The VirtFogSim Simulator

A quick guide

Enzo Baccarelli [*] , **Michele Scarpiniti** , **Alireza Momenzadeh**

February 2019, Rome
Version 4.0

[*] **Correspondence: Enzo Baccarelli**
Department of Information Engineering, Electronics and Telecommunications (DIET), Sapienza University of Rome
Web page: http://enzobaccarelli.site.uniroma1.it
Email: enzo.baccarelli@uniroma1.it

FOREWORD: The virtual Fog Simulator (VirtFogSim) simulates the task and resource allocation in a three-tier networked virtualized computing infrastructure composed by a Mobile device, a first virtualized clone hosted by a proximate Fog data center (i.e. the Fog clone), and a second virtualized clone hosted by a remote Cloud center (i.e. the Cloud clone). The simulator allows the user to set 67 input parameters, in order to properly configure the desired mobile Fog scenario to be simulated.
The software platform utilized for the *VirtFogSim* development is MATLAB. The overall simulator script is composed by about 5,000 lines of MATLAB code. The MATLAB release 2015 (or later) is recommended for running the *VirtFogSim* code.
The VirtFogSim toolkit has been developed by prof. Enzo Baccarelli and Michele Scarpiniti at the "Information, Electronic and Telecommunication Department"(DIET) of "Sapienza" University of Rome (Italy). It has been partially supported by the PRIN2015 project no. 2015YPXH4W_004: "A green adaptive Fog computing and networking architecture (GAUChO)", funded by the Italian MIUR. It has been developed under the research activity planned for the fourth work-package (namely WP4) of the GAUChO project (see the project web site at: www.gaucho.unifi.it).

1 VIRTFOGSIM – THE CONSIDERED FOG NETWORKED INFRASTRUCTURE

The reference three-tier Mobile-Fog-Cloud virtualized networked computing architecture is sketched in Fig. 1. It is composed by a virtualized Mobile device, a proximate virtualized Fog node, and a remote virtualized Cloud node. These nodes inter-communicate through Transport-layer TCP/IP multipath connections. Specifically (see Fig. 1):

i. Mobile-Fog communication is supported by a two-way WiFi-based (possibly mobile) single-hop TCP/IP connection. R_U (bit/s) (respectively R_D (bit/s)) is the steady-state throughput of the Mobile-to-Fog (respectively Fog-to-Mobile) upstream (respectively downstream) TCP/IP one-way connection. R_U^{MAX} (bit/s) and R_D^{MAX} (bit/s) are the corresponding maximal per-connection throughputs which, in turn, depend on the adopted WiFi technology and the size of the TCP maximum congestion window;

ii. Mobile-Cloud communication is supported by a two-way 3G/4G cellular (possibly mobile) single-hop TCP/IP connection. B_U (bit/s) (respectively B_D (bit/s)) is the steady-state throughput of the Mobile-to-Cloud (respectively Cloud-to-Mobile) upstream (respectively downstream) TCP/IP one-way connection. B_U^{MAX} and B_D^{MAX} (bit/s) are the corresponding maximal per-connection throughputs which, in turn, depend on the adopted 3G/4G cellular technology and the size of the TCP maximum congestion window;

iii. Cloud-Fog communication is supported by a two-way (possibly multi-hop) backbone TCP/IP connection. R_0 (bit/s) is the steady-state throughput of the one-way Fog-to-Cloud and Cloud-to-Fog connections.

In order to support these connections, we require that:

i. the Mobile device is equipped with a WiFi Network Interface Card (NIC) and a Cellular NIC;

ii. the Fog node is equipped with a WiFi NIC and an Ethernet NIC;

iii. the Cloud node is equipped with a Cellular NIC and an Ethernet NIC;

iv. the protocol stack of multipath TCP (MTCP) equips the mobile device, in order to simultaneously manage mobile-cloud and mobile-fog up/down transport-layer connections, featuring the multi-homing environment of Fig. 1.

The Mobile device, Fog node, and Cloud node are also equipped with virtual (possibly multi-core) processors that run as virtual containers. f_M (bit/s), f_F (bit/s), and f_C (bit/s) are respectively the per-core processing frequencies at the Mobile, Fog, and Cloud nodes, while f_M^{MAX} (bit/s), f_F^{MAX} (bit/s), and f_C^{MAX} (bit/s) are the corresponding maximal per-core processing frequencies.

The Mobile device must run a given application that is composed by multiple inter-connected tasks, i.e. subroutines or methods. The Mobile device may decide to execute each task locally or offload it to the Fog node or Cloud node due to the fact that the Mobile device is energy limited and equipped with restricted computing resources. In this regard, the Fog node (respectively the Cloud node) is virtualized in order to host a Fog clone (respectively a Cloud clone) of the Mobile device. These clones act as virtual processors and execute the offloaded tasks on behalf of the Mobile device. At the Application layer, inter-task communication exploit the (aforementioned) Mobile-Fog, Mobile-Cloud, and Cloud-Fog TCP/IP connections. It is assumed that the overall protocol stacks of these clones are already stored by the Cloud and Fog nodes.

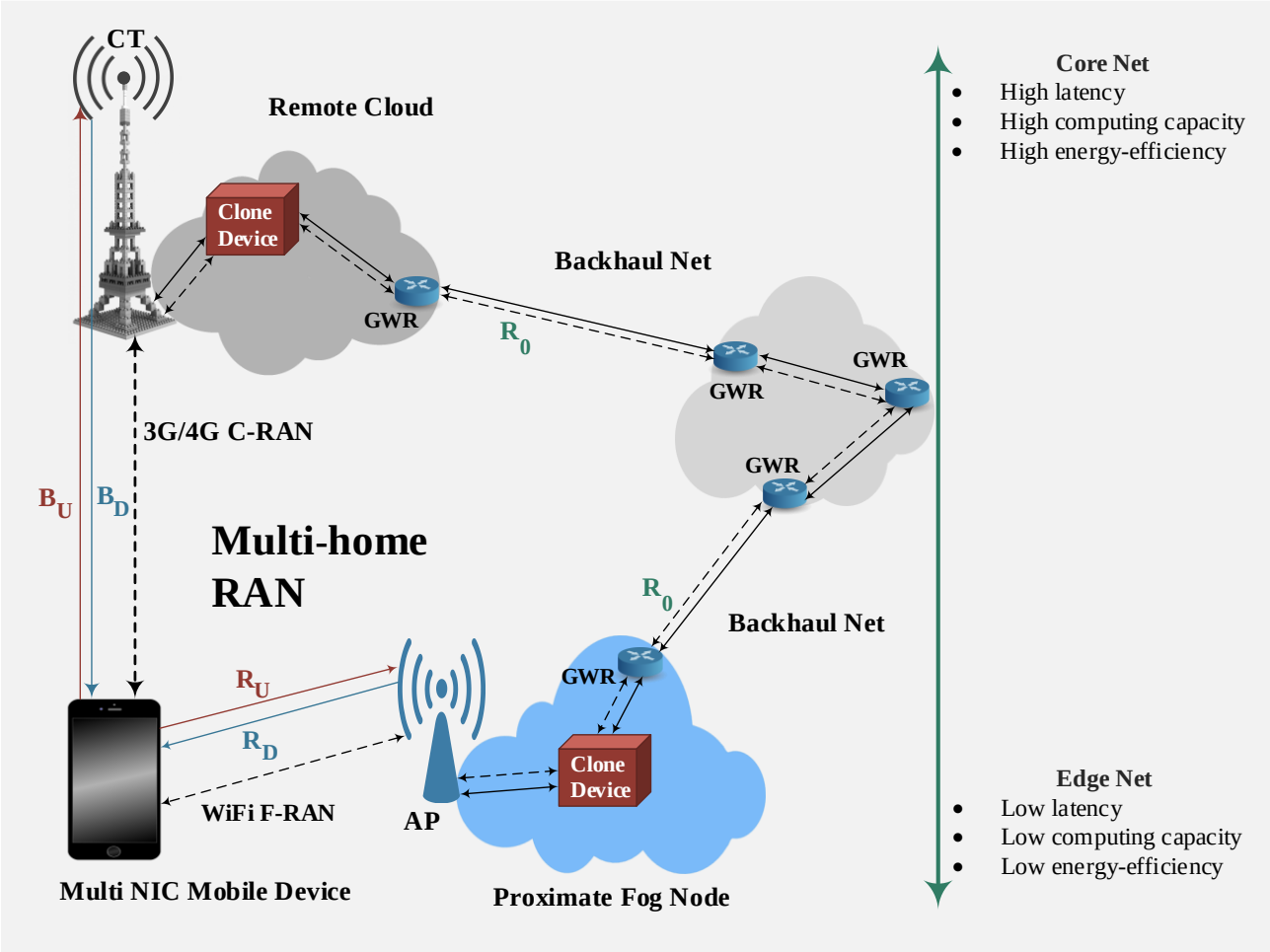

FIGURE 1. The multi-homing *VirtFogSim* reference architecture. WAN: Wide Area Network; LAN: Local Area Network; CT: Cellular Tower; AP: Access Point; GWR: GateWay Router. Single-arrowed (respectively double-arrowed) paths indicate one-way (respectively two-way) single-path TCP/IP connections. Multipath TCP is used by the mobile device for orchestrating the mobile-fog and mobile-cloud transport layer connections.

1.1 THE CONSIDERED CONTAINER-BASED VIRTUALIZATION OF THE COMPUTING NODES

Virtualization is employed in Cloud-based and Fog-based data centers, in order to:

i. dynamically multiplex the available physical computing, storage, and networking resources over the spectrum of the served mobile devices;

ii. provide homogeneous user interface atop (possibly) heterogeneous served devices;

iii. isolate the applications running atop a same physical server, so to provide trustworthiness.

Roughly speaking, in virtualized data centers, each served physical device is mapped into a virtual clone which acts as a virtual processor and executes the programs on behalf of the cloned device. In principle, two main virtualization technologies could be used to attain device virtualization, namely the (more traditional) Virtual Machine (VM)-based technology, and the (emerging) CoNTainer (CNT)-based technology. In a nutshell, their main architectural differences are:

i. the VM technology relies on a Middleware software layer (e.g. the so-called Hypervisor) that statically performs hardware virtualization while the CNT technology uses an Execution Engine in order to dynamically carry out resource scaling and multiplexing.

ii. a VM is equipped with an own (typically heavy-weight) Guest Operating System (GOS), while a container comprises only application-related (typically light-weight) libraries and shares the Host Operating System (HOS) provided by the host physical server with the other containers.

Shortly, the main advantages of the CNT-based virtualization are:

i. containers are light-weight and can be deployed significantly quicker than VMs, since they do not require the virtualization of a whole GOS;

ii. the physical resources required by a container can be scaled up/down in real-time by the corresponding Execution Engine while, in general, physical resources are statically assigned to a VM during its bootstrapping.

Hence, due to the expected large number of devices which are virtualized in emerging IoT-based Fog-supported application environments, the number of virtual clones per physical server (i.e. the so-called virtualization density) would be allowed to increase by making use of the CNT-based virtualization.

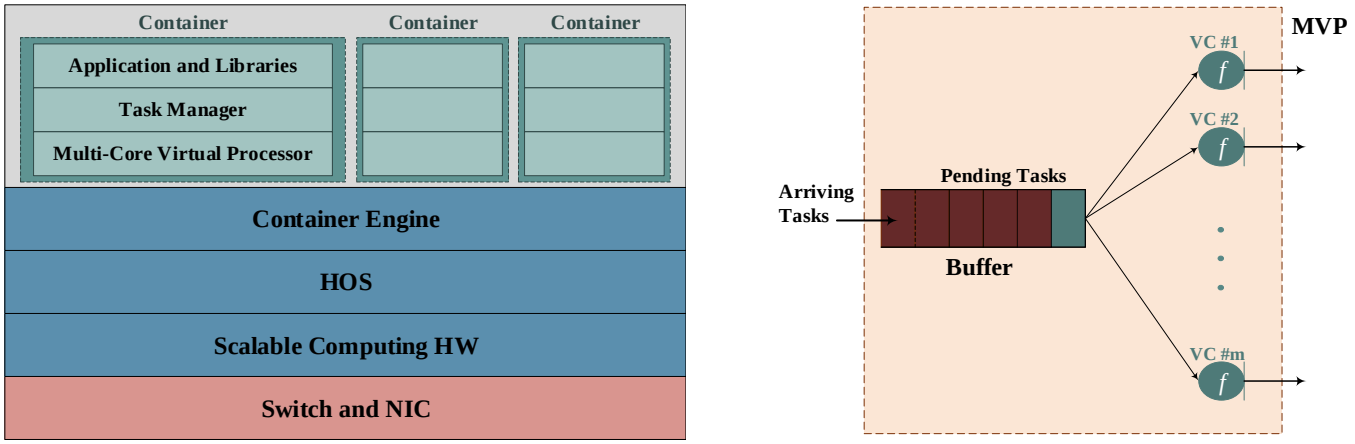

FIGURE 2. Container-based virtualization of a physical server equipping the Mobile, Fog, and Cloud nodes. (a) Virtualized server architecture; (b) Architecture of a multi-core virtual processor. HW:= CPU HardWare; NIC:= Network Interface Card; HOS:= Host Operating System; MVP:= Multi-core Virtual Processor; VC:= Virtual Core; n:= Number of virtual cores; f:= Per-core processing frequency. The application libraries of the application DAGs to be run are assumed to be pre-stored by the mobile clones deployed at the Cloud and Fog nodes.

Motivated by this consideration, Fig. 2a reports the main functional blocks of the virtualized architecture of the physical servers of the Mobile, Fog, and Cloud nodes. In this regard, two main explicative remarks are in order.

First, each server at the Mobile, Fog, and Cloud nodes hosts $nc \geq 1$ containers in number. All the containers, which are hosted by the same physical server, share: (i) the server's Host Operating System (HOS); and (ii) the pool of computing (e.g. CPU cycles) and networking (e.g. I/O bandwidth) physical resources made available by the CPU and NICs that equip the host server. The task of the Container Engine of Fig. 2a is to dynamically allocate to the requiring containers the bandwidth and computing resources made available by the host server. For this purpose, a number of Sequential or Weighted Processor Sharing (WPS) service disciplines are typically implemented by the Container Engine.

Second, each container at the Fog and Cloud nodes plays the role of a virtual clone for the associated Mobile device. Hence, the containers at the Fog and Cloud nodes act as a virtual processors and execute the tasks offloaded by the Mobile device on behalf of it. For this purpose, each container is equipped with a *Multi-core Virtual Processor* (MVP). This last comprises (see Fig. 2b): (i) a buffer, that stores the currently offloaded application tasks; and (ii) $n \geq 1$ (typically homogeneous) Virtual Cores (VCs) in number which run at the processing frequency f dictated by the Container Engine. Thus, the goal of the Task Manager of Fig. 2a is to allocate the pending application tasks over the set of virtual cores of Fig. 2b in a balanced and dynamic way. This is conducted according to the actually adopted Sequential or WPS task service discipline. The DAGs of the applications to be run by the clones are assumed to be pre-stored by the husting Fog and Cloud nodes.

1.2 THE RESULTING CLIENT-SERVER NETWORKING ARCHITECTURE

From the outset, it follows that, in Fig. 1, the Mobile device acts as a Client of the associated container-based Fog and Cloud clones, which play the role of (virtual) Servers. The main building blocks of the protocol stack of the resulting Client-Server networking architecture are reported in Fig. 3. In this figure, the red block is specific of the real-world implementation of the joint task and resource allocation framework simulated by *VirtFogSim*, while blue continuous (respectively dotted) arrow paths indicate data (respectively control) TCP/IP flows.

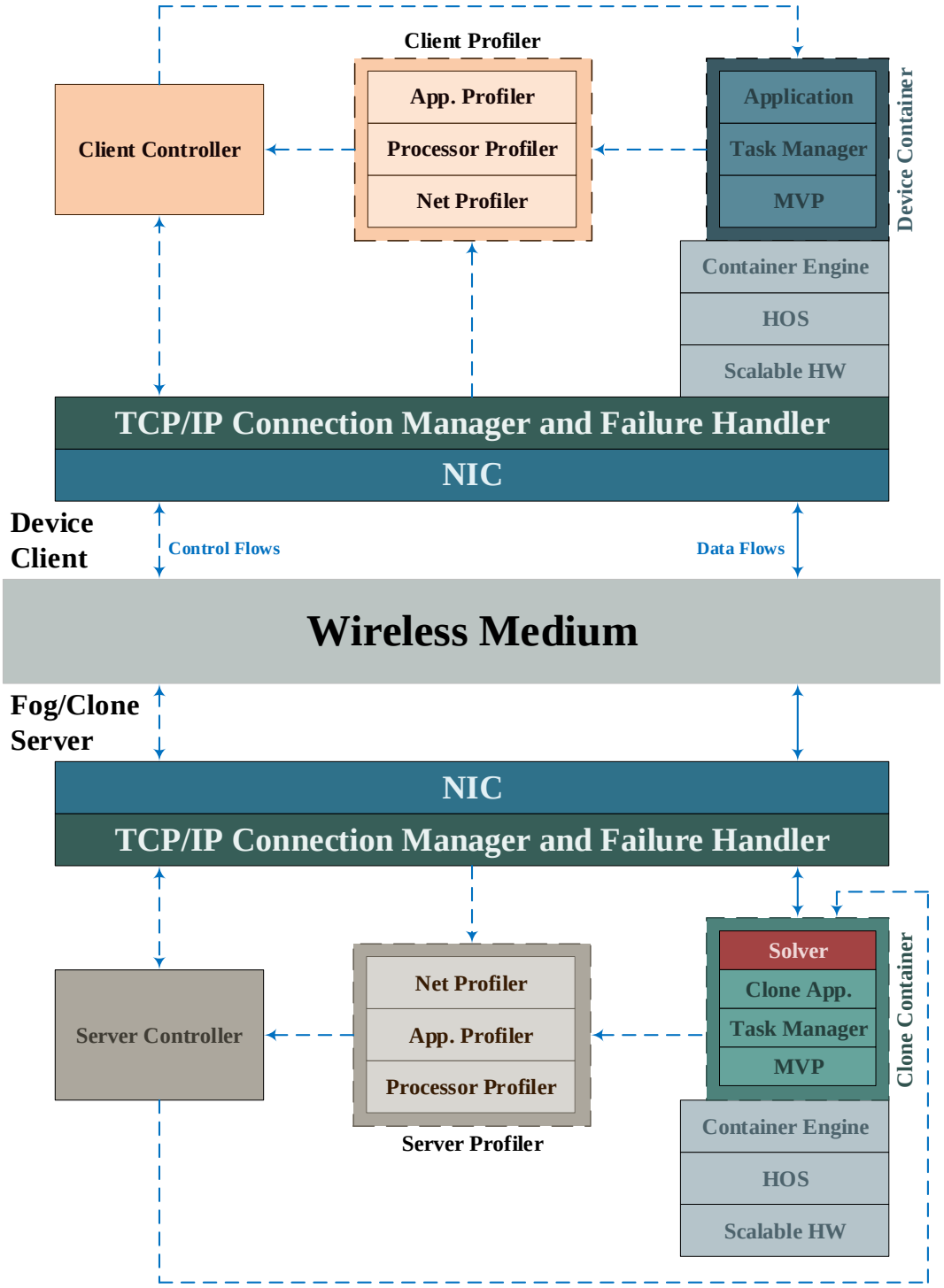

FIGURE 3. Main components of the Client-Server virtualized networked computing architecture needed for the real-world implementation of the joint task and resource allocation framework simulated by *VirtFogSim*.

2 VIRTFOGSIM – THE UNDERLYING OPTIMIZATION PROBLEM

In this section, the basic definitions and formal assumptions of the constrained optimization problem are shortly explained. The reference framework is that already reported in Figs. 1, 2 and 3.

2.1 APPLICATION DAG

The application to be run (and possibly offloaded) by the Mobile device is formally described by a Direct Acyclic Graph (DAG),

$$\mathcal{G} \stackrel{\text{def}}{=} (\mathcal{V}, \mathcal{D}), \tag{1}$$

typically referred to as Application DAG, Component Dependency Graph or Task-Call Graph. Specifically, $\mathcal{V}$ and $\mathcal{D}$ in Eq. (1) are defined as follows:

i. $\mathcal{V} \stackrel{\text{def}}{=} \{s(i) : i = 1, 2, \ldots, V\}$ is the set of tasks to be run. Each task represents an application thread or method. $V \geq 1$ is the number of tasks, and $s(i)$ (bit) is the size (i.e. the workload) of the i-th task;

ii. $\mathcal{D} \stackrel{\text{def}}{=} \{d(i,j) : i, j \in \mathcal{V}\}$ is the set of the direct edges of the DAG. They represent inter-task dependency and $d(i,j)$ (bit) is the size of the traffic that the j-th task receives in input from the i-th one.

Application DAGs must meet the following formal properties:

1. the in-degree value of each node $i \in \mathcal{V}, i \neq 1$ is at least unit. The input node $i = 1$ has vanishing in-degree value;
2. the out-degree value of each node $i \in \mathcal{V}, i \neq V$ is at least unit. The output node $i = V$ has vanishing out-degree value;
3. both the first ($i = 1$) and the last ($i = V$) nodes cannot be offloaded and must be executed at the Mobile device;
4. each inner node $i \in \mathcal{V}$, $i \neq 1$, has at least one direct path from the first task, so that it depends on the input task of the application;
5. each inner node $i \in \mathcal{V}$, $i \neq V$, has at least one path to the last V-th task, so that the application execution may move from any inner task to the final one;
6. the DAG is loop-free, so that the length of each input-to-output direct path is finite;
7. at the beginning of the application execution, the DAG and its application libraries are already stored by the Mobile device and its Cloud and Fog clones.

2.2 DAG DESCRIPTION

From a SW point of view, *VirtFogSim* assumes that the application DAG is formally described by three elements, namely:

i. the $(1 \times V)$ row vector $\vec{s} \stackrel{\text{def}}{=} [s(1), s(2), \ldots, s(V)]$ (bit) of the task sizes, i.e. $s(i)$, $i = 1, 2, \ldots, V$, is the size (measured in bit) of the i-th task of the application DAG;

ii. the $(V \times V)$ $\{0,1\}$-binary adjacency matrix A; Specifically, the (i,j)-th element $a(i,j)$, $i, j = 1, \ldots, V$ of A is unit (respectively zero) if the DAG presents (respectively does not present) a direct edge from the i-th node to the j-th one;

iii. the $(V \times V)$ nonnegative matrix D_a of the edge labels of the DAG. Specifically, the (i,j)-th element $d(i,j)$, $i, j = 1, \ldots, V$ of D_a is the traffic (measured in (bit)) that the i-th node gives in input to the j-th one. If there is not a direct edge from the i-th node to the j-th one then the corresponding $d(i,j)$ vanishes.

As an illustrative example, consider the five-state application DAG represented in Fig. 4. In this case, $V = 5$ and

$$\vec{s} = c \times [10.00, 2.13, 10.89, 30.13, 15.00]\ (bit),$$
$$A = [0,1,0,0,0\ ;\ 0,0,1,1,0\ ;\ 0,0,0,0,1\ ;\ 0,0,0,0,1\ ;\ 0,0,0,0,0],$$

and

$$D_a = c \times [0, 0.70, 0, 0, 0\ ;\ 0, 0, 17.89, 52.98, 0\ ;\ 0, 0, 0, 0, 18.54\ ;\ 0, 0, 0, 0, 42.50\ ;\ 0, 0, 0, 0, 0]\ (bit),$$

where $c \geq 1$ is a (dimensionless) scaling factor.

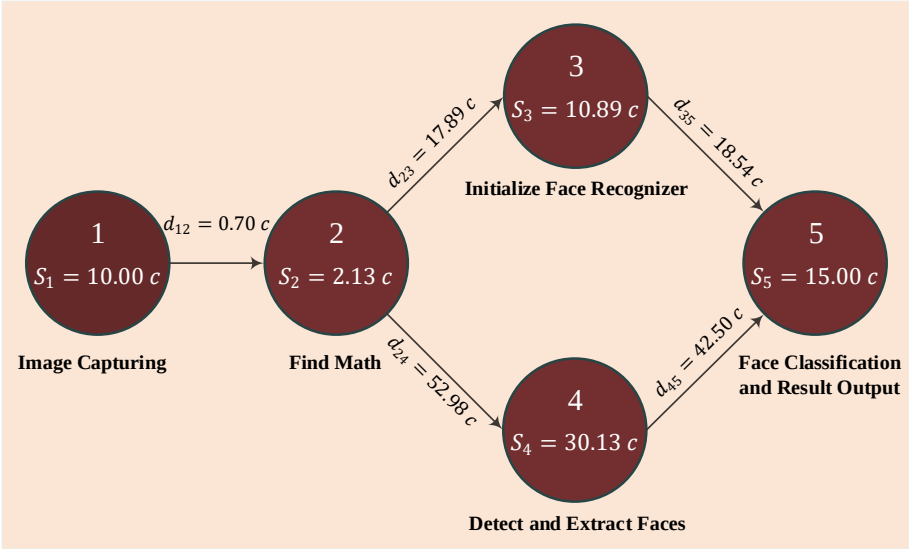

FIGURE 4. A five-state DAG for face detection and classification applications.

2.3 TASK ALLOCATION AND RESOURCE ALLOCATION VECTORS

By definition, a task allocation vector:

$$\vec{x} \stackrel{\text{def}}{=} [x(1) \equiv 1, x(2), \ldots, x(V-1), x(V) \equiv 1], \tag{2}$$

is a dimensionless $(1 \times V)$ ternary vector, whose components are defined as follows:

$$x(i) \stackrel{\text{def}}{=} \begin{cases} 0, & \text{if the } i\text{-th task is executed at the Fog clone,} \\ 1, & \text{if the } i\text{-th task is executed at the Mobile Device,} \\ 2, & \text{if the } i\text{-th task is executed at the Cloud clone,} \end{cases} \qquad i = 1, \ldots, V.$$

By design, the first and last tasks are always executed at the Mobile device and therefore $x(1) \equiv x(V) \equiv 1$ (see Eq. (2)). Due to this fact, the size of the set of the admissible task allocation vectors is 3^{V-2}.

By definition, a resource allocation vector

$$\overrightarrow{RS} \stackrel{\text{def}}{=} [f_M, f_F, f_C, R_U, R_D, B_U, B_D]\ (bit/s), \tag{3}$$

is a (1×7) nonnegative row vector whose components are measured in (bit/s). It *orderly* reports the values assumed by (see Fig. 1):

i. the processing frequency at the Mobile device;

ii. the processing frequency at the Fog clone;

iii. the processing frequency at the Cloud clone;

iv. the throughput of the Mobile-to-Fog TCP/IP connection;

v. the throughput of the Fog-to-Mobile TCP/IP connection;

vi. the throughput of the Mobile-to-Cloud TCP/IP connection;

vii. the throughput of the Cloud-to-Mobile TCP/IP connection.

