AN "XCITY" OPTIMIZATION APPROACH TO DESIGNING PROVING GROUNDS FOR CONNECTED AND AUTONOMOUS VEHICLES

**Rui Chen**  
University of Michigan, Ann Arbor.  
500 S. State Street, Ann Arbor, MI 48109 USA  
Email: richen@umich.edu

**Mansur Arief**  
Carnegie Mellon University  
5000 Forbes Avenue, Pittsburgh, PA 15213  
Email: marief@andrew.cmu.edu

**Ding Zhao, Ph.D.**  
Carnegie Mellon University  
5000 Forbes Avenue, Pittsburgh, PA 15213  
Email: dingzhao@cmu.edu

Word Count: 5096 words + 8 figures × 250 + 1 table × 250 = 7346 words  
Submission Date: August 1, 2018
ABSTRACT

Proving ground, or on-track testing has been an essential part of testing and validation process for connected and autonomous vehicles (CAV). Several world-class CAV proving grounds, such as Mcity at the University of Michigan and The Castle of Waymo, have already been built, and many more are currently under construction. In this paper, we propose the first optimization approach to CAV proving ground designing and refer to any such CAV-centric design problem as “Xcity” to emphasize the enormous investment, the multi-dimensional spatial consideration, and the immense construction effort emerging globally. Inspired by the recent progress on traffic encounter clustering, we further define “road assets” as fundamental building blocks and formulate the whole design process into nonlinear optimization problems. We have shown that such framework can be utilized to adaptively generate CAV proving ground designs with optimized capability and flexibility and can further be extended to evaluate an existing “Xcity” design.

*Keywords:* connected and autonomous vehicle, testing and validation, proving ground design, traffic encounter, optimization
INTRODUCTION

Development of self-driving technologies has drawn significant public attention with major concerns on safety issues. Extensive efforts have been made on testing and validation (T&V) of both autonomy and safety of connected and autonomous vehicle (CAV) systems before deployment. The "V-model" for CAV systems \(^1\) illustrates the T&V process from component level to traffic level. An approach for such process, visualized as a T&V pyramid \(^2\), has identified proving ground testing as an essential level required by CAV systems. CAV proving ground is a reserved area designed and constructed to simulate real-world traffic environment. The fully controllable environment of proving grounds provides CAV testing and validation with foreseeable risks and flexible fidelities. Those advantages have led to the establishment of several world-class CAV proving grounds: Mcity at the University of Michigan \(^3\), "The Castle" of Waymo \(^4\), Uber’s ALMONO \(^5\), Smart Mobility Advanced Research and Test Center (SMART) by Transportation Research Center (TRC) \(^6\), Security Smart Mobility Analysis and Research Test (SMART) Range in Israel by HARMAN \(^7\), American Center for Mobility (ACM)’s test facility at Willow Run in Michigan \(^8\), etc. Since massive time and resources have been invested in CAV proving ground construction, it is natural and essential to aim at effective and efficient designs. However, design theory of CAV proving grounds is rarely seen in the existing literature.

When designing an efficient and effective CAV proving ground, one should make clear the testing capability required by CAV T&V starting from elementary system integration, such as path following, till high-fidelity traffic simulation, such as CAV tests. Due to the infeasibility of complete testing on CAV systems \(^9\), it is natural for one to break multi-vehicle-multi-event and long-term CAV tests into a variety of small and distinguished use cases which represent typical usage scenarios for autonomous driving \(^10\). Such typical use cases or scenarios are similarly identified in \(^11,12\) as behavioral competency, the ability of a CAV to operate in traffic conditions which it will regularly encounter; these conditions include keeping the vehicle in the lane, obeying traffic laws, following reasonable etiquette, and responding to other vehicles, road users, or commonly encountered hazards. Due to the highly complicated and stochastic nature of real traffic, it would be impractical to require tests for all possible use cases. Therefore, it is reasonable for one to identify minimum behavioral competencies as required. Such efforts lead to CAV capability checklists which will be practical for one to use as evaluation criterion for CAV systems \(^11,13\). Thus, a CAV proving ground is considered both practical and effective when it provides wide coverage of identified behavioral competency tests within constrained space. To the best of our knowledge, the literature provides no such systematic approach that employs optimization models to maximize the testing capability of a CAV proving design. This work contributes to the formulation of a systematic proving ground design that cognizantly addresses the CAV evaluation challenges to assist the design of CAV proving grounds in terms of maximizing testing capability and flexibility.

Challenges of systematically designing an effective and efficient CAV proving ground come from two aspects: a) it is not clear how to classify and extract typical driving scenarios from a large scale naturalistic driving data; b) it is not clear how to map those scenarios into space-constrained proving grounds. To address these problems, we propose the development of Xcity: a mathematical approach to design CAV proving ground road layout based on traffic encounter theory \(^14-16\) and nonlinear optimization methods. We applied and illustrated our framework with a minimal example. Then we extracted driving scenarios from Mcity road map and from clustered driving scenarios by \(^15\), thereby exploring the scalability of our approach.
XCITY DESIGN FRAMEWORK

The underlying idea of Xcity design framework is to generate a CAV proving ground road map which provides maximum coverage of typical CAV driving scenarios within constrained space and maximum flexibility of scenario transitions. In this section, we describe the systematic approach in terms of scenario extraction and representation, scenario selection, and placement optimization.

As mentioned in introduction section, we evaluate the capability of a CAV proving ground based on the CAV driving scenarios it supports. It is worth mentioning that this paper focuses on scenarios involving at least two vehicles. The reason behind this is that single-vehicle tests such as lane following, traffic rule obeying, and intersection navigations can be triggered by simply removing all the other road users from the corresponding multi-vehicle scenarios, e.g. car-following including Stop & Go & Emergency Stop and 2-way stops obeying with cross traffic ([7], [13]). Thus, we argue that the evaluation requirements of single-vehicle tests can be trivially incorporated whenever more complicated multi-vehicle scenarios are conductable.

