A Coupled Discrete-Event and Motion Planning Methodology for Automated Safety Assessment in Construction Projects

Md Mahbubur Rahman[1], Triana Carmenate[1], Leonardo Bobadilla[1], Sebastian Zanlongo[1], Ali Mostafavi[2]

Abstract— **Collisions between moving machinery and human workers in construction 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 construction project plan dangers before their execution. Our methodology has the following steps: 1) Plans are translated from a high-level activity graph to a discrete event simulation model; 2) Trajectories are simulated using sampling based and combinatorial motion planning algorithms; and 3) Safety scores and risk-based heatmaps are calculated based on the trajectories of moving equipment. Finally, we present an illustrative case study to demonstrate the usability of our model.**

I. INTRODUCTION

Construction zones are a source of potential accidents which include a significant loss of lives every year due to struck-by accidents involving moving machinery and workers [1]. Recent data show 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 plan to minimize project timelines without taking safety into consideration. The sequence of activities chosen in order to complete the project might not be the optimal one in terms of safety measures. Therefore, we propose a model to investigate alternate job sequences to ensure better safety.

In one stream of research, different studies (e.g., [21]) have developed optimization-based methodologies for safety assessment of construction site layouts. In another stream of research, discrete event simulation has been adopted for construction planning [20]. The studies related to these streams of research have two main limitations in terms of their use in safety planning: (1) lack of consideration of the impact of the layout of construction jobsites on the spatio-temporal 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 and the sequence of construction activities. Using our predictive assessment, jobsite layout and sequences of activities could be evaluated during the planning phase to identify the *safest* construction plan and *hazardous zones* in a construction jobsite.

Our ideas are connected to approaches that use Linear Temporal Logic, [5], [7], [13], [14], to create high-level specifications that can be translated to low-level trajectories. In contrast with these approaches, we used two representations commonly used in construction planning: Activity Graphs and Discrete Event Simulations instead of Logic Based representations such as LTL or PDDL. This will allow us to evaluate our methodology in existing construction plans and will help close the gap between high-level plans and low level state trajectories.

Some attempts have been made in the construction community to incorporate motion planning algorithms in the analysis of projects. In [28], [8] and [30] a modification of the RRT algorithm for replanning of crane motions was used in real time along with positioning systems for simulation and safety purposes. However, these tools are intended only to capture a small part of the activities in a construction project.

In this paper we focused on calculating the safety score for different construction plans and selecting the optimal plan. Different models are used to simulate the construction activities ([4], [29], [10]). However these tools are intended mainly to provide graphical modeling [10]. We developed an automated system that can help planners to realize the safety level of the planned activities at discrete times of a construction project at the pre-planning phase. These ideas can help managers re-plan a sequence of activities in order to reduce the chance of fatalities and injuries during a construction project.

To our best knowledge, our approach is one of the first to consider using motion planning techniques to evaluate safety scores or determine obstacle free trajectories for workers and moving equipment. The concrete contributions of our work are the following: 1) We generated a number of alternate construction plans rather than the one that gives the minimum project completion time; 2) We developed an activity and event scheduler to simulate all the plans using discrete event simulation and motion planning. A number of trajectories were generated and coordinated to avoid collisions; 3) We decomposed the layout of a construction site into a grid to calculate the safety score. This enabled us to generate heatmaps of the construction layout to identify dangerous hotspots of a site at discrete times.

The rest of the paper is organized as follows: Section II presents the preliminaries and the problem formulation. Section III introduces algorithms to transform a construction plan to a discrete event model. Activities are then scheduled using a discrete event scheduler and safety is calculated for the construction site. Section IV presents an illustrative case study of a construction site. Finally, conclusions and directions for further research are discussed in Section V.

[1]M. Rahman, T. Carmenate, L. Bobadilla and S. Zanlongo are with the School of Computing and Information Sciences, Florida International University, Miami, FL, 33199, USA. {mrahm025,tcarm002,szanl001}@fiu.edu bobadilla@cs.fiu.edu

[2]A. Mostafavi is with the OHL School of Construction, Florida International University Miami, FL, 33199, USA. almostaf@fiu.edu

II. Preliminaries

A. Activity Graph

The *Critical Path Method(CPM)* [19] 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, $e \in \mathcal{E}$, where $e = v \rightarrow v';\ v, v' \in \mathcal{V}$ is formed if and only if v is a precondition of v'. It is helpful to consider v as a parent of v'. The starting node set, $\mathcal{V}_s \subset \mathcal{V}$, is a collection of nodes who have no incoming edges. Similarly, every node in the set of finish nodes, $\mathcal{V}_f \subset \mathcal{V}$, have no outgoing edges. A sequence of all nodes, $\mathcal{P}$, conserving precedence constraints form a *construction plan*.

