Human-Intent-Driven Cellular Configuration Generation Using Program Synthesis

Fuliang Li, *Member, IEEE,* Chenyang Hei, Jiaxing Shen, *Member, IEEE,* Qing Li, Xingwei Wang, *Member, IEEE*

***Abstract*—Cellular networks are vital for emerging applications like the Metaverse, which impose demanding quality and quantity requirements. This necessitates frequent reconfiguration of both new and existing base stations to balance network service quality (e.g., ultra-low latency and high bandwidth) and resource consumption. Existing data-driven configuration methods learn from historical data, but have two key limitations. First, they yield only approximate solutions, lacking precision. Second, poor bootstrapping for new base stations with previously unobserved attributes. In this paper, we pioneer intent-driven configuration synthesis by designing an intent language and utilizing satisfiability modulo theory (SMT) for cellular networks to enable exact and precise solutions. We formulate synthesis as an SMT problem, permitting verification of precision. First, we cast configuration generation as a program synthesis problem via novel modeling to bridge the intent-configuration gap. Second, we extend SMT synthesis to scale to large networks. However, vanilla SMT approaches have poor scalability. Hence, we propose an optimization using sampling for constraint verification instead of exhaustive forward solving. We also design a domain-specific optimization to prune the sample space and improve efficiency. Experiments on various network scales demonstrate the effectiveness of our proposed SMT-based cellular network configuration synthesis.**

***Index Terms*—Intent-based network, Cellular configuration, Program synthesis, Metaverse**

I. Introduction

RECENT years have witnessed the rapid evolution of cellular networks, which can now provide reliable services for digital applications (e.g., the Metaverse) with demanding requirements like high bandwidth, low latency, and high-speed transmission. Cellular networks also feature robust mobility support, enabling seamless handovers during user movement and thereby facilitating an immersive user experience. However, ensuring the quality of service of cellular networks necessitates deploying a massive number of base stations. This poses a severe challenge for network engineers owing to the configuration complexity involved. Cellular network configuration encompasses various interdependent tasks like wireless connection management, mobility management, and radio resource management. The complex interplay between configuration parameters and performance goals often exceeds human analytical capabilities, resulting in suboptimal or incorrect configurations. Such improper configurations can degrade network services and cause failures. Moreover, rendering Metaverse environments demands extensive computing and data acquisition, with dynamic and uncertain network requirements as users move [1]. To deliver a seamless and immersive Metaverse experience, cellular base stations need adaptive configuration for guaranteed services like ultra-low latency and high bandwidth amidst fluctuating demands.

To alleviate the complexity of network configuration while enhancing the accuracy and efficiency of dynamic configuration, it is imperative to implement a configuration synthesis methodology. *Configuration synthesis* refers to a method that can automatically generate accurate network configurations aligned with operational management objectives. With configuration synthesis, network configuration no longer relies on manual mapping of high-level abstract policies to low-level device configurations using rulebooks.

Existing approaches for automatic cellular network configuration are largely data-driven [2]–[4]. These approaches generate configurations by referring to similar historical cases or using reinforcement learning to optimize configurations based on network feedback. However, data-driven approaches have limitations. First, they struggle with new configuration scenarios with a different distribution from the training data, especially under distribution uncertainty. Second, the high-dimensional deep learning models provide poor interpretability into the generated configurations. This impedes subsequent maintenance and limits human experts' ability to comprehend and refine the configurations.

Intent-driven configuration synthesis offers a secure and reliable alternative that enables operators to effectively "plan" network configurations aligned with management objectives. The intent-driven configuration synthesis problem is NP-hard [5], [6], meaning exact solutions satisfying intent exist but finding them is non-trivial. Intent-driven approaches have been widely adopted in traditional networks [5]–[18] but not yet extended to cellular networks. Moreover, modeling techniques from existing work do not readily generalize. Graph algorithms often lack granularity to represent low-level configurations. This leaves a semantic gap between intent and configuration. Formal modeling requires precise mathematical formulations, making it difficult to capture the many interdependent cellular network parameters. Additionally, these modeling approaches lack generality and require significant reworking when extended to diverse cellular network functions.

In this paper, we propose Drone, an intent-driven framework for cellular network configuration synthesis. Drone can automatically derive high-level management intent written by

Fuliang Li, Chenyang Hei, Qing Li, and Xingwei Wang are with the Northeastern University, Shenyang 110819, China. E-mail: heichenyang429@163.com, lifuliang@cse.neu.edu.cn, george_wxd@163.com, wangxw@mail.neu.edu.cn.

Jiaxing Shen is with Lingnan University, Hong Kong, China. Email: jiaxingshen@LN.edu.hk.

(Corresponding authors: Xingwei Wang.)

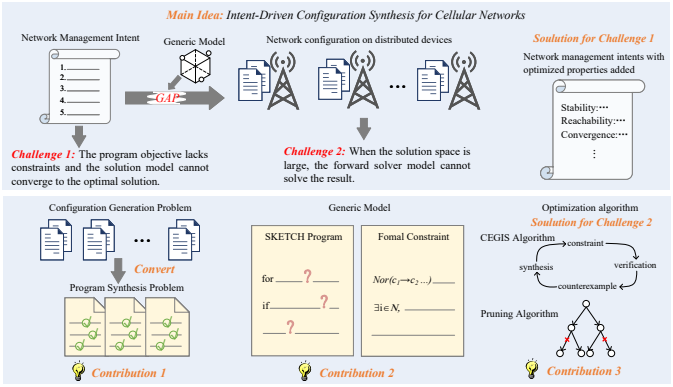


Fig. 1. Illustration of Drone's main idea, challenges, corresponding solutions, and contributions.

operators into low-level device configurations. First, to flexibly and accurately capture operating objectives, we design an abstraction layer and propose the Cellular Network Intent Language (CEL). CEL provides communication primitives to abstract related parameters, simplifying policy writing. Second, to bridge the semantic gap between intent and configuration, we introduce a novel modeling approach using SKETCH programs with "holes". Third, to handle complex parameter dependencies, we recast configuration generation as a program synthesis problem and adopt a satisfiability modulo theory (SMT)-based technique. The objective function formulated is $Consistency(INTENT, POs)$ rather than an optimization over configurations. In contrast to data-driven approaches, which yield probabilistic and approximate solutions, SMT-based synthesis enables comprehensive and exact solutions by encoding network behavior in first-order logic. Finally, we build models over program objectives rather than parameters directly. This increases the abstraction level and eases accurate modeling.

However, we encountered two key challenges in solving the model: First, the lack of constraints on program objectives prevents the solution from converging optimally. Second, encoding all possible constraints scales poorly as network size increases, hampering synthesis efficiency. To address the first issue, we add network optimization constraints to the intent specification. Program objectives then converge to satisfy these constraints. For the second issue, we replace the forward synthesis solver with a counterexample-guided inductive synthesis (CEGIS) algorithm [19], [20]. This accelerates the synthesis process. We also propose a domain-specific optimization to prune the search space and further improve efficiency.

We extensively evaluated the efficiency and scalability of configuration synthesis across network sizes. Experiments demonstrate our approach can steadily synthesize configurations within 1 second for small networks and 10 minutes for large networks. These results validate the efficacy of our method for improving management efficiency. We also conducted comparison experiments to verify the optimization algorithm. Results show the algorithm accelerated synthesis speed by 3.83-485x, with 97.50x average improvement. Overall, the experiments prove the effectiveness of our approach in enhancing configuration synthesis performance and scalability.

