SmartParcels: constructing smart communities through human-in-the-loop urban IoT planning

Tung-Chun Chang, Chih-Chun Wu, Georgios Bouloukakis, Cheng-Hsin Hsu, Nalini Venkatasubramanian

SmartParcels: Constructing Smart Communities Through Human-in-the-Loop Urban IoT Planning

TUNG-CHUN CHANG, University of California, Irvine, USA
CHIH-CHUN WU, National Tsing Hua University, Taiwan
GEORGIOS BOULOUKAKIS, Télécom SudParis, IP Paris, France
CHENG-HSIN HSU, National Tsing Hua University, Taiwan
NALINI VENKATASUBRAMANIAN, University of California, Irvine, USA

The growth of smart communities has expanded the use of IoT as a critical element in the urban planning toolkit by city officials whose goal is to improve the quality of life for citizens. Smart applications such as air pollution monitoring, intelligent transportation, and smart buildings pose diverse information needs from the underlying sensing, communication, and computation infrastructure. In this article, we propose SmartParcels, a framework that exploits the service needs of communities to generate a comprehensive and cost-effective plan for instrumenting designated regions (often called parcels). SmartParcels embeds a cross-layer approach incorporating information/data, infrastructure, and geospatial layout as interdependent layers. We explore a suite of algorithms (optimal, partial optimal, heuristic) that can be composed in a plug-and-play manner to achieve performance-cost tradeoffs. SmartParcels can be utilized for clean-slate planning (from scratch) or retrofitting communities with existing smart infrastructure. SmartParcels allows planners to explore the possible impact of unexpected events to improve infrastructure resilience with what-if analyzers. Two real-world settings at Hsinchu, Taiwan, and Irvine, California are leveraged in the evaluation, which reveals that SmartParcels enables a 2X - 7X improvement in cost/performance metrics compared to baseline algorithms and achieves a 57% higher communication throughput while dealing with unexpected events.

CCS Concepts: • **Software and its engineering → Layered systems**; **Distributed systems organizing principles**; • **Computer systems organization → Sensors and actuators**; **Dependable and fault-tolerant systems and networks**; • **Information systems → Decision support systems**.

Additional Key Words and Phrases: IoT planning, urban planning, placement problem, what-if analysis, human-in-the-loop

ACM Reference Format:
Tung-Chun Chang, Chih-Chun Wu, Georgios Bouloukakis, Cheng-Hsin Hsu, and Nalini Venkatasubramanian. 2025. SmartParcels: Constructing Smart Communities Through Human-in-the-Loop Urban IoT Planning. 1, 1 (February 2025), 30 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Smart Communities today incorporate a variety of consumer-grade devices through the Internet-of-Things (IoT) ecosystem - applications built using these devices can dramatically improve quality of life for residents. For example, smart surveillance technologies help monitor (and capture) crime; similarly, real-time environmental monitoring can help pinpoint the location of toxic emissions and ensure safety of citizens. Prompt notifications from these smart applications to stakeholders can

Authors' addresses: Tung-Chun Chang, University of California, Irvine, USA; Chih-Chun Wu, National Tsing Hua University, Taiwan; Georgios Bouloukakis, Télécom SudParis, IP Paris, France; Cheng-Hsin Hsu, National Tsing Hua University, Taiwan; Nalini Venkatasubramanian, University of California, Irvine, USA.

ACM XXXX-XXXX/2025/2-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

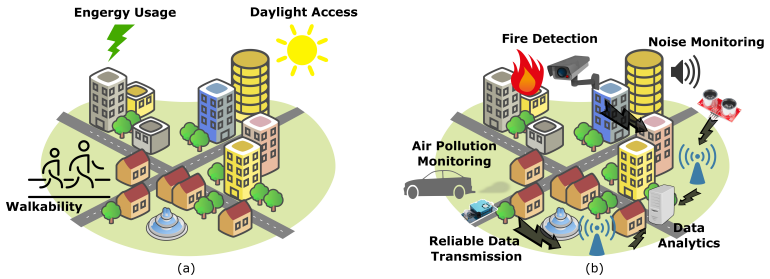


Fig. 1. Different aspects considered in urban planning tools: (a) traditional and (b) IoT-enabled.

enable timely interventions. Market reports indicate that investments in smart city technologies were valued at $1226.9 billion globally in 2022 with an expected annual growth rate of 25.8% between 2023 and 2030 [23]. Incorporating IoT infrastructure to create smart communities complicates the already difficult urban planning problem - it requires effective deployment of heterogeneous sensing, communication, and computation technologies that are interconnected. Traditional urban planning tools, such as umi [46] and CitySim [50] must be extended to integrate IoT infrastructure as a prime element to create comprehensive and cost-effective plans for urban infrastructures that are robust and sustainable, as shown in Fig. 1(a).

The diverse needs of communities can be realized by multiple applications with data from heterogeneous devices, as shown in Fig. 1(b) - a custom IoT deployment for each application can turn out to be prohibitively expensive. Fortunately, data obtained from a single device can often be used to accomplish tasks for multiple applications. Cost-effective design must take such data reuse from devices into account since local/city governments must work with limited budgets and resources to create new infrastructure or renovate existing deployments that can operate for several years. Embedding acoustic sensors in communities enables multiple applications, such as ambient noise monitoring, gunshot detection, and traffic accident detection [5]. Note, however that data quality obtained by a sensing infrastructure can also vary. Consider the case of wildfire-prone communities – here early detection of emerging wildfires can be accomplished using expensive multi-spectral cameras that have wide (15-60 km) sensing ranges [2, 6] or using inexpensive gas sensors with limited sensing coverage [48]. Each application requires a certain level of computation for corresponding analytics; hence, computing devices with different capabilities are required. Communication technologies designed for various purposes–low-power protocols, such as LoRa and Bluetooth, are usually limited to transmitting lightweight data, whereas high-bandwidth technologies, like WiFi and cellular networks, support more voluminous multimedia data, such as surveillance videos. An optimized deployment that can support multiple applications for communities must take into account the above opportunities and tradeoffs.

Ensuring the underlying IoT infrastructure's smooth and undisruptive operation is critical, especially for time-sensitive applications. Unexpected events often cause damage to the deployed devices and/or alter their network traffic pattern. Instances like an earthquake can potentially break sensors and network access points; these instances are usually followed by heavy network data exchange that disseminates the latest information. However, such events are unique to different geographical locations, and their occurrence is unpredictable. It requires help from subject matter experts to investigate possible events and potential damages. Then, a plan could be made to reduce potential negative impacts on communities accordingly. We argue that it must involve multiple entities (usually with people) to build resilient IoT infrastructure. An innovative planning tool must streamline their domain knowledge into the design process.

In this article, we study the urban IoT planning problem to help urban planners make critical decisions while designing smart communities. The main trade-offs and challenges are: *what/where* infrastructures (sensing, networking, and computing devices) should be instrumented, *how* the information units (data) flow via the infrastructures, *how* to design the infrastructures (e.g., reuse a sensor for various applications, or install an extra networking device at the location with a higher communication coverage) with a fixed budget, *how* the existing infrastructure can be reused to reduce the cost, and *how* knowledge from subject matter experts can be leveraged.

To the best of our knowledge, this work is among the very first urban planning frameworks with integrated IoT planning, and the related literature is summarized in Sec. 8. We propose to design tools to conduct what-if analyses and explore design space options at the community scale, which require a holistic view that works across multiple system layers with the assistance of domain experts. To this end, we design, implement, and evaluate a system called **SmartParcels**. The name is derived from urban planning literature where *Parcels* refer to a piece of *designated land* slated for development with a specific use and purpose [18]. Our current work, extended from Chang et al. [16], develops a human-in-the-loop tool to optimize and instrument smart community land parcels. SmartParcels turns traditional urban planning into IoT-enabled urban planning, a.k.a. urban IoT planning in smart communities. More specifically, the proposed SmartParcels consists of three main components: an IoT planner, a what-if analyzer, and a human-in-the-loop refinement procedure. The IoT planner generates a sophisticated plan to deploy applications in communities by jointly considering software and hardware resources. The what-if analyzer then provides methods to explore various types of network traffic and failures caused by unexpected events. Last, domain experts and urban planners can iteratively use the what-if analyzer to explore different scenarios and refine urban IoT deployment plans.

1.1 Contributions and Paper Organization

To develop the SmartParcels framework, this article makes the following contributions:

- We propose a cross-layer architecture for urban IoT planning that separates required applications, data transmission paths, physical devices, and deployment locations. Our two-step framework generates potential mappings and feasible deployment plans by examining relationships between layers and allocating applications to devices at designated deployment locations. Utilizing cost-utility functions for different deployment options, we formulate an optimization problem to address community service requests, which is proven to be NP-hard.
- We develop a suite of algorithms for the two-step process in SmartParcels. Initially, we create an optimal technique based on enumeration and dynamic programming, although its complexity limits its use in medium- to large-scale settings. To address this, we develop practical planning graph generation algorithms that take into account component reusability, service utility, and communication coverage. The suite of algorithms in SmartParcels provides high flexibility for urban planners by offering different levels of optimality: optimal, partially optimal, and heuristic. SmartParcels supports two scenarios: clean-slate (starting from scratch) and retrofit (integrating with existing infrastructure) for smart communities.
- We implement what-if analyzers that allow urban planners to explore the impact of unexpected events on IoT infrastructure by identifying its vulnerable parts, particularly focusing on the robustness of communication networks. Experts such as urban planners, subject matter experts, and emergency responders can be involved in the design process to develop strategies and remedies that maintain a desired level of robustness. To address requests with varying urgency, we provide two what-if analyzers: a lightweight queuing model for coarse-grained analysis and a simulator-based model for detailed, fine-grained analysis of the damage to IoT infrastructure.

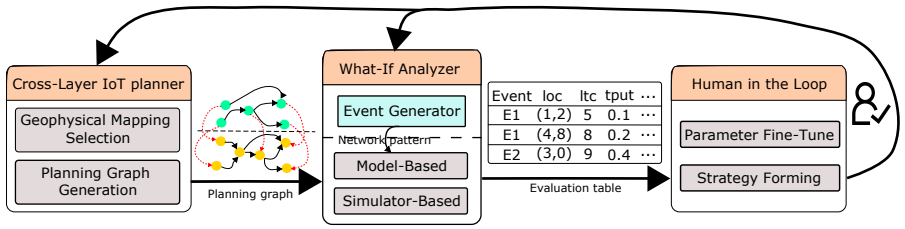


Fig. 2. The proposed SmartParcels framework.

We evaluate SmartParcels in two real-world community settings: National Tsing Hua University in Taiwan (featuring a smart streetlight application) and Irvine, California, USA (focusing on environmental sensing and community safety applications). The evaluation indicates that SmartParcels intelligently identifies opportunities for device reuse and outperforms the baseline algorithm in overall service utility by up to 2 times and enhances performance in retrofit scenarios by up to 7.64 times. Additionally, it enables urban planners to adeptly balance optimality with efficiency when computing resources are limited, achieving up to 57% higher network throughput by leveraging domain knowledge from human inputs for deployment adjustments.

The remainder of this article is structured as follows: Section 2 provides an overview of the proposed framework, SmartParcels. Section 3 discusses our cross-layer design and formulates the urban IoT planning problem. In Section 4, we design a suite of algorithms for the IoT planner. The what-if analyzer and human-in-the-loop mechanism are detailed in Section 5. Implementation details are covered in Section 6, and evaluation results are presented in Section 7. Finally, related works and conclusions are presented in Sections 8 and 9.

2 SMARTPARCELS – A HUMAN-IN-THE-LOOP APPROACH

We aim to exploit domain knowledge and requests from each stakeholder involved, such as government agencies, community members, urban designers, and subject matter experts, into the design loop of constructing robust smart communities. Community residents provide the applications required to improve their life quality; engineers compile possible ways of implementation and required software and hardware; government agencies and residents jointly provide potential locations to deploy devices; subject matter experts provide insight into the potential impact of unexpected events. We argue that a comprehensive urban IoT plan must address the various needs of community residents and the concerns of authorities and domain experts about infrastructure resilience. To this end, we design an urban planning tool, SmartParcels, that assists in embedding IoT into community infrastructure. As an initial step, we design a suite of algorithms to generate an initial IoT deployment plan to fulfill the requests of community residents. Two common scenarios are supported: *clean-slate* for completely new communities and *retrofiting* existing infrastructures. Possible settings/configurations can be explored to generate different IoT deployments–the best one can be identified. Then, various assumptions of unexpected events can be carried out during the design phase and/or at requests with a what-if analyzer to explore the communication robustness of the infrastructure. The fragile parts are identified, and corresponding refinement suggestions are provided to accommodate possible network failures or delays by, for instance, installing redundant devices or upgrading networking devices. Users can decide whether to accept the suggestions or apply their own strategies, followed by another round of analysis. The process iterates until the users are satisfied with the analysis results. Fig. 2 shows the proposed workflow.

Executing a cross-layer IoT planner. First, *community profiling* is conducted to gather geophysical data, existing devices, hardware/software requirements for applications, and deployment/operational budgets. To implement the required applications within specific budgets, the IoT planner deploys hardware devices to geophysical locations to capture physical phenomena (sensing devices), transmit data (networking devices), and analyze data (computing devices). However, the relationship between components is complex, especially when implementing multiple applications in communities with limited budgets. Sharing data from common sensors across multiple applications reduces costs, making strategic deployment of sensors and network devices crucial. We adopt a *cross-layer design* to capture relationships across different layers: application, information (software and data), infrastructure (hardware), and geophysical. We then present an *IoT planning problem* and prove its NP-hardness. To reduce complexity, we divide the problem into two sub-problems: *geophysical mapping selection*, which identifies possible locations for devices, and *planning graph generation*, which creates a deployment plan. Finally, we propose a suite of algorithms that allow urban planners to adjust the execution time and optimality of urban IoT deployments.

Exploring robustness of the deployment to connectivity disruptions via a what-if analyzer. Communication is fundamental for IoT deployments in smart communities. SmartParcels offers a series of analysis tools that urban planners can use to assess the communication robustness of the infrastructure in the deployment plan. The analyzer focuses on the network impacts (traffic, failure) caused by *unexpected events*. Note that we explore unexpected events, as they have a higher probability of causing connectivity disruptions. However, SmartParcels can also analyze the impact of regular events, such as traffic jams during rush hours. Specifically, we explore connectivity failures at two levels: a *queuing theoretic model* for coarse-grained analysis of network properties like latency under changing traffic conditions, and a *fine-grained communication simulator* built on the NS3 simulator to generate detailed network behavior affecting latency and throughput. We note that these two models are *examples*; other analytics or simulators/emulators [10] can be used for more accurate analysis at the expense of higher complexity. We introduce a component that generates unexpected events and corresponding network conditions to enhance standard off-the-shelf analytical tools and simulators. Various scenarios, such as unpredictable traffic and different levels of communication failure, are inspected. For example, people flocking to a shelter (e.g., a school) after an earthquake could alter an application's traffic pattern (e.g., pedestrian counting) and increase traffic to public WiFi APs, potentially saturating and disrupting the network. During an aftershock, critical data transmission may be delayed, and the earthquake could even destroy parts of the infrastructure. To manage these dynamics, the what-if analyzer generates an *evaluation table* recording *fragile locations*, network performance metrics such as latency and throughput, and *suggested reactions* to events along with modifications to the deployed infrastructure.