B. Construction Physical State Space

Assume that a construction project takes place in a 2D world, $\mathcal{W} = \mathbb{R}^2$. A nonempty set of *physical system state spaces* can be represented as, $X = X^1 \times X^2 \times \dots X^\iota$, which is a finite or countably infinite set of *states*. The set of a number of initial static obstacles is, $\mathcal{O}(t) \subset \mathcal{W}$ where the obstacle list, $\mathcal{O}(t)$, is a time variant dynamic list, since new obstacles may appear and old obstacles may disappear as the construction project goes on. A construction project has a set of time constrained objectives which are considered individual sub-components. To realize the abstraction of discretization, we define an index set, $\mathcal{I} = \{1, 2, \dots, \iota\}$. A sub index is defined for an activity, v, as $\mathcal{I}_v \subset \mathcal{I}$. Thus the system state is divided into a number of sub-states, $X = \times X^j$ where $j \in \mathcal{I}_v$ and each set, $X^j \subset X$, is assigned to some individual activity or node, $v \in \mathcal{V}$, in the planning graph.

A system state, $x^j \in X^j$, is composed from a number of parameters that describe a subproblem. The parameters in x^j can be configurations, orientations and velocities of moving bodies (such as trucks or cranes) as well as the amount of resources used by an activity.

A *time* attribute has to be introduced in order to aid the scheduling of activities properly in a time varying discrete event system. 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 are a number of moving bodies/equipment in the system represented by the set, $\mathcal{B}(x)$. Considering both the moving bodies and static obstacles, the obstacle state space is defined as,

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

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

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

A state time space, Z^v, is assigned to an activity, v, where $Z^v = X^v \times T$. A particular time varying configuration, $z^v \in Z^v$, of a sub problem is $z^v = (x^v, t)$. After simulating all the nodes, $v_i \in \mathcal{V}$, in a plan we generate a set of trajectories, $\widetilde{Z}[0, 1] \rightarrow Z_{free}$ by coupling discrete event simulations with low level motion planning techniques.

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)* [27] 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 [27] and detailed in [24] and [25] 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 $v \in V$ and is represented as:

$$ES^v = (E^v, Z^v, EL^v, f^v_\eta, f^v_z, z_I), \tag{3}$$

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 i^{th} event is denoted by η_i and if there are ξ unique events, we define the finite event set as, $E^v = \{\eta_1, \eta_2, \dots, \eta_\xi\}$. The event list EL^v is defined by $EL^v = \{(\eta_1, t_1), (\eta_2, t_2), \dots\}$.

The system starts at time t^v_0 with starting state, z_I. The system state is modified based on the current state and an event of an activity:

$$f^v_z : Z^v \times E^v \rightarrow Z^v. \tag{4}$$

In some cases f^v_z 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^v_η, based on the current event and system state:

$$f^v_\eta : E^v \times Z^v \rightarrow E^v. \tag{5}$$

An activity is finished when the event list, EL^v, becomes empty and we schedule the next activity. Then we need to extract the collision free trajectories in Z_{free} space knowing the initial and goal configurations in X space.

Problem 1: Generation of collision free trajectories $\widetilde{Z}[0, 1] \rightarrow Z_{free}$ for moving equipment through multilevel evaluations

Given an initial configuration, x_I, a set of goal states ,X_G, and the set of static obstacles, $\mathcal{O}(t)$, at time, t, find a collision free trajectory, $\widetilde{Z}[0, 1]$, such that $\widetilde{Z}(0) = z_I$ and $\widetilde{Z}(1) \in Z_G$.

D. Safety Evaluation for Different Plans

A *safety score* is associated to every node in a particular construction plan, $\mathcal{P} = (v_s, v_2, v_3, \dots, v_f)$. 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,

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

where $\widetilde{Z}$ is the set of all trajectories, 0 is the safest score and 1 is the most dangerous score for a plan. Therefore we can calculate the safety score for a plan based on the trajectory paths.