2.4 THE CONSIDERED QoS JOINT TASK AND RESOURCE ALLOCATION PROBLEM

In this section, we give a glimpse to the constrained optimization problem whose solution is numerically evaluated by *VirtFogSim*. For this purpose, let $\mathcal{E}_M$, $\mathcal{E}_F$, and $\mathcal{E}_C$ $(Joule)$ be the computing energies consumed by the Mobile device, Fog clone, and Cloud clone of Fig. 1 in order to process the assigned tasks of the considered application DAG. Furthermore, let $\mathcal{E}_{WiFi}$, $\mathcal{E}_{CELL}$, and $\mathcal{E}_{WD}$ $(Joule)$ be the energies consumed by the Mobile-Fog, Mobile-Cloud and Cloud-Fog two-way TCP/IP connections, in order to transport the inter-task data among the Mobile, Cloud and Fog computing nodes. Finally, let

$$\mathcal{E}_{TOT} \stackrel{\text{def}}{=} \vartheta_M \times \mathcal{E}_M + \vartheta_F \times \mathcal{E}_F + \vartheta_C \times \mathcal{E}_C + \mathcal{E}_{WiFi} + \mathcal{E}_{CELL} + \mathcal{E}_{WD} \ \ (Joule)\,, \tag{4}$$

be the resulting computing-plus-communication total consumed energy, with ϑ_M, ϑ_F, and ϑ_C binary $\{0,1\}$-valued constants.

Hence, the considered Joint Optimization Problem *(JOP)* is defined as follows:

$$\min_{\overrightarrow{RS},\vec{x}} \mathcal{E}_{TOT}\,, \tag{5a}$$

s.t.:

$$(1/T_{DAG}) \geq TH\_MIN\_0\,, \tag{5b}$$

$$0 \leq f_M \leq f_M^{MAX}\,, \tag{5c}$$

$$0 \leq f_F \leq f_F^{MAX}\,, \tag{5d}$$

$$0 \leq f_C \leq f_C^{MAX}\,, \tag{5e}$$

$$0 \leq R_U \leq R_U^{MAX}\,, \tag{5f}$$

$$0 \leq R_D \leq R_D^{MAX}\,, \tag{5g}$$

$$0 \leq B_U \leq B_U^{MAX}\,, \tag{5h}$$

$$0 \leq B_D \leq B_D^{MAX}\,, \tag{5i}$$

$$x\,(1) = x\,(V) = 1\,, \tag{5j}$$

$$x\,(i) \in \{0,1,2\}\,,\ i = 2,\ldots,(V-1)\ . \tag{5k}$$

In the above equations, we have that:

i. $\overrightarrow{RS}$ is the resource allocation vector defined in Eq. (3);

ii. $\vec{x}$ is the task allocation vector defined in Eq. (2);

iii. $T_{DAG}\,(sec)$ is the overall time needed to execute the assigned application DAG, shortly, the DAG execution time. It generally depends on the optimization variables $\overrightarrow{RS}$ and $\vec{x}$;

iv. $TH\_MIN\_0\,(app/sec)$ is the required minimum application throughput, i.e., the minimum number of application DAGs to be performed into the time interval of a one second. It is a nonnegative real number;

v. equations (5c)-(5i) account for the maximum available resources; and,

vi. equations (5j)-(5k) account for the ternary nature of the considered task allocation vectors.

Before proceeding, four main explicative remarks about the formulated *JOP* are in order.

First, it aims to *jointly* optimize the allocation of the tasks over the Mobile, Fog and Cloud computing nodes of Fig. 1 *and* the corresponding computing-plus-communication resources. Formally speaking, it is a mixed-integer non-convex optimization problem, that resists closed-form solution. In the sequel, we will denote as: $\overrightarrow{RS}^*$ and $\vec{x}^*$ the solution of the constrained *JOP* in (5).

Second, for positive TH_MIN_0, the constraint in Eq. (5b) enforces a *hard* (e.g., deterministic) *QoS* constraint on the minimum desired application throughput. As a consequence, too much larger values of TH_MIN_0 may give rise to infeasible *JOP*.

Third, the actual values of the binary constants: ϑ_M, ϑ_F and ϑ_C depend on the application service model adopted by the Service Provider. Specifically, ϑ_M and/or ϑ_F and/or ϑ_C are unit valued (resp., vanish) when the energies consumed by the Mobile, Fog and Cloud computing nodes are for free (resp., are taxed).

Forth, the summation of the first three terms in the objective function in (4) represents the consumed computing energy, i.e.

$$\mathcal{E}_{CMP} \stackrel{\text{def}}{=} \vartheta_M \times \mathcal{E}_M + \vartheta_F \times \mathcal{E}_F + \vartheta_C \times \mathcal{E}_C \, (Joule) \, , \tag{6}$$

while the summation of the remaining three terms is the corresponding wasted network energy $\mathcal{E}_{NET}$, formally defined as follows:

$$\mathcal{E}_{NET} \stackrel{\text{def}}{=} \mathcal{E}_{WiFi} + \mathcal{E}_{CELL} + \mathcal{E}_{WD} \, (Joule) \, . \tag{7}$$

2.5 SIMULATED PROFILES OF THE COMPUTING AND NETWORK ENERGIES

In this section, we (shortly) present the analytical expressions of the computing and network energies involved by the objective function in (5a), together with the corresponding analytical expression of the (constrained) DAG T_{DAG} in (5b). The goal is to allow the *VirtFogSim* user to acquire a basic insight about the played roles and possible impacts of the input parameters in Table A of the simulator on the resulting solution: $\overrightarrow{RS}$, $\vec{x}$ of the underlying *JOP* in (5).

In this regard, four introductory remarks are in order.

First, since the Mobile device of Fig. 1 may both upload to and download from the connected Fog an Cloud clones, the previously defined network energies: $\mathcal{E}_{Wifi}$ and $\mathcal{E}_{CELL}$ split in the sum of the corresponding up and down network energies plus the underlying idle energies, so that we can write:

$$\mathcal{E}_{WiFi} = \mathcal{E}_{WiFi-IDLE} + \mathcal{E}^{U}_{WiFi} + \mathcal{E}^{D}_{WiFi} \, (Joule) \, , \tag{8}$$

and

$$\mathcal{E}_{CELL} = \mathcal{E}_{CELL-IDLE} + \mathcal{E}^{U}_{CELL} + \mathcal{E}^{D}_{CELL} \, (Joule) \, . \tag{9}$$

Second, in the considered framework of Fig. 1, the Fog-Cloud (typically wired) backhaul is assumed sustained by a two-way TCP/IP transport connection that operates in the steady-state. Hence, according to the seminal analysis reported for example in [1], the corresponding transport throughput R_0 may be directly evaluated as in:

$$R_0 = \frac{1.22 \times MSS}{RTT_{WD} \times \sqrt{Prob_{LOSS}}} \ (bit/sec) \, , \tag{10}$$

where (see Fig. 1): (i) $MSS \, (bit)$ is the maximum size of a TCP segment; (ii) $RTT_{WD} \, (sec)$ is the steady-state Round-Trip-Time of the two-way Fog-Cloud TCP/IP connection; and, (iii) $Prob_{LOSS}$ is the associated steady-state segment loss probability.

Third, according to a number of both analytical and experimental models (see, for example, [2], [3] and references therein), each the computing and network energy present in the objective function in (5) is the summation of a static energy and a dynamic energy. Specifically, we note that:

i. the static energy accounts for the energy wasted by the device in the idle state (e.g., the device is turned ON but it is not running). As a consequence, since the power consumed by the device in the idle state: $\mathcal{P}_{IDLE} \, (Watt)$ does not depend on the optimization variables, the static energy depends on the optimization variables $\overrightarrow{RS}$ and $\vec{x}$ only through the corresponding DAG execution time: T_{DAG}; and,

ii. the dynamic energy accounts for the energy wasted by the device when it is in the running state. As a consequence, since the resulting dynamic power depends on the operating computing frequency or communication bit rate, the dynamic energy of each device depends on the on the optimization variables $\overrightarrow{RS}$ and $\vec{x}$ through both the DAG execution time: T_{DAG} and the corresponding dynamic power.

Forth, a number of both analytical and experimental studies even recently appeared in the open literature [3], [4], [5], [6], [7], [8], support the conclusion that the dynamic power: $\mathcal{P}_{DYN} \, (y) \, (Watt)$ consumed by a computing (resp., network) device operating at the computing frequency (resp., bit-rate) y may be accurately profiled by a power-like relationship, e.g., $\mathcal{P}_{DYN} = K \, (y)^{\gamma} \, (Watt)$, where both the coefficient K and exponent γ depend on the specifically considered device and operating conditions[1]. Hence, by definition, the corresponding dynamic energy $\mathcal{P}_{DYN} \, (y) \, (Joule)$ equates:

$$\mathcal{E}_{DYN} \, (y) \stackrel{\text{def}}{=} \mathcal{P}_{DYN} \, (y) \, / y = K \, (y)^{\gamma - 1} \, (Joule) \, ,$$

Therefore, after indicating by:

[1] According to the standard MATLAB style, the programming-oriented notation adopted in Table A is of underscored-type, while the scientific-oriented one used in the above equations is of subscript-type. This means, for example, that: $\mathcal{P}_{IDLE-CPU-M}$ and γ_M in Eq. (11) are translated into: $\mathcal{P}\_IDLE\_CPU\_M$ and $gamma\_M$ in Table A.

i. $\delta(x)$ the Kronecker's delta (e.g., $\delta(x)=1$ for $x=1$ and $\delta(x)=0$, otherwise); and by:

ii. $u_{-1}(x)$ the unit-step function (e.g., $u_{-1}(x)=1$ for $x>0$ and $u_{-1}(x)=0$, otherwise),

the above considerations lead to the following sum-power-like analytical models for the profiles of the (previously introduced) computing energies (measured in $(Joule)$):

$$\begin{aligned}\mathcal{E}_M = &\left(\frac{\mathcal{P}_{IDLE-CPU-M}}{nc_M}\right)\times T_{DAG}\times u_{-1}\left(\sum_{i=1}^{|V|}\delta(x(i)-1)\right)\\ &+\left(\sum_{i=1}^{|V|}s(i)\,\delta(x(i)-1)\right)K_M(1-r_M)(f_M)^{\gamma M-1},\end{aligned}\tag{11a}$$

$$\begin{aligned}\mathcal{E}_F = &\left(\frac{\mathcal{P}_{IDLE-CPU-C}}{nc_C}\right)\times T_{DAG}\times u_{-1}\left(\sum_{i=1}^{|V|}\delta(x(i)-2)\right)\\ &+\left(\sum_{i=1}^{|V|}s(i)\,\delta(x(i)-2)\right)K_C(1-r_C)(f_C)^{\gamma C-1},\end{aligned}\tag{11b}$$

$$\begin{aligned}\mathcal{E}_C = &\left(\frac{\mathcal{P}_{IDLE-CPU-C}}{nc_C}\right)\times T_{DAG}\times u_{-1}\left(\sum_{i=1}^{|V|}\delta(x(i)-2)\right)\\ &+\left(\sum_{i=1}^{|V|}s(i)\,\delta(x(i)-2)\right)K_C(1-r_C)(f_C)^{\gamma C-1},\end{aligned}\tag{11c}$$

where all the device-depending parameters: $\mathcal{P}_{IDLE-CPU}$, nc, K, r, and γ involved in the above energy-profiling relationships are detailed in the Table A, together with their meaning/role and proper measuring units[1].

In order to introduce the energy profiles of the five up/down WiFi, up/down Cellular and two-way backhaul connections referred by Eqs. (8), (9), and (10), let: $\mathcal{V}_{M\to F}$, $\mathcal{V}_{F\to M}$, $\mathcal{V}_{M\to C}$, $\mathcal{V}_{C\to M}$, and $\mathcal{V}_{F\leftrightarrow C}$ indicate the volumes of data (all measured in bit) transported (see Fig. 1): (i) from the Mobile to Fog; (ii) from the Fog to Mobile; (iii) from the Mobile to Cloud; (iv) from the Cloud to Mobile; and, (v) exchanged between Fog and Cloud, respectively. Hence, directly from the definitions of the adjacency A and D_a connection matrices of Section 2.2, as well as the task allocation vector $\vec{x}$ in Eq. (2), these volumes may be formally expressed as:

$$\mathcal{V}_{M\to F}=\left(1+\overline{NF}_{WiFi}\right)\left(\sum_{i,j=1}^{|V|}a_{ij}\delta(x(i)-1)\,\delta(x(j))d_{ij}\right)(bit),\tag{12a}$$

$$\mathcal{V}_{F\to M}=\left(1+\overline{NF}_{WiFi}\right)\left(\sum_{i,j=1}^{|V|}a_{ij}\delta(x(i))\,\delta(x(j)-1)\,d_{ij}\right)(bit),\tag{12b}$$

$$\mathcal{V}_{M\to C}=\left(1+\overline{NF}_{CELL}\right)\left(\sum_{i,j=1}^{|V|}a_{ij}\delta(x(i)-1)\,\delta(x(j)-2)\,d_{ij}\right)(bit),\tag{12c}$$

$$\mathcal{V}_{C\to M}=\left(1+\overline{NF}_{CELL}\right)\left(\sum_{i,j=1}^{|V|}a_{ij}\delta(x(i)-2)\,\delta(x(j)-1)\,d_{ij}\right)(bit),\tag{12d}$$

$$\mathcal{V}_{F\leftrightarrow C}=\left(1+\overline{NF}_{WD}\right)\left(\sum_{i,j=1}^{|V|}a_{ij}\left(\delta(x(i)-2)\,\delta(x(j))+\delta(x(i))\,\delta(x(j)-2)\right)d_{ij}\right)(bit),\tag{12e}$$

where $\overline{NF}$ is the average number of failures of the considered referred connection [7], [8] (see also Table A). Hence, the idle, up and down energies of the WiFi and Cellular network connections in Eqs. (8) and (9) may be profiled as follows:

$$\mathcal{E}_{WiFi-IDLE}=(\vartheta_M+\vartheta_F)\times\mathcal{P}_{IDLE-WiFi-M}\times T_{DAG}\times u_{-1}\left(\sum_{i=1}^{|V|}\delta(x(i))\right),\tag{13a}$$

$$\mathcal{E}^U_{WiFi}=\mathcal{V}_{M\to F}\left(\vartheta_M\Omega_{WiFi-TX}(R_U)^{\xi_{WiFi-TX}-1}+\vartheta_F\Omega_{WiFi-RX}(R_U)^{\xi_{WiFi-RX}-1}\right),\tag{13b}$$

$$\mathcal{E}^D_{WiFi}=\mathcal{V}_{F\to M}\left(\vartheta_M\Omega_{WiFi-RX}(R_D)^{\xi_{WiFi-RX}-1}+\vartheta_F\Omega_{WiFi-TX}(R_D)^{\xi_{WiFi-TX}-1}\right),\tag{13c}$$

and

$$\mathcal{E}_{CELL-IDLE}=(\vartheta_M+\vartheta_C)\times\mathcal{P}_{IDLE-CELL-M}\times T_{DAG}\times u_{-1}\left(\sum_{i=1}^{|V|}\delta(x(i)-2)\right),\tag{14a}$$

$$\mathcal{E}^U_{CELL}=\mathcal{V}_{M\to C}\left(\vartheta_M\Omega_{CELL-TX}(B_U)^{\xi_{CELL-TX}-1}+\vartheta_C\Omega_{CELL-RX}(B_U)^{\xi_{CELL-RX}-1}\right),\tag{14b}$$

$$\mathcal{E}^D_{CELL}=\mathcal{V}_{C\to M}\left(\vartheta_M\Omega_{CELL-RX}(B_D)^{\xi_{CELL-RX}-1}+\vartheta_C\Omega_{CELL-TX}(B_D)^{\xi_{CELL-TX}-1}\right).\tag{14c}$$

Finally, the energy consumed by the backhaul connection reads as in:

$$\begin{aligned}\mathcal{E}_{WD} =& \left(\vartheta_C \mathcal{P}_{IDLE-ETH-C} + \vartheta_F \mathcal{P}_{IDLE-ETH-F}\right) \times T_{DAG} \times u_{-1} \\ &\times \left(\sum\nolimits_{i,j=1}^{|V|} a_{ij} \left(\delta\left(x\left(i\right)\right)\delta\left(x\left(j\right)-2\right) + \delta\left(x\left(i\right)-2\right)\delta\left(x\left(j\right)\right)\right)\right) + \mathcal{V}_{F\leftrightarrow C}\left(\frac{\mathcal{P}_{WD}}{R_0}\right),\end{aligned} \tag{15a}$$

where the corresponding wasted dynamic power $\mathcal{P}_{WD}\left(Watt\right)$ is given by the product:

$$\mathcal{P}_{WD} = no_{HOP} \times \mathcal{P}_{HOP} \tag{15b}$$

of the number of hops: no_{HOP} by the per-hop consumed power: $\mathcal{P}_{HOP}\left(Watt\right)$ (see Table A).

2.6 SIMULATED PROFILES OF THE PER-DAG AND PER-TASK EXECUTION TIMES

The analytical expression assumed by T_{DAG} in Eq. (5b) depends on the vectors $\overrightarrow{RS}$, $\vec{x}$, as well as the adopted inter-node Task Scheduling and intra-node Task Service disciplines. The current implementation of *VirtFogSim* assumes that the adopted Task Scheduling discipline over the computing nodes and the adopted Task service discipline at each computing node are both of *sequential* type. They are, indeed, the Task Scheduling and Task Service disciplines currently adopted by a number of Middleware software tools for the supporting of mobile distributed computing, such as, MAUI, Volare, Cuckoo and CloneCloud (see, for example, [9], [10] and references therein for an overview of state-of-the-art middleware software solutions for the support of computing-intensive smartphone-based applications).

As direct consequence, the following sum expression holds for the the analytical profile of T_{DAG}:

$$T_{DAG} = \sum\nolimits_{i=1}^{|V|} \left(\delta\left(x\left(i\right)-1\right)T_M\left(i\right) + \delta\left(x\left(i\right)\right)T_F\left(i\right) + \delta\left(x\left(i\right)-2\right)T_C\left(i\right)\right), \tag{16}$$

where $T_M\left(i\right)$, $T_F\left(i\right)$, and $T_C\left(i\right)$ are the total execution times (measured in *seconds*) of the i-th task of the considered DAG when it is executed at the Mobile or Fog or Cloud nodes, respectively.

In this regard, two main remarks are in order. First, each total execution time $T_N\left(i\right),\ N \in \{M,F,C\}$, in Eq. (16) is by design, the summation of two terms. The first term is the computing time (e.g., the service time): $T_N^{SER}\left(i\right),\ N \in \{M,F,C\}$, wasted for the processing of the i-th task at the computing node N, while the second term is the communication delay: $T_N^{NET}\left(i\right),\ N \in \{M,F,C\}$, induced by the transport from the other two computing nodes of the input data needed for the processing of the i-th task at the computing node N. Second, when any set of tasks is processed by the same computing node, the resulting overall communication delay is, by design, vanishing [8], [11], [12].

Therefore, according to these two remarks, the following analytical expressions hold for the profiles of the nodal execution times in Eq. (16):

$$\begin{aligned}T_M\left(i\right) =& T_M^{SER}\left(i\right) + T_M^{NET}\left(i\right) = \left(\frac{\sum_{j=1}^{|V|} s\left(j\right)\delta\left(x\left(j\right)-1\right)}{n_M f_M}\right) \\ &+ \left[\left(1+\overline{NF}_{WiFi}\right)\left(\frac{\sum_{j=1}^{|V|} a_{ji}d_{ji}\delta\left(x\left(j\right)\right)}{R_D}\right) + \left(1+\overline{NF}_{CELL}\right)\left(\frac{\sum_{j=1}^{|V|} a_{ji}d_{ji}\delta\left(x\left(j\right)-2\right)}{B_D}\right)\right],\end{aligned} \tag{17a}$$

$$\begin{aligned}T_F\left(i\right) =& T_F^{SER}\left(i\right) + T_F^{NET}\left(i\right) = \left(\frac{\sum_{j=1}^{|V|} s\left(j\right)\delta\left(x\left(j\right)\right)}{n_F f_F}\right) \\ &+ \left[\left(1+\overline{NF}_{WiFi}\right)\left(\frac{\sum_{j=1}^{|V|} a_{ji}d_{ji}\delta\left(x\left(j\right)-1\right)}{R_U}\right) + \left(\frac{\sum_{j=1}^{|V|} a_{ji}d_{ji}\delta\left(x\left(j\right)-2\right)}{R_0}\right)\right],\end{aligned} \tag{17b}$$

$$\begin{aligned}T_C\left(i\right) =& T_C^{SER}\left(i\right) + T_C^{NET}\left(i\right) = \left(\frac{\sum_{j=1}^{|V|} s\left(j\right)\delta\left(x\left(j\right)-2\right)}{n_C f_C}\right) \\ &+ \left[\left(1+\overline{NF}_{CELL}\right)\left(\frac{\sum_{j=1}^{|V|} a_{ji}d_{ji}\delta\left(x\left(j\right)-1\right)}{B_U}\right) + \left(\frac{\sum_{j=1}^{|V|} a_{ji}d_{ji}\delta\left(x\left(j\right)\right)}{R_0}\right)\right],\end{aligned} \tag{17c}$$

where $i = 1,\ldots,|V|$, and n_M,n_F and n_C are the number of cores equipping the Mobile, Fog and cloud nodes, respectively (see Table A).

2.7 SIMULATED ADAPTIVE RESOURCE ALLOCATION

In this subsection, we shortly present the general analytical framework implemented by the *RAP_p* and *FogTracker* functions of Sections 3.2 and 3.12, in order to perform the adaptive updating of the computing frequencies: f_M, f_F, f_C, the wireless network bandwidths: R_U, R_D, B_U, B_D, and the Lagrange multiplier λ associated to the throughput constraint in Eq. (5b). The final goal is to allow the user to acquire insights about the role played by the input step-size: a_{MAX} and related input step-size vector $\vec{a}_{MAX-FT}$ on the adaptive capability of the implemented resource allocation framework.

For this purpose, after indicating by $\mathcal{L}$ the Lagrangian function of the stated *JOP* of Eq. (5), let:
$y \in \{f_M, f_F, f_C, R_U, R_D, B_U, B_D, \lambda\}$ indicates a (scalar and non-negative) variable to be optimized and let y_{MAX} be its allowed maximum value. Furthermore, after denoting by t an integer-valued iteration index, let: $\nabla_y \mathcal{L}(t)$ be the derivative of the Lagrangian function $\mathcal{L}$ with respect to the considered optimization variable y at iteration t. Hence, according to the so-called primal-dual iteration-based approach recently customized in [13] for broadband networked application scenarios, the adaptive updating of $y(t+1)$ reads as follows:

$$y(t+1) = \max\{0, \min\{y_{MAX}, y(t) - g_y(t) \nabla_y \mathcal{L}(t)\}\}, \tag{18}$$

where the outer $\max\{.\}$ (resp., the inner $\min\{.\}$) accounts for the non-negative (resp., maximum) value of y.

The peculiar feature of the updating relationship in (18) is that it resort to a y-depending) and time-varying gain function: $g_y(t)$, in order to guarantee the *quick adaptation* of the optimization variable y in response to abrupt changes of the wireless environment of Fig. 1. On the basis of the general result of [13] about the convergence to the steady-state of the iteration in (18), we planned to implement in the current version 4.0 of *VirtFogSim* the following time-varying gain formula:

$$g_y(t+1) = \max\{a_{MAX}, \min\{a_{MAX} \times y_{MAX}, \frac{1}{2}(y(t))^2\}\}, \tag{19}$$

An examination of the above formula unveils the role played by the (positive) step-size a_{MAX}, as well as its potential impact on the adaptive capability of the overall simulator. Specifically, on the basis of (19), we expect that large (resp., small) values of a_{MAX} lead to quick (resp., slow) response to abrupt environmental changes, together with large (resp., small) possible oscillations in the steady-state. At this regard, we anticipate that the final goal of the *FogTracker* function of Section 3.12 is to guide the user towards a right trade-off among these two a_{MAX}-depending contrasting behaviors.

3 *VIRTFOGSIM* – THE BUILT-UP TASK ALLOCATION STRATEGIES AND THEIR PARALLEL EXECUTION

The engine of the *VirtFogSim* simulator is built up by a main core of eight software routines that implement a number of strategies (e.g., optimization policies), in order to check the feasibility and, then, numerically solve the constrained optimization problem in (5). MATLAB is the native environment under which the optimization routines are developed and run.

In this regard, it must be remarked that the current version 4.0 of the *VirtFogSim* package is capable to *automatically* turn ON and exploit the multi-core capability possibility retained by the supporting hardware platform, in order to run parallel programming and, then, speed up the execution of the implemented task allocation policies. This is done in a fully *transparent* way, e.g., without any direct user involvement. For this purpose, some instructions for parallel programming on multi-core hardware platforms specifically supported by the *Parallel Tool Box* of MATLAB are utilized by the *VirtFogSim* software. However, *VirtFogSim* is capable to *automatically* switch to the *sequential* execution mode when the underlying hardware platform is single-core or the MATLAB *Parallel Tool Box* is not installed.