Carrying out human-in-the-loop refinements. SmartParcels allows human-in-the-loop interventions in two different stages. First, users can explore various settings/configurations of instrumenting smart communities, such as budgets and locations to deploy devices. The initial result can be used as a starting point to refine the plan. It can also be a reference for users to raise investment and obtain permission to install devices at certain locations. Secondly, SmartParcels' what-if analyzer allows users to explore the impact of unexpected events on the IoT deployment. It identifies a set of fragile locations and suggests corresponding reactions. Urban planners can adopt one of the suggestions or design their own strategies. The deployment plan is then modified/refined accordingly. After a modification is determined, urban planners analyze the modified deployment plans with the what-if analyzer again. The process iterates until urban planners are satisfied with the deployment plan or no fragile locations are identified.

Balancing Design Constraints Across Multiple Budgets. Different entities have different budgets for designing smart communities. In particular, computation resources are the most

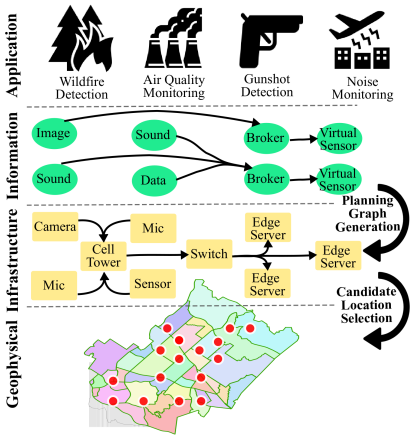


Fig. 3. Overview of the IoT planning problem considered in SmartParcels.

important when urban planning tools come into the picture. SmartParcels provides a control nob that trade-offs budget limits and optimality. Entities with limited computation capabilities can use it to derive a sub-optimal deployment plan. A lightweight model-based what-if analyzer is also provided. Quick analyses of unexpected events can be carried out for a broad test of various setups. Then, the simulation-based analyzer can be leveraged to generate a more sophisticated analysis of the most promising ones. Such workflow helps reduce design time and computation costs.

3 IOT PLANNER AND ITS CROSS-LAYER DESIGN

Fig. 3 gives an overview of the considered cross-layer IoT planning problem in SmartParcels. Each *application* can be realized by multiple *information flows*, which consist of multiple software processing units. Each information flow needs to be overlaid on an *infrastructure flow*, which dictates the hardware devices to install. Last, individual hardware devices, chosen by one or more infrastructure flows, need to be placed at the selected candidate locations. The goal of our cross-layer IoT planning problem is to compute the optimal mappings across the layers, so as to maximize the service utility without incurring excessive costs. Our proposed algorithms return the types and new IoT devices, edge servers, and network switches to urban planners.

To rigorously solve the IoT planning problem in SmartParcels, we formulate the problem as an optimization problem to maximize the overall *service utility* of the required applications after the IoT devices, edge servers, and network switches in the computed plan are deployed. The service utility is defined as a function of *coverage*, which represents the geographical area where events can be detected and *accuracy*, which represents the probability an event can be correctly detected. The IoT plans must satisfy several constraints, including the deployment and operational budgets, sensing ranges, computing power, network bandwidth, and application QoS requirements. Solving the resulting IoT planning problem is not an easy task because of the complex interplay among the four layers: application, information, infrastructure, and geophysical as illustrated in Fig. 3. While the *mappings* across the layers offer tremendous flexibility for higher *reusability* of existing resources and higher *efficiency* of the smart city, the mappings result in an extremely large *search space* for the problem formulation formulated in this section.

3.1 System Models

Community profiles. A smart city is considered to consist of a set of communities. Each community provides a set of service sites $\mathcal{S}$, where $s_i \in \mathcal{S}$ is the i-th service site. For example, a plaza (community) has a set of service sites, including a grocery store, a restaurant, and a bank. s_i demands for a set of applications $\mathcal{A}_i$, where $a_{i,j} \in \mathcal{A}_i$ is the j-th application. For instance, a bank demands gunshot detection for security, and a restaurant demands air quality for outdoor dining. Each community has a tuple $(\mathcal{S}, \{\mathcal{A}_i | \forall s_i \in \mathcal{S}\})$ that indicates its service sites and applications. For simplicity, a service site is represented by the center point of its boundary. The communities provide a set of candidate locations $\mathcal{L}$, like traffic poles, street lights, and apartment rooftops, for installing devices, including IoT devices with sensors, edge servers, and network devices. Note that different applications could have different importance levels. For each service site s_i of a community, we assign a weight $\beta_{i,j}$ to each required application $a_{i,j}$, where $\sum_{\forall a_{i,j} \in \mathcal{A}_i} \beta_{i,j} = 1, \forall s_i$.

Information flows for an application. To implement $a_{i,j}$, a set of information flows $\mathcal{F}^{ifo}$ can be adopted, where $f^{ifo} \in \mathcal{F}^{ifo}$ is an information flow. More precisely, $f^{ifo} = (V^{ifo}, E^{ifo})$ is a directed weighted graph, where $v \in V^{ifo}$ represents an information unit (which can be *raw data or middleware components*), and $e(u,v) \in E^{ifo}$ represents the data flow between two information units. Both the vertices and edges are associated with weights. The weight of a vertex $w(v)$ represents the computing resources consumed by the information unit, whereas the weight of an edge $w(e(u,v))$ is the bandwidth consumption over the edge. Besides, each information flow specifies the number of sensors, e.g., three microphones are needed for gunshot detection for triangular localization.

Infrastructure flows implementing an information flow. Each information flow f^{ifo} can be implemented by a set of infrastructure flows $\mathcal{F}^{ifr}$, where $f_k^{ifr} \in \mathcal{F}^{ifr}$ is the k-th infrastructure flow. $f_k^{ifr} = (V^{ifr}, E^{ifr})$ is a directed weighted graph, where $v \in V^{ifr}$ represents a device and $e(u,v) \in E^{ifr}$ represents the data flow between two devices. We consider very general devices, which could be sensors, like microphones or cameras, computing devices, like edge servers, and network switches, like 5G small cells or Ethernet switches. The weights of a vertex $w(v)$ and an edge $w(e(u,v))$ represent the computing resource and the network bandwidth offered by them, respectively. A tuple $(\mathcal{F}^{ifo}, \{\mathcal{F}^{ifr} | \forall f^{ifo} \in \mathcal{F}^{ifo}\})$ summarizes all information and infrastructure flows for each application $a_{i,j}$. Given an f^{ifo} and an f_k^{ifr}, each processing unit $v \in V^{ifo}$ is assigned to a device $v' \in V^{ifr}$ by a function $R(v) = v'$. Besides, for an edge $e(u,v) \in E^{ifo}$, let $\langle R(u), R(v) \rangle$ denote the shortest path on f_k^{ifr} consisting of involved devices, i.e., the actual data flow on the infrastructure layer. We assume that $\langle R(u), R(v) \rangle$ contains at least one network switch unless $R(u)$ and $R(v)$ are the same device. If u and v run on the same device, they are well-connected, and we write $w(e(R(u), R(v))) = \infty$.

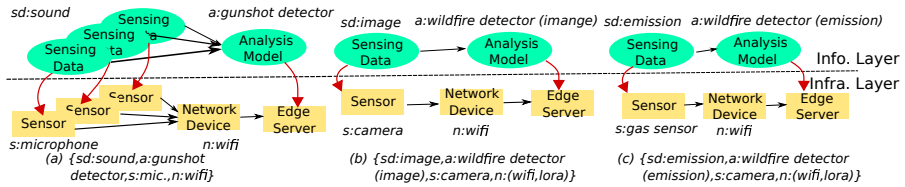


Fig. 4. Sample info/infra flows used in our evaluations: (a) gunshot detection and (b), (c) wildfire detection.

Examples: Fig. 4 shows three sample flows. We use a colon to indicate the corresponding infrastructure/information unit for a flow, e.g., *{sensing data:sound}*. Moreover, we use parentheses

to indicate multinetwork choices, e.g., *{network device:(wifi,lora)}*. In addition, we use *sd* for sensing data, *a* for analysis model, *s* for sensors, and *n* for network devices.

Planning graph. We define a planning graph as a two-layer graph $G^p = (V^p, E^p)$, where the first layer $G_1^p = (V_1^p, E_1^p)$ contains a set of information flows, and the second layer $G_2^p = (V_2^p, E_2^p)$ consists of a set of infrastructure flows. In both layers, flows may share vertices or edges. Besides, a set of *assignment edges* E^r has each edge $e(v, R(v)) \in E^r$ indicates the assignment of $v \in V^{ifo} \subset V_1^p$ to $R(v) \in V^{ifr} \subset V_2^p$ for f^{ifo} and f_k^{ifr}. We collectively write the planning graph as $V^p = \{V_1^p, V_2^p\}$ and $E^p = \{E_1^p, E_2^p, E^r\}$.

Geophysical mapping function of infrastructure. To select the candidate locations, we define a geophysical mapping function $f(v)$ that maps a vertex $v \in V_2^p$ of the infrastructure layer in a planning graph G^p to a candidate location $l \in \mathcal{L}$. Each $v \in V_2^p$ is captured by a tuple $t_v = (r_v^{tr}, r_v^{sen}, \tau_v)$. The device's transmission and sensing ranges are represented by r_v^{tr} and r_v^{sen}, respectively. τ_v indicates the device type, which could be *sensing*, *computing*, or *networking*. If $\tau_v \neq networking$, r_v^{tr} equals the transmission range of its connected network device u, i.e., $r_v^{tr} = r_u^{tr}$, $e(v, u) \in E_2^p$. Besides, multiple devices can be mapped to the same candidate location. For ease of presentation, we let V_k^{sen} denote the sensors of f_k^{ifr}, i.e., $\forall v \in V_k^{sen}$, $\tau_v = sensing$.

Service utility of an infrastructure flow. We write the Euclidean distance between two candidate locations $l_1, l_2 \in \mathcal{L}$ as $dist(l_1, l_2)$. An infrastructure flow is *connected* if all its devices are connected after mapping, i.e., $dist(f(u), f(v)) \leq \min(r_u^{tr}, r_v^{tr})$, $\forall e = (u, v) \in f_k^{ifr}$; otherwise, the flow is not connected. If f_k^{ifr} is connected, its *service utility* to a service site s_i is:

$$U(f_k^{ifr}, s_i) = A(f_k^{ifr}) \times P(V_k^{sen}, s_i), \tag{1}$$

where $A(f_k^{ifr})$ and $P(V_k^{sen}, s_i)$ are the *accuracy* and *sensing probability*. If f_k^{ifr} is unconnected, we set $U(f_k^{ifr}, s_i)$ to 0. Each f_k^{ifr} implementing application $a_{i,j}$ comes with an accuracy model $A(f_k^{ifr})$ depending on the implemented method. For example, for wildfire detection, image-based detection has higher accuracy than emission-based detection.

Our sensing probability models are inspired by the truncated attenuated model [51]. For a sensor $v \in V_k^{sen}$, the probability is attenuated (decaying) with its distance to s_i, $dist(f(v), s_i)$, and is truncated by its sensing range, r_v^{sen}. Hence, if $dist(f(v), s_i) \leq r_v^{sen}$, $\forall v \in V_k^{sen}$, the mean truncated attenuated sensing probability is $\bar{p} = \sum_{\forall v \in V_k^{sen}} e^{-\alpha_v f(v) s_i} / |V_k^{sen}|$, where α_v is a parameter related to v; otherwise, $\bar{p} = 0$. The sensing probability is bounded by the sensing range of sensors as follows:

$$P(V_k^{sen}, s_i) = \begin{cases} \bar{p}, \ if \ dist(f(v), s_i) \leq r_v^{sen}, \quad \forall v \in V_k^{sen}; \\ 0, \ otherwise. \end{cases} \tag{2}$$

This completes our definition of service utility. We emphasize that our proposed algorithms do not rely on the mathematical properties of the service utility. Hence, administrators have total freedom to apply different models, e.g., those with non-line-of-sight sensing ranges.

Costs. Each device $v \in V_2^p$ in the infrastructure layer incurs two types of cost: (i) *deployment cost* $\delta_{deploy}(v, l)$ due to deploying v at candidate location $l \in \mathcal{L}$ and (ii) *operational cost* $\delta_{op}(v)$ due to maintaining its daily operations. The deployment cost is one-time, while the operational cost is recurring. Besides, B_{dp} and B_{op} are the budgets for deploying and operating all the devices.

3.2 Problem Formulation

Given the above-defined notations and models, our problem is to generate the optimal planning graph G^{p^*} and select the optimal geophysical mapping functions $\mathcal{F}^* = \{f^*(v) | \forall v \in V_2^{p^*}\}$. More

specifically, our IoT planning problem is formulated as follows:

$$\max \sum_{\forall s_i \in \mathcal{S}} \sum_{\forall f_k^{ifr^*} \in G_2^{p^*}} \beta_{i,j} U(f_k^{ifr^*}, s_i) \tag{3a}$$

$$\text{subject to: } \sum_{\forall v \in V_2^{p^*}} \delta_{deploy}(v, f^*(v)) \leq B_{dp}; \tag{3b}$$

$$\sum_{\forall v \in V_2^{p^*}} \delta_{op}(v) \leq B_{op}; \tag{3c}$$

$$\sum_{\forall e(u,v) \in E^{r^*}} w(u) \leq w(v), \forall v \in V_2^{p^*}; \tag{3d}$$

$$\sum_{\forall e(u,v) \in E_2^{p^*}} w(e(u,v)) \leq \sum_{\forall e(v,u') \in E_2^{p^*}} w(e(v,u')), \ \forall v \in V_2^{p^*}; \tag{3e}$$

$$w(e(u,v)) \leq \min_{\forall e' \in \langle R(u), R(v) \rangle} w(e'), \forall e(u,v) \in E_1^{p^*}. \tag{3f}$$

The objective function in Eq. (3a) finds G^{p^*} and $\mathcal{F}^*$ to maximize the total service utility. Eqs. (3b) and (3c) are the budget constraints of the deployment cost (B_{dp}) and operational cost (B_{op}). Note that the deployment budget covers one-time device deployment. The cost of device replacement and maintenance due to depreciation can be amortized into the operational cost. For each infrastructure $v \in V_2^{p^*}$, Eq. (3d) ensures that v has enough computing resources to process all assigned information unit u, i.e., v's weight $w(v)$ is no less than the sum of all u's weights. Eq. (3e) checks that v's output bandwidth is no less than its input bandwidth. For each data flow $e(u,v) \in E_1^{p^*}$, Eq. (3f) ensures that the minimum bandwidth within the assigned path $\langle R(u), R(v) \rangle$ meets its bandwidth requirement.

