Mapping Imprecise Computation Tasks on Cyber-Physical Systems

Lei Mo, Angeliki Kritikakou

Peer-to-Peer Networking and Applications manuscript No.
(will be inserted by the editor)

Mapping Imprecise Computation Tasks on Cyber-Physical Systems

Lei Mo · Angeliki Kritikakou

Received: date / Accepted: date

Abstract By allocating a set of tasks onto a set of nodes and adjusting the execution time of tasks, task mapping is an efficient approach to realize distributed computing. Cyber-Physical Systems (CPS), as a particular case of distributed systems, raise new challenges in task mapping, because of the heterogeneity and other properties traditionally associated with Wireless Sensor and Actuator Networks (WSAN), including shared sensing, acting and real-time computing. In addition, many of the real-time tasks of CPS can be executed in an imprecise way. Such systems accept an approximate result as long as the baseline Quality-of-Service (QoS) is satisfied and they can execute more computations to yield better results, if more system resources is available. These systems are typically considered under the Imprecise Computation (IC) model, achieving a better tradeoff between QoS and limited system resources. However, determining a QoS-aware mapping of these real-time IC-tasks onto the nodes of a CPS creates a set of interesting problems. In this paper, we firstly propose a mathematical model to capture the dependency, energy and real-time constraints of IC-tasks, as well as the sensing, acting, and routing in the CPS. The problem is formulated as a Mixed-Integer Non-Linear Programming (MINLP) due to the complex nature of the problem. Secondly, to efficiently solve this problem, we provide a linearization method that results in a Mixed-Integer Linear Programming (MILP) formulation of our original problem. Finally, we decompose the transformed problem into a task allocation subproblem and a task adjustment subproblem, and, then, we find the optimal solution based on subproblem iteration. Through the simulations, we demonstrate the effectiveness of the proposed method.

Keywords Cyber-Physical Systems · Task Mapping · Imprecise Computation · Problem Linearization and Decomposition

This research is funded by ANR ARTEFACT (AppRoximaTivE Flexible Circuits and Computing for IoT) project (Grant No. ANR-15-CE25-0015), and National Natural Science foundation of China (Grant No. 61403340).

Lei Mo and Angeliki Kritikakou
Univ Rennes, INRIA, CNRS, IRISA, 35042 Rennes Cedex, France
E-mail: lei.mo@inria.fr, angeliki.kritikakou@irisa.fr

1 Introduction

Cyber-Physical Systems (CPS) consist of a set of wireless nodes with sensors and actuators that not only have the capability to measure the specific physical variables (e.g., temperature, illumination intensity, voltage), but also have the capability to control them through the actions preformed by the actuators. *Energy efficiency* and *real-time execution* are critical and challenging issues for the system design [1]. This is because most of the nodes have energy constraints, especially when they operate on battery power. In addition, real-time responsiveness is required by many CPS applications, e.g., target tracking, since missing of the task deadline can cause serious results.

Processing and transmitting large amount of sensing data and control commands in real-time applications require in-network processing [2]. To reduce the network traffic, instead of collecting and sending all data to a remote base station, part of the processing is done on-site with nodes have computational capability, so that only a small part of pre-processed data needs to be sent. For instance, in target tracking application, the sensor nodes analyze the captured images on-site by extracting the important features, and send only the features for further processing to the base station or actuator nodes. This model of computation is known as "Fog/Edge-computing" [3]. In this context, task mapping plays an essential role in in-network processing by solving the matching problem between tasks and nodes subject to application requirements.

In some application domains, such as image processing, target tracking, real-time heuristic search, and control engineering, less accurate results computed before the deadline are preferable than accurate, but too late, results [4]. This statement holds because a real-time application has to provide a result before a given deadline. When not enough time is available, approximate results are acceptable as long as the baseline Quality-of-Service (QoS) is satisfied and the results are provided in time. For instance, in target tracking application, frames with a lower quality are better than missing frames; an estimation of target's location in time is better than an accurate location arriving too late. In these domains, the applications can be modeled as *Imprecise Computation* (IC) tasks [5], where a task is logically decomposed into a *mandatory subtask* and an *optional subtask*. All the mandatory subtasks must be completed before the deadline to have an acceptable result, while the optional subtasks can be left incomplete at the cost of reduced quality. The QoS of such systems increases with the longer execution of the optional subtasks [6].

We can classify prior work on task mapping according to the following criteria: whether the: 1) tasks are precise or imprecise, 2) platform is multicore system or networked system, and 3) solution is optimal or heuristic. Table 1 provides a summary of some representative papers from the literature. Task mapping is a well-known problem in embedded system community, no matter the tasks are precise or imprecise, and the multicore platforms are homogeneous [4,7,8,12–15] or heterogeneous [3,9–11,16]. However, the work deals with task mapping problem on networked system is rare, and most of them consider precise tasks [2,6,17–20]. Two main keypoints that differentiate our work compared to the existing literature:

1. Compared with the task allocation in multicore platform, the task allocation in networked system, such as CPS, is constrained in the sense that some tasks have

Table 1 Classification of some task mapping approaches

Reference	*Task*		*Targeted Platform*		*Solution*	
	Precise	Imprecise	Multicore Sys.	Networked Sys.	Optimal	Heuristic
[3]	√		√			√
[7]	√		√		√	
[8]	√		√		√	
[9]	√		√			√
[10]	√		√		√	
[11]	√		√		√	
[4]		√	√		√	
[6]		√	√			√
[12]		√	√		√	
[13]		√	√			√
[14]		√	√			√
[15]		√	√		√	
[16]		√	√			√
[2]	√			√		√
[17]	√			√		√
[18]	√			√		√
[19]	√			√	√	
[20]	√			√		√
Proposed		√		√	√	

a one-to-one correspondence with the nodes, while the allocation constraints of other tasks may not be restricted. For example, a temperature measurement task and a temperature control task can be placed only on nodes with temperature sensors and actuators, respectively. On the other hand, the task that processes the measurement readings and determines the control action has less restriction in the allocation. Moreover, the applications in CPS consist of communication tasks. This data communication between tasks on different nodes in a CPS also affects other nodes in the system, because the nodes involved in routing have to spend energy for the data transmission.

2. In the classical task mapping on networked system, the tasks are assumed to be precise. However, in the cases where the tasks are imprecise, a set of additional variables with respect to the optional subtask adjustment, are introduced into problem. The values of these variables affect the 1) objective function, as the aim of the QoS-aware task mapping is to maximize QoS function constructed by the optional subtasks, and 2) energy and real-time constraints, since the longer optional subtasks are executed, the more energy and time is required to execute them. The extension from the precise task framework to the imprecise task framework is not straightforward, as the non-linear relationship between the optimization variables makes the problem hard to solve directly. Since different task allocation schemes lead to different optional subtask adjustment, the task allocation and optional subtask adjustment should be jointly addressed to find the optimal solution.

1.1 Related Work

1.1.1 Energy-aware Task Mapping

Existing works that focus on the energy-aware task mapping problem aim at minimizing the energy consumption or the execution time of tasks (makespan) un-

der system resource and application constraints. In multicore task mapping scenarios [3,7–11], Mixed-Integer Non-Linear Programming (MINLP) is a popular method to formulate the problems of mapping independent tasks [9] or dependent tasks [7,8] onto the multicore platforms. To efficiently solve these complex problems, in [9] the MINLP problem is relaxed to a convex problem by replacing the binary variables with the continuous variables, and, then, the relaxed problem is solved using polynomial-time methods. Based on the structure of the MINLP problem, it can be transformed to a Mixed-Integer Linear Programming (MILP) by approximating the quadratic function with a linear function [7] or by introducing additional constraints to decouple the non-linear items [8]. Thus, the transformed problem (MILP) can be solved using commercial solver, such as CPLEX. In [10], the independent task mapping problem on homogeneous multicore (e.g., ARM big.LITTLE) is formulated as Linear Programming (LP) as the platform supports task migration. When tasks are dependent, the task mapping problem is formulated as Integer Non-Linear Programming (INLP) in [3] and solved using heuristic. Similar problem is studied in [11], but the problem is formulated as Integer Linear Programming (ILP), and, thus, this problem can be optimally solved using Benders Decomposition (BD) [21].

In networked system task mapping scenarios [2,17–20], the dependent task mapping and Dynamic Voltage and Frequency Scaling (DVFS) joint-design problem is studied in [2]. This problem is formulated as MINLP and solved by a heuristic method. The works in [17,20] mainly focus on the task-to-node allocation problem, and, thus, INLP is used to formulate the problem of mapping dependent tasks onto the nodes in networked system. The INLP problem is first transformed to an ILP by introducing additional constraints, and, then, solved using heuristics. In [18,19], the energy consumption of nodes spent in communication and the makespan can be minimized by allocating the tasks to proper nodes. However, the tasks in aforementioned approaches are not modifiable, and, thus, no exploration of the QoS improvement through the optional subtasks adjustment is considered.