Passing to describe the software architecture of the simulator, *VirtFogSim* act as the main program that:

i. allows the user to setup 67 input parameters that characterize the multi-homing scenario to be simulated by the user (see 1);

ii. calls the *GeneticTA_par* function for parallel execution and returns the corresponding

   a. $x\_best \in \{0,1,2\}^V$, i.e. ternary V-tuple best allocation vector;

   b. $RS\_best \stackrel{\text{def}}{=} [f_M^*, f_F^*, f_C^*, R_U^*, R_D^*, B_U^*, B_D^*]$ (bit/s), i.e. the 7-tuple vector of the RAP-returned optimal resource allocation of the Mobile computing frequency, Fog computing frequency, Cloud computing frequency, Mobile-to-Fog transport throughput, Fog-to-Mobile transport throughput, Mobile-to-Cloud transport throughput, and Cloud-to-Mobile transport throughput (see Fig. 1);

   c. E_best $(Joule)$, i.e., the total computing-plus-network energy consumed by the infrastructure of Fig. 1 under the returned x_best task allocation vector.
   For this purpose, the *GeneticTA_par* function calls, in turn, the auxiliary functions *RAP*, *Crossover* and *Mutation*;

iii. optionally, calls the *OM_S* function and returns:

   a. $RS\_OM \stackrel{\text{def}}{=} [f_M, f_F, f_C, R_U, R_D, B_U, B_D]$ (bit/s), i.e., the 7-tuple vector of the *RAP*-computed optimal resource allocation under the following $(1 \times V)$ *All-Mobile* task allocation vector: $x\_OM \stackrel{\text{def}}{=} [1, 1, \ldots, 1, 1]$;

   b. E_OM $(Joule)$, i.e., the total computing-plus-network energy consumed by the infrastructure of Fig. 1 under x_OM. If the returned E_OM is infinite, then, the *All-Mobile* task allocation x_OM is infeasible.
   OM_S function calls, in turn, the *RAP* function;

iv. optionally, calls the *OF_S* function and returns:

   a. $RS\_OF \stackrel{\text{def}}{=} [f_M, f_F, f_C, R_U, R_D, B_U, B_D]$ (bit/s), i.e., the 7-tuple vector of the *RAP*-computed optimal resource allocation under the following $(1 \times V)$ *All-Fog* task allocation vector: $x\_OF \stackrel{\text{def}}{=} [1, 0, \ldots, 0, 1]$;

   b. E_OF $(Joule)$, i.e., the total computing-plus-network energy consumed by the infrastructure of Fig. 1 under x_OF. If the returned E_OF is infinite, then, the *All-Fog* task allocation x_OF is infeasible.
   OF_S function calls, in turn, the *RAP* function;

v. optionally, calls the *OC_S* function and returns:

   a. $RS\_OC \stackrel{\text{def}}{=} [f_M, f_F, f_C, R_U, R_D, B_U, B_D]$ (bit/s), i.e., the 7-tuple vector of the *RAP*-returned optimal resource allocation under the following $(1 \times V)$ *All-Cloud* task allocation vector: $x\_OC \stackrel{\text{def}}{=} [1, 2, \ldots, 2, 1]$;

   b. E_OC $(Joule)$, i.e., the total computing-plus-network energy consumed by the infrastructure of Fig. 1 under x_OC. If the returned E_OC is infinite, then, the *All-Cloud* task allocation x_OC is infeasible.
   OC_S function calls, in turn, the *RAP* function;

vi. optionally, calls the *O_TAS_ par* function for parallel execution and returns:

   a. $x\_OTAS \in \{0, 1, 2\}^V$, i.e., the ternary V-tuple best allocation vector computed by *O_TAS* under the assumption that the optimization of the resource allocation is *not* performed;

   b. E_OTAS $(Joule)$, i.e., the total computing-plus-network energy consumed by the infrastructure of Fig. 1 under x_OTAS. If the returned E_OTAS is infinite, then, the *O_TAS* task allocation x_OTAS is infeasible.
   O_TAS_ par function calls, in turn, the *Crossover*, *Mutation*, and *evaluatestaticenergy_ p* functions;

vii. optionally, calls the *ES_S_ par* function for parallel execution and returns:

   a. $x\_ESS \in \{0, 1, 2\}^V$, i.e., the ternary V-tuple best task allocation vector computed by performing the exhaustive search over the full population of task allocation vectors of the size 3^{V-2};

   b. $RS\_ESS \stackrel{\text{def}}{=} [f_M, f_F, f_C, R_U, R_D, B_U, B_D]$ (bit/s), i.e., the 7-tuple vector of the *RAP*-computed optimal resource allocation under x_ESS;

   c. E_ESS $(Joule)$, i.e., the total computing-plus-network energy consumed by the infrastructure of Fig. 1 under x_ESS. If the returned E_ESS is infinite, then, the overall afforded optimization problem is infeasible.
   ES_S_ par function calls, in turn, the *RAP* function and requires that the task allocation vectors x_best and x_OTAS are already available;

viii. optionally, calls the *FogTracker* function. It returns the time plots over the interval: $[1, iteration\_number]$ of the:

   a. total energy E_TOT consumed by the overall Mobile-Fog-Cloud ecosystem;

   b. the corresponding energy E_NET consumed by the up/down WiFi, Cellular and Backhaul connections of Fig. 1; and,

   c. the behavior of the *lambda* multiplier associated to the hard throughput constraint of Eq. (5b), when abrupt changes in the pattern of the allocated tasks and/or maximum bandwidths of the WiFi/Cellular connections occur. The user may set the magnitude of these changes, in order to test various (possibly, mobility induced) time-fluctuations of the simulated environment of Fig. 1 (see Section 3.13 in the sequel for a full description of the *FogTracker* function and the supported options).

The current 4.0 version of the VirtFogSim simulator is equipped with a (rich) Graphical User Interface (GUI) that displays:

i. the numerical values of the up/down WiFi-Cellular-Backhual bandwidths and Mobile-Fog-Cloud computing frequencies returned by the allocation policies *GeneticTA_par*, *O_TAS_par*, *ES_S_par*, *OM_S*, *OF_S*, and *OC_S* in the form of colored bar plots; and,

ii. the (generally, different) task allocation patterns performed by *GeneticTA_par*, *O_TAS_par*, *ES_S_par*, *OM_S*, *OF_S*, and *OC_S* in form of suitably colored labeled graphs of the underlying application DAG (see Section 4 in the sequel for a description of the main screen-shoots returned by the GUI of *VirtFogSim*).

3.1 THE INPUT PARAMETERS TO THE VIRTFOGSIM SIMULATOR

VirtFogSim exposes a GUI which allows the user to:

i. set 67 input parameters, in order to define the desired computing and communication setup of the overall infrastructure of Fig. 1;

ii. select any subset of the (aforementioned) *GeneticTA_p*, *OM_S*, *OC_S*, *OF_S*, *O_TAS_par*, *ES_S_par*, and *FogTracker* functions, in order to test and compare various task and/or resource allocation strategies under the scenario dictated by the desired input parameters.

In the sequel, the input parameters of the *VirtFogSim simulator* are listed, together with their meaning, measuring units and ranges of likelihood values.

TABLE A. Input parameters of the *VirtFogSim* simulator version 4.0. *NIC*: Network Interface Card; *RTT*: Round Trip Time.

INPUT PARAMETERS	MEANING/ROLE	MEASURING UNITS	SIMULATED SETTINGS
V	Number of tasks of the application DAG	Dimensionless	$9 \leq \mathrm{V} \leq 45$
s	V-tuple row vector of the task workloads	(bit)	Computing and communication-intensive DAG
A	$(V \times V)$ binary (i.e. $\{0,1\}$) DAG adjacency matrix	Dimensionless	
Da	$(V \times V)$ weight matrix of the (directed) edges of the DAG	(bit)	Computing and communication-intensive DAG
n_M	Number of the (virtual) computing cores equipping the Mobile device	Dimensionless	$\mathrm{n\_M} = 1$
n_F	Number of the (virtual) computing cores equipping the device clone at the Fog node	Dimensionless	$\mathrm{n\_F} = 4$
n_C	Number of (virtual) computing cores equipping the device clone at the Cloud node	Dimensionless	$\mathrm{n\_C} = 12$
NF_WiFi	Average per- connection number of failures of the one-way: Mobile $\rightarrow$ Fog, and: Fog $\rightarrow$ Mobile WiFi TCP/IP connections	Dimensionless	$\mathrm{NF\_WiFi} = 1.1$
NF_CELL	Average per-connection number of failures of the cellular one-way: Mobile $\rightarrow$ Cloud, and: Cloud $\rightarrow$ Mobile TCP/IP connections	Dimensionless	$\mathrm{NF\_CELL} = 0.1$
NF_WD	Average per-connection number of failures of the two-way: Cloud $\leftrightarrow$ Fog TCP/IP backhaul connection	Dimensionless	$\mathrm{NF\_WD} = 0.01$
f_MAX_M	Per-core maximum computing frequency at the Mobile device	(bit/s)	$\mathrm{f\_MAX\_M} = 12 \times 10^6$
f_MAX_F	Per-core maximum computing frequency at the Fog clone	(bit/s)	$\mathrm{f\_MAX\_F} = 12 \times 10^6$
f_MAX_C	Per-core maximum computing frequency at the Cloud clone	(bit/s)	$\mathrm{f\_MAX\_F} = 12 \times 10^6$
R_MAX_U	Maximum bit rate of the WiFi-based one-way: Mobile $\rightarrow$ Fog TCP/IP connection	(bit/s)	$\mathrm{R\_MAX\_U} = 8.0 \times 10^6$
R_MAX_D	Maximum bit rate of the WiFi-based one-way: Fog $\rightarrow$ Mobile TCP/IP connection	(bit/s)	$\mathrm{R\_MAX\_D} = 9.0 \times 10^6$
B_MAX_U	Maximum bit rate of the cellular one-way: Mobile $\rightarrow$ Cloud TCP/IP connection	(bit/s)	$\mathrm{B\_MAX\_U} = 6.5 \times 10^6$

TABLE A (Continued).

INPUT PARAMETERS	MEANING/ROLE	MEASURING UNITS	VALUE RANGE
B_MAX_D	Maximum bit rate of the cellular one-way: Cloud $\rightarrow$ Mobile TCP/IP connection	(bit/s)	B_MAX_D $= 7.0 \times 10^6$
TH_MIN_0	Scalar non-negative real parameter. It is the minimum required application throughput	(DAG/s)	$0.333 \leq$ TH_MIN_0 ≤ 3.33
Teta_M	Binary $\{0, 1\}$-parameter. It depends on the utilized application service model at the Mobile device	Dimensionless	Teta_M $= 0$ (resp., Teta_M $= 1$) if computing-plus-networking energy consumed by the Mobile device is (resp., is not) for free
Teta_F	Binary $\{0, 1\}$-parameter. It depends on the utilized application service model at the Fog node	Dimensionless	Teta_F $= 0$ (resp., Teta_M $= 1$) if the computing-plus-networking energy consumed by the Fog clone is (resp., is not) for free
Teta_C	Binary $\{0, 1\}$-parameter. It depends on the utilized application service model at the Cloud node	Dimensionless	Teta_C $= 0$ (resp., Teta_C $= 1$) if computing-plus-networking energy consumed by the Cloud clone is (resp., is not) for free
P_IDLE_CPU_M	Power consumed in the idle state by the physical CPU at the Mobile device	$(Watt)$	P_IDLE_CPU_M $= 1.2$
P_IDLE_CPU_F	Power consumed by a single physical server in the idle state at the Fog node	$(Watt)$	P_IDLE_CPU_F $= 220$
P_IDLE_CPU_C	Power consumed by a single physical server in the idle state at the Cloud node	$(Watt)$	P_IDLE_CPU_C $= 440$
nc_M	Number of containers simultaneously running atop the Mobile CPU	Dimensionless	nc_M $= 1$
nc_F	Number of containers simultaneously running atop a single physical server at the Fog node	Dimensionless	nc_F $= 16$
nc_C	Number of containers simultaneously running atop a single physical server at the Cloud node	Dimensionless	nc_C $= 24$
gamma_M	Positive exponent of the dynamic power consumption of the CPU at the Mobile device	Dimensionless	gamma_M $= 3.2$
gamma_F	Positive exponent of the dynamic power consumption of a single physical server at the Fog node	Dimensionless	gamma_F $= 3.1$
gamma_C	Positive exponent of the dynamic power consumption of a single physical server at the Cloud node	Dimensionless	gamma_C $= 3.0$
k_M	Positive parameter. It profiles of the dynamic power consumption of the CPU at the Mobile device	$(Watt)$ / $(bit/s)^{\text{gamma\_M}}$	k_M $= 7.5 \times 10^{-21}$
k_F	Positive parameter. It profiles of the dynamic power consumption of a single physical server at the Fog node	$(Watt)$ / $(bit/s)^{\text{gamma\_F}}$	k_F $= 9.78 \times 10^{-20}$
k_C	Positive scalar parameter. It profiles of the dynamic power consumption of a single physical server at the Cloud node	$(Watt)$ / $(bit/s)^{\text{gamma\_C}}$	k_F $= 1.14 \times 10^{-19}$
r_M	Scalar fraction of the overall computing power shared by the computing cores at the Mobile device	Dimensionless	r_M $= 0$
r_F	Scalar fraction of the overall computing power shared by the computing cores of a single physical server at the Fog node	Dimensionless	r_F $= 0.2$
r_C	Scalar fraction of the overall computing power shared by the computing cores of a single physical server at the Cloud node	Dimensionless	r_C $= 0.1$

TABLE A (Continued).

INPUT PARAMETERS	MEANING/ROLE	MEASURING UNITS	VALUE RANGE
P_IDLE_ETH	Power consumed in the idle state by each physical Ethernet NIC at the Fog and Cloud nodes	$(Watt)$	P_IDLE_ETH $= 1.0 \times 10^{-4}$
P_IDLE_WiFi_M	Power consumed in the idle state by each physical WiFi NIC at the Mobile device and Fog node	$(Watt)$	P_IDLE_WiFi_M $= 1.3$
P_IDLE_CELL_M	Power consumed in the idle state by each physical cellular NIC at the Mobile device and Cloud node	$(Watt)$	P_IDLE_CELL_M $= 0.82$
zeta_TX_WiFi	Positive scalar exponent of the dynamic power consumption of the WiFi NIC in the transmit mode	Dimensionless	zeta_TX_WiFi $= 2.40$
zeta_RX_WiFi	Positive scalar exponent of the dynamic power consumption of the WiFi NIC in the receive mode	Dimensionless	zeta_RX_WiFi $= 2.20$
zeta_TX_CELL	Positive scalar exponent of the dynamic power consumption of the cellular NIC in the transmit mode	Dimensionless	zeta_TX_CELL $= 2.45$
zeta_RX_CELL	Positive exponent of the dynamic power consumption of the cellular NIC in the receive mode	Dimensionless	zeta_RX_CELL $= 2.34$
eta	Positive exponent of the RTTs of the TCP/IP WiFi and Cellular connections	Dimensionless	eta $= 0.6$
RTT_WiFi	Average RTT of the TCP/IP WiFi up/down connections	(s)	RTT_WiFi $= 1.0 \times 10^{-2}$
RTT_CELL	Average RTT of the TCP/IP cellular up/down connections	(s)	RTT_CELL $= 1.0 \times 10^{-1}$
RTT_WD	Average RTT of the (possibly, multi-hop) two-way backhaul	(s)	RTT_WD $= 1.0$
MSS	Maximum size of a TCP segment	(bit)	Typically MSS $= 1.2 \times 10^{4}$
Prob_LOSS	Loss probability of the: Cloud $\leftrightarrow$ Fog TCP/IP connection	Dimensionless	Prob_LOSS $= 1.56 \times 10^{-5}$
chi_TX_WiFi	Power profile of the WiFi NIC in the transmit mode	$(Watt)$ / $\left((bit/s)^{\text{zeta\_TX\_WiFi}} \times (s)^{\text{eta}}\right)$	chi_TX_WiFi $= 5.0 \times 10^{-14}$
chi_RX_WiFi	Power profile of the WiFi NIC in the receive mode	$(Watt)$ / $\left((bit/s)^{\text{zeta\_RX\_WiFi}} \times (s)^{\text{eta}}\right)$	chi_RX_WiFi $= 1.4 \times 10^{-14}$
chi_TX_CELL	Power profile of the cellular NIC in the transmit mode	$(Watt)$ / $\left((bit/s)^{\text{zeta\_TX\_CELL}} \times (s)^{\text{eta}}\right)$	chi_TX_CELL $= 2.31 \times 10^{-13}$
chi_RX_CELL	Power profile of the cellular NIC in the receive mode	$(Watt)$ / $\left((bit/s)^{\text{zeta\_RX\_CELL}} \times (s)^{\text{eta}}\right)$	chi_RX_CELL $= 8.1 \times 10^{-15}$
I_MAX	Maximum number of primal-dual iterations performed by the *RAP_ p* function	Dimensionless	$500 \leq$ I_MAX ≤ 700
a_MAX	Speed-factor of the iterations performed by the *RAP_ p* function	Dimensionless	$1 \times 10^{-7} \leq$ a_MAX $\leq 8 \times 10^{-7}$
no_HOP	Number of hops of the two-way: Cloud $\leftrightarrow$ Fog backhaul connection	Dimensionless	no_HOP $= 4$
P_HOP	Per-hop average power consumed the two-way backhaul connection	$(Watt)$	P_HOP $= 5.7 \times 10^{-1}$
PS	Population size of the simulated *Genetic* functions. It must be even and not less than 4	Dimensionless	$120 \leq$ PS ≤ 400
CF	Fraction of the population size that undergoes Genetic crossover	Dimensionless	CF $= 0.5$

TABLE A (Continued).

INPUT PARAMETERS	MEANING/ROLE	MEASURING UNITS	VALUE RANGE
G_MAX	Positive scalar integer parameter. It is the number of generations run by the *Genetic* functions	Dimensionless	$20 \leq$ G_MAX ≤ 100
MN	Number of elements of each task allocation vector that undergo *Genetic* mutation	Dimensionless	MN = round $((V-2)/2)$
a_max_vec_FT	(3)-tuple vector of positive speed-factors of the gradient-based iterations performed by *FogTracker*	Dimensionless	Each elements of the vector a_max_vec_FT is in the range: $[7.0 \times 10^{-8}, 9.0 \times 10^{-7}]$
jump1_WiFi	Non-negative parameter. It is the 1st multiply scaling applied by *FogTracker* to R_MAX_U and R_MAX_D	Dimensionless	jump1_WiFi $= 0.0$
jump1_CELL	Non-negative parameter. It is the 1st multiply scaling applied by *FogTracker* to B_MAX_U and B_MAX_D	Dimensionless	jump1_CELL $= 1.0$
jump2_WiFi	Non-negative parameter. It is the 2nd multiply scaling applied by *FogTracker* to R_MAX_U and R_MAX_D	Dimensionless	jump2_WiFi $= 1.0$
jump2_CELL	Non-negative parameter. It specifies the 2nd multiply scaling applied by *FogTracker* to B_MAX_U and B_MAX_D	Dimensionless	jump2_CELL $= 0.0$
iteration_number	Number of iterations performed by *FogTracker*. It must be in multiples of 5	Dimensionless	iteration_number $= 5 \times 10^3$

3.2 THE *RAP_p* FUNCTION

The *RAP_p* function:

```
[y_tilde, E_TOT_tilde, E_NET_tilde] = RAP(x,z,s,Da,S_RAP),
```

implements primal-dual adaptive iterations on the cluster of available parallel workers that supports the execution of *VirtFogSim*. The goal is to perform the optimal resource allocation under the input task allocation vector x. Its input formal variables are the vectors: x, z, and s and the matrices: A,Da, and S_RAP. The corresponding output variables are the vector: y_tilde and the scalars E_TOT_tilde and E_NET_tilde.

Specifically, we have that:

i. x is a $(1 \times V)$-dimensional $\{0,1,2\}$-ternary vector. It fixes the allocation to the Fog/Mobile/Cloud nodes of the V tasks that compose the considered application DAG. Specifically, $x(i) == 0$, 1 and 2 means that the i-th application task is executed at the Fog clone, Mobile device and Cloud clone respectively;

ii. $z = [f\_M\_0, f\_F\_0, f\_C\_0, R\_U\_0, R\_D\_0, B\_U\_0, B\_D\_0, lambda\_0]$ is a (1×8) vector of real-valued non-negative scalars. It fixes the vector starting point of the primal-dual iterations to be performed. Its first seven components are measured in (bit/s), while the starting Lagrange multiplier lambda_0 is measured in $Joule$.

iii. $y\_tilde = [f\_M\_tilde, f\_F\_tilde, f\_C\_tilde, R\_U\_tilde, R\_D\_tilde, B\_U\_tilde, B\_D\_tilde]$ is a (1×7) vector of real-valued non-negative scalars. It returns the vector of the OPTIMAL resource allocation attained by the performed primal-dual iterations. Its seven components are measured in bit/s;

iv. E_TOT_tilde is the total communication-plus-computing energy consumed by the computed optimal resource allocation y_tilde under the given task allocation x. E_TOT_tilde is measured in $Joule$.

v. E_NET_tilde is the overall network energy consumed by the computed optimal resource allocation y_tilde under the given task allocation x. E_NET_tilde is measured in $Joule$;

vi. s, A and Da are respectively the global workload vector, adjacency matrix and edge matrix of the underlying DAG;

vii. S_RAP is the (1×54)-dimensional vector of all other global variables that used by RAP_p for its execution.

Since *RAP_ p* is executed by the calling functions in the body of **parfor**-cycles, the global variables: s, A, and Da, and S_RAP must be passed to the *RAP_ p* function as input parameters, in order to guarantee the synchronization of the parfor-cycles performed by multiple parallel workers.

The *RAP_ p* function requires $(110 + 6 \times V)$ scalar local variables, and two $(1 \times I\_MAX)$ vector local variables. The I_MAX parameter is stored by S_RAP and fixes the maximum number of allowed primal-dual iterations.

A pseudo-code of the *RAP_ p* function is reported in Algorithm 1. The asymptotic implementation complexity of the *RAP_ p* function scales as: $\mathcal{O}(8 \times I\_MAX)$.

Algorithm 1 — *RAP_ p* function

function: $[y\_tilde,\ E\_TOT\_tilde,\ E\_NET\_tilde] =$ RAP_ p $(x,\ z,\ s,\ Da,\ S\_RAP)$.
Input: $(1 \times |V|)$ ternary task allocation vector $x \in \{0,\ 1,\ 2\}^{|V|}$, the (1×8) vector z of the starting points of the iterations.
Output: The (1×7) vector of the allocated resources:
$y\_tilde \triangleq [f\_M\_tilde,\ f\_F\_tilde,\ f\_C\_tilde,\ R\_U\_tilde,\ R\_D\_tilde,\ B\_U\_tilde,\ B\_D\_tilde]\ (bit/s)$. The total computing-plus-networking energy $E\_TOT\_tilde\,(Joule)$ and network energy $E\_NET\_tilde\,(Joule)$ wasted by the performed resource allocation.

▷ **Begin RAP_ p function**

1: **if** the feasibility condition of RAP_ p fails **then**
2: $y := \vec{1}_7 \times$ (Inf); ▷ Set to infinite all the components of y
3: $E\_TOT\_tilde :=$ (Inf); ▷ Set to infinite E_TOT
4: $E\_NET\_tilde :=$ (Inf); ▷ Set to infinite E_NET
5: **return** $[y\_tilde,\ E\_TOT\_tilde,\ E\_NET\_tilde]$; ▷ Exit the RAP function
6: **end if**
7: **parfor** $m = 0 : (I_{MAX} - 1)$ **do**
8: Compute the 7 gradients of T_{DAG} with respect to f_M, f_F, f_C, R_U, R_D, B_U, and B_D;
9: Compute the 21 gradients of E_M, E_F, and E_C with respect to f_M, f_F, f_C, R_U, R_D, B_U, and B_D;
10: Compute the 21 gradients of the connection energies E_WiFi, E_CELL, and E_WD with respect to f_M, f_F, f_C, R_U, R_D, B_U, and B_D;
11: Update the 8 step-sizes;
12: Update the 8 gradients of the Lagrangian function;
13: Update the 8 primal-dual variables f_M, f_F, f_C, R_U, R_D, B_U, B_D, and $lambda$;
14: Update the 3 computing energies E_M, E_F, and E_C $(Joule)$;
15: Update the 3 connection energies E_WiFi, E_CELL, and E_WD $(Joule)$;
16: Update the consumed energies $E\_TOT\_tilde\,(Joule)$ and $E\_NET\_tilde\,(Joule)$
17: **end parfor**
18: $y\_tilde := [f\_M\_tilde,\ f\_F\_tilde,\ f\_C\_tilde,\ R\_U\_tilde,\ R\_D\_tilde,\ B\_U\_tilde,\ B\_D\_tilde]\ (bit/s)$;
19: **return** $[y\_tilde,\ E\_TOT\_tilde,\ E\_NET\_tilde]$. ▷ **End RAP_ p function**;

3.3 THE *Crossover* FUNCTION

The *Crossover* function:

```
[Child1_v, Child2_v] = Crossover(Parent1_v, Parent2_v),
```

assumes that the input parameters of *VirtFogSim* have been already set. Parent1_v and Parent2_v are two V-tuple parent vectors to cross-over. Child1_v and Child2_v are the resulting V-tuple vectors produced by the performed crossover operation. The crossover operation generates an integer index I over the set $\{2, 3, \ldots, (V-1)\}$. It is the pointer to the crossover point. Afterwards, Child1_v and Child2_v are obtained by swapping the first I components of Parent1_v and Parent2_v with the last $(V - I)$ components of Parent2_v and Parent1_v, respectively.

3.4 THE *Mutation* FUNCTION

The *Mutation* function:

```
out_v = Mutation(x_v),
```

assumes that the input parameters of *VirtFogSim* are set. In particular, the input parameter MN fixes the number of the components of the input vector x_v to be mutated. x_v is the V-tuple $(0, 1, 2)$-ternary input vector to be mutated. out_v is the V-tuple $(0, 1, 2)$-ternary mutated output vector. It is guaranteed that the first and last components of out_v are unit valued.

The function *Mutation* generates MN random positions (with $1 \leq \text{MN} \leq (V - 2)$) and MN random $(0, 1, 2)$-ternary mutation numbers. Then, it mutates x_v at the generated MN positions by replacing the generated MN mutation numbers to the corresponding components of x_v.

3.5 THE *GeneticTA_par* FUNCTION

The parallel Genetic Task Allocation *GeneticTA_par* function:

```
[x_best, RS_best, E_best, E_NET_best, RB_best] = GeneticTA_par,
```

assumes that all the global variables of parallel *VirtFogSim* are already setup. Furthermore, it utilizes the following main local variables:
$Init\_v(1 \times 8)$, $Ral\_v(1 \times 7)$, $E\_tilde(1 \times 1)$, $Polist\_m(PS \times (V + 8))$, $Dlist1\_m(PS \times (V + 8))$, $Dlist2\_m(PS \times (V + 8))$, $h1\_v(1 \times V)$, $h2\_v(1 \times V)$, $Ch1\_v(1 \times V)$, and $Ch2\_v(1 \times V)$.

The *GeneticTA_par* function jointly optimizes the task allocation and the resource allocation in an adaptive way. It returns:

i. the $(0, 1, 2)$-ternary V-long vector x_best of the best searched task allocation;

ii. the corresponding (1×7) vector

RS_best = [f_M_star, f_F_star, f_C_star, R_U_star, R_D_star, B_U_star, B_D_star] (bit/s),

of the optimal resource allocation computed by the *RAP_p* function under the best task allocation vector x_best;

iii. the total and network energies: E_best, E_NET_best $(Joule)$ that are consumed under the returned RS_best; and

iv. the actual transmission rate: R_best (bit/s) of the two-way Fog $\leftrightarrow$ Cloud backhaul connection.

Task allocation is performed by *GeneticTA_par* by running a Genetic algorithm. For this purpose, *GeneticTA_par* calls the *Crossover* and *Mutation* functions.

Resource allocation is performed by *GeneticTA_par* in an adaptive way. For this purpose, at each parallel iteration called by a **parfor**-cycle, *GeneticTA_par* calls the *RAP_p* function. Afterwards, *GeneticTA_par* picks up and stores:

i. the best (that is, the minimum energy) task allocation vector x_best;

ii. the corresponding resource allocation vector RS_best; and,

iii. the total consumed energy: E_best and the corresponding network energy: E_NET_best computed up to the current generation.

PS is the size of each generation of task allocation vectors, G_MAX is the number of carried out generations, CF is the fraction of cross-over elements of the current generation, and MN is the number of mutated locations on a per-task-allocation basis.

Algorithm 2 reports a pseudo-code of the *GeneticTA_par* function. Let $n_{core} \geq 1$ be the number of parallel cores (e.g., parallel workers) managed by the *Parallel Tool Box* of MATLAB, in order to support the execution of the *VirtFogSim* package. Hence, the resulting asymptotic implementation complexity scales as in: $\mathcal{O}\left((PS \times G\_MAX \times 8 \times I\_MAX) \,/\, (n\_core)\right)$

3.6 THE *OM_S* FUNCTION

The *Only Mobile Strategy* (*OM_S*) function:

```
[RS_OM, E_OM, E_NET_OM] = OM_S,
```

assumes that all the needed input parameters of *VirtFogSim* are set. It assumes that all the V tasks of the assigned application DAG are executed at the Mobile device. Then, it returns:

1. the (1×7) vector

RS_OM = [f_M, f_F, f_C, R_U, R_D, B_U, B_D] (bit/s),

of the corresponding resource allocation;

Algorithm 2 — *GeneticTA_ par* function

function: $[x\_best,\ RS\_best,\ E\_best,\ E\_NET\_best,\ RB\_best]$ = GeneticTA_ par.