Problem 2: Safety Evaluation

Given a set of time variant system trajectories, $\widetilde{Z}$, calculate a safety score for the plan, $\mathcal{P}$, in the closed interval range $[0, 1]$.

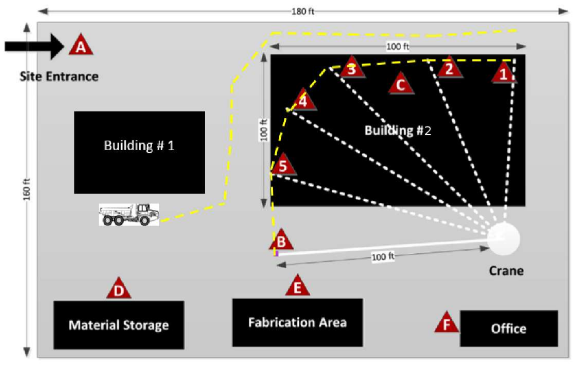


Fig. 1. An example layout of a construction site. Excavation and concrete pouring need to be done in two buildings. Yellow dotted lines are trajectories of moving truck and crane's hook.

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

Problem 3: Mulicriteria Optimization Problem
Given a number of safety scores for several plans $\mathcal{P}_1, \mathcal{P}_2 \ldots \mathcal{P}_i$, calculate the optimal plan which minimizes the project's finishing time while optimizing the safety score.

III. Methods

An example site layout is shown in Figure 1, where a construction plan is needed to carry out two excavation activities in *Building1* and *Building2* followed by concrete pouring activities. The system block diagram of our model to extract the safest plan is shown in Figure 2. An activity scheduler subsystem is responsible for generating alternate sequence of activities and the other subsystems simulate each sequence/plan to estimate the safety attributes. We will describe these shortly.

Definition 3.1: Moving equipment, $\mathcal{B}$, do 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, affect the safety of one another.

Definition 3.2: 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 activity.

Proposition 3.3: Different plans yield different safety scores.

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 3.1 their safety score is the same. However by definition 3.2 if the static obstacles generated by u and v are not same, then the plans yield different safety scores. ■

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) [12].

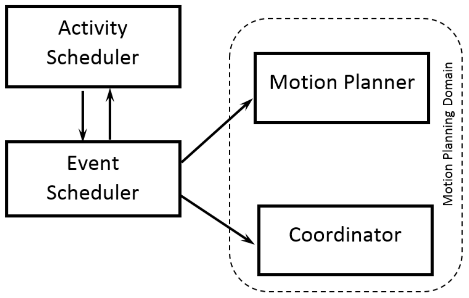


Fig. 2. System submodel interaction.

There is more than one start and finish activity. 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 for the plan (nodes S and F in Figure 3). 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, $\mathcal{V}_f \subset \mathcal{V}$.

Algorithm 1 ActivityScheduler($\mathcal{P}$)

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

Given $\mathcal{P}$, produced by the topological sorting algorithm, Algorithm 1 is used for scheduling the activities inside $\mathcal{P}$. A queue, Q, is initialized to hold the active (not yet scheduled/visited) activities in Line 1. Line 2 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 4 uses the $ParentVisited()$ function to check whether all the parents of the current activity have been scheduled. If not, the activities in Q are scheduled by calling the $EventSchedule(Q)$ routine. It guarantees scheduling all parent nodes of the current node, because the nodes in $\mathcal{P}$ are organized in topological order. The corresponding activity nodes are all set as $Visited$ from lines 6 to 8. Q is flushed at line 9 to load the current activity. At line 11, the current activity node is loaded into

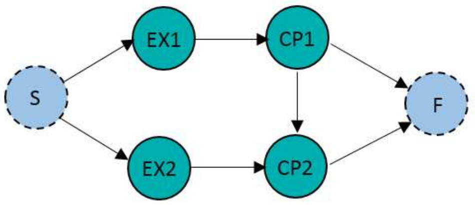


Fig. 3. An example activity graph of a construction site.

Q, whose parent nodes have already been scheduled.

B. *Event Scheduling Using Augmented DEVS*

A queue of activities is received from the $ActivityScheduler$ routine. Each activity is a collection of events, $\eta \in E$. All of 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, $(\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_{|Q|})$, each of which contains a sequence of configurations.

