Delay-Aware and User-Adaptive Offloading of Computation-Intensive Applications with Per-Task Delay in Mobile Edge Computing Networks

Tarik Chanyour*[1], Youssef HMIMZ[2], Mohamed EL GHMARY[3], Mohammed Ouçamah CHERKAOUI MALKI[4]
FSDM, LIIAN
Sidi Mohamed Ben Abdellah University
P.O. Box 1796, Atlas-Fez, Morocco

***Abstract*—Mobile-edge computing (MEC) is a new paradigm with a great potential to extend mobile users capabilities because-of its proximity. It can contribute efficiently to optimize the energy consumption to preserve privacy, and reduce the bottlenecks of the network traffic. In addition, intensive-computation offloading is an active research area that can lessen latencies and energy consumption. Nevertheless, within multi-user networks with a multi-task scenario, select the tasks to offload is complex and critical. Actually, these selections and the resources' allocation have to be carefully considered as they affect the resulting energies and delays. In this work, we study a scenario considering a user-adaptive offloading where each user runs a list of heavy computation-tasks. Every task has to be processed in its associated MEC server within a fixed deadline. Hence, the proposed optimization problem target the minimization of a weighted-sum normalized function depending on three metrics. The first is energy consumption, the second is the total processing delays, and the third is the unsatisfied processing workload. The solution of the general problem is obtained using the solutions of two sub-problems. Also, all solutions are evaluated using a set of simulation experiments. Finally, the execution times are very encouraging for moderate sizes, and the proposed heuristic solutions give satisfactory results in terms of users cost function in pseudo-polynomial times.**

***Keywords*—Mobile edge computing; user-adaptive offloading; computation-intensive offloading; per-task delay; tasks satisfaction optimization**

I. Introduction

A variety of recent Smart Mobile Devices (SMD) have capabilities to process some emerging attractive computation-intensive applications. Besides, these applications which are resource-hungry led to appear a novel kind of constraints related to resources insufficiency. They yield new defies particularly with regard to energetic and latency concerns. The offloading technique [1] in the context of Mobile Edge Computing (MEC) [2], [3] offers a chance to free these devices from generated heavy tasks by migrating them to a nearby edge infrastructure with extra powerful resources. Therefore, the power consumption of a mobile device as well as the latency responses can be reduced if the device can offload some of its heavy tasks to a MEC server.

The multi-user offloading problems with the single-task scenario in MEC networks was extensively considered. Previous works studied the energy consumption or processing delay optimization. To reduce the overall energy consumption, the authors of [4], studied the offloading decisions and the allocation of communication resources. Alike, in [5] an optimization problem is derived to select the best offloading policy while saving the energy consumption. The next work [6] present, in our knowledge, the first try to enhance the energy consumption while considering devices with multiple independent tasks. But, the authors impractically consider mobile devices with the same tasks' number. Lately, the authors of [7] studied a single-user scenario with multi-task setting while they optimize radio resources and local frequency. For the issues of joint resources optimization in MEC networks, many previous works jointly addressed the radio and computation resources optimization for the multi-user scenarios [8]. Besides, in [9] the authors jointly decide the allocation of resources (remote computation and communication) to optimize the energy consumption while devices intend to offload their tasks to a MEC server within a 5G heterogeneous network. Likewise in [10], the authors jointly optimize the offloading decisions and resources' allocation (both computation and communication). Nevertheless, they neglected the energy consumption during tasks processing at the server. Also, withing a cloud Fog environment in [11], the problem of resource allocation to optimize energy consumption with load balancing is studied.

Different to [12] where the system's energy only is optimized while all the offloadable tasks of a given SMD are constrained to the same delay, this paper presents a generalization scenario regarding mainly the following three points: the first generalization point relies on the per-task delay property where we consider every task with its proper delay constraint. The second point concerns the existence of a one-to-many association that links every SMD to many available MEC servers. Accordingly, each task has to be processed in its associated ES. Finally, the third point relies on the user-adaptive offloading property where we target the optimization of a normalized objective function that considers not only the energy consumption metric, but two other important metrics. The first is the sum of the accumulated processing delays, and the second is the total offloaded workload. Thus, the proposed system architecture offers the possibility for every user to adapt its offloading concerns according to its needs or/and constraints.

We organized the rest of this work as follows. We describe the system's model in Section II. The obtained optimization problems are presented in Section III, and their resolution approach is summarized in Section IV. Evaluation and results

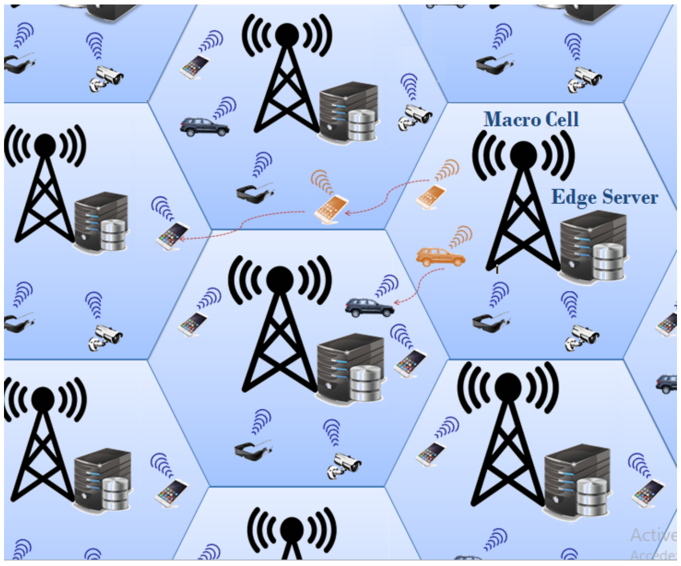

Fig. 1. A multi-server multi-task mobile edge-computing system

are presented in Section V. Finally, Section VI concludes the paper.