In summary, our contributions are as follows:

- We pioneer the application of intent-driven configuration synthesis to cellular networks.
- Our core insight is introducing program synthesis techniques to tackle the cellular network configuration problem.
- Experiments validate our approach can effectively enhance configuration efficiency, synthesizing configurations within 1 second for small networks and 10 minutes for large networks.

The remainder of this paper is organized as follows. Section II presents background and motivation for this work. Section III details the design of the abstraction layer, intent representation language, and SKETCH program modeling in our proposed synthesis framework. Section IV describes the constraint encoding, program synthesis algorithm, and optimization approach developed. Section V presents experiments evaluating configuration synthesis efficiency and results. Section VI discusses directions for future work. Finally, Section VII concludes by summarizing this work.

II. Background and Motivation

Cellular network function management includes wireless access management, mobility management, wireless resource management, etc. Some of these functions require parametric precision, such as cellular network security, so we employ a formal approach that can be verified to be completely accurate. Such methods perform well in scenarios that require both precision and completeness. In this paper, we focus on the design of the configuration synthesis scheme, and take the mobility management function as an example to illustrate, in this section, we describe the basic protocol process of the mobility management function, network failure caused by the configuration error, our motivation for proposing the configuration synthesis scheme, and research related to cellular network configuration synthesis.

A. Mobility Management

Mobility management is a key mechanism for the cellular network to provide seamless handover for mobile users. As the core technology in mobility management, handover is mainly divided into two categories, unconnected state handover, and connected state handover. Next, we will introduce the basic process of mobility handover from two perspectives of connected state and unconnected state [21]:

1) Unconnected state handover: This process is also known as cell selection and reselection. When the user equipment (UE) is in the idle state or inactive state, it will select a certain cell as the resident cell and continue to perform cell reselection. The main purpose of this process is to balance the random access load between different frequency points and to enable the UE to obtain a better quality of service, it is necessary to select a cell with better signal quality when selecting the resident cell.

To perform cell selection/reselection, the UE needs to perform cell measurement. To measure the signal quality of the cell, the gNB needs to configure parameters for the cell,

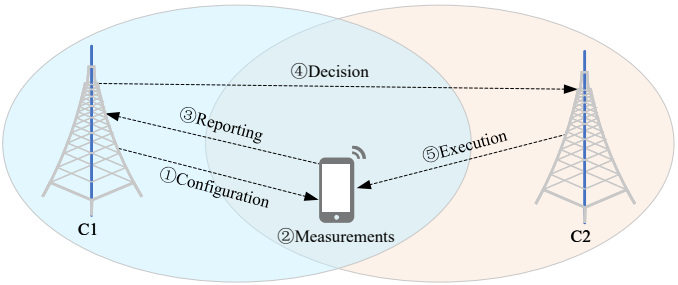


Fig. 2. The handover process of UE in inter-cell consists of five steps: ①configuration broadcast ②UE measurement ③measurement report ④handover decision ⑤handover execution.

such as cell priority, handover threshold, etc. The cell selection process follows the S criterion:

$$S_{rxlev} = Q_{rxlevmeas} - (Q_{rxlevmin} + Q_{rxlevminoffset}) - P_{compensation} - Qoffset_{temp} \quad (1)$$

$$S_{qual} = Q_{qualmeas} - (Q_{qualmin} + Q_{qualminoffset}) - Qoffset_{temp} \quad (2)$$

$$S_{rxlev} > 0 \bigwedge S_{qual} > 0 \quad (3)$$

where S_{rxlev} and S_{qual} are the cell selection reception and quality values, $Q_{rxlevmeas}$ and $Q_{qualmeas}$ are the RSRP and RSRQ values of the measuring cell, $Q_{rxlevmin}$ and $Q_{qualmin}$ are the minimum reception strengths of RSRP and RSRQ in the cell, and $Q_{rxlevminoffset}$ and $Q_{qualminoffset}$ are the signal offsets.

The cell reselection process is divided into intra-frequency cell reselection and inter-frequency cell reselection, wherein intra-frequency cell reselection follows the R criterion:

$$R_s = Q_{meas,s} + Q_{hyst} - Qoffset_{temp} \quad (4)$$

$$R_n = Q_{meas,n} - Qoffset - Qoffset_{temp} \quad (5)$$

where R_s and R_n represent the R-value of the serving cell and the neighboring cell, Q_{hyst} is the hysteresis value of the sorting standard, and Q_{meas} denotes the cell RSRP value for cell reselection.

The NR inter-frequency cell reselection process follows the priority-based cell reselection process of LTE, and the network realizes load balancing of different frequency points by designing reasonable priority parameters.

2) Connected state handover: When the UE is in the connected state, its mobility process is completely controlled by the network. Handover may be triggered when the UE channel conditions change and may also occur for load balancing. The handover process includes a control plane process and a data plane process. The handover process of the control plane includes three stages: handover preparation stage (parameter transfer, handover decision), handover execution stage (signaling generation, handover command transmission), and handover completion stage (random access, path switching); data plane handover process includes data forwarding, etc., and it happens simultaneously with the control plane handover process.

The process of cell handover is graphically depicted in Figure 2. The figure shows that the process is distributed and there is no centralized node as the center for collecting measurement results or making handover decisions. Each decision is made by the local cell or UE, and the target cell is selected by the decision made by the serving cell or UE. The handover process includes the following five steps [22], [23]: step ①, the current serving cell sends a system broadcast to all UEs it serves, the broadcast contains the configuration related to handover and measurement; step ②, the UE turns on the radio signal quality measurement of adjacent cells according to the received configuration. When the UE satisfies the handover conditions in the configuration broadcast by the serving cell, it triggers an "event"; step ③, the UE sends a summary of the event in the form of a measurement report to the serving cell; step ④, the serving cell makes a handover decision; step ⑤, the target cell executes the handover decision (including reserving radio resources for the accessing UE).

B. Configuration Failure

The above-mentioned handover process includes many interactions between system configuration parameters and terminal measurement values. Existing studies have shown that uncoordinated handover parameter configurations will cause the network to fall into an unstable state, such as handover loops and handover oscillations [21]. Such instability will lead to frequent handovers, which will generate a large amount of control signaling, seriously occupying channel and network bandwidth resources. In addition, when the cell configuration is missing or is not coordinated with the operator's service mechanism, handovers cannot converge to the target cell [24]. In Sections III and IV, we further illustrate these misconfiguration cases and the performance loss they cause and propose corresponding constraints to solve such problems.