Input: Population Size PS; Fraction of the Crossed over population CF; Number of Generation G_MAX; Number of mutated components per-task allocation vector MN.

Output: $(1 \times |V|)$ ternary vector of the best task allocation $x\_best \in \{0,1,2\}^{|V|}$; 1×7 best resource allocation vector: $RS\_best \triangleq [f_M^\star,\ f_F^\star,\ f_C^\star,\ R_U^\star,\ R_D^\star,\ B_U^\star,\ B_D^\star]\ (bit/s)$ under x_best, scalar total consumed energy $E\_best\ (Joule)$ and networking energy E_NET_best under x_best.

▷ **Begin GeneticTA_par function**

▷ **Initialization phase**

1: Randomly generate a list $\left\{x_i \in \{0,1,2\}^{(|V|)},\ i = 1,\ldots,PS\right\}$ of task allocation vectors and store it into [Poplist];
2: Compute and store in [Poplist] the (1×7) vector of the allocated resources and the energy for each $x \in$ [Poplist] by calling PS times the *RAP_ p* function;
3: Sort the PS element of [Poplist] for increasing values of their energies;
4: copy the first row of the sorted [Poplist] into x_best, RS_best, and E_best and E_NET_best;
5: Set the number $Q = \lceil CF \times PS \rceil$ of the elements of [Poplist] to be crossover at each generation;
6: **parfor** $j = 1 : G\_MAX$ **do** ▷ **Iterative phase**
7: Perform the pair-wise crossover of the first Q elements of [Poplist] by calling $(Q/2)$ times the **Crossover** function and store the Q crossover elements in [Childlist];
8: Randomly mute in MN positions the last $(PS - Q)$ elements of [Poplist] by calling $(PS - Q)$ times the **Mutation** function and store the mutated elements into [Mutationlist];
9: Compute and store the resource allocation vector and consumed energy of each element of [Childlist] and [Mutationlist] by calling PS times the **RAP_p** function;
10: Copy the first Q elements of [Poplist] and the overall [Childlist] and [Mutationlist] into [Candidatelist];
11: Sort the $(PS + Q)$ elements of [Candidatelist] for increasing values of their total energies;
12: Copy the first PS elements of [Candidatelist] into [Poplist];
13: **if** the total energy of the first elemnt of [Poplist] is lower than E_best **then**
14: Copy the first row of [Poplist] into x_best, RS_best, E_best and E_NET_best;
15: **end if**
16: **end parfor**
17: **return** $(x\_best,\ RS\_best,\ E\_best,\ E\_NET\_best)$. ▷ **End GeneticTA_ par function**

2. the corresponding total computing-plus-communication consumed energy E_OM $(Joule)$ and (vanishing) network energy: E_NET_OM $(Joule)$. For this purpose, the function OM_S calls the sequential version *RAP* of the *RAP_ p* function with task allocation vector:

   x_OM = ones(1,V), (i.e., all the tasks must be executed at the mobile devices),

   and initializing vector:

   init_vec = [0, 0, 0, 0, 0, 0, 0, 1].

It is expected that the returned f_M is not zero, while the returned f_C, f_F, R_U, R_D, B_U, B_D and E_NET_OM vanish. The asymptotic complexity of the *OM_S* function scales up as in: $\mathcal{O}\,(8 \times I\_MAX)$.

3.7 THE *OF_S* FUNCTION

The *Only Fog Strategy* (*OF_S*) function:

[RS_OF , E_OF , E_NET_F] = OF_S ,

assumes that all the input parameters of *VirtFogSim* are set. By design, it assumes that the first and last tasks of the assigned application DAG are executed at the Mobile device, while all the remaining inner $(V - 2)$ tasks are executed at the Fog clone. Then, *OF_S* returns:

1. the (1×7) vector

   RS_OF = [f_M, f_F, f_C, R_U, R_D, B_U, B_D] (bit/s) ,

of the corresponding resource allocation;

2. the corresponding total computing-plus-networking consumed energy E_OF and network energy $E\_NET\_F\,(Joule)$. For this purpose, the function OM_F calls the sequential version *RAP* of the *RAP_p* function with task allocation vector:

   x_OF = [1, zeros(1,V-2)],

   (i.e., all the inner tasks must be executed at the Fog clone, while the first and last tasks are executed at the Mobile device) and start vector:

   init_vec = [0, 0, 0, 0, 0, 0, 0, 1].

It is expected that the returned f_M, f_F, R_U, and R_D are not zero, while the returned f_C, B_U, and B_D vanish.

3.8 THE *OC_S* FUNCTION

The *Only Cloud Strategy* (*OC_S*) function:

[RS_OC, E_OC, E_NET_OC] = OC_S,

assumes that all the needed input parameters of *VirtFogSim* are set up. By design, it assumes that the first and last tasks of the assigned application DAG are executed at the Mobile device, while all the remaining inner $(V-2)$ tasks are executed at the Cloud clone. Then, it returns:

1. the (1×7) vector

   RS_OC = [f_M, f_F, f_C, R_U, R_D, B_U, B_D] (bit/s),

   of the corresponding resource allocation;

2. the corresponding total computing-plus-networking consumed energy E_OC $(Joule)$ and the network energy E_NET_OC $(Joule)$. For this purpose, the function OC_F calls sequential version *RAP* of the *RAP_p* function with task allocation vector:

   x_OC = [1, 2 $\times$ ones(1,V-2), 1],

   (i.e., all the inner tasks must be executed at the Cloud clone, while the first and last tasks are executed at the Mobile device) and start vector:

   init_vec = [0, 0, 0, 0, 0, 0, 0, 1].

It is expected that the returned f_M, f_C, B_U, and B_D are not zero, while the returned f_F, R_U, and R_D vanish.

3.9 THE *evaluatestaticenergy_p* FUNCTION

The *evaluatestaticenergy_p* function for parallel execution:

[Eaux, Enet] = evaluatestaticenergy_p(x_v, s, A, Da, S_RAP),

assumes that all the global variables of parallel *VirtFogSim* are already set. x_v is the input $(1 \times V)$-long $(0,1,2)$-ternary task allocation vector to be checked. Since the function *evaluatestaticenergy_p* is called in the body of **parfor**-cycle by the O_TAS_par function, all the needed global variables: s, A, Da and S_RAP must be passed to *evaluatestaticenergy_p* as input parameters. This guarantees the synchronization of the used global variables under parallel execution.

If x_v is a feasible task allocation pattern, $Eaux$ and $Enet$ are the total and network energies consumed by x_v under the static maximal resource allocation vector RS_MAX, formally defined as follows:

$$RS\_MAX = [f\_MAX\_M, f\_MAX\_F, f\_MAX\_C, R\_MAX\_U, R\_MAX\_D, B\_MAX\_U, B\_MAX\_D]\,(bit/s)\,,$$

If x_v is an infeasible task allocation pattern, then, $Eaux == Enet == Inf$ is returned. The asymptotic complexity of the *evaluatestaticenergy_p* function is: $\mathcal{O}(7)$, where 7 is the number of the computed resource variables.

3.10 THE *O_TAS_ par* FUNCTION

The parallel Only-Task Allocation Strategy *(O_TAS_par)* function:

$$[\text{x\_OTAS}\ ,\text{E\_OTAS},\text{E\_NET\_OTAS}]\ =\ \text{O\_TAS\_par}\ ,$$

acts as follows:

1. it performs task allocation by implementing (in a parallel way) a Genetic algorithm. For this purpose, *O_TAS_par* calls the *Crossover* and *Mutation* functions over a randomly generated population of $0, 1, 2$-ternary task allocation vectors of size PS;

2. it evaluates the energy of each tested ternary allocation vector by computing the corresponding energies: E_M, E_F, E_C, E_WiFi, E_WD, and E_CELL under the static (e.g., not optimized) maximal resource allocation vector:

$$RS\_MAX = [f\_MAX\_M, f\_MAX\_F, f\_MAX\_C, R\_MAX\_U, R\_MAX\_D, B\_MAX\_U, B\_MAX\_D, R\_0]\,(bit/s)\,.$$

For this purpose, *O_TAS_par* does not longer call the *RAP_p* function, but calls the *evaluatestaticenergy_ p* function.

The *O_TAS_ par* outputs are:

i. the V-tuple $(0, 1, 2)$-ternary minimum-energy task allocation vector: x_OTAS, that is computed by applying the Genetic algorithm run by *O_TAS_par* under RS_MAX; and

ii. the corresponding total energy: $E\_OTAS\ (Joule)$ and network energy: $E\_NET\_OTAS\ (Joule)$ consumed by x_OTAS under the (aforementioned) fixed maximal resource allocation vector RS_MAX.

The function *O_TAS_ par* assumes that all the global variables of parallel *VirtFogSim* are already setup. It also utilizes the following main set of local variables:
$Popmatri \times (PS \times (V+1))$, $Dmatrix1(PS \times (V+1))$, $Dmatrix2(PS \times (V+1))$, $x\_OTAS(1 \times V)$, $h1\_v(1 \times V)$, $h2\_v(1 \times V)$, $Ch1\_v(1 \times V)$, and $Ch2\_v(1 \times V)$.

Algorithm 3 reports a pseudo-code of the *O_TAS_ par* function. Due to the utilization of the **parfor**-cycle, the resulting asymptotic computational complexity of the *O_TAS_ par* function scales up as: $\mathcal{O}\left((PS \times G\_MAX)\,/n\_core\right)$, where

i. PS is the population size of the checked task allocation vectors;

ii. G_MAX is the number of iterations (that is, the number of generations) run by the Genetic algorithm implemented by O_TAS_par; and,

iii. $n\_core \geq 1$ is the number of available parallel workers that support the execution of *VirtFogSim*.

3.11 THE *ES_S_par* FUNCTION

The parallel *Exhaustive Search-Strategy (ES_S_par)* function:

$$[\text{x\_ESS}\ ,\text{RS\_ESS},\text{E\_ESS},\text{E\_NET\_ESS},\text{RB\_ESS}]\ =\ \text{ES\_S\_par}$$

assumes that all the needed global variables of *VirtFogSim* are already set up.

After its execution, it returns:

i. the V-tuple $(0, 1, 2)$-ternary x_ESS. It is the globally best task allocation vector, that is generated by performing the exhaustive search over the full population of the $3^{(V-2)}$ ternary task allocation vectors. The first and last components of each ternary task allocation vector are unit valued, i.e., the first and last tasks of the application DAG are executed, by design, at the Mobile device;

ii. the 7-tuple resource allocation vector:

$$RS\_ESS = [f\_M\_star, f\_F\_star, f\_C\_star, R\_U\_star, R\_D\_star, B\_U\_star, B\_D\_star]\,(bit/s)\,,$$

corresponding to the returned global task allocation vector x_ESS; and,

iii. the total communication-plus-computing energy: $E\_ESS\,(Joule)$, network energy: $E\_NET\_ESS\,(Joule)$, and bandwidth: $RB\_ESS\,(bit/s)$ of the backhaul connection consumed under the returned searched global best task allocation vector x_ESS.

Algorithm 3 — *O_TAS_ par* function

function: $[x\_OTAS,\ E\_OTAS,\ E\_NET\_OTAS] = O\_TAS\_par.$

Output: $(1 \times |V|)$ ternary vector of the best task allocation $x\_OTAS \in \{0,1,2\}^{|V|}$ under the *fixed* maximal resource allocation vector; scalar total energy E_OTAS $(Joule)$ and network energy E_NET_OTAS $(Joule)$ consumed by x_OTAS under the fixed maximal allocation vector.

▷ **Begin O_TAS_ par function**

▷ **Initialization phase**

1: Randomly generate a list $\left\{x_i \in \{0,1,2\}^{(|V|)},\ i = 1,\ldots,PS\right\}$ of task allocation vectors and store it into [Popmatrix];
2: Compute and store into the $(|V|+1)$-th column of [Popmatrix] the energy consumed by each $x \in$ [Popmatrix] by calling PS times the **evaluatestaticenergy_ p** function;
3: Sort the PS elements of [Popmatrix] for increasing values of their energies;
4: Copy the first row of the sorted [Popmatrix] into x_OTAS, E_OTAS and E_NET_OTAS;
5: Set the number $Q = \lceil CF \times PS \rceil$ of the elements of [Popmatrix] to be crossover at each generation;
6: **parfor** $j = 1 : G\_MAX$ **do** ▷ **Iterative phase**
7: Perform the pair-wise cross-over of the first Q elements of [PopMatrix] by calling $(Q/2)$ times the **Crossover** function and store the Q cross-overred elements in [Childlist];
8: Randomly mute MN positions of the last $(PS - Q)$ elements of [PopMatrix] by calling $(PS - Q)$ times the **Mutation** function and store the mutated elements into [Mutationlist];
9: Compute and store the consumed energy of each element of [Childlist] and [Mutationlist] by calling PS times the function **evaluatestaticenergy_ p**;
10: Copy the first Q elements of [Popmatrix] and the overall [Childlist] and [Mutationlist] into [Candidatelist];
11: Sort the $(PS + Q)$ elements of [Candidatelist] for increasing values of their energies;
12: Copy the first PS elements of [Candidatelist] into [Popmatrix];
13: **if** the total energy of the first element of [Popmatrix] is lower than E_OTAS **then**
14: Copy the first row of [Popmatrix] into x_OTAS, E_OTAS and E_NET_OTAS;
15: **end if**
16: **end parfor**
17: **return** $(x\_OTAS,\ E\_OTAS,\ E\_NET\_OTAS)$. ▷ **End O_TAS_ par function**

Being obtained through an exhaustive search, E_ESS is, by design, the global minimum total energy consumed under the considered application scenario of Fig. 1.

In order to guarantee that the full search space of the the task allocation vectors is actually explored, the *ES_S_ par* function calls the auxiliary function: *find_ allocations*, which returns the: $\left(3^{(V-2)} \times V\right)$-dimensional matrix of all possible $(0,1,2)$-ary task allocation patterns. Afterwards, *ES_S_ par* calls the *RAP_p* function under each ternary task allocation vector returned by *find_ allocations*, in order to evaluate the resulting best resource allocation vector, the corresponding total and network consumed energies, and the utilized backhaul bandwidth. After checking all the $3^{(V-2)}$ task allocations, *ES_S_ par* sorts them for increasing values of their total consumed energies, and then, picks out the first element of the attained sorted list. This last comprises the globally best task allocation x_ESS, its associated resource allocation vector: RS_ESS, the corresponding consumed energies: E_ESS and E_NET_ESS, and the utilized backhaul bandwidth: RB_ESS.

A pseudo-code of the *ES_E_ par* function is reported in Algorithm 4. The resulting asymptotic computational complexity scales up as: $\mathcal{O}\left(\left(3^{(V-2)} \times 8 \times G_{MAX}\right)/n\_core\right)$

3.12 THE *FogTracker* FUNCTION

The numerical outputs of the *FogTracker* function:

```
[energy_m_aux, net_energy_m_aux, lambda_m_aux] = FogTracker(x1, x2, x3),
```

are three matrices that store the time-behaviors of:

1. the consumed total energy $E\_TOT(Joule)$;
2. the corresponding consumed network energy $E\_NET(Joule)$; and
3. the $lambda$ multiplier $(Joule)$,

Algorithm 4 — *ES_S_ par* function

function: $[x\_ESS,\ RS\_ESS,\ E\_NET\_ESS,\ RB\_ESS] = ES\_S\_par.$

Output: $(1 \times |V|)$ ternary vector of the best task allocation over the full population x_ESS; (1×7) resource allocation vector: $RS\_ESS \triangleq [f\_M\_star, f\_F\_star, f\_C\_star, R\_U\_star, R\_D\_star, B\_U\_star, B\_D\_star]\,(bit/s)$ under x_ESS; total energy $E\_ESS\ (Joule)$ and network energy $E\_NET\_ESS\,(Joule)$ and backhaul bandwidth $RB\_ESS\,(bit/s)$ consumed by x_ESS.

▷ **Begin ES_S_ par function**

1: Call the find-allocation function, in order to generate the full set list $\left\{x_i \in \{0,1,2\}^{(|V|)},\ i = 1, \ldots, 3^{|V|-2}\right\}$ of ternary V-tuple candidate task allocation vectors, with the first and last elements set to the unit;
2: Store (by row) $\{x_i\}$ into the first $|V|$ columns of the matrix [POP];
3: **parfor** $i = 1 : 3^{V-2}$ **do**
4: Compute the resource allocation vector and the total consumed energy of each $\{x_i\}$ by calling the *RAP_ p* function and store them (by row) into the last columns of the [POP] matrix;
5: **end parfor**
6: Sort the $3^{|V|-2}$ elements of the [POP] matrix for increasing values of their total energies;
7: Store the first row of the sorted [POP] matrix into x_ESS, RS_ESS, E_ESS, E_NET_ESS and RB_ESS;
8: **return** $(x\_ESS,\ RS\_ESS,\ E\_ESS,\ E\_NET\_ESS,\ RB\_ESS)$. ▷ **End ES_S_ par function**

for values of the iteration index going from: 1 to the given: $iteration\_number$. The goal of *FogTracker* is to test the convergence rate to the steady-state and the steady-state stability of the primal-dual iterations performed by the *RAP_ p* function when abrupt changes of the maximum allowed WiFi and Cellular up/down bandwidths and task allocation vectors simultaneously happen.

For this purpose, at the time indexes:

i. $round((1/5) \times iteration\_number) + 1$

ii. $round((2/5) \times iteration\_number) + 1$

iii. $round((3/5) \times iteration\_number) + 1$

iv. $round((4/5) \times iteration\_number) + 1$

the initial values of the global variables: R_MAX_U, R_MAX_D, B_MAX_U, B_MAX_D, undergo abrupt changes. These changes are obtained by multiplying them by the (non-negative) input factors: $jump1\_WiFi$, $jump1\_CELL$, $jump2\_WiFi$, and $jump2\_CELL$. These jump coefficients are declared as global variables and set by the user. At the same time, the task allocation vector passes from: $x1$ to: $x2$, and then, from: $x2$ to: $x3$. Finally, it comes back to $x1$. These test allocation vectors are passed to the *FogTracker* function as input parameters by the main program *VirtFogSim*.

The resulting time behaviors of the total energies, network energies and lambda multiplier values over the time window: $[1, iteration\_number]$ are plotted under the three user-specified values of a_max, that are stored by the (1×3) global input vector $a\_max\_vec\_FT$. Furthermore, $x1$, $x2$, and $x3$ are three (possibly, partially coincident) $(0, 1, 2)$-ary task allocation vectors. They are passed as input parameters to the FogTracker function when it is called by the main program VirtFogSim. So doing, the FogTracker function tests the convergence rate of the *RAP_ p* function when the task allocation vector undergoes abrupt changes. The actual feasibility of the passed task allocation vectors: $x1$, $x2$, and $x3$ are explicitly check in the body of the *FogTracker* function. If at least one of these feasibility checks fails, the *FogTracker* function generates a suitable error message and, then, halts to run.

In detail, the *FogTracker* function performs the following nine steps:

1) it begins to run the *RAP_ p* function over the time interval: $[1, round((1/5) \times iteration\_number)]$, under:
the original settings of: R_MAX_U, R_MAX_D,B_MAX_U,B_MAX_D dictated by the main program *VirtFogSim* and the first input task allocation vector $x1$;

2) at the end of iteration number: $round((1/5) \times iteration\_number)$, the WiFi maximum bandwidths: R_MAX_U, R_MAX_D are multiplied by the (non-negative) scaling factor: $jump1\_WiFi$, while the maximum cellular bandwidths: B_MAX_U and B_MAX_D are multiplied by the corresponding (possibly, coincident) scaling factor: $jump1\_CELL$. Furthermore, the task allocation vector is changed into $x2$. So doing, both the maximum available bandwidths and the task allocation vector undergo abrupt (typically, user mobility induced) variations;

3) the *RAP_ p* function runs over the time-interval:
$[round(\ (1/5) \times iteration\_number) + 1, round(\ (2/5) \times iteration\_number)]$
under the setting of Step 2. Its initial vector is the last returned vector of the *RAP_ p* at the previous iteration: $round((1/5) \times iteration\_number)$;

4) at the end of iteration number: $round((2/5) \times iteration\_number)$, all the four maximum bandwidths: R_MAX_U, R_MAX_D, B_MAX_U, B_MAX_D are restored to their original values. Furthermore, even the task allocation vector is set back to its original value $x1$;

5) the *RAP_ p* function runs over the time interval:
$[round(\ (2/5) \times iteration\_number) + 1, round(\ (3/5) \times iteration\_number)]$
under the setting of Step 4. Its initial vector is the last returned vector of the *RAP_ p* at the previous iteration number: $round((2/5) \times iteration\_number)$;

6) after the iteration number:$round((3/5) \times iteration\_number)$, the WiFi maximum up/down bandwidths: R_MAX_U, R_MAX_D are multiplied by the non-negative scaling factor $jump2\_WiFi$, while the cellular maximum up/down bandwidths: B_MAX_U, B_MAX_D are multiplied by the (possibly, coincident) scaling factor: $jump2\_CELL$. At the same time, even the task allocation vector is changed to the third value $x3$. So doing, both the maximum available bandwidths and the task allocation vector undergo abrupt (typically, user mobility induced) variations;

7) the *RAP_ p* function runs over the time-interval:
$[round(\ (3/5) \times iteration\_number) + 1, round(\ (4/5) \times iteration\_number)]$
under the setting of Step 6. Its initial vector is the last returned vector of the RAP_p at the previous iteration number: $round((3/5) \times iteration\_number)$;

8) after the iteration number: $round((4/5) \times iteration\_number)$, all the four WiFi/CELL maximum up/down bandwidths are restored to their original values. Furthermore, even the task allocation vector is set back to the first value $x1$;

9) finally, the *RAP_ p* function runs over the time-interval:
$[round(\ (4/5) \times iteration\_number) + 1, iteration\_number)]$
under the setting of Step 8. Its initial vector is the last returned vector of the *RAP_ p* at the previous iteration number: $round((3/5) \times iteration\_number)$.

Graphic plots of the time-behaviors of the returned E_TOT, E_NET, and $lambda$ are displayed at the end of each $FogTracker$ run (see Section 5 in the sequel). From the outset, it follows that the asymptotic complexity of the *FogTracker* function scales up as in: $\mathcal{O}\left(8 \times iteration\_number \times 3\right)$

3.13 COMPUTATIONAL COMPLEXITY OF *VIRTFOGSIM*

Table B reports a synoptic view of the asymptotic computational complexities of the described *GeneticTA_ par*, *OM_S*, *OF_S*, *OC_S*, *O_TAS_ par*, *ES_S_ par* and *FogTracker* functions under their parallel implementation.

TABLE B. Asymptotic computational complexities of the main functions implemented by the version 4.0 of parallel *VirtFogsim*.

Function	***Asymptotic computational complexity***
GeneticTA_ par	$\mathcal{O}\left((PS \times G\_MAX \times 8 \times I\_MAX)\ /\ (n\_core)\right)$
OM_ S	$\mathcal{O}\left(8 \times I\_MAX\right)$
OF_ S	$\mathcal{O}\left(8 \times I\_MAX\right)$
OC_ S	$\mathcal{O}\left(8 \times I\_MAX\right)$
O_TAS_ par	$\mathcal{O}\left((PS \times G\_MAX)\ /n\_core\right)$
ES_S_ par	$\mathcal{O}\left(\left(3^{(V-2)} \times 8 \times G_{MAX}\right)\ /n\_core\right)$
FogTracker	$\mathcal{O}\left(8 \times iteration\_number \times 3\right)$

Consequently, the resulting asymptotic computational complexity of each execution of the *VirtFogSim* package that runs all the above reported seven functions scale as:

$$\mathcal{O}\left(\left(8 \times I\_MAX \times \left[(PS \times G\_MAX) + 3^{(V-2)} + 3\right] + PS \times G\_MAX\right)/n\_core + 8 \times iteration\_number \times 3\right),$$

as a function of the values of the main input parameters.

4 THE SUPPORTED DUAL-MODE USER INTERFACES

The current version 4.0 of the simulator supports two user interfaces, thereinafter referred to as *VirtFogSim* and *VirtFogSim Graphic User Interface (VirtFogSimGUI)*, respectively. As detailed in the following two sub-sections, both interfaces make available the same set of basic optimization routines of Table B, and, then, they provide the same set of numerical results. However,

- the VirtFogSim interface is oriented to a scientific use of the simulator. Its utilization requires some basics about the MATLAB environment. Hence, it may be appealing for skilled research users who desire to work in an interactive way and are mainly interested to check and optimize the performance of own customized DAGs under (possibly, multiple) customized simulation setups;
- the *VirtFogSimGUI* interface provides a rich set of self-explicative ready-to-use native facilities that allow less (or even not) skilled users to directly run the simulator under a number of pre-loaded (but, in any case, customizable) application scenarios. Hence, since its utilization does not require any specific skill about the MATLAB environment, it allows the user to interact with the simulator as a "black-box". For this purpose, (i) the obtained numerical results are rendered in form of graphics, colored maps and plots, in order to make more intuitive their interpretation and mining; and (ii) an easy-to-consult on-line version of this user guide is also enclosed.

4.1 THE *VirtFogSim* INTERFACE

This interface is activated by entering the command:

```
VirtFogSim
```

in the command line of a running MATLAB session. The script acts as the main program of the *VirtFogSim* package. This script allows the user to select any subset of the natively supported optimization algorithms by setting some binary flags to 1 (for running it) or to 0 (for do not running it), and to choose which DAG should be analyzed by writing the name of the related configuration script. If the user desires to change the configuration, the script defining the DAG can be opened in the editor window of MATLAB. A screen-shot of this script is shown in Fig. 5.

By programming in the MATLAB language, it allows the user to:

- edit the desired DAG by setting the corresponding workload vector s, adjacency matrix A and edge weight matrix Da;
- define the desired simulation setup by editing the input parameters listed in Table A;
- select any subset of the supported optimization functions of Table B;
- resort to the on-line *help* of MATLAB, in order to display information about any specific function supported by the *VirtFogSim* package; and,
- call user-defined customized functions that are not natively supported by the current version 4.0 of the simulator.

Depending of the selected functions (see Table B), after running *VirtFogSim*, the attained numerical results are displayed in tabular form, and/or as colored time-plots, and/or as colored bar plots, and/or as colored DAG maps (see Section 5 for additional details about the graphic rendering capabilities and related options supported by the simulator).

4.2 THE *VirtFogSimGUI* INTERFACE

The GUI interface is open by entering the command:

```
VirtFogSimGUI
```

in the command line of a running MATLAB session. The screen-shoot of the displayed graphic window is reported in Fig. 6

This interface is supported by the MATLAB code: *VirtFogSim_ Run*, that acts as the main script of the overall simulator.