1.1.2 QoS-aware Task Mapping

Other works consider the QoS-aware task mapping problem using the IC-task model and having as a goal to maximize the QoS under a set of real-time and/or energy supply constraints, e.g., [4,6,12–16]. The target platform considered in [12] is a single core platform. Therefore, the task allocation problem is not taken into account. Although the works in [4,6,13–16] target at multicore platforms, some assumptions are made during the problem formulation. More precisely, the task-to-processor allocation is fixed and given in advance for all the tasks in [6], while in [4,14,16] the tasks are independent, and in [13,15] the multicore platforms are homogeneous. When taking multiple system requirements into account, the complex coupling between the optimization variables makes the problem difficult to solve, especially when the coupling is non-linear and non-convex. The methods that used to solve the aforementioned problems can be classified into two main classes. The first class includes the methods based on heuristics, e.g., [6,13,14,16]. The second class includes the methods that always produce an optimal solution, e.g., [4,12,15]. Although the heuristics are able to find the feasible solution in a short time, they do not provide the bounds on solution quality, whereas they are

sensitive to changes in the problem structures. To the best of our knowledge, no existing work consider QoS-aware task mapping problem in CPS.

1.2 Contributions

Our goal is to solve the mapping problem of data communication dependent IC-tasks on the nodes in CPS. We determine which node the task should execute on (*task allocation*), and adjust the optional part of each task (*task adjustment*), such that the system QoS is maximized, while meeting the energy and the real-time constraints of tasks. Our main contributions are summarized as follows:

1. We formulate the QoS-aware IC-task mapping problem in CPS as an MINLP problem, which takes the dependency, energy and real-time constraints of IC-tasks, as well as the sensing, acting, and routing of the CPS into account.
2. We prove that by introducing the auxiliary variables and the additional constraints into the problem, the nonlinear items, which are introduced by the product of the variables related to the task allocation and optional subtask adjustment, can be linearized. Thus, the MINLP problem can be transformed to an MILP. As this linearization does not change the objective function as well as the feasible region of the problem, the problem transformation does not cause performance degradation.
3. A novel Optimal Task Mapping algorithm, referred to as OTM, is proposed to optimally solve the transformed problem. The basic idea of OTM algorithm is similar to closed-loop control and it is based on BD framework. BD decomposes an MILP problem into a Master Problem (MP) and a Slave Problem (SP), and, then, solves them iteratively by utilizing the solution of one into the other. The MP involves an ILP only considering the binary variables. The continuous variables are considered in the SP, which is a LP. Unlike the classical BD approach, we prove that by relaxing the MP to an LP to find a feasible solution and by replacing the optimal solution of MP with the feasible solution during the iteration between the MP and the SP, the optimality of the solution is still guaranteed.

1.3 Paper Organization

The remainder of this paper is organized as follows. Section 2 presents the system model and formulates the problem under study. Section 3 and Section 4 design the problem linearization method and the optimal task mapping algorithm, respectively. Finally, Section 5 shows the simulation results and Section 6 concludes this work.

2 System Models and Problem Formulation

2.1 Motivational Example

As the temperature control example [17] illustrated in Fig. 1, a room is instrumented with six wireless nodes $\{\theta_1, \ldots, \theta_6\}$, where three nodes are equipped with

temperature sensors, and two nodes are connected to actuators to control the temperature of the room. The nodes equipped with temperature sensors and actuators are marked with T and A, respectively. We need to periodically determine the average temperature in the room, compare it with a given threshold, and produce the corresponding action. A task represents the sensing, processing, or action activity in CPS. Tasks τ_1, τ_2 and τ_3 are temperature measurement tasks and generate temperature readings of size s_{14}, s_{24}, s_{34}. Task τ_4 processes these readings (e.g., calculates average value or performs state estimation [1]) and transmits the result to task τ_5, which determines the action to be taken through the control algorithm. Tasks τ_6 and τ_7 act upon the data generated by task τ_5 and control the outputs of the actuators.

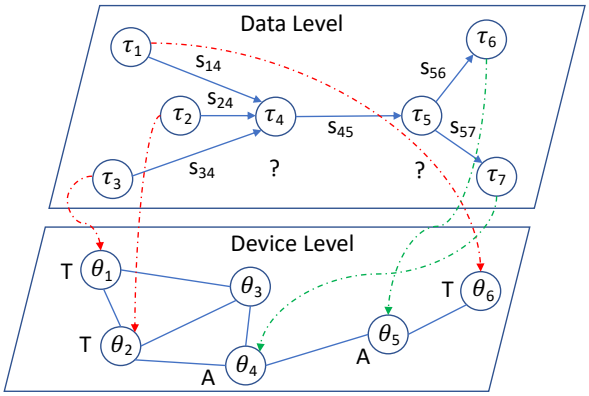

Fig. 1 Temperature monitor and control application.

Each task τ_i is composed of a mandatory subtask and an optional subtask, characterized in terms of the number of execution cycles M_i and o_i, respectively. The optional subtask executes immediately after its corresponding mandatory subtask completes and the QoS of the obtained result highly depends on the optional subtasks. Therefore, the execution time t_i^e and computation energy of each task τ_i will change with the optional part of this task.

The example of Fig. 1 shows the task mapping problem (i.e., task allocation and adjustment) we address in this work.

- First, the placement (task allocation) of the sensing tasks (τ_1, τ_2, τ_3) and the action tasks (τ_6, τ_7) are restricted to the nodes that have relevant capabilities. However, tasks (τ_4, τ_5) can be placed on any of the nodes. Moreover, the task allocation scheme will influence the energy spend at the nodes during the communication since the tasks are dependent.
- Second, of find an optimal schedule for the tasks (task adjustment), two objectives that must be simultaneously achieved, namely: 1) meet the deadlines of mandatory and optional subtasks and the energy supply of the nodes, and 2) adjust the optional subtasks to maximize the QoS. Specifically, as the task mapping example shown in Fig. 2, each task τ_i must be completed within a relative deadline d_i and all the tasks assigned to each node θ_k (including task

execution time t_i^e and data communication time t_{ij} between τ_i and τ_j) must be finished within a global deadline H. Moreover, the energy of node θ_k spent for computation and communication should be no more than its available energy E_k^l. These two objectives are both important, yet often incompatible one with the other. In other words, the real-time and energy constraints may require to sacrifice optional subtasks with great value of system.

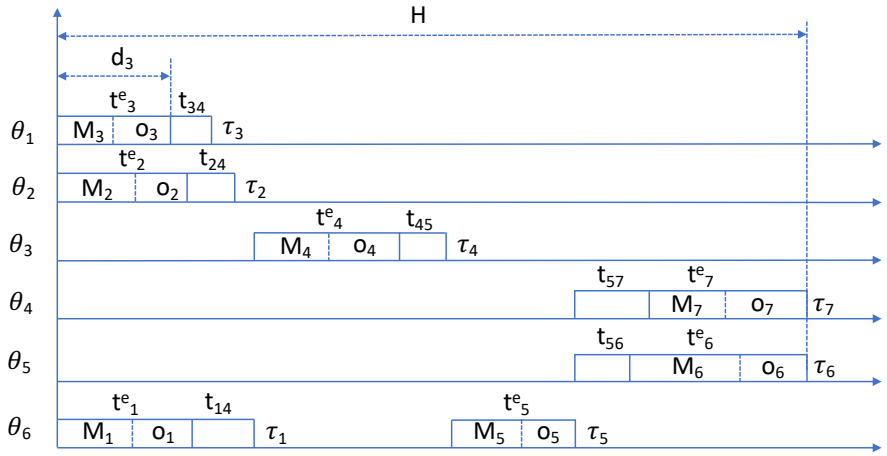

Fig. 2 Task mapping scheme associated with the example shown in Fig. 1.

The notations followed in this paper are: for a matrix $\boldsymbol{m}$, m_{ij} is the $(i,j)^{th}$ element of $\boldsymbol{m}$; for a vector $\boldsymbol{v}$, v_i is the i^{th} element of $\boldsymbol{v}$; $(\cdot)^T$ is the operator for the transpose of a matrix/vector. Let $\boldsymbol{x} = [x_1 \ldots, x_n]^T$ and $\boldsymbol{y} = [y_1 \ldots, y_n]^T$. $\boldsymbol{x} \preceq \boldsymbol{y}$ represents $x_i \leq y_i$, $(1 \leq i \leq n)$.

2.2 Task Model

We consider a task set $\mathcal{T} = \{\tau_1, \ldots, \tau_i, \ldots, \tau_N\}$ of N real-time IC-tasks that are released at the same time 0 and share a common scheduling horizon H. Tasks are dependent, periodic and non-preemptive. The task set $\mathcal{T}$ is modeled by a Directed Acyclic Graph (DAG) $G(V, E)$, where V represents the tasks and E indicates the data dependencies between the tasks. We introduce a $N \times N$ Task Execution Order (TEO) matrix $\boldsymbol{p} = [p_{ij}]$ to describe the dependency on task execution. If $p_{ij} = 1$, task τ_i precedes task τ_j and τ_j is the closest task of τ_i, otherwise, $p_{ij} = 0$. The TEO matrix associated with the example illustrated in Fig. 1 is expressed as:

$$\boldsymbol{p} = \begin{bmatrix} 0&0&0&1&0&0&0 \\ 0&0&0&1&0&0&0 \\ 0&0&0&1&0&0&0 \\ 0&0&0&0&1&0&0 \\ 0&0&0&0&0&1&1 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \end{bmatrix}$$

Each task τ_i is described by a tuple $\{o_i, M_i, O_i, d_i, l_i\}$. l_i is the period of task τ_i, which is also equal to the hyper-period H. O_i is the maximum cycles of the optional subtask of τ_i, i.e., $0 \leq o_i \leq O_i$. A common assumption of existing studies is that the QoS produced is a linear function of optional cycles [4,6], where this function increases its value uniformly with the optional cycles. To quantify the relationship between QoS and optional cycles, each task τ_i is associated with a linear function $f_i(o_i)$ [15].