Therefore, it is essential to first develop a systematic way to identify, extract, and represent typical multi-vehicle CAV usage cases from a large-scale driving data. Such task is non-trivial since the real-world traffic is a vast, stochastic, and dynamic cyber-physical system consisting of large number of road users, non-road users, and stochastic environment factors ([14]). The method developed in ([14]–[16]) allows the processing of massive multidimensional traffic data to extract traffic primitives and traffic encounters. In this context, traffic primitives, which are the most informative traffic features from the dataset, are then used as principal compositions of the entire traffic ([14]). Traffic encounters, which represent the scenario where two or multiple vehicles are spatially close to and interact with each other on roadways ([15]), allow higher fidelity representation of sophisticated driving scenarios.

Our design approach aims to map such scenarios into the given space in a way that maximizes CAV evaluation capability. To enable tests on a particular scenario, we map the group of roads and intersections which supports the realization of traffic encounters into the proving ground design. Additional factors such as other road users, traffic control devices, and control commands are then embedded on demand according to the test specifications. For example, one could add two stops signs and deploy one other vehicle at an X-intersection to support the scenario of 2-way stop-obeying with cross traffic. With the same physical layout, one could also evaluate any scenarios of the form left-turn-on-green with incoming traffic. Therefore, we consider the combination of roads and intersections as basic construction element in our design formulation and refer to such combination as road assets hereafter. Once selected and mapped into the design, a road asset a will be treated as a 2D rigid body afterwards and all corresponding testing scenarios it supports (based on traffic-primitives analysis) will be enabled. Furthermore, we introduce a value metric $\nu : a \rightarrow \mathbb{R}$ to evaluate road assets since the number of supported scenarios for different assets could vary. For a road asset $a$, its value comes from both its versatility to support multiple scenarios and the universality of the supported scenarios.

Given a preprocessed set of road assets $A = \{a_1, a_2, ..., a_{NA}\}$, value measure $\nu$, and constrained construction space $S$, we propose the following constraints, definitions, and objectives to model an Xcity design problem.

**Constraints:**

1. Roads in any single asset $a_i$ cannot overlap with roads in other assets.
2. Road assets should completely reside within the constrained space $S$.

**Definitions:**

1. A subset, $A' = \{a'_1, a'_2, \ldots, a'_{NA'}\}$, of $A$ is considered feasible if all road assets in $A'$ can be mapped into $S$ simultaneously under the above constraints.

2. Two road assets $a_i$ and $a_j$ are considered directly connectible if it is possible to construct a straight transition road to connect them without colliding with any other assets. The number of directly connectible asset pairs $(a_i, a_j)$ in a road asset set $A$ is referred to as direct connectivity hereafter and denoted as $C : A \rightarrow \mathbb{R}$.

**Objectives:**

1. The feasible subset $A^*$ with highest total asset value $\nu(A^*)$ is selected and mapped in $S$.

2. The assets in $A^*$ are arranged within $S$ such that $C(A^*)$ is maximized.

**Constraint 1** guarantees that no test fidelity is lost due to the physical design. Each scenario test should be conducted in its originally dedicated road asset with unaltered roads and intersections. Merging distinguished assets would generate undesired and inaccurate physical environment, which attenuates the fidelity of the simulated scenario. It would consequently cause non-measurable degeneration to the confidence of public street deployment gained from on-track tests.

As previously mentioned, one criteria which we use to evaluate a CAV proving ground is its coverage of behavioral competency (BC) tests. BC is defined as a list of independent scenarios that are expected to be handled safely by a sophisticated autonomous driving system. Since BC implies no connections between different scenarios, we treat the mapped road assets independently and simply accumulate their value measures to form **Objective 1**. On the other hand, Proving ground test and validation, a comprehensive and multi-integration-level process, calls for not only a wide range of independent scenario-based tests but also cascaded and potentially randomly ordered scenarios to assess CAV systems’ robustness and performance on enduring operations. Thus, the success in emulating scenario sequences further boosts the flexibility and fidelity of the proving ground. Tests involving multiple independent but connected scenarios highly demand that the CAV proving ground be flexible on navigating among different scenarios. In our case, such flexibility will be ensured by transition roads built to connect different road assets. Although roads with arbitrary shapes and lengths would highly likely to serve as transitions between any pair of road assets, it is worth looking for a design with more directly connectible asset pairs. Therefore, we only consider straight transitions for evaluation purpose. The flexibility and fidelity measure depending on possible straight transition roads are defined as **Objective 2**. Note that the the outcome of this paper will be a high-fidelity graph representation of the road map structure. Detailed information on road construction, such as exact road curvatures, lane widths, and sidewalks, needs to be added to our design, so as to achieve a final construction blueprint. Further discussion on blueprint rendering is out of scope and is omitted.

**METHODOLOGY**

In this section, we describe the mathematical formulation and optimization models of our approach. With road assets pre-produced and represented, the design process is decomposed into two phases. In phase 1, the feasible road asset subset $A^*$ with highest total asset value $\nu(A^*)$ is selected from all pre-produced road assets $A$. In phase 2, the placement of the pre-selected $A^*$ within given space
$S$ will be optimized in terms of cross-scenario transition flexibility. Both two phases are modeled into nonlinear optimization problems.