II. System Model

The adopted system's architecture in this work is shown in Fig. 1. It involves a set of Macro Cell (MC) where each MC is optionally equipped with an Edge Server (ES). The set of M available, supposed heterogeneous, edge servers is denoted $\mathbb{S} = \{s_1, s_2, ..., s_M\}$. Besides their traditional services, all macro-cells with ES can provide a set of independent computation-offloading services using the virtualization technology. Furthermore, we propose a distributed solution in each MC that considers its communication resources and the offloading decisions related to all SMDs within its coverage. Accordingly, a given MC, in this work, consists of a set of N Smart Mobile Devices (SMDs) denoted $\mathbb{E} = \{e_1, e_2, ..., e_N\}$. Each SMD is characterized by the parameters that are briefly presented in Table I.

Mainly, SMD i holds a list of n_i independent heavy tasks denoted $\tau_i = \{\tau_i^1, \tau_i^2, ..., \tau_i^{n_i}\}$. These tasks are assumed to be computationally intensive and delay sensitive. In addition, each task is viewed as an atomic input-data task, and cannot be divided into sub-tasks. Moreover, it is characterized by the following three properties $\tau_i^j \triangleq \langle d_i^j, \lambda_i^j, L_i^j \rangle$. In bits, the first property refers to the amount of the input parameters and program codes to transfer from the SMD to the edge server. In cycles, the second property specifies the workload referring to the computation amount needed to accomplish the task's processing. In seconds, the third property identifies the maximum tolerated delay within which τ_i^j has to be processed locally or at its associated edge server.

Moreover, within a given SMD i, its tasks are grouped into a set of n_i^g groups denoted $\mathbb{G}_i = \{g_1^1, g_1^2, ..., g_1^{n_i^g}\}$. Each group g_i^j is associated to a service hosted in an ES. The tasks of a given group are processed locally or transmitted via the current MC to their associated edge server. Accordingly, the processing frequencies related to SMD i are f_i^L for its local processing, and $f_{i,k}^S$ for remote processing at server s_k. For ease of use, the edge server s_k is denoted k. Moreover, we use a mapping function $g_i(.)$ to represent the association GROUP-SERVER of the tasks' of SMD i. Fig. 2 shows the list τ_1 and the corresponding group-server associations. Accordingly, the mapping function $g_1(.)$ is defined by the following mapping list $[(j, g_1(j))]_{j \in [\![1;6]\!]}$. This list contains three groups where the first $g_1 = \{\tau_1^2, \tau_1^3, \tau_1^5\}$ is associated to server s_1; the second group $g_2 = \{\tau_1^4, \tau_1^6\}$ is associated to server s_2, and the third $g_3 = \{\tau_1^1\}$ is associated to server s_3. Thus, the g_1 function is given by the mapping list $[(2,1),(3,1),(5,1),(4,2),(6,2),(1,3)]$. Here $g_1(j) = k$ refers to an association of task τ_1^j with server s_k. Moreover, it is assumed that the offloading process to remote servers experiences additional delay. Similar to the model in [7], we consider that this delay is proportional to the length of the data with a scaling factor δ_k that depends on the backhaul between the remote MC that shelter the ES s_k and the current MC. Then, the total experienced delay of a d bytes task equals $\delta_k.d$. Moreover, due to the complexity of evaluating the delay and energy consumption of the communication process between remote MCs and without loss of generality, we ignore the power consumption occurred at the backhaul network.

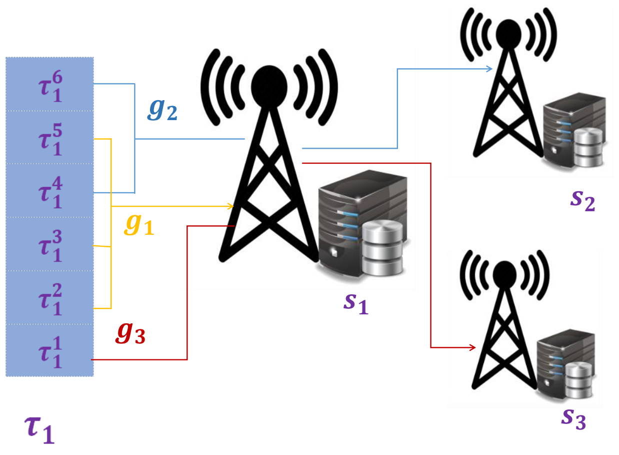

Fig. 2. Group-server association example

Further, the network model of our previous work [12] is adopted. Thus, we assume that e_i uses an estimated uplink rate r_i in each allocated subchannel. Additionally, with a number β_i of allocated subchannels, the total of its allocated uplink rate is $R_i = \beta_i r_i$.