Line 12 calls the $Coordination()$ routine to generate a set of collision-free-time-variant trajectories, $(\tilde{z}_1, \tilde{z}_2, \dots, \tilde{z}_{|Q|})$, for each activity. Lines 13-16 are the updating steps of the system states. On line 14 a new system state, z_v, is calculated based on a current state and event. State z_v keeps track of the *resources*, *configurations*, etc 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 15 will generate a $Null$ event. The routine stops when no activity generates any event other than $Null$.

C. *Motion Planner*

The $MotionPlanner()$ routine called in Line 10 of the $EventScheduler()$ routine works based on existing motion planning algorithms. Sampling based algorithms like RRT [16] or PRM [11] can be applied to calculate the trajectories, $\tilde{x}$, of the moving equipment. We used RRT to generate the paths. If an event in an activity has no motion, then the trajectory is a simple point in the configuration space, X [17]. Also in the case of a crane, it's hook can move over an obstacle to pour concrete into it (see trajectory in Figure 1).

D. *Coordination Space*

The sequence of trajectories, $\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_{|Q|}$, each of which is formed by a sequence of configurations, is generated by the motion planner for each activity regardless of whether they collide or not with the bodies of the other activities which may run in parallel. Hence, the bodies following the trajectories may collide with the bodies of other parallel activities. Given m moving bodies, an m-dimensional coordination space, $\Gamma = [0,1]^m$, is represented as a unit cube to schedule collision free paths for the moving equipment [15]. The i^{th} 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, $\Gamma^{ij}_{obs} = \{(\gamma_1, \dots, \gamma_m) \in \Gamma | \mathcal{B}^i(\tilde{x}_i(\gamma_i)) \cap \mathcal{B}^j(\tilde{x}_j(\gamma_j)) \neq \emptyset\}$