**Road Asset Representation**

Traffic encounters are clustered as GPS location sequences of involved vehicles. Thus, given a traffic encounter, one natural and direct approach to retrieve the essential *road asset* is to directly locate the traffic encounter on the world map and extract surrounding roads and intersections. To map the extracted *road assets* into the given space, we will need to define a geometric descriptor for their structure and size. On one hand, such descriptor should be rigorous enough for accurate localization and space-reserving for the corresponding asset. On the other hand, the descriptor should be concise enough to truncate unnecessary computations during optimization. Two of the most recognized map representations for autonomous driving are Lanelets (17) and OpenDRIVE (18). Both of them are efficient and detailed enough for semantic environment modeling and motion planning. However, our approach only aims at the placement and transitions of *road assets* regardless of their semantic meanings or navigational information. These knowledge is already contained in the supported scenarios and can be applied on demand after the design is finished. Therefore, both Lanelets and OpenDRIVE contain more information than needed and create unnecessary computations.

![FIGURE 1 Examples of road asset structure graph and corresponding extracted traffic encounters. Deep red and blue dots are the start positions of vehicles. White dots are the end positions of vehicles. Yellow dots refer to nodes, which are connected by yellow lines representing internal segments. Nodes are labeled with their indexes within the assets. Red circles with white $x'$ label, in (e)-(f), mark nodes that are ignored for graph simplification.](image)

Considering both representativeness and simplicity, we use graphs to model the geometric structure of *road assets* (see examples in Figure [1]). Such descriptor is inspired by the map format from OpenStreetMap (OSM) (19), a collaborative project to create a free editable map of the world.
We mark the intersections, road ends, and way-points along curved roads with nodes \( x^{(i)} \). Then, we connect each node pair \((x^{(i)}, x^{(j)})\), if necessary, with a line segment \( \ell(x^{(i)}, x^{(j)}) \) to model one road segment in the real world. We refer to those line segments as the internal segments of road assets. We ignore nodes that are unnecessary or trivial, thereby maintaining the whole structure to introduce the least possible internal segments and save computation efforts. Typical trivial nodes include nodes at X-intersection centers (see Figure 1.e). Notably, the number of internal segments turns out to be one major factor contributing to model complexity.

**Phase 1: Most-valuable Feasible Subset Selection Model**

**Single-Asset Constraint Set (SACS)**

We map road assets into the given space \( S \) by mapping their corresponding graph representations while retaining the relative placement of the nodes. This is done through the introduction of a set of 2D geometric constraints on the nodes of which the original node placement is the only realization. A non-trivial start point would be mapping a triangle defined by the original vertex positions \( x^{(i)}_o, x^{(j)}_o, x^{(k)}_o \in \mathbb{R}^2 \) into space \( S \) at \( x^{(i)}, x^{(j)}, x^{(k)} \in \mathbb{R}^2 \) without deformation or flipping. We propose the following constraint set \( K_{\Delta}(x^{(i)}, x^{(j)}, x^{(k)}) \), referred to as triplet constraint set, which retains the triangle’s original shape and size. Note that the superscript number refers to the node index within a single asset. The subscript ‘o’ refers to the original location of nodes on world map.

\[
\begin{align*}
\text{dist}(x^{(i)}, x^{(j)})^2 &= \text{dist}(x^{(i)}_o, x^{(j)}_o)^2 + \delta_{ij} \quad (1) \\
\text{dist}(x^{(i)}, x^{(k)})^2 &= \text{dist}(x^{(i)}_o, x^{(k)}_o)^2 + \delta_{ik} \quad (2) \\
\text{dist}(x^{(j)}, x^{(k)})^2 &= \text{dist}(x^{(j)}_o, x^{(k)}_o)^2 + \delta_{jk} \quad (3) \\
\Omega_{CCW} &= \mathbb{I} \left[ \Omega(x^{(i)}_o, x^{(j)}_o, x^{(k)}_o) > 0 \right] \quad (4) \\
(-1)^{\Omega_{CCW}} \Omega(x^{(i)}, x^{(j)}, x^{(k)}) &\leq 0 \quad (5) \\
x^{(i)}, x^{(j)}, x^{(k)} &\in S \quad (6)
\end{align*}
\]

\( \Omega(x,y,z) \), the orientation test, is positive if \( x,y,z \in \mathbb{R}^{2\times1} \) are placed counter-clock-wise (CCW), or negative if clock-wise (CW), or zero if collinear (LNR) \((7)\). See Equation \([7]\) for its definition.

The \( \delta \) variables in constraint \([3]\) are introduced as slack variables for equality constraints to avoid numerical issues in solvers. The sum of all squared slack variables will be include in the objective to be minimized.

\[
\Omega(x,y,z) = \det \begin{bmatrix}
x^T & 1 \\
y^T & 1 \\
z^T & 1
\end{bmatrix}
\]

The relative placement of all nodes \( x^{(1)}_r, x^{(2)}_r, ..., x^{(n_r)}_r \in \mathbb{R}^2 \) in road asset \( a_r \) can be retained by applying \( K_{\Delta}(x^{(1)}, x^{(2)}, x^{(k)}), k \geq 3 \) repeatedly to nodes in \( a_r \). We refer to the constraint set which regulates the relative placements of all nodes within a single asset \( a_r \) as Single-Asset Constraint Set (SACS), denoted by \( K_{sa}(a_r) \) (see constraint set \([8]\)).

\[
K_{sa}(a_r) := \{ K_{\Delta}(x^{(1)}, x^{(2)}, x^{(k)}) \ | k = 3, 4, ..., n_r \} \quad (8)
\]
FIGURE 2 Example of mapping a single road asset into space $S$. With $x^{(1)}$ and $x^{(2)}$ satisfying their distance constraint, each of the remaining nodes $x^{(k)}$ can be located uniquely by applying a triplet constraint set with respect to $x^{(1)}$, $x^{(2)}$, and $x^{(k)}$.