C. Related Work

Configuration synthesis. The core idea of configuration synthesis technology is to automatically derive low-level configurations on distributed devices through the operator's high-level management intent. AED [6] proposes an incremental synthesis and repair scheme that models configuration updates as the addition/deletion of nodes on the configuration syntax tree. It uses the MaxSMT constraint system to encode high-level policies as soft constraints. Propane [9] synthesizes BGP protocol configuration by proposing a high-level policy compiler and modeling topology, routing policies, and fault tolerance requirements into an intermediate representation. Lucid [13] suggests an event-based high-level abstraction language to reduce the difficulty of writing data plane applications. The language puts control functions into the data plane while proposing a specific type-checking and syntax system to prevent data plane state errors. Config2Spec [11] mines the formal configuration specification of the network according to the input network configuration and fault model. This method mainly relies on the combined algorithm of data plane analysis and control plane verification to synthesize large-scale policies. All these research works mentioned above

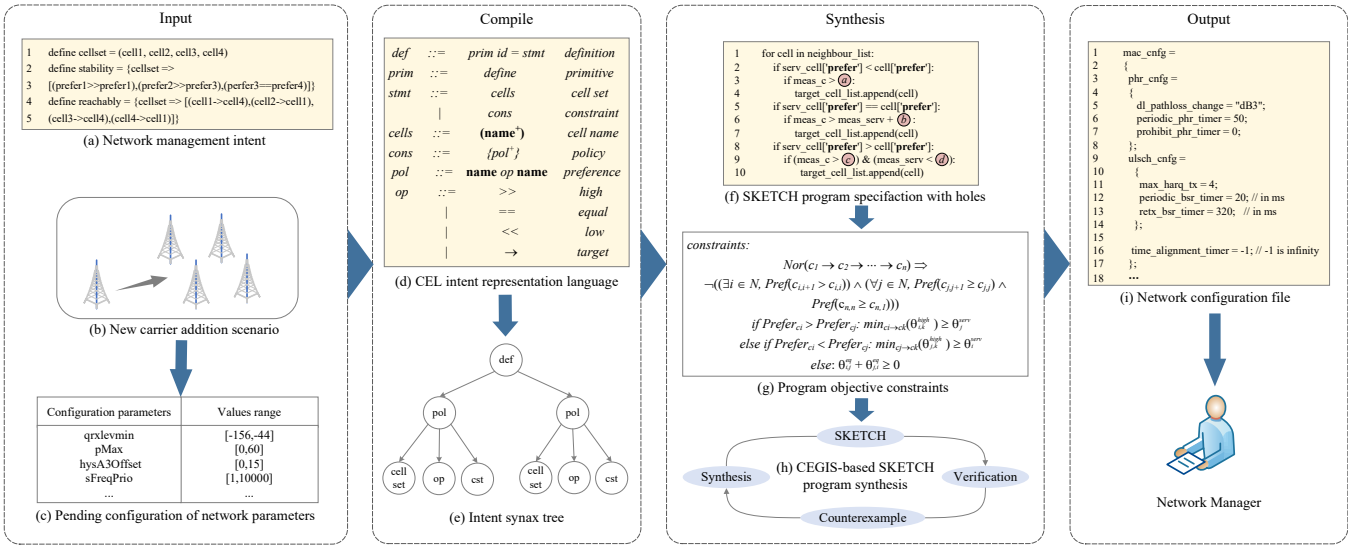


Fig. 3. Overview of Drone. The inputs are: (a) the intent to manage the network, (b) the network configuration scenarios, and (c) the range of network parameters. The framework abstraction layer includes: (d) intent compiler and (e) intent intermediate representation. The modeling and solution part includes: (f) modeling of the SKETCH program with "holes", (g) program objectives constraint modeling, and (h) counterexample-guided inductive synthesis solution algorithm. The output is: (i) the configuration of the distributed target devices.

are configuration synthesis studies under wired networks such as traditional networks, SDN networks, or programmable data plane networks. The main research objectives are routing control and data plane forwarding rules, and they do not extend to cellular network scenarios or model the policy requirements of cellular networks.

Cellular network configuration generation. The existing research solutions are all based on the data-driven approach to train a set of configuration parameters that enable the network to perform relatively well. Chuai Jie et al. [4] proposed a multi-cell data collaborative learning method to optimize cellular network configuration parameters. This method combines context bandit algorithm and transfer learning to improve learning and decision-making efficiency and network performance. Auric [2] is a scheme that uses learning algorithms and geographic proximity to generate recommended configurations for newly added carriers. However, it cannot bootstrap configurations, i.e., it can only recommend observed configurations with similar attribute carriers. If there are unobserved attributes, its recommended solution may not be optimal, and engineers are still required to adjust these parameters using manual methods. It does not allow for flexible configurations according to management intent (for example, different parameters are configured for similar carriers according to different policies).

III. Configuration Abstraction Layer and Program Modeling

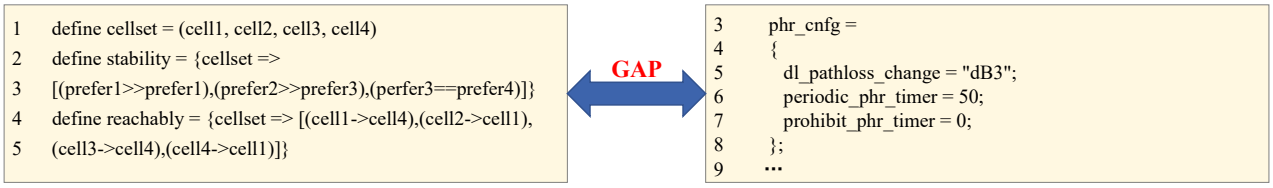
In this section, we systematically introduce the overall framework of Drone and present two important designs in the Drone framework: 1) to address the problem of low level of configuration abstraction, we propose a configuration abstraction layer and design an intent representation language and corresponding compiler (Section 3.1), and 2) to address the problem of lack of configuration semantics, we propose a configuration behavior program modeling (Section 3.2). Figure 3 shows an overview of Drone, and this work addresses the automatic generation of cold start configurations for newly added devices and the configuration updates according to dynamically changing scenario requirements.

A. Overview of Drones

Drone is an intent-driven cellular network configuration synthesis system. Its input is the operator's high-level management intent (Figure 3 (a)), helping the operator to flexibly manage the network configuration. At the same time, the input of the system also includes network configuration scenarios (Figure 3 (b)), including the cold start of newly added devices, configuration update of existing devices, network topologies, and parameter value ranges of network configuration (Figure 3 (c)), the value ranges of the parameters are used to add constraints to the solver.

Drone contains an abstraction layer, which is mainly composed of a compiler that can parse the input high-level intent language (Figure 3 (d)), and the compiler converts the intent language into an intermediate representation that can be flexibly manipulated (Figure 3 (e)).

The synthesis part of Drone includes three key parts, the first is general program modeling (Figure 3 (f)), the configuration is modeled as a SKETCH program with a higher semantic level and context, and the configuration parameters are modeled as key nodes ('holes' in the program). The second critical step establishes the program objectives of the configuration semantic program as formal constraints (Figure 3 (g)). Finally, the formal constraints are solved by a counterexample-guided inductive synthesis algorithm (Figure 3 (h)). Putting the solved configuration parameters into the configuration template is the final output configuration file of the system (Figure 3 (i)).



a) high-level intent

b) low-level configuration

Fig. 4. The gap between high-level intent and low-level configuration.

def	::=	*prim* **id** = *stmt*	*definition*
prim	::=	**define**	*primitive*
stmt	::=	*cells*	*cell set*
	|	*cons*	*constraint*
cells	::=	**(name$^+$)**	*cell name*
cons	::=	$\{pol^+\}$	*policy*
pol	::=	**name** *op* **name**	*preference*
op	::=	$\gg$	*high*
	|	==	*equal*
	|	$\ll$	*low*
	|	$\rightarrow$	*target*

Fig. 5. The syntactic paradigm of the intent language in the Drone abstraction layer.

