithms, such as RRTs [26] or Probabilistic Road Maps [22], can be applied to calculate the trajectories, $\tilde{x}$, of the moving equipment. Also, RRT* [21], an optimized version of RRT, can be applied to generate a better path than a nonoptimized RRT. If the planning domain is low dimensional (i.e., a 2-D domain), then certain combinatorial planning algorithms, such as *trapezoidal decomposition* (see [24, Ch. 6]), can be applied to achieve an efficient path.

To apply any motion planning algorithms, we have to take motion constraints into consideration. As an example, trucks have the differential constraint of not being able to move sideways. To model the motion of such a truck, let the speed of the truck and the steering angle be specified by the actions u_s and u_ϕ, respectively. The transition equation for two consecutive configurations is, $\dot{x}_{\text{tr}} = u_s \cos\theta_{\text{tr}}$, $\dot{y}_{\text{tr}} = u_s \sin\theta_{\text{tr}}$, and $\dot{\theta}_{\text{tr}} = (u_s/L)\tan u_\phi$ (see [24, Ch. 13]), where L is the length of the truck.

2) Planning for the Workers: The subroutine $MotionPlannerWorker$ called in Line 12 of the Algorithm 2 is responsible for generating a number of safe trajectories $\tilde{x}_{\text{worker}}$, for the workers, avoiding static obstacles, $\mathcal{O}(t)$ and moving equipment $\mathcal{B}(t)$.

First, we discuss the $static\ obstacle$ avoidance policies. We will use the *generalized Voronoi diagram* (GVD) or *maximum-clearance roadmap* [24], [27] to compute safe trajectories $\tilde{x}_{\text{worker}}$ for workers. The GVD was chosen, because it is a roadmap whose paths provide maximum clearance from static obstacles (see Fig. 6). Recall that the set of static obstacles at time t is given by $\mathcal{O}(t) = \{O_1, O_2, \ldots, O_n\}$. We assume

Algorithm 3: Calculate All Paths $(\mathcal{O}, q_I, q_G)$

1: $P \leftarrow \mathcal{O}.getEdges().midPoints$
2: $P.Append(Lines_{boundary}.GetSamplePoints())$
3: $L \leftarrow GetVoronoi(P)$
4: **for** $l \in L$ **do**
5: **for** $l' \in \mathcal{O}.getEdges()$ **do**
6: **if** $l.Intersect(l') == True$ **then**
7: $L.Remove(l)$
8: **end if**
9: **end for**
10: **end for**
11: $G(V, E) = G(L.EndPoints, L)$
12: $E.Add(q_I, Nearest(E, q_I))$
13: $E.Add(q_G, Nearest(E, q_G))$
14: $S = GenAllPath(q_I, q_G, G)$
15: $return\ S$

that the obstacles are *convex polygons*. If the obstacles are not convex, they can be approximated by surrounding them with a convex shape.

Algorithm 3 presents the pseudocode of the implemented procedure to find all possible safe paths using the algorithm for a Voronoi diagram of a set of points [6]. The obstacle set, $\mathcal{O}$, contains both the static obstacles and the boundary region as the boundary walls are also considered obstacles. In Lines 1 and 2 of Algorithm 3, we obtain a set of points, P, containing the midpoints of all the polygonal obstacles and sample points from the boundary region. We apply an existing Voronoi diagram algorithm [14] (which takes $O(|P|\log|P|)$ time) in Line 3 ($GetVoronoi$) to P to generate the Voronoi diagram. Let L be the set of Voronoi edges.

Once the Voronoi edges are generated, we remove the edges that pass through the obstacles. Removing all the Voronoi edges (Line 6) that intersect with the *obstacle line segments* results in a set of line segments that approximate the GVD (see Fig. 6). Two for loops in Lines 4 and 5 run in $O(|L|^2)$.

Finally, we construct a weighted undirected graph, $G = (V, E)$, with weights given by $w : E \rightarrow \mathbb{R}_{\geq 0}$. In this graph, V is the set of vertices of the Voronoi diagram and an edge, e, is added for each Voronoi edge. The weight, $w(e)$, for $e \in E$ is given by the Euclidean distance between the vertices that compose the edge, e.

Let $x_I = (q_I, t_I)$ be the initial configuration of a worker and let his goal configuration be $x_G = (q_G, t_G)$, where the points $q_I, q_G \in \mathcal{W} \setminus \mathcal{O}$. We need to connect these two points on the roadmap given by the GVD. In Line 13 of Algorithm 3, this is achieved by connecting q_I to the nearest Voronoi line, $e \in E$. This introduces a new point on e which is the intersection of e and the normal line of q_I on e. The same procedure is also applied to q_G.

To choose a safe trajectory $\tilde{x}_{\text{worker}}$ for the workers, we first compute all possible paths in graph G from q_I to q_G. The method $GenAllPath(q_I, q_G, G)$ in Line 14 generates all possible paths using a variation of breadth first search [11]. Afterward, a path is selected as safe if it has no or infrequent collisions with the moving bodies $\mathcal{B}(t)$. This procedure is discussed in Section IV-C3.

Algorithm 4: CalcVelocityProfile $(\omega_{\text{worker}}, \omega_i, \tilde{x}_{\text{worker}}, \tilde{X}_i)$

1: $line = CalcLine(\tilde{x}_{worker})$
2: **for** $i = 0\ to\ |\tilde{X}|$ **do**
3: $obs_{s\times t} = FindObs(\tilde{x}_{worker}, \tilde{x}_i, \omega_{worker}, \omega_i)$
4: $obsList.Add(obs_{s\times t})$
5: **end for**
6: **for** $b \in obsList$ **do**
7: **if** $intersect(b, line)$ **then**
8: $\pi(b.lowerLeftY) = STOP$
9: $\pi(b.upperLeftY) = MOVE$
10: $line = UpdateLine(line)$
11: **end if**
12: **end for**
13: $return\ \pi$

The method $GenAllPath$ takes linear time and overall the running time of Algorithm 3 is $O(|L|^2)$ which is taken by Lines 4–9.

3) Safest Path Avoiding Moving Bodies: The worker $\mathcal{A}^j$ must move along his path from $\tilde{x}_{\text{worker}}(t_i)$ to $\tilde{x}_{\text{worker}}(t_f)$ while the equipment $B^i(t)$ must move along its path over the time interval $T = [t_i, t_f]$ at a speed of ω_i. To avoid colliding with moving equipment on a trajectory, a worker must yield and make a $STOP$ to let the moving equipment pass.

We will obtain a *plan* for the workers with a fixed speed, and two actions, $STOP$ and $MOVE$. Let $U = \{STOP, MOVE\}$ be the two allowable actions. We call a *policy* a mapping, $\pi : T \rightarrow U$.