Cross-Asset Constraint Set (CACS)

Another requirement of road asset mapping is that internal segments of different road assets should not collide or intersect. To formulate such requirement into constraints, we first define the intersection test $\chi(\mathcal{L}, \ell)$ for straight line $\mathcal{L}$ passing points $x^{(1)}_{\mathcal{L}}, x^{(2)}_{\mathcal{L}} \in \mathbb{R}^2$ and line segment $\ell$ formed by points $x^{(1)}_{\ell}, x^{(2)}_{\ell} \in \mathbb{R}^2$ as Equation 9:

$$\chi(\mathcal{L}, \ell) = \Omega(x^{(1)}_{\mathcal{L}}, x^{(2)}_{\mathcal{L}}, x^{(1)}_{\ell}) \Omega(x^{(1)}_{\mathcal{L}}, x^{(2)}_{\mathcal{L}}, x^{(2)}_{\ell})$$

$\chi(\mathcal{L}, \ell)$ is then positive if $\ell$ doesn’t intersect with $\mathcal{L}$ and non-positive otherwise. See Figure 3 for illustrative examples.

FIGURE 3 Examples of intersection test with straight line $\mathcal{L}$ and line segment $\ell$. (a) $\Omega(x^{(1)}_{\mathcal{L}}, x^{(2)}_{\mathcal{L}}, x^{(1)}_{\ell}) > 0 \Rightarrow \chi(\mathcal{L}, \ell) > 0$. (b) $\Omega(x^{(1)}_{\mathcal{L}}, x^{(2)}_{\mathcal{L}}, x^{(2)}_{\ell}) = 0 \Rightarrow \chi(\mathcal{L}, \ell) = 0$. (c) $\Omega(x^{(1)}_{\mathcal{L}}, x^{(2)}_{\mathcal{L}}, x^{(1)}_{\ell}) > 0 \Rightarrow \chi(\mathcal{L}, \ell) < 0$.

Given two internal segments $\ell_p \equiv \ell(x^{(1)}_p, x^{(2)}_p) \in a_i$ and $\ell_q \equiv \ell(x^{(1)}_q, x^{(2)}_q) \in a_j$, we denote the straight lines passing $(x^{(1)}_p, x^{(2)}_p)$ and $(x^{(1)}_q, x^{(2)}_q)$ as $\mathcal{L}_p \equiv \mathcal{L}(x^{(1)}_p, x^{(2)}_p)$ and $\mathcal{L}_q \equiv \mathcal{L}(x^{(1)}_q, x^{(2)}_q)$ respec-
tively. Then \( \ell_1 \) doesn’t intersect with \( \ell_2 \) \((\ell_1 \cap \ell_2 = \emptyset)\) if Constraint 10 holds.

\[
\chi(\mathcal{L}_p, \ell_q) > 0 \lor \chi(\mathcal{L}_q, \ell_p) > 0.
\] (10)

To simplify the constraints, we skip considering the corner case where \( \ell_p \) and \( \ell_q \) are collinear but not overlapped. Such case can be trivially made feasible under our constraints by rotating or moving one of the line segments by a tiny amount. It is worth mentioning that, since strict inequalities are not permitted in most solvers, we implement constraint 10 using non-strict equalities with a constant offset \( \epsilon = 10^{-4} \). Auxiliary binary variables \( b_1, b_2 \in \{0, 1\} \) are introduced as well to decompose the logic OR statement into a set of AND statements for each instance of constraint 10. Constraint 10 is finally formed as constraint set 11. We refer to such constraint set that eliminates the collision of \( \ell_p \in a_i \) and \( \ell_q \in a_j \) as Cross-Asset Constraint Set (CACS), denoted as \( K_{ca}(\ell_p, \ell_q) \).

\[
K_{ca}(\ell_p, \ell_q) : \quad b_1 \cdot \chi(\mathcal{L}_p, \ell_q) \geq b_1 \cdot \epsilon, \quad b_2 \cdot \chi(\mathcal{L}_q, \ell_p) \geq b_2 \cdot \epsilon, \quad b_1 + b_2 \geq 1
\] (11)

**Phase 1 Optimization Model**

With both SACS and CACS defined, we can formulate phase 1 in a single optimization model (see Expression 12). For each subset \( A' = \{a'_1, a'_2, \ldots, a'_{NA'}\} \subseteq A \), let \( X = \{x^{(1)}_1, x^{(2)}_1, \ldots, x^{(n)}_1, x^{(1)}_2, x^{(2)}_2, \ldots, x^{(n)}_2, \ldots, x^{(1)}_{NA'}, x^{(2)}_{NA'}, \ldots, x^{(n)}_{NA'}\} \) be the collection of all mapped nodes in space \( S \) from \( A' \). Let \( \delta \in \mathbb{R}^{N_A \times 1} \) be a column vector containing all slack variables introduced in distance equality constraints. Let \( b_{CACS} \) be the set of binary variables introduced in CACSs.

\[
\begin{align*}
\text{minimize} & \quad \delta^T \cdot \delta - \nu(A') \\
\text{subject to} & \quad \sum_{a \in A' \subseteq A} \chi(a) \quad \tau = 1, 2, \ldots, NA' \\
& \quad K_{ca}(\ell_p, \ell_q) \quad \forall \ell_p \in a'_i, \ell_q \in a'_j \Rightarrow a'_i, a'_j \in A', i \neq j
\end{align*}
\] (12a, b, c)

Notably, we factor out the selection of asset subset \( A' \) during implementation and manually assign the *road assets* to be mapped. When global minimum of the remaining objective \( \delta^T \cdot \delta \) is reached, all distance Constraints 12 are strictly satisfied. The corresponding \( X \) will be a feasible mapping of \( A' \) into \( S \).

