Design of a general complex problem-solving architecture based on task management and predictive optimization

Shabir Ahmad\(^1\), Salman Khan\(^2\), Faisal Jamil\(^1\), Faiza Qayyum\(^3\), Abid Ali\(^4\) and DoHyeun Kim\(^3\)

Abstract

Many real-life problems have different contradicting goals and no simple solution. Therefore, an analysis is made to select the appropriate solution based on the scenario, which is considered the best compromise toward the achievement of a goal. In literature, it is known as complex problem-solving and is a kind of paradigm that has been around since the last century, but the cognition involved in complex problem-solving has purely relied on experts in the field. However, with the evolution of the current stack of technologies such as artificial intelligence and the Internet of Things, it is quite possible to perform the cognition process with the help of machines based on the previously-trained historical data. Our previous work proposed the complex problem-solving as a service for smart cities. In this article, we extend this work and propose a generic architecture for complex problem-solving using task orchestration and predictive optimization in Internet of Things–enabled generic smart space. The proposed framework makes use of historical data for artificial cognition of the complexity of the given problem. For this, predictive optimization is used, which identifies the problem and intelligently predict the solution based on the given constraints. The task orchestration architecture is used to decompose the complex problem into small tasks for real-world deployment into sensors and actuators. The architecture is evaluated against different load conditions and different categories of problems, and the results suggest that the proposed architecture can be used a commonplace for different smart space solutions.

Keywords

Complex problem-solving, task modeling, embedded Internet of Things systems, predictive optimization, artificial cognition, task orchestration

Date received: 5 November 2020; accepted: 25 April 2022

Handling editor: Peio Lopez Iturri

Introduction

Internet of Things (IoT)\(^1\) has enabled the connectivity of huge number of miniature devices using wide-range technologies. IoT is among the leading technologies since its inception and became a buzzword.\(^1,4\) Nowadays, application domains like smart healthcare, smart home, and smart grid, would not be realized without the IoT. The IoT architecture, however, has evolved with the passage of time. Over the past few
years, the architecture of IoT systems is cloud based. The sensors send context data to the cloud for permanent storage, processing, and analytics. However, due to growing demand of rapid response time and computation, the services are moved closer to devices to boost up the performance. The geographically distributed edge nodes provide several advantages like response time and reliability.

Modern world application requires real-time processing of data generated from IoT nodes (sensors) to make intelligent decision in a complex scenario. A phenomenon in which an agent (computer program or human expert) picks an appropriate path from different alternative part is known as complex problem-solving (CPS), and the problem in hand is often referred to as complex problem (CP). With the advancement of IoT and artificial intelligence, there should be an autonomous mechanism to provide an efficient and accurate solution for CPs that consider learning from incidents of previous statistics and predicts future forecasting. In literature, there are numerous instances of investigating complex problem-solutions based on a single problem having various contradicting solutions. In the modern IoT world, due to the abundance of context, data generated from a wide array of sensors have different dimensions of the problem. Therefore, a more robust architecture based on more than a single parameter to find a solution for a given CP needs investigation, and thus this article is the first step to approach the multi-parameter CPS in an IoT environment. Figure 1 shows the existing solutions and improved solutions.

In our previous studies, we proposed a specific case of smart cities and developed an architecture to propose the complex problem-solution as a service (CPSaaS). However, the framework was specific and covers only problems relevant to smart cities. This study extends our previous architecture and provides in-depth analysis of state-of-the-art approaches for solving CPs not only in smart cities but also in general IoT space. Furthermore, the use of edge platform can make the applications efficient, and the use of optimization and learning modules can make the application intelligent.

In this regard, the architecture aims to support predictive, optimized, controlled, and scheduling-based algorithms to be deployed as a microservice on edge computing platforms in order to solve real-life CPs for bringing more values to human lives. The ultimate goal of the proposed work is to provide a modular architecture of loosely coupled and highly cohesive generic open-source, which supports scheduling algorithms, control, optimization, and prediction. The proposed architecture is an edge-compliant that supports different services and microservices for real-time applications in edge computing environment. This architecture also support as a bare bone platform for implementing different use cases such as smart healthcare, smart farm, and smart grid.

The proposed architecture supports different services which are needed for human comfort. In reality, the stakeholders of this architecture are the technology experts and business enthusiast who eventually strive to add comfort and value in the lives of human being. Another goal is to make the system available so that improvements can be welcomed from anyone around the globe, making it a collaborative space for technologist and researchers.