2.3 System Model

We consider a CPS contains M wireless nodes where M_s nodes $\{\theta_1, \ldots, \theta_{M_s}\}$ equipped with sensors and M_a nodes $\{\theta_{M_s+1}, \ldots, \theta_{M_s+M_a}\}$ equipped with actuators ($M_s + M_a \leq M$). The system operates in rounds (i.e., scheduling horizon H), where in each round all the tasks in task set $\mathcal{T}$ are executed once. The processor of each node θ_k is characterized by a given Voltage/Frequency (V/F) pair (v_k, f_k). Processors of sensors and actuators can operate in two model: one is *run* model, where the power consumption consists of dynamic power P_k^d and static power P_k^s, i.e., $P_k^c = P_k^s + P_k^d$; the other one is *idle* model, where the power consumption is P_k^{idle}. The dynamic power and static power of a processor θ_k under the given V/F pair (v_k, f_k) are expressed as $P_k^d = C_k^d f_k v_k^2$ and $P_k^s = C_k^s v_k^{\rho_k}$, respectively [10]. C_k^s, ρ_k and C_k^d are constants depending on processor type. We assume that when a processor has no task to execute, it transmits into idle mode immediately. The transition time and energy overhead is considered very small compared to that required to execute a task, and is assumed to be incorporated into the execution time and energy of the task [9]. Processors of sensors and actuators are assumed to be *heterogeneous*. When task τ_i is executed on node θ_k, the corresponding task execution time is calculated as $\frac{M_i+o_i}{f_k \lambda_{ik}}$, where $\lambda_{ik} \in (0, 1]$ is the execution efficiency of processor θ_k when it executes task τ_i [9]. The energy consumption of transmitting and receiving l-bit data over a distance d that is less than a threshold d_{th} are defined as $E_{tx}(l, d) = E_{elec}l + \varepsilon_{amp}ld^2$ and $E_{rx}(l) = E_{elec}l$, respectively, where E_{elec} and ε_{amp} are the hardware parameters [2]. We consider system is energy constrained in the sense that the energy budget E_k^l of θ_k at the l^{th} round is fixed that cannot be replenished during the scheduling horizon H.

2.4 Problem Formulation

The problem consists of the objective function to maximize the QoS of the system subject to a set of real-time and energy constraints. Therefore, we determine 1) which node the task should executed on (*task allocation*), and 2) the optional cycles of each task (*task scheduling*). To formulate the problem, we introduce the following variables: 1) task allocation matrix $\boldsymbol{s} = [q_{ik}]_{N \times M}$, and 2) task scheduling vector $\boldsymbol{o} = [o_i]_{N \times 1}$. In addition, we assume that the networked system runs with a given routing protocol [22], and, thus, the following matrices are known [17]: 1) routing energy cost matrix $\boldsymbol{r} = [r_{ijk}]_{M \times M \times M}$, 2) communication data matrix $\boldsymbol{s} = [s_{ij}]_{N \times N}$, and 3) communication time matrix $\boldsymbol{t} = [t_{ij}]_{N \times N}$. The symbols used in the problem formulation are summarized in Table 2.

Table 2 Symbols used in the problem formulation

Parameters	Description
M_a	number of actuators
M_s	number of sensors
M	number of nodes
N	number of tasks
$\mathcal{R}_a$	set of actuators
$\mathcal{R}_s$	set of sensors
$\mathcal{T}$	set of tasks
H	scheduling horizon
θ_k	the k^{th} node
τ_i	the i^{th} task
(v_k, f_k)	voltage/frequency pair of node θ_k
P_k^s	power consumption of node θ_k
P_k^{idle}	idle power consumption of node θ_k
E_k^l	available energy of node θ_k at the l^{th} round
M_i	mandatory cycles of task τ_i
O_i	maximum optional cycles of task τ_i
$f_i(o_i)$	QoS function associated with task τ_i
λ_{ik}	execution efficiency of node θ_k when it executes task τ_i
p_{ij}	$= \begin{cases} 1 & \text{if task } \tau_i \text{ proceeds } \tau_j \text{ and } \tau_j \text{ is the nearest task of } \tau_i \\ 0 & \text{else} \end{cases}$
r_{ijk}	energy consumed per unit of data at node θ_k while routing messages from θ_i to θ_j
s_{ij}	size of data that task τ_i produces for τ_j
t_{ij}	upper bound of time required to transmit the data with size s_{ij} from task τ_i to τ_j
Variables	Description
q_{ik}	$= \begin{cases} 1 & \text{if task } \tau_i \text{ executes on node } \theta_k \\ 0 & \text{else} \end{cases}$
o_i	optional cycles of task τ_i
$g_{i\beta i\gamma}$	auxiliary (binary) variable
h_{ik}	auxiliary (continuous) variable

Let $\mathcal{N} = \{1, \ldots, N\}$, $\mathcal{M} = \{1, \ldots, M\}$, $\mathcal{M}_s = \{1, \ldots, M_s\}$ and $\mathcal{M}_a = \{M_s + 1, \ldots, M_s + M_a\}$. Taking the objective and all the constraints mentioned above, the Primal Problem (PP) is given by

$$\mathbf{PP}: \min_{\boldsymbol{q},\boldsymbol{o}} -\sum_{i\in\mathcal{N}} f_i(o_i) \tag{1}$$
$$\text{s.t.} \begin{cases} \boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{C}_3, \boldsymbol{C}_4, \\ q_{ik} \in Z^+, o_i \in R^+, 0 \le o_i \le O_i, \ \forall i \in \mathcal{N}, \ \forall k \in \mathcal{M}. \end{cases}$$

In PP, we need to deal with the following constraints:

– Task allocation constraints:

$$\boldsymbol{C}_{1a} : \sum_{k \in \mathcal{M}} q_{ik} = 1, \ \forall i \in \mathcal{N}. \tag{2}$$

$$\boldsymbol{C}_{1b} : q_{ik} = 1, \ \forall i, k \in \mathcal{M}_m. \tag{3}$$

$\boldsymbol{C}_{1a}$ imposes that each task τ_i is executed on one node. Let $\mathcal{M}_m$ denote the subscript set of nodes (i.e., sensors and actuators) and tasks that have fixed matching. $\boldsymbol{C}_{1b}$ shows that some task allocation decisions are restricted. As the example illustrated in Fig. 1, we have $q_{16} = q_{22} = q_{31} = q_{65} = q_{74} = 1$.

– Energy constraints:

$$\boldsymbol{C}_{2a} : E_k^{comm} + E_k^{comp} + E_k^s \leq E_k^l, \ \forall k \in \mathcal{M}_s, \tag{4}$$

$$\boldsymbol{C}_{2b} : E_k^{comm} + E_k^{comp} + E_k^a \leq E_k^l, \ \forall k \in \mathcal{M}_a, \tag{5}$$

$$\boldsymbol{C}_{2c} : E_k^{comm} + E_k^{comp} \leq E_k^l, \ \forall k \in \mathcal{M}, \ \forall k \notin \mathcal{M}_s, \mathcal{M}_a. \tag{6}$$

$\boldsymbol{C}_2$ indicates that the total energy consumed by node θ_k during the scheduling horizon H should not exceed the energy supply E_k^l, where E_k^{comm}, E_k^{comp}, E_k^s and E_k^a are the communication cost, computation cost, sensing cost and action cost, respectively. At each node θ_k, the energy spent in data transmission in each round is given by

$$E_k^{comm} = \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} \sum_{\beta \in \mathcal{M}} \sum_{\gamma \in \mathcal{M}} s_{ij} q_{i\beta} q_{i\gamma} r_{ijk}, \ \forall k \in \mathcal{M}.$$

On the other hand, if $q_{ik} = 1$, task τ_i is assigned to node θ_k. Based on the task allocation vector $[q_{1k}, \ldots, q_{Nk}]^T$ with respect to node θ_k, the tasks allocated to node θ_k is $\sum_{i \in \mathcal{N}} q_{ik}(M_i + o_i)$. If node θ_k is assigned with V/F level (v_k, f_k), the time required to executed the tasks allocated to θ_k is $\sum_{i \in \mathcal{N}} q_{ik} \frac{M_i + o_i}{f_k \lambda_{ik}}$. Subtracting the task execution time, the idle time of node θ_k being in the hyper-period H is $H - \sum_{i \in \mathcal{N}} q_{ik} \frac{M_i + o_i}{f_k \lambda_{ik}}$. At each node θ_k, the computation cost in each round is given by

$$\begin{aligned} E_k^{comp} &= \left(\sum_{i \in \mathcal{N}} q_{ik} \frac{M_i + o_i}{f_k \lambda_{ik}} \right) P_k^c + \left(H - \sum_{i \in \mathcal{N}} q_{ik} \frac{M_i + o_i}{f_k \lambda_{ik}} \right) P_k^{idle} \\ &= \left(P_k^c - P_k^{idle} \right) \left(\sum_{i \in \mathcal{N}} q_{ik} \frac{M_i + o_i}{f_k \lambda_{ik}} \right) + H P_k^{idle}, \ \forall k \in \mathcal{M}. \end{aligned}$$

– Real-time constraints

$$\boldsymbol{C}_3 : \sum_{k \in \mathcal{M}} q_{ik} \frac{M_i + o_i}{\lambda_{ik} f_k} \leq d_i, \ \forall i \in \mathcal{N}, \tag{7}$$

$$\boldsymbol{C}_4 : \sum_{i \in \mathcal{N}} q_{ik} \left(\frac{M_i + o_i}{\lambda_{ik} f_k} + \sum_{m \in \mathcal{M}} p_{mi} t_{mi} \right) \leq H, \ \forall k \in \mathcal{M}. \tag{8}$$

Before executing a task τ_i, we need to collect all the data generated from its previous dependent tasks. If $p_{mi} = 1$, task τ_m precedes τ_i is τ_m is the closest task of τ_i, and, thus, in the worst case, the time spent to transmit the data required by the execution of task τ_i is $\sum_{m \in \mathcal{M}} p_{mi} t_{mi}$. $\boldsymbol{C}_3$ bounds the execution time of task τ_i cannot exceed the relative deadline d_i and $\boldsymbol{C}_4$ implies that all the tasks assigned to node θ_k must be executed within the hyper-period H.