Initially, at t_i, both the worker and the moving body $B^i(t)$ start at their initial point of their respective trajectories $\tilde{x}_{\text{worker}}$ and $\tilde{x}_i$. Moving bodies at different times in $T = [t_i, t_f]$ occupy different spaces on the worker's trajectory, $\tilde{x}_{\text{worker}}$. The solution to the problem of avoiding moving bodies lies in a *space-time* coordinate system. Let $S = [0, |\tilde{x}_{\text{worker}}|]$ be the space axis, where $|\tilde{x}_{\text{worker}}|$ is the length of the trajectory, $\tilde{x}_{\text{worker}}$. We define the time-space as $Y = S \times T$ in which each (s, t) indicates a worker's position along the path, $s \in S$, and time, $t \in T$ [20], [24]. The space occupied by the moving body on the workers' path (obstacles in Y) can be calculated in this space-time coordinate system (see [24, Sec. 7.1.3] for details).

Algorithm 4 presents a procedure that creates a plan for the worker using the space-time coordinate system. In Line 1, we calculate the straight line in the space-time system from the original trajectory $\tilde{x}_{\text{worker}}$. In an S-T system, the worker starts in $(0, 0)$ and moves along a line having a slope, $m = dt/ds$ and $m = 1/\omega_{\text{worker}}$ (see Fig. 7), where ω_{worker} is the speed of the worker. Lines 2–5 calculate all the obstacle regions in the space-time system (blue blocks in Fig. 7). An obstacle list $obsList$ in the S-T system is generated in Line 4 for all the moving equipment that cross the workers' trajectory in workspace $\mathcal{W}$. Lines 6–12 find a policy π which avoids all the space-time obstacles. In Line 7, we check whether the line intersects an obstacle region. To avoid the moving equipment, the worker needs to stop, which is recorded by updating the policy π in Lines 8 and 9. When the line in the

S-T system intersects with the computed obstacle regions, it goes up vertically which means that time is moving forward but the worker does not move $((ds/dt) = (0/dt) = 0)$ (see STOP mark in Fig. 7). This waiting essentially makes a delay for the worker to complete his trajectory $\tilde{x}_{\text{worker}}$. The $UpdateLine$ method in Line 10 shifts the starting point of the remaining line section to the upper left corner of the intersecting obstacle region and the process repeats for other obstacles in the S-T system.

In Algorithm 4, Lines 2–5 that are responsible for calculating the *obstacle blocks* that dominate the total running time. Method $FindObs$ runs in $O(|\tilde{x}_{\text{worker}}|.|\tilde{x}_i|)$ [35]. Therefore, the running time of Lines 2–5 is roughly $O(n^3)$.

D. Coordination Space to Prevent Robot–Robot Collision

The sequence of trajectories of moving equipment, $\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_{|Q|}$, is generated by the motion planner for each activity regardless of whether they collide or not with the bodies (equipment or worker) of the other activities which may run in parallel. Hence, the bodies following the trajectories may collide with the bodies of other parallel activities or the moving workers. Given m moving bodies, an m-dimensional coordination space, $\Gamma = [0, 1]^m$, is represented as a unit cube that schedules collision-free paths for the moving equipment [25]. The ith coordinate of Γ represents the domain, $\Gamma_i = [0, 1]$, of the path $\tilde{x}_i$. Let γ_i denote a point in Γ_i. The pairwise robot–robot (body–body) obstacle region is, $\Gamma_{\text{obs}}^{ij} = \{(\gamma_1, \ldots, \gamma_m) \in \Gamma | \mathcal{B}^i(\tilde{x}_i(\gamma_i)) \cap \mathcal{B}^j(\tilde{x}_j(\gamma_j)) \neq \emptyset\}$ which is combined to yield $\Gamma_{\text{obs}} = \bigcup_{i,j\ i\neq j} \Gamma_{\text{obs}}^{ij}$. Therefore, $\Gamma_{\text{free}} = \Gamma \setminus \Gamma_{\text{obs}}$.

At state $(0, 0, \ldots, 0) \in \Gamma$, all bodies are in their initial configurations, $x_i = \tilde{x}(0)$, and at state $(1, 1, \ldots, 1) \in \Gamma$, all bodies are in their goal configurations. Any continuous path, $h : [0, 1] \to \Gamma_{\text{free}}$, for which $h(0) = (0, 0, \ldots, 0)$ and $h(1) = (1, 1, \ldots, 1)$ moves the bodies to their goal configurations (see [24, Ch. 7]). We applied the A^* search algorithm [37] on Γ to generate a path h avoiding robot–robot collisions. A body is allowed to move with a constant speed or directed to remain stopped to yield the other bodies to pass by moving horizontally or vertically in Γ (see [24, Ch. 7]). This A^* search takes polynomial time as the search space is finite and we have a single goal.

By applying the above-mentioned methodologies of coordination among the bodies and workers, we get the set of time variant trajectories, $\tilde{Z}$.

E. Safety Model

We consider a construction boundary to be a perfect rectangular area. If it is not perfect, we can approximate it with a bounding rectangle and convert the added area into static obstacles. We need to decompose the environment into a number of regions to assign a safety score and generate a risk heatmap that provides a visualization of dangerous regions in a workspace over time. Any primitive geometric shape can be used. We used squares, because we approximated the environment as a rectangle. The size of the squares does not affect the computation of the algorithms as all the algorithms are used to either generate trajectories or conduct discrete-event simulations.

We decompose the workspace $\mathcal{W}$ into δ number of squares. The safety scores of all the squares at a time, t, contribute to the safety of the plan at that time. Safety score of a square is dynamic, time dependent and is inversely proportional to its distance to moving equipment.

Assume that the duration of a plan, $\mathcal{P}$, is T, where T is divided into a number of discrete time points $\mathcal{T} = \{0, \Delta t, 2\Delta t, \ldots, j\Delta t\}$ with constant time intervals, Δt, such as $j = T/\Delta t$. We calculate the safety scores in discrete times of $\mathcal{T}$. Let $R(g_i, t)$ denote the score for square g_i of the grid at time, t. Then, the definition of $R(g_i, t)$ is

$$R(g_i, t) = \sum_{j=0}^{|Q_t|} \sum_{k=0}^{|B_j|} \frac{\alpha}{d(g_i, B_k(t)) + \beta} \tag{7}$$

where $d(.,.)$ is a distance function (such as the *Euclidean Distance*) and Q_t is the queue of activities at time t. Parameters α and β are the scaling factors for a better score. The safety scores for the squares inside the obstacles (static or dynamic) are

$$R(g_i, t) = 1. \tag{8}$$

The average safety score, $r_{\text{grid}} : T \to [0, 1]$, for a grid with δ squares at time t is

$$r_{\text{grid}}(t) = \frac{\sum_{i=0}^{\delta} R(g_i, t)}{\delta}. \tag{9}$$

The safety score at time, t for a particular activity plan, $\mathcal{P}$, depends on $r_{\text{grid}}(t)$ and equipment–equipment distances (e.g., vehicle–vehicle, vehicle–crane from coordination). Therefore, the total safety score $r_{\mathcal{P}}$ can be calculated by averaging these values over $\mathcal{T}$

$$r_{\mathcal{P}} = \frac{1}{|\mathcal{T}|} \sum_{t=0}^{|\mathcal{T}|} \left[r_{\text{grid}}(t) + \sum_{i=1}^{|\mathcal{B}(t)|} \sum_{j=i+1}^{|\mathcal{B}(t)|} \frac{1}{d(B_i, B_j)} \right]. \tag{10}$$

We also calculate aggregated safety score over a time interval, $[t_i, t_f]$ where $t_i, t_f \in \mathcal{T}$. The safety score $r_{\text{agg}}(g_i)$ for a square g_i, then

$$r_{\text{agg}}(g_i) = \frac{\sum_{t=t_i}^{t_f} R(g_i, t)}{t_f - t_i}. \tag{11}$$

F. Optimal Plan Computation

The proposed discrete-event-based simulation system is a novel decision support tool that presents the project manager with a quantified safety score. However, a safe plan may be the slowest one or an increase in resources may incur additional safety hazards while competing the project early. These phenomena lead to Pareto optimality where we may not have a plan that is better in terms of all the attributes to be optimized.