In this article, we model the above-mentioned problems and lay them out in the proposed generic CPS platform for various solutions. As part of the CP modeling, we add the problem along with constraints, prediction goals, and optimization goals. The jobs such as energy prediction and load demand prediction can be solved using prediction solvers, whereas optimization solver does the job of optimal energy management with available resources.

The rest of the article is organized as follows. Section “Related works” highlights the state-of-the-art relevant to this work. Section “Design architecture of proposed HCPSA” presents the general model of the system. Section “Development model of HCPSA” presents the objective formulation for different optimization problem. Section “Implementation of HCPSA” covers the implementation of the architecture and highlights the working of the system with snapshots of different
modules. Section “Results” presents performance comparisons of the system, and section “Conclusion” concludes the article and draws future work.

Related works

Complex problem-solution has always been started with the planning to approach a solution. The role of planning is a very important aspect and is observed in literature. Many research questions describe the length of planning interval related to complex problem-solution performance. The interaction, variation, and time effects between different solutions are the major area which has been widely studied in different literature. Eichmann et al. investigated different planning activities to solve CP and operational planning that indicate three behavioral measures of longest planning interval and delay of longest and variance intervals between two successive interactions.

In another effort, a proposal highlighted the multi-ple complex system (MCS) approach to assess general-domain of (CPS) skills that processes the knowledge application and acquisition. After constructing the formal frameworks for describing CPS, it measures some of inherent issues of assessing CPS single-task skills (i.e. difficult fixed items, unknown reliability, and large impact of random errors). Greiff et al. determined the complexity factor into three sub-categories (low, medium, and high) for different inputs. In this article, it is suggested that using greater number of inputs leads to higher complexity issue and more difficult is the solution.

Apart from planning and problem categorization, there are numerous studies that highlighted the involvement of the data for complex problem-solutions. Statistical analysis of CPS data leads to different pattern of individual ability approaching toward the problem. For example, Chen et al. considered the final outcome of complex problem-solutions and run prediction interval during task completion and recorded the stored data in system log files. The trained model better understands the individuals and provides us a problem-solving pattern that leads to a good design of automation. The proposed work attempted to solve few questions, such as Does the individual need? Will the individual succeed or fail? and How much additional time during task completion?

However, the evolution of modern technologies has changed the real aspects of complex problem-solution. For example, in previous studies, the solution of CPs consist of human-based cognition and psychology that were part of the solution process. However, due to the advancement in machine learning and IoT technologies, the solution is purely based on artificial intelligence techniques. The key idea in various modern studies relating to CP decomposes it into sub-problems. Chen et al. transform and map the simpler connected problem components to different IoT layers to solve complex information problems. This is a similar approach to Fourier transform where a complex signal is distilled based on range of frequencies, and then multi-stage linear solution is developed. These frequencies are amplified, attenuated, and eliminated altogether and transformed back to the original realm to implement it with the physical components.

A same approach is applied to business domain and solving them using IoT. Similarly, there are different studies which are not related to the solving process, but rather focus on modeling the CP.

The concept of task is important in CPS. The task orchestration concept of IoT takes the benefit of inherent flexibility in task-oriented composition other than service-oriented composition. The term task orchestration often pops up in task-based deployment rather than service-based deployment. Service-oriented deployment is a popular architecture, however, a small change from users makes it difficult to change in the core. In other words, the service-oriented architecture is not robust to any frequent changes. In this work, the gap of artificial cognition toward CPS is combined with the modern, scalable, and flexible task orchestration-based solution to design a novel architecture, named hierarchical CPS architecture (HCPSA), which is the first attempt in this domain and the best to the authors knowledge.

Design architecture of proposed HCPSA

In this section, the architecture design of HCPSA is elucidated. The design pattern proposed in this work has different modules loosely coupled and can be omitted if required without any modification to the core. Low coupling and high cohesion is the preferred approach for systems which has the need of scalability. Every problem have many sub-problems known as tasks represented with different functional units. A task is the building block of the proposed architecture, which is referred to as a monolithic entity responsible for carrying out a singular functionality with a definite goal. The top module of the proposed architecture is the task generator module, which takes requirements from the external client and autonomously generates tasks.