A. The Offloading Model

All tasks are time constrained and can be processed either locally or offloaded to the edge server. Thus, the binary offloading decision variable for task τ_i^j is denoted α_i^j where $\alpha_i^j = 1$ refers to an offloading decision, whereas $\alpha_i^j = 0$ indicates a local processing decision.

$$\alpha_i^j \in \{0;1\} \qquad ; i \in \mathbb{E}; j \in [\![1;n_i]\!] \tag{1}$$

Additionally, the processing satisfaction of task τ_i^j is introduced using the binary variable γ_i^j where $\gamma_i^j = 0$ refers to a

TABLE I. IMPORTANT NOTATIONS

Notation	Definition
$\mathbb{E}$	Set of smart mobile devices
N	Total number of smart mobile devices
K	Total number of allocatable subchannels
K_t	Maximum allocatable number of subchannels per-SMD
N_t	Number of tasks threshold to enable "Heuristic Tasks Distribution"
N_m	Average tasks' number for all SMDs
n_i	Number of tasks handled by SMD i
τ_i	Computation tasks' set of SMD i
π_i	The parameters' set of SMD i
$D_{i,0}, D_i$	Total initial and offloaded tasks' data size at SMD i
$\Lambda_{i,0}, \Lambda_i$	Total initial and offloaded tasks' workload at SMD i
f_i^L	Local CPU frequency of SMD i (cycles/s)
$f_{i,k}^S$	Allocated CPU frequency for SMD i at s_k (cycles/s)
ξ_i^L	Energy coefficient depending on the chip architecture of SMD i and f_i^L
$\xi_{i,k}^S$	Energy coefficient depending on the chip architecture of s_k and $f_{i,k}^S$
p_i^T	Data transmission power of SMD i
r_i	Uplink rate of SMD i
$\delta_{i,k}$	Backhaul delay scaling factor related to s_k in SMD i

satisfied processing, otherwise $\gamma_i^j = 1$.

$$\gamma_i^j \in \{0;1\} \qquad ; i \in \mathbb{E}; j \in [\![1;n_i]\!] \tag{2}$$

This variable is defined such that $t_{i,j}^L + t_{i,j}^O \leqslant L_i^j \iff \gamma_i^j = 0$. Here L_i^j is the τ_i^j task's latency requirement. $t_{i,j}^L + t_{i,j}^O$ is the time to process τ_i^j and it is given in equations (7) and (10). This expression is formulated using the following constraint where M is a sufficiently large constant:

$$-M\gamma_i^j \leqslant L_i^j - t_{i,j}^L - t_{i,j}^O < M(1-\gamma_i^j) \qquad ; i\in\mathbb{E}; j\in[[1;n_i]] \tag{3}$$

The total number of subchannels assigned to e_i is denoted β_i where:

$$\beta_i \in [\![0;K_t]\!] \qquad ; i \in \mathbb{E} \tag{4}$$

Here K_t is a threshold value that is chosen according to N in such a way that it can take K where $N = 1$; and decreases with increasing value of N. Additionally, the sum of all allocated subchannels must not exceed K the total available subchannels, which gives:

$$\sum_{i\in\mathbb{E}} \beta_i \leqslant K \tag{5}$$

The decision $\beta_i = 0$ forbids offloading for e_i. In this case, it has to locally process all its tasks; whereas $\beta_i \neq 0$ indicates that e_i has to offload at least one task. This fact leads to the following offloading property $(\beta_i = 0) \iff (\sum_{j=1}^{n_i} \alpha_i^j = 0)$ which must hold for every SMD i. It can be formulated as:

$$\beta_i + \sum_{j=1}^{n_i} \alpha_i^j \leqslant 2\beta_i \sum_{j=1}^{n_i} \alpha_i^j \qquad ; i \in \mathbb{E} \tag{6}$$

B. Processing Model

If e_i locally executes task τ_i^j, its processing time in seconds lasts $\frac{\lambda_i^j}{f_{i,L}}$. Then, with its processing order, the actual local processing time of task τ_i^j is:

$$t_{i,j}^L = (1-\alpha_i^j)\sum_{k=1}^{j}(1-\alpha_i^k)\frac{\lambda_i^k}{f_{i,L}} \tag{7}$$

If e_i offload task τ_i^j to server s_k, its offloading time $t_{i,j}^O$ includes both, the transmission time $t_{i,j}^{Trans}$ and the waiting time $t_{i,j}^{Wait}$; which gives $t_{i,j}^O = t_{i,j}^{Trans} + t_{i,j}^{Wait}$. The second part includes the backhaul delay $t_{i,j}^{Delay}$, the execution time $t_{i,j}^{Exec}$, and the time to receive the result out from the server $t_{i,j}^{Res}$; which gives $t_{i,j}^{Wait} = t_{i,j}^{Delay} + t_{i,j}^{Exec} + t_{i,j}^{Res}$. Because the data size of the result is much smaller than the input data size, we ignore the time and the energy consumption of receiving out the result(see [6], [8], [13], [14]). Furthermore, without considering the delays relative to the processing order of task τ_i^j, these delays in seconds are given by :

$$\left(t_{i,j}^{Trans}, t_{i,j}^{Delay}, t_{i,j}^{Exec}\right) = \left(\frac{d_i^j}{\beta_i r_i}, \delta_{g_i(j)} d_i^j, \frac{\lambda_i^j}{f_{i,g_i(j)}^S}\right) \tag{8}$$