**Phase 2: Transition Flexibility Optimization Model**

With the most valuable feasible subset \( A^* \) selected in phase 1, we will find a placement \( X^* \) of all containing *road assets* such that the *direct connectivity*, \( C(A^*) \), is maximized.

**Asset Transition**

In the graph representation of a *road asset*, we define nodes that are owned by only one *internal segment* as *boundary nodes* (for instance, \( x^{(1)}, x^{(2)}, x^{(5)} \) in Figure 1[a]) and the rest as *internal nodes*. To avoid causing bias to the road layout and achieve high-fidelity scenarios, we only allow *road assets* to be connected to each other using *boundary nodes* and skip the *internal nodes* when considering transition road construction. Two *road assets* are considered directly connectible if it is possible to construct at least one straight transition road between them using only *boundary nodes* without intersecting any *internal segments* of any *road asset*. See Figure 4 for a mini example.
FIGURE 4 Example of possible transition roads between two road assets. Blue dashed lines indicate possible transition roads. Red dashed lines indicate infeasible transition roads due to use of internal nodes \((x^{(4)}_i)\) and intersection with internal segments \((\ell(x^{(4)}_i, x^{(5)}_i))\).

**Phase 2 Optimization Model**

Given a the pre-selected best feasible asset set \(A^*\) from phase 1, the objective is to maximize \(C(A^*)\) subject to both SACS and CACS. Given a road asset \(a_i\) with nodes \(\{x^{(1)}_i, x^{(1)}_i, \ldots, x^{(n_i)}_i\}\), we propose the following definitions. Note that the superscript * indicates that the variable is pre-selected in phase 1 and kept constant in phase 2.

- \(b_{dc} = \{b_{dc}(a^*_i, a^*_j) \mid \forall a^*_i, a^*_j \in A^*, i \neq j\}\): set of binary decision variables indicating the direct connecting status of \(a_i\) and \(a_j\).
- \(\beta(a^*_i)\): set of boundary node in \(a_i\).
- \(L_{tr}(A^*)\): set of transition road candidates \(\{\ell_{tr}(x^{(p)}_i, x^{(q)}_j) \mid x^{(p)}_i \in \beta(a^*_i), x^{(q)}_j \in \beta(a^*_j), i \neq j\}\).
- \(b_{tr}(x^{(p)}_i, x^{(q)}_j)\): binary variable which indicates the selection status of \(\ell_{tr}(x^{(p)}_i, x^{(q)}_j)\).
- \(L_{\chi}(\ell_{tr})\): set of internal segments that don’t have common node with \(\ell_{tr}\) by construction.

With SACS \(K_{sa}\) and CACS \(K_{ca}\) defined in Equation 8 and 11, we can write the complete optimization model as Expression \(13\).

\[
\begin{align*}
\text{minimize} & \quad \delta^T \delta - \sum \sum b_{dc} & \quad & A^* \\
\text{subject to} & \quad K_{sa}(a_\tau) & \quad & \tau = 1, \ldots, NA^*, \quad (13a) \\
& \quad K_{ca}(\ell_p, \ell_q) & \quad & \forall \ell_p \in a^*_i, \ell_q \in a^*_j \Rightarrow a^*_i, a^*_j \in A^*, i \neq j, \quad (13b) \\
& \sum \sum b_{tr}(x^{(p)}_i, x^{(q)}_j) & \quad & \geq 1 \quad \forall a^*_i, a^*_j \in A^*, i \neq j, \quad (13c) \\
& \quad K_{ca}(\ell_{tr}, \ell_{\chi}) & \quad & \forall \ell_{tr} \Rightarrow b_{tr}(\ell_{tr})_{true}, \forall \ell_{\chi} \in L_{\chi}(\ell_{tr}) \quad (13e)
\end{align*}
\]

Constraints [13b] and [13c] ensure that the feasibility of the mapping as identically done in phase 1. Constraint [13d] ensures that for each pair of assets to be connected, at least one transition road candidate connecting their boundary nodes should be selected. Constraint [13e] ensures that each selected transition road candidate should not intersect with any other internal segment unless they share a node by construction. In Figure 4, \(\ell(x^{(4)}_1, x^{(5)}_1)\) to \(\ell_{tr}(x^{(5)}_1, x^{(1)}_2)\) is one example of such exception.
OPTIMIZATION APPROACHES

Based on our formulation, the resulting model is NP hard, essentially due to the extensive use of binaries to ensure the validity of the asset shapes and their combined designs. From (12) and (13), it is clear that even though the final objective function in (13) is arguably convex, some of the constraints in both (12) and (13) are not. As such, the model is of mixed integer programming (IP) type and the natural approaches to numerically deal with it take on some form of branch-and-bound (BNB) methods. Loosely speaking, the computation procedure of BNB requires lower-and upper-bound governing how variable branches are determined to obtain integer solution at each numerical iteration (22). Modern solvers employ various numerical scheme deemed best to solve these branch-level problems, e.g. BMIBNB use linear relaxations to solve for the bounds from the branch problems (23). Furthermore, other solvers have gained improved efficiencies from more sophisticated techniques to assist the branching, e.g. SCIP (24) employs branch-cut-and-price (BCP) while BARON (25) uses branch-and-reduce (BAR). In what follows, we describe the performance of these three solvers in regard to our formulation.