The virtual objects are supplied from the repository, and the methods of task and the behavior of devices are matched to find the best map. The sensor data alongside tasks are applied to the prediction module to find different forecasting attributes. The optimization module keeping in mind of the constraint finds the optimal
goal which is then represented into the form of optimal tasks. The optimal tasks are scheduled and then control algorithms are run to find the optimal duration.

As part of the CP modeling, we add the problem along with constraints, prediction goals and optimization goal. Sample jobs such as energy prediction and load demand prediction can be solved using prediction solvers, whereas optimization solver does the job of optimal energy management with available resources.

The architectural model of the HCPSA is shown in Figure 2. Each complex has different goals with different set of tasks. The tasks methods are executed on devices once the tasks are generated. A prediction algorithm is used to finds the target features of appropriate tasks. In the microgrid scenario, load prediction, demand prediction, and renewable energy prediction are the few output features. These output features find the optimal values of tasks and discarded those if not found. The optimal tasks are mapped to virtual objects using virtualization technology. The control algorithm is used to prevent the over-use of devices for optimal scheduling of tasks. Finally, the tasks are allocated using data function and uniform resource identifier (URI) to the corresponding grid devices. The whole process has four main steps. First, the problem is defined in terms of different micro-problems (MPs), each of them have a defined goal. For instance, the optimal cost is a CP which can be split into two MPs such as renewable use maximization and non-renewable, such as diesel, use minimization. The goal is to either minimize or maximize. The next layer is the solution composition layer which also has three sub-layers: task management, prediction, and optimization. The tasks are the representations of sensors and actuators that are stored to predict patterns. The model coupled with other constraints is given as an input to perform optimization. The optimization result is also represented as optimal tasks mapped to virtual objects. The virtual objects are scheduled in cyber domain to predict the sequence of execution, and finally, the control module finds the duration before deployment on physical devices.

Microgrid indoor climate control includes various crucial parameters that need to be kept within certain desired range to ensure productive environment to meet the load demand.

**Design of task orchestration for smart service delivery**

Analysis and decomposition are the first phase in problem module. This phase analyzes the input problem and generates MPs and complex tasks. The description of the problem is analyzed using text mining techniques to extract tokens which indicates a unit function of the problem. The functional unit can be either atomic or
composite. For composite functional unit, the system generate an MP, otherwise, a complex task is generated for atomic functional unit.

The first layer of the multi-stage process consists of MPs and solvers and other candidate MP, candidate solver objects (SOs), and state-of-the-art algorithms. The mapping approach can be either manual or semi-autonomous. The best solver is computed on different correlative analysis in semi-autonomous approach while MPs are dragged and save it in a repository in manual approach.

The optimal tasks are mapped to the relevant pair of problem and solver. These optimal tasks are mapped to virtual objects in the next level of mapping phase, where consensus algorithm is applied that can be either manual or semi-autonomous. The result of the mapping algorithm is a task, virtual object which is stored in the repository later to be used by scheduling solvers and control solvers. Figure 3 shows the multi-level mapping architecture.

First, the CP is analyzed using different techniques such as text mining and so on. Second, the problem is decomposed into multiple MPs. Every problem has a goal, and every MP has a sub-goal. Every MP contributes toward a solver. A solver is based on a condition which is either to minimize or maximize any other conditions. The visualization space has different problems and MPs on one side. The next side is the output side which is the goal of the problem.

The middle area is the solution modeling area. Solution modeling takes a solver and maps it on optimization, prediction, scheduling, and control algorithm. The optimization algorithm takes different objective functions while the scheduling and control algorithm takes certain conditions into consideration. The modeling is based on the MP’s goal. For instance, if the goal is economy, the modeling will contain the use of cheap component, and the control algorithm will try the minimum time to save operational cost. Similarly, the scheduling algorithm will also take certain goal such as reliability to prioritize those tasks which impact the most on the final outcome (solution) of the problem. Finally, the spreadsheet contains different solutions. Each solution considers optimal tasks, scheduled tasks (the ordering of the tasks), control (the optimal duration of the tasks), and finally the allocation mechanism. The solution line can be deployed using different communication protocol on end-devices as shown in Figure 4.

**Predictive optimization in HCPSA**

Informed and optimal decisions are the core functionality of the smart city service implementation. In this way, CPSaaS architecture solves four types of
problems: prediction, optimization, scheduling, and control. A prediction problem refers to behavior of a certain parameter in terms of future perspective and considered the CP, if the nature of prediction problem is unknown and many contradicting solutions. In this work, a prediction problem can be assigned different prediction solvers based on the analysis of data and make solution of possible feasibility. The proposed architecture also provides the efficiency of solvers and quality of solution.