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:     $EL_v \leftarrow ((\eta^v_1, 0) : \eta^v_1 \in E^v)$
5: end for
6: while $[(EL_1 \neq \emptyset) \vee (EL_2 \neq \emptyset) \vee \cdots \vee (EL_{|Q|} \neq \emptyset)]$ do
7:     for all $v \in Q$ do
8:         $t^v \leftarrow min\{t : (\eta, t) \in EL^v\}$
9:         $\eta^v \leftarrow \{\eta : t^v \in (\eta, t)\}$
10:        $\tilde{x}_v \leftarrow MotionPlanner(\eta^v, z_v)$
11:    end for
12:    $(\tilde{z}_1, \tilde{z}_2, \dots, \tilde{z}_{|Q|}) \leftarrow Coordination(\tilde{x}_1, \tilde{x}_2, \dots, \tilde{x}_{|Q|})$
13:    for all $v \in Q$ do
14:        $z_v \leftarrow f^v_z(\eta, z_v)$
15:        $\eta^*_{new} \leftarrow f^v_\eta(\eta, z_v)$
16:        $EL_v \leftarrow (EL_v \setminus (\eta, t)) \cup (\eta^*_{new}, t + \tilde{z}_v.t)$
17:    end for
18: end while
```

which is combined to yield $\Gamma_{obs} = \bigcup_{i,j\ i \neq j} \Gamma^{ij}_{obs}$. Therefore, $\Gamma_{free} = \Gamma \setminus \Gamma_{obs}$.

At state $(0, 0, \dots, 0) \in \Gamma$, all bodies are in their initial configurations, $x_i = \tilde{x}(0)$, and at state $(1, 1, \dots, 1) \in \Gamma$, all bodies are in their goal configurations. Any continuous path, $h : [0,1] \rightarrow \Gamma_{free}$, for which $h(0) = (0, 0, \dots, 0)$ and $h(1) = (1, 1, \dots, 1)$ moves the bodies to their goal configurations [17]. We applied the A^* search algorithm [23] on Γ to generate a path, h, avoiding robot-robot collisions. Moving diagonally along h in Γ, moves all three bodies whose relative speeds depend on the slope of the path. A body is allowed to move with constant speed or directed to remain stopped to yield the other bodies to pass by moving horizontally or vertically in Γ [17]. We divide the path, h, into a number of equally spaced points $\mathcal{H} = (0, \Delta h, 2\Delta h, \dots\ , j\Delta h)$ with fixed spacing, Δh, such as $j = \frac{h}{\Delta h}$. Summation of times is calculated using velocities in each step (e.g. between Δh and $2\Delta h$) to give the total time taken by the bodies to complete a collision free path.

E. *Safety Model*

We decompose the workspace, $\mathcal{W}$, into a grid composed of δ number of squares. A safety score is computed for each square. Together the safety scores of all the squares at a time, t, contribute to the risk of the plan at that time. The safety score for a square is inversely proportional to its distance to the moving equipment.

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

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

Earlier we defined 0 as being the safest score in a plan and 1 as the most dangerous score for a plan. The average safety score, $r_{grid} : t \to [0, 1]$, for a grid with δ squares at time t is,

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

Therefore the total safety score, r_{tot}, for a particular activity plan, $\mathcal{P}$, averaged over $\mathcal{T}$ is,

$$r_{tot} = \frac{1}{|\mathcal{T}|} \sum_{t=0}^{|\mathcal{T}|} r_{grid}(t). \tag{10}$$

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 $r_{agg}(g_i)$ for a square g_i then is,

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

F. Optimal Planning Model

We present the alternate job sequences with safety analysis attributes to planning managers who then carefully decide about a suitable plan according to project requirements. The safest plan is a partially ordered sequential plan which may take an undesirably long amount of time to finish a project. Therefore, we need a plan which optimizes both the risks and finishing times by using task parallelism. Exact solutions for multicriteria optimizations (often called *Pareto Optimal*) are NP-hard [6], so we focus on producing an approximate solution using our activity scheduler algorithm. The safety model then calculates the safety score over discrete time points. The acceptance of a plan, therefore is decided based on the following criterion:

- Average safety score(μ) of a site($\mu = r_{tot}$ from (10)).
- Standard deviation(σ) of r_{grid} over time.

If μ and σ both are low, this yields a very safe plan. If μ is low but σ is high, this is an acceptable plan. A plan with high values of μ and low values of σ is a very risky plan. Plans with high values of μ and high values of σ are also risky. For projects with deadline constraints, project managers can choose a higher safety score *threshold*. Below that *threshold*, a plan with high parallel activities is acceptable. If a planning manager wants to choose a specific period to be safer than another, episodic safety scores during a specific interval using (11) can be computed.