An examination of Fig. 6 points out that the GUI interface of the version 4.0 of the simulator supports seven pre-built functions (namely, *Help*, *Algorithm*, *Archived DAG*, *Edit DAG*, *Save DAG*, *Run* and *Close*), that may be activated by the user

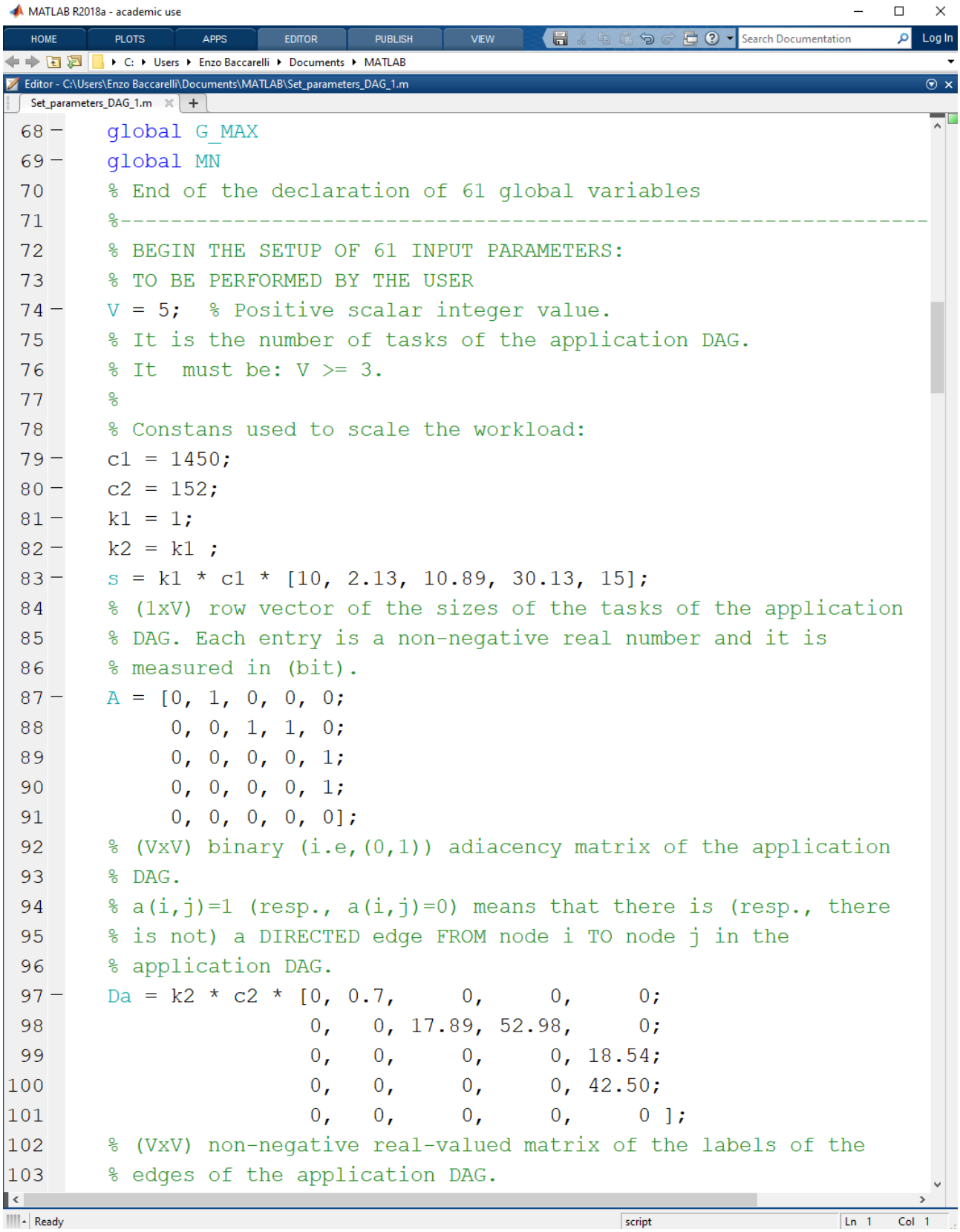

FIGURE 5. A screen-shot of the *VirtFogSim* interface

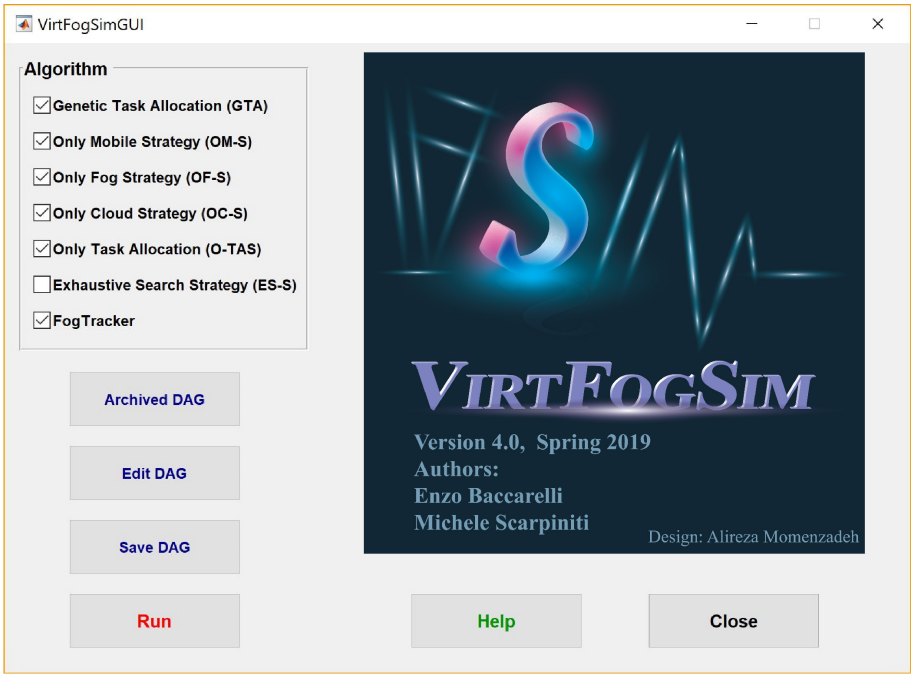

FIGURE 6. A screen-shot of the *VirtFogSim Graphic User Interface*.

TABLE C. Native functions supported by the version 4.0 of the *VirtFogSimGUI* interface and associated actions.

Available GUI Functions	*Associated Actions*
Help	Allow to access the User Guide of the simulator by opening a dedicated PDF file
Algorithm	Allow to select any subset of the natively supported optimization algorithms by clicking the corresponding labels
Archived DAG	Allow to retrieve an already archived DAG with the corresponding simulation setup, in order to run it
Edit DAG	Allow to edit a new DAG and/or a new simulation setup by compiling the list of input parameters of Table A
Save DAG	Allow to save the lastly edit DAG and assign it an identification label
Run	Allow to run the selected optimization algorithm under the retrieved/edit DAG and associated simulation setup
Close	Shut down the current working session of the *VirtFogSim* simulator

with simply clicking over the corresponding bottoms. Table C lists these native GUI functions and points out their meaning and associated actions.

From the user perspective, a typical *VirtFogGUI*-enabled working session should proceed according to the following ordered list of steps:

i. enter ***VirtFogGUI*** at the MATLAB prompt, in order to open the GUI interface;

ii. click the ***Help*** bottom, in order to access the PDF version of the User Guide. This step is optional;

iii. select any desired subset of the supported optimization procedures by clicking the corresponding labels in the ***Algorithm*** window. This step is mandatory;

iv. retrieve an already stored DAG in *.mat format and the associated list of input parameters by accessing the ***Archived DAG***. Alternatively, build up a new DAG and set the desired input parameters of Table A by clicking the ***Edit DAG*** bottom. This step is mandatory;

v. store in the ***Archived DAG*** database the lastly edit DAG in *.mat format by clicking the Save DAG bottom. An identification label for the ***Save DAG*** is required. This step is optional;

vi. click the ***Run*** bottom, in order to start the execution of the selected algorithms under the (retrieved or edit) selected DAG. At the end of the execution, the attained results are returned in tabular form, and/or as colored time-plots, and/or as colored bar plots and/or as colored DAG maps (see the Section 5 for additional details about the graphic rendering capabilities and related options supported by the simulator);

vii. after finishing the current working session of the simulator, click the ***Close*** bottom, in order to shut down the *VirtFogSimGUI* interface.

5 THE SUPPORTED GRAPHIC FORMATS OF THE RENDERED DATA

Under the current version 4.0 of the simulator, both the *VirtFogSim* and *VirtFogSimGUI* interfaces support four main formats, in order to render the results output by the seven optimization algorithms of Table B. These rendering formats are:

- the **Tabular** format. It is enabled by the *print_ solution* graphic function;
- the **Colored Bar Plot** format. It is enabled by the *plot_ solution* graphic function;
- the **Colored Time Plot** format. It is enabled by the *plot_ FogTracker* graphic function;
- the **Colored Labeled DAG Map** format. It is enabled by the *plot_ DAG* and *customTextNode* graphic function.

Table D gives a synoptic view of the rendering formats available for each optimization algorithm, together with the required supporting rendering functions.

As illustrative examples, the following Figs. 7, 8 and 9 report the screenshots of the Tabular, Bar and DAG Map rendering formats we have obtained by running the *GeneticTA_par* algorithm under a test DAG. In particular,

i. Fig. 7 reports the numerical values of all involved optimization variables of Eq. (5), together with the corresponding values of the consumed total: E_TOT and network: E_NET energies;

ii. Fig. 8 reports the corresponding: (i) task allocation (upper part); (ii) computing frequencies and consumed bandwidths (middle part); and, (iii) consumed energies (bottom part), in form of colored bar plots; and,

iii. Fig. 9 reports a labeled version of the considered DAG, in which the colors of the DAG nodes correspond to the allocation of the tasks actually performed by *GeneticTA_ par* on the Mobile (Green color), Fog (Red color) and Cloud (Azure color) computing infrastructures.

Similar screenshots are rendered when the *OM_S*, *OC_S*, *OF_S*, *O_TAS_par* and *ES_S_par* optimization algorithms are run (see the first six rows of Table C).

Finally, Fig. 10 is an example of the time plots that are obtained by running the *FogTracker* function in correspondence of three values of the step-size a_max (see Table A). Specifically, the upper, middle and bottom parts of Fig. 10 report the time behaviors of E_TOT, E_NET and $lambda$ multiplier that are obtained under the (aforementioned) illustrative scenario.

In the following sub-sections, we pass to (shortly) describe the syntax and semantic of the rendering functions listed in the last column of Table C.

5.1 THE *print_solution* rendering function

The *print_solution* function:

```
print_solution(RS,E,RB,strategy)
```

prints on the MATLAB prompt the result obtained by running the tested strategy under the selected DAG and given input parameters (see Table A).

Its input parameters are:

i. RS, e.g., the vector collecting the 7 allocated resources;

TABLE D. Formats of the rendered data done available by the version 4.0 of the *VirtFogSim* and *VirtFogSimGUI* interfaces and supporting rendering functions.

	Tabular Format	*Colored Bar Plot*	*Colored Time Plot*	*Colored Labeled DAG Map*	*Supporting Rendering Functions*
GeneticTA_par	✓	✓	✗	✓	• *print_solution* • *plot_solution* • *plot_DAG* • *customTextNode*
OM_S	✓	✓	✗	✓	• *print_solution* • *plot_solution* • *plot_DAG* • *customTextNode*
OF_S	✓	✓	✗	✓	• *print_solution* • *plot_solution* • *plot_DAG* • *customTextNode*
OC_S	✓	✓	✗	✓	• *print_solution* • *plot_solution* • *plot_DAG* • *customTextNode*
O_TAS_par	✓	✓	✗	✓	• *print_solution* • *plot_solution* • *plot_DAG* • *customTextNode*
ES_S_par	✓	✓	✗	✓	• *print_solution* • *plot_solution* • *plot_DAG* • *customTextNode*
FogTracker	✗	✗	✓	✗	• *plot_FogTracker*

ii. E, e.g., the vector collecting the total computing–plus-networking consumed energy;

iii. RB, e.g., the bandwidth of the Fog↔Cloud two-way backhaul connection; and,

iv. $strategy$, e.g., the name of the tested strategy to be used as label of the overall print.

The function prints the following results in a numerical form:

- the computing/communication ratio (*CCR*);
- the required minimum DAG execution throughput (*TH_MIN_0*);
- the per-core computing frequency at the Mobile device (f_M);
- the per-core computing frequency at the Fog clone (f_F);
- the per-core computing frequency at the Cloud clone (f_C);
- the transport bit rate of the WiFi-based TCP/IP Mobile-to-Fog connection (R_U);
- the transport bit rate of the WiFi-based TCP/IP Fog-to-Mobile connection (R_D);
- the transport bit rate of the 3G/4G Cellular TCP/IP Mobile-to-Cloud connection (B_U);
- the transport bit rate of the 3G/4G Cellular TCP/IP Cloud-to-Mobile connection (B_D);
- the transport rate of the TCP/IP two-way Fog↔Cloud backhaul connection (R_B);

```
DAG with a CCR of 0.492
Throughput T_DAG: 3.333 (app/s)
Resource allocation performed by GeneticTA:
f_M:        11.947 (Mb/s)
f_F:         2.532 (Mb/s)
f_C:         2.796 (Mb/s)
R_U:         7.641 (Mb/s)
R_D:         8.708 (Mb/s)
B_U:         1.424 (Mb/s)
B_D:         6.784 (Mb/s)
RB:          3.707 (Mb/s)
-------------------------
E_T:        29.517 (J)
E_C:        19.966 (J)
E_N:         9.550 (J)
E_N / E_T:  32.36 (%)

End the run of the GeneticTA function

Begin the run of the OM-S function

DAG with a CCR of 0.492
Throughput T_DAG: 3.333 (app/s)
Resource allocation performed by OM-S:
f_M:        11.796 (Mb/s)
f_F:         0.000 (Mb/s)
f_C:         0.000 (Mb/s)
R_U:         0.000 (Mb/s)
R_D:         0.000 (Mb/s)
B_U:         0.000 (Mb/s)
B_D:         0.000 (Mb/s)
RB:          0.000 (Mb/s)
-------------------------
E_T:        90.299 (J)
E_C:        90.299 (J)
E_N:         0.000 (J)
E_N / E_T:   0.00 (%)

End the run of the OM-S function
```

FIGURE 7. An illustrative screenshot of the rendered Tabular format.

- the total computing-plus-networking consumed energy (E_TOT);
- the consumed computing energy (E_COM);
- the consumed networking energy (E_NET);
- the per-cent networking-to-total ratio of the consumed energies: $\%\left((E\_NET)\,/\,(E\_TOT)\right)$.

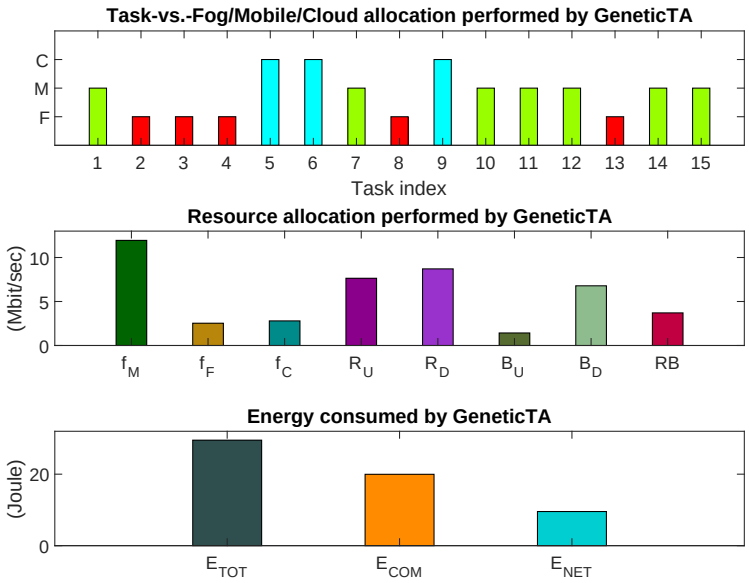

FIGURE 8. An illustrative screenshot of the rendered Bar Plot format.

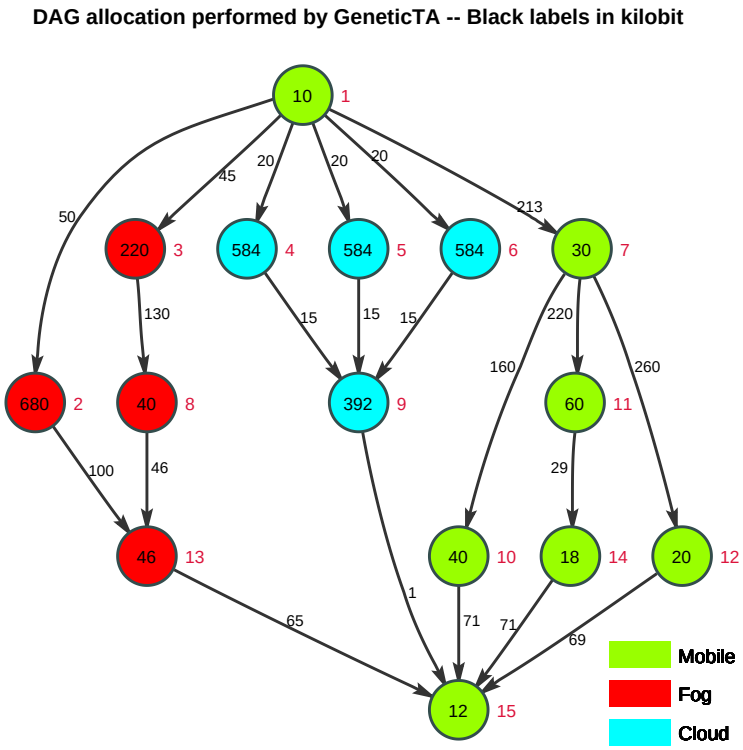

FIGURE 9. An illustrative screenshot of the rendered Labeled DAG Map format.

If the input vector RS is empty, only a reduced set of information is printed.

5.2 THE *plot_solution* rendering function

The *plot_solution* function:

```
plot_solution(x,RS,E,RB,fignumber,strategy)
```

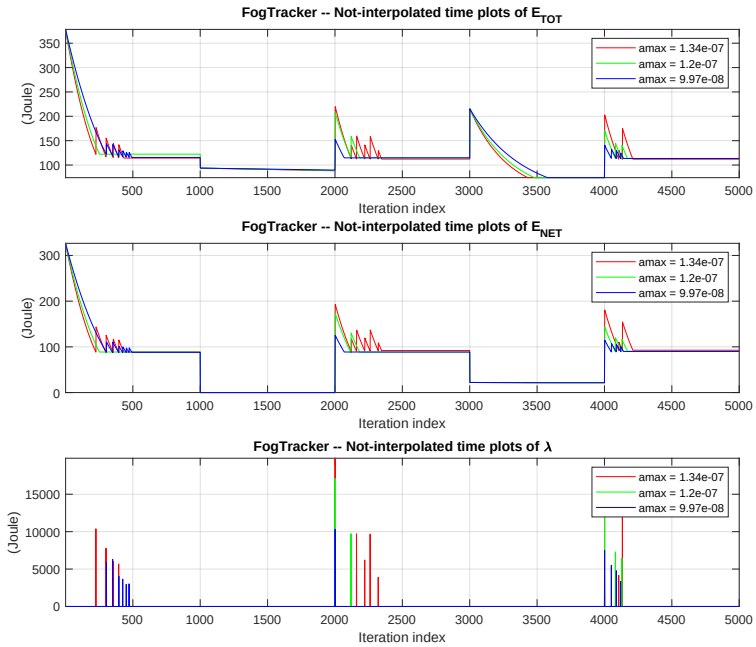

FIGURE 10. An illustrative screenshot of the time-plots rendered by a run of the *FogTracker* function.

displays the: (i) task allocation pattern; (ii) utilized computing/bandwidth resources; and, (iii) consumed energies, returned by the carried out *VirtFogSim* run in terms of suitable three-colored bar plots.

Its input parameters are:

- x, e.g, the $(0, 1, 2)$-ary allocation vector to be displayed;
- RS, e.g., the vector of the corresponding consumed resources;
- E, e.g., the 3-dimensional vector of the consumed total, computing and networking energies;
- RB, e.g., the utilized transport bit rate of the two-way TCP/IP Fog↔Cloud backhaul connection;
- $fignumber$, e.g., the number of the figure to be displayed;
- $strategy$, e.g., the name of the tested strategy to be used in the title of the figure.

The function renders a figure that is composed by three horizontal subplots. These subplots display:

1) the returned task allocation in the form of a three-color bar plot;
2) the allocated computing frequencies and network bandwidths in the form of a plot with eight bars; and,
3) the total, computing and networking consumed energies in the form of a plot with three bars.

If the passed task allocation vector x and/or resource allocation vector RS are empty, then, only one or two subplots are rendered.

5.3 THE *plot_DAG* rendering function

The *plot_DAG* function:

```
plot_DAG(s,A,Da,x,strategy,type)
```

returns a graphic representation of the Directed Acyclic Graph (DAG) passed as input. Its input parameters are:

- s, e.g., the vector containing the nodes' weights of the DAG;

- A, e.g., the adjacency matrix of the DAG;
- Da, e.g., the matrix of the weights of the DAG edges;
- x, e.g., the vector of task allocation to be visualised;
- $strategy$, e.g., a string of characters with the name of the policy that is used for computing the colored map of the displayed tasks allocation; and,
- $type$, e.g., an optional three-valued parameter, that allows to select the desired DAG visualisation format as follows:
  - $\star$ $type == 1$ all DAG nodes with the same color and un-labeled;
  - $\star$ $type == 2$ all DAG nodes with the same color and labeled by increasing identification numbers;
  - $\star$ $type == 3$ each DAG node is numbered and colored on the basis of the allocation actually stored by the input vector x.

The function *plot_DAG* calls the auxiliary function: *customTextNode*, in order to provide the aforementioned customized labeling of the DAG nodes.

By default, the graph returned by *plot_DAG* retains the following features:

i. all the workload values labeling the DAG nodes and edges are expressed in kilobit;

ii. red-colored tasks are understood to be allocated to the Fog node;

iii. green-colored tasks are understood to be allocated to the Mobile node; and,

iv. azure-colored tasks are understood to be allocated to the Cloud node.

5.4 THE *customTextnode* rendering function

The *customTextnode* function:

```
hg_handles = customTextnode(node_handle)
```

is called by the *plot_DAG* function, in order to draw a customized graph of the underlying DAG. Its input and output parameters are of *struct*-type.

The function allows the user to customize the rendered DAG nodes by:

i. selecting the color of each DAG node accordingly to its allocation (Mobile, Fog or Cloud); and/or,

ii. adding a further label to the right of each DAG node that reports its integer-valued identification number.

5.5 THE *plot_FogTracker* rendering function

The *plot_fogTracker* function:

```
plot_FogTracker(energy_m, net_energy_m, lambda_m, fignumber, interpolation)
```

provides the graphic capabilities needed for a proper plot of the time-traces of the total energies, networking energies and lambda multipliers generated by the FogTracker function under the three values of the step-size that are stored by the vector *a_ma_vec_FT* (see Table A). Its input parameters are:

- *energy_m*, e.g., the matrix of the total consumed energies returned by *FogTracker*;
- *net_energy_m*, e.g., the matrix of the consumed networking energies returned by *FogTracker*;
- *lambda_m*, e.g., the matrix of the values of the lambda multipliers returned by *FogTracker*;
- *fignumber*, e.g., the number of the figure to be displayed;
- *interpolation*, e.g., an optional parameter allowing the plot of an interpolated version of the figures.

The function renders a figure composed by three horizontal sub-plots, that trace:

1) the total consumed computing-plus-networking energies under the three values of the step-size stored by *a_max_vec_FT*;

2) the corresponding consumed networking energies; and,

3) the related values of the lambda multiplier.

6 THE SET OF PRE-LOADED APPLICATION DAGs

The (aforementioned) archive of the current version 4.0 of the *VirtFogSimGUI* interface (see Fig. 6) stores ten test DAGs of application interest, together with the related sets of (suitably tuned) input parameters. These DAGs are ready-for-the-use, e.g., they may be retrieved by the interested user and, then, run under both the (previously described) interfaces of the simulator.

The archived DAGs are retrieved from the current literature [10], [11], [12], [14], [15], [16], [17], [18] and feature a number of heterogeneous real-world applications of practical interest. Their topologies cover a large spectrum (e.g., tree, fork, parallel, mesh and hybrid topologies, just to name a few), and their number of nodes ranges from: $V = 9$, to: $V = 45$.

The remaining part of this section reports the defining workload vector s, adjacency matrix A and matrix of the edge weights Da of these preloaded DAGs, together with the corresponding labeled graphs.