Optimization plays key functionality role relating to core solutions in the proposed architecture and selects optimal design values based on input parameters with a set of constraints. Optimization refers to the process of finding the minima or maxima of higher-order cost functions of linear or complex functions. In this architecture, the aim is to optimize the problem of relevant category and get the optimal values of design parameters.

**Development model of HCPSA**

The general development model is aimed to have a loosely coupled repository of components which can be provided on demand. The repositories are not made part of the core architecture in order to make it lightweight and efficient. The pools comprise of solver pool, constraints pool, and tasks pool. Solvers pool, in turn, has four types of sub-pools: optimization solvers, prediction solvers, scheduling solvers, and control solvers. All these pools provide different functionality on fly. The loading of the components is lazy and dynamically loaded once instantiated, as shown in Figure 5.

Based on the hierarchical model in Figure 2, the database is modeled. On top of the hierarchy is the project entity. A project is said to consist of different CPs. A CP has a certain goal. For better performance, we decompose a CP into MPs. Likewise, the MP has a sub-goal which is the final outcome which we would like to achieve. The MP has many tasks. Similarly, it has a solution as well.

The solution is a known path toward achieving the goal of the MP and problem. Thus, the solution has solver and either of solver has different solver types, such as scheduling, optimization, control, and prediction. The solver also has objective function and constraints which affect the final outcome of the goal of the problem. Once the solution is made, it has solution
tasks which are those tasks that contribute to the solution. The tasks are mapped to virtual objects, scheduled them, and then finally allocated to physical devices.

Figure 6 shows the entity-relationship diagram of the proposed architecture. A project has connection of many problems which in turn has a connection of many MPs. Similarly, an MP has different tasks which are related to different tables such as mapped_task, deployed_task, scheduled_task, and allocated task. Similarly, solution task is the tasks which take part in the solution process. The solution entity has links with solvers, objective functions, and prediction goals. Solver, in turns, has different specialized entities such as optimization, prediction, scheduling, and control. Either of them corresponds to a distinct set of solver and algorithms.

Formulation of general objective function in CPS architecture

A problem $P$ is characterized by different design parameters. Suppose $X$ represents the input vector of $n$ dimensions $[x_1, x_2, x_3, \ldots, x_n]^T$. The output is also a $k$-dimensional vector $[y_1, y_2, y_3, \ldots, y_k]^T$ the goal of the CPS is to find a transformation vector which moves the input vector to the output vector. The objective function which represents the goal of the problem denoted by $O(X)$ has of the following generic form.

$$O(X) = \min \sum_{i=1}^{N} (a_n F_n(x)) + \sum_{n=1}^{N} (c_n P_n(x)) \quad (1)$$

where $F_n(x)$ is the cost function whose goal is minimization of the output value, $G_n(x)$ is the cost function whose goal is the maximization of the output value. $P_n(x)$ is the general form of constraints which are taken as interior or exterior penalty function as a standard representation of non-constrained optimization problem. Finally, $a_n$, $b_n$, and $c_n$ represents the user-defined weights of each objective.

In order to illustrate the generalization of the objective function, we take the following example:

$$X = [x_1, x_2]$$

$$F_1(X) = \frac{1}{3} (x_1 + 1) + x_2$$

$$F_2(X) = x_1 + x_2 + 1$$

$$G_1(X) = x_1 + 4x_2 + 3$$

$$G_2(X) = 5x_1 + 7x_2 + 2$$
Subjected to constraints

\[ g_1(X) = -x_1 + 1 \leq 0 \]
\[ g_2(X) = -x_2 \leq 0 \]

Based on equation (1), the objective function will be given as follows:

\[ O(X) = \min \left( \frac{2}{\frac{1}{x_1} + \frac{1}{x_2}} + P(x_1 + 1, x_2) \right) \]

Penalty function generally takes the log of the constraints of the inverse of the constraints to convert it into a non-constrained optimization problem. The simplified form of the equation is given as

\[ O(X) = \min \left( \frac{2}{\frac{1}{x_1} + \frac{1}{x_2}} + \frac{1}{x_1 + 1} - \frac{1}{x_2} \right) \]

**Implementation of HCPSA**

In this section, the execution results of the proposed architecture are illustrated in details. The proposed architecture is implemented in Python. HTTP is used as an application protocol, and Flask is used as a framework to ease the creation of client-server topology. We have used NEOS server for fetching external solvers for different optimization algorithms. Interfacing with NEOS Server is performed using AMP Language using remote procedure calls.