Project Duration: T is defined as the project completion time that we get from the $DEVS$ simulation model. This is usually the difference of the starting and finishing times of simulation.

Cost: A slow sequential plan yields the lowest safety score that has less obstacles present at any time. But it is undesirable as modern construction projects have cost, resource, and time constraints. Therefore, a plan $\mathcal{P}$ is partitioned and all activities in a partition Q_t (from Algorithm 1) are carried on in parallel. The total cost L of a construction project is defined as

$$L = \sum_{Q_t} \sum_{j=0}^{|Q_t|} \left(\sum_{B \in \rho(Q_t[j])} l_B \cdot \kappa(B, Q_t[j]) \cdot t_{Q_t[j]} \right) + l_F \cdot t_{Q_t} \tag{12}$$

where $\rho : \mathcal{V} \rightarrow \mathcal{B}$ gives the name of required moving equipment $B \in \mathcal{B}$ (e.g., truck and crane) for an activity $v \in \mathcal{V}$ and $\kappa : \mathcal{B} \times \mathcal{V} \rightarrow \mathbb{N}$ gives the count of each piece of equipment. l_B is the rental cost of the equipment and l_F is the fixed cost (salary, material cost, etc.) per day. $t_Q \in [0, T]$ is the time required to complete all activities in partition Q.

Safety: The quantified safety values $r_{\text{grid}}(t)$ from (9) over discrete times, $\mathcal{T}$ are used to calculate the mean safety value $\mu_{\mathcal{P}}$ and standard deviation of safety $\sigma_{\mathcal{P}}$ for a particular plan $\mathcal{P}$. A plan is safer if both the values are low.

The decision to select an optimal plans requires the evaluation of all the above-mentioned attributes/objectives and is defined as a tuple

$$Y_{\mathcal{P}_i} = (L_{\mathcal{P}_i}, r_{\mathcal{P}_i}, \mu_{\mathcal{P}_i}, \sigma_{\mathcal{P}_i}, T_{\mathcal{P}_i}). \tag{13}$$

Optimal Plan Selection: Therefore, we need a plan which optimizes the construction cost, safety scores, and finishing times. There might not be any single tuple ($Y_{\mathcal{P}_i}$) that minimizes all of these objectives. This tradeoff among the attributes leads to a well-known *Pareto Optimization* [18] problem. Exact solutions for multicriteria optimizations are NP-hard [9].

We, therefore, design an approximate solution model that is compromise of all the objectives. Accordingly, an optimum tuple is computed by taking the minimum for each objective from all the available tuples $Y_{\mathcal{P}_i}$

$$Y^{\min} = \left(L^{\min}, r_{\mathcal{P}}^{\min}, \mu_{\mathcal{P}}^{\min}, \sigma_{\mathcal{P}}^{\min}, T^{\min}\right). \tag{14}$$

A plan $\mathcal{P}_i$ dominates $\mathcal{P}_j$ (denoted by $\mathcal{P}_i \prec \mathcal{P}_j$) if $\forall k\ Y_{\mathcal{P}_i}[k] \leq Y_{\mathcal{P}_j}[k]$. We discard all such dominated plans and the remaining nondominated plans are then Pareto optimal [38]. The normalized tuples, $Y_{\mathcal{P}_i}^{\text{norm}}$ for the plans are

$$Y_{\mathcal{P}_i}^{\text{norm}} = \frac{Y_{\mathcal{P}_i}}{Y^{\min}} = \left(\frac{L_{\mathcal{P}_i}}{L^{\min}}, \frac{r_{\mathcal{P}_i}}{r_{\mathcal{P}}^{\min}}, \frac{\mu_{\mathcal{P}_i}}{\mu_{\mathcal{P}}^{\min}}, \frac{\sigma_{\mathcal{P}_i}}{\sigma_{\mathcal{P}}^{\min}}, \frac{T_{\mathcal{P}_i}}{T^{\min}} \right). \tag{15}$$

Therefore, we choose the plan $\mathcal{P}_i$ that yields: 1) the closest distance of $Y_{\mathcal{P}_i}^{\text{norm}}$ (e.g., Euclidean distance) to the optimal tuple $Y^{\min}$ and 2) the safety score in which $r_{\mathcal{P}_i}$ is below the median ($\Phi = \text{med}([r_{\mathcal{P}_1}, r_{\mathcal{P}_2}, \ldots, r_{\mathcal{P}_k}])$) safety score

$$\underset{\mathcal{P}_i}{\text{argmin}} \left[\sum_{k=0}^{|Y_{\mathcal{P}_i}|} \left(Y_{\mathcal{P}_i}^{\text{norm}}[k] - Y^{\min}[k] \right)^d \right]^{\frac{1}{d}}, \text{s.t. } r_{\mathcal{P}_i} < \Phi. \tag{16}$$

The term in the bracket represents the Euclidean distance when $d = 2$.

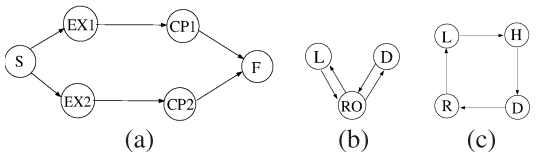


Fig. 4. (a) CPM activity graph for a construction plan. $DEVS$ event transition models for (b) crane and (c) truck.

V. Case Study Examples

In the activity graph shown in Fig. 4(a), the nodes S and F are dummy nodes created to hold starting and final points. Concrete pouring in building site 1 ($CP1$) cannot be carried out before excavation($EX1$). Therefore, $CP1$ has precedence constraints on $EX1$. Likewise, the activity $EX2$ must be completed before $CP2$ as it depends on the completion of $EX2$.