Theorem 1. *The problem in Eq. (3) is NP-hard and can not be approximated within* $1 - 1/e$.

The proof is omitted; interested readers are referred to Chang et al. [16].

4 SMARTPARCELS' PLANNING ALGORITHMS

SmartParcels can efficiently navigate through the different layers to identify the necessary data flows and physical devices for each application by leveraging the models defined in Section 3. Consequently, we decompose the IoT planning problem into two subproblems. The first subproblem, *geophysical mapping selection*, identifies possible deployments for each device to implement applications required by the communities. The second subproblem, *planning graph generation*, generates a graph representing the final deployment by examining cost and utility trade-offs. SmartParcels aims to find the optimal planning graph $G^{p^*} = (V^{p^*}, E^{p^*})$, where $G_1^{p^*} = (V_1^{p^*}, E_1^{p^*})$ contains a set of information flows, and $G_2^{p^*} = (V_2^{p^*}, E_2^{p^*})$ contains a set of infrastructure flows. We design various algorithms for each subproblem, allowing optimal, sub-optimal, and heuristic solutions based on available computational resources and different requirements. Specifically, we leverage the reusability of devices to support more applications in our heuristic approach.

4.1 Multi-Phase Approach

Geophysical mapping selection. The goal is to select the optimal geophysical mapping functions $\mathcal{F}^*$ containing a mapping $f^*(v)$ for each infrastructure $v \in V_2^{p^*}$, i.e., $\mathcal{F}^* \leftarrow \{f^*(v) | \forall v \in V_2^{p^*}\}$. It takes the following inputs: (i) a tuple $(\mathcal{S}, \{\mathcal{A}_i | \forall s_i \in \mathcal{S}\})$ indicating the set of service sites and each service site's required applications for every community, (ii) a tuple $(\mathcal{F}^{ifo}, \{\mathcal{F}^{ifr} | f^{ifo} \in$

$\mathcal{F}^{ifo}\}$) representing the set of information flows and the infrastructure flows implementing each information flow for each application $a_{i,j} \in \mathcal{A}_i$, (iii) a set of candidate locations $\mathcal{L}$ for deploying the infrastructures, and (iv) a set of existing devices $\mathcal{D}$ in the community (for retrofit). Afterwards, a set of possible mappings of each infrastructure flow is selected. For each infrastructure flow $f_k^{ifr} = (V^{ifr}, E^{ifr}) \in \mathcal{F}^{ifr}$, a geophysical mapping function $f(v)$ for each infrastructure $v \in V^{ifr}$ is selected as follows: (i) for a pair of infrastructures u and v on edge $(u, v) \in E^{ifr}$, they must be within each other's transmission range r_u^{tr} and r_v^{tr} after mapping, i.e., $dist(f(u), f(v)) \leq \min(r_u^{tr}, r_v^{tr})$, and (ii) if v is a sensor, service site s_i must be within its sensing range r_v^{sen}, i.e., $dist(f(v), s_i) < r_v^{sen}$. We refer to an infrastructure flow f_k^{ifr} along with mapping $f(v)$ for infrastructure $v \in f_k^{ifr}$ as a Mapped Infrastructure Flow (MIF).

Planning graph generation. The goal is to compute the optimal planning graph G^{p^*} based on the outputs of geophysical mapping selection, i.e., various sets of MIFs for each application $a_{i,j} \in \mathcal{A}_i$. The set of all generated MIF sets $\mathcal{M}_k^{ifr}$ for $a_{i,j}$ is denoted by $\hat{\mathcal{M}} = \{\mathcal{M}_k^{ifr} | \forall f_k^{ifr} \in \mathcal{F}^{ifr}, \forall f^{ifo} \in \mathcal{F}^{ifo}\}$. Besides, the set of all required applications at all service sites of all communities is $\hat{\mathcal{A}} = \{a_{i,j} | \forall a_{i,j} \in \mathcal{A}_i, \forall s_i \in \mathcal{S}\}$. The optimal planning graph is generated by determining: (i) which application $a_{i,j} \in \hat{\mathcal{A}}$ to implement and (ii) which MIF $\hat{\mathcal{F}}$ (i.e., mapping) from which set $\mathcal{M}_k^{ifr} \in \hat{\mathcal{M}}$ (i.e., which infrastructure and information flows) to implement $a_{i,j}$.

In the rest of this section, we develop a suite of algorithms to solve the above two subproblems.

Intermediate planning graph. During planning graph generation, an intermediate planning graph $\hat{G}(K) = (\hat{V}(K), \hat{E}(K))$ is built with K MIFs, where $\hat{G}_{1(2)}(K) = (\hat{V}_{1(2)}(K), \hat{E}_{1(2)}(K))$ is the first (second) layer. For each MIF $\hat{\mathcal{F}}$, its infrastructure flow and corresponding information flow are included in $\hat{G}_2(K)$ and $\hat{G}_1(K)$, respectively. $\langle R(u_1), R(u_2) \rangle$ represents the actual data flow at the infrastructure layer for each edge $(u_1, u_2) \in \hat{E}_1(K)$, where information units u_1 and u_2 are assigned to infrastructure $R(u_1) \in \hat{V}_2(K)$ and $R(u_2) \in \hat{V}_2(K)$, respectively. $\hat{\delta}_{dp}(K)$ and $\hat{\delta}_{op}(K)$ denote the cumulative deployment and operational costs of $\hat{G}(K)$, respectively. Besides, $\mathcal{T}_K(l)$ represents the set of infrastructures mapped to a candidate location $l \in \mathcal{L}$ for $\hat{G}(K)$.

Fusing operations. To generate the planning graph, an extra MIF $\hat{\mathcal{F}}$ is *fused* into the intermediate planning graph $\hat{G}(K)$, which is denoted as $\hat{G}(K) \leftarrow \hat{G}(K-1) + \hat{\mathcal{F}}$. $\hat{\mathcal{F}} = \{f(u) | \forall u \in V^{ifr}\}$ is a mapping result of infrastructure flow $f_k^{ifr} = (V^{ifr}, E^{ifr}) \in \mathcal{F}^{ifr}$ implementing information flow $f^{ifo} = (V^{ifo}, E^{ifo}) \in \mathcal{F}^{ifo}$. Besides, $\hat{G}(K)$ and $\mathcal{T}_K(l)$ are set to $\hat{G}(K-1)$ and $\mathcal{T}_{K-1}(l)$ initially; similarly, $\hat{G}(K)$'s cumulative costs $\hat{\delta}_{dp}(K)$ and $\hat{\delta}_{op}(K)$ are initialized to $\hat{\delta}_{dp}(K-1)$ and $\hat{\delta}_{op}(K-1)$. The operation includes each infrastructure $v \in V^{ifr}$ and assigns v's corresponding information unit $u \in V^{ifo}$ to $\hat{G}(K)$ as follows.

(1) When a (previously included) infrastructure $v' \in \mathcal{T}_K(f(v))$ is identical to v in $\hat{G}(K)$, we have the following three cases.
   (a) u and v are *merged* into $\hat{G}(K)$, i.e., reuse the existing infrastructure and information unit if one of the following conditions holds: (i) v' has been assigned an information unit u', i.e., $R(u') \leftarrow v'$, and u' is identical to u, (ii) u and u' have the same inward information units, i.e., $\exists e(u'_{in}, u') \in \hat{E}_1(K)$, such that u'_{in} is identical to u_{in}, $\forall e(u_{in}, u) \in E^{ifo}$, and (iii) the identical inward information units are mapped to the same infrastructure, i.e., $R(u'_{in}) = R(u_{in})$.
   (b) v is *merged* into $\hat{G}(K)$ while u is *branched* out of $\hat{G}(K)$, i.e., reuse the existing infrastructure while adding an extra information unit to $\hat{G}_1(K)$ and an edge to v' if: (i) all information units assigned to v are distinct from u, i.e., u is distinct from u', $\forall u' \in \hat{V}_1(K)$ whose $R(u') = v$, and (ii) v has enough resources, i.e., Eqs. (3d)–(3f) hold. Besides, if u has

an inward information unit u', i.e., $\exists(u', u) \in f^{ifo}$, an edge between u' and u is added in $\hat{G}(K)$ after u' and u are included.

(c) u and v are not included if only condition (i) of case (b) holds. Then, $\hat{\mathcal{F}}$ is excluded from $\mathcal{M}_k^{ifr}$ since v has insufficient computing resources[1].

(2) When no infrastructure $v' \in \mathcal{T}_K(f(v))$ is identical to v, u and v are *branched* out of $\hat{G}(K)$, i.e., adding an extra information unit and infrastructure to $\hat{G}(K)$. Then, the connections to u and v in f^{ifo} and f_k^{ifr} are attached to the corresponding vertices in $\hat{G}(K)$. Finally, the costs of v are accumulated, i.e., $\hat{\delta}_{dp}(K) \leftarrow \hat{\delta}_{dp}(K) + \delta_{deploy}(v, f(v))$ and $\hat{\delta}_{op}(K) \leftarrow \hat{\delta}_{op}(K) + \delta_{op}(v)$. Then, $\mathcal{T}_K(f(v)) \leftarrow \mathcal{T}_K(f(v)) \cup v$.

4.2 Optimal Solution for IoT Planning

SmartParcels derives the optimal solution by adopting a composite solution including: (i) Enumeration (ENUM), which outputs every possible geophysical mapping of infrastructures for all possible infrastructure flows, and (ii) Dynamic Programming (DP), which outputs the planning graph by systematically examining all possible combinations of the generated mappings.

Algorithm 1: Enumeration

Input: $\mathcal{F}^{ifo}, \forall \mathcal{A}_i, \forall \mathcal{S}, \mathcal{F}^{ifr}, \mathcal{L}$

1 $\mathcal{M}_k^{ifr} \leftarrow \emptyset, \forall f_k^{ifr} \in \mathcal{F}^{ifr}$
2 **for** $f_k^{ifr} \in \mathcal{F}^{ifr}, f^{ifo} \in \mathcal{F}^{ifo}$ **do**
3 Check all $f(v) \in \mathcal{L}, \forall v \in f_k^{ifr}$, expand from sensors
4 **if** *(v == sensor at $f(v)$ and can cover s_i) or (v != sensor and $f(v)$ keeps the flow connected)* **then**
5 $\hat{\mathcal{F}} \leftarrow \hat{\mathcal{F}} \cup f(v)$
6 **if** $|\hat{\mathcal{F}}| = |V^{ifr}|, f_k^{ifr} = (V^{ifr}, E^{ifr})$ **then**
7 $\mathcal{M}_k^{ifr} \leftarrow \mathcal{M}_k^{ifr} \cup \hat{\mathcal{F}}$

Algorithm 2: Dynamic Programming

Input: $\mathcal{M}_k^{ifr}, \forall f_k^{ifr} \in \mathcal{F}^{ifr}$

1 $\hat{G}(0) \leftarrow \emptyset, K \leftarrow 0$
2 **while** $\hat{\mathcal{A}} \neq \emptyset$ *or all constraints are satisfied* **do**
3 Start dynamic programming
4 **for** $a_{i,j} \in \hat{\mathcal{A}}, \hat{\mathcal{F}} \in \mathcal{M}_k^{ifr}, \forall \mathcal{M}_k^{ifr} \in \hat{\mathcal{M}}$ **do**
5 $\hat{G}(K+1) \leftarrow \hat{G}(K) + \hat{\mathcal{F}}$
6 Record service utility gain of $\hat{G}(K+1)$
7 $\hat{\mathcal{A}} \leftarrow \hat{\mathcal{A}} \setminus a_{i,j}, K \leftarrow K+1$
8 Proceed to next round based on $\hat{G}(K+1)$
9 Return $\hat{G}(K)$ with the maximum service utility

Geophysical mapping via enumeration (ENUM). To derive the optimal solution, ENUM conducts an exhaustive search for all possible MIFs for each infrastructure flow of all required applications, which is shown in Algorithm 1. Specifically, for each application $a_{i,j} \in \mathcal{A}_i$ required at a service site $s_i \in \mathcal{S}$ by a community, ENUM examines each infrastructure flow $f_k^{ifr} = (V^{ifr}, E^{ifr}) \in \mathcal{F}^{ifr}$ that implements information flow $f^{ifo} \in \mathcal{F}^{ifo}$ for $a_{i,j}$ (lines 3–5) as follows.

(1) ENUM starts from selecting all possible geophysical mapping functions $f(v)$ for each sensor $v \in V_k^{sen} \subset V^{ifr}$ such that s_i is within v's sensing range r_v^{sen}, i.e., $dist(f(v), s_i) \leq r_v^{sen}$. That is, ENUM selects a combination of mappings for the sensors, e.g., C_4^{10} for mapping 4 sensors to 10 candidate locations, and each possible arrangement is denoted as $\hat{\mathcal{F}} = \{f(v) | \forall v \in V_k^{sen}\}$.

(2) For each $\hat{\mathcal{F}}$, ENUM further selects all possible mappings $f(u)$ for each infrastructure u adjacent to the sensors in f_k^{ifr}, i.e., $dist(f(v), f(u)) \leq \min(r_v^{tr}, r_u^{tr})$. Similarly, a combination of mappings is generated, and each possible arrangement is attached to $\hat{\mathcal{F}}$ separately as a new result, i.e., $\hat{\mathcal{F}} \leftarrow \hat{\mathcal{F}} \cup \{f(u) | \forall u, \exists(v, u) \in E^{ifr}\}$.

(3) ENUM recursively performs step (2) for the infrastructures adjacent to the previously examined ones until all infrastructures are examined.

[1]Here, we assume a candidate location hosts one device for each device type for simplicity.

Finally, ENUM outputs a set of all possible MIFs $\mathcal{M}_k^{ifr}$ for each f_k^{ifr}. Each MIF is represented by $\hat{\mathcal{F}}$ containing the selected mappings for all infrastructures, i.e., $|\hat{\mathcal{F}}| = |V^{ifr}|$ (lines 6–7). That is, $\forall f(v) \in \hat{\mathcal{F}}, \forall \hat{\mathcal{F}} \in \mathcal{M}_k^{ifr}$, if $v \in V_k^{sen}, dist(f(v), s_i) \leq r_v^{sen}$; otherwise, if $\exists (v,u) \in E^{ifr}, dist(f(v), f(u)) \leq \min(r_v^{tr}, r_u^{tr})$. Let $\mathcal{F}$ be the set of all possible infrastructure flows. ENUM is $O(|\mathcal{F}||\mathcal{L}|^{V_M})$, where V_M is the maximum number of vertices that an infrastructure flow could have since ENUM examines mappings to all possible candidate locations ($\mathcal{L}$) for each vertex of each infrastructure flow.