IV. Case Study

$EX1$ and $CP1$ in the activity graph of Figure 3 denote the excavation and concrete pouring activities in Building 1 and vice versa. Nodes S and F are dummy nodes to hold starting and final points. $EX1$ and $EX2$ are two starting activities. $CP1$ depends on $EX1$ while $CP2$ depends on $EX2$ and $CP1$.

A. Alternative Plans

We used the Python programming language to implement a topological sorting algorithm as proposed in [26]. Three alternate plans generated for the activity graph of Figure 3 are,

$\mathcal{P}_1 = [EX1, CP1, EX2, CP2]$
$\mathcal{P}_2 = [EX1, EX2, CP1, CP2]$
$\mathcal{P}_3 = [EX2, EX1, CP1, CP2]$.

B. Activity Scheduling

For the plan, $\mathcal{P}_1 = [EX1, CP1, EX2, CP2]$, the Activity Scheduler routine in Algorithm 1 loads $EX1$ in Q as its parent, the dummy starting node, S, is $Visited$. During the second iteration of the algorithm's loop, it tries to load $CP1$ whose precondition activity, $EX1$, is still not $Visited$. So $EX1$ is scheduled using Algorithm 2. Once the scheduling is done, $EX1$ is marked as $Visited$ and $CP1$ is loaded into Q. $EX2$ is also loaded in the next iteration, since its parent S has been $Visited$. $CP2$ is not loaded since $EX2$ has not been $Visited$. So, the two activities in the queue, $(CP1, EX2)$, are simulated simultaneously using the event scheduler. In the final run $CP2$ is simulated.

For the plan, $\mathcal{P}_2 = [EX1, EX2, CP1, CP2]$, the activities $(EX1,\ EX2)$ are loaded first into Q and simulated together. Then $(CP1, CP2)$ are simulated. The plan, $\mathcal{P}_3 = [EX2, EX1, CP1, CP2]$, can be simulated in a similar way.

C. 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 Figure 4(a) 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 Figure 4(b), a dump truck in charge of carrying soil has four such states: *Load(L), Haul(H), Dump(D) and Return(R)*.

- Therefore, the set of events for concrete pouring is $E^{CP} = \{L, RO, D\}$ and the set of events for excavation is $E^{EX} = \{L, H, D, R\}$.
- The state, Z, contains the configuration of all parameters such as *resources, interruption, deadline*, etc.

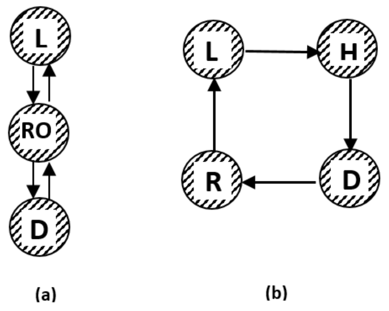


Fig. 4. (a) $DEVS$ state transition model for the crane. (b) $DEVS$ state transition model for the truck.

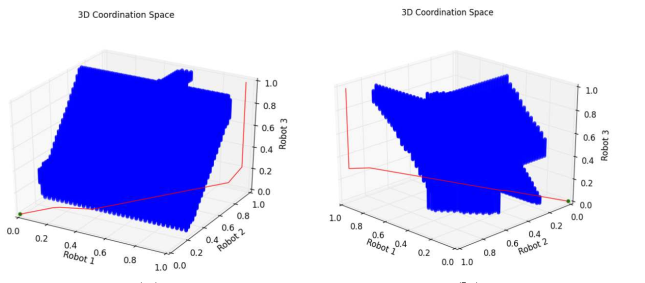
- The configuration of the dump truck is $\mathbb{R}^2 \times \mathbb{S}^1$ while the configuration for the *non-holonomic* [18] crane is $\mathbb{RP}^2$ as it can rotate with $pitch$ and yaw, but no $roll$.
- An example state we use for the truck is, $z = (x_{tr}, y_{tr}, \theta_{tr}, \eta_{ex}, r_{ex}, t_{ex})$, and an example state we use for the crane is, $z = (\theta_{cr}^{pitch}, \theta_{cr}^{yaw}, \eta_{cp}, r_{cp}, t_{cp})$.
- An example event transition for the dump truck is, $f_\eta^{EX}(L, z) = H$, as hauling is carried out after loading. Similarly, the crane starts rotating once it is loaded with concrete, $f_\eta^{CP}(L, z) = RO$ (See Figure 4).
- An example state transition for an excavation is, $f_z^{EX}(L, z) = (x_{tr}^{new}, y_{tr}^{new}, \theta_{tr}^{new}, H, r_{ex} - r', t_{ex} + t')$. $(x_{tr}^{new}, y_{tr}^{new}, \theta_{tr}^{new})$ 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.

D. Motion Planning and Coordination

We used the Motion Strategy Library (MSL) [9] to generate trajectories of moving equipment for different activities (See Figure 5(a)). Two trucks, colored red and green, from simultaneous activities $(EX1, EX2)$ are shown moving around the two building sites in Figure 5(a). Sample trajectories, $\tilde{x}_1, \tilde{x}_2$(colored blue and green), of the trucks are shown in Figure 5(b). Trucks have differential constraints. Suppose the speed of the truck and steering angle are specified by the actions u_s and u_ϕ respectively. The transition equation for two consecutive configurations is, $\dot{x}_{tr} = u_s \cos\theta_{tr}$, $\dot{y}_{tr} = u_s \sin\theta_{tr}$, $\dot{\theta}_{tr} = \frac{u_s}{L}\tan u_\phi$ [17], where L is the length of the truck.