Now, with its processing order, the actual transmission delay of task τ_i^j is $t_{i,j}^{Trans} = \frac{\alpha_i^j}{\beta_i r_i}\sum_{k=1}^{j}\alpha_i^k d_i^k$. For e_i and with the sequential processing of the tasks' group in each server, the waiting delay of task τ_i^j is:

$$t_{i,j}^{Wait} = \delta_{g_i(j)} \sum_{k=1}^{j} \mathbb{1}(g_i(j)=g_i(k))\alpha_i^k d_i^k + \frac{1}{f_{i,g_i(j)}^S}\sum_{k=1}^{j}\mathbb{1}(g_i(j)=g_i(k))\alpha_i^k\lambda_i^k \tag{9}$$

Subsequently, its offloading delay is:

$$t_{i,j}^O = \frac{\alpha_i^j}{\beta_i r_i}\sum_{k=1}^{j}\alpha_i^k d_i^k + \sum_{k=1}^{j}\left(\mathbb{1}(g_i(j)=g_i(k))\alpha_i^k\left(\delta_{g_i(j)} d_i^k + \frac{\lambda_i^k}{f_{i,g_i(j)}^S}\right)\right) \tag{10}$$

Also, for e_i and the tasks' processing orders we have:

$$t_i^{Trans} = \frac{1}{\beta_i r_i}\sum_{j=1}^{n_i}\alpha_i^j d_i^j = \frac{D_i}{\beta_i r_i} \tag{11}$$

C. Energetic Model

In this section, all energy expressions are given in Joule. Then, the processing of task τ_i^j consumes the amount of energy $e_{i,j} = \xi_i \lambda_i^j$ where ξ_i is a power coefficient depending on the processing unit's chip architecture and the processing frequency [9], [14]. Thus, the local energy consumption of task τ_i^j is $e_{i,j}^L = \xi_{i,L}\lambda_i^j$. Accordingly, the e_i's local energy consumption is:

$$e_i^L = \xi_{i,L}\sum_{j=1}^{n_i}\left(1-\alpha_i^j\right)\lambda_i^j = \xi_{i,L}(\Lambda_{i,0} - \Lambda_i) \tag{12}$$

Here, $\xi_{i,L}\Lambda_{i,0} = E_{i,0}^L$ is the energy consumption of the local processing of all tasks.

Furthermore, the energy consumption of the offloading process is $e_i^O = e_i^{Trans} + e_i^{Exec}$. Here e_i^{Trans} is the transmission consumption occurred at e_i, and e_i^{Exec} is the energy consumption occurred during the processing of offloaded tasks within all ESs. The first is obtained by multiplying the transmission period in the offloading processes by the transmission power. Thus:

$$e_i^{Trans} = P_i^T t_i^{Trans} = \frac{P_i^T D_i}{\beta_i r_i} \tag{13}$$

Again, the energy consumption while executing task τ_i^j at its dedicated edge server is $e_{i,j}^{Exec} = \xi_{i,g_i(j)}^S \lambda_i^j$ [9], [14]. As a result, the overall offloading energy consumption occurred at e_i is :

$$e_i^O = \begin{cases} \dfrac{P_i^T D_i}{\beta_i r_i} + \sum_{j=1}^{n_i} \xi_{i,g_i(j)}^S \alpha_i^j \lambda_i^j & ;\beta_i \neq 0 \\ 0 & ;\beta_i = 0 \end{cases} \tag{14}$$

D. The Cost Function

Now, the total e_i's energy consumption can be formulated using its tasks' offloading allocations given by a vector of n_i binary variables: $\alpha_i = \left(\alpha_i^1, ..., \alpha_i^{n_i}\right)$ and its allocated number of uplink subchannel(s) β_i. This energy includes the local processing consumption (if some tasks are locally processed) and the offloading process consumption (if some tasks are offloaded). It is formulated as:

$$E_i(\alpha_i,\beta_i) = \begin{cases} \xi_{i,L}(\Lambda_{i,0} - \Lambda_i) + \dfrac{P_i^T D_i}{\beta_i r_i} + \sum\limits_{j=1}^{n_i} \xi_{i,g_i(j)}^S \alpha_i^j \lambda_i^j & ;\beta_i \neq 0 \\ \xi_{i,L}\Lambda_{i,0} & ;\beta_i = 0 \end{cases} \tag{15}$$

Similarly, w.r.t. α_i and β_i, the e_i's overall processing time denoted $T_i\left(\alpha_i, \beta_i\right)$ includes the processing delays of the local and the offloaded tasks using equations (7) and (10). Thus, we have:

$$T_i\left(\alpha_i, \beta_i\right) = \sum_{j=1}^{n_i} \left(t_{i,j}^L + t_{i,j}^O\right) \tag{16}$$

Finally, w.r.t. the e_i's tasks satisfactions given by a n_i binary variables' vector $\gamma_i = \left(\gamma_i^1, ..., \gamma_i^{n_i}\right)$, the total workload of its unsatisfied tasks is:

$$W_i\left(\gamma_i\right) = \sum_{j=1}^{n_i} \gamma_i^j \lambda_i^j \qquad ;i \in \mathbb{E} \tag{17}$$