Generate planning graph via Dynamic Programming (DP). Algorithm 2 shows that, initially, the intermediate planning graph is empty, i.e., $\hat{G}(0)$. DP recursively builds the planning graph by fusing an extra MIF $\hat{\mathcal{F}}$ into the planning graph, i.e., $\hat{G}(K) \leftarrow \hat{G}(K-1) + \hat{\mathcal{F}}$. In other words, DP recursively implements an application $a_{i,j} \in \hat{\mathcal{A}}$ at service site $s_i \in \mathcal{S}$ required by a community by applying $\hat{\mathcal{F}}$ (lines 4–8). Particularly, DP examines each application $a_{i,j} \in \hat{\mathcal{A}}$ as follows.

(1) For each $a_{i,j}$, DP generates an intermediate planning graph $\hat{G}(1)$ for each MIF $\hat{\mathcal{F}} \in \mathcal{M}_k^{ifr}$, $\forall \mathcal{M}_k^{ifr} \in \hat{\mathcal{M}}$ by fusing $\hat{\mathcal{F}}$ into $\hat{G}(0)$, i.e., $\hat{G}(1) \leftarrow \hat{G}(0) + \hat{\mathcal{F}}$.
(2) For each derived $\hat{G}(1)$, DP calculates its *service utility gain* by summing all the service utility derived by fusing $\hat{\mathcal{F}}$. That is, let $\hat{s}$ denote the set of service sites which require the same application as $a_{i,j}$ and within the sensing range of sensors after mapping. The gain is the sum of Eq. (1) for each $s_i \in \hat{s}$. Then, DP records the service utility of each $\hat{G}(1)$.
(3) DP: (i) excludes $a_{i,j}$ from $\hat{\mathcal{A}}$, (ii) excludes the application required by $s_i \in \hat{s}$ from $\hat{\mathcal{A}}$, (iii) selects the next application from $\hat{\mathcal{A}}$, and (iv) performs the same process on top of each generated $\hat{G}(1)$ and then generates a set of $\hat{G}(2)$.
(4) DP recursively repeats step (3), i.e., $\hat{G}(K+1) \leftarrow \hat{G}(K) + \hat{\mathcal{F}}$, until all MIFs in $\mathcal{M}_k^{ifr}, \forall \mathcal{M}_k^{ifr} \in \hat{\mathcal{M}}$, $\forall a_{i,j} \in \hat{\mathcal{A}}$ cannot be fused, i.e., at least one Eqs. (3b)–(3f) is violated for all MIFs.

The planning graph with the maximum service utility is returned. We analyze the time complexity of DP as follows. Each recursion fuses one extra MIF (number of vertices bounded by V_M) into the planning graph. The recursive process stops after examining $V_M + 2V_M + \cdots + |\mathcal{M}|V_M$ vertices, where $\mathcal{M}$ contains all MIF derived in the previous phase. The process is repeated for at most $|\mathcal{M}|$ times. Hence, DP is $O(|\mathcal{M}|^3 V_M)$.

4.3 Clean-Slate Planning Heuristics

Since an optimal solution could take an extremely long time to compute, we propose two heuristic algorithms: Selection (SEL), which selects the (more promising) mappings with higher expected service utility and then with higher communication coverage to reduce the number of mappings and Maximum Reusability (MR), which iteratively selects the infrastructure flow with the maximum reusability of the instrumented devices.

Algorithm 3: Maximum Reusibility

Input: $\mathcal{M}_k^{ifr}, \forall f_k^{ifr} \in \mathcal{F}^{ifr}$
1 $\hat{G}(0) \leftarrow \emptyset, K \leftarrow 0$
2 **for** $a_{i,j} \in \hat{\mathcal{A}}, \hat{\mathcal{F}} \in \mathcal{M}_k^{ifr}, \forall \mathcal{M}_k^{ifr} \in \hat{\mathcal{M}}$ **do**
3 **if** $\mathcal{I}(\hat{\mathcal{F}}, \hat{G}(K))$ *is the maximum* **then**
4 $\hat{G}(K+1) \leftarrow \hat{G}(K) + \hat{\mathcal{F}}, \hat{\mathcal{A}} \leftarrow \hat{\mathcal{A}} \setminus a_{i,j}, K \leftarrow K+1$
5 Return $\hat{G}(K)$ when no more flows can be fused

Selection (SEL). We propose novel selection policies to prune out less-promising mappings at the runtime of ENUM. The policies are driven by the following intuitions: MIFs with higher utilities should be included earlier, and MIFs with higher communication coverage (how many candidate

locations are covered) should be included earlier. For each f_k^{ifr}, the utility can be estimated by Eq. (1) after mapping all the sensors (assuming the graph is connected); similarly, the communication coverage of a network device can be estimated after mapping. Specifically, let M and N denote user-specified pruning criteria for top M utility and top N communication coverage. The pruning policies are applied as follows.

(1) In step (1) of ENUM, after ENUM selects all possible mapping functions for sensors, only $\hat{\mathcal{F}}$ with the top M utilities are passed to step (2).
(2) In step (2) of ENUM, only the mappings with the top N communication coverages are included in $\hat{\mathcal{F}}$ if the examined infrastructure is a network device.

By doing so, SEL outputs a set of mappings with a smaller size but higher service utility and communication coverage as compared to ENUM. Let V_M^s be the maximum of sensors an infrastructure flow could have. SEL first examines the sensors of all possible infrastructure flows in $\mathcal{F}$; then, the partial flows with top M sensing coverage are further exaimed. Hence, SEL is $O(|\mathcal{F}|(|\mathcal{L}|^{V_M^s}) + M|\mathcal{L}|^{V_M - V_M^s})$. Selecting a small enough M could decrease its computation time significantly.

Maximum Reusability (MR). We propose a novel MR algorithm to generate the planning graph efficiently. The pseudo-code is provided in Algorithm 3. Instead of examining all possible combinations of MIFs, MR iteratively: (i) selects an application $a_{i,j} \in \hat{\mathcal{A}}$ to implement and (ii) fuses a MIF $\hat{\mathcal{F}} \in \mathcal{M}_k^{ifr}$, where $\mathcal{M}_k^{ifr} \in \hat{\mathcal{M}}$, into the planning graph $\hat{G}(K)$ according to $\hat{\mathcal{F}}$'s *reusability*. We define a reusability index, which considers both the *investment efficiency* and *communication coverage gain* when fusing $\hat{\mathcal{F}}$ as follows.

First, *investment efficiency* is the ratio of the *service utility gain* to the *cost gain* after fusing $\hat{\mathcal{F}}$ into $\hat{G}(K)$. That is, the more infrastructures that are reused, the lower the costs while fusing $\hat{\mathcal{F}}$. Specifically, let $\Delta U(\hat{\mathcal{F}}, \hat{G}(K))$ denote the service utility gain after fusing $\hat{\mathcal{F}}$ into $\hat{G}(K)$, which can be derived by the same procedure in step (2) of DP. After fusing $\hat{\mathcal{F}}$ into $\hat{G}(K)$, the cost gain $\Delta\delta(\hat{\mathcal{F}}, \hat{G}(K)) \leftarrow \hat{\delta}_{dp}(K) - \hat{\delta}_{dp}(K-1) + \hat{\delta}_{op}(K) - \hat{\delta}_{op}(K-1)$. Hence, $\mathcal{I}_{eff}(\hat{\mathcal{F}}, \hat{G}(K)) \leftarrow \frac{\Delta U(\hat{\mathcal{F}}, \hat{G}(K))}{\Delta\delta(\hat{\mathcal{F}}, \hat{G}(K))}$ is the investment efficiency of fusing $\hat{\mathcal{F}}$ into $\hat{G}(K)$.

Communication coverage gain is determined by the locations of network devices. If a network device has a higher communication coverage after deploying, fewer network devices are needed. Let $\mathcal{L}_{cov}(K) \subset \mathcal{L}$ denote the candidate locations within the communication coverage of the network devices in $\hat{G}(K)$. The communication coverage gain after fusing $\hat{\mathcal{F}}$ into $\hat{G}(K)$ is $\mathcal{I}_{cov}(\hat{\mathcal{F}}, \hat{G}(K)) \leftarrow \mathcal{L}_{cov}(K) - \mathcal{L}_{cov}(K-1)$.

Reusability index is defined as a weighted sum of investment efficiency and communication coverage gain when fusing a MIF $\hat{\mathcal{F}}$ into the intermediate planning graph $\hat{G}(K)$. Let α_{eff} and α_{cov} denote the weights of investment efficiency and communication coverage gain. The reusability index is written as $\mathcal{I}(\hat{\mathcal{F}}, \hat{G}(K)) \leftarrow \alpha_{eff}\mathcal{I}_{eff}(\hat{\mathcal{F}}, \hat{G}(K)) + \alpha_{cov}\mathcal{I}_{cov}(\hat{\mathcal{F}}, \hat{G}(K))$, where $\alpha_{eff} + \alpha_{cov} = 1$.

With the above notations, MR starts with an empty planning graph $\hat{G}(0)$. MR iteratively selects an application $a_{i,j} \in \hat{\mathcal{A}}$ to implement by fusing a MIF $\hat{\mathcal{F}} \in \mathcal{M}_k^{ifr}$, where $\mathcal{M}_k^{ifr} \in \hat{\mathcal{M}}$, into the current intermediate planning graph $\hat{G}(K)$ (lines 3–5) as follows.

(1) For each $a_{i,j} \in \hat{\mathcal{A}}$, MR examines the reusability index $\mathcal{I}(\hat{\mathcal{F}}, \hat{G}(K))$ of each MIF $\hat{\mathcal{F}} \in \mathcal{M}_k^{ifr}$, $\forall \mathcal{M}_k^{ifr} \in \hat{\mathcal{M}}$.
(2) MR fuses the MIF $\hat{\mathcal{F}}$ with the maximum $\mathcal{I}(\hat{\mathcal{F}}, \hat{G}(K))$ into $\hat{G}(K)$ and excludes the corresponding applications from $\hat{\mathcal{A}}$ (the same application for each $\hat{\mathcal{F}}$ as step (3) in DP).
(3) MR repeats steps (1) and (2) until at least one of the constraints in Eqs. (3b)–(3f) is violated for all MIFs of the remaining applications, or all derived reusability indices are zero.

Finally, MR outputs the planning graph. The time complexity analysis of MR is similar to DP, i.e., adding one MIF to the planning graph. The only difference is that it does not repeat the process for all MIFs. Hence, MR is $O(|\mathcal{M}|^2 V_M)$.

Extended MR (MR$^+$). We extend MR to MR$^+$ in order to provide a control knob for urban planners to trade-off complexity, i.e., execution time, and optimality, i.e., overall service utility. MR$^+$ exploits DP to derive an intermediate planning graph. Then, MR$^+$ executes MR and fuses additional MIFs into the generated planning graph to derive the planning graph. MR$^+$ supports user-specified termination criteria for DP, including but not limited to execution time and the number of included MIFs (i.e., implemented applications). MR$^+$ works with ENUM and SEL to generate the final planning graph. Although the time complexity of some algorithms is high, we can significantly reduce the execution overhead with proper parameter tuning with MR$^+$. The best way to fine-tune the parameters is out of scope and will be the future direction of this work.

4.4 Planning for Retrofit

Communities may have been equipped with some devices that urban planners can exploit. By reusing these devices, the urban planners can derive a higher service utility. Hence, we extend SmartParcels for the retrofit scenario, which includes existing devices. We propose a RETrofit (RET) algorithm that takes a set of existing devices $\mathcal{D}$ into consideration while selecting the mappings for each infrastructure flow. Let l_d denote the location of each existing device $d \in \mathcal{D}$. We modify ENUM into RET as follows.

(1) In step (1) of ENUM, if (i) $d \in \mathcal{D}$ is identical to sensor v and (ii) service site s_i is within d's sensing range r_d^{sen}, i.e., $dist(l_d, s_i) < r_d^{sen}$, where $r_d^{sen} = r_v^{sen}$, an extra mapping result, $f(v) \leftarrow l_d$ is included in $\hat{\mathcal{F}}$.

(2) If (i) $d \in \mathcal{D}$ is identical to infrastructure u (step (2) of ENUM) and (ii) d and sensor v or previously examined adjacent infrastructure u' are within each other's transmission range (step (3) of ENUM), i.e., $dist(l_d, f(v)) \leq \min(r_d^{tr}, r_v^{tr})$ or $dist(l_d, f(u')) \leq \min(r_d^{tr}, r_{u'}^{tr})$, where $r_d^{tr} = r_u^{tr}$, an extra mapping result, $f(u) \leftarrow l_d$ is included in $\hat{\mathcal{F}}$.

After the modifications, RET generates a subset of all possible MIFs for each infrastructure flow. We notice that the pruning techniques of SEL can be applied to RET as well. Like ENUM and SEL, RET works with DP and MR.

5 HUMAN-IN-THE-LOOP URBAN IOT PLANNING

SmartParcels supports *what-if analyses* on the planning graph for users to examine the network performance under unexpected events and conduct *human-in-the-loop* refinements of the IoT deployment. Fig. 5 presents the general workflow of human-in-the-loop urban IoT planning. Detail on the workflow follows.

5.1 Generation of Unexpected Events and Network Patterns

An *event generator* creates unexpected events (natural and man-made) in communities based on a specific scenario, such as an earthquake or a mass shooting. The event generator consists of multiple components. An *event environment module* creates an unexpected event in communities; a *human movement module* then generates the trajectories of humans impacted by the event. Finally, a *network pattern generator* outputs the network traffic incurred by human movement and device failures caused by the event.

Event environment module. An unexpected event is configured with *type*, *duration*, *affected areas*, and *the number of affected people*. For instance, an event with the type of earthquake occurs from 11:00 Nov. 22, 2022, to 11:05 Nov. 22, 2022, and affects 2,500 people within a 1 km radius area

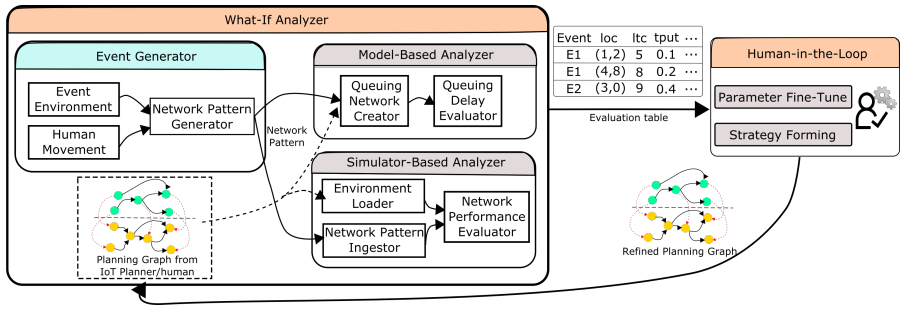


Fig. 5. General workflow: human-in-the-loop Urban IoT Planning.

centered at latitude and longitude of 33.649574 and -117.743885. Each affected area is divided into several *affected sub-areas* with different *damage levels* caused by the event.