We used Bootstrap 3 for visualization which provides a set of CSS-based design widgets and JavaScript-
based animation. Bootstrap makes front-end development very organized and fast. To view MVC for template, we utilized the default template engine Jinja which is shipped with Flask core. Different add-on and libraries, such as Google MAPS API, AMPL, NEOS Server API, and JSPlumb have been utilized to perform different tasks toward the goal. For example, Google maps provides the visualization of devices, while AMPL used to translate objective function from database into concerned format which is then forwarded as an input to NEOS Server in the form of XML. Finally, JSPlumb is a JavaScript-based library which allows users to design the problem in design plan and map different tasks on it. The technology stack is shown in Figure 7.

The main administrative panel is shown in Figure 8. The first step, which is referred to as the problem analysis module has an inherent algorithm to consume the description of problem via a webform and automatically generates MPs and complex tasks. The generated tasks are stored in the database and can be repopulated in different interfaces for the design of solution process.

Afterwards, the problem modeling layer which is the main module of the proposed system has designated a pool of solutions. A solution is correlated with the problem using different solvers and using drag-and-drop features. This interface performs the mapping mechanism on different levels in hierarchical fashion. For instance, first prediction solver is mapped on a problem with prediction goal, then on solver with optimization goal and so on until final mapping which is the device mapping.

The main administrative panel has a map for locating different IoT devices in real time. It also has various listing such as problems listing, tasks listing, device listing, virtual objects listing, to name a few. The right sidebar contains a flow of the whole process by providing necessary links to each process from problem description to resource allocation.

The solution modeling is a prerequisite to solution composition. In this modeling, the components participating in the composition process is provided to the architecture. These components include data optimization, data prediction, solvers, and objective functions.

To populate these components to compose a solution in the next phase, the components are added to the system. The basic flow of the architecture is demonstrated in Figure 9 in the form of state machine.

As described in earlier sections, the CP has different sub-problems having different goals. Every goal can be achieved with a set of tasks. The tasks have methods and executed on the corresponding devices once the tasks are generated and apply prediction algorithm to find the target features for appropriate data tasks. In case of microgrid scenario, load prediction, demand prediction, and renewable energy prediction are the few output features. The output features are used to find the optimal values of tasks and discarded otherwise. These optimal tasks are mapped to virtual objects using virtualization and are scheduled to find the optimal ordering of time. After finding the optimal ordering, the control algorithm is run to prevent the over-use of devices. Finally, the tasks are allocated using data function and URI method to the corresponding IoT devices.
The execution results are shown in Figure 10. The label on each interface shows the order of the flow in the architecture. The interfaces show the execution interfaces of various processes. The architecture has mainly four phases: problem analysis and tasks generation, solvers management, solution modeling, and resource allocation.

**Problem analysis and description**

The top row shows the problem analysis and tasks generation interface. The problem is added to the algorithm, and the resultant interface autonomously generates contents based on the input text. The MPs and complex tasks are generated based on text mining and pattern analysis techniques as shown in figure with label 2. The tasks can be either added to the repository or discarded if it does not match with the requirement of the problem as shown in interface marked as label 3.

**Solver management**

Interfaces with label 4.1, 4.2, and so on are operations belong to a single layer which can be performed concurrently. Different optimization algorithms, prediction algorithms, problem objective functions, and scheduling algorithms are added to the system with these interfaces. In this architecture, they are called solver. If the solver is type of optimization function, an additional component called an objective function or cost function is required. The objective function is the quantitative measurement of the goal to achieve an optimization.
solver as shown in label 4.2. For example, an optimization solver aims to minimize the cost, then it tends to achieve computation cost.

The objective function consists of design variables, cost function, and constraints, and the function is given to the system in a well-documented form which must be consumed by the optimization solver.

Solution modeling
This is the most important modules of this architecture for the composition of solution using a do-it-yourself (DIY) plane and the problems and solution components are populated in this plane. This plane consist of four distinct section: solver mapping, task mapping, scheduling mapping, and control mapping. The two sub-modules: prediction mapping and optimization mapping lie in solver mapping.