6.1 DAG2

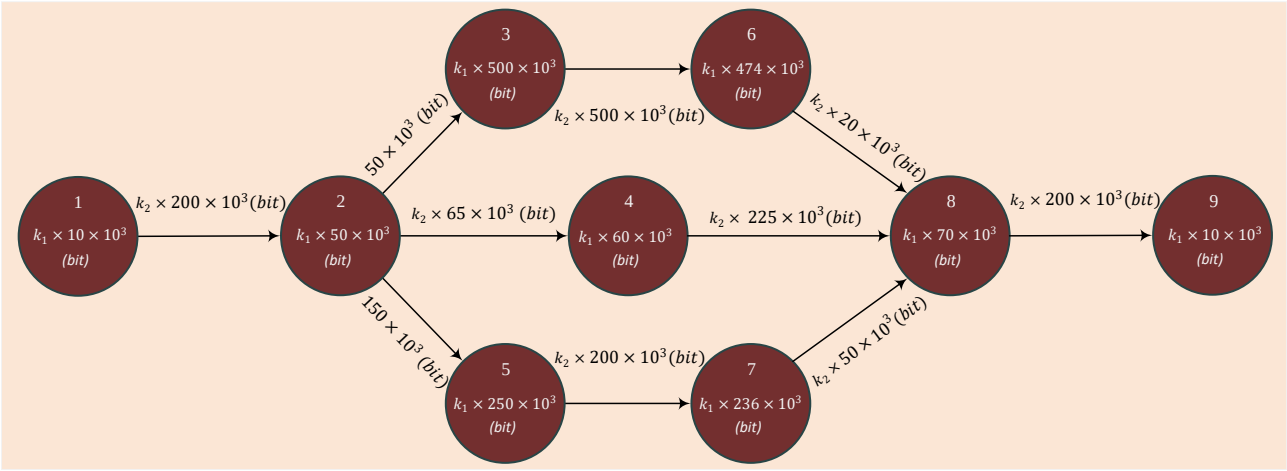

FIGURE 11. DAG2

$$\vec{s} = k_1 \times c_1 \times [10, 50, 500, 60, 250, 474, 236, 70, 10]\ (bit)\ ,$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$D_a = k_2 \times c_2 \times \begin{bmatrix} 0 & 200 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 50 & 65 & 150 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 225 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 200 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 20 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 200 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} (bit) .$$

6.2 DAG3

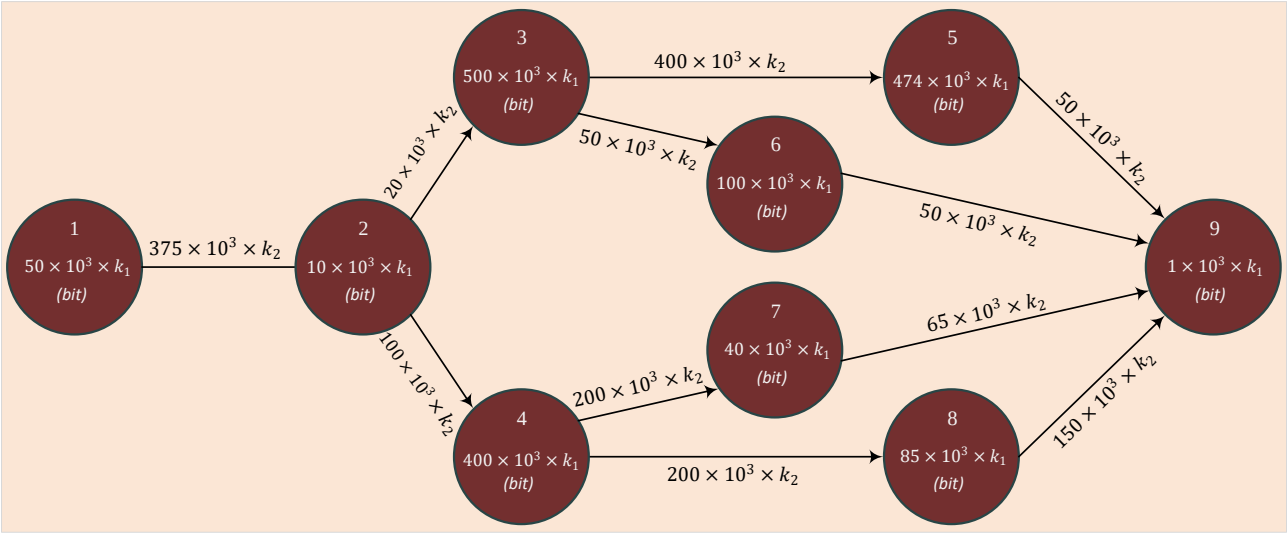

FIGURE 12. DAG3

$$\vec{s} = k_1 \times c_1 \times [50, 10, 500, 400, 474, 100, 40, 85, 1] \, (bit) ,$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} ,$$

$$D_a = k_2 \times c_2 \times \begin{bmatrix} 0 & 375 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 20 & 100 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 400 & 50 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 200 & 200 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 50 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 50 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 65 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 150 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} (bit) .$$

6.3 DAG4

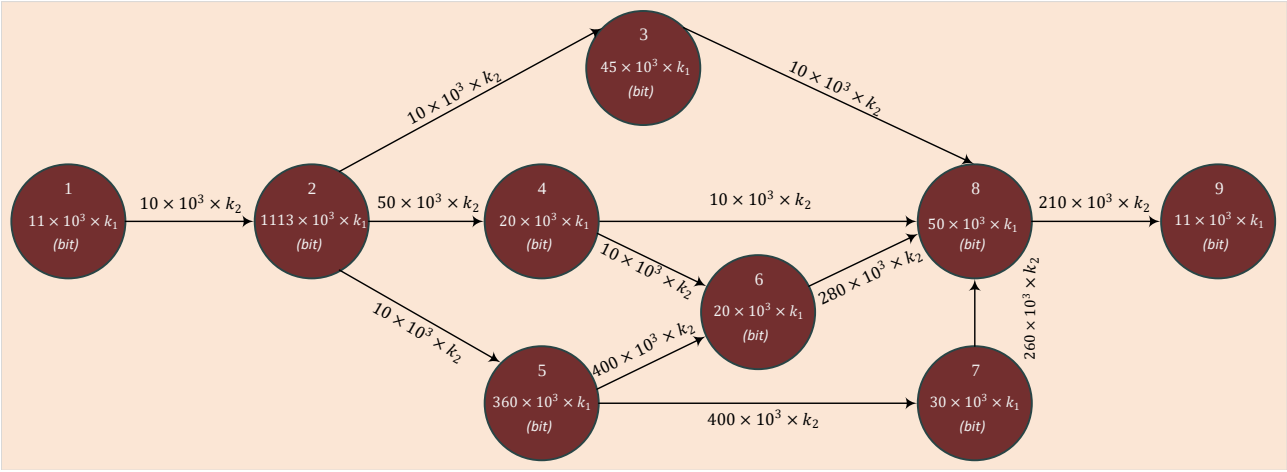

FIGURE 13. DAG4

$$\vec{s} = k_1 \times c_1 \times [11, 1113, 45, 20, 360, 20, 30, 50, 11] (bit) ,$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} ,$$

$$D_a = k_2 \times c_2 \times \begin{bmatrix} 0 & 10 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 50 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 400 & 400 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 280 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 260 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 210 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} (bit) .$$

6.4 DAG5

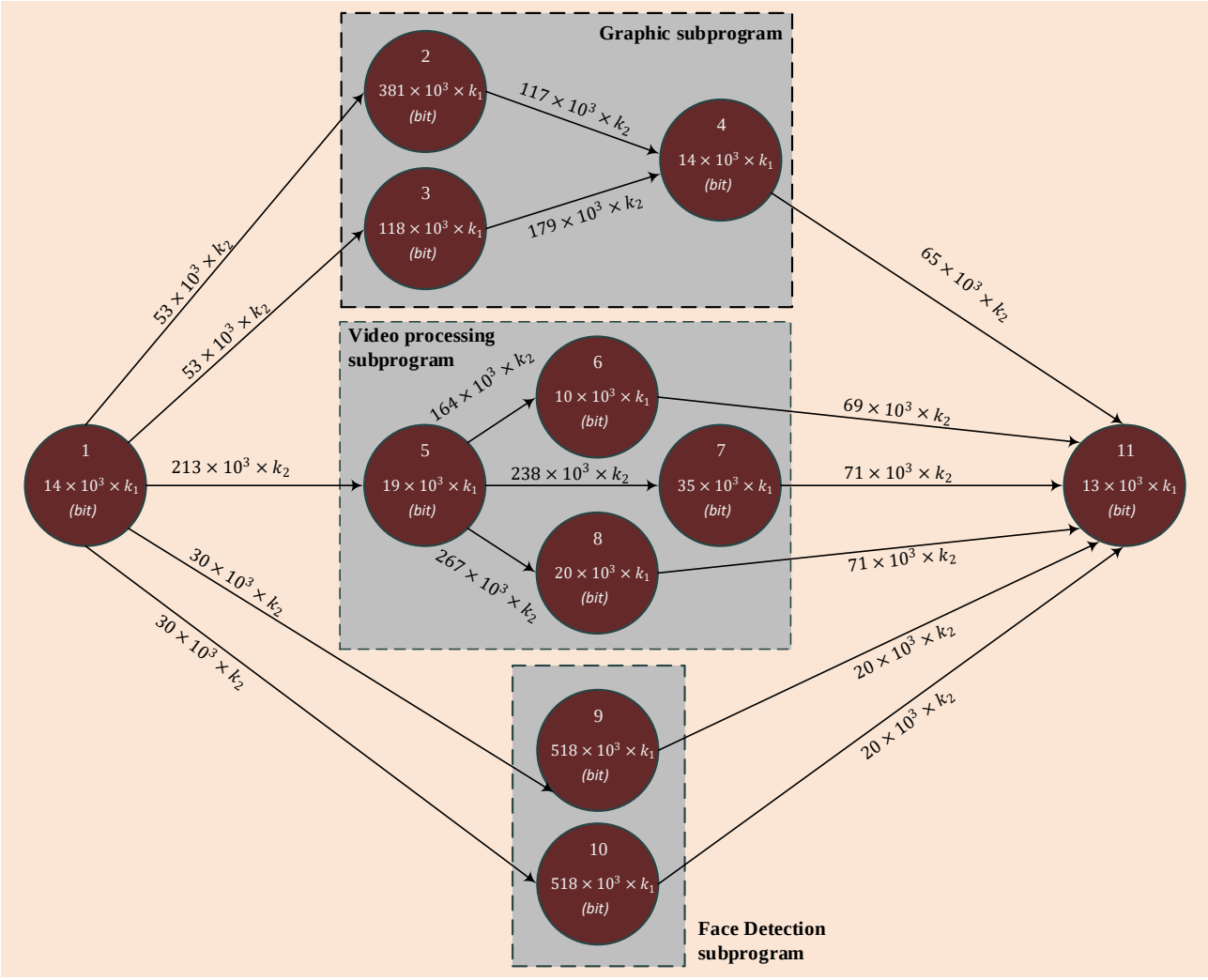

FIGURE 14. DAG5

$$\vec{s} = k_1 \times c_1 \times [14, 381, 118, 14, 19, 10, 35, 20, 518, 518, 13]\,(bit)\,,$$

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$D_a = k_2 \times c_2 \times \begin{bmatrix} 0 & 53 & 53 & 0 & 213 & 0 & 0 & 0 & 30 & 30 & 0 \\ 0 & 0 & 0 & 117 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 179 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 65 \\ 0 & 0 & 0 & 0 & 0 & 164 & 238 & 267 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 69 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 71 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 71 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 20 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 20 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} (bit)\,.$$

6.5 DAG6

$$\vec{s} = k_1 \times c_1 \times [1, 114, 300, 300, 300, 89, 50, 140, 140, 140, 85, 1]\,(bit)\,,$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

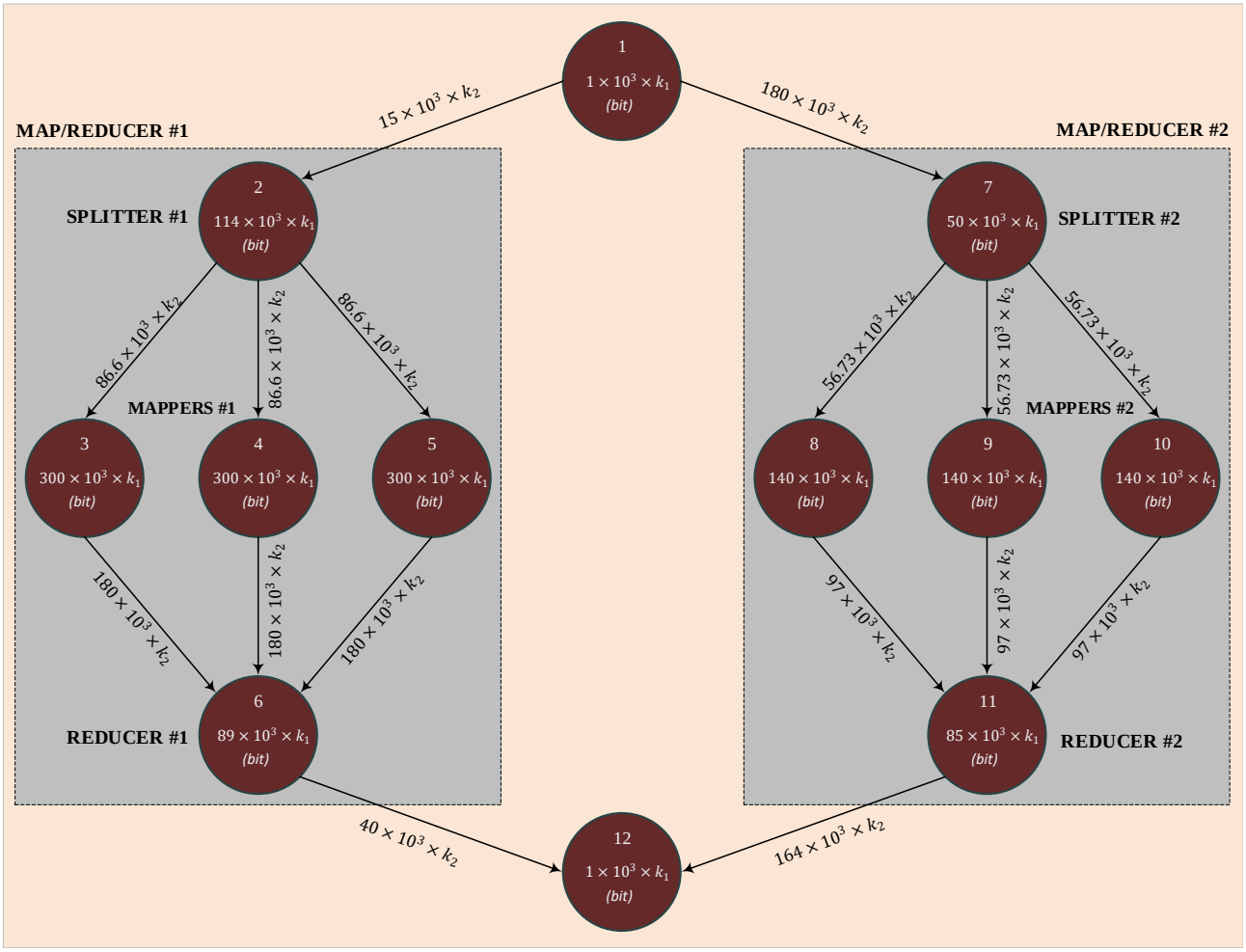

FIGURE 15. DAG6

$$D_a = k_2 \times c_2 \times \begin{bmatrix} 0 & 15 & 0 & 0 & 0 & 0 & 180 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 86.6 & 86.6 & 86.6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 180 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 180 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 180 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 40 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 56.73 & 56.73 & 56.73 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 97 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 97 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 97 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 164 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} (bit) .$$

6.6 DAG7

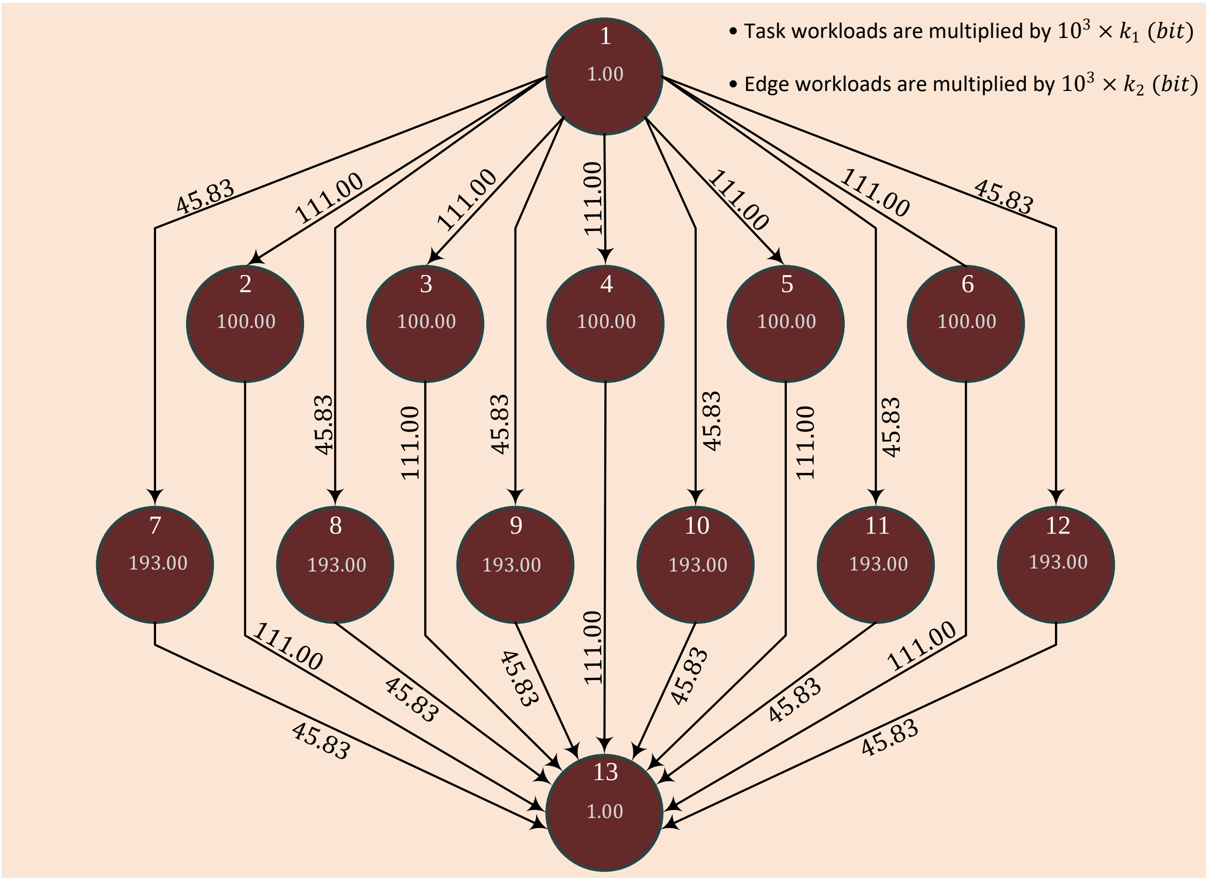

FIGURE 16. DAG7

$$\vec{s} = k_1 \times c_1 \times [1, 100, 100, 100, 100, 100, 193, 193, 193, 193, 193, 193, 1]\,(bit)\,,$$

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$D_a = k_2 \times c_2 \times \begin{bmatrix} 0 & 111 & 111 & 111 & 111 & 111 & 45.83 & 45.83 & 45.83 & 45.83 & 45.83 & 45.83 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 111 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 111 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 111 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 111 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 111 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 45.83 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 45.83 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 45.83 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 45.83 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 45.83 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 45.83 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} (bit) .$$

6.7 DAG8

$$\vec{s} = k_1 \times c_1 \times [5, 320, 320, 320, 320, 30, 60, 60, 60, 70, 30, 30, 30, 5] \, (bit) ,$$

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} ,$$

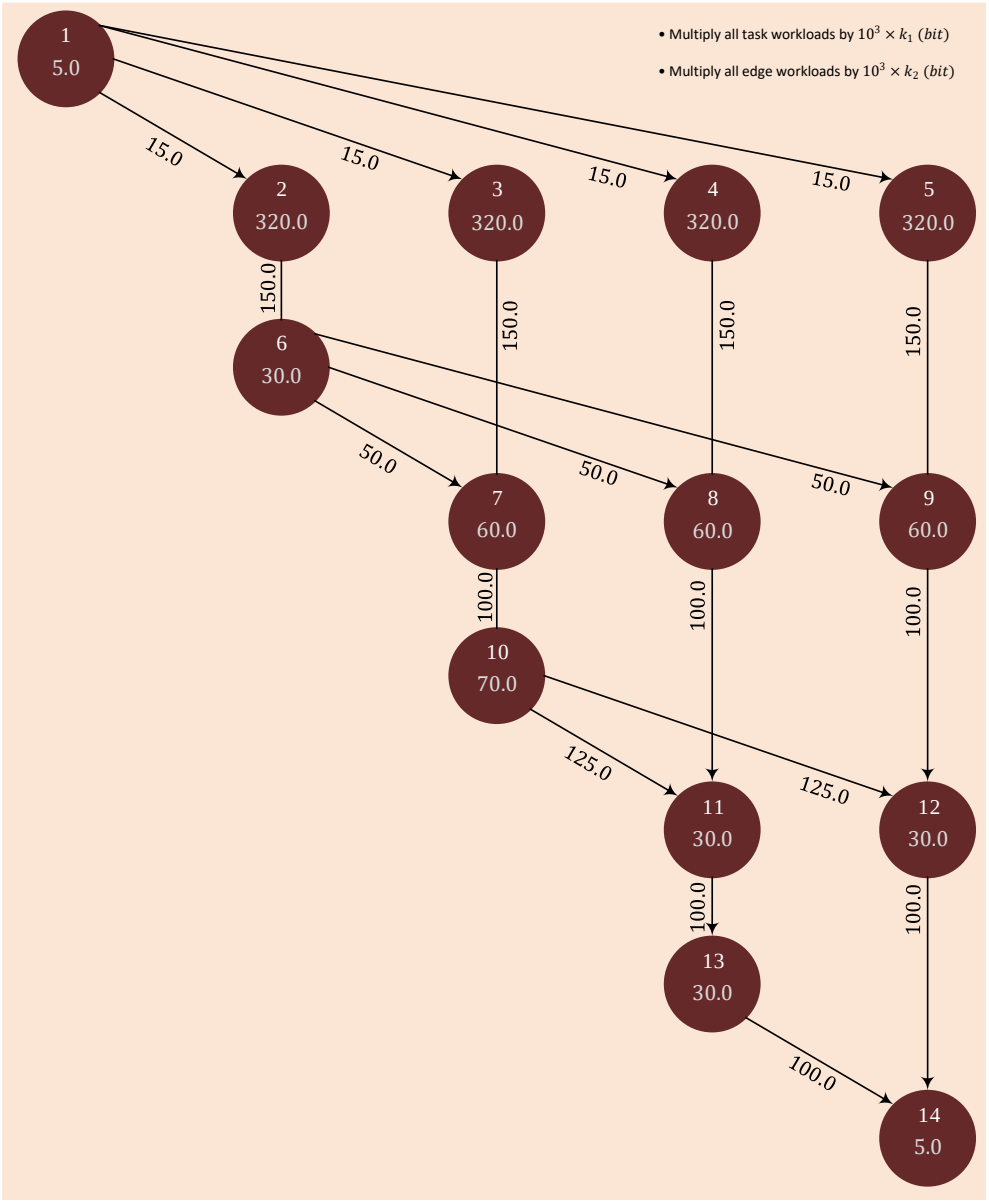

FIGURE 17. DAG8

$$D_a = k_2 \times c_2 \times \begin{bmatrix} 0 & 15 & 15 & 15 & 15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 150 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 150 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 150 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 150 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 50 & 50 & 50 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 125 & 125 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} (bit) .$$

6.8 DAG9

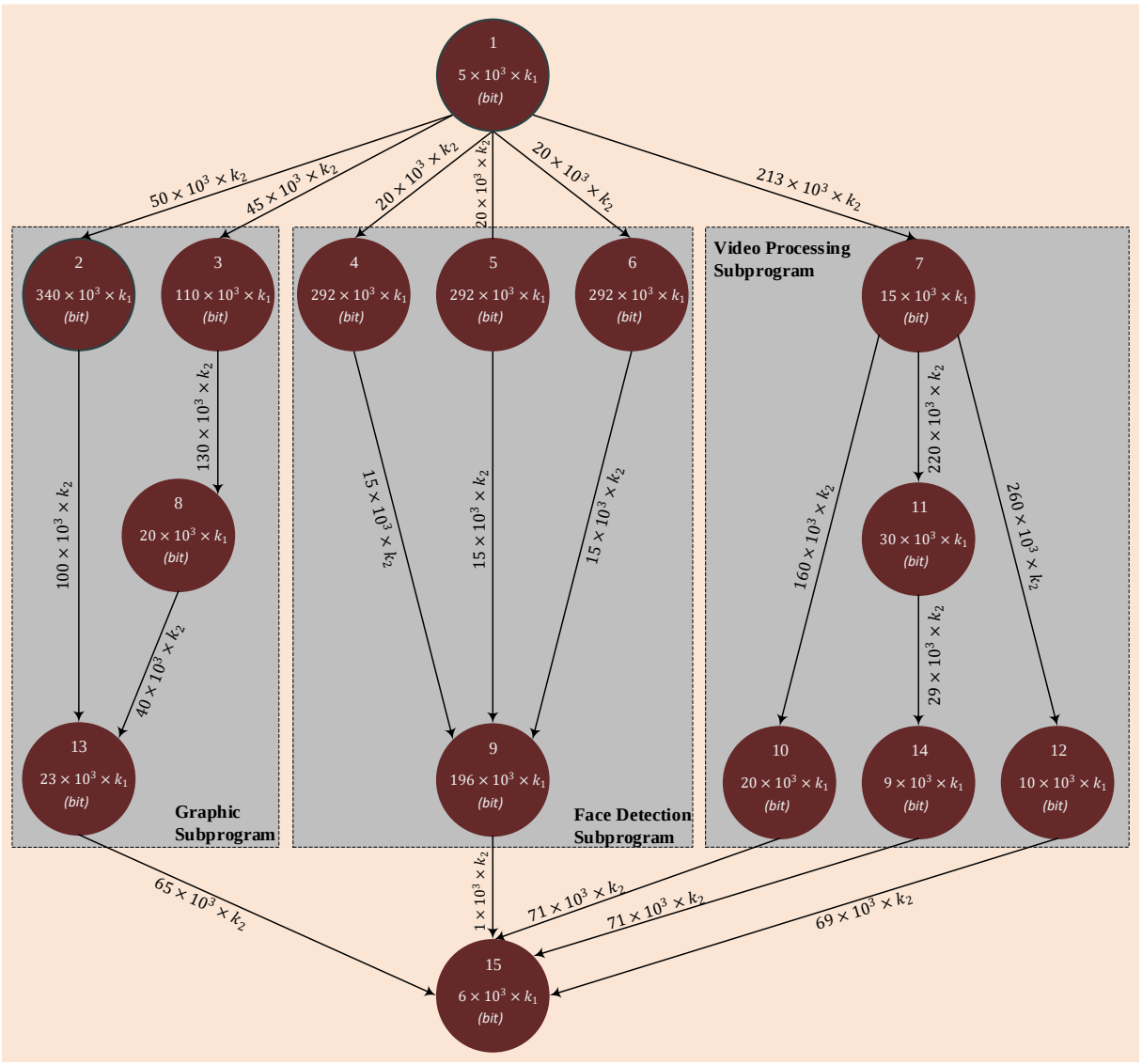

FIGURE 18. DAG9

$$\vec{s} = k_1 \times c_1 \times [5, 340, 110, 292, 292, 292, 15, 20, 196, 20, 30, 10, 23, 9, 6]\ (bit)\ ,$$

$$A = \begin{bmatrix}
0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix},$$

$$D_a = k_2 \times c_2 \times \begin{bmatrix}
0 & 50 & 45 & 20 & 20 & 20 & 213 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 100 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 130 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 15 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 15 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 15 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 160 & 220 & 260 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 46 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 71 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 29 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 69 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 65 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 71 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} (bit).$$

6.9 DAG10

$$\vec{s} = k_1 \times c_1 \times [5, 10, 10, 380 \times \text{ones}(1,4), 10 \times \text{ones}(1,4), 10 \times \text{ones}(1,4), 15, 15, 5]\,(bit),$$