Human movement module. Each affected person has a *starting location* in one of the affected (sub-)areas. Each person moves from location to location and stops at a specific location based on a model, such as semantic-based models [17], Markov models [41], and Generative Adversarial Networks (GANs) [25, 29, 40, 47]. That is, each person's *trajectory*, essentially a sequence of locations, is generated by the model.

Network pattern generator. The network patterns generated by the events include *extra network traffic* and *device failures*. First, human movement incurs extra network traffic to sensors and network devices in the IoT infrastructure. Second, the damage level of each device's deployed location determines its failure probability. In the worst case, a device is marked out-of-service in the deployment. The network patterns are used along with the planning graph for further analysis.

5.2 Model-Based Analyzer

A model-based analyzer promptly analyzes the IoT infrastructure's resilience by leveraging a *queueing network model*. The network patterns generated by the event generator for each unexpected event are injected into the IoT infrastructure; then, the model-based analyzer derives network metrics, such as end-to-end latency, by analyzing the IoT infrastructure with added network patterns. The network metrics are used to generate an evaluation table to indicate the areas that are more fragile to the event.

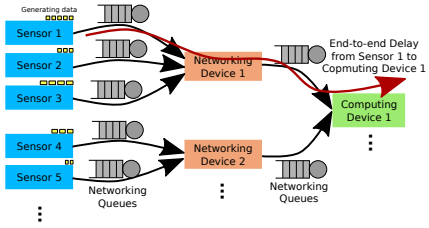


Fig. 6. Sample queuing network and end-to-end delays

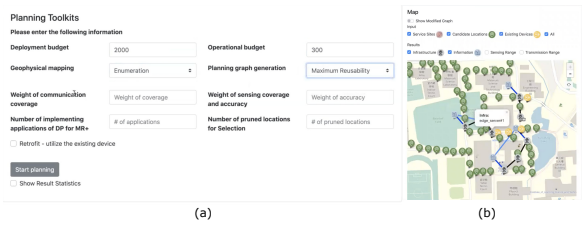


Fig. 7. Cross-layer IoT planner's web portal: (a) planning toolkits and (b) IoT deployment on a map.

In particular, we create a parameterized queueing network to evaluate the performance of the selected flows in a planning graph. Fig. 6 shows a sample queueing network containing only a planning graph's infrastructure flows for brevity. The queueing network models different performance metrics, such as delay, depending on the data size, data generation rate, and bandwidth

capacity. Each network link is modeled by a *networking queue*. In this article, we model each queue as an M/M/1 queue[2], which has an average delay of $\frac{1}{\mu-\lambda}$ [31], where μ and λ are the queue's average service rate and arrival rate, respectively. We model the delay of each network link in an infrastructure flow f_k^{ifr} of the resulting planning graph G^{p^*}. Since each link $e(u,v)$ transmits data with an average bandwidth of $w(e(u,v))$, which can be derived from the link's weight in f_k^{ifr}, we let $e(u,v)$'s average service rate $\mu_{u-v} = w(e(u,v))$. Let λ_v denote the arrival rate in bits per second of each device v in f_k^{ifr}. Each v's arrival rate is the sum of its neighbors' arrival rates with an inbound edge. Let u denote one of such neighbors; $\lambda_v = \sum_{\forall u} \lambda_u$. Note that if v is a sensor, the link's arrival rate is the sensor's data generation rate in bits per second. Thus each link $e(u,v)$'s delay is as follows:

$$T_{u-v} = \frac{1}{\mu_{u-v} - \lambda_u}. \tag{4}$$

Finally, let $v_1, v_2, v_3, \cdots, v_n$ denote a path in f_k^{ifr}. Its end-to-end delay is the sum of delays of all networking queues on the path, written as follows:

$$T_{v_1-v_n} = T_{v_1-v_2} + T_{v_2-v_3} + \cdots + T_{v_{n-1}-v_n}. \tag{5}$$

After analyzing each queue's end-to-end delay, an evaluation table is generated to indicate fragile locations that have deteriorated network performance and corresponding reaction suggestions.

5.3 Simulator-Based Analyzer

We developed an *IoT simulator* for more detailed analyses based on fine-grained models of the technology stack, such as the physical communication channel device, medium access control, and application layer protocols. The IoT simulator mimics the behavior of real devices in a modeled environment (realistic to some extent) and generates various performance metrics (including end-to-end, and per node). We introduce the IoT simulator's main software components in the following, and describe their detailed implementations in a later section.

Environment loader. The IoT simulator loads environment settings, such as geophysical location and community profiles and the planning graph generated by the cross-layer IoT planner. The infrastructure is constructed by deploying each device at the geophysical location according to the geophysical mappings indicated in the planning graph; each information unit is then installed on its designated device for execution. Each network link's traffic is further specified according to the planning graph for later network performance analyses.

Network pattern ingestor. In addition to the network traffic incurred by the information flows in the planning graph, the IoT simulator incorporates the network patterns generated by the event generator into the IoT infrastructure to analyze the network performance under the influence of unexpected events. Incorporating the network patterns affects the infrastructure's communication capability as follows. Each extra network traffic pattern increases the load and possibly creates network congestion; device failures introduced to the infrastructure cause unconnected data flows and service interruptions.

Network performance evaluator. The IoT simulator comprehensively analyzes the network performance of the IoT infrastructure with network traffic/patterns jointly created by the planning graph and unexpected events. In each simulation, sophisticated models of physical links and well-developed network protocols, such as WiFi, LoRaWAN, TCP, and UDP, are leveraged to generate each device's network performance, such as throughput, latency, and packet loss rate. Moreover, the IoT simulator combines the network performance of the devices on a path for end-to-end performance. Then, an evaluation table and reaction suggestions are created accordingly.

[2]More sophisticated queueing models can be readily adopted.

5.4 Human-in-the-Loop IoT Deployment Refinements

An *evaluation table* lists the network metrics of *fragile locations* derived from a what-if analyzer. The metrics are leveraged to generate a series of *reaction suggestions* for urban planners (or domain experts) to refine the IoT deployment to increase the resilience under these events. The suggestions can be generated based on the domain expert's experiences and/or the deterioration rate of network metrics. Sample suggestions include adding redundant devices to increase fault tolerance and network bandwidth. Urban planners can determine whether to adopt a suggestion; otherwise, they can design specific strategies based on recommendations from domain experts.

6 PROTOTYPE IMPLEMENTATION

Web-Based Cross-Layer IoT Planner: The cross-layer IoT planner enables the optimal IoT deployment on a geophysical map with the inputs (e.g., community profiles) provided by users (e.g., community administrators and/or urban planners). We implement a web-based IoT planner that provides a Graphical User Interface (GUI) portal. We give the design intuition for a comprehensive understanding of SmartParcels; however, due to the limited space, please refer to our demo paper [15] for detailed architecture.

The IoT planner's front end is a web portal that helps users to select or provide: (i) community boundaries (e.g., business and residential areas) and other important zones (e.g., fire hazard severity zones), (ii) a list of well-known applications (e.g., wildfire and gunshot detection) that can be associated with communities (i.e., service sites) along with their QoS requirements (e.g., bandwidth), (iii) budget constraints in terms of both deployment and operation, (iv) candidate locations (e.g., public spaces, traffic lights) to place devices, and (v) designing scenario (i.e., clean-slate or retrofit). Note that communities and zone boundaries can be loaded from the *OpenStreetMaps* [39] repository, where contributors have defined existing communities.

The IoT planner's back end consists of a database and a design agent. The design agent executes the core algorithms of SmartParcels. The algorithms generate planning graphs, which are stored in the database. Finally, the GUI presents each device's location and the relationship between devices on a map. Fig. 7 shows screenshots of the IoT planner's portal. As revealed in Fig. 7(a), it provides multiple user interfaces to specify necessary information and the planning method. The IoT deployment is illustrated on a map in Fig. 7(b), where users can manually move devices to explore different deployment plans.

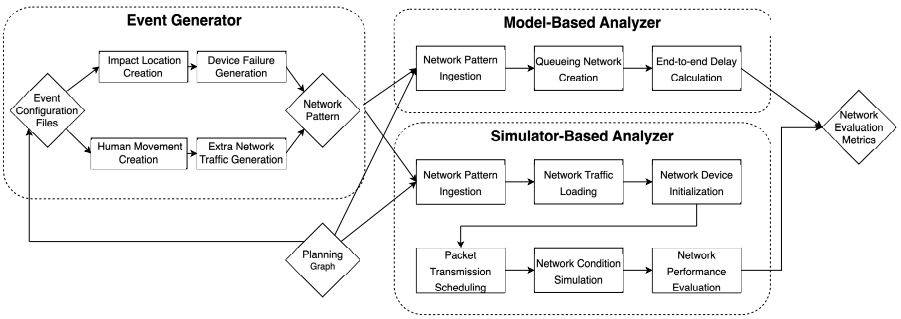


Fig. 8. What-if analyzer: software components and workflow.

Implementing What-If Analyzer: The workflow of the proposed what-if analyzer consists of two phases, as shown in Fig. 8. First, an event generator creates a network pattern based on a user-specified unexpected event. Then, a what-if analyzer evaluates the IoT deployment, generated

by the cross-layer IoT planner, under this network pattern's impact. Users can input data from urban planners, government agencies, and domain experts to create event configuration files detailing the event type, duration, affected locations, and number of affected people. These files guide the event generator in creating network patterns, such as device failures and increased traffic. Unexpected events can cause device failures, like an earthquake leading to malfunctions. Devices near the event center (e.g., epicenter) often have a higher failure probability. Thus, we associate the event with a predefined center and implement a component to select impacted locations. A user-specified threshold/distance determines the event's impact, with devices within this range considered failed: (i) deterministically (all fail) or (ii) randomly (fail uniformly with a predefined probability). Different failure models can be easily integrated into the component as input arguments to our API. Moreover, unexpected events often lead to increased network usage due to human behavior/movement. We simulate human movements by synthesizing individuals starting at predefined locations and moving towards a target (e.g., a shelter). Each movement is captured at each timestamp, forming a trajectory. Real-world data shows that movement during disasters follows a *power-law distribution* [53]; we use this to generate distances between consecutive locations for each person. At each timestamp, an extra packet is generated if a person is within a sensor's or network access point's range. Finally, device failures and extra traffic are recorded in a network pattern file and injected into the original IoT deployment plan for analysis.

We implement a component to load necessary settings, such as device locations, data flows, extra network traffic, and device failures, from the network pattern file and planning graph into memory for the what-if analyzer execution. In the prototype, we use a queuing theoretical model and NS3 [4], a well-established network simulator, for proof of concept; replacements can be adopted. We create a queuing network where each node represents a device and each edge captures the data flow rate per second, including extra data from unexpected events. Failed devices are excluded from the network, and their data flows are disregarded. The delay for each device pair with an edge is derived using Eq. (4), and the model-based analyzer outputs the end-to-end delay based on Eq. (5) as the key performance metric. We utilize NS3's official WiFi module and a open-source LoRa module [36] for comprehensive network analysis. A component loads network traffic into each device's profile, defining connected devices, communication technologies, network requirements, and data flows. Profiles also specify extra traffic from human movement during unexpected events. Devices are initialized in NS3's virtual environment based on profiles unless marked as failed. Each device is deployed at its geophysical location with necessary communication technologies (e.g., WiFi, LoRaWAN) and configurations (e.g., traffic load). WiFi access points configure sensors and edge servers as a local network with a subnet mask. LoRa devices are identified by a 32-bit *DevAddr* in the frame header. NS3 is a discrete-event network simulator, which requires scheduling each event occurring in one simulation run. Hence, we implement a component to schedule all transmission events, in which sensors send packets to edge servers via specific networking devices; then, each task is passed to NS3's simulation scheduler. We then implement a component to build the network environment based on the deployment plan and network pattern by triggering NS3 to simulate the network conditions. After the simulation, a component is responsible for computing various network performance metrics and recording each data link's metrics and average network metrics in separate log files. Finally, an evaluation table that lists critical performance metrics of each network link is generated as a reference for urban planners to evaluate the deployment.

7 PERFORMANCE EVALUATIONS

In this section, we evaluate SmartParcels from two aspects depending on how urban planners would use it. First, various reasons (land use, government policies) often restrict planning. IoT planners should optimize the deployment to account for these limitations, such as possible deployment

locations and budgets. Our results show that SmartParcels derives a better cost/performance metric with the same limitation compared to baseline methods. Second, we mimic urban planners to use the what-if analyzers to refine the deployment plan against an unexpected event (earthquake). Our results show that the deployment is more robust after the refinement and achieves a higher network throughput. More details follow.

7.1 Setup

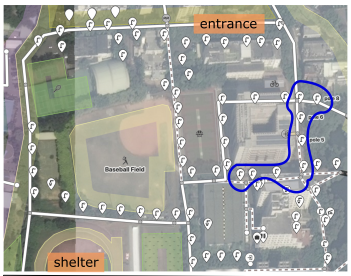
Settings. We conduct the experiments in two real-world settings: a *smart campus setting* consisting of a faculty housing community and the EECS college of National Tsing Hua University in Taiwan and a *smart city setting* consisting of four diverse communities of Irvine in California, USA. Fig. 9 shows the smart campus testbed with streetlights, where the smart one are in the mirrored L-shape region on the right. The following sensing, computing, and network devices are installed on the smart streetlights: 5 cameras, 1 motion sensor, 4 WiFi routers, and 2 edge servers. The smart streetlights are connected through a gateway to the EECS building via a Gigabit Ethernet cable. The university deploys an auto-dimming application for the streetlights to save energy. There are two ways to achieve auto-dimming: image-based analysis and motion sensor detection. The sets of service sites and candidate locations are identical, including the smart and regular streetlights. The details of streetlight locations and installed devices are added to the public OpenStreetMap (OSM) database [39], which is queried in our evaluations.

In the smart city, we consider the following communities: A, Irvine Spectrum, an outdoor shopping center; B, Quail Hill, a residential area close to a highway; C, Shady Canyon Open Space Preserve, a wildland; and D, Shady Canyon, a residential area located next to a wildland. C and D are identified as *very high fire hazard severity zones* by the Orange County fire authority [7]. The applications required by each community are A demanding gunshot detection, B demanding air quality and noise monitoring, C demanding wildfire detection, and D demanding wildfire detection and air quality monitoring. Moreover, we derive service sites and candidate locations using OSM through the OSMnx [12] library as follows. First, we retrieve the boundary of each community and draw the smallest square that can completely surround the community. Then, we split the square into 16 equal-sized cells and used the center of each cell to represent a service site of the corresponding community. Second, we consider the public facilities (found in OSM) as candidate locations since they are managed by the authority with easier access. We also consider road intersections as candidate locations because they are often close to city infrastructures, such as electricity and the Internet. We query the OSM database to get all road segments for identifying intersections, which are shown in Fig. 10, where the table lists the number of public facilities and intersections in each community. Last, we uniformly sample candidate locations across communities so that they have similar numbers of candidate locations.