At this stage, we formulate a multi-criteria offloading with an elastic budget model for every SMD. Thus, each device looks at three metrics: its energy consumption and latencies while processing all its tasks, and its unsatisfied total processing workload. The proposed multi-objective function for each SMD is expressed with a weighted sum of the three proposed metrics. Moreover, the contribution of e_i is formulated using the following weighted-sum function:

$$\mathbb{F}_i(\alpha_i,\beta_i,\gamma_i) = x_i \frac{E_i(\alpha_i,\beta_i)}{E_{i,0}^L} + y_i \frac{T_i(\alpha_i,\beta_i)}{L_i} + (1 - x_i - y_i)\frac{W_i(\gamma_i)}{\Lambda_{i,0}} \tag{18}$$

Here, x_i and y_i are two parameters related to SMD i that determine its offloading policy. Their values are choosen by the SMD policy such that the following three weights x_i, y_i and $1 - x_i - y_i$ are in the interval [0,1]. By deciding these weights, the user can adjust the priority to give to each metric. This operation depends on its preference regarding delay-sensitivity, energetic constraints, and its processing capability. For example, (x_i, y_i) can be set to (0,1) for SMDs running delay sensitive applications whereas they can be set to (1,0) for energy-constrained devices. Also, for SMDs with bad processing capability, (x_i, y_i) can be set to (0,0).
Additionally, the three denominators in this expression are variable-independent terms that serve to normalize the cost function. As a result, the overall cost function of all SMDs is finally expressed as:

$$\mathbb{F}\left(\alpha, \beta, \gamma\right) = \sum_{i \in \mathbb{E}} \mathbb{F}_i\left(\alpha_i, \beta_i, \gamma_i\right) \tag{19}$$

The variables are given by α (a global vector composed of N vectors, each vector α_i of length n_i contains the e_i's tasks offloading decisions), γ (a global vector composed of N vectors, each vector γ_i of length n_i contains the e_i's tasks satisfactions variables) and β (a global vector containing N radio spectrum allocation β_i for all SMDs).

III. The Optimization Problems

A. The General Problem Formulation

In our proposed optimization problem, we target to minimize three important metrics that influence the system's performance and the users' satisfaction. Accordingly, we formulate our optimization problem which we denote problem $\mathscr{P}1$ as :

$$\begin{aligned} &\mathscr{P}1\text{:}\underset{\{\alpha,\beta,\gamma\}}{\text{minimize}}\mathbb{F}(\alpha,\beta,\gamma) \\ &\text{s.t. } (1),(2),(3),(4),(5),(6) \end{aligned} \tag{20}$$

B. Problem Decomposition

The $\mathscr{P}1$ problem is an integer programming optimization problem. To deal with its high computational complexity, we decompose it into a sub-optimal scheme. $\mathscr{P}1$ decides for the offloading and satisfaction of tasks as well as the subchannels allocations. Thus, we derive a set of sub-problems denoted User-adaptive Offloading and Satisfaction Decisions (UOSD) where each instance concerns a given SMD and a subchannels allocation.

1) The User-Adaptive Offloading and Satisfaction Decisions sub-problem: The UOSD sub-problem for user e_i is denoted $\mathscr{P}2(i, \beta_i)$. Its formulation uses a known fixed number of subchannels β_i which satisfies $\beta_i \in [\![1; K_t]\!]$. Then, the objective function is $\mathbb{F}_i\left(\alpha_i, \beta_i, \gamma_i\right)$. Accordingly, the variables of decision are the following two vectors $\alpha_i = \left(\alpha_i^1, \alpha_i^2, ..., \alpha_i^{n_i}\right)$ and $\gamma_i = \left(\gamma_i^1, \gamma_i^2, ..., \gamma_i^{n_i}\right)$. Finally, it is formulated as:

$$\begin{array}{lll} \mathscr{P}2(i,\beta_i)\text{:} & \underset{\{\alpha_i,\gamma_i\}}{\text{minimize}} & \mathbb{F}_i(\alpha_i,\beta_i,\gamma_i) \\ \text{s.t. } (C_{21}) & \alpha_i^j,\gamma_i^j \in \{0;1\} & ;j \in [\![1;n_i]\!]. \\ (C_{22}) & -M\gamma_i^j \leqslant L_i^j - t_{i,j}^L - t_{i,j}^O < M(1-\gamma_i^j) & ;j \in [\![1;n_i]\!]. \\ (C_{23}) & \sum\limits_{j=1}^{n_i}\alpha_i^j \geqslant 1 & \end{array} \tag{21}$$