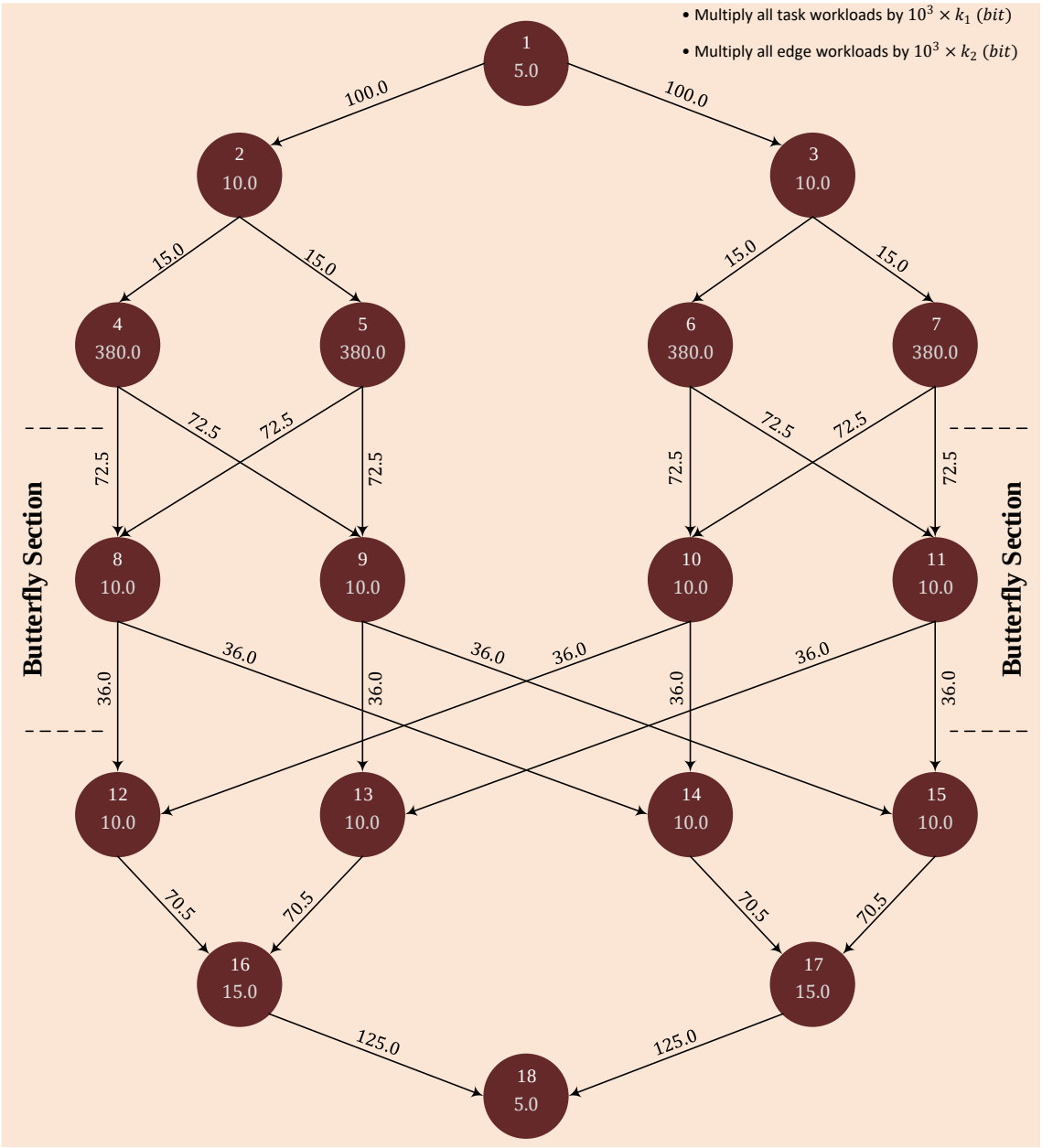

FIGURE 19. DAG10

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & \text{zeros}(1,14) \\ \text{zeros}(1,3) & 1 & 1 & 0 & \text{zeros}(1,12) \\ \text{zeros}(1,5) & 1 & 1 & 0 & \text{zeros}(1,10) \\ \text{zeros}(1,7) & 1 & 1 & 0 & \text{zeros}(1,8) \\ \text{zeros}(1,7) & 1 & 1 & 0 & \text{zeros}(1,8) \\ \text{zeros}(1,9) & 1 & 1 & 0 & \text{zeros}(1,6) \\ \text{zeros}(1,9) & 1 & 1 & 0 & \text{zeros}(1,6) \\ \text{zeros}(1,11) & 1 & 0 & 1 & \text{zeros}(1,4) \\ \text{zeros}(1,12) & 1 & 0 & 1 & \text{zeros}(1,3) \\ \text{zeros}(1,11) & 1 & 0 & 1 & \text{zeros}(1,4) \\ \text{zeros}(1,12) & 1 & 0 & 1 & \text{zeros}(1,3) \\ \text{zeros}(1,14) & 0 & 1 & 0 & 0 \\ \text{zeros}(1,14) & 0 & 1 & 0 & 0 \\ \text{zeros}(1,14) & 0 & 0 & 1 & 0 \\ \text{zeros}(1,14) & 0 & 0 & 1 & 0 \\ \text{zeros}(1,14) & 0 & 0 & 0 & 1 \\ \text{zeros}(1,14) & 0 & 0 & 0 & 1 \\ \text{zeros}(1,14) & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$D_a = k_2 \times c_2 \times \begin{bmatrix} 0 & 100 & 100 & 0 & \text{zeros}(1,14) \\ \text{zeros}(1,3) & 15 & 15 & 0 & \text{zeros}(1,12) \\ \text{zeros}(1,5) & 15 & 15 & 0 & \text{zeros}(1,10) \\ \text{zeros}(1,7) & 72.5 & 72.5 & 0 & \text{zeros}(1,8) \\ \text{zeros}(1,7) & 72.5 & 72.5 & 0 & \text{zeros}(1,8) \\ \text{zeros}(1,9) & 72.5 & 72.5 & 0 & \text{zeros}(1,6) \\ \text{zeros}(1,9) & 72.5 & 72.5 & 0 & \text{zeros}(1,6) \\ \text{zeros}(1,11) & 36 & 0 & 36 & \text{zeros}(1,4) \\ \text{zeros}(1,12) & 36 & 0 & 36 & \text{zeros}(1,3) \\ \text{zeros}(1,11) & 36 & 0 & 36 & \text{zeros}(1,4) \\ \text{zeros}(1,12) & 36 & 0 & 36 & \text{zeros}(1,3) \\ \text{zeros}(1,14) & 0 & 70.5 & 0 & 0 \\ \text{zeros}(1,14) & 0 & 70.5 & 0 & 0 \\ \text{zeros}(1,14) & 0 & 0 & 70.5 & 0 \\ \text{zeros}(1,14) & 0 & 0 & 70.5 & 0 \\ \text{zeros}(1,14) & 0 & 0 & 0 & 125 \\ \text{zeros}(1,14) & 0 & 0 & 0 & 125 \\ \text{zeros}(1,14) & 0 & 0 & 0 & 0 \end{bmatrix} (bit) .$$

6.10 DAG11

$\vec{s} = k_1 \times c_1 \times [26, 50, 350, 30 \times \text{ones}(1,4), 130 \times \text{ones}(1,4), 40 \times \text{ones}(1,4), 4, 40 \times \text{ones}(1,4), 4, 40 \times \text{ones}(1,4), 30 \times \text{ones}(1,2), 92 \times \text{ones}(1,2), 26]$,

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & \text{zeros}(1,21) \\ 0 & 0 & 0 & \text{ones}(1,4) & \text{zeros}(1,18) \\ \text{zeros}(1,5) & 0 & 0 & \text{ones}(1,4) & \text{zeros}(1,14) \\ \text{zeros}(1,11) & 1 & \text{zeros}(1,3) & 1 & \text{zeros}(1,9) \\ \text{zeros}(1,12) & 1 & \text{zeros}(1,2) & 1 & \text{zeros}(1,9) \\ \text{zeros}(1,13) & 1 & 0 & 1 & \text{zeros}(1,9) \\ \text{zeros}(1,14) & 1 & 1 & 0 & \text{zeros}(1,8) \\ \text{zeros}(1,15) & 1 & 1 & 0 & \text{zeros}(1,7) \\ \text{zeros}(1,15) & 1 & 0 & 1 & \text{zeros}(1,7) \\ \text{zeros}(1,15) & 1 & \text{zeros}(1,2) & 1 & \text{zeros}(1,6) \\ \text{zeros}(1,15) & 1 & \text{zeros}(1,2) & 1 & \text{zeros}(1,6) \\ \text{zeros}(1,15) & 1 & \text{zeros}(1,3) & 1 & \text{zeros}(1,5) \\ \text{zeros}(1,18) & 0 & 0 & 1 & \text{zeros}(1,4) \\ \text{zeros}(1,18) & 0 & 0 & 1 & \text{zeros}(1,4) \\ \text{zeros}(1,19) & 0 & 0 & 1 & \text{zeros}(1,3) \\ \text{zeros}(1,21) & 0 & 0 & 0 & 1 \\ \text{zeros}(1,21) & 0 & 1 & 0 & 0 \\ \text{zeros}(1,21) & 0 & 1 & 0 & 0 \\ \text{zeros}(1,21) & 0 & 0 & 1 & 0 \\ \text{zeros}(1,21) & 0 & 0 & 1 & 0 \\ \text{zeros}(1,21) & 0 & 0 & 0 & 1 \\ \text{zeros}(1,21) & 0 & 0 & 0 & 1 \\ \text{zeros}(1,21) & 0 & 0 & 0 & 1 \\ \text{zeros}(1,21) & 0 & 0 & 0 & 1 \\ \text{zeros}(1,21) & 0 & 0 & 0 & 0 \end{bmatrix} ,$$

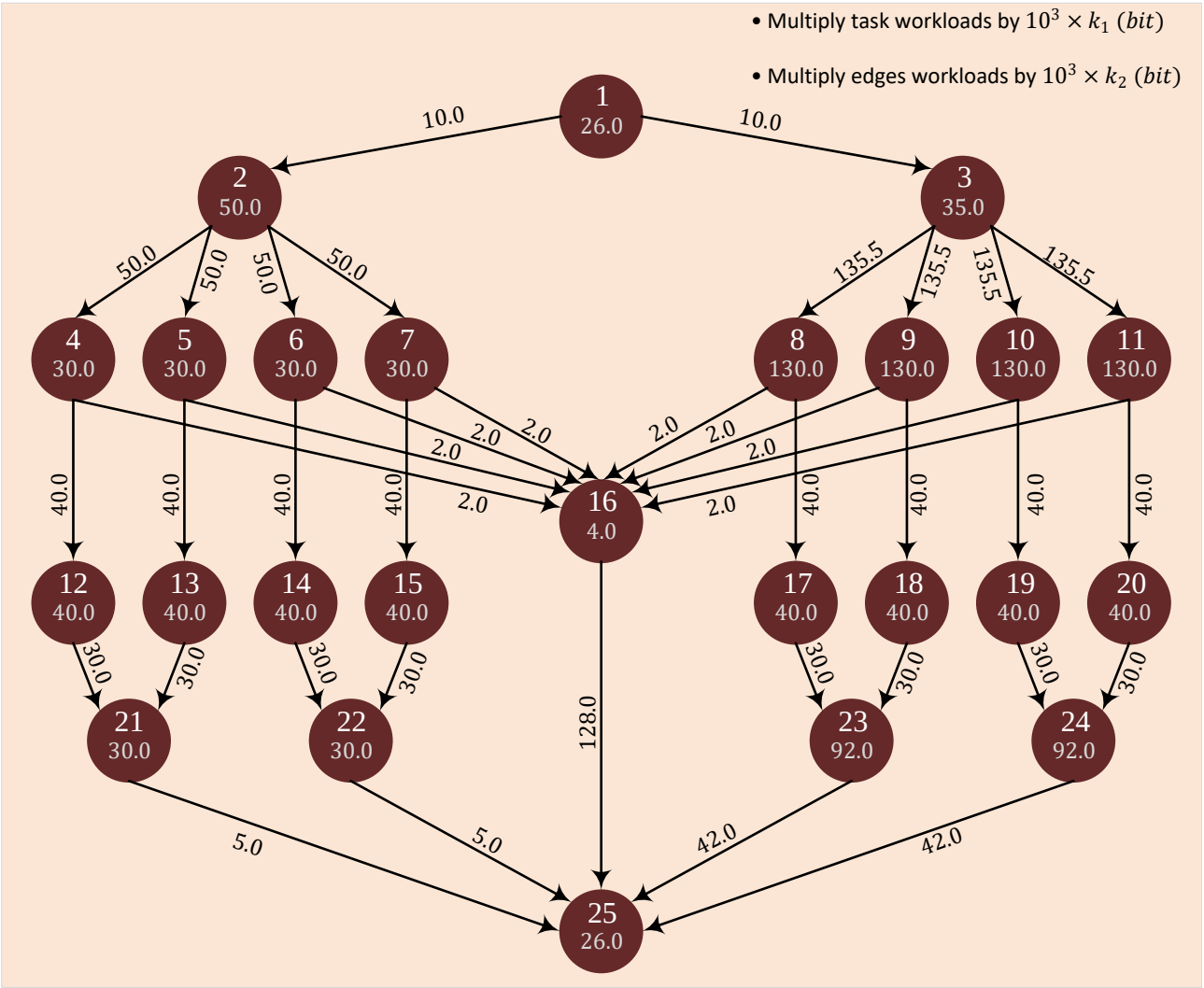

FIGURE 20. DAG11

$$D_a = k_2 \times c_2 \times \begin{bmatrix} 0 & 10 & 110 & 0 & \text{zeros}(1,21) \\ 0 & 0 & 0 & 50.5 \times \text{ones}(1,4) & \text{zeros}(1,18) \\ \text{zeros}(1,5) & 0 & 0 & 13.5 \times \text{ones}(1,4) & \text{zeros}(1,14) \\ \text{zeros}(1,11) & 40 & \text{zeros}(1,3) & 2 & \text{zeros}(1,9) \\ \text{zeros}(1,12) & 40 & \text{zeros}(1,2) & 2 & \text{zeros}(1,9) \\ \text{zeros}(1,13) & 40 & 0 & 2 & \text{zeros}(1,9) \\ \text{zeros}(1,14) & 40 & 2 & 0 & \text{zeros}(1,8) \\ \text{zeros}(1,15) & 2 & 40 & 0 & \text{zeros}(1,7) \\ \text{zeros}(1,15) & 2 & 0 & 40 & \text{zeros}(1,7) \\ \text{zeros}(1,15) & 2 & \text{zeros}(1,2) & 40 & \text{zeros}(1,6) \\ \text{zeros}(1,15) & 2 & \text{zeros}(1,3) & 40 & \text{zeros}(1,5) \\ \text{zeros}(1,19) & 0 & 30 & 0 & \text{zeros}(1,3) \\ \text{zeros}(1,19) & 0 & 30 & 0 & \text{zeros}(1,3) \\ \text{zeros}(1,20) & 0 & 30 & 0 & \text{zeros}(1,2) \\ \text{zeros}(1,20) & 0 & 30 & 0 & \text{zeros}(1,2) \\ \text{zeros}(1,21) & 0 & 0 & 0 & 128 \\ \text{zeros}(1,21) & 0 & 30 & 0 & 0 \\ \text{zeros}(1,21) & 0 & 30 & 0 & 0 \\ \text{zeros}(1,21) & 0 & 0 & 30 & 0 \\ \text{zeros}(1,21) & 0 & 0 & 30 & 0 \\ \text{zeros}(1,21) & 0 & 0 & 0 & 5 \\ \text{zeros}(1,21) & 0 & 0 & 0 & 5 \\ \text{zeros}(1,21) & 0 & 0 & 0 & 42 \\ \text{zeros}(1,21) & 0 & 0 & 0 & 42 \\ \text{zeros}(1,21) & 0 & 0 & 0 & 0 \end{bmatrix} (bit).$$

6.11 DAG12

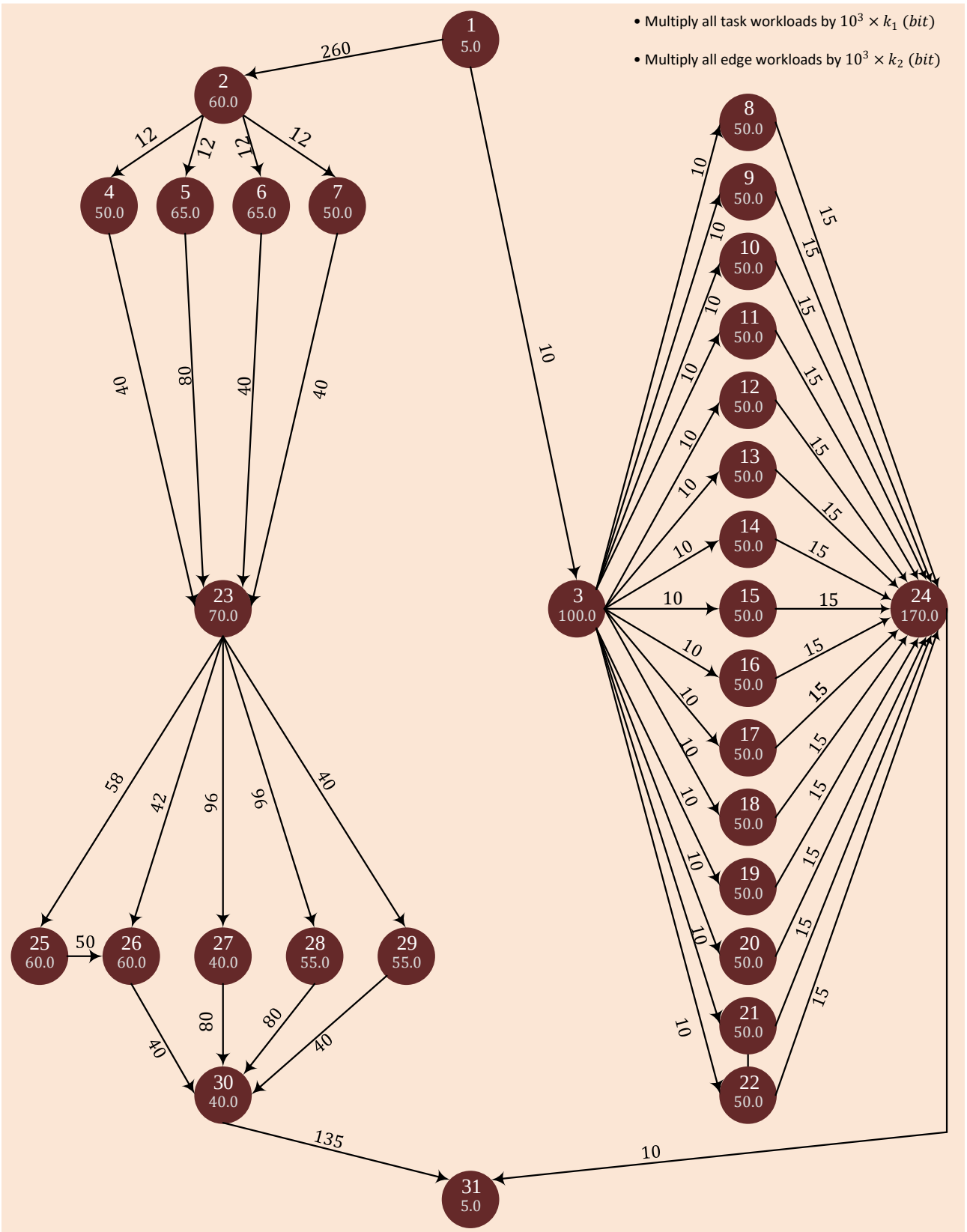

FIGURE 21. DAG12

$$\vec{s} = k_1 \times c_1 \times [5, 60, 100, 50, 65, 65, 50, 50 \times \text{ones}(1, 15), 70, 170, 60, 60, 40, 55, 55, 40, 5]\,,$$

$$A = \begin{bmatrix}
0 & 1 & 1 & 0 & 0 & 0 & \text{zeros}(1, 25) \\
\text{zeros}(1, 3) & 1 & 1 & 1 & 1 & 0 & \text{zeros}(1, 23) \\
\text{zeros}(1, 7) & 1 \times \text{ones}(1, 15) & 0 & 0 & 0 & 0 & \text{zeros}(1, 5) \\
\text{zeros}(1, 20) & 0 & 0 & 1 & 0 & 0 & \text{zeros}(1, 6) \\
\text{zeros}(1, 20) & 0 & 0 & 1 & 0 & 0 & \text{zeros}(1, 6) \\
\text{zeros}(1, 20) & 0 & 0 & 1 & 0 & 0 & \text{zeros}(1, 6) \\
\text{zeros}(1, 20) & 0 & 0 & 1 & 0 & 0 & \text{zeros}(1, 6) \\
\text{zeros}(1, 21) & 0 & 0 & 1 & 0 & 0 & \text{zeros}(1, 5) \\
\text{zeros}(1, 21) & 0 & 0 & 1 & 0 & 0 & \text{zeros}(1, 5) \\
\text{zeros}(1, 21) & 0 & 0 & 1 & 0 & 0 & \text{zeros}(1, 5) \\
\text{zeros}(1, 21) & 0 & 0 & 1 & 0 & 0 & \text{zeros}(1, 5) \\
\text{zeros}(1, 21) & 0 & 0 & 1 & 0 & 0 & \text{zeros}(1, 5) \\
\text{zeros}(1, 21) & 0 & 0 & 1 & 0 & 0 & \text{zeros}(1, 5) \\
\text{zeros}(1, 21) & 0 & 0 & 1 & 0 & 0 & \text{zeros}(1, 5) \\
\text{zeros}(1, 21) & 0 & 0 & 1 & 0 & 0 & \text{zeros}(1, 5) \\
\text{zeros}(1, 21) & 0 & 0 & 1 & 0 & 0 & \text{zeros}(1, 5) \\
\text{zeros}(1, 21) & 0 & 0 & 1 & 0 & 0 & \text{zeros}(1, 5) \\
\text{zeros}(1, 21) & 0 & 0 & 1 & 0 & 0 & \text{zeros}(1, 5) \\
\text{zeros}(1, 21) & 0 & 0 & 1 & 0 & 0 & \text{zeros}(1, 5) \\
\text{zeros}(1, 21) & 0 & 0 & 1 & 0 & 0 & \text{zeros}(1, 5) \\
\text{zeros}(1, 24) & 1 & 1 & 1 & 1 & 1 & \text{zeros}(1, 2) \\
\text{zeros}(1, 25) & 0 & 0 & 0 & 0 & 0 & 1 \\
\text{zeros}(1, 23) & 0 & 0 & 1 & 0 & 0 & \text{zeros}(1, 3) \\
\text{zeros}(1, 25) & 0 & 0 & 0 & 0 & 1 & 0 \\
\text{zeros}(1, 25) & 0 & 0 & 0 & 0 & 1 & 0 \\
\text{zeros}(1, 25) & 0 & 0 & 0 & 0 & 1 & 0 \\
\text{zeros}(1, 25) & 0 & 0 & 0 & 0 & 1 & 0 \\
\text{zeros}(1, 25) & 0 & 0 & 0 & 0 & 0 & 1 \\
\text{zeros}(1, 26) & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix},$$

$$D_a = k_2 \times c_2 \times \begin{bmatrix}
0 & 260 & 10 & 0 & 0 & 0 & \text{zeros}(1, 25) \\
\text{zeros}(1, 3) & 12 & 12 & 12 & 12 & 0 & \text{zeros}(1, 23) \\
\text{zeros}(1, 7) & 10 \times \text{ones}(1, 15) & 0 & 0 & 0 & 0 & \text{zeros}(1, 5) \\
\text{zeros}(1, 20) & 0 & 0 & 40 & 0 & 0 & \text{zeros}(1, 6) \\
\text{zeros}(1, 20) & 0 & 0 & 80 & 0 & 0 & \text{zeros}(1, 6) \\
\text{zeros}(1, 20) & 0 & 0 & 40 & 0 & 0 & \text{zeros}(1, 6) \\
\text{zeros}(1, 20) & 0 & 0 & 40 & 0 & 0 & \text{zeros}(1, 6) \\
\text{zeros}(1, 21) & 0 & 0 & 15 & 0 & 0 & \text{zeros}(1, 5) \\
\text{zeros}(1, 21) & 0 & 0 & 15 & 0 & 0 & \text{zeros}(1, 5) \\
\text{zeros}(1, 21) & 0 & 0 & 15 & 0 & 0 & \text{zeros}(1, 5) \\
\text{zeros}(1, 21) & 0 & 0 & 15 & 0 & 0 & \text{zeros}(1, 5) \\
\text{zeros}(1, 21) & 0 & 0 & 15 & 0 & 0 & \text{zeros}(1, 5) \\
\text{zeros}(1, 21) & 0 & 0 & 15 & 0 & 0 & \text{zeros}(1, 5) \\
\text{zeros}(1, 21) & 0 & 0 & 15 & 0 & 0 & \text{zeros}(1, 5) \\
\text{zeros}(1, 21) & 0 & 0 & 15 & 0 & 0 & \text{zeros}(1, 5) \\
\text{zeros}(1, 21) & 0 & 0 & 15 & 0 & 0 & \text{zeros}(1, 5) \\
\text{zeros}(1, 21) & 0 & 0 & 15 & 0 & 0 & \text{zeros}(1, 5) \\
\text{zeros}(1, 21) & 0 & 0 & 15 & 0 & 0 & \text{zeros}(1, 5) \\
\text{zeros}(1, 21) & 0 & 0 & 15 & 0 & 0 & \text{zeros}(1, 5) \\
\text{zeros}(1, 21) & 0 & 0 & 15 & 0 & 0 & \text{zeros}(1, 5) \\
\text{zeros}(1, 24) & 58 & 42 & 96 & 96 & 40 & \text{zeros}(1, 2) \\
\text{zeros}(1, 25) & 0 & 0 & 0 & 0 & 0 & 10 \\
\text{zeros}(1, 23) & 0 & 0 & 50 & 0 & 0 & \text{zeros}(1, 3) \\
\text{zeros}(1, 25) & 0 & 0 & 0 & 0 & 40 & 0 \\
\text{zeros}(1, 25) & 0 & 0 & 0 & 0 & 80 & 0 \\
\text{zeros}(1, 25) & 0 & 0 & 0 & 0 & 80 & 0 \\
\text{zeros}(1, 25) & 0 & 0 & 0 & 0 & 40 & 0 \\
\text{zeros}(1, 25) & 0 & 0 & 0 & 0 & 0 & 135 \\
\text{zeros}(1, 26) & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} (bit)\,.$$