B. Intent Language

Figure 4 shows the comparison between the high-level intent language and the low-level configuration. It can be seen that the high-level intent language has a structured format and complete context, so it has higher-level semantics and is easier for operators to understand. The configuration file only contains the assignment relationship between parameter names and parameter values, and it is not easy to see the dependencies between parameters, and the semantic level is low. For this purpose, we designed the CEL language, an intent-based policy description language for cellular networks. At the same time, we designed a series of primitives to support the expression of cellular network policies to customize the different management needs of network operators. Network operators only need to design high-level intents and don't have to care about how the intents are implemented. And there is no need to understand complex low-level configuration. So they can easily deploy a network that meets performance expectations. Figure 5 illustrates the syntactic paradigm of Drone.

We designed the CEL language from scratch, including designing its syntax paradigm and developing a language compiler, making it a declarative domain-specific language (DSL). The language implemented in this way is also called an external DSL. In contrast, the internal DSL is implemented by embedding an existing language. Generally speaking, the internal DSL can help operators quickly implement applications and development, but the language implemented by it lacks flexibility and cannot be directly analyzed for correctness. And our custom compiler can ensure the correctness of the written DSL through static analysis technology, and enhance the flexibility of the DSL through the custom-designed syntax.

Policy Definition: The core abstraction of Drone for configuration is to describe globally uniform high-level deployment policies through an intuitive syntax. Each policy consists of four parts, including (1) the definition primitive (2) the policy name (3) the deployment object (4) the policy body.

define *stability* $=\{cellset \Rightarrow [(prefer1 \gg prefer2), (prefer2 \ll prefer3), (prefer==prefer4), (prefer4 \ll prefer1)]\}$

Since we need to verify program objectives in the program synthesis algorithm part (Section 4.3), the program objectives must converge. This makes it necessary to add network optimization constraints to the intent. Our intent constraints include stability constraints, convergence optimization constraints, etc.

The stability policy is proposed for the convergence problem in the cell handover function. The operator needs to define the priority relationship of each cell. The priority in the intent is globally unified, which is the basic guarantee of handover stability.

The convergence optimization policy needs to point out the optimal target cell expected in the handover decision when each cell serves as the serving cell.

define *convergence* $= \{cellset \Rightarrow [(cell1 \rightarrow cell4), (cell2 \rightarrow cell3), (cell3 \rightarrow cell4), (cell4 \rightarrow cell1)]\}$

The intent language defines the serving cell and the target cell that is expected to be converged during the execution of the handover, such that there are no two types of problems in which the handover does not converge to the desired target cell: 1) due to lack of configuration, there is no path from the serving cell to the target cell where convergence is expected, and 2) there is an inconsistency between the first-come-first-served (FCFS) response set by the operator and the measurement order of the device. This leads to a situation where early convergence prevents target handover decisions [24].

Define the set of cells to deploy the policy: In addition to defining the policy, the definition primitive can also define

the set of deployment objects, i.e., the set of cells.

define *cellset = (cell1, cell2, cell3, cell4)*

The definition of cell collection is a parameter aggregation definition, which aggregates the basic parameters of cells, including static attributes such as operating band, center frequency, physical cell ID (PCI), public land mobile network (PLMN), mobile country code (Mcc), mobile network code (Mnc), etc. This definition can realize the automatic generation of static parameters when triggered. Among them, the PCI assignment has a global perspective, which can avoid the problem of PCI assignment conflicts. The definition also reflects the abstraction of the intent language, which can configure the cells in the set to be neighbor cells to each other and configure the handover rules and neighbor cell parameters by simply putting the cells into the same set.

Parameter aggregation and abstract definition greatly improve configuration efficiency, reduce the work difficulty of network operation and maintenance personnel, and free them from the complicated and cumbersome process of mapping configuration manuals. The configuration process can avoid configuration errors caused by human errors and reduce the occurrence of catastrophic downtime events.

Intermediate representation: CEL models the operator input intent as a syntax tree, which is encoded as the basis for reasoning about constraints during program synthesis.

C. Configuration Semantic Modeling

Unlike traditional networks, cellular network configuration parameters lack configuration semantics and are difficult to model as high-level abstractions represented through data structures (cost values in traditional network OSPF protocols are modeled as weights on edges in a route propagation graph). Therefore, we propose a program modeling approach for cellular networks to increase the level of abstraction of low-level configuration parameters. This approach models the cellular network functions with configuration behavior as a program to be filled, called SKETCH. A SKETCH program is a program specification with "holes" (i.e., unknown variables in the program), and we model configuration parameters as "holes". We fill the "holes" by solving for the accurate values so that the program satisfies the target constraints.

Handover logic SKETCH program modeling: The priority-based handover logic SKETCH program is shown in Figure 6. This program performs the handover decision by comparing the service cell with each eligible candidate cell among neighbor cells. There are two types of configurable parameters for the serving cell and each neighboring cell, which are the priority of the cell and the signal quality threshold for the handover decision.

The verdict of priority in the SKETCH program is input into Drone by the network operator by writing the intent. Then we need to solve the filling problem of the unknown variable (e.g., the (a) in Figure 6) based on the semantics. The filling problem is modeled by us as an auto-completion synthesis problem based on the desired objectives of the program. The unknown variables in the SKETCH program simulate the signal quality

```
1   for cell in neighbour_list :
2       if serv_cell ['prefer'] < cell ['prefer'] :
3           if meas_c > (a) :
4               target_cell_list.append (cell)
5       if serv_cell ['prefer'] == cell ['prefer'] :
6           if meas_c > meas_serv + (b) :
7               target_cell_list.append (cell)
8       if serv_cell ['prefer'] > cell ['prefer']:
9           if (meas_c > (c)) & (meas_serv < (d)) :
10              target_cell_list.append (cell)
```

Fig. 6. SKETCH program modeling of priority-based inter-frequency handover behavior.

thresholds for handovers at different priority levels. The event-based measurement logic is similar to it and can be easily extended.

IV. SKETCH Program Synthesis

We use a counterexample-guided inductive synthesis algorithm to synthesize the "holes" to be filled in the SKETCH program. We remodel the parameter generation problem as a program synthesis problem and formally model the program objective instead of directly modeling complex parameters, which reduces the difficulty of modeling and solving. However, the program objectives transformed from the basic intent lack constraints. Therefore, we survey existing advanced cellular network measurement efforts [22], [25]. Among them, Li et al. proposed a very comprehensive study on cellular network reliability analysis [25]. We refer to these efforts and summarize the existing problems and potential demand scenarios in current cellular networks, and design network optimization constraints to add to the intent. Then, we designed the constraints for the counter-example verification part. We design constraints on the quality of handover convergence for the handover problem and prove the feasibility of the constraints. Meanwhile, we build an abstract model to simulate the handover decision. This section will introduce the constraint modeling process and the program synthesis algorithm flow in detail.