2) The Subchannels Allocations sub-problem: The UOSD sub-problem's instances related to e_i are obtained by varying the value of β_i in the interval $[\![1; K_t]\!]$ with only one possible allocation. The expected result is the vector of subchannels allocation given by $\beta = (\beta_1, \beta_2, ..., \beta_N)$. Consequently, we build a matrix M with N rows and $K_t + 1$ columns. The first column of M contains the cost functions obtained with $\beta_i = 0$. For the other K_t columns, every cell M_{ij} stores the solution of problem $\mathscr{P}2(i, j)$. Then, to decide an allocation of β_i in row M_i, we use a binary indicator x_i^j to denote the decision of the M_{ij} element selection in row M_i ($x_i^j = 1$ refers to the allocation $\beta_i = j$ and $x_i^j = 0$ leads to ignore the M_{ij} cell in the cost function). Furthermore, one possible value for β_i gives the constraint $\sum_{j=0}^{K_t} x_i^j = 1$. Additionally, the total allocation of subchannels for e_i in row i is exactly $\sum_{j=0}^{K_t} j.x_i^j$. Therefore, the total allocation of subchannels gives the number $\sum_{i\in\mathbb{E}}\sum_{j=0}^{K_t} j.x_i^j$ and must be less or equal to K. Consequently, the general formulation of the resulting problem becomes:

$$\begin{aligned}
&\mathscr{P}3: && \underset{\{x_i^j\}}{\text{minimize}} \sum_{i\in\mathbb{E}}\sum_{j=0}^{K_t} x_i^j M_{ij} \\
&\text{s.t. } (C_{31}) && \sum_{i\in\mathbb{E}}\sum_{j=0}^{K_t} j.x_i^j \leqslant K. \\
&(C_{32}) && \sum_{j=0}^{K_t} x_i^j = 1 \qquad ; i\in\mathbb{E}. \\
&(C_{33}) && x_i^j \in \{0;1\} \qquad ; i\in\mathbb{E}; j\in[\![0;K_t]\!].
\end{aligned} \tag{22}$$

This formulation corresponds to a general case of the LSP problem presented in [15]. Also, it is the minimization form of a special case of the Multiple-Choice Knapsack Problem (MCKP) [16] where the weights are the integers in $[\![0;K_t]\!]$. For more details about this last problem, interested readers can refer to [17], [16]

IV. Problems' Resolution

To solve problem $\mathscr{P}3$ we propose two algorithms: the first is the recent BISSA [17] solution which demonstrated its effectiveness in our previous work [12]. It is a pseudo-polynomial time complexity algorithm. The second is the Exact Brute Force Search algorithm which we denote (MCKP-BFS). With the condition $K \gg K_t$, its time complexity equals $O({K_t}^N)$. Its pseudo-code is summarized in Algorithm 1. To run this algorithm, we use the matrix M that is build s.t. its first column contains the results of the local processing. Its last K_t columns of N rows are built s.t. each cell M_{ij} contains the $\mathscr{P}2(i,j)$ sub-problem solution given by three components $(\alpha^*_{i,j}, \gamma^*_{i,j}, F^*_{i,j})$. The pseudo-code represents a recursive implementation of the MCKP-BFS solution. M is the matrix, N is the rows count, K_t is the columns count - 1, and k is a variable representing the number of available subchannels.

A. The UOSD Exact Solution

On the one hand, the tasks distribution sub-problem $\mathscr{P}2(i,\beta_i)$ relies on determining α_i and γ_i that correspond to the minimum cost function for SMD i. The exhaustive search over all possible solutions using a Brute Force Search that we denote (BFS-TD) is an $O(2^{n_i})$ time complexity solution. It is presented in Algorithm 2. While resolving the global problem, this solution is used in the first phase to construct matrix M. It is based on solving $N * Kt$ instances of sub-problem $\mathscr{P}2(i,\beta_i)$. Then, M is used as an input to the second sub-problem $\mathscr{P}3$.

1) Experiment 1: To investigate the feasibility and limitation of Algorithm 2, we carry the first experiment where we measure the achieved times of each phase. additionally, the construction of M is highly influenced by the distribution of the tasks' count n_i in each row, whereas the MCKP resolution is not affected. To achieve that, we vary N between 10 and 100 while we take K=600, Kt=15, and n_i in 8;9, and we use the simulation parameters described in Table II.

Algorithm 1 : MCKP-BFS

```
Require: $M, N, K_t$, and $k$
Ensure: the subchannels allocation $\beta^* = (\beta_1^*, \beta_2^*, ..., \beta_N^*)$;
 1: if $k < 0$ then
 2:     return $\varnothing$;
 3: end if
 4: if $N = 1$ then
 5:     return $\beta = (min(k, K_t))$;
 6: else
 7:     $F^* \leftarrow \infty$;
 8:     for $\beta_N = 0$ to $K_t$ do
 9:         $X \leftarrow$ MCKP-BFS$(M, N-1, K_t, k-\beta_N)$;
10:         if $X \neq \varnothing$ then
11:             $\beta^+ \leftarrow (X, \beta_N) = (\beta_1^+, ..., \beta_{N-1}^+, \beta_N)$;
12:             $\alpha \leftarrow \left(\alpha^*_{1,\beta_1^+}, ..., \alpha^*_{N-1,\beta_{N-1}^+}, \alpha^*_{N,\beta_N},\right)$;
13:             $\gamma \leftarrow \left(\gamma^*_{1,\beta_1^+}, ..., \gamma^*_{N-1,\beta_{N-1}^+}, \gamma^*_{N,\beta_N},\right)$;
14:             $F \leftarrow \sum_{i=1}^N M[i][\beta_i^+]$;
15:             if $F < F^*$ then
16:                 $(F^*, \beta^*) \leftarrow (F, \beta^+)$
17:             end if
18:         end if
19:     end for
20: end if
```