For tractability reasons, when solving the above problem, we consider o_i as a continuous variables and then we round the result down. Since tasks execute typically hundreds of thousands of cycles, one cycle is a very fine-grained unit [12]. Due to the product terms $q_{i\beta}q_{j\gamma}$ and $q_{ik}o_i$ appear in $\boldsymbol{C}_2 - \boldsymbol{C}_4$, PP is an MINLP problem, which is $\mathcal{NP}$-hard.

3 Problem Linearization

Lemma 3.1 *Let b_1, b_2 and g denote the binary variables. The nonlinear constraint $g = b_1 b_2$ can be replaced by the following three linear constraints: $g \leq b_1$, $g \leq b_2$ and $g \geq b_1 + b_2 - 1$.*

Proof The inequalities $g \leq b_1$ and $g \leq b_2$ ensure that g will be zero if either b_1 or b_2 are zero. The inequality $g \geq b_1 + b_2 - 1$ makes sure that g takes the value of 1 if both variables b_1 and b_2 are set to 1. □

Lemma 3.2 *Given constants $s_1, s_2 > 0$ and two constraint spaces $S_1 = \{[h, b, x] | h = bx, -s_1 \leq x \leq s_2, b \in \{0, 1\}\}$ and $S_2 = \{[h, b, x] | -bs_1 \leq h \leq bs_2, h + bs_1 - x - s_1 \leq 0, h - bs_2 - x + s_2 \geq 0, b \in \{0, 1\}\}$, then $S_1 = S_2$.*

Proof Based on $h = bx$ and $-s_1 \leq x \leq s_2$, we have $-bs_1 \leq h \leq bs_2$. And further, we obtain $(b-1)(x+s_1) \leq 0$ and $(b-1)(x-s_2) \geq 0$ due to $-s_1 \leq x \leq s_2$ and $b \in \{0, 1\}$. Hence, we have $h + bs_1 - x - s_1 \leq 0$ and $h - bs_2 - x + s_2 \geq 0$. $S_1 \to S_2$ holds.

If $b = 0$, based on $-bs_1 \leq h \leq bs_2$, $h + bs_1 - x - s_1 \leq 0$ and $h - bs_2 - x + s_2 \geq 0$, we have $h = 0$ and $-s_1 \leq x \leq s_2$. For the same reason, if $b = 1$, we have $-s_1 \leq h = x \leq s_2$. $S_2 \to S_1$ holds. □

Based on Lemma 3.1 and Lemma 3.2, we propose a two-step linearization method. The details are as follows.

- **Step 1**: We replace each item $q_{i\beta}q_{i\gamma}$ with an *auxiliary* (binary) variable $g_{i\beta i\gamma}$ and add the following constraints into the problem formulation:

$$\boldsymbol{C}_5 : g_{i\beta j\gamma} \leq q_{i\beta},\ \forall i, j \in \mathcal{N},\ \forall \beta, \gamma \in \mathcal{M}, \tag{9}$$

$$\boldsymbol{C}_6 : g_{i\beta j\gamma} \leq q_{j\gamma},\ \forall i, j \in \mathcal{N},\ \forall \beta, \gamma \in \mathcal{M}, \tag{10}$$

$$\boldsymbol{C}_7 : g_{i\beta j\gamma} \geq q_{i\beta} + q_{j\gamma} - 1,\ \forall i, j \in \mathcal{N},\ \forall \beta, \gamma \in \mathcal{M}. \tag{11}$$

- **Step 2**: We replace each item $x_{ik}o_i$ with an *auxiliary* (continuous) variable h_{ik} and add the following constraints into the problem formulation:

$$\boldsymbol{C}_8 : h_{ik} \leq q_{ik}O_i,\ \forall i \in \mathcal{N},\ \forall k \in \mathcal{M}, \tag{12}$$

$$\boldsymbol{C}_9 : h_{ik} - o_i \leq 0,\ \forall i \in \mathcal{N},\ \forall k \in \mathcal{M}, \tag{13}$$

$$\boldsymbol{C}_{10} : h_{ik} - q_{ik}O_i - o_i + O_i \geq 0,\ \forall i \in \mathcal{N},\ \forall k \in \mathcal{M}. \tag{14}$$

Introducing the auxiliary variables $g_{i\beta i\gamma}$, h_{ik} and the additional constraints $\boldsymbol{C}_5 - \boldsymbol{C}_{10}$ in PP, this problem can be linearized as follow:

$$\mathbf{PP1}: \min_{\boldsymbol{q},\boldsymbol{g},\boldsymbol{o},\boldsymbol{h}} -\sum\nolimits_{i\in\mathcal{N}} f_i(o_i) \tag{15}$$
$$\text{s.t.} \begin{cases} \sum\limits_{k\in\mathcal{M}} q_{ik} = 1,\ \forall i \in \mathcal{N}, \\ q_{ik} = 1,\ \forall i, k \in \mathcal{M}_m, \\ E_k^{comm} + E_k^{comp} + E_k^s \le E_k^l,\ \forall k \in \mathcal{M}_s, \\ E_k^{comm} + E_k^{comp} + E_k^a \le E_k^l,\ \forall k \in \mathcal{M}_a, \\ E_k^{comm} + E_k^{comp} \le E_k^l,\ \forall k \in \mathcal{M},\ \forall k \notin \mathcal{M}_s, \mathcal{M}_a, \\ \sum\limits_{k\in\mathcal{M}} \dfrac{q_{ik}M_i + h_{ik}}{\lambda_{ik} f_k} \le d_i,\ \forall i \in \mathcal{N}, \\ \sum\limits_{i\in\mathcal{N}} \left(\dfrac{q_{ik}M_i + h_{ik}}{\lambda_{ik} f_k} + q_{ik} \sum\limits_{m\in\mathcal{M}} p_{mi} t_{mi} \right) \le H,\ \forall k \in \mathcal{M}, \\ \boldsymbol{C}_5 - \boldsymbol{C}_{10}, \\ q_{ik}, g_{i\beta j\gamma} \in Z^+, o_i, h_{ik} \in R^+, 0 \le o_i, h_{ik} \le O_i, \\ \forall i, j \in \mathcal{N},\ \forall k, \beta, \gamma \in \mathcal{M}. \end{cases}$$

Since binary variables $(\boldsymbol{q}, \boldsymbol{g})$ and continuous variables $(\boldsymbol{o}, \boldsymbol{h})$ are coupled with each other linearly, PP1 is an MILP problem, which is much easier to solve than the MINLP-based PP.

Remark 3.1 Since the linearization does not change the feasible region of the problem, and the objective functions of PP and PP1 are the same, solving PP1 is equivalent to solving PP, i.e., the optimal values of objective function of PP and PP1 are the same.

4 Optimal Task Mapping Algorithm

In this section, we propose an optimal algorithm OTM to efficiently solve PP1. Its structure is shown in Fig. 3. Using concise notions, the objective function and constraints of PP1 are rewritten as

$$\mathbf{PP2}: \min_{\boldsymbol{x},\boldsymbol{y}}\ \Phi = \boldsymbol{f}^T \boldsymbol{y} \tag{16}$$
$$\text{s.t.} \begin{cases} \boldsymbol{A}\boldsymbol{x} \preceq \boldsymbol{b}_1, \\ \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{y} \preceq \boldsymbol{b}_2, \end{cases}$$

where $\boldsymbol{x}$ ($\boldsymbol{y}$) is the vector of binary (continuous) variables. $\boldsymbol{f}$ is the vector of the objective function coefficient. $\boldsymbol{A}$, $\boldsymbol{C}$ and $\boldsymbol{D}$ are the constraint coefficient matrices. $\boldsymbol{b}_1$ ($\boldsymbol{b}_2$) is a u (v)-dimensional vector.

Instead of simultaneously solving the binary variables $\boldsymbol{x}$ and the continuous variables $\boldsymbol{y}$, OTM: 1) decomposes PP2 into two smaller subproblems with fewer variables: an ILP-based *Master Problem* (MP) with binary variables $\boldsymbol{x}$ and an LP-based *Slave Problem* (SP) with continuous variables $\boldsymbol{y}$, and 2) solves them by using the solution of one to the other. By doing so, the computational complexity can be significantly reduced.