The red trajectory in Figure 5(b) is the path of a moving worker. *Generalized Voronoi diagrams* were used in our previous work [22] to calculate the safest paths for workers while avoiding static obstacles. We added a worker trajectory from a *Voronoi diagram roadmap* for a moving worker to create a three robot problem. A trajectory for the crane's hook was generated by taking the arc of a circle centered at the base of the crane that goes through the initial and goal configurations of the hook.

The coordination space for the three moving bodies is shown in Figure 6. For better visual understanding we present the 3D 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, $(1, 1, 1)$.



(a) (b)

Fig. 6. Coordination space for robots from two different viewing angle. Blue regions are obstacle areas Γ_{obs}. Red line is the collision free path

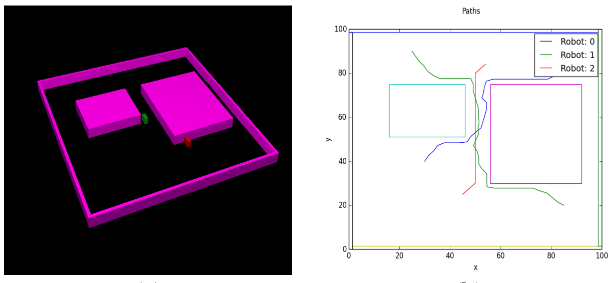
E. Safety Evaluation

We used Python to implement our safety model. It was mentioned earlier that the construction site is decomposed into a grid to calculate the safety score for individual squares in the grid. A safety score for each square was calculated in the range $[0, 1]$ using equations (7) and (8). Sample heatmaps were generated for the parallel activities $(EX1, EX2)$ at different times where two dump trucks are moving as shown in Figure 7. The green colored regions are the safest and red regions are the most dangerous. Other colors, ranging from green to red, were applied based on safety scores. The heatmaps give construction planning managers a detailed representation about the construction jobsite risks (i.e, the hotspots on the heatmaps) at discrete times.

Heatmaps based on aggregated safety scores(using 11) are shown in Figure 8 for two different sub-activities: $CP1EX2$ and $CP1CP2$. Aggregated scores, r_{agg}, for individual squares are calculated using (11) to give a concise episodic risk visualization in different areas of the construction site.

F. Optimal Plan Extraction

Two plans, $\mathcal{P}_1$ and $\mathcal{P}_2$, were evaluated and the following table is the result of the averages and standard deviations of safety scores for the entire life of the construction plan.



(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.

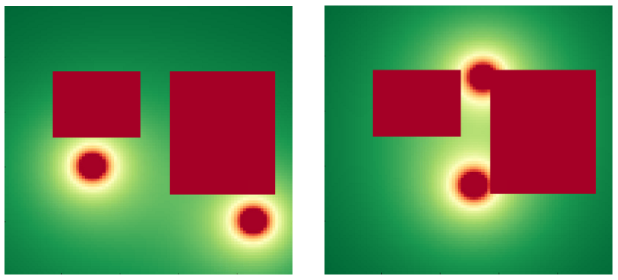

(a) (b)

Fig. 7. Sample heatmap snapshots/frames during time (a) t=0; (b) t=125.

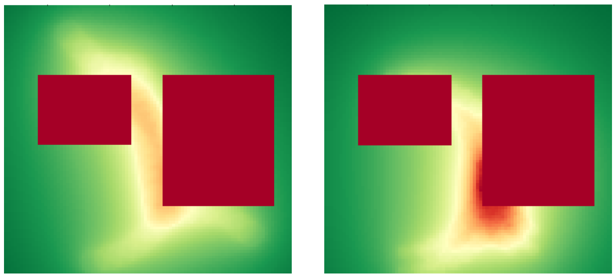

(a) (b)

Fig. 8. Aggregated heatmaps (using 11) for the activities (a) CP1EX2; (b) CP1CP2.

Plan	μ	σ
$\mathcal{P}_1$	0.16	0.024
$\mathcal{P}_2$	0.17	0.014

According to our heuristics from section III-F, $\mathcal{P}_1$ is the better plan as it has low average and standard deviation values compared to plan, $\mathcal{P}_2$. But using the heatmaps and aggregated scores, r_{agg}, the planning manager can choose alternative plans based on the construction project requirements. He can also take into account the partial safety scores and the safety scores at a grid level granularity. If the project has a tight deadline, then the safety threshold can be used for acceptable results.

V. Conclusions and Future Work

In this work, we developed an easily implementable methodology 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 simulate them using discrete event scheduler algorithms. Motion planning methodologies generate the collision free trajectories for the moving bodies. A proactive safety visualization is provided during preplanning phase using heatmaps which effectively distinguishes among the safe and dangerous places in a construction site.

The model presented can be used for most construction activities to carry out simulation and safety metrics together. The formalism presented in the paper provides measurement metrics to construction project managers, such as safety scores and time spent by a trajectory. Based on this measure, the safety policy and guidelines can be calculated for workers in a construction site.

One immediate extension of our work 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.

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

VI. Acknowledgements

We would like to thank Mr. Winston Newman and the loss prevention team at Balfour Beatty Construction Company for their valuable feedback on the ideas presented in the paper.

References

[1] www.osha.gov/SLTC/etools/construction/struckby/mainpage.html.
[2] Simpy: A process-based discrete-event simulation framework.