BMIBNB is a YALMIP-based BNB technique that can be easily implemented using standard computing environment Matlab (23). YALMIP itself is an interface program that is originally designed to allow Matlab to solve semi definite programming problems using external solvers including SeDuMi and Mosek, but its implementation has currently included wider array of choices including fmincon, SDPT3, Gurobi, and CPLEX which can be tuned easily (23) to obtain the numerical bounds. In dealing with nonlinear IP, BMIBNB uses linear relaxation and convex envelope approximations. As such, it trades the accuracy of solution optimality for the efficiency of computation. With this view, one shall notice that when the original problem is non-convex and high-dimensional, then the performance of BMIBNB suffers from yet-to-converge bounds. Unfortunately, our Xcity design problem is one such program and thus, as expected, the corresponding numerical experiment advises to using more customized solvers for our formulation.

SCIP is a BNB-based solver for integer programming enhanced with BCP framework which was designed to handle large-scale discrete optimization problem. The cuts and variable can be generated dynamically through the branching process to explore the direction of optimality. This dynamic ‘exploration’ capability is achieved through the implementation of column generation technique, which further implies SCIP has adopted the large-scale Dantzig-Wolfe decomposition method (24, 26). Furthermore, the most recent version has included Bender’s decomposition, allowing the solver to efficiently deal with problems with block-angular characteristics (27). Our preliminary numerical experiment shows that the SCIP’s inherent column generation does not yield an appealing performance due to the insurmountable growth of the constraint size in both the number and shape-complexity of the assets.

BARON (Branch-And-Reduce Optimization Navigator) is designed to solve a generalized nonlinear programs including mixed integer nonlinear problem (25, 28). The underlying idea of this technique is that after branching, a numerical analysis is performed to reduce the potential optimal regions which in turn allows a faster computation, and if conducted appropriately, global optimality assurance (29). This notion of ‘appropriateness’ is achieved by adapting the cutting plane and probing techniques from mixed integer linear programming to mixed integer nonlinear problems. We refer interested readers to (29) for extensive computational experiment investigating the effectiveness of BARON scheme to achieve optimality. Our numerical results show that BARON indeed yields to much better performance both in terms of improved computational ef-
Rui, Mansur, and Ding

Efficiency and solution quality compared to BMIBNB and SCIP. As such, we will use BARON for our further analysis in this study. It is noted that BARON requires proper license in order to be deployed.

EXPERIMENTS AND RESULTS

Road Asset Collection

From totally 49,998 vehicle encounters recorded by the University of Michigan Safety Pilot Model Development (SPMD) program, (15) selected 2,568 naturalistic driving encounters with adequate duration. 10 typical driving scenarios were then successfully extracted. Since there is currently no method that can analytically map traffic encounters to necessary roads and intersections, we extract road assets through manual annotation on OSM-logged road maps. After locating typical driving encounters on OSM via GPS locations, we exported the necessary surrounding road structure in .osm format and imported it into MATLAB using (30) (see Figure 5). Besides, we export the OSM road map of CAV proving ground at Mcity (3) and segmented it into 22 road assets which support different typical driving scenarios (see Figure 6). We down-sampled the nodes in OSM raw data to simplify graph representations and assign random integer to value each road asset.

FIGURE 5 Example of extracted road asset from a traffic encounter by (15). (a) Orange and blue curves refer to GPS trajectory of both vehicles. Blue dashed lines indicate OSM-logged roads and sidewalks. Red dots refer to nodes. (b) Road asset is extracted based on available nodes. Red dots refer to nodes. Solid lines represent internal segments

Solver Performance on Phase 1

We run numerical experiments on BMIBNB, SCIP, BARON to choose the one we will use to proceed. Since BMIBNB is a BNB technique that relies on external solvers, we test it with two solver configurations: a) LP: SCIP, lower bound: SeDuMi, upper bound: fmincon; b) LP: Gurobi, lower bound: Gurobi, upper bound: fmincon. The preliminary testing suite is based on model 12 with three trivial road asset candidates. We initiate 7 tests with each one challenging the solver to map one non-empty subset of those assets into a given squared space. One single asset is selected in trial 1 to 3; two are involved in trial 4 to 6; and finally all three assets are selected in trial 7. Solving time for each tested solver is recorded and plotted (see Figure 7).

We set the timeout to 200s since the tests are based on trivial road assets with no more than four nodes each. Such timeout turns out to be more than enough because BARON passed every
FIGURE 6 Left: OSM-logged Mcity road map. Blue dashed lines indicate roads and sidewalks. Red dots refer to nodes. Middle: OSM map of Mcity overlapped with extracted road assets. Right: All road assets extracted. Road assets that support common types of scenarios are shown in the same color.

FIGURE 7 Left: Solver performance on preliminary test suite. Note that the optimization timeout has been set to 200 s. Right: Preliminary road assets used in the benchmark.
	rial in around 1 second. Solving times reaching 200s indicate failed optimization due to timeout. Comparing the performance of two BMIBNB configurations on trial 5 and 7, we notice that the performance of BMIBNB highly depends on the utilized external solvers. SCIP ranks the last with 3 failed trials. We choose BARON to proceed with due to its obvious advantage over alternatives.
Model Scalability and Complexity Analysis

Using BARON, we explore the scalability of our optimization models with road assets $A_M$ which are extracted from Mcity road map. We gradually increase the number of available road assets until BARON fails to solve within 100s. First, we run feasible mapping searches (phase 1) with 1, 2, and 3 randomly selected road assets. BARON passed all tests with average solving time among 10 trials being 0.1678s, 2.9325s, and 30.9906s respectively for each road asset quantity. Then, we run flexibility optimizations (phase 2) with all three preliminary road assets and most valuable subsets $A^*$ from previous tests. See Figure 8 for example outputs.

FIGURE 8 Examples of (a) feasible mapping search (phase 1), (b)-(c) flexibility optimization (phase 2). Blue dashed lines indicate the transition roads added by Xcity design process.