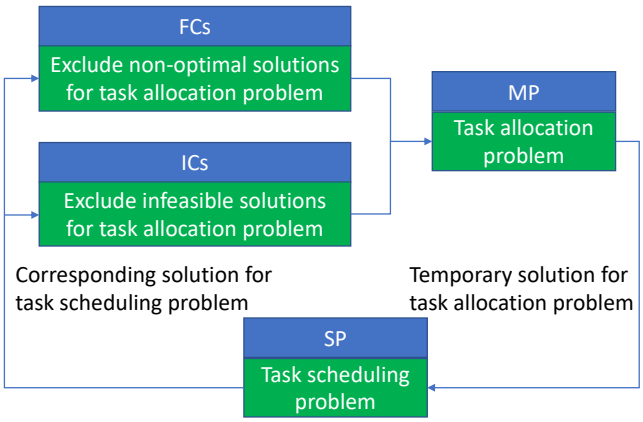

Fig. 3 The structure of OTM algorithm.

4.1 MP and SP formulation

The MP considers all the binary variables $\boldsymbol{x}$ and the corresponding part of the objective function and the constraints of PP2. It also includes a set of constraints called *Benders cuts*. The SP, on the other hand, is a LP incorporating the binary variables as parameters whose values are determined by solving the MP. Based on the structure of PP2, at the k^{th} iteration, the corresponding MP and SP are formulated as follows:

$$\mathbf{MP}: \Phi_l(k) = \min_{\boldsymbol{x},\hat{\Phi}} \ \hat{\Phi} \tag{17}$$
$$\text{s.t.} \begin{cases} \boldsymbol{A}\boldsymbol{x} \preceq \boldsymbol{b}_1, \\ \boldsymbol{C}_{11}: \hat{\Phi} \geq \boldsymbol{\lambda}(i)^T(\boldsymbol{C}\boldsymbol{x} - \boldsymbol{b}_2), \ \forall i \in \mathcal{A}, \\ \boldsymbol{C}_{12}: 0 \geq \hat{\boldsymbol{\lambda}}(j)^T(\boldsymbol{C}\boldsymbol{x} - \boldsymbol{b}_2), \ \forall j \in \mathcal{B}, \end{cases}$$

$$\mathbf{SP}: \Phi_u(k) = \min_{\boldsymbol{y} \succeq 0} \ \boldsymbol{f}^T\boldsymbol{y} \tag{18}$$
$$\text{s.t. } \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{y} \preceq \boldsymbol{b}_2,$$

where $\boldsymbol{x}(k)$ is the optimal solution of the MP at the k^{th} iteration. $\boldsymbol{C}_{11}$ is the set of *Feasibility Constraints* (FCs) and $\boldsymbol{C}_{12}$ is the set of *Infeasibility Constraints* (ICs). The constraints in $\boldsymbol{C}_{11}$ and $\boldsymbol{C}_{12}$ are Benders cuts. $\mathcal{A}$ and $\mathcal{B}$ are the sets of iterations that the dual of the SP (19) has the bounded and unbounded solution, respectively. Comparing SP with PP2, we observe that they have the same problem formulation with the exception that the binary variables $\boldsymbol{x}$ in the SP are fixed and given in advance.

Let Φ^* denote the optimal value of Φ. Note that MP is a minimization problem and the constraints regarding the continuous variables $\boldsymbol{y}$ are relaxed. Solving the MP yields a *lower bound* $\Phi_l(k)$ of Φ^*. On the other hand, as the values for binary variables $\boldsymbol{x}(k)$ may be just a feasible solution (not optimal yet), solving the SP yields an *upper bound* $\Phi_u(k)$ of Φ. The upper and lower bounds are updated by introducing new constraints into $\boldsymbol{C}_{11}$ and $\boldsymbol{C}_{12}$.

Remark 4.1 Since the objective function of the PP2 only contains the continuous variables $\boldsymbol{y}$, and the MP only considers the binary variables $\boldsymbol{x}$, an *auxiliary* variable (continuous) $\hat{\Phi}$, which has the same physical meaning as Φ, is introduced into the MP as the objective function, as well as to facilitate the iteration between the MP and the SP. Although the MP includes the continuous variable $\hat{\Phi}$, this problem can be solved by only considering the binary variables $\boldsymbol{x}$ [23].

4.2 Iterations Between MP and SP

At the initial iteration ($k = 0$), $\boldsymbol{C}_{11}$ and $\boldsymbol{C}_{12}$ are set to null. The lower and upper bounds are set to $\Phi_l(0) = -\infty$ and $\Phi_u(0) = +\infty$, respectively. The MP solution $\boldsymbol{x}(0)$ is given arbitrarily, as long as it satisfies the constraints $\boldsymbol{A}\boldsymbol{x}(0) \preceq \boldsymbol{b}_1$. We continue iterations until a stopping criterion $\Phi_u(k) - \Phi_l(k) \leq \epsilon$ is met, where ϵ is a small positive tolerance.

- **Step 1**: Solve the dual of the SP (DSP)

$$\begin{aligned} \textbf{DSP}: \max_{\boldsymbol{\lambda} \succeq 0} \ & \boldsymbol{\lambda}^T(\boldsymbol{C}\boldsymbol{x}(k) - \boldsymbol{b}_2) \\ \text{s.t.} \ & \boldsymbol{f} + \boldsymbol{D}'\boldsymbol{\lambda} \succeq 0. \end{aligned} \tag{19}$$

  where $\boldsymbol{\lambda} = [\lambda_i]_{v \times 1}$ are the Lagrange multipliers. Since DSP is a LP, it can be solved very fast using polynomial-time algorithms such as simplex method.
- **Step 2**: Based on the solution of the DSP, a new constraint is generated.
  Case 1: If DSP is *infeasible*, PP2 has no feasible solution.
  Case 2: If DSP has a *bounded solution* $\boldsymbol{\lambda}(k)$, SP is feasible under the given $\boldsymbol{x}(k)$ due to the strong duality [24]. Thus, we set $\mathcal{A} \leftarrow \{k\} \cup \mathcal{A}$, and the upper bound is updated by

$$\Phi_u(k+1) = \min\{\Phi_u(k), \boldsymbol{\lambda}(k)^T(\boldsymbol{C}\boldsymbol{x}(k) - \boldsymbol{b}_2)\}.$$

  Since $\Phi_u(k) - \Phi_l(k) > \epsilon$, $\boldsymbol{x}(k)$ is a non-optimal solution. To exclude the non-optimal solution $\boldsymbol{x}(k)$, a new FC:

$$\hat{\Phi} \geq \boldsymbol{\lambda}(k)^T(\boldsymbol{C}\boldsymbol{x} - \boldsymbol{b}_2) \tag{20}$$

  is generated and added into $\boldsymbol{C}_{11}$ at the $(k+1)^{th}$ iteration.
  Case 3: If DSP has an *unbounded solution*, i.e., $\boldsymbol{\lambda}(k)^T(\boldsymbol{C}\boldsymbol{x}(k) - \boldsymbol{b}_2) = +\infty$, SP is infeasible under the given $\boldsymbol{x}(k)$ due to the strong duality. Thus, we set $\mathcal{B} \leftarrow \{k\} \cup \mathcal{B}$. To exclude the infeasible solution $\boldsymbol{x}(k)$, a new IC:

$$0 \geq \hat{\boldsymbol{\lambda}}(k)^T(\boldsymbol{C}\boldsymbol{x} - \boldsymbol{b}_2) \tag{21}$$

  is generated and added into $\boldsymbol{C}_{12}$ at the $(k+1)^{th}$ iteration, where $\hat{\boldsymbol{\lambda}}(k)$ is the optimal solution of DFCP (23) at the k^{th} iteration.
- **Step 3**: With a new FC or IC added into MP at the $(k+1)^{th}$ iteration, the MP is solved again to obtain a new solution $\boldsymbol{x}(k+1)$ for the next iteration.

4.3 Theoretical analysis

Theorem 4.1 *The non-optimal and infeasible values of the binary variables $\boldsymbol{x}$ are excluded by the constraints $\boldsymbol{C}_{11}$ and $\boldsymbol{C}_{12}$.*

Proof Let $\hat{\Phi}(k)$ denote the optimal solution of MP at the k^{th} iteration. In *Case 2*, since $\boldsymbol{x}(k)$ is not the optimal solution of PP2, we have $\hat{\Phi}(k) < \boldsymbol{\lambda}(k)^T(\boldsymbol{C}\boldsymbol{x}(k) - \boldsymbol{b}_2)$. Thus, the non-optimal solution $\boldsymbol{x}(k)$ can be excluded by the constraint $\hat{\Phi} \geq \boldsymbol{\lambda}(k)^T(\boldsymbol{C}\boldsymbol{x} - \boldsymbol{b}_2)$, i.e., we have (20).

In *Case 3*, the SP has no feasible solution under the given solution $\boldsymbol{x}(k)$. This problem may be feasible if the positive variables $\boldsymbol{\xi} = [\xi_i]_{v\times 1}$ are introduced to relax the constraints $\boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{y} \preceq \boldsymbol{b}_2$. Based on this idea, we construct a Feasibility Check Problem (FCP):

$$\begin{aligned} \mathbf{FCP}: \min_{\boldsymbol{y},\boldsymbol{\xi}\succeq 0} \ & \mathbf{1}^T\boldsymbol{\xi} \\ \text{s.t.} \ & \boldsymbol{C}\boldsymbol{x}(k) + \boldsymbol{D}\boldsymbol{y} \preceq \boldsymbol{b}_2 + \boldsymbol{\xi}, \end{aligned} \tag{22}$$

and solve its dual problem (DFCP):

$$\begin{aligned} \mathbf{DFCP}: \max_{\hat{\boldsymbol{\lambda}}\succeq 0} \ & \hat{\boldsymbol{\lambda}}^T(\boldsymbol{C}\boldsymbol{x}(k) - \boldsymbol{b}_2) \\ \text{s.t.} \ & \begin{cases} \mathbf{1} - \hat{\boldsymbol{\lambda}} \succeq 0, \\ \boldsymbol{D}^T\hat{\boldsymbol{\lambda}} \succeq 0, \end{cases} \end{aligned} \tag{23}$$

where $\hat{\boldsymbol{\lambda}} = [\hat{\lambda}_i]_{v\times 1}$ are the Lagrange multipliers.