A. SMT Encoding

First, we formulate the program auto-completion synthesis problem as a constraint-solving problem. We encode the priority term and the target path term in the intent as constraints in satisfiability modulo theory (SMT) and then encode the holes in the SKETCH program as variables to be solved. The specific representation is as follows:

Handover convergence quality constraint: The handover convergence quality problem includes handover convergence

stability, handover convergence target reachability, and handover convergence speed, as follows:

1) Handover convergence stability: The research by Li et al. [21] reveals the handover misconfiguration problem in cellular networks. This problem leads to a kind of persistent cyclic handover, such as handover loops and handover oscillations. This often occurs when the cell handover configuration is not coordinated. For the handover instability problem, we construct the following formal constraints [21]:

$$\neg \begin{pmatrix} Nor(c_1 \to c2 \to \cdots \to c_n) \Rightarrow \\ (\exists i \in N, Pref(c_{i,i+1} > c_{i,i})) \wedge \\ (\forall j \in N, Pref(c_{j,j+1} \geqslant c_{j,j}) \wedge \\ Pref(c_{n,n} \geqslant c_{n,1})) \end{pmatrix} \tag{6}$$

$$if\ Prefer_{c_i} > Prefer_{c_j}: \quad min_{c_i \to c_k}(\theta_{i,k}^{high}) \geqslant \theta_j^{serv} \tag{7}$$

$$else\ if\ Prefer_{c_i} < Prefer_{c_j}: \quad min_{c_j \to c_k}(\theta_{j,k}^{high}) \geq \theta_i^{serv} \tag{8}$$

$$else: \theta_{i,j}^{eq} + \theta_{j,i}^{eq} \geqslant 0 \tag{9}$$

Constraint (6) represents the cell priority constraint that needs to be satisfied to avoid handover loops. Hard constraints (7)-(9) represent the uncoordinated RSRP threshold avoidance constraints at different priority levels when the cell priority level satisfies global coordination.

Proof of constraints: Taking the stability constraints of two cells as an example, when the signal quality of the serving cell is lower than its pre-configured threshold, the serving cell enters the handover process. If the priority of the serving cell $Cell_i$ is higher than the measured neighbor cell $Cell_j$, the handover logic taken at this time should be low-priority handover: $meas_i < \theta_i^{serv} \cap meas_j > \theta_{i,j}^{low}$. We assume that the fluctuation of the signal quality measurement value of the source cell $Cell_i$ before and after the handover is negligible, to make the handover converge rather than generate oscillations, $meas_i$(the serving cell signal quality measurements) should not be greater than the high priority handover threshold θ_i^{high} configured for $Cell_j$. No matter what value $meas_i$ takes, it should not be in the interval $[\theta_{j,i}^{high}, \theta_i^{serv}]$, ie $[\theta_{j,i}^{high}, \theta_i^{serv}] = \emptyset$, so it follows that $\theta_{j,i}^{high} \geqslant \theta_i^{serv}$. The remaining constraints are proved in the same way.

2) Handover convergence target reachability: The problem of suboptimal handover convergence quality occurs when the converging cell is not the best target cell. Our common sense is that the cell for handover convergence should be the cell that can provide the best signal-quality service. The target cell is evaluated by the operator in advance through the customized evaluation metrics and input in the intent section.

3)Handover convergence speed: A large amount of control signaling generated by frequent handover will cause serious channel overhead, which will increase network latency, jitter, and packet loss rate, resulting in the degradation of network service. Therefore, to minimize the number of handovers from the initial serving cell to the converging cell, we need to construct soft constraints to limit the number of handovers.

B. Handover Path Modeling

We first model the handover between the initial cell and the converged target cell in the stability constraint as the shortest path problem (shown in Figure 7). Then we model the cell priority and handover threshold as path weights. Finally, we model the calculation process of the distributed, destination-based hop-by-hop handover path as a centralized handover path calculation with continuity between the initial serving cell and the desired target cell. The premise of our model is the low-speed handover scenario where the channel environment and user coordinates change slightly when the handover occurs, i.e. the execution of the handover is instantaneous.

Finding the shortest path in our model is simple and should be instructive by the fact that the smaller the number of handovers the better. Our optimization goal is to set the handover threshold and priority values (the weights of the edges in the modeled handover graph) to make the number of handovers reach the target cell as little as possible. However, there is a limitation of this strategy: there cannot be a situation where it is impossible to handover to an undesired target cell even when the signal quality of the desired target cell is very poor. At this time, the system should then make an alternative plan in time to select one of the other candidates with good signal quality as the handover target cell. In simple terms, the intent is to set a tolerance range for signal quality.

The cell coverage is abstracted into the form of a cell topology, the cell handover problem is transformed into the shortest path roaming problem for the cell topology, and the paths are pruned using stability constraints.

The stability constraint allows the handover graph to be pruned, and the remaining possible handover paths after pruning are $(n-1)!$. We need to set the optimal path as the shortest path by setting the weight on each edge. After calculating the weights a backward derivation is performed to derive the corresponding handover threshold parameters.

We introduce the integer variable $T_{i,j}$ to denote the handover weight from $Cell_i$ to $Cell_j$. The handover weight of the whole path is the sum of the handover weights of the individual links along the path. For example, the handover path shown in Figure 7(c), the handover weight of the path $Cell_i \to Cell_l \to Cell_k \to Cell_j$ can be expressed as, $Threshold(i \to l \to k \to j)$, which is equal to $T_{i,k}+T_{k,j}$. We use $Path(i,j)$ to denote the set of all handover paths between two cells. We can encode the path $Cell_i \to Cell_l \to Cell_k \to Cell_j$ to have the minimum handover weight among all other paths by constraining it as follows:

$$\begin{aligned} &\forall X \in Path(i,j) \backslash \\ &S.Threshold(i \to l \to k \to j) < Threshold(X), \\ &where\ S = \{i \to j, i \to k \to j, i \to l \to j, \\ &i \to k \to l \to j, i \to l \to k \to j\} \end{aligned} \tag{10}$$

The coding intent corresponding to this constraint is the convergence optimization policy mentioned in Section 3.1. The complete convergence optimization policy captures the following constraints: 1) each ($Celli \to Cellj$) represents the best handover target cell for the serving cell, i.e. its handover weight is lower than all other handover paths, 2) the overall

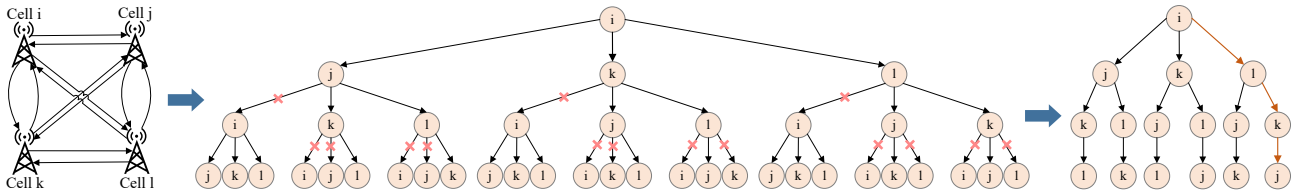


Fig. 7. Modeling of shortest handover paths. (a) is the network base station topology, (b) is all the paths where handover may occur, (c) is the handover path after pruning with stability constraints, and the orange path is the optimal handover path.

handover still maintains a part of the distributed properties, so that the entire convergence optimization constraint can be connected into a handover path, and the connection path has the lowest handover weight sum of all paths (which is certain since it is the weighted sum of all desired handover).

C. Program Synthesis Algorithm

Forward solving: This approach requires all paths to be encoded and then the correct path weights are found using a constraint solver. The solution space of this solving model is too large to be extended to large networks.

For this reason, this paper adopts the inductive synthesis algorithm guided by counterexample as our program synthesis algorithm (as shown in Figure 8). The main idea of the CEGIS algorithm is to verify whether the constraints are satisfied by sampling conditions and to design a set of counterexamples to guide the synthesis. In practice, the counterexamples tend to be small, which means that a small number of iterations are needed to find the correct solution, and this method tends to have better performance.