A. Alternative Plans and Activity Scheduling

We used the Python programming language to implement a topological sorting algorithm as proposed in [41]. The following are three alternate plans (sequence of activities) generated for the activity graph shown in Fig. 4(a):

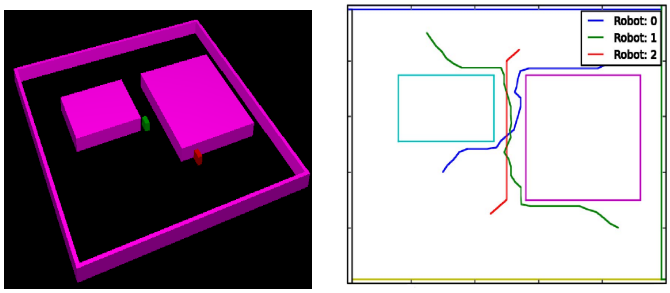
Plan 1($\mathcal{P}_1$)	$EX1 \rightarrow CP1 \rightarrow EX2 \rightarrow CP2$
Plan 2($\mathcal{P}_2$)	$EX1 \rightarrow EX2 \rightarrow CP1 \rightarrow CP2$
Plan 3($\mathcal{P}_3$)	$EX2 \rightarrow EX1 \rightarrow CP1 \rightarrow CP2$

For the plan, $\mathcal{P}_1 = [EX1, CP1, EX2, CP2]$, the Activity Scheduler routine in Algorithm 1 initially loads activity $EX1$ in Q. Q always holds the activities that can be executed in parallel. During the second iteration of the algorithm's loop, it cannot load activity $CP1$ into Q as its parent activity $EX1$ is still in Q. Therefore, $EX1$ is scheduled using Algorithm 2 and $CP1$ is loaded into Q. $EX2$ is also loaded in the next iteration, since its parent S is a dummy node. Before loading $CP2$, we simulate the two activities in the queue ($CP1$, $EX2$) simultaneously using the event scheduler. In the final run, $CP2$ is simulated. $\mathcal{P}_2$ is also simulated in the same way, but we simulate $\mathcal{P}_3$ sequentially by scheduling one activity at a time to compare it with the parallel plans $\mathcal{P}_1$ and $\mathcal{P}_2$.

B. Discrete-Event Scheduling

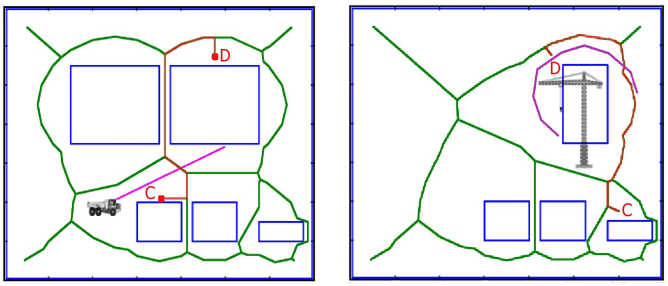
A Python program with the *SimPy* simulation module [2] was used to simulate the discrete-event scheduler of Algorithm 2. An event scheduling model, $ES = \{E, Z, EL, f_\eta, f_z, z_I\}$, for each activity is created. For example, in Fig. 4(b), there are three possible repeating events shown for the crane in charge of concrete pouring (CP). These are *Load (L), Rotate (RO), and Dump (D)*. For excavation (EX), shown in Fig. 4(c), a dump truck in charge of carrying soil has four such states: *Load (L), Haul (H), Dump (D), and Return (R)*. The following are two example DEVS models for excavation and concrete pouring activities.

1) The set of events for concrete pouring is $E^{CP} = \{L, RO, D\}$ and the set of events for excavation is $E^{\text{EX}} = \{L, H, D, R\}$.



(a) (b)

Fig. 5. (a) Two trucks in MSL library colored red and green moving around pink excavation areas. (b) Trajectories generated by the MSL library (blue and green). Red trajectory was added to simulate moving worker.



(a) (b)

Fig. 6. GVD of two sample construction sites. (a) Truck is moving along the pink colored trajectory. (b) Crane is moving along a pink semicircle. The shortest trajectory colored in red from position C to position D for the workers is shown.

2) The state, Z, contains the configuration of all parameters such as *resources, interruption, deadline*, etc.
3) The configuration of the dump truck is $\mathbb{R}^2 \times \mathbb{S}^1$, while the configuration for the *nonholonomic* [29] crane is $\mathbb{RP}^2$ as it can rotate with $pitch$ and yaw, but no $roll$.
4) An example state we use for the truck is $z = (x_{\text{tr}}, y_{\text{tr}}, \theta_{\text{tr}}, \eta_{\text{ex}}, r_{\text{ex}}, t_{\text{ex}})$, and an example state we use for the crane is, $z = (\theta_{\text{cr}}^{\text{pitch}}, \theta_{\text{cr}}^{\text{yaw}}, \eta_{\text{cp}}, r_{\text{cp}}, t_{\text{cp}})$.
5) An example event transition for the dump truck is $f_{\eta}^{\text{EX}}(L, z) = H$, as hauling is carried out after loading. Similarly, the crane starts rotating once it is loaded with concrete, $f_{\eta}^{\text{CP}}(L, z) = \text{RO}$ (see Fig. 4).
6) An example state transition for an excavation is, $f_{z}^{\text{EX}}(L, z) = (x_{\text{tr}}^{\text{new}}, y_{\text{tr}}^{\text{new}}, \theta_{\text{tr}}^{\text{new}}, H, r_{\text{ex}} - r', t_{\text{ex}} + t')$. $(x_{\text{tr}}^{\text{new}}, y_{\text{tr}}^{\text{new}}, \theta_{\text{tr}}^{\text{new}})$ is the new configuration of the truck. The constant, $r' \in \mathbb{N}$, denotes the units of soil/resources consumed per iteration and $t' \in \mathbb{R}^{>0}$ is calculated from a $Coordination$ function as described previously.

C. Motion Planning and Coordination

We used the Motion Strategy Library (MSL) [17] to generate the trajectories of moving equipment for different activities [see Fig. 5(a)]. Sample trajectories for two equipment, $\tilde{x}_1, \tilde{x}_2$ (colored blue and green) are shown in Fig. 5(b). The red trajectory in Fig. 5(b) is the path of a worker that we have generated using the GVDs [35].

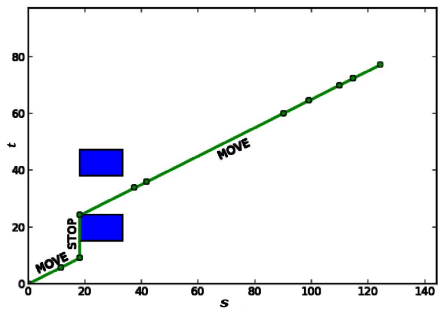


Fig. 7. Obstacles in $s \times t$ space. Vertical line: STOP. Diagonal lines: MOVE.

Two exemplary GVDs generated by Algorithm 3 of a site are shown in Fig. 6. The red lines are the shortest trajectories ($\tilde{x}_{\text{worker}}$) derived for the workers, following the safe Voronoi edges. A dump truck is moving back and forth in Fig. 6(a) while a crane in Fig. 6(b) is following a semicircular path (see pink trajectory).

1) Safe Trajectory Avoiding Dump Truck: Fig. 7 is the space-time system generated by Algorithm 4. Initially, the worker starts moving freely along the trajectory $\tilde{x}_{\text{worker}}$, at a constant speed of ω_{worker}. At some point (marked with "STOP"), the worker has a possibility of colliding with the dump truck. The duration of the collision is 24 units − 15 units = 9 units. We advise the worker to stop, which is indicated by the vertical green line from time 10 to 24. The truck will come back to the opposite direction on the workers' path at time 43 units as indicated by another rectangle centered at $(25, 43)$. This time the worker has already passed, so there will be no collision. The worker finishes at $(124, 76)$ which means that the worker took 76 units of time to complete a 124 unit long path.