Since DFCP is a LP, it can be solved using the similar method for solving DSP. Let $\boldsymbol{\xi}(k)$ denote the optimal solution of FCP at the k^{th} iteration. If the SP is infeasible, some constraints in (18) cannot be satisfied, and, thus, their related relaxation variables are non-zero. We have $\mathbf{1}^T\boldsymbol{\xi}(k) > 0$, and further, $\mathbf{1}^T\boldsymbol{\xi}(k) = \hat{\boldsymbol{\lambda}}(k)^T(\boldsymbol{C}\boldsymbol{x}(k) - \boldsymbol{b}_2) > 0$ due to the strong duality. Thus, the infeasible solution $\boldsymbol{x}(k)$ can be excluded by the constraint $0 \geq \hat{\boldsymbol{\lambda}}(k)^T(\boldsymbol{C}\boldsymbol{x} - \boldsymbol{b}_2)$, i.e., we have (21). □

Theorem 4.2 *The FC and IC generated by solving the DSP with $\boldsymbol{x}'(k)$ do not exclude the optimal solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ of PP2, where $\boldsymbol{x}'(k)$ is an arbitrary feasible solution of the MP.*

Proof If the DSP has a bounded solution $\boldsymbol{\lambda}'(k)$ with $\boldsymbol{x}'(k)$, a new FC:

$$\hat{\Phi} \geq \boldsymbol{\lambda}'(k)^T(\boldsymbol{C}\boldsymbol{x} - \boldsymbol{b}_2) \tag{24}$$

is generated. On the other hand, if the DSP has an unbounded solution with $\boldsymbol{x}'(k)$, a new IC:

$$0 \geq \hat{\boldsymbol{\lambda}}'(k)^T(\boldsymbol{C}\boldsymbol{x} - \boldsymbol{b}_2) \tag{25}$$

is generated, where $\hat{\boldsymbol{\lambda}}'(k)$ is the solution of DFCP with $\boldsymbol{x}'(k)$ at the k^{th} iteration.

Next, we prove that the solution $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ does not violate the constraints (24) and (25). If the DSP has a bounded solution $\boldsymbol{\lambda}'(k)$ with $\boldsymbol{x}'(k)$, suppose that $\boldsymbol{x}^*$ and $\Phi^* = \boldsymbol{f}^T\boldsymbol{y}^*$ violate the constraint (24), i.e., $\Phi^* < \boldsymbol{\lambda}'(k)^T(\boldsymbol{C}\boldsymbol{x}^* - \boldsymbol{b}_2)$. This

contradicts the fact that Φ^* is the optimal value of the objective function of the DSP with $\boldsymbol{x}^*$:

$$\Phi^* = \max_{\boldsymbol{\lambda} \succeq 0} \boldsymbol{\lambda}(k)^T(\boldsymbol{C}\boldsymbol{x}^* - \boldsymbol{b}_2) \geq \boldsymbol{\lambda}'(k)^T(\boldsymbol{C}\boldsymbol{x}^* - \boldsymbol{b}_2).$$

Hence, the FC does not exclude the optimal solution $\boldsymbol{x}^*$. If the DSP has an unbounded solution with $\boldsymbol{x}'(k)$, $\boldsymbol{x}'(k)$ is excluded by the constraint (25). Since $\boldsymbol{x}^* \neq \boldsymbol{x}'(k)$, $\boldsymbol{x}^*$ does not violate the constraint (25). □

Remark 4.2 Since MP is an ILP, this problem is still hard to solve directly, compared with the LP-based SP. Moreover, a new FC or IC is added into the MP at each iteration. The size of MP will increase with the number of iterations. Therefore, the computational complexity of OTM is dominated by the cost of solving the MP at each iteration. Based on Theorem 4.2, we can circumvent the above difficulties by replacing the optimal solution of MP $\boldsymbol{x}(k)$ with the feasible solution $\boldsymbol{x}'(k)$ during the iteration between the MP and the SP. Such a feasible solution can be efficiently found by using the heuristics such as Feasibility Pump (FP) method [25].

Theorem 4.3 *At each iteration with a new FC* (24) *or IC* (25) *added into the MP, the solution obtained by OTM converges to the global optimal value Φ^* within a finite number of iterations.*

Proof Based on Theorem 4.1 and Theorem 4.2, as well as the fact that the dimension of binary variables $\boldsymbol{x}$ is finite, the solution converges to the global optimal value within a finite number of iterations. □

Remark 4.3 Since SP and DSP are equivalent due to the strong duality, and FCs and ICs can be constructed by the solution of DSP, we solve DSP instead of solve SP. Similarly, since FCB and DFCB are equivalent due to the strong duality, and $\hat{\boldsymbol{\lambda}}^T(\boldsymbol{C}\boldsymbol{x} - \boldsymbol{b}_2)$ is a function with respect to variables $\boldsymbol{x}$, but not $\mathbf{1}^T\boldsymbol{\xi}$ (i.e., $0 \geq \mathbf{1}^T\boldsymbol{\xi}$ is an invalid constraint for the MP), we solve DFCB instead of solve FCB.

Algorithm 1 summarizes the implementation details of OTM algorithm.

5 Simulation Results

For evaluating our approach, we use an indoor environment management application for monitoring Heating, Ventilation, and Air Conditioning (HVAC) [1,17]. We consider a set of nodes spread across a room, with some nodes equipped with temperature sensors and actuators that are able to control the temperature of a region. The aim of the system is to maintain a desirable temperature level at some points of interesting in the room, by correlating and processing the data from the sensors installed in the room, and using it to adjust the outputs of the actuators.

The processor parameters are adopted from [10] and listed in Table 3. We assume that the nodes attached to actuators operate with V/F levels of big core, while the other nodes operate with V/F levels of LITTLE core, as the actuator nodes usually have higher capabilities than the sensor nodes. The Worst Case Execution Cycles (WCECs) of M_i and O_i are assumed to be in the range of

Algorithm 1: Optimal Task Allocation (OTM) Algorithm

Input: $\boldsymbol{A}, \boldsymbol{C}, \boldsymbol{D}, \boldsymbol{f}, \boldsymbol{b}_1, \boldsymbol{b}_2$;
Output: $\boldsymbol{x}^*, \boldsymbol{y}^*$;
Set initial values: $k = 0$, $\Phi_l(0) = -\infty$, $\Phi_u(0) = +\infty$, $\{\boldsymbol{x}(0) | \boldsymbol{A}\boldsymbol{x}(0) \preceq \boldsymbol{b}_1\}$, ϵ;
$\boldsymbol{C}_{11}$ and $\boldsymbol{C}_{12}$ are set to null;
while $|\Phi_u(k) - \Phi_l(k)| > \epsilon$ **do**
  Solve MP (17) to obtain $\boldsymbol{x}'(k)$ and $\Phi_l(k)$;
  Solve DSP (19) with $\boldsymbol{x}'(k)$;
  if *DSP has bounded or unbounded solution* **then**
    if *Solution is bounded* **then**
      $\mathcal{A} \leftarrow \{k\} \cup \mathcal{A}$;
      Add constraint (24) to $\boldsymbol{C}_{11}$;
      $\Phi_u(k+1) = \min\{\Phi_u(k), \boldsymbol{\lambda}'(k)^T(\boldsymbol{C}\boldsymbol{x}'(k) - \boldsymbol{b}_2)\}$;
    else
      $\mathcal{B} \leftarrow \{k\} \cup \mathcal{B}$;
      Add constraint (25) to $\boldsymbol{C}_{12}$;
    end
  else
    PP2 (16) is infeasible;
  end
  $k \leftarrow k + 1$;
end
$\boldsymbol{x}^* = \boldsymbol{x}(k)$;
Submit $\boldsymbol{x}^*$ into PP2 to solve $\boldsymbol{y}^*$;

Table 3 V/F levels and power model parameters of a big core and a little core

big									
v_k (V)	0.93	0.96	1.0	1.04	1.08	1.1	1.15	1.2	1.23
f_k (GHz)	0.8	0.9	1.0	1.1	1.2	1.3	1.4	1.5	1.6
$C_k^s = 1.478$, $C_k^d = 0.471$, $\rho_k = 0.379$									
LITTLE									
v_k (V)	0.9	0.94	1.01	1.09	1.2	$C_k^s = 1.191$, $C_k^d = 0.153$			
f_k (GHz)	0.25	0.3	0.4	0.5	0.6	$\rho_k = 0.757$			

$[4 \times 10^7, 6 \times 10^8]$ [16]. The QoS function is set to $\sum_{i \in \mathcal{N}} f_i(o_i) = \sum_{i \in \mathcal{N}} o_i$ [15]. The relative deadline of task τ_i and the scheduling horizon are assumed to be $d_i = \min_{\forall k}\{\frac{M_i + O_i}{f_k}\}$ and $H = \sum_{i \in \mathcal{N}} d_i$, respectively. The available energy of node θ_k at the l^{th} round is set to $E_k^l = \eta E_k^h$, where $E_k^h = p_k^c \frac{\sum_{i \in \mathcal{N}}(M_i + O_i)}{f_k}$ is the minimum energy required to execute all the tasks and $\eta = 0.9$ is an energy efficiency factor. The parameters of energy consumption of transmitting E_{tx} and receiving E_{rx} are set to $E_{elec} = 50$ nJ/b, $\varepsilon_{amp} = 10$ pJ/b/m^2 and $d_{th} = 10$ m [2]. The nodes are randomly deployed. Each node can communicate with the nodes within its communication range and the network is assumed to be fully connected. The routing energy cost matrix $\boldsymbol{r}$ and the communication time matrix $\boldsymbol{t}$ are obtained by using a shortest path algorithm on the network, assuming equal energy spent by all nodes on a route, and all data items are assumed to be of unit size ($s_{ij} = 1$) [17]. We set $M = 20$, $M_s = 10$, $M_a = 5$ and randomly generate DAGs with 30, 35, 40, 45 and 50 tasks.