The plane use JavaScript-based library known as JSPlumb facilitates non-technical users to compose the solutions based on their requirements without the need of any kind of coding. JSPlumb facilitate users to connect one component of the solution using drag and drop features. Usually, there are MPs having distinct prediction goals and the prediction solver used to make a prediction solution line. Prediction and optimization solution are the first and foremost things for MPs to be performed. Composition of prediction solution line and the composition of optimization solution line are two important aspects. Suitable algorithm is applied to map MP into its prediction objective. For example, a weather reliability problem in micro-grids makes smart decision for energy forecasting, and for this energy, prediction goal is mapped to artificial neural network (ANN) solver. So, that the first line in solution repository belongs to this configuration, as marked in label 5.

Resource allocation
The final step is resource allocation and result monitoring. As labeled 7, the solution, which consists of tasks are allocated on different IoT nodes to perform a complex task with sensors and actuators. For instance, an optimization solver is hosted in NEOS server for optimization solution. The host has a variety of optimization solvers in different categories. The solver gets and visualizes the results once the solution is deployed as marked 8.1, 8.2, and 8.3. For example, in case of load projection, the solver gets the results as shown in graphical form of solution page (label 8.3). The optimization solver generates optimal values rather than prediction and is returned to CPS architecture from the NEOS server.

Results
The stability of any architecture depends on how it responds different load conditions. There are four major workflows in the proposed architecture which we have assessed with varying different input parameters. The optimization module is the most dependant module among all due to external RPC calls. However, the
The result of optimization is cached and the subsequent similar call is not made to the remote server, but instead taken from the cache. The number of cache hits increases once the system operates on more and more problems. The more the number of cache hits, the faster is the response time. Moreover, the problem space is another

Figure 10. Execution results of proposed HCPSA prototype implementation.
crucial measure which checks the scalability of the system. The problem size is calculated based on the number of input parameters and output parameters. A simplest problem is a problem with three inputs and two outputs. Any number lesser than this is not considered as a CP and thus is outside the scope of this work. Figure 11 portrays the results of different scenarios. The first plot exhibits the effect of cache hit once the system is started execution. The cache hit keeps increasing with time, and at some points, it gets saturated. The saturation point is where the response time of the system is optimal. Once the optimal condition is achieved, the problem size is varied. For increasing number of parameters under optimal conditions, the response time increases. Nevertheless, the response time depends on the number of input data set, which is kept constant. The response time for a very massive problem space of 20 is recorded as 70 ms which is not very slow considering the solution exists and the solver used by the NEOS server can solve it. The latter case is omitted for experimental purpose, since solver can fail to solve a problem for a very simple problem due to different reasons such as unsuitability of selection of solver, wrong type of solver, to name a few.

Finally, the response time of each individual workflow is recorded. The bar graph shows the optimization workflow, the two line graphs each shows scheduling time and prediction time, respectively. As mentioned earlier, the optimization problem due to its dependence on external server (NEOS), the response time is much slower than the rest of the modules. The response time of the prediction workflow is more efficient as compared to the scheduling and allocation. The scheduling algorithm gets exhausted for increasing the number of parameters and thus takes time to schedule tasks in appropriate window of time. The overall response time of the system, even for an unrealistically large problem space in the order of 20s can still generate a response time within 200 ms, which proves that the architecture is robust and scalable.

Conclusion

CPS is an everyday human life problem which cannot be approached easily due to many contradicting goals and parameters. Previous studies show the solution of CPs using statistical data and expert system. In our previous work, we have implemented a smart city-based case study to propose automated and intelligent solutions using a variety of solvers. This work further extended this approach and proposed a generic architecture that employs artificial intelligence techniques to solve CPs using task orchestration in any IoT-based smart space. We also propose a design pattern for complex problem-solution having different parameters and goals. The architecture pattern comprises of optimization, prediction, scheduling, and control pattern, leveraging the software best practices of coupling and modularity as paper’s goal. We employed DIY planes which encourages the non-technical people to compose their problems. This increases the contribution and encourages non-technical people having less technical and creative skills to think with technologies and contribute their thinking without the need to code in the technical aspect. The proposed architecture has the advantage to add flexibility and scalability using task orchestration approach. It provides users with interfaces to describe their problems, compose their solutions and deploy them. The proposed architecture gives the following features, easy to use, robust from faults, adequate processing time, and even scale up in the peak time.