We considered three alternative paths shown in Fig. 8. The path of Fig. 8(a) is a good one in terms of safety as it has no collision and takes 77 units of time to finish the length of 155 units, which is longer than the shortest path, but safer. The path in Fig. 8(c) is also safe, as it has no collisions, but it is long. Similarly, the path in Fig. 8(e) is the longest with a length 300 units and with some collision risk [see Fig. 8(f)].

2) Safe Trajectory Avoiding Moving Crane: A high boom crane is present in the site to pour concrete into the Building#2 as shown in Fig. 1. The workers must avoid the hook of the crane as the attached bucket full of concrete can suddenly fall on them which can cause serious injuries and fatalities.

Suppose a worker wants to visit the site from location $C \rightarrow D$. The shortest trajectory $\tilde{x}_{\text{worker}}$ using the Voronoi diagram is shown in Fig. 6(b). Fig. 9 is the $s \times t$ space for other alternate trajectories. The workers trajectory collides with the crane's hook twice [as shown in Fig. 9(b)], if he selects the shortest path shown in Fig. 9(a). The worker must wait until the hook clears his path. We conclude that the worker will reach to his destination at time 115 units while the path is 130 units long. The alternate trajectory in Fig. 9(c) is the second shortest path and does not have a collision. According to our *safety score*, this is a better choice than the shortest path in Fig. 9(a). The worker reaches his destination in 115 time units travailing a path of length 160 units. Even though the path is longer, since it has no collisions, it has a lower safety

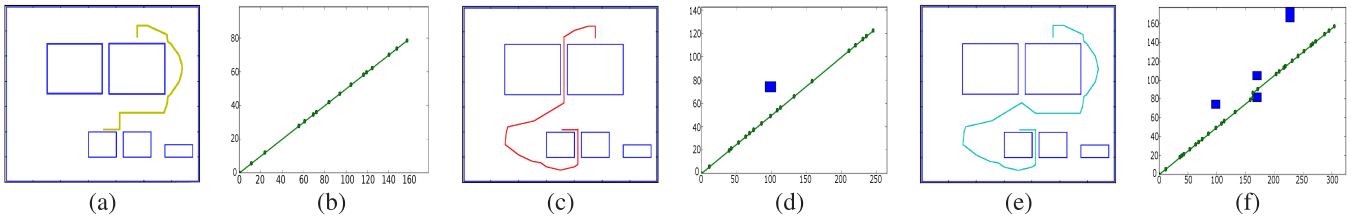


Fig. 8. (a), (c), and (e) Three alternate paths that are not the shortest. (b), (d), and (f) Corresponding velocity profile guidelines from the $s \times t$ graph. There are no collisions for (b) and (d), but the paths are longer. (f) Totally unacceptable as it traverses a long distance having a high chance of collision too.

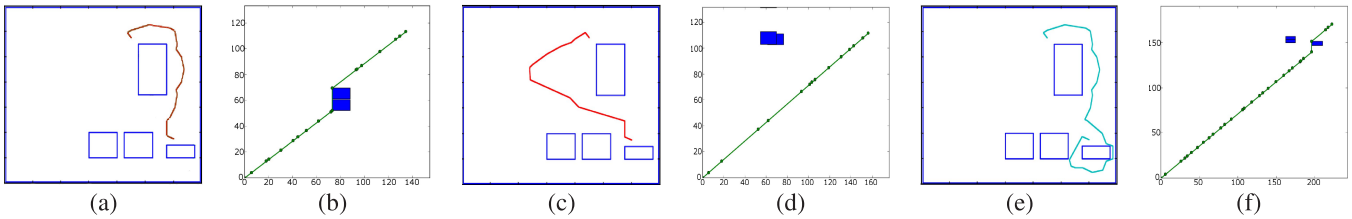


Fig. 9. $s \times t$ space guiding the velocity profile for a crane. (a) Two consecutive obstacle regions found. (c) and (e) Two alternative paths that are not the shortest. (d) and (f) Corresponding velocity profile guidelines from $s \times t$ graph.

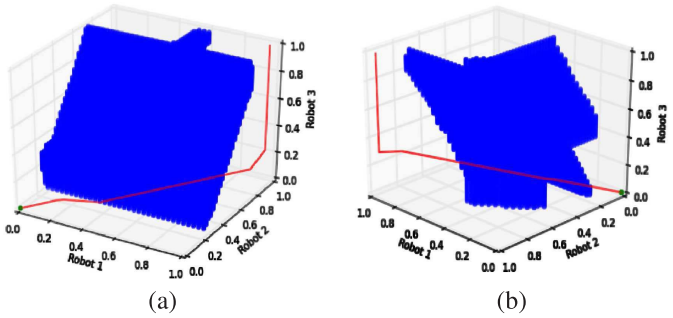


Fig. 10. 3-D coordination space for robots from two different viewing angle. Blue regions are obstacle areas Γ_{obs}. Red line is the collision-free path.

score than the shortest path. We tested another alternate path as shown in Fig. 9(e) which is much longer and involved in a collision [see Fig. 9(f)].

D. Coordination:

The *Coordination* subsystem generates policies for equipment–equipment and robot–worker collision avoidance. A three body coordination space is shown in Fig. 10 using the trajectories of Fig. 5(b). For better visual understanding, we present the 3-D image from two different viewing angles. Blue regions comprise collision configurations, Γ_{obs}, for three possible combinations of truck1–truck2, truck1–worker and truck2–worker. The continuous red path, h, is computed using an A^* search algorithm which connects the point from the initial configuration, (0, 0, 0), to the goal configuration, (1, 1, 1).

E. Safety Evaluation

The safety scores were calculated using the motion profiles.We developed a Python tool for safety heatmap visualization as shown in Fig. 11. A safety score for each square g_i in a grid was calculated by taking aggregated safety over time using (11). The green colored regions are the safest and red regions are the most dangerous. Fig. 11(a) and (b) shows the hazardous zones for two sample activities $(CP1, EX2)$ and $(EX1, EX2)$, where a dump truck and a crane were allocated for excavation and concrete pouring, respectively. Two other heatmaps for an alternate plan, where we double the resources (two trucks per excavation and two cranes per concrete pouring), are shown in Fig. 11(c) and (d). Finally, we generated heatmaps for another plan, where we relocated the equipment's starting and goal locations as shown in Fig. 11(e) and (f) to complete the sensitivity analysis.