Information and infrastructure flows. Besides the sample flows shown in Fig. 4, we consider the following flows in our evaluations. The flow representation can be found in Sec. 3.1.

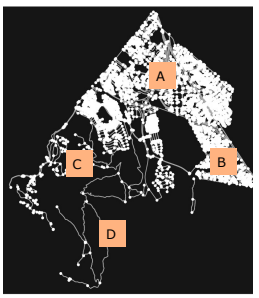
(1) **Object detection:** *{sd:image,a:object detector via image, s:camera,n:wifi}* and *{sd:motion,a:object detector via motion, s:motion sensor,n:wifi}*.
(2) **Noise monitoring:** *{sd:sound,a:noise monitor,s:mic.,n:wifi}*.
(3) **Air quality monitoring:** *{sd:emission,a:air quality monitor,s:gas sensor,n:(wifi,lora)}*.

Parameters of information units and infrastructures. We derive the parameters via specifications, empirical experiments, or reasonable assumptions to the best of our knowledge, which are summarized in Table 1. For the costs, we refer to online retails, specifications, minimum wages, cost of power, or reasonable assumptions. Table 2 lists the deployment (one time) and operational (for 24 hours) costs of each infrastructure.



	Regular streelights	Smart streetlights
Amount	63	8

Fig. 9. Smart campus and the location of streetlights.



Community	A	B	C	D
Traffic signal	185	0	0	13
Power pole	44	0	52	0
Train station	3	0	0	0
Fire station	1	0	0	0
Hospital	1	0	0	0
School	1	0	0	2
Retail	34	0	0	6
Intersection	9189	1721	734	2051

Fig. 10. Smart city and the number of possible candidate locations.

Table 1. Parameters Used in the Experiments

Parameters		Values	Parameters		Values
Bandwidth consumption	Image	10 Mbps	Trans. range & bandwidth	WiFi AP	50 m/100 Mbps
	Motion sample	1.92 Kbps		Lora gateway	1 km/50 Kbps
	Emission reading	0.64 Kbps	Sensing range	Camera	5 m (campus) 400 m (suburban)
	Sound	128 Kbps		Motion sensor	10 m
Computing resource requirement	Image	80 Mbps		Gas sensor	600 m
	Motion sample	3.84 Kbps		Microphone	300 m
	Emission reading	1.28 Kbps	Sensing parameter α_v in Eq. (2)	All sensors	Reciprocal of sensing range
	Sound	384 Kbps			
Computing resource	Edge server	6.8 GHz (4 × 1.7 GHz)			

Table 2. Costs in USD

Infras.	Deployment cost (campus)	Deployment cost (city)	Operational cost (both)
camera	1399	1863	12.02
Motion sensor	360	-	8.38
Gas sensor	-	735	5.51
Mic.	-	686	15.75
WiFi AP	236	700	8.02
Lora Gateway	-	632	20.04
Edge Server	376	840	67.03

Measurement Studies for Accuracy Models. We derive accuracies for applications with various implementation models. First, we study the auto-dimming applications with image- and motion-based detection models with accuracy of 92.7% [42] and 80% [54], respectively. We then validate the numbers by conducting experiments on our smart streetlight testbed at the smart campus and derive similar results. In the smart city, we set the accuracy of monitoring applications to 100% since they merely report sensor readings, and a 89.6% accuracy is adopted for gunshot detection based on a prior report [33]. We implement image- and emission-based models for wildfire detection with data from two real datasets: HPWREN [1] (wildland fire videos), and WAQI [3] (air quality). Over 10,000 wildland fire images are used to train a CNN model, which achieves 94.41% accuracy. We then implement a model with the support vector machine to predict wildfire events with over 1000 days of emission data (pm 2.5, O3) and derive 62.62% accuracy. Last, we normalize the accuracy to the maximum achievable accuracy for each application to mitigate the influence on our algorithms. Moreover, the weight $\beta_{i,j}$ in Eq. (3a) of both wildfire detection and gunshot detection is 0.35, and each monitoring application has 0.15 weight.

Generating unexpected events and incurred network overhead. We use the implemented event generator that incorporates an earthquake scenario where people move toward a given shelter from a common starting point (e.g., the main entrance of the campus). For instance, in Fig. 9, people flood in from the entrance and move toward the shelter. For the smart city setting, the entrance and shelter are randomly generated. The following events are generated with a timestamp and a generation rate of one second. First, each *people movement event* captures how each person moves at each timestamp; the moving distances follow the power-law distribution [53]. Each sensor has a *periodic sending event* that generates a packet per second. We approximate each sensor's traffic load by setting its packet size based on its required bandwidth consumption in Table 1. In order to create a significant load difference, the traffic load is then enlarged by 100 times. For instance, a motion sensor is transmitting a packet of 192K bits per second in our experiments. People's movement creates extra traffic in two different manners. First, besides the periodic traffic, a person

can trigger a *sensing event* that incurs the transmission of an extra packet if the person is within a sensor's sensing range at a timestamp. Secondly, an *information inquiry event* captures that people are accessing WiFi resources to derive information via social networks, VoIP, websites, etc. We assume that if a person is within a WiFi access point's transmission range at a certain timestamp, he/she will send a packet of 10M bytes via the access point. Finally, we adopt the random creator mentioned in Sec. 6 to uniformly create failures for each device.

Modeling user behaviors. We then evaluate the effectiveness of the proposed human-in-the-loop urban IoT planning tool by simulating a user's behavior in an iterative manner as follows. At each iteration, the user leverages the event generator to generate a network pattern incurred by an unexpected event (earthquake) and inject it into the IoT deployment; then, either the model- or simulator-based analyzer is used to evaluate the deployment's network performance with a few metrics. The network links with a deteriorating metric above a certain threshold, which is set to 10%, along with the corresponding suggestions, are provided for users' reference. Suggestions include: (i) high delay–add network device or increase bandwidth, (ii) low throughput–increase bandwidth, (iii) high packet loss rate–add a network device, and (iv) device failure–duplicate device. We assume that a user uniformly applies one of the suggestions at the end of each iteration. After the selected suggestions are applied to the corresponding devices on the network links, the what-if analyzer generates a new set of performance metrics with the other set of network patterns.

7.2 Evaluating SmartParcels IoT Planner with Various Deployment Options

We evaluate the IoT planner with possible scenarios that involve multiple parameters that users would encounter while designing smart communities. To the best of our knowledge, the cross-layer IoT planning problem has not been solved in the literature. Therefore, we developed a baseline algorithm called Maximum Utility (MU), which mimics the behaviors of urban planners. Specifically, MU generates a planning graph by iteratively selecting an application's MIF with the maximum utility value until running out of resources. We compare SmartParcels with MU in both smart campus and smart city settings. We consider the following metrics: (i) the overall service utility, (ii) the number of MIFs included in the planning graph, (iii) the service rate, which is the fraction of services provided by the planning graph, and (iv) the cost performance index, which is the ratio of the overall service utility to the overall cost. We list and explore common parameters that users would encounter in the following: (i) the number of candidate locations, (ii) the weights of MR to study their implications on various performance metrics, (iii) the deployment and operational budgets, (iv) the pruning conditions of SEL, and (v) the termination criteria of MR$^+$. Due to the space limit, we refer readers to Chang et al. [16] for execution time study, which shows that SmartParcels can trade off optimality and execution time well. For statistically meaningful results, we repeat the experiment of the same parameters ten times and report the average results with one standard deviation interval if applicable.

Impact of various numbers of candidate locations and budgets on smart campus. We first explore different numbers of candidate locations that can be used to build a smart campus with fixed deployment and operational budgets at 25,000 and 300 USD. We execute MR and MU to generate planning graphs from ENUM's results for clean-slate IoT plans in the smart campus. Besides, we evaluate MR with various weights: (i) MR (eff), where $\alpha_{eff} = 1$, $\alpha_{cov} = 0$, (ii) MR (cov), where $\alpha_{eff} = 0$, $\alpha_{cov} = 1$, (iii) MR (e25c75), where $\alpha_{eff} = 0.25$, $\alpha_{cov} = 0.75$, (iv) MR (e50c50), where $\alpha_{eff} = 0.5$, $\alpha_{cov} = 0.5$, and (v) MR (e75c25), where $\alpha_{eff} = 0.75$, $\alpha_{cov} = 0.25$. Fig. 11 shows that MR (e75c25) achieves the best overall performance. In Fig. 11(a), MR outperforms MU by up to 2 times on the overall service utility. That is, MR instruments infrastructure at proper locations to *double* the overall service utility. Even though maximizing the communication coverage, i.e., MR (cov), seems to be futile, the importance of communication coverage manifests when the number

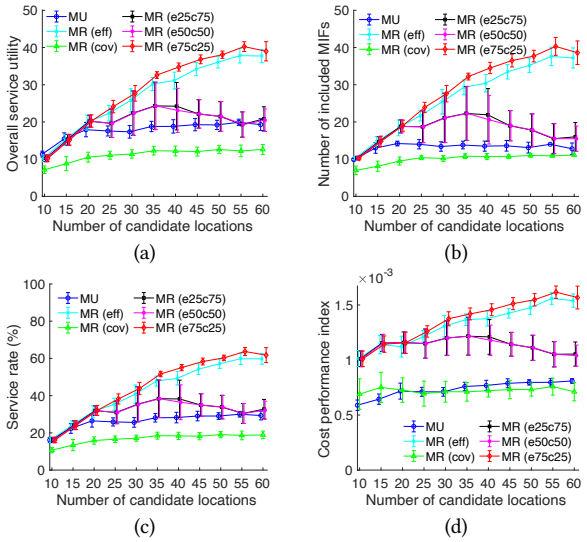


Fig. 11. Experiment results of clean-slate plan in the smart campus: (a) overall service utility, (b) number of included MIFs, (c) service rate, and (d) cost performance index.

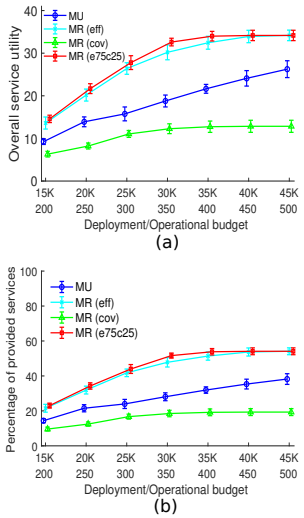


Fig. 12. Experiment results under different budgets: (a) overall service utility and (b) service rate.

of candidates grows. In fact, MR (e75c25) derives higher overall service utility than MR (eff), which only considers efficiency and ends up with deploying more network devices. Besides, MR (e75c25) is a steady approach for urban planners because it has a low performance variation. Fig. 11(b) shows that MR (e75c25) has the highest number of infrastructure flows in the planning graph, which explains its high service utility. Fig. 11(c) shows that MR (e75c25) supports the most services in the smart campus: around 60% of the demand. Last, MR (e75c25) invests carefully and derives a higher service utility per unit cost, as shown in Fig. 11(d). In summary, we empirically compare MR's performance under different weights, and find the best weights of $\alpha_{eff} = 0.75$ and $\alpha_{cov} = 0.25$, which serve as the default in following experiments. Secondly, we study the impact of different budgets on the same smart campus setting with 35 candidate locations. Intuitively, higher budgets lead to higher overall service utility. Fig. 12 shows that MR achieves much better utilization of the budgets as compared to MU. In fact, MR obtains the maximum overall service utility that can be provided by the candidate locations under lower budgets. However, MR's overall service utility saturates even when the budgets exceed 30K/350. This is because the quantity (number) and quality (distance to service sites) of candidate locations restrict the performance of MR.

Generating deployment in the smart city with a pruning algorithm. We evaluate our algorithms for a city-level environment. The deployment and operational budgets are 65,000 and 450 USD, respectively. We compare ENUM with SEL to demonstrate the effect after pruning, when MR and MU are exploited to generate the planning graph. Fig. 13(a) reports the number of generated mappings by SEL with different pruning criteria compared to ENUM. The number of all possible mappings (ENUM) increases exponentially when the number of candidate locations grows. Though the number of generated mappings from SEL also grows exponentially, SEL prunes up to 90.57% of them. Fig. 13(b) shows that MR has a tolerable loss in performance; however, MU's performance deteriorates when the number of options decreases. Interestingly, as shown in Fig. 13(c), MR identifies the most reusable MIFs and derives a high service utility. This means that ENUM-MR

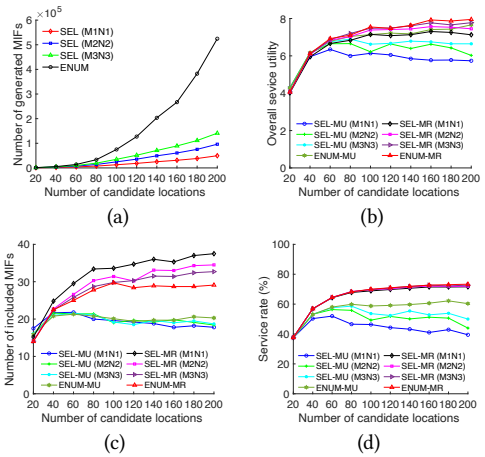


Fig. 13. Experiment results in the smart city: (a) number of generated mappings, (b) overall service utility, (c) number of included MIFs, and (d) service rate.

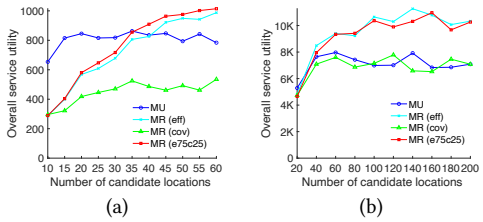


Fig. 14. Overall service utility from our event-driven simulator, sample results from the clean-slate problems in: (a) smart campus and (b) smart city.

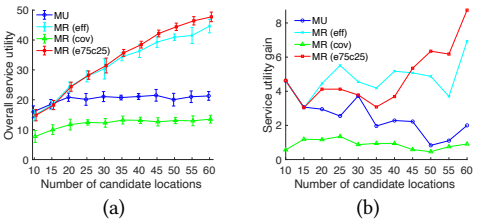


Fig. 15. Experiment results from the retrofit-smart campus: (a) overall service utility, and (b) service utility gain.

includes fewer MIFs, yet achieves the highest overall service utility. Fig. 13(d) shows that SEL imposes minimal influence on MR in terms of service rate, whereas MU is incapable of pinpointing the MIFs for higher reusability and higher overall service utility. In summary, we empirically confirm that SEL effectively prunes less-promising mappings, and MR carefully identifies MIFs with high reusability to maximize the overall service utility in larger (city-level) problems.