Algorithm 2 : Brute Force Search based Tasks distribution with β_i subchannel

```
Require: $\pi_i, \beta_i$
Ensure: the offloading and satisfaction vectors $\alpha_i^*$, $\gamma_i^*$ and the
    corresponding cost $F^*$
 1: if $\beta_i = 0$ then
 2:     $\alpha_i^* \leftarrow \boldsymbol{0}_{n_i}$;
 3:     Build $\gamma_i^*$ using $\alpha_i^*, \beta_i$ and equation (3);
 4:     $F^* \leftarrow \mathbb{F}_i(\alpha_i^*, \gamma_i^*, \beta_i)$ according to (18);
 5: else
 6:     $F^* \leftarrow \infty$;
 7:     for k=1 to $2^{n_i} - 1$ do
 8:         $\alpha_i \leftarrow bin(i)$;
 9:         Build $\gamma_i$ using $\alpha_i, \beta_i$ and equation (3);
10:         $F \leftarrow \mathbb{F}_i(\alpha_i, \gamma_i, \beta_i)$ according to (18);
11:         if $F < F^*$ then
12:             $(\alpha_i^*, \gamma_i^*, F^*) \leftarrow (\alpha_i, \gamma_i, F)$
13:         end if
14:     end for
15: end if
16: return $(\alpha_i^*, \gamma_i^*, F^*)$
```

Fig. 3 shows the average execution time of the BFS-TD algorithm (for both values n_i = 8 and 9) and BISSA solution while we vary the total number of SMDs N between 10 and 100. In view of the obtained results, the MCKP resolution using BISSA realizes stable execution times. Indeed, for N=10 the corresponding execution times for BFS-TD(n_i=8), BFS-TD(n_i=9), BISSA are respectively 15.92, 69.87, 0.01ms. For N=50 they respectively reach 264.9,5 393.76 and 0.07ms. For N=100 they respectively reach 244.95, 383.76 and 5.61ms. It shows also an important increasing of the execution time w.r.t. N. Accordingly, this experiment shows that the exact resolutions of the first sub-problem is time consuming especially for important values of n_i and highly exceeds the MCKP resolution time using BISSA. Subsequently, an approximate solution is solicited which is the concern of the following section.

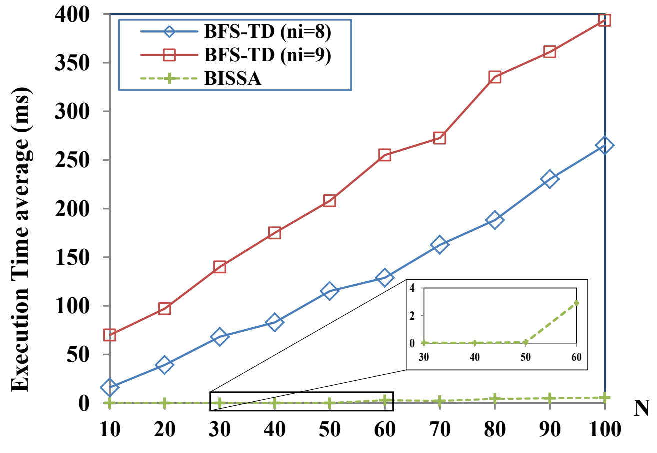

Fig. 3. Execution Time with N; K=600, Kt=15, $n_i \in \{8; 9\}$

B. The UOSD Heuristic Solution

On the other hand, given the binary form of the problem and the small decision time constraint, we propose a Simulated Annealing based heuristic solution. This heuristic optimization technique is characterized by its simplicity and general applicability features while being very efficient in terms of speed compared to other techniques. Probabilistically, this algorithm accepts not only cost gain, but also cost degradation in order to leave the local minima.

Accordingly, we propose to implement a relevant variant: Very Fast Simulated Annealing [18] which we denote VFSA-TD. Different to the classical Simulated Annealing (SA) algorithm which we adopted in our previous work [12] which denote SA-TD, This variant is characterized with a small convergence time. In this algorithm, the thermodynamic system's energy is represented by the cost function $\mathbb{F}_i$. During the solutions' space probabilistic iteration, the acceptance of the current state is done according to the following principle: we obviously accept the new state when its energy is less than its previous energy; otherwise, the new state is accepted when the probability $min\left\{exp\left(\frac{F^*-F_{new}}{T}\right),1\right\}$ is greater than a random value p. Here p is picked from a uniform distribution $U[0,1]$. Besides, a decreasing temperature, leads to a decrease in the probability for the system to shift to a new state. The temperature schedule in the SA-TD and the VFSA-TD algorithms are respectively given by:

$$T_k = T_0(a^k) \tag{23}$$

$$T_k = T_0 exp\left(-0.5k^{\frac{1}{2n_i}}\right) \tag{24}$$

Here, k is the current iteration number, a is a decreasing factor s.t. $0.5 < a < 1$. The detail of the solution is presented in Algorithm (3).