F. Sensitivity Analysis

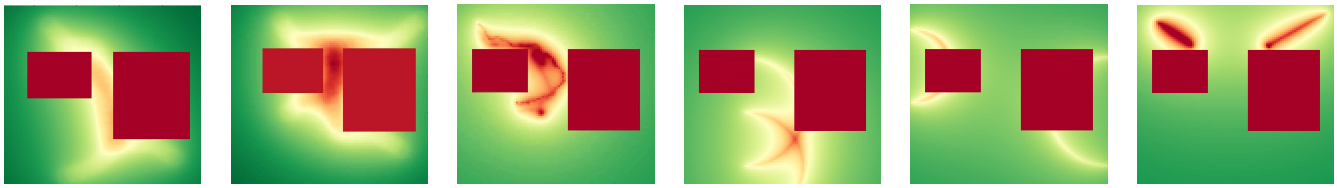
A sensitivity analysis is used to identify the objective [see (13)] that affects construction safety most. The sensitivity test is conducted by keeping one attribute fixed while varying the other inputs. Here, we carry out an exemplary sensitivity test that evaluates the cost ($L_{\mathcal{P}}$), safety ($r_{\mathcal{P}}$), and timeline ($T_{\mathcal{P}}$) attributes.

The timeline chart for the plans, $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$, is shown in Fig. 12(a). Each box of this chart under a particular plan represents a partition Q_t composed of one or more activities that can be simulated together using $DEVS$. Fig. 12(b) shows a graph that presents the change of safety scores over time for different plans.

In Fig. 12(c), we demonstrate the effect of resource increase for activities that dominate the safety score of the plans. Most importantly, it increases the safety score $r_{\mathcal{P}}$ following the increase of dump trucks and cranes as shown in Fig. 12(c).

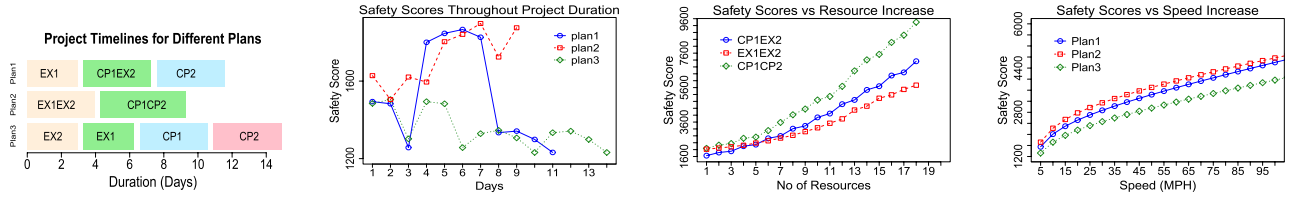
Next, we increase the speed of the equipment that make construction faster. The effect of speed increase was evaluated according to the speed-collision relationship described in [12] which is adapted here in the form $r^n_{\mathcal{P}} = r^{n-1}_{\mathcal{P}} + \xi(\frac{s+\triangle s}{s})^4$. Here, $\triangle s$ is the speed change and ξ is the user-defined weighting factor. The safety score curves are shown in Fig. 12(d) where we see that the scores are increasing rapidly with the increase in speed for all the plans.

A comparison analysis over various plans is presented in Table I where based on the original plans, we generate additional plans by changing: 1) the amount of



(a) (b) (c) (d) (e) (f)

Fig. 11. Aggregated heatmaps (using [11)] for the activities. (a) CP1EX2. (b) EX1EX2. (c) EX1 with increased number of trucks (two trucks). (d) CP2 with two cranes. (e) CP1CP2 with two cranes that have been relocated. (f) EX1EX2 when the initial loading and final dumping positions are changed for the trucks.



(a) (b) (c) (d)

Fig. 12. (a) Time chart for different plans. (b) Variation of safety over time. (c) Variation of safety due to resource increase. (d) Variation of safety due to space relocation.

TABLE I

SAFETY ANALYSIS AND OPTIMAL PLAN SELECTION

Plan	Sensitivity	Cost ($L_{\mathcal{P}}$)	$r_{\mathcal{P}}$	Timeline ($T_{\mathcal{P}}$)	$L_{\mathcal{P}}^{norm}$	$r_{\mathcal{P}}^{norm}$	$T_{\mathcal{P}}^{norm}$	Domination	$Y_{\mathcal{P}}^{norm} - Y^{min}$
$\mathcal{P}_1$	Original ($\mathcal{P}_{11}$)	2250	1525	11	1.07	1.17	2.2	Dominated by $\mathcal{P}_{13}$	1.21
	Resource Increase ($\mathcal{P}_{12}$)	2100	1745	6	1.0	1.34	1.2	Not Dominated	0.39
	Space Changed ($\mathcal{P}_{13}$)	2100	1429	10	1.0	1.09	2.0	Not Dominated	1.00
	Speed Increase ($\mathcal{P}_{14}$)	2150	1605	8	1.02	1.23	1.6	Dominated by $\mathcal{P}_{23}$	0.64
$\mathcal{P}_2$	Original ($\mathcal{P}_{21}$)	2300	1721	9	1.09	1.32	1.8	Dominated by $\mathcal{P}_{14}$	0.86
	Resource Increase ($\mathcal{P}_{22}$)	2350	2076	5	1.11	1.59	1.0	Not Dominated	0.60
	Space Changed ($\mathcal{P}_{23}$)	2100	1553	8	1.0	1.19	1.6	Not Dominated	0.63
	Speed Increase ($\mathcal{P}_{24}$)	2150	1830	7	1.02	1.4	1.4	Dominated by $\mathcal{P}_{12}$	0.57
$\mathcal{P}_3$	Original ($\mathcal{P}_{31}$)	2500	1300	14	1.19	1.0	2.8	Not Dominated	1.81
	Resource Increase ($\mathcal{P}_{32}$)	2450	1592	7	1.16	1.22	1.4	Not Dominated	0.48
	Space Changed ($\mathcal{P}_{33}$)	2200	1348	12	1.04	1.03	2.4	Not Dominated	1.40
	Speed Increase ($\mathcal{P}_{34}$)	2300	1466	10	1.09	1.12	2.0	Dominated by $\mathcal{P}_{13}$	1.01
	Y^{min}	2100	1300	5	1	1	1	Φ =median()=1.19; Selected Plan:$\mathcal{P}_{13}$	

resources ($\mathcal{P}_{*2}$); 2) site space organization ($\mathcal{P}_{*3}$); and 3) speed of the equipment ($\mathcal{P}_{*4}$). Speed and resource-based plans are similar to the above-mentioned description while space changed plans are achieved by changing the starting and goal locations of the equipment, changing the positions of cranes and by relocating the temporary buildings (fabrication, material storage, etc) in order to minimize safety score.

We assign the rental cost for the truck and crane as 50 and 100 per day, respectively. The fixed cost varies in between 100 and 150 depending on the plan. The optimization tuples $Y_{\mathcal{P}}$ are calculated using (13) from which the minimum tuple $Y^{\min}$ is computed. Accordingly, the attributes are normalized ($Y_{\mathcal{P}}^{\text{norm}}$) using (15) and the difference ($Y_{\mathcal{P}}^{\text{norm}} - Y^{\min}$) is calculated from the minimum normalized tuple $Y^{\min} = (1, 1, \ldots, 1)$. After discarding all the dominated plans, we have the remaining plans $\mathcal{P}_{12}$, $\mathcal{P}_{13}$, $\mathcal{P}_{22}$, $\mathcal{P}_{23}$, $\mathcal{P}_{31}$, $\mathcal{P}_{32}$, and $\mathcal{P}_{33}$. Among those, only the plans, $\mathcal{P}_{13}$, $\mathcal{P}_{31}$, and $\mathcal{P}_{33}$ are candidate optimal plans according to (16) as these plans have the safety scores smaller than the median safety ($\Phi = 1.19$). Finally, we select plan $\mathcal{P}_{13}$ that is the closest to the minimum among the three plans.