The CEGIS algorithm, given a set of cell topologies and a set of target cell convergence intents, outputs handover signal quality thresholds that enforce these requirements (subject to satisfying different cell priorities).

The idea of program verification is introduced in the counter-example-guided induction synthesis:

First, a SKETCH program with unknown variables (holes) is given, and a candidate SKETCH is generated by synthetically filling the unknown variables through the automatic solution engine. Then, verify the consistency of intent and program objectives (POs). The objective function is the consistency function $Consistency(INTENT, POs)$. Third, the problem to be verified is encoded as SMT logic formulas (constraints). Finally, the logic formula is proven to be correct or not. If it is verified to be incorrect, a counterexample is generated, and added to the counterexample set, leading to the synthesis of the automatic solver, and the cycle is repeated until a satisfiable solution is found (if any). Handover convergent optimization problem amounts to finding a logically constrained model of the form:

$$\exists T.ENCODEHANDOFF(T, I, Path(I)) \tag{11}$$

where T is the handover link weight, I is the intent, and $Path(I)$ is the set of all paths between the source cell and the target cell provided in the intent. This formal constraint returns a logical formula that encodes the satisfiability of the intent.

Algorithm 1: CEGIS algorithm for synthesizing handover path threshold weights concerning convergence target intent.

input : Convergence Target Intent $Intent = \cup_i p_i$, link weight variables T, path number n
output: Shortest handoff link weight

for $p \in Intent$ **do**
  $S_p = \emptyset$
while $true$ **do**
  $\varphi = true$
  for $p \in Intent$ **do**
    $S_p \leftarrow S_p \cup$ SAMPLEPATHS(p, n)
    $\varphi_p =$ ENCODEHANDOFF$(T, I, Path(I))$
    $\varphi = \varphi \wedge \varphi_p$
  if *UNSAT(φ)* **then**
    return $false$
  $M \leftarrow MODEL(\varphi)$
  if *CHECKINTENT(M, $Intent$)* **then**
    return M(T)
  $(p, path) \leftarrow$ COUNTEREXAPLE$(M, Intent)$
  $S_p \leftarrow S_p \cup \{path\}$

Algorithm 1 shows the main steps of counterexample-guided inductive synthesis [8] in solving the threshold weight on the shortest handover path. First, for each expected handover path in the intention $p \in Intent$, we define a convergence target set S_p. Then the algorithm enters a counterexample iterative cycle. For the desired handover path p, the algorithm samples n paths from the source to the target, adds them to the set S_p and then encodes S_p to generate constraint φ, returning "False" if it is unsatisfied, which means that the intent cannot be satisfied. On the contrary, if it is satisfied, the constraint model $Model$ is obtained, which includes weights on each link. Then verify whether the model satisfies the intent and if so, return $Model(T)$, i.e. the weights on all paths. Otherwise, counterexamples are generated and added to the set of counterexamples, which ensures that the counterexample is avoided in the next iteration to guide the synthesis of the automatic solver. These steps are repeated in a loop until either a satisfiable solution is found (if any) or the intent is

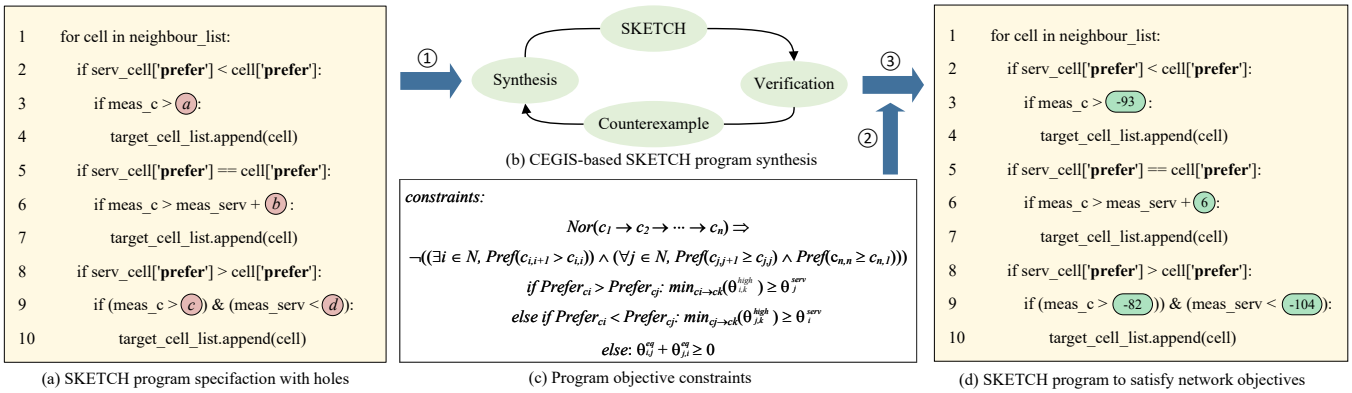


Fig. 8. The counterexample-guided inductive synthesis algorithm solution process. The input is: (a) a SKETCH program with "holes". The CEGIS algorithm solution process is (b), where SKETCH is first input to the synthesis system, and then (c) constraint verification is performed. If the verification synthesis result does not satisfy the constraint, counterexamples are generated and iterated until (d), a program that satisfies the network intent management objective, is generated.

considered unsatisfiable.

D. Domain-specific Optimization Algorithms

The topological complexity increases as the network size grows. We can calculate the number of handover paths under full base station coverage as $\sum_{i=0}^{N-2} A_{N-2}^{i}$, where N denotes the total number of network nodes. Therefore when the scale of the network topology nodes is large, the number of paths involved in the synthesis will be very large. Due to our initial random generation method, such large-scale paths can seriously affect the efficiency of the parameter search, and the results cannot be synthesized even for more than one day at larger network sizes. We can reduce this effect by pruning the parameter search space. According to actual operation and maintenance experience, base stations with similar geographic locations or relatively close distances also maintain similar operating configuration parameters (parameter values are close). First, we propose a geographic location similarity-based algorithm. We rank the base stations according to the relevance of their static attribute parameters and elect the set of base stations with the highest similarity to the new base stations. Then we extract the maximum and minimum values of each base station parameter from the elected similar base stations, replace the search range in the solution model, and complete the pruning of the synthesis search space. The algorithm flow is shown in Algorithm 2.