Note that different processor and task parameters lead to different values in the parameters $\boldsymbol{A}, \boldsymbol{C}, \boldsymbol{D}, \boldsymbol{f}, \boldsymbol{b}_1, \boldsymbol{b}_2$ for the PP2. However, the problem structures under

different values of parameters are the same, and, thus, the proposed methods are still applicative.

In this section, we present the following evaluation results:

- The QoS, the convergence and the computation time of the proposed OTM algorithm with optimal approach: Branch and Bound method (B&B) [26] and evolutionary approach: Genetic Algorithm (GA) [27].
- The QoS and the energy consumption of the OTM solving another IC-task mapping problem: Energy-Optimization (NRG-OPT).

$$\textbf{NRG-OPT}: \min_{\boldsymbol{q},\boldsymbol{g},\boldsymbol{o},\boldsymbol{h}} E^{all} \tag{26}$$
$$\text{s.t.} \begin{cases} \sum_{k\in\mathcal{M}} q_{ik} = 1,\ \forall i \in \mathcal{N}, \\ q_{ik} = 1,\ \forall i, k \in \mathcal{M}_m, \\ \sum_{k\in\mathcal{M}} \frac{q_{ik}M_i + h_{ik}}{\lambda_{ik} f_k} \le d_i,\ \forall i \in \mathcal{N}, \\ \sum_{i\in\mathcal{N}} \left(\frac{q_{ik}M_i + h_{ik}}{\lambda_{ik} f_k} + q_{ik} \sum_{m\in\mathcal{M}} p_{mi} t_{mi} \right) \le H,\ \forall k \in \mathcal{M}, \\ \boldsymbol{C}_5 - \boldsymbol{C}_{10}, \\ q_{ik}, g_{i\beta j\gamma} \in Z^+, o_i, h_{ik} \in R^+, 0 \le o_i, h_{ik} \le O_i, \\ \forall i, j \in \mathcal{N},\ \forall k, \beta, \gamma \in \mathcal{M}. \end{cases}$$

where $E^{all} = \sum_{k\in\mathcal{M}_s}(E_k^{comm} + E_k^{comp} + E_k^s) + \sum_{k\in\mathcal{M}_a}(E_k^{comm} + E_k^{comp} + E_k^a) + \sum_{k\in\mathcal{M},k\notin\mathcal{M}_s,\mathcal{M}_a}(E_k^{comm} + E_k^{comp})$ is the total energy consumption of the nodes.

The simulations are performed on a laptop with quad-core 2.5 GHz Intel i7 processor and 16 GB RAM, and the algorithms are implemented in Matlab 2016a.

The QoS of using OTM and B&B to solve PP1 and using GA to solve PP1 under different task number N is shown in Fig. 4(a). From it we observe that the solutions achieved by OTM and B&B are same, and, thus, OTM also finds the optimal solution. Moreover, the QoS achieved by OTM is higher than GA. Although GA is able to solve complex non-linear programming problem, such as MINLP, the optimality of the solution is hard to guarantee [28].

The computation time of OTM, B&B and GA under different task number N is evaluated in Fig. 4(b). We observe that with N increasing, the computation time of OTM, B&B and GA both grows. However, OTM has a shorter computation time than B&B and GA. This is because for an optimization problem, its computational complexity increases significantly with the number of variables and constraints. Thus, solving smaller problems iteratively with less variables (i.e., MP and SP) is more efficient than solving a single large problem. This result is in line with the comparison in [29]. Moreover, compared with OTM, the structure of GA is more complex. Thus, GA has a longer computation time. From Fig. 4, we can see that the problem transformation from MINLP-based PP to MILP-based PP1 is necessary, as it can simplify the problem structure and make the optimal solution easier to find.

Fig. 5(a) shows the convergence of OTM algorithm, with task number $N = 30$. From it we observe that by introducing the feasibility and infeasibility constraints

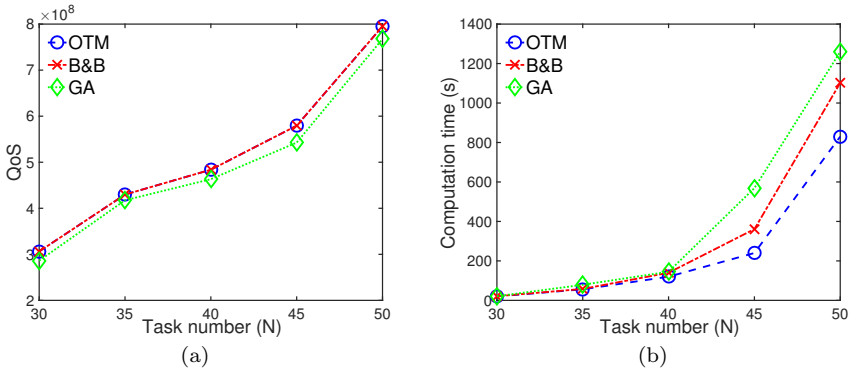

Fig. 4 (a) QoS of OTM, B&B and GA with N varying; (b) Computation time of OTM, B&B and GA with N varying.

into the MP during the iterations $\{1, 2, 4, 5\}$ and $\{3, 6\}$, respectively, the upper bound $\theta_u(k)$ and the lower bound $\theta_l(k)$ quickly converge to the optimal value θ^*. Fig. 5(b) shows the number of iterations required by OTM to converge under different task number N. The result shows that the convergence iteration of OTM almost increases linearly with N. An implementation detail about OTM is that when the gap between the upper and lower bounds is small, the convergence speed is slow. To deal with this limitation, we apply the classical BD when the gap close to zero.

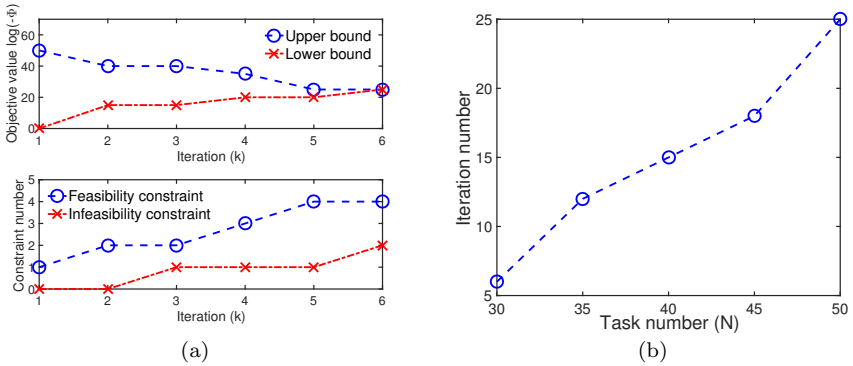

Fig. 5 (a) Convergence of OTM; (b) Convergence iteration of OTM with N varying.

Fig. 6 explores the performance (QoS and computation time) of OTM with the stopping criteria ϵ varying. We set task number $N = 50$ and change the value of ϵ between the range of $[0.1, 1, 10, 100]$. We assume that when $\epsilon = 0.1$, the optimal solution is found. On the other hand, if we select $\epsilon \neq 0.1$, the iterations between the MP and the SP will continue until the gap between $\theta_u(k)$ and $\theta_l(k)$ is smaller than ϵ and the DSP is feasible under the solution of the MP ($\boldsymbol{x}(k)$). Thus, the

value of ϵ can be used to control the quality of the solution. From Fig. 6, we observe that the QoS and computation time decrease when ϵ increases and the convergence speed of OTM are very fast at the beginning of the iteration process. If we just want to find a feasible solution, we can run OTM and stop the iteration when the fist time that the DSP has a bounded solution (assume that at the k^{th} iteration). Thus, the optimal task adjustment decision $\boldsymbol{y}(k)$ is found under the given task allocation decision $\boldsymbol{x}(k)$.

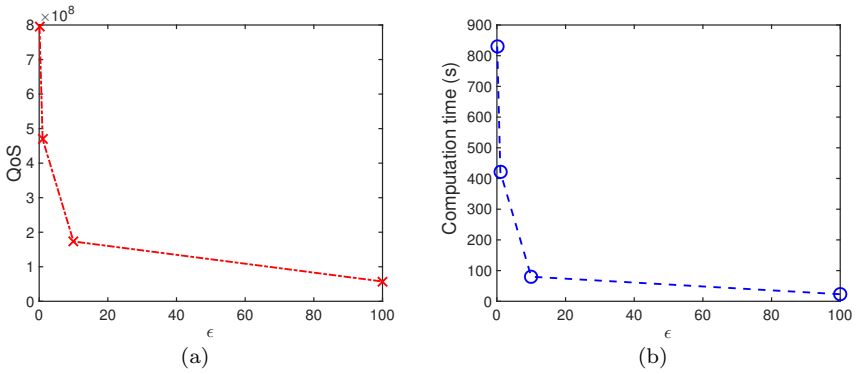

Fig. 6 (a) QoS of OTM with ϵ varying; (b) Computation time of OTM with ϵ varying.