Having solved smaller problems successfully, BARON fails in searching for a feasible mapping for more than four assets in phase 1, or in optimizing transition flexibility of more than three assets in phase 2. In both cases, BARON fails to secure any feasible solution during preprocessing. The upper bound consequently stays at default value ($10^{51}$) after more than 5 hours’ solving. Noticing the significant performance drop in solving, we analyze our model complexity and present the results in this section. We calculate the number of nonlinear constraints and binary variables introduced by each constraint set (see Table 1). Some notations are explained as follow:

- $n_i$: number of nodes in road asset $a_i$.
- $\ell_i$: all internal segments in $a_i$.
- $N_\Delta$: cardinality of $\Delta$.
- $X = \sum_{a_i,a_j}N_{\beta(a_i)}N_{\beta(a_j)}$: the number of all transition road candidates.

Analytical result shows that road asset quantity and internal segment quantity are both major factors contributing to model complexity. The usage of excessive binary variables and nonlinear constraints requires a significant amount of branching process and relaxation solving, which has become the major computational burden. Notably, road asset quantity, internal segment quantity, and constraint type are all determined by problem construction and formulation, thereby fixed for a given problem. Therefore, we will work on problem decomposition to address the computational difficulty.
TABLE 1 Model Complexity Analysis

| Source | Constraints / Binaries | Quantity | Complexity | Type   |
|--------|------------------------|----------|------------|--------|
| SACS   | Dist Cons. [1][3]       | $\sum 2n_i - 3$ | $O(N)$     | Quadratic |
|        | Orient Cons. (5)        | $\sum n_i - 2$  | $O(N)$     | Bilinear |
| CACS   | Intersection Ineq. 11, 12c | $2\left( C_{N\ell}^2 - \sum a_i C_{N\ell_i}^2 \right)$ | $O(N_{\ell}^2)$ | Polynomial |
|        | Auxiliary Bin. 11       | $2\left( C_{N\ell}^2 - \sum a_i C_{N\ell_i}^2 \right)$ | $O(N_{\ell}^2)$ | Element-wise |
| Connect.| Intersection Ineq. 13e  | $2X(N_\ell - 2)$ | $O(N_{\ell}^2 N_\ell)$ | Polynomial |
|        | Auxiliary Bin. 13c, 13e | $C_{N_A}^2 + X + 2X(N_\ell - 2)$ | $O(N_{\ell}^2 N_\ell)$ | Element-wise |

CONCLUSION

In this study, we proposed the first procedural scenario-based optimization approach for CAV proving ground design, and refer to it as an “Xcity” design problem. The notion of traffic encounters is employed to define road assets as the fundamental construction elements of a CAV testbed. This infrastructure enables the use of scenario-based capability evaluation and direct-connectivity-based flexibility evaluation for CAV proving grounds. Then the Xcity design problem is formulated as general nonlinear optimization problems to fulfill the demanding fidelity requirement of CAV testing and validation tasks. We also presented a two-phase optimization model and demonstrated the design procedures using limited number of road assets. Our numerical results showed the effectiveness of the formulation to guide Xcity constructions strategically. Due to the exponential computational complexity of the resulting formulation, the use of decomposition methods signifies our future direction to extend the tractability of the proposed formulation.

AUTHOR CONTRIBUTION STATEMENT

The authors confirm contribution to the paper as follows: study conception and design: Rui Chen, Mansur Arief, Ding Zhao; data collection: Rui Chen; analysis and interpretation of results: Rui Chen, Mansur Arief, Ding Zhao; draft manuscript preparation: Rui Chen, Mansur Arief, Ding Zhao. All authors reviewed the results and approved the final version of the manuscript.
REFERENCES

[1] I. Passchier, G. v. Vugt, and M. Tideman. “An Integral Approach to Autonomous and Co-operative Vehicles Development and Testing”. 2015 IEEE 18th International Conference on Intelligent Transportation Systems. Sept. 2015, pp. 348–352. doi: 10.1109/ITSC.2015.66.

[2] Z. Szalay. “Structure and Architecture Problems of Autonomous Road Vehicle Testing and Validation”. Structure and Architecture Problems of Autonomous Road Vehicle Testing and Validation. An optional note. 15TH Mini Conference on Vehicle System Dynamics. The publisher, Nov. 2016.

[3] Mcity. Mcity Test Facility. 2018. url: https://mcity.umich.edu/our-work/mcity-test-facility/.

[4] T. STEVENS. Crashing Castle: An autonomous ride in Waymo’s playground. 2017. url: https://www.cnet.com/roadshow/news/google-waymo-castle-visit/.

[5] D. Muoio. Uber built a fake city in Pittsburgh with roaming mannequins to test its self-driving cars. 2018. url: https://www.businessinsider.com/ubers-fake-city-pittsburgh-self-driving-cars-2017-10.

[6] TRC. Transportation Research Center Inc. (TRC) announces all-new Smart Mobility Advanced Research and Test Center. 2017. url: http://www.trcpg.com/blog/2017/01/27/transportation-research-center-inc-trc-announces-new-smart-mobility-advanced-research-test-center-45-million-new-infrastructure-expansion-test-connected-driverless-vehicles/.

[7] HARMAN. HARMAN to Co-Develop and Operate the International Cyber Security Smart Mobility Analysis and Research Test (SMART) Range in Israel. 2017. url: https://news.harman.com/releases/releases-20171114.

[8] ACM. THE PROJECT. 2018. url: http://www.acmwillowrun.org/the-project/.

[9] P. Koopman and M. Wagner. “Challenges in Autonomous Vehicle Testing and Validation”. SAE International Journal of Transportation Safety 4.1 (2016), pp. 15–24. issn: 23275626, 23275634. url: http://www.jstor.org/stable/26167741.