G. Managerial Implications and Discussion

The methodologies and case studies described earlier guide planning managers through choosing a suitable plan. Our system presents graphical heatmaps (such as in Fig. 11) that enable practitioners to virtually realize the scenarios of the real plan execution. From Fig. 12(a) and (b), we can conclude that the sequential plan, $\mathcal{P}_3$ has low safety variation while the other two plans take less time to finish. In Fig. 12(c), we observed that the increased resource raises the fixed cost per day ($l_F \uparrow$) as more workers and other resources are required to operate additional equipment. Therefore, the planning manager can set a threshold safety score $r_{\mathcal{P}}$th to prevent excessive increase of resources and compute the maximum number of resources that keeps the safety score under the allowed level ($r_{\mathcal{P}} \leq r_{\mathcal{P}}$th). The same conclusion can be drawn for speed increase [see Fig. 12(d)] which essentially raises the safety score by adding more chances of collision and also increases fixed cost (more material). Therefore, a threshold similar to resource increase is used in order to get the maximum allowed speed that keeps the safety score under the allowed limit.

Finally, a detailed analysis similar to Table I helps managers to select a plan out of all possible alternate plans. This multiobjective optimization model also guides the planning managers to choose a slightly lesser safe plan, if this results in significant improvement to the other attributes (e.g., project duration and cost).

VI. Conclusions and Future Work

In this paper, we developed an easily implementable methodology for *ex ante* analysis of construction plans in terms of their safety hazards to minimize the risk of struck-by accidents in construction jobsites. Given an initial activity graph, our model extracts different sequences of activities, converts them to discrete-event models and simulates them using discrete-event scheduler algorithms. Motion planning methodologies generate the collision-free trajectories for the moving bodies and workers. An *ex ante* simulation and proactive safety visualization is provided during preplanning phase using heatmaps and sensitivity analysis which effectively distinguish among the safe and dangerous places in a construction site.

The formalism presented in this paper provides measurement metrics to construction project managers, such as quantified safety scores, cost, and time spent by a construction plan. Based on these measures, the best plan and guidelines for workers can be calculated in a construction site.

One immediate extension of this paper is to take into account the stochastic nature of construction jobsites. We assumed that the motions performed by the moving obstacles were deterministic, so in the future, we plan to incorporate models that include bounded and probabilistic uncertainty. Another extension is to incorporate the movement of equipment in 3-D to investigate possible collision states.

These results provide valuable information for project managers to evaluate construction plans in terms of their safety performance during the planning phase. In addition, the results could be used during the project execution for training workers and equipment operators with regard to hazardous zones and the corresponding safety policies. We would like to closely study deployments linked with real-time monitoring of construction activities to evaluate how likely it is for a worker to follow a suggested plan and what alternate action spaces for workers can be used.

In this paper, we evaluated two commonly performed construction tasks: excavation and concrete pouring. We will extend this paper to evaluate our methodology using information for larger construction projects involving different activities with large equipment fleets and a large number of workers.

References

[1] *Occupational Safety and Health Administration: Struck by Accidents*, accessed on Nov. 29, 2016. [Online]. Available: https://www.osha.gov/SLTC/etools/construction/struckby/mainpage.html

[2] K. Muller and T. Vignaux, *Simpy: Simulating Systems in Python*. Team SimPy, 2003.

[3] (2013). *Leading Causes of Fatal and Non-Fatal Injuries in Construction*. [Online]. Available: http://stopconstructionfalls.com/wp-content/uploads/2013/07/Leading-Causes-of-Fatal-and-Nonfatal-Injuries-in-Construction-2013-update.pdf

[4] H. AlBahnassi and A. Hammad, "Near real-time motion planning and simulation of cranes in construction: Framework and system architecture," *J. Comput. Civil Eng.*, vol. 26, no. 1, pp. 54–63, 2011.

[5] A. Bhatia, L. E. Kavraki, and M. Y. Vardi, "Sampling-based motion planning with temporal goals," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2010, pp. 2689–2696.

[6] P. Bhattacharya and M. L. Gavrilova, "Roadmap-based path planning—Using the Voronoi diagram for a clearance-based shortest path," *IEEE Robot. Autom. Mag.*, vol. 15, no. 2, pp. 58–66, Jun. 2008.

[7] L. Bobadilla, A. Mostafavi, T. Carmenate, and S. Bista, "Predictive assessment and proactive monitoring of struck-by safety hazards in construction sites: An information space approach," in *Proc. 15th Int. Conf. Comput. Civil Building Eng.*, 2014, pp. 989–996.

[8] T. Cheng and J. Teizer, "Real-time resource location data collection and visualization technology for construction safety and activity monitoring applications," *Autom. Construction*, vol. 34, pp. 3–15, Sep. 2013.

[9] A. Chinchuluun and P. M. Pardalos, "A survey of recent developments in multiobjective optimization," *Ann. Oper. Res.*, vol. 154, no. 1, pp. 29–50, 2007.

[10] J. Choi and E. Amir, "Combining planning and motion planning," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2009, pp. 238–244.

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 2nd ed. Cambridge, MA, USA: MIT Press, 2001.

[12] R. Elvik, P. Christensen, and A. Amundsen, "Speed and road accidents: An evaluation of the power model," Nordic Road Transp. Res., TØI Rep. 740, 2004.

[13] V. Faghihi, A. Nejat, K. F. Reinschmidt, and J. H. Kang, "Automation in construction scheduling: A review of the literature," *Int. J. Adv. Manuf. Technol.*, vol. 81, nos. 9–12, pp. 1845–1856, 2015.

[14] S. Fortune, "Voronoi diagrams and delaunay triangulations," in *Computing in Euclidean Geometry*, vol. 1, 1992, pp. 193–233.

[15] C. Galindo, J. A. Fernandez-Madrigal, and J. Gonzalez, "Improving efficiency in mobile robot task planning through world abstraction," *IEEE Trans. Robot.*, vol. 20, no. 4, pp. 677–690, Aug. 2004.

[16] M. Ghallab, D. Nau, and P. Traverso, *Automated Planning: Theory and Practice*. San Francisco, CA, USA: Morgan Kaufman, 2004.

[17] *Motion Strategy Library*, accessed on Nov. 29, 2016. [Online]. Available: http://planning.cs.uiuc.edu/

[18] I. Kacem, S. Hammadi, and P. Borne, "Pareto-optimality approach for flexible job-shop scheduling problems: Hybridization of evolutionary algorithms and fuzzy logic," *Math. Comput. Simul.*, vol. 60, nos. 3–5, pp. 245–276, 2002.

[19] V. R. Kamat and J. C. Martinez, "Visualizing simulated construction operations in 3D," *J. Comput. Civil Eng.*, vol. 15, no. 4, pp. 329–337, 2001.