Fig. 7(a) compares the QoS and the energy consumption of QoS-OPT (15) with NRG-OPT (26) under different task number N. From Fig. 7(a), we observe that when using NRG-OPT to perform IC-task mapping, the QoS is always equal to 0, which is obviously lower than the QoS achieved by QoS-OPT. This is because if the mandatory subtask of a task is fixed, the smaller the optional subtask, the lower the energy consumed to execute this task. On the other hand, Fig. 7(b) shows that QoS-OPT consumes more energy than NRG-OPT, as QoS-OPT maximizes QoS and therefore executes more optional subtasks than NRG-OPT. However, the consumed energy of QoS-OPT is always no more than the supplied energy $\sum_{k\in\mathcal{M}} E_k^l$, as the energy constraint $\boldsymbol{C}_2$ must be satisfied. Thus, using QoS-OPT to perform IC-task mapping provides a better balance between QoS-enhancing and energy-utilizing.

6 Conclusion

In this paper, we address the problem of IC-tasks mapping on CPS, with the goal of maximizing system QoS without violating the real-time and the energy constraints. We first formulate an MINLP model to describe this joint-design problem. And then, we propose an MILP formulation of this model without performance degradation. By doing so, we avoid high computational complexity and we use a simpler model to find an optimal solution. To efficiently solve the MILP problem, we propose an OTM algorithm. It reduces the computational complexity by iterating two smaller, but highly correlated, subproblems: an MP for task allocation,

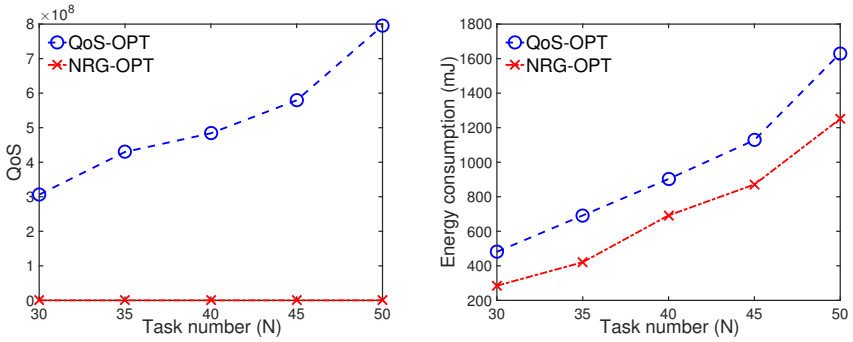

Fig. 7 (a) QoS of QoS-OPT and NRG-OPT with N varying; (b) Energy consumption of QoS-OPT and NRG-OPT with N varying.

and a SP for task adjustment. We prove that the proposed algorithm converges to the optimal solution through finite number of iterations. The simulation results showed the effectiveness of our proposed algorithm, which outperforms other algorithms with respect to QoS-enhancing and energy-utilizing.

References

1. L. Mo, X.H. Cao, Y.Q. Song and A. Kritikakou, "Distributed node coordination for real-time energy-constrained control in wireless sensor and actuator networks," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 151–4163, 2018.
2. Y. Tian and E. Ekici, "Cross-layer collaborative in-network processing in multihop wireless sensor networks," *IEEE Trans. Mobile Comput.*, vol. 6, no. 3, pp. 297–310, 2006.
3. H. Zahaf and A. E. H. Benyamina, R. Olejnik and G. Lipari, "Energy-efficient scheduling for moldable real-time tasks on heterogeneous computing platforms," *Journal of Systems Architecture*, vol. 74, pp. 46–60, 2017.
4. H. Aydin, R. Melhem, D. Mosse and P. Mejia-Alvarez, "Optimal reward-based scheduling for periodic real-time tasks," *IEEE Trans. Comput.*, vol. 50, no. 2, pp. 111–130, 2001.
5. J. W. S. Liu, W. K. Shih, K. J. Lin, R. Bettati and J. Y. Chung, "Imprecise computations," *Proc. IEEE*, vol. 82, no. 1, pp. 83–94, 1994.
6. H. Yu, Y. Ha and B. Veeravalli, "Quality-driven dynamic scheduling for real-time adaptive applications on multiprocessor systems," *IEEE Trans. Comput.*, vol. 62, no. 10, pp. 2026–2040, 2013.
7. L. Leung, C. Tsui and W. Ki, "Simultaneous task allocation, scheduling and voltage assignment for multiple-processors-core systems using mixed integer nonlinear programming," *Proc. IEEE International Symposium on Circuits and Systems*, pp. 309–312, 2003.
8. G. Chen, K. Huang and A. Knoll, "Energy optimization for real-time multiprocessor system-on-chip with optimal DVFS and DPM combination," *ACM Trans. Embed. Comput. Syst.*, vol. 13, no. 3, pp. 111:1–111:21, 2014.
9. D. Li and J. Wu, "Minimizing energy consumption for frame-based tasks on heterogeneous multiprocessor platforms," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 3, pp. 810–823, 2015.
10. H. S. Chwa and J. Seo and J. Lee and I. Shin, "Optimal real-time scheduling on two-type heterogeneous multicore platforms," *Proc. IEEE Real-Time Systems Symposium*, pp. 119–129, 2015.
11. A. Emeretlis, G. Theodoridis, P. Alefragis and N. Voros, "A logic-based Benders decomposition approach for mapping applications on heterogeneous multicore platforms," *ACM Trans. Embed. Comput. Syst.*, vol. 15, no. 1, pp. 1539–9087, 2016.

12. L. A. Cortes, P. Eles and Z. Peng, "Quasi-static assignment of voltages and optional cycles in imprecise-computation systems with energy considerations," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 14, no. 10, pp. 1117–1129, 2006.
13. R.C. Ravindran, C. M. Krishna, I. Koren and Z. Koren , "Scheduling imprecise task graphs for real-time applications," *International Journal of Embedded Systems*, vol. 6, no. 1, pp. 73–85, 2014.
14. I. Mendez-Diaz, J. Orozco, R. Santos and P. Zabala, "Energy-aware scheduling mandatory/optional tasks in multicore real-time systems," *International Transactions in Operational Research*, vol. 24, no. 12, pp. 173–198, 2017.
15. L. Mo, A. Kritikakou and O. Sentieys, "Energy-quality-time optimized task mapping on DVFS-enabled multicores," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2428–2439, 2018.
16. T. Wei, J. Zhou, K. Cao, P. Cong, M. Chen, G. Zhang, X. S. Hu and J. Yan, "Cost-constrained QoS optimization for approximate computation real-time tasks in heterogeneous MPSoCs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 9, pp. 1733–1746, 2018.
17. A. Pathak and V. K. Prasanna, "Energy-efficient task mapping for data-driven sensor network macroprogramming," *IEEE Trans. Comput.*, vol. 59, no. 7, pp. 955–967, 2010.
18. A. Voinescu, D. S. Tudose and N. Tapus, "Task scheduling in wireless sensor networks," *Proc. IEEE International Conference on Networking and Services*, pp. 12–17, 2010.
19. L. Dai, Y. Chang and Z. Chen, "An optimal task scheduling algorithm in wireless sensor networks," *International Journal of Computers Communications and Control*, vol. 11, no. 1, pp. 101–112, 2011.
20. B. Billet and V. Issarny, "From task graphs to concrete actions: a new task mapping algorithm for the future Internet of Things," *Proc. IEEE International Conference on Mobile Ad Hoc and Sensor Systems*, pp. 470–478, 2014.
21. J. F. Benders, "Partitioning procedures for solving mixed-variables programming problems," *Numerische Mathematik*, vol. 4, no. 1, pp. 238–252, 1962.
22. K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks," *Ad Hoc Networks*, vol. 3, no. 3, pp. 325–349, 2005.
23. L. Mo, A. Kritikakou and O. Sentieys, "Decomposed task mapping to maximize QoS in energy-constrained real-time multicores," *IEEE International Conference on Computer Design*, pp. 493–500, 2017.
24. S. Boyd and L. Vandenberghe, "Convex optimization," *Cambridge University Press*, 2004.
25. M. Fischetti, F. Glover and A. Lodi, "The feasibility pump," *Math. Program.*, vol. 104, no. 1, pp. 91–104, 2005.
26. S. Boyd, A. Ghosh and A. Magnani, "Branch and bound methods," *Notes for EE364b, Stanford University*, pp. 1–11, 2007.
27. E. Rothberg, "An evolutionary algorithm for polishing mixed integer programming solutions," *INFORMS Journal on Computing*, vol. 19, no. 4, pp. 534–541, 2007.
28. K. Genova and V. Guliashki, "Linear integer programming methods and approaches - a survey," *Cybernetics and Information Technologies*, vol. 11, no. 1, pp. 1–23, 2011.
29. C. D. Randazzo and H. P. L. Luna, "A comparison of optimal methods for local access uncapacitated network design," *Annals of Operations Research*, vol. 106, no. 1, pp. 263–286, 2001.