Algorithm 2: Geographical Similarity Pruning Algorithm

input : Collection of Base Stations BS, Base station configuration parameter set P
output: Range of parameters after pruning $[S_{min}, S_{max}]$

for $p \in P$ **do**
  for $b \in BS$ **do**
    $max_p = max(max_p, p_b)$
    $min_p = min(min_p, p_b)$
  $S_{max} \leftarrow S_{max} \cup max_p$
  $S_{min} \leftarrow S_{min} \cup min_p$

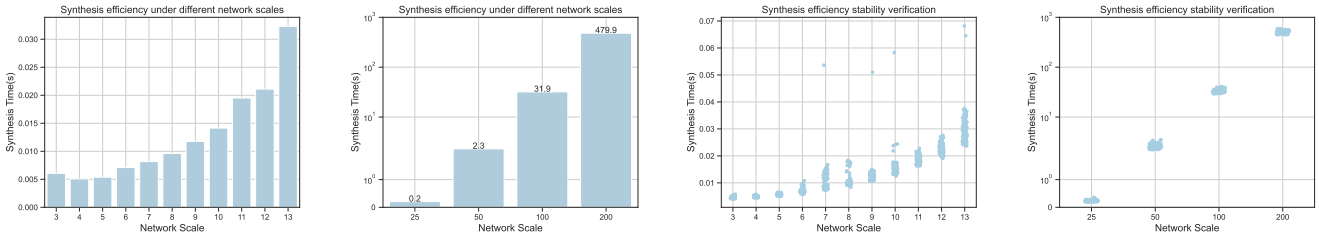
V. Experimental Evaluation

A. Program Synthesis Time at Different Topology Sizes

We design a synthesis efficiency experiment at different network sizes, simulating an open scenario with few base stations (e.g., suburban areas far from the city) and a scenario with dense deployment of base stations with high human traffic (e.g., transportation hubs, urban commercial centers, etc.). Our parameters are initially set using a random Dijkstra path generation scheme, with weights simulating the gap between the handover threshold and the signal strength of different UEs, and the random generation scheme is used because of the random nature of user mobility.

Our experimental results prove that the synthesis efficiency of the system is very high, and the parameter synthesis can be completed within 50ms for all small-scale (less than 13 base station nodes) scale scenarios. Figure 9(a) and 9(b) show the results of our synthesis efficiency experiments. When we expand the size of the base station node to 200 base stations, the average synthesis time is still only 500.7s. It can be concluded that the configuration efficiency of our Drone system for large-scale networks is extremely high, which greatly reduces the tedious and repeated operation of network operation and maintenance personnel, and can greatly reduce the risk caused by human configuration errors.

At the same time, to prove that our experimental results were not accidental, we conducted system stability experiments with more than one hundred experiments at different scales. The results are shown in Figure 9(c) and 9(d), which demonstrate the stability of our system and again prove our extremely high synthesis efficiency, with synthesis times of up to 1s for small-scale scenarios and up to 10min for large-scale scenarios. Then we calculated the mean value and confidence interval of the synthesis time of the more than one hundred stability experiments under different topologies, and the experimental results are shown in Figure 10.



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

Fig. 9. Experimental results for synthesis time across various network scales and synthesis stability at different network topology scales, with over one hundred experiments conducted at each scale.

TABLE I
COMPARING THE SYNTHESIS TIME WITH AND WITHOUT THE OPTIMIZATION ALGORITHM

network scale	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
with optimization	0.006s	0.005s	0.005s	0.007s	0.008s	0.009s	0.014s	0.016s	0.022s	0.025s	0.031s
w/o optimization	0.023s	0.023s	0.022s	0.134s	3.88s	>24h	>24h	>24h	>24h	>24h	>24h

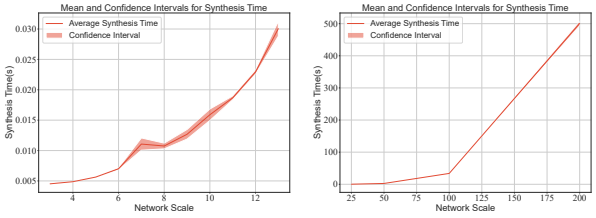


Fig. 10. Experimental results of the mean value and confidence interval of configuration synthesis time at different network topology scales.

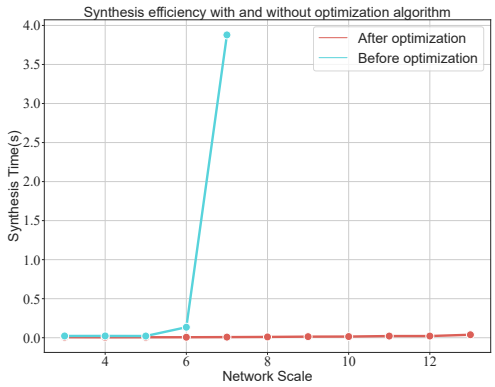


Fig. 11. The results of the experiments compare the synthesis efficiency by adding the optimization algorithm and not adding the optimization algorithm.

B. Optimized Algorithm Verification Experiment

To verify the effectiveness of our proposed optimization algorithm, we compare the basic CEGIS algorithm with the CEGIS algorithm after adding the optimization algorithm for experiments. The experimental results show that the CEGIS algorithm without adding the optimization has a larger parameter search space and can cause a serious impact of random factors on the path search during the initial path generation, which often fails to synthesize the results for more than one day when the network size is large.

From Table 1, we can calculate that the synthesis efficiency of the method is significantly improved by adding the optimization algorithm to the small-scale network (3 to 7 base station nodes), and the synthesis speed is improved by 3.83x to 485x, respectively, with an average improvement of 97.50x. The method without optimization cannot solve the results in less than 24h for more than 7 base station nodes. It can be more intuitively seen from Figure 11 that the optimization algorithm brings significant performance improvement to the CEGIS method.

VI. FUTURE WORK

1) Online parameter optimization: This work addresses the issue of cold start configuration deployment for newly added devices. Future online partial tuning of cellular parameters is a direction worth exploring, such as incremental configuration updates. Dynamic, sequential, and runtime updates will be serious challenges.

2) Functional expansion: This work currently only models configuration semantics programs for a few basic functional modules. Future expansion to full-featured program modeling would be a very interesting task, such as extending to uplink power control in radio resource management, cellular networks security features, etc. This work aims to propose a new approach to solve complex cellular network parameter configuration problems, so the above function expansion will be our main research work in the future.

VII. CONCLUSION

We present Drone, the first work extending intent-driven configuration synthesis to cellular networks. Drone introduces a novel general methodology for modeling and solving the cellular network configuration problem using program synthesis techniques. This work provides a detailed semantic modeling scheme for cellular network function configurations. Drone encodes program objectives into SMT-based constraints

and proposes an efficient counterexample-guided inductive synthesis algorithm. Compared to existing solutions, Drone offers an intent syntax enabling operators to customize high-level management goals and addresses modeling challenges for low-level parameters with complex dependencies. We demonstrate the feasibility and effectiveness of Drone, including the optimization algorithm, through extensive experiments. Overall, Drone pioneers intent-driven configuration synthesis in cellular networks via principled program synthesis.

ACKNOWLEDGMENT

We would like to express our gratitude to the editors and reviewers of the IEEE JSAC special issue on Human-Centric Communication and Networking for Metaverse over 5G and Beyond Networks for their review work and valuable feedback. This work is supported by the National Natural Science Foundation of China under Grant Nos. 62072091, 62032013, U22B2005, and 92267206, and the financial support of Lingnan University (LU) (DB23A9) and Lam Woo Research Fund at LU (871236).

REFERENCES

[1] N. H. Chu, D. T. Hoang, D. N. Nguyen, K. T. Phan, and E. Dutkiewicz, "Metaslicing: A novel resource allocation framework for metaverse," *arXiv preprint arXiv:2205.11087*, 2022.

[2] A. Mahimkar, A. Sivakumar, Z. Ge, S. Pathak, and K. Biswas, "Auric: using data-driven recommendation to automatically generate cellular configuration," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, 2021, pp. 807–820.