Algorithm 3 : Fast Simulated Annealing Tasks Distribution with β_i subchannel

```
Require: $\pi_i$,$\beta_i$,$k^{max}$,$T_0$, and an initial non-empty offloading vector $\alpha_0$
Ensure: the offloading and satisfaction vectors $\alpha_i^*$, $\gamma_i^*$ and the corresponding cost $F^*$
1:  if $\beta_i$=0 then
2:      $\alpha_i^* \leftarrow \mathbf{0}_{n_i}$;
3:      Build $\gamma_i^*$ using $\alpha_i^*$,$\beta_i$ and equation (3);
4:      $F^* \leftarrow \mathbb{F}_i(\alpha_i^*,\gamma_i^*,\beta_i)$ according to (18);
5:  else
6:      $\alpha_i^* \leftarrow \alpha_i \leftarrow \alpha_0$;
7:      Build $\gamma_i$ using $\alpha_i$,$\beta_i$ and equation (3);
8:      Calculate $F^*=\mathbb{F}_i(\alpha_i^*,\gamma_i^*,\beta_i)$ according to (18);
9:      for k=1 to $k^{max}$ do
10:         $T \leftarrow T_0 e^{-0.5k^{\frac{1}{2n_i}}}$;
11:         $\alpha_{new} \leftarrow rand\_neighbour(\alpha_i)$;
12:         if $\alpha_{new} \neq \mathbf{0}_{n_i}$ then
13:             $\alpha_i \leftarrow \alpha_{new}$;
14:             Build $\gamma_{new}$ using $\alpha_{new}$,$\beta_i$ and equation (3);
15:             $F_{new} \leftarrow \mathbb{F}_i(\alpha_{new},\gamma_{new},\beta_i)$ according to (18)
16:             if $min\left\{e^{\frac{F^*-F_{new}}{T}},1\right\} \geqslant random(0,1)$ then
17:                 $(\alpha_i^*,\gamma_i^*,F^*) \leftarrow (\alpha_{new},\gamma_{new},F_{new})$
18:             end if
19:         end if
20:     end for
21: end if
22: return $(\alpha_i^*,\gamma_i^*,F^*)$
```

As input, Algorithm 3 requires the parameters' vector π_i of SMD i, the allocated subchannel(s) number β_i, the maximum iterations count parameter k^{max}, the initial temperature value T_0, and an initial offloading decision vector α_0. In lines 1 to 5, we handle the all-local case. We build the tasks' satisfaction vector (line 7), then we initialize the optimal cost F^*(line 8). Then we use a for loop (line 9) to repeat the annealing process using k^{max} iterations. At each step, the temperature value T is updated (line 10); then, we generate a neighboring state α_{new} of the current state α_i (line 11). If it is non-null, we build its corresponding tasks' satisfaction vector (line 14), then we calculate the new cost F_{new}(line 15). Then, we try to accept the new state using a probabilistic test(lines 16 to 18). Here, $random(0,1)$ is a function's call that uniformly generates a random number in $[0,1]$.

Despite the exponential temporal complexity of the BFS-TD method compared to the pseudo-linear complexity of SA-based methods, the BFS-TD temporal performance is acceptable for moderate values of the number of tasks n_i. Even, they are largely superior for the values in the interval [1,8]. Subsequently, to exploit this fact, we introduce an integer threshold parameter N_t. It is used s.t. if $n_i > N_t$ we use the SA based approximate heuristic solutions; otherwise, we use the exact Brute Force Search solution.

V. Evaluation and Results

In this section, we present the proposed experiments used to evaluate our proposed solutions. We were mainly based on the execution time and the cost function metrics. An algorithm's *execution time* is given by its averaged running-time, while its *cost function* is given by the averaged realizations of the cost function $\mathbb{F}$. Besides, all presented results in this work are averaged with 100 times executions.

A. Simulation Setup

All developed C++ simulation programs were built with GCC version 6.4.0. and run using a 2.4GHz Intel Core i5 processor in a PC with a maximum 8GB of RAM. Moreover, the basic parameters of the simulation experiments are listed in Table II.

TABLE II. SIMULATIONS' PARAMETERS

Parameter	values
n_i	$[2, 13]$
$D_{i,0}$	$[0.2, 2]$ MB
$\Lambda_{i,0}$	$[1, 2]$ GCycle
L_i^j	$[10, 1000]$ seconds
f_i^L	$[100, 300]$ MHz
$f_{i,k}^S$	$[3, 4]$ GHz
ξ_i^L	$[1, 10] * 10^{-10} s^{-2}$
$\xi_{i,k}^S$	$[9, 40] * 10^{-11} s^{-2}$
P_i^T	0.1 Watt
r_i	$[200, 400]$ Kb/s
$\delta_{i,k}$	$\{0\} \cup [1, 2] * 10^{-3}$ s/Kb
T_0	300

B. Experiment 2

The second experiment studies the performance of the BISSA algorithm compared to the optimal MCKP-BFS algorithm. In this experiment we vary the SMDs' number (N) between 2 and a maximum feasible experimentation value $N = 8$. Fig. 4 shows the averaged two metrics (per-SMD cost function and execution time) w.r.t. the total number of SMDs N. The right side of this figure shows the variation of the execution time of the proposed solutions. Accordingly, the BISSA solution realizes stable averaged execution times. In fact, with $n_i = 4$ and $N = 8$ it reaches only 0.42; whereas the optimal MCKP-BFS solution attains $185696.81ms$. Similarly, with $n_i = 8$ and $N = 8$ it reaches $5.37ms