[20] K. Kant and S. W. Zucker, "Toward efficient trajectory planning: The path-velocity decomposition," *Int. J. Robot. Res.*, vol. 5, no. 3, pp. 72–89, 1986.

[21] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2011, pp. 1478–1483.

[22] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[23] D. E. Knuth and J. L. Szwarcfiter, "A structured program to generate all topological sorting arrangements," *Inf. Process. Lett.*, vol. 2, no. 6, pp. 153–157, 1974.

[24] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2006. [Online]. Available: http://planning.cs.uiuc.edu/

[25] S. M. LaValle and S. A. Hutchinson, "An objective-based framework for motion planning under sensing and control uncertainties," *Int. J. Robot. Res.*, vol. 17, no. 1, pp. 19–42, Jan. 1998.

[26] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," in *Algorithmic and Computational Robotics: New Directions*, B. R. Donald, K. M. Lynch, and D. Rus, Ed. Wellesley, MA, USA: A K Peters, 2001, pp. 293–308.

[27] D. T. Lee and R. L. Drysdale, "Generalization of Voronoi diagrams in the plane," *SIAM J. Comput.*, vol. 10, no. 1, pp. 73–87, 1981.

[28] H. Li, N. Chan, T. Huang, H. Guo, W. Lu, and M. Skitmore, "Optimizing construction planning schedules by virtual prototyping enabled resource analysis," *Autom. Construction*, vol. 18, no. 7, pp. 912–918, 2009.

[29] Y. Lin, D. Wu, X. Wang, X. Wang, and S. Gao, "Lift path planning for a nonholonomic crawler crane," *Autom. Construction*, vol. 44, pp. 12–24, Aug. 2014.

[30] S.-S. Liu and C.-J. Wang, "Resource-constrained construction project scheduling model for profit maximization considering cash flow," *Autom. Construction*, vol. 17, no. 8, pp. 966–974, 2008.
[31] M. Lu and H. Li, "Resource-activity critical-path method for construction planning," *J. Construction Eng. Manage.*, vol. 129, no. 4, pp. 412–420, 2003.
[32] J. C. Martinez, *STROBOSCOPE: State and Resource Based Simulation of Construction Processes*. Ann Arbor, MI, USA: Univ. Michigan, 1996.
[33] X. Ning and K. C. Lam, "Cost–safety trade-off in unequal-area construction site layout planning," *Autom. Construction*, vol. 32, pp. 96–103, Jul. 2013.
[34] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Motion planning with dynamics by a synergistic combination of layers of planning," *IEEE Trans. Robot.*, vol. 26, no. 3, pp. 469–482, Jun. 2010.
[35] M. M. Rahman, T. Carmenate, L. Bobadilla, and A. Mostafavi, "Ex-ante assessment of struck-by safety hazards in construction projects: A motion-planning approach," in *Proc. IEEE Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2014, pp. 277–282.
[36] M. M. Rahman, T. Carmenate, L. Bobadilla, S. Zanlongo, and A. Mostafavi, "A coupled discrete-event and motion planning methodology for automated safety assessment in construction projects," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 3849–3855.
[37] S. Russell and P. Norvig, *Artificial Intelligence—A Modern Approach*. Upper Saddle River, NJ, USA: Prentice-Hall, 2009, ch. 3.
[38] Z. Tarapata, "Selected multicriteria shortest path problems: An analysis of complexity, models and adaptation of standard algorithms," *Int. J. Appl. Math. Comput. Sci.*, vol. 17, no. 2, pp. 269–287, 2007.
[39] H. Vangheluwe, "The discrete event system specification (DEVS) formalism," McGill Univ., Montreal, QC, Canada, Tech. Rep. COMP522A (Course Notes, Course: Modeling and Simulation), 2001.
[40] H. Vangheluwe. (2001). *Discrete Event Modelling and Simulation*. [Online]. Available: http://www.cs.mcgill.ca/~hv/classes/MS/discreteEvent.pdf
[41] Y. L. Varol and D. Rotem, "An algorithm to generate all topological sorting arrangements," *Comput. J.*, vol. 24, no. 1, pp. 83–84, 1981.
[42] B. P. Zeigler, *Multifacetted Modelling and Discrete Event Simulation*. San Diego, CA, USA: Academic, 1984.
[43] C. Zhang, H. AlBahnassi, and A. Hammad, "Improving construction safety through real-time motion planning of cranes," in *Proc. Int. Conf. Comput. Civil Building Eng.*, 2010, pp. 105–115.
[44] C. Zhang, A. Hammad, and J. Bentahar, "Multi-agent-based approach for real-time collision avoidance and path re-planning on construction sites," in *Proc. 28th Int. Symp. Autom. Robot. Construction*, Seoul, South Korea, 2011, pp. 286–291.

Md Mahbubur Rahman received the master's degree in computer science from Florida International University, Miami, FL, USA, and completed numerous projects on intelligent systems, where he is currently pursuing the Ph.D. degree.

He has published several journals and conference papers in top-ranked venues and most of his works are funded by the Army Research Office. He holds a U.S. patent for his work on hair transplantation robot. His current research interests include robotics, autonomous systems, safety optimization, vehicle navigation, and multi robot formation. His solutions can be effective in military, vehicle, and construction robotics.

Leonardo Bobadilla received the Ph.D. degree in computer science from the University of Illinois at Urbana–Champaign, Champaign, IL, USA.

He is currently an Assistant Professor with the School of Computing and Information Sciences, Florida International University, Miami, FL, USA. He has published in major conferences and journals in robotics, automation, and sensor networks. His research has been sponsored by the Army Research Office and the Ware Foundation. His current research interests include understanding the information requirements for solving fundamental robotics tasks, such as navigation, patrolling, tracking, and motion safety, and has deployed robotic test-beds that can control large number of mobile robots that require minimal sensing, actuation, and computation.

Dr. Bobadilla has received several awards.

Ali Mostafavi received the M.Sc. degree in industrial administration (one-year accelerated MBA) degree from the Krannert School of Management, Purdue University, West Lafayette, IN, USA, and the Ph.D. degree in civil engineering from Purdue University in 2013.

He is currently an Assistant Professor with the Zachry Department of Civil Engineering, Texas A&M University, College Station, TX, USA, where he supervises the Infrastructure System-of-Systems Research Group. His current research interests include system-of-systems paradigm that bridges the boundaries between complex systems science, network theory, and civil infrastructure systems to address sustainability and resilience challenges.

Triana Carmenate received the bachelor's degree in information technology–software development and the master's degree in computer science from Florida International University, Miami, FL, USA, in 2014 and 2016, respectively.

Her current research interests include mobile robotics, sensing, and energy analysis in buildings.

Sebastian A. Zanlongo is currently pursuing the Ph.D. degree in computer science from Florida International University (FIU), Miami, FL, USA, with a specialization in motion planning and machine learning.

He is currently a Department of Energy Fellow working at the Applied Research Center, FIU, on multipurpose robotic platforms for inspection of hazardous areas. His current research interests include cooperative control of robot swarms and the application of control and scheduling of multiple robots for human–robot interaction.