Figure 11. Performance evaluation of different metrics of the proposed architecture.
Declaration of conflicting interests
The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding
The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This research was supported by Energy Cloud R&D Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT (2019M3F2A1073387), and this work was conducted by Center for Applied Research and This work was supported by the Institute for Information & communications Technology Promotion (IITP) (NO. 2022-0-00980, Cooperative Intelligence Framework of Scene Perception for Autonomous IoT Device) and by the National Research Foundation of Korea (NRF) Grant funded by the Ministry of Basic Education under Grant 2021R1I1A1A01045177.

ORCID iDs
Shabir Ahmad https://orcid.org/0000-0002-8788-2717
Salman Khan https://orcid.org/0000-0003-0277-5021

References
1. Xia F, Yang LT, Wang L, et al. Internet of things. Int J Commun Syst 2012; 25(9): 1101–1102.
2. Aliee FS, Kashfi H and Farahani B. The evolving enterprise architecture: a digital transformation perspective. In: Proceedings of the international conference on omni-layer intelligent systems, Crete, Greece, 5–7 May 2019. New York: ACM.
3. Vermesan O, Friess P, Guillemin P, et al. Internet of things strategic research roadmap. Internet Things Glob Technol Soc Trends 2011; 1: 9–52.
4. Ahmad S, Hang L and Kim DH. Design and implementation of cloud-centric configuration repository for DIY IoT applications. Sensors 2018; 18(2): 474.
5. Ahmad S, Hussain I, Fayaz M, et al. A distributed approach towards improved dissemination protocol for smooth handover in MediaSense IoT platform. Processes 2018; 6: 46.
6. Ahmad S, Malik S, Ullah I, et al. An Adaptive approach based on resource-awareness towards power-efficient real-time periodic task modeling on embedded IoT devices. Processes 2018; 6(7): 90.
7. Ahmad S, Malik S, Ullah I, et al. Towards the design of a formal verification and evaluation tool of real-time tasks scheduling of IoT applications. Sustainability 2019; 11(1): 204.
8. Eichmann B, Goldhammer F, Greiff S, et al. The role of planning in complex problem solving. Comput Educ 2019; 128: 1–12.
9. Greiff S, Fischer A, Stadler M, et al. Assessing complex problem-solving skills with multiple complex systems. Think Reasoning 2015; 21(3): 356–382.
10. Chen Y, Li X, Liu J, et al. Statistical analysis of complex problem-solving process data: an event history analysis approach. Front Psychol 2019; 10: 486.
11. Ahmad S, Ali J, Jamil F, et al. Complex problems solution as a service based on predictive optimization and tasks orchestration in smart cities. Comput Mater Con 2021; 69: 1271–1288.
12. Complex problem solving and Internet of Things, https://thingsmatrix.com/2015/09/12/leveraging-internet-of-things-iot-in-solving-complex-business-problems/ (accessed 28 May 2020).
13. Ahmad S, Khudoyberdiev A and Kim D. Towards the task-level optimal orchestration mechanism in multi-device multi-task architecture for mission-critical IoT applications. IEEE Access 2019; 7: 140922–140935.
14. Ahmad S and Kim DH. A multi-device multi-tasks management and orchestration architecture for the design of enterprise IoT applications. Future Gener Comp Syst 2020; 106: 482–500.
15. Ciccozzi F, Crnkovic I, Di Ruscio D, et al. Model-driven engineering for mission-critical IoT systems. IEEE Software 2017; 34(1): 46–53.
16. Casadei R, Fortino G, Pianini D, et al. A development approach for collective opportunistic edge-of-things services. Inform Sciences 2019; 498: 154–169.
17. Velasquez K, Abreu DP, Gonalves D, et al. Service orchestration in fog environments. In: 2017 IEEE 5th international conference on future internet of things and cloud (FiCloud), Prague, 21–23 August 2017. New York: IEEE.
18. Velasquez K, Abreu DP, Assis MRM, et al. Fog orchestration for the internet of everything: state-of-the-art and research challenges. J Internet Serv App 2018; 9(1): 14.
19. Liu X, Liu Y, Song H, et al. Big data orchestration as a service network. IEEE Commun Mag 2017; 55(9): 94–101.
20. Ahmad S, Mehmood F, Mehmood A, et al. Design and implementation of decoupled IoT application store: a novel prototype for virtual objects sharing and discovery. Electronics 2019; 8(3): 285.

Appendix I

Notation
NEOS remote API server for hosting different solvers
AMPL a mathematical programming language