6.12 DAG13

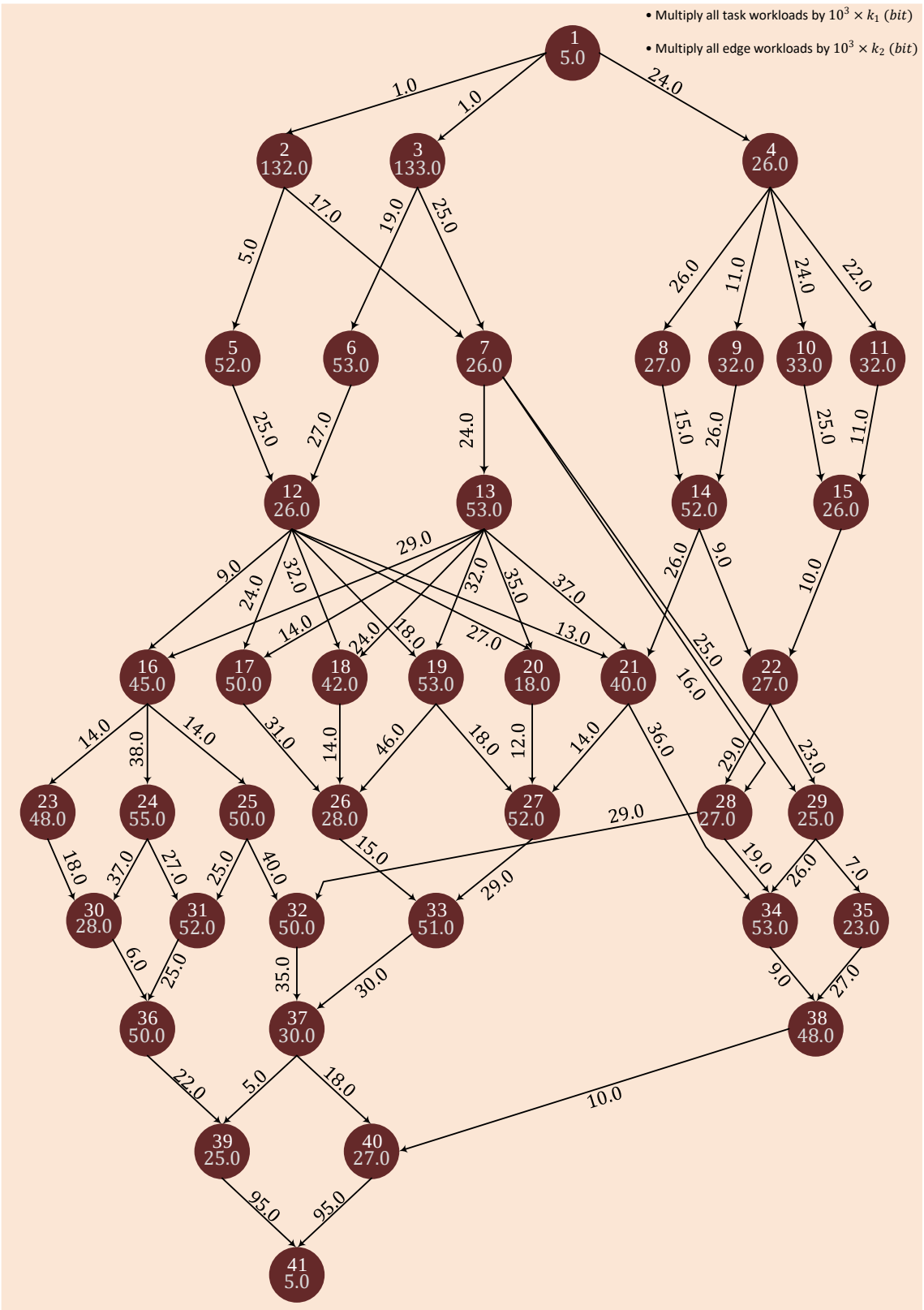

FIGURE 22. DAG13

$$\vec{s} = k_1 \times c_1 \times [5, 132, 133, 26, 52, 53, 26, 27, 32, 33, 32, 26, 53, 52, 26, 45, 50, 42, 53, 18, 40, 27, 48, 55, 50, 28, 52, 27, 25, 28, 52, 50, 51, 53, 23, 50, 30, 48, 25, 27, 5],$$

$$A = \begin{bmatrix}
0 & 1 & 1 & 1 & 0 & 0 & 0 & \text{zeros}(1,34) \\
\text{zeros}(1,4) & 1 & 0 & 1 & 0 & 0 & 0 & \text{zeros}(1,31) \\
\text{zeros}(1,5) & 1 & 1 & 0 & 0 & 0 & 0 & \text{zeros}(1,30) \\
\text{zeros}(1,7) & 1 & 1 & 1 & 1 & 0 & 0 & \text{zeros}(1,28) \\
\text{zeros}(1,11) & 1 & 0 & 0 & 0 & 0 & 0 & \text{zeros}(1,24) \\
\text{zeros}(1,11) & 1 & 0 & 0 & 0 & 0 & 0 & \text{zeros}(1,24) \\
\text{zeros}(1,12) & 1 & 0 & 0 & \text{zeros}(1,12) & 1 & 1 & \text{zeros}(1,12) \\
\text{zeros}(1,13) & 1 & 0 & 0 & 0 & 0 & 0 & \text{zeros}(1,22) \\
\text{zeros}(1,13) & 1 & 0 & 0 & 0 & 0 & 0 & \text{zeros}(1,22) \\
\text{zeros}(1,14) & 1 & 0 & 0 & 0 & 0 & 0 & \text{zeros}(1,21) \\
\text{zeros}(1,14) & 1 & 0 & 0 & 0 & 0 & 0 & \text{zeros}(1,21) \\
\text{zeros}(1,15) & 1 & 1 & 1 & 1 & 1 & 1 & \text{zeros}(1,20) \\
\text{zeros}(1,15) & 1 & 1 & 1 & 1 & 1 & 1 & \text{zeros}(1,20) \\
\text{zeros}(1,18) & 0 & 0 & 1 & 1 & 0 & 0 & \text{zeros}(1,17) \\
\text{zeros}(1,19) & 0 & 0 & 1 & 0 & 0 & 0 & \text{zeros}(1,16) \\
\text{zeros}(1,20) & 0 & 0 & 1 & 1 & 1 & 0 & \text{zeros}(1,15) \\
\text{zeros}(1,23) & 0 & 0 & 1 & 0 & 0 & 0 & \text{zeros}(1,12) \\
\text{zeros}(1,23) & 0 & 0 & 1 & 0 & 0 & 0 & \text{zeros}(1,12) \\
\text{zeros}(1,23) & 0 & 0 & 1 & 1 & 0 & 0 & \text{zeros}(1,12) \\
\text{zeros}(1,24) & 0 & 0 & 1 & 0 & 0 & 0 & \text{zeros}(1,11) \\
\text{zeros}(1,26) & 1 & 0 & 0 & \text{zeros}(1,4) & 1 & 0 & \text{zeros}(1,6) \\
\text{zeros}(1,25) & 0 & 0 & 1 & 1 & 0 & 0 & \text{zeros}(1,10) \\
\text{zeros}(1,27) & 0 & 0 & 1 & 0 & 0 & 0 & \text{zeros}(1,8) \\
\text{zeros}(1,27) & 0 & 0 & 1 & 1 & 0 & 0 & \text{zeros}(1,8) \\
\text{zeros}(1,28) & 0 & 0 & 1 & 1 & 0 & 0 & \text{zeros}(1,7) \\
\text{zeros}(1,30) & 0 & 0 & 1 & 0 & 0 & 0 & \text{zeros}(1,5) \\
\text{zeros}(1,30) & 0 & 0 & 1 & 0 & 0 & 0 & \text{zeros}(1,5) \\
\text{zeros}(1,31) & 1 & 0 & 1 & 0 & 0 & 0 & \text{zeros}(1,4) \\
\text{zeros}(1,31) & 0 & 0 & 1 & 1 & 0 & 0 & \text{zeros}(1,4) \\
\text{zeros}(1,33) & 0 & 0 & 1 & 0 & 0 & 0 & \text{zeros}(1,2) \\
\text{zeros}(1,33) & 0 & 0 & 1 & 0 & 0 & 0 & \text{zeros}(1,2) \\
\text{zeros}(1,34) & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
\text{zeros}(1,34) & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
\text{zeros}(1,32) & 0 & 0 & 0 & 0 & 0 & 1 & \text{zeros}(1,3) \\
\text{zeros}(1,32) & 0 & 0 & 0 & 0 & 0 & 1 & \text{zeros}(1,3) \\
\text{zeros}(1,34) & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
\text{zeros}(1,34) & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
\text{zeros}(1,34) & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
\text{zeros}(1,34) & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\text{zeros}(1,34) & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\text{zeros}(1,34) & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix},$$

$$D_a = k_2 \times c_2 \times \begin{bmatrix}
0 & 1 & 1 & 24 & 0 & 0 & 0 & \text{zeros}(1,34) \\
\text{zeros}(1,4) & 5 & 0 & 17 & 0 & 0 & 0 & \text{zeros}(1,31) \\
\text{zeros}(1,5) & 19 & 25 & 0 & 0 & 0 & 0 & \text{zeros}(1,30) \\
\text{zeros}(1,7) & 26 & 11 & 24 & 22 & 0 & 0 & \text{zeros}(1,28) \\
\text{zeros}(1,11) & 25 & 0 & 0 & 0 & 0 & 0 & \text{zeros}(1,24) \\
\text{zeros}(1,11) & 27 & 0 & 0 & 0 & 0 & 0 & \text{zeros}(1,24) \\
\text{zeros}(1,12) & 24 & 0 & 0 & \text{zeros}(1,12) & 16 & 25 & \text{zeros}(1,12) \\
\text{zeros}(1,13) & 15 & 0 & 0 & 0 & 0 & 0 & \text{zeros}(1,22) \\
\text{zeros}(1,13) & 26 & 0 & 0 & 0 & 0 & 0 & \text{zeros}(1,22) \\
\text{zeros}(1,14) & 25 & 0 & 0 & 0 & 0 & 0 & \text{zeros}(1,21) \\
\text{zeros}(1,14) & 11 & 0 & 0 & 0 & 0 & 0 & \text{zeros}(1,21) \\
\text{zeros}(1,15) & 9 & 24 & 32 & 18 & 27 & 13 & \text{zeros}(1,20) \\
\text{zeros}(1,15) & 29 & 14 & 24 & 32 & 35 & 37 & \text{zeros}(1,20) \\
\text{zeros}(1,18) & 0 & 0 & 26 & 9 & 0 & 0 & \text{zeros}(1,17) \\
\text{zeros}(1,19) & 0 & 0 & 10 & 0 & 0 & 0 & \text{zeros}(1,16) \\
\text{zeros}(1,20) & 0 & 0 & 14 & 38 & 14 & 0 & \text{zeros}(1,15) \\
\text{zeros}(1,23) & 0 & 0 & 31 & 0 & 0 & 0 & \text{zeros}(1,12) \\
\text{zeros}(1,23) & 0 & 0 & 14 & 0 & 0 & 0 & \text{zeros}(1,12) \\
\text{zeros}(1,23) & 0 & 0 & 46 & 18 & 0 & 0 & \text{zeros}(1,12) \\
\text{zeros}(1,24) & 0 & 0 & 12 & 0 & 0 & 0 & \text{zeros}(1,11) \\
\text{zeros}(1,26) & 14 & 0 & 0 & \text{zeros}(1,4) & 36 & 0 & \text{zeros}(1,6) \\
\text{zeros}(1,25) & 0 & 0 & 29 & 23 & 0 & 0 & \text{zeros}(1,10) \\
\text{zeros}(1,27) & 0 & 0 & 18 & 0 & 0 & 0 & \text{zeros}(1,8) \\
\text{zeros}(1,27) & 0 & 0 & 37 & 27 & 0 & 0 & \text{zeros}(1,8) \\
\text{zeros}(1,28) & 0 & 0 & 25 & 40 & 0 & 0 & \text{zeros}(1,7) \\
\text{zeros}(1,30) & 0 & 0 & 15 & 0 & 0 & 0 & \text{zeros}(1,5) \\
\text{zeros}(1,30) & 0 & 0 & 29 & 0 & 0 & 0 & \text{zeros}(1,5) \\
\text{zeros}(1,31) & 29 & 0 & 19 & 0 & 0 & 0 & \text{zeros}(1,4) \\
\text{zeros}(1,31) & 0 & 0 & 26 & 7 & 0 & 0 & \text{zeros}(1,4) \\
\text{zeros}(1,33) & 0 & 0 & 6 & 0 & 0 & 0 & \text{zeros}(1,2) \\
\text{zeros}(1,33) & 0 & 0 & 25 & 0 & 0 & 0 & \text{zeros}(1,2) \\
\text{zeros}(1,34) & 0 & 0 & 35 & 0 & 0 & 0 & 0 \\
\text{zeros}(1,34) & 0 & 0 & 30 & 0 & 0 & 0 & 0 \\
\text{zeros}(1,32) & 0 & 0 & 0 & 0 & 0 & 9 & \text{zeros}(1,3) \\
\text{zeros}(1,32) & 0 & 0 & 0 & 0 & 0 & 27 & \text{zeros}(1,3) \\
\text{zeros}(1,34) & 0 & 0 & 0 & 0 & 22 & 0 & 0 \\
\text{zeros}(1,34) & 0 & 0 & 0 & 0 & 5 & 18 & 0 \\
\text{zeros}(1,34) & 0 & 0 & 0 & 0 & 0 & 10 & 0 \\
\text{zeros}(1,34) & 0 & 0 & 0 & 0 & 0 & 0 & 95 \\
\text{zeros}(1,34) & 0 & 0 & 0 & 0 & 0 & 0 & 95 \\
\text{zeros}(1,34) & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} (bit) .$$

6.13 DAG14

$$\vec{s} = k_1 \times c_1 \times [30, 34 \times \text{ones}(1,10), 25 \times \text{ones}(1,10), 35 \times \text{ones}(1,10), 33 \times \text{ones}(1,10), 20 \times \text{ones}(1,2), 318, 2] ,$$

$$
A = \begin{bmatrix}
0 & 1 \times \text{ones}(1,10) & \text{zeros}(1,34) \\
\text{zeros}(1,11) & 1 & \text{zeros}(1,33) \\
\text{zeros}(1,12) & 1 & \text{zeros}(1,32) \\
\text{zeros}(1,13) & 1 & \text{zeros}(1,31) \\
\text{zeros}(1,14) & 1 & \text{zeros}(1,30) \\
\text{zeros}(1,15) & 1 & \text{zeros}(1,29) \\
\text{zeros}(1,16) & 1 & \text{zeros}(1,28) \\
\text{zeros}(1,17) & 1 & \text{zeros}(1,27) \\
\text{zeros}(1,18) & 1 & \text{zeros}(1,26) \\
\text{zeros}(1,19) & 1 & \text{zeros}(1,25) \\
\text{zeros}(1,20) & 1 & \text{zeros}(1,24) \\
\text{zeros}(1,21) & 1 & \text{zeros}(1,23) \\
\text{zeros}(1,22) & 1 & \text{zeros}(1,22) \\
\text{zeros}(1,23) & 1 & \text{zeros}(1,21) \\
\text{zeros}(1,24) & 1 & \text{zeros}(1,20) \\
\text{zeros}(1,25) & 1 & \text{zeros}(1,19) \\
\text{zeros}(1,26) & 1 & \text{zeros}(1,18) \\
\text{zeros}(1,27) & 1 & \text{zeros}(1,17) \\
\text{zeros}(1,28) & 1 & \text{zeros}(1,16) \\
\text{zeros}(1,29) & 1 & \text{zeros}(1,15) \\
\text{zeros}(1,30) & 1 & \text{zeros}(1,14) \\
\text{zeros}(1,31) & 1 & \text{zeros}(1,13) \\
\text{zeros}(1,32) & 1 & \text{zeros}(1,12) \\
\text{zeros}(1,33) & 1 & \text{zeros}(1,11) \\
\text{zeros}(1,34) & 1 & \text{zeros}(1,10) \\
\text{zeros}(1,35) & 1 & \text{zeros}(1,9) \\
\text{zeros}(1,36) & 1 & \text{zeros}(1,8) \\
\text{zeros}(1,37) & 1 & \text{zeros}(1,7) \\
\text{zeros}(1,38) & 1 & \text{zeros}(1,6) \\
\text{zeros}(1,39) & 1 & \text{zeros}(1,5) \\
\text{zeros}(1,40) & 1 & \text{zeros}(1,4) \\
\text{zeros}(1,41) & 1 & \text{zeros}(1,3) \\
\text{zeros}(1,41) & 1 & \text{zeros}(1,3) \\
\text{zeros}(1,41) & 1 & \text{zeros}(1,3) \\
\text{zeros}(1,41) & 1 & \text{zeros}(1,3) \\
\text{zeros}(1,41) & 1 & \text{zeros}(1,3) \\
\text{zeros}(1,42) & 1 & \text{zeros}(1,2) \\
\text{zeros}(1,42) & 1 & \text{zeros}(1,2) \\
\text{zeros}(1,42) & 1 & \text{zeros}(1,2) \\
\text{zeros}(1,42) & 1 & \text{zeros}(1,2) \\
\text{zeros}(1,42) & 1 & \text{zeros}(1,2) \\
\text{zeros}(1,43) & 1 & 0 \\
\text{zeros}(1,43) & 1 & 0 \\
\text{zeros}(1,43) & 0 & 1 \\
\text{zeros}(1,43) & 0 & 0
\end{bmatrix},
$$

$$D_a = k_2 \times c_2 \times \begin{bmatrix} 0 & 17 \times \text{ones}(1,10) & \text{zeros}(1,34) \\ \text{zeros}(1,11) & 47 & \text{zeros}(1,33) \\ \text{zeros}(1,12) & 47 & \text{zeros}(1,32) \\ \text{zeros}(1,13) & 47 & \text{zeros}(1,31) \\ \text{zeros}(1,14) & 47 & \text{zeros}(1,30) \\ \text{zeros}(1,15) & 47 & \text{zeros}(1,29) \\ \text{zeros}(1,16) & 47 & \text{zeros}(1,28) \\ \text{zeros}(1,17) & 47 & \text{zeros}(1,27) \\ \text{zeros}(1,18) & 47 & \text{zeros}(1,26) \\ \text{zeros}(1,19) & 47 & \text{zeros}(1,25) \\ \text{zeros}(1,20) & 47 & \text{zeros}(1,24) \\ \text{zeros}(1,21) & 4.4 & \text{zeros}(1,23) \\ \text{zeros}(1,22) & 4.4 & \text{zeros}(1,22) \\ \text{zeros}(1,23) & 4.4 & \text{zeros}(1,21) \\ \text{zeros}(1,24) & 4.4 & \text{zeros}(1,20) \\ \text{zeros}(1,25) & 4.4 & \text{zeros}(1,19) \\ \text{zeros}(1,26) & 4.4 & \text{zeros}(1,18) \\ \text{zeros}(1,27) & 4.4 & \text{zeros}(1,17) \\ \text{zeros}(1,28) & 4.4 & \text{zeros}(1,16) \\ \text{zeros}(1,29) & 4.4 & \text{zeros}(1,15) \\ \text{zeros}(1,30) & 4.4 & \text{zeros}(1,14) \\ \text{zeros}(1,31) & 91 & \text{zeros}(1,13) \\ \text{zeros}(1,32) & 91 & \text{zeros}(1,12) \\ \text{zeros}(1,33) & 91 & \text{zeros}(1,11) \\ \text{zeros}(1,34) & 91 & \text{zeros}(1,10) \\ \text{zeros}(1,35) & 91 & \text{zeros}(1,9) \\ \text{zeros}(1,36) & 91 & \text{zeros}(1,8) \\ \text{zeros}(1,37) & 91 & \text{zeros}(1,7) \\ \text{zeros}(1,38) & 91 & \text{zeros}(1,6) \\ \text{zeros}(1,39) & 91 & \text{zeros}(1,5) \\ \text{zeros}(1,40) & 91 & \text{zeros}(1,4) \\ \text{zeros}(1,41) & 0.5 & \text{zeros}(1,3) \\ \text{zeros}(1,41) & 0.5 & \text{zeros}(1,3) \\ \text{zeros}(1,41) & 0.5 & \text{zeros}(1,3) \\ \text{zeros}(1,41) & 0.5 & \text{zeros}(1,3) \\ \text{zeros}(1,41) & 0.5 & \text{zeros}(1,3) \\ \text{zeros}(1,42) & 0.5 & \text{zeros}(1,2) \\ \text{zeros}(1,42) & 0.5 & \text{zeros}(1,2) \\ \text{zeros}(1,42) & 0.5 & \text{zeros}(1,2) \\ \text{zeros}(1,42) & 0.5 & \text{zeros}(1,2) \\ \text{zeros}(1,42) & 0.5 & \text{zeros}(1,2) \\ \text{zeros}(1,43) & 6 & 0 \\ \text{zeros}(1,43) & 6 & 0 \\ \text{zeros}(1,43) & 0 & 49 \\ \text{zeros}(1,43) & 0 & 0 \end{bmatrix} (bit) .$$

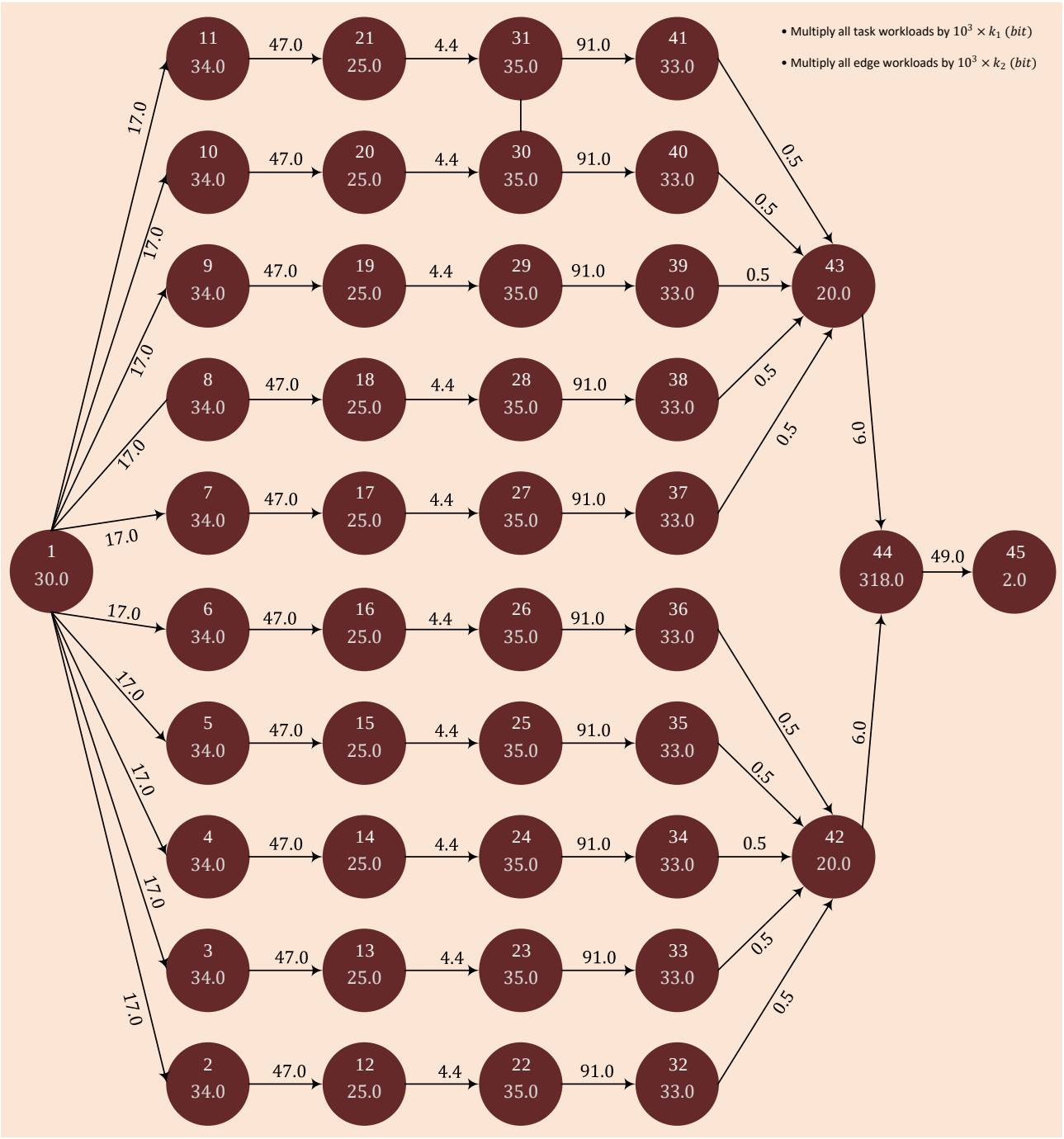

FIGURE 23. DAG14

7 AVAILABILITY OF THE VIRTFOGSIM SOFTWARE PACKAGE

The complete software package of the *VirtFogSim* simulator and the corresponding full *User Guide* may be downloaded *for free* at:

http://enzobaccarelli.site.uniroma1.it

by accessing the section: DOWNLOADABLE PACKAGES.

AKNOWLEDGEMENTS

This work has been partially supported by the PRIN2015 project no. 2015YPXH4W_004: "A green adaptive Fog computing and networking architecture (GAUChO)", funded by the Italian MIUR. It has been developed under the research activity planned for the fourth work-package (namely WP4) of the GAUChO project (see the project web site at: www.gaucho.unifi.it).

References

[1] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: A simple model and its empirical validation," in *Proceedings of the ACM SIGCOMM '98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, SIGCOMM '98, (New York, NY, USA), pp. 303–314, ACM, 1998.

[2] N. Vallina-Rodriguez and J. Crowcroft, "Energy management techniques in modern mobile handsets," *IEEE Commun. Surv. Tutor.*, vol. 15, pp. 179–198, First Quarter 2013.

[3] M. Altamimi, A. Abdrabou, K. Naik, and A. Nayak, "Energy cost models of smartphones for task offloading to the cloud," *IEEE T. Emerg. Top. Com.*, vol. 3, pp. 384–398, September 2015.

[4] J. Kwak, O. Choi, S. Chong, and P. Mohapatra, "Processor-network speed scaling for energy: delay tradeoff in smartphone applications," *IEEE/ACM Trans. Netw.*, vol. 24, pp. 1647–1660, June 2016.

[5] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *2010 IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, SIGCOMM '98, (Scottsdale, USA), pp. 105–114, IEEE, October 2010.

[6] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A close examination of performance and power characteristics of 4G LTE networks," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*, MobiSys '12, (Lake District, UK), pp. 225–238, ACM, 2012.

[7] Y. Xiao, Y. Cui, P. Savolainen, M. Siekkinen, A. Wang, L. Yang, A. Ylä-Jääski, and S. Tarkoma, "Modeling energy consumption of data transmission over Wi-Fi," *IEEE Trans. Mob. Comput.*, vol. 13, pp. 1760–1773, August 2014.

[8] Y.-s. Lim, Y.-C. Chen, E. M. Nahum, D. Towsley, and R. J. Gibbens, "Improving energy efficiency of MPTCP for mobile devices," *arXiv preprint arXiv:1406.4463*, 2014.

[9] A. Mukherjee, D. De, and D. G. Roy, "A power and latency aware cloudlet selection strategy for multi-cloudlet environment," *IEEE Trans. Cloud Comput.*, 2016.

[10] S. Yang, D. Kwon, H. Yi, Y. Cho, Y. Kwon, and Y. Paek, "Techniques to minimize state transfer costs for dynamic execution offloading in mobile cloud computing," *IEEE Trans. Mobile Comput.*, vol. 13, pp. 2648–2660, November 2014.

[11] H. Topcuoglu, S. Hariri, and M.-y. Wu, "Performance–effective and low–complexity task scheduling for heterogeneous computing," *IEEE Trans. Parall Dist. Syst.*, vol. 13, pp. 260–274, March 2002.

[12] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: making smartphones last longer with code offload," in *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services*, MobiSys '10, (New York, NY, USA), pp. 49–62, ACM, 2010.

[13] Q. Peng, A. Walid, J. Hwang, and S. H. Low, "Multipath TCP: Analysis, design, and implementation," *IEEE/ACM Trans. Netw.*, vol. 24, pp. 596–609, February 2016.

[14] M.-R. Ra, A. Sheth, L. Mummert, P. Pillai, D. Wetherall, and R. Govindan, "Odessa: enabling interactive perception applications on mobile devices," in *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, MobiSys '11, (New York, NY, USA), pp. 43–56, ACM, 2011.

[15] L. Yang, J. Cao, Y. Yuan, T. Li, A. Han, and A. Chan, "A framework for partitioning and execution of data stream applications in mobile cloud computing," *SIGMETRICS Perform. Eval. Rev.*, vol. 40, pp. 23–32, April 2013.

[16] V. De Maio and I. Brandic, "First hop mobile offloading of dag computations," in *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, CCGRID '18, pp. 83–92, IEEE, 2018.

[17] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, "Characterizing and profiling scientific workflows," *Future Gener. Comput. Syst.*, vol. 29, pp. 682–692, March 2013.

[18] S. E. Mahmoodi, R. Uma, and K. Subbalakshmi, "Optimal joint scheduling and cloud offloading for mobile applications," *IEEE Trans. Cloud Comput.*, April 2016.