[3] A. Mahimkar, Z. Ge, X. Liu, Y. Shaqalle, Y. Xiang, J. Yates, S. Pathak, and R. Reichel, "Aurora: conformity-based configuration recommendation to improve lte/5g service," in *Proceedings of the 22nd ACM Internet Measurement Conference*, 2022, pp. 83–97.

[4] J. Chuai, Z. Chen, G. Liu, X. Guo, X. Wang, X. Liu, C. Zhu, and F. Shen, "A collaborative learning based approach for parameter configuration of cellular networks," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 1396–1404.

[5] K. Subramanian, L. D'Antoni, and A. Akella, "Synthesis of fault-tolerant distributed router configurations," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 2, no. 1, pp. 1–26, 2018.

[6] A. Abhashkumar, A. Gember-Jacobson, and A. Akella, "Aed: Incrementally synthesizing policy-compliant and manageable configurations," in *Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies*, 2020, pp. 482–495.

[7] A. El-Hassany, P. Tsankov, L. Vanbever, and M. Vechev, "Network-wide configuration synthesis," in *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part II 30*. Springer, 2017, pp. 261–281.

[8] El-Hassany, Ahmed and Tsankov, Petar and Vanbever, Laurent and Vechev, Martin, "Netcomplete: Practical network-wide configuration synthesis with autocompletion," in *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*, 2018, pp. 579–594.

[9] R. Beckett, R. Mahajan, T. Millstein, J. Padhye, and D. Walker, "Don't mind the gap: Bridging network-wide objectives and device-level configurations," in *Proceedings of the 2016 ACM SIGCOMM Conference*, 2016, pp. 328–341.

[10] Beckett, Ryan and Mahajan, Ratul and Millstein, Todd and Padhye, Jitendra and Walker, David, "Network configuration synthesis with abstract topologies," in *Proceedings of the 38th ACM SIGPLAN conference on programming language design and implementation*, 2017, pp. 437–451.

[11] R. Birkner, D. Drachsler-Cohen, L. Vanbever, and M. T. Vechev, "Config2spec: Mining network specifications from network configurations." in *NSDI*, 2020, pp. 969–984.

[12] K. Subramanian, L. D'Antoni, and A. Akella, "Genesis: Synthesizing forwarding tables in multi-tenant networks," in *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*, 2017, pp. 572–585.

[13] J. Sonchack, D. Loehr, J. Rexford, and D. Walker, "Lucid: A language for control in the data plane," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, 2021, pp. 731–747.

[14] X. Gao, T. Kim, M. D. Wong, D. Raghunathan, A. K. Varma, P. G. Kannan, A. Sivaraman, S. Narayana, and A. Gupta, "Switch code generation using program synthesis," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 2020, pp. 44–61.

[15] H. Zhao, B. Yang, J. Cui, Q. Xing, J. Shen, F. Zhu, and J. Cao, "Effective fault scenario identification for communication networks via knowledge-enhanced graph neural networks," *IEEE Transactions on Mobile Computing*, 2023.

[16] H. Dai, X. Wang, X. Lin, R. Gu, S. Shi, Y. Liu, W. Dou, and G. Chen, "Placing wireless chargers with limited mobility," *IEEE Transactions on Mobile Computing*, 2021.

[17] B. Tian, X. Zhang, E. Zhai, H. H. Liu, Q. Ye, C. Wang, X. Wu, Z. Ji, Y. Sang, M. Zhang, *et al.*, "Safely and automatically updating in-network acl configurations with intent language," in *Proceedings of the ACM Special Interest Group on Data Communication*, 2019, pp. 214–226.

[18] C. Prakash, J. Lee, Y. Turner, J.-M. Kang, A. Akella, S. Banerjee, C. Clark, Y. Ma, P. Sharma, and Y. Zhang, "Pga: Using graphs to express and automatically reconcile network policies," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 29–42, 2015.

[19] A. Solar-Lezama, L. Tancau, R. Bodik, S. Seshia, and V. Saraswat, "Combinatorial sketching for finite programs," in *Proceedings of the 12th international conference on Architectural support for programming languages and operating systems*, 2006, pp. 404–415.

[20] A. Solar-Lezama, C. G. Jones, and R. Bodik, "Sketching concurrent data structures," in *Proceedings of the 29th ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2008, pp. 136–148.

[21] Y. Li, H. Deng, J. Li, C. Peng, and S. Lu, "Instability in distributed mobility management: Revisiting configuration management in 3g/4g mobile networks," in *Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science*, 2016, pp. 261–272.

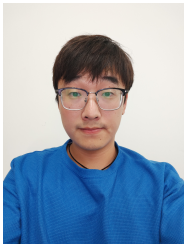
[22] A. Hassan, A. Narayanan, A. Zhang, W. Ye, R. Zhu, S. Jin, J. Carpenter, Z. M. Mao, F. Qian, and Z.-L. Zhang, "Vivisecting mobility management in 5g cellular networks," in *Proceedings of the ACM SIGCOMM 2022 Conference*, 2022, pp. 86–100.

[23] Q. Li and C. Peng, "Reconfiguring cell selection in 4g/5g networks," in *2021 IEEE 29th International Conference on Network Protocols (ICNP)*. IEEE, 2021, pp. 1–11.

[24] C. Peng and Y. Li, "Demystify undesired handoff in cellular networks," in *2016 25th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2016, pp. 1–9.

[25] Y. Li, H. Lin, Z. Li, Y. Liu, F. Qian, L. Gong, X. Xin, and T. Xu, "A nationwide study on cellular reliability: measurement, analysis, and enhancements," in *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, 2021, pp. 597–609.

Fuliang Li (Member, IEEE) received the BSc degree in computer science from Northeastern University, China in 2009, and the PhD degree in computer science from the Tsinghua University, China in 2015. He is currently an associate professor at the School of Computer Science and Engineering, Northeastern University, China. He has published more than 50 Journal/conference papers. His research interests include network management and measurement, cloud computing, and network security.



Chenyang Hei received his BS degree in Intelligence Science and Technology from Northeast Electric Power University, Jilin, China in 2021, and is currently pursuing a Ph.D. degree at the School of Computer Science and Technology, Northeastern University, Shenyang, China. His research interests include network configuration synthesis, SmartNIC, and distributed training networks.

Qing Li is pursuing the M.S. degree in computer science and technology with the Department of Computer Science and Engineering, Northeastern University, Shenyang. She received the B.S. degree in computer science and technology from Northeastern University (Qinhuangdao) in 2021. Her research interests include network management, cellular network configuration, and intent-based network.

Jiaxing Shen (Member, IEEE) received the B.E. degree in Software Engineering from Jilin University in 2014, and the Ph.D. degree in Computer Science from the Hong Kong Polytechnic University in 2019. He was a visiting scholar at the Media Lab, Massachusetts Institute of Technology in 2017. His research interests include mobile computing, data mining, and IoT systems. He has published in various top-tier journals including IEEE TMC, ACM TOIS, ACM IMWUT, and IEEE TKDE. He has received two best paper awards at leading conferences, including IEEE INFOCOM.

Xingwei Wang (Member, IEEE) received the BS, MS, and PhD degrees in computer science from Northeastern University, Shenyang, China, in 1989, 1992, and 1998, respectively. He is currently a professor with the College of Computer Science and Engineering, Northeastern University, Shenyang, China. He has authored or coauthored more than 100 journal articles, books and book chapters, and refereed conference papers. His research interests include cloud computing and future Internet. He was the recipient of several best paper awards.