[3] Leading causes of fatal and non-fatal injuries in construction. *http://stopconstructionfalls.com/wp-content/uploads/2013/07/Leading-Causes-of-Fatal-and-Nonfatal-Injuries-in-Construction-2013-update.pdf*, 2013.
[4] Homam AlBahnassi and Amin Hammad. Near real-time motion planning and simulation of cranes in construction: Framework and system architecture. *Journal of Computing in Civil Engineering*, 26(1):54–63, 2011.
[5] Amit Bhatia, Lydia E Kavraki, and Moshe Y Vardi. Sampling-based motion planning with temporal goals. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2689–2696. IEEE, 2010.
[6] Altannar Chinchuluun and Panos M Pardalos. A survey of recent developments in multiobjective optimization. *Annals of Operations Research*, 154(1):29–50, 2007.
[7] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas. Temporal logic motion planning for dynamic mobile robots. *Automatica*, 45(2):343–352, February 2009.
[8] Amin Hammad and Cheng Zhang. Towards real-time simulation of construction activities considering spatio-temporal resolution requirements for improving safety and productivity. In *Simulation Conference (WSC), Proceedings of the 2011 Winter*, pages 3533–3544. IEEE, 2011.
[9] http://msl.cs.uiuc.edu/msl/. Motion strategy library.
[10] Vineet R Kamat and Julio C Martinez. Visualizing simulated construction operations in 3d. *Journal of Computing in Civil Engineering*, 15(4):329–337, 2001.
[11] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics & Automation*, 12(4):566–580, June 1996.
[12] Donald E Knuth and Jayme L Szwarcfiter. A structured program to generate all topological sorting arrangements. *Information Processing Letters*, 2(6):153–157, 1974.
[13] H. Kress-Gazit. *Transforming high level tasks to low level controllers*. PhD thesis, University of Pennsylvania, 2008.
[14] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. Translating structured english to robot controllers. *Advanced Robotics*, 22(12):1343–1359, 2008.
[15] S. M. LaValle and S. A. Hutchinson. An objective-based framework for motion planning under sensing and control uncertainties. *International Journal of Robotics Research*, 17(1):19–42, January 1998.
[16] S. M. LaValle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In B. R. Donald, K. M. Lynch, and D. Rus, editors, *Algorithmic and Computational Robotics: New Directions*, pages 293–308. A K Peters, Wellesley, MA, 2001.
[17] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.
[18] Yuanshan Lin, Di Wu, Xin Wang, Xiukun Wang, and Shunde Gao. Lift path planning for a nonholonomic crawler crane. *Automation in Construction*, 44:12–24, 2014.
[19] Ming Lu and Heng Li. Resource-activity critical-path method for construction planning. *Journal of construction Engineering and Management*, 129(4):412–420, 2003.
[20] J. C. Martinez. Stroboscope: State and resource based simulation of construction processes. *Doctoral dissertation*, 1996.
[21] Lam K. C. Ning X. Costsafety trade-off in unequal-area construction site layout planning. *Automation in Construction*, 32:96–103, 2013.
[22] 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 *Automation Science and Engineering (CASE), 2014 IEEE International Conference on*, pages 277–282.
[23] S Russell and P Norvig. The chapter 3 in . *Artificial Intelligence–A Modern Approach, Prentice Hall, Inc*, 2009.
[24] Hans Vangheluwe. The discrete event system specification (devs) formalism. Technical report.
[25] Hans Vangheluwe. Discrete event modelling and simulation;http://www.cs.mcgill.ca/ hv/classes/ms/discreteevent.pdf. 2001.
[26] Yaakov L. Varol and Doron Rotem. An algorithm to generate all topological sorting arrangements. *The Computer Journal*, 24(1):83–84, 1981.
[27] Bernard P Zeigler. *Multifacetted modelling and discrete event simulation*. Academic Press Professional, Inc., 1984.
[28] Cheng Zhang, H AlBahnassi, and A Hammad. Improving construction safety through real-time motion planning of cranes. In *Proceedings of International Conference on Computing in Civil and Building Engineering*, pages 105–115, 2010.
[29] Cheng Zhang, Amin Hammad, and Homam Bahnassi. Collaborative multi-agent systems for construction equipment based on real-time field data capturing. *Electron. J. Inf. Technol. Constr.*, 14:204–28, 2009.
[30] Cheng Zhang, Amin Hammad, and Jamal Bentahar. Multi-agent-based approach for real-time collision avoidance and path re-planning on construction sites. In *28th Int. Symp. on Autom. & Robotics in Const., Seoul*, 2011.