[10] W. Wachenfeld, H. Winner, J. C. Gerdes, B. Lenz, M. Maurer, S. Beiker, E. Fraedrich, and T. Winkle. “Use Cases for Autonomous Driving”. Autonomous Driving: Technical, Legal and Social Aspects. Ed. by M. Maurer, J. C. Gerdes, B. Lenz, and H. Winner. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 9–37. isbn: 978-3-662-48847-8. doi: 10.1007/978-3-662-48847-8_2. url: https://doi.org/10.1007/978-3-662-48847-8_2.

[11] C. Nowakowski, S. E. Shladover, C.-Y. Chan, and H.-S. Tan. “Development of California Regulations to Govern Testing and Operation of Automated Driving Systems”. Transportation Research Record: Journal of the Transportation Research Board 2489 (2015), pp. 137–144. doi: 10.3141/2489-16. eprint: https://doi.org/10.3141/2489-16. url: https://doi.org/10.3141/2489-16.

[12] NHTSA. Automated Driving Systems 2.0. 2018. url: https://www.nhtsa.gov/sites/nhtsa.dot.gov/files/documents/13069a-ads2.0_090617_v9a_tag.pdf.
[13] TORC. Asimov self-driving car system capabilities. 2018. url: https://torc.ai/wp-content/uploads/Torc_capabilities_checklist.pdf

[14] W. Wang and D. Zhao. “Extracting Traffic Primitives Directly From Naturalistically Logged Data for Self-Driving Applications”. IEEE Robotics and Automation Letters 3.2 (Apr. 2018), pp. 1223–1229. doi: 10.1109/LRA.2018.2794604.

[15] W. Wang, A. Ramesh, and D. Zhao. “Clustering of Driving Scenarios Using Connected Vehicle Datasets” (2018). arXiv: 1807.08415 url: https://arxiv.org/abs/1807.08415.

[16] S. Li, W. Wang, Z. Mo, and D. Zhao. “Clustering of Naturalistic Driving Encounters Using Unsupervised Learning”. CoRR abs/1802.10214 (2018). arXiv: 1802.10214 url: http://arxiv.org/abs/1802.10214.

[17] P. Bender, J. Ziegler, and C. Stiller. “Lanelets: Efficient map representation for autonomous driving”. 2014 IEEE Intelligent Vehicles Symposium Proceedings. June 2014, pp. 420–425. doi: 10.1109/IVS.2014.6856487.

[18] OpenDRIVE contributors. OpenDRIVE. [Online; accessed 28-July-2018]. 2018. url: http://www.opendrive.org/index.html.

[19] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org. https://www.openstreetmap.org. 2018.

[20] Wikipedia contributors. OpenStreetMap — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=OpenStreetMap&oldid=852156690. [Online; accessed 28-July-2018]. 2018.

[21] J. R. Shewchuk. “Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates”. Discrete & Computational Geometry 18.3 (Oct. 1997), pp. 305–363.

[22] E. L. Lawler and D. E. Wood. “Branch-and-bound methods: A survey”. Operations research 14.4 (1966), pp. 699–719.

[23] J. Lofberg. “YALMIP: A toolbox for modeling and optimization in MATLAB”. Computer Aided Control Systems Design, 2004 IEEE International Symposium on. IEEE. 2004, pp. 284–289.

[24] A. Gleixner, M. Bastubbe, L. Eifler, T. Gally, G. Gamrath, R. L. Gottwald, G. Hendel, C. Hojny, T. Koch, M. E. Lübbecke, S. J. Maher, M. Miltenberger, B. Müller, M. E. Pfetsch, C. Puchert, D. Rehfeldt, F. Schlösser, C. Schubert, F. Serrano, Y. Shinano, J. M. Viernickel, M. Walter, F. Wegscheider, J. T. Witt, and J. Witzig. The SCIP Optimization Suite 6.0. Technical Report. Optimization Online, July 2018. url: http://www.optimization-online.org/DB_HTML/2018/07/6692.html.

[25] M. Tawarmalani and N. V. Sahinidis. “A polyhedral branch-and-cut approach to global optimization”. Mathematical Programming 103 (2 2005), pp. 225–249.
[26] A. Gleixner, M. Bastubbe, L. Eifler, T. Gally, G. Gamrath, R. L. Gottwald, G. Hendel, C. Hojny, T. Koch, M. E. Lübbecke, S. J. Maher, M. Miltenberger, B. Müller, M. E. Pfetsch, C. Puchert, D. Rehfeldt, F. Schlösser, C. Schubert, F. Serrano, Y. Shinano, J. M. Viernickel, M. Walter, F. Wegscheider, J. T. Witt, and J. Witzig. *The SCIP Optimization Suite 6.0*. ZIB-Report 18-26. Zuse Institute Berlin, July 2018. url: [http://nbn-resolving.de/urn:nbn:de:0297-zib-69361](http://nbn-resolving.de/urn:nbn:de:0297-zib-69361).

[27] A. M. Geoffrion. “Generalized benders decomposition”. *Journal of optimization theory and applications* 10.4 (1972), pp. 237–260.

[28] N. V. Sahinidis. *BARON 17.8.9: Global Optimization of Mixed-Integer Nonlinear Programs*, User’s Manual. 2017.

[29] M. R. Kılınç and N. V. Sahinidis. “Exploiting integrality in the global optimization of mixed-integer nonlinear programming problems with BARON”. *Optimization Methods and Software* 33.3 (2018), pp. 540–562.

[30] Ioannis Filippidis. *OpenStreetMap Functions*, [https://www.mathworks.com/matlabcentral/fileexchange/35819-openstreetmap-functions](https://www.mathworks.com/matlabcentral/fileexchange/35819-openstreetmap-functions) [Online; accessed 28-July-2018]. 2018.