A Scalability Study. We study the scalability of SmartParcels since the previous simulations are conducted by assuming that an event occurs only at the service site. We implement a simulator in Python that: (i) generates events uniformly within the boundary of the testbeds, (ii) determines whether an event is captured by a planning graph, and (iii) calculates the service utility for the captured events using Eq. (1). We first generate 10,000 events for each application and compare the planning graph generated by ENUM+MR and ENUM+MU in both testbeds: the smart campus and smart city. Fig. 14(a) shows sample clean-slate results for the smart campus. We observe that MU derives higher overall service utility with fewer candidate locations, which can be attributed to the more than sufficient budget (since the campus is relatively small). However, when the number of candidate locations increases, MR performs better since it carefully instruments the devices with higher reusability. Fig. 14(b) reports sample clean-slate results from the smart city, which is larger than the smart campus. This figure reveals that MR captures the events and derives higher overall service utility, which can be explained by the relatively insufficient budgets. In summary, the results confirm that our proposed algorithms outperform the baseline algorithm under system dynamics, especially when the budget are scarce.

Retrofitting the smart campus. We next evaluate SmartParcels by solving the retrofit problems in the smart campus with the smart streetlight testbed. We compare the performance of our proposed algorithms in retrofit and clean-slate problems. For a fair comparison, we reuse the settings of the clean-slate problems in the smart campus, except for adding the existing devices on the smart streetlights to the retrofit problems. The resulting IoT plans in the retrofit problems are quite

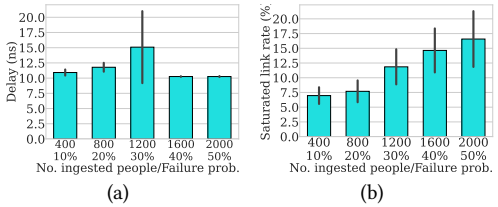


Fig. 16. Evaluation results of the deployment in the smart campus setting with the model-based analyzer: (a) delay and (b) saturated link rate.

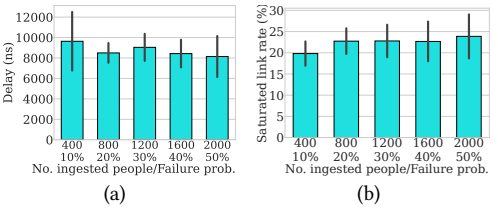


Fig. 17. Evaluation results of the deployment in the smart city setting with the model-based analyzer: (a) delay and (b) saturated link rate.

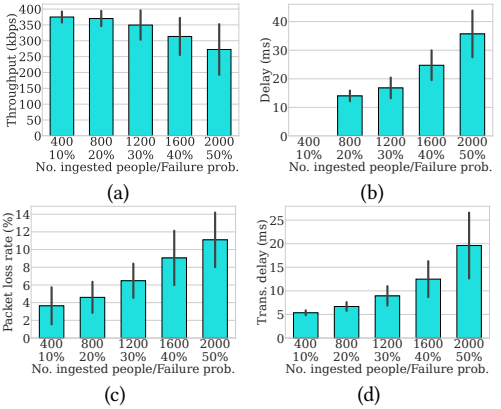


Fig. 18. Evaluation results of the deployment in the smart campus setting with the simulator-based analyzer: (a) throughput, (b) delay, (c) packet loss rate, and (d) packet transmission delay.

similar to those in the corresponding clean-slate problems. Fig. 15(a) shows the overall service utility derived by MR and MU, while Fig. 15(b) reveals the retrofit's gain of service utility, i.e., the difference of service utility between clean-slate and retrofit. It shows that MR exploits the existing devices for a significant performance improvement, which is as high as 7.64 times compared to MU.

7.3 Exploration of the Impact of Unexpected Events via the What-If Analyzers

After inspecting all possible deployment options, we select the settings that derive a relatively better performance: 35 and 120 candidate locations for smart campus and smart city. We explore the impact of unexpected events on 10 randomly selected plans generated by the clean-slate algorithm with SmartParcels' what-if analyzers. We use the setup mentioned earlier to generate unexpected events and the corresponding network pattern and device failures We generate 5 different network patterns for each deployment plan; hence, we have a total of 50 what-if analyses for each setting. The model-based analyzer computes each network link's delay, whereas the simulator-based analyzer generates more sophisticated performance metrics, including throughput, delay, and packet loss rate. The performance metrics of all links are averaged, and we report the mean average performance metrics for both analyzers, where the error bar shows one standard deviation interval.

Results from the model-based analyzer. We evaluate both the smart campus and smart city settings with different network patterns generated by injecting different numbers of people and different failure probabilities. We show the results from the proposed model-based what-if analyzer in Figs. 16 and 17 for NTHU and Irvine, respectively. Fig. 16(a) shows that the delay increases by up to 37.6% with the number of additional traffic increases. However, the model-based analyzer's drawback emerges under heavy load conditions since it cannot correctly model the delay after the network link is saturated (the last two network patterns). Fig. 17 shows a worse condition, where the delay decreases when the traffic load increases. Note that although the number of injected people is identical for both settings, the traffic load in the smart city is much higher due to the abundant data generated by sensors. We also perform experiments without device failure and observe similar results, showing that the impact of device failures is negligible.

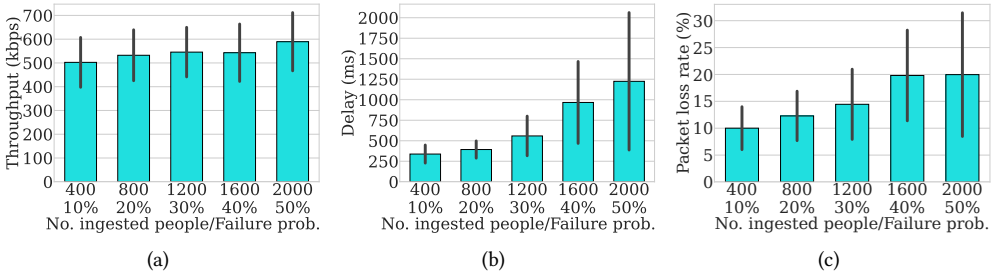


Fig. 19. Evaluation results of the deployment in the smart city setting with the simulator-based analyzer: (a) throughput, (b) delay, and (c) packet loss rate.

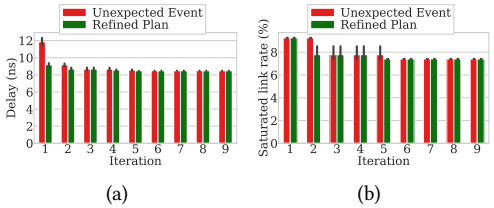


Fig. 20. Human-in-the-loop evaluations of the smart campus with model-based analyzer: (a) delay and (b) saturated link rate.

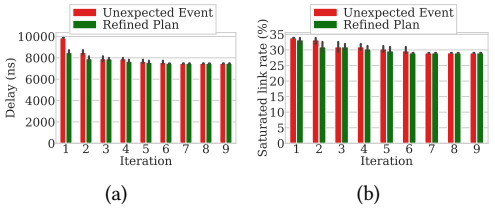


Fig. 21. Human-in-the-loop evaluations of the smart city with model-based analyzer: (a) delay and (b) saturated link rate.

Results from the simulator-based analyzer. We then evaluate the same settings with the proposed simulator-based what-if analyzer, and the results are shown in Figs. 18 and 19 for the smart campus and smart city settings. We observe that, in the smart campus and smart city, the delay increases by 5 times and 1.5 times, and the packet loss rate also increases by 2.06 times and 49.41%, which is as expected. While the throughput in the smart city increases by 18%, it decreases by 27.03% in the smart campus. Fig. 18(d) shows the packet transmission delay incurred by the random back-off mechanism of WiFi protocol, which increases by 1.05 times in the smart campus. Such a waste caused by channel idling decreases the overall throughput. In summary, we demonstrate that the proposed what-if analyzers can effectively capture the influence created by different network patterns with the IoT deployment in both smart campus and smart city settings.

7.4 Human-in-the-Loop IoT Deployment Plan Refinement

We evaluate plan refinement with the human behavior model designed in Sec. 7.1 and only focus on the effectiveness of SmartParcels. Hence, to the best of our knowledge, we devise refinement suggestions that improve the network performance, as shown in the results. The performance could be further enhanced once domain experts provide more sophisticated suggestions. We randomly select one deployment plan in both the smart campus and smart city and simulate 9 iterations of the human behavior mentioned earlier with the settings presented in the previous section. Results with the slightest impact (400 ingested people, 10% failure probability) are reported. At each iteration, we report the results under the impact of the unexpected event and the refined plan. We simulate 10 different iterative processes with randomly generated human behaviors and report the average metrics with error bars representing one standard deviation interval.

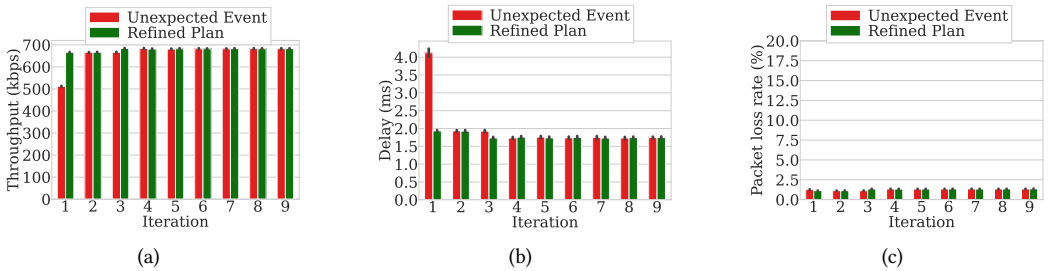


Fig. 22. Human-in-the-loop evaluations of the smart campus with simulator-based analyzer: (a) throughput, (b) delay, and (c) packet loss rate.

Evaluating the refinement of each iteration. We evaluate the model-based analyzer in both the smart campus and smart city settings. In addition to delay, we also keep track of the *saturated link rate*, which is the fraction of the saturated links. The results are shown in Figs. 20 and 21, which show significant improvement in network performance while performing human-in-the-loop refinements. In both the smart campus and smart city settings, the delay is improved by up to 28.7% and 23.4%; the saturated link rate is improved by up to 18.7% and 14.7%. The results have a similar trend; hence, we next report one of the refinement deployments for brevity. In the smart campus, after 6 iterations, 4 WiFi APs are upgraded to support higher bandwidth, and 3 extra WiFi APs are deployed to handle the excessive traffic load. Similarly, in the smart city, the bandwidth of 3 LoRa gateways and 2 WiFi APs is increased. The results from the simulator-based analyzer are shown in Figs. 22 and 23, also demonstrating our framework's effectiveness. In the smart campus and smart city, the throughput is improved by 36% and 1.4 times, respectively; similarly, the delay improves by 60.98% and 62.50%, respectively. In the smart city, the packet loss rate improves by 42.86%. In contrast, the packet loss rate in the smart campus remains low for the whole simulation, which stays fairly low even when we intentionally increased the traffic load. It is worth noting that the network performance converges faster with the simulator-based analyzer (at the fourth iteration). After convergence, 10 WiFi APs are upgraded to higher bandwidth, and 8 extra ones are deployed in the smart campus, while 8 LoRa gateways and 6 WiFi APs are upgraded in the smart city.

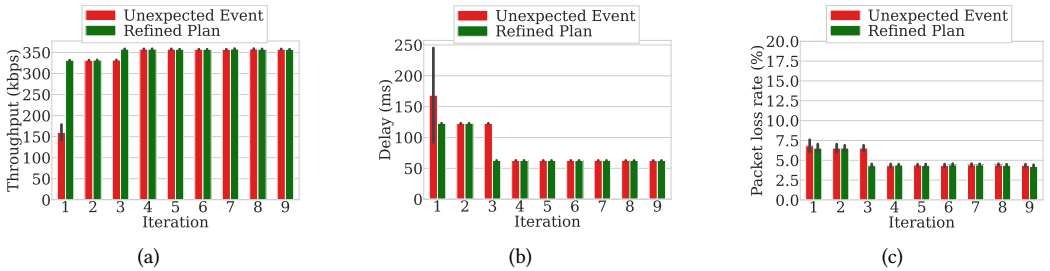


Fig. 23. Human-in-the-loop evaluations of the smart city with simulator-based analyzer: (a) throughput, (b) delay, and (c) packet loss rate.

Final deployment evaluations. We then evaluate the improvement of the final deployment after applying the human-in-the-loop process by generating another set of network patterns for each setting (number of injected people, failure probability). Note that we explore an extreme

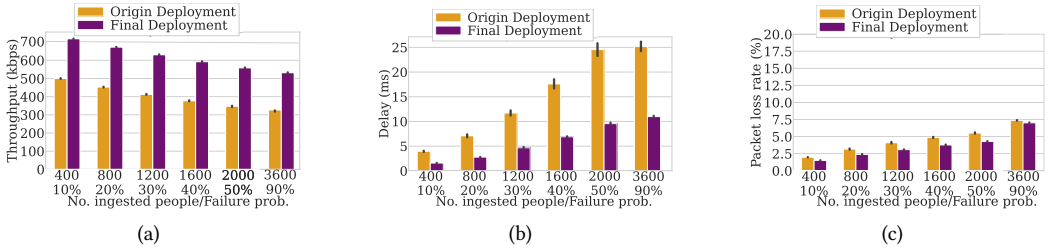


Fig. 24. The results of comparing the original and refined IoT deployments with human-in-the-loop in the smart campus with simulator-based analyzer: (a) throughput, (b) delay, and (c) packet loss rate.

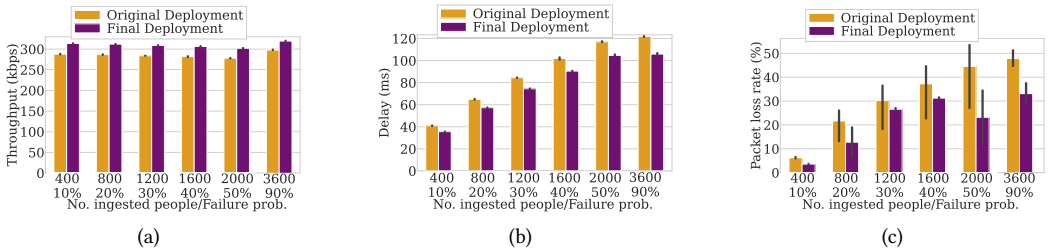


Fig. 25. The results of comparing the original and refined IoT deployments with human-in-the-loop in the smart city with simulator-based analyzer: (a) throughput, (b) delay, and (c) packet loss rate.

case where the failure probability is 90%. Each new pattern is injected into a randomly selected planning graph. We compare the network performance of the original and final (derived in the previous experiment) IoT deployment for each setting. We report the average network performance and use one standard deviation interval as the error bars. Figs. 24 and 25 show the results from both the smart campus and smart city settings. In the smart campus, throughput increases by up to 57%, delay decreases by up to 62.5%, and packet loss rate decreases by up to 22%. Due to the space limit, we report the refinement of one of the deployments–11 WiFi APs are upgraded for higher bandwidth; 10 extra WiFi APs are added to handle high traffic load. In the smart city, throughput increases by up to 9%, delay decreases by up to 8.6%, and packet loss rate decreases by up to 47.7%. Such improvements are brought by upgrading 8 LoRa gateways and 10 WiFi APs based on our suggestions. In summary, the results show that SmartParcels improves overall network performance in both the smart campus and smart city. Our framework effectively refines the IoT deployment to make it more robust to the abrupt network traffic incurred by unexpected events.

8 RELATED WORK

Related work in planning smart infrastructures has been conducted at different layers of the system stack, often independently. Early work originated in network planning for deploying wireless infrastructures and sensor placement in wireless sensor networks. The canonical *sensor placement problem* [51], which aims to maximize the coverage area for a wireless sensor network, has yielded multiple sensing models. Omnidirectional coverage models [14, 52, 56] assume that sensors have a 360-degree sensing angle, while directional coverage models are used when sensors have limited sensing angles [8, 13, 22, 35]. The distance between the sensor and sensing target is a factor that impacts coverage; this is typically represented using models where sensing probabilities decay with

distance [37]; other techniques truncate sensor values when distance thresholds are met [58, 59]. At the network layer, communication coverage problems focus on providing network connectivity to devices through gateways. Gateway placement has been studied from many perspectives, including collecting data in neighborhood wireless networks [45]. Algorithmic approaches to gateway placement problems have been designed to maximize throughput in multi-hop wireless mesh networks with multiple gateways [32] and under cost-QoS constraints [9, 20, 27]. Multi-network placement solutions [24] combine expensive gateways with low-cost transmission equipment to reduce gateway deployment costs. Preliminary efforts consider sensing in a simple connectivity model [19] to address coverage concerns. Contrary to the above efforts, SmartParcels takes an integrated approach that deals with the sensing, communication, computing, and application layers using heterogeneous capabilities of devices while incorporating community structure and needs.

While not exactly addressing the planning problem, resource management problems of IoT infrastructures have also been investigated. For example, Xavier et al. [55] considered the management problem of heterogeneous networks to increase the overall usage of computing resources. Their proposed solution leverages the diverse properties of IoT applications to improve the collaboration among edge devices. Similarly, Miele et al. [38] studied the runtime resource management problem of fog-computing infrastructures. Their objectives were to balance the dynamic workload among distributed devices and to minimize overall power consumption. Lin et al. [34] introduced an algorithm for optimizing IoT service placement in fog computing environments. The algorithm minimizes resource utilization and delays while balancing multiple objectives (service cost, response time, throughput). Studies have been proposed to enhance the overall performance of sensing across various aspects, such as energy-efficient sensing [26], quality-aware sensing [30], crowd-augmented sensing [57], and resource-efficient adaptive monitoring [49]. Furthermore, edge proxies have demonstrated their capability to improve both QoS and energy efficiency for IoT applications [11]. Additionally, with each device in the IoT infrastructure potentially equipped with multiple network interfaces, dynamic traffic management can be facilitated by software-defined networks [43, 44]. Different aspects of human involvement are studied. Elmalaki [21] presented a reinforcement learning framework to address variability among users for personalized adaptations to improve user experience, especially when multiple users interact with the same system. Jin et al. [28] proposed an incentive mechanism to encourage users with reliable data to participate in mobile crowd-sensing systems while ensuring user privacy protection. These works are orthogonal to our considered urban IoT planning problem.

9 CONCLUSION

The emergence of IoT technologies facilitates sensing, networking, and computing capabilities in our communities. Smart applications are hence becoming achievable and accessible. We propose SmartParcels, a framework that assists planning such smart communities by generating a sophisticated deployment plan of devices with discretion in communities' requirements. The framework encompasses two general design scenarios, clean-slate and retrofit, that build smart communities from scratch and on top of existing devices. It provides a control knob to trade off efficiency (execution time) and optimality (overall service utility) for exploring different design-space possibilities of customized applications that are relevant to the community at hand. SmartParcels then includes experts in the design loop to reinforce the deployment against unexpected events like natural disasters. Its what-if analyzers allow experts to bring in their domain knowledge while generating the plan. Hypothetical scenarios and corresponding influences are generated to assess the deployment's robustness. Remedies are then applied to reinforce the plan. We perform comprehensive experiments, which demonstrate that SmartParcels can generate a deployment plan that provides sufficient and robust service of smart applications. In two real-world settings,

SmartParcels' deployment plan achieves over 7 times improvement in service utility and benefits from the human-in-the-loop design with a 57% higher network throughput when a natural disaster occurs. These usecase studies confirm the value that SmartParcels brings to the planning of smart communities, and we believe that its design is general enough to be applied to other scenarios.

ACKNOWLEDGEMENT

We acknowledges the funding support from the University of California Laboratory Fees Research Program funded by the UCOffice of the President, grant ID LFR-20-653572, NSF IIS Award #40284388, and J. Yang & Family Foundation fellowship.

REFERENCES

[1] 2000. HPWREN . https://hpwren.ucsd.edu/.
[2] 2003. IQ FireWatch . https://www.iq-firewatch.com/technology.
[3] 2007. Air Quality Open Data Platform . https://aqicn.org/data-platform/register/.
[4] 2011. NS3 Network Simulator . https://www.nsnam.org/.
[5] 2015. Securaxis Acoustic Monitoring . https://securaxis.com/sounds-analytics.
[6] 2018. Wildfire Detection System. https://www.insightrobotics.com/en/services/wildfire-detection-system/.
[7] 2023. Orange County Fire Authority . https://https://www.ocfa.org/.
[8] Jing Ai et al. 2006. Coverage by directional sensors in randomly deployed wireless sensor networks. *Journal of Combinatorial Optimization* 11, 1 (2006), 21–41.
[9] Bassam Aoun, Raouf Boutaba, Youssef Iraqi, and Gary Kenward. 2006. Gateway placement optimization in wireless mesh networks with QoS constraints. *IEEE Journal on Selected Areas in Communications* 24, 11 (2006), 2127–2136.
[10] Shrey Baheti, Shreyas Badiger, and Yogesh Simmhan. 2021. VIoLET: An Emulation Environment for Validating IoT Deployments at Large Scales. *ACM Transactions on Cyber-Physical Systems* 5, 3 (2021), 1–39.
[11] Kyle Benson et al. 2018. Ride: A resilient IoT data exchange middleware leveraging SDN and edge cloud resources. In *IEEE/ACM IoTDI*.
[12] Geoff Boeing. 2017. OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems* 65 (Sep 2017), 126–139.
[13] Yanli Cai et al. 2007. Target-oriented scheduling in directional sensor networks. In *IEEE INFOCOM*. 1550–1558.
[14] Mihaela Cardei et al. 2005. Maximum network lifetime in wireless sensor networks with adjustable sensing ranges. In *IEEE WiMob*, Vol. 3. 438–445.
[15] Tung-Chun Chang et al. 2021. SmartParcels: A What-If Analysis and Planning Tool for IoT-Enabled Smart Communities: Demo Abstract. In *ACM/IEEE IoTDI*. 267–268.
[16] Tung-Chun Chang and otehrs. 2021. SmartParcels: Cross-Layer IoT Planning for Smart Communities. In *ACM IoTDI*.
[17] Andrew Chio et al. 2022. Smartspec: Customizable smart space datasets via event-driven simulations. In *IEEE PerCom*.
[18] Michael Davidson et al. 2004. *A planners dictionary*. American Planning Association, Planning Advisory Service.
[19] Runliang Dou and Guofang Nan. 2015. Optimizing sensor network coverage and regional connectivity in industrial IoT systems. *IEEE Systems Journal* 11, 3 (2015), 1351–1360.
[20] Yasir Drabu et al. 2008. Gateway placement with QoS constraints in wireless mesh networks. In *IEEE ICN*. 46–51.
[21] Salma Elmalaki. 2021. Fair-iot: Fairness-aware human-in-the-loop reinforcement learning for harnessing human variability in personalized iot. In *ACM/IEEE IoTDI*. 119–132.
[22] Giordano Fusco et al. 2009. Selection and orientation of directional sensors for coverage maximization. In *IEEE SECON*.
[23] Grand View Research. 2022. Smart Cities Market Size & Analysis, Industry Report, 2023-2030. https://www.grandviewresearch.com/industry-analysis/smart-cities-market.
[24] Ilias Gravalos, Prodromos Makris, Kostas Christodoulopoulos, and Emmanouel A Varvarigos. 2018. Efficient network planning for internet of things with QoS constraints. *IEEE Internet of Things Journal* 5, 5 (2018), 3823–3836.
[25] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. 2018. Social gan: Socially acceptable trajectories with generative adversarial networks. In *IEEE CVPR*. 2255–2264.
[26] Qi Han et al. 2004. Energy efficient data collection in distributed sensor environments. In *IEEE ICDCS,*.
[27] Bing He et al. 2007. Optimizing the internet gateway deployment in a wireless mesh network. In *IEEE MASS*. 1–9.
[28] Haiming Jin, Lu Su, Houping Xiao, and Klara Nahrstedt. 2018. Incentive mechanism for privacy-aware data aggregation in mobile crowd sensing systems. *IEEE/ACM Transactions on Networking* 26, 5 (2018), 2019–2032.
[29] Vaibhav Kulkarni et al. 2018. Generative models for simulating mobility trajectories. *arXiv:1811.12801* (2018).
[30] Iosif Lazaridis et al. 2004. Quasar: quality aware sensing architecture. *ACM SIGMOD Record* 33, 1 (2004), 26–31.

[31] Edward D. Lazowska, John Zahorjan, G. Scott Graham, and Kenneth C. Sevcik. 1984. Quantitative system performance, computer system analysis using queuing network models, by prentice-hall. *Inc. Englewood Cliffs, NJ* (1984).
[32] Fan Li et al. 2008. Gateway placement for throughput optimization in wireless mesh networks. *Mobile Networks and Applications* 13, 1-2 (2008), 198–211.
[33] Hyungui Lim et al. 2017. Rare sound event detection using 1D convolutional recurrent neural networks. In *DCASE*.
[34] Zhen Lin et al. 2023. An efficient and autonomous planning scheme for deploying IoT services in fog computing: A metaheuristic-based approach. *IEEE Transactions on Computational Social Systems* 11, 1 (2023), 1415–1429.
[35] Liang Liu et al. 2008. On directional k-coverage analysis of randomly deployed camera sensor networks. In *IEEE ICC*.
[36] Davide Magrin, Martina Capuzzo, and Andrea Zanella. 2019. A thorough study of LoRaWAN performance under different parameter settings. *IEEE Internet of Things Journal* 7, 1 (2019), 116–127.
[37] Seapahn Megerian et al. 2002. Exposure in wireless sensor networks: Theory and practical solutions. *Wireless Networks* 8, 5 (2002), 443–454.
[38] Antonio Miele and otehrs. 2022. A Runtime Resource Management and Provisioning Middleware for Fog Computing Infrastructures. *ACM Transactions on Internet of Things* 3, 3 (2022), 1–29.
[39] OpenStreetMap. 2017. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org.
[40] Kun Ouyang, Reza Shokri, David S Rosenblum, and Wenzhuo Yang. 2018. A non-parametric generative model for human trajectories. In *International Joint Conference on Artificial Intelligence*, Vol. 18. 3812–3817.
[41] Luca Pappalardo and Filippo Simini. 2018. Data-driven generation of spatio-temporal routines in human mobility. *Data Mining and Knowledge Discovery* 32, 3 (2018), 787–829.
[42] Qiwei Peng et al. 2016. Pedestrian detection for transformer substation based on gaussian mixture model and YOLO. In *IEEE IHMSC*, Vol. 2. 562–565.
[43] Zhijing Qin et al. 2014. A software defined networking architecture for the internet-of-things. In *IEEE NOMS*.
[44] Zhijing Qin, Luca Iannario, Carlo Giannelli, Paolo Bellavista, Grit Denker, and Nalini Venkatasubramanian. 2014. MINA: A reflective middleware for managing dynamic multinetwork environments. In *IEEE NOMS*.
[45] Lili Qiu et al. 2004. Optimizing the placement of integration points in multi-hop wireless networks. In *IEEE ICNP*.
[46] Christoph Reinhart et al. 2013. Umi-an urban simulation environment for building energy use, daylighting and walkability. In *IBPSA BS*, Vol. 1. 476–483.
[47] Luca Rossi, Marina Paolanti, Roberto Pierdicca, and Emanuele Frontoni. 2021. Human trajectory prediction and generation using LSTM models and GANs. *Pattern Recognition* 120 (2021), 108136.
[48] Ana Solórzano et al. 2017. Fire detection using a gas sensor array with sensor fusion algorithms. In *ISOCS/IEEE ISOEN*.
[49] Praveen Venkateswaran et al. 2020. Ream: Resource efficient adaptive monitoring of community spaces at the edge using reinforcement learning. In *IEEE SMARTCOMP*.
[50] Emmanuel Walter et al. 2015. A verification of CitySim results using the BESTEST and monitored consumption values. In *BPSA BSA*. 125–222.
[51] Bang Wang. 2011. Coverage Problems in Sensor Networks: A Survey. *Comput. Surveys* 43, 4 (2011), 1–53.
[52] Jiong Wang and Sirisha Medidi. 2007. Energy efficient coverage with variable sensing radii in wireless sensor networks. In *IEEE WiMob*. 61–68.
[53] Qi Wang and E Taylor. 2016. Patterns and limitations of urban human mobility resilience under the influence of multiple types of natural disaster. *PLoS one* 11, 1 (2016), e0147299.
[54] Libo Wu, Ya Wang, and Haili Liu. 2018. Occupancy detection and localization by monitoring nonlinear energy flow of a shuttered passive infrared sensor. *IEEE Sensors Journal* 18, 21 (2018), 8656–8666.
[55] Tiago Xavier et al. 2022. Managing heterogeneous and time-sensitive IoT applications through collaborative and energy-Aware resource allocation. *ACM Transactions on Internet of Things* 3, 2 (2022), 1–28.
[56] Zongheng Zhou, Samir Das, and Himanshu Gupta. 2009. Variable radii connected sensor cover in sensor networks. *ACM Transactions on Sensor Networks (TOSN)* 5, 1 (2009), 1–36.
[57] Qiuxi Zhu et al. 2018. Spatiotemporal scheduling for crowd augmented urban sensing. In *IEEE INFOCOM*.
[58] Yi Zou et al. 2004. Sensor deployment and target localization in distributed sensor networks. *ACM Transactions on Embedded Computing Systems* 3, 1 (2004), 61–91.
[59] Yi Zou and Krishnendu Chakrabarty. 2005. A distributed coverage-and connectivity-centric technique for selecting active nodes in wireless sensor networks. *Transactions on Computers* 54, 8 (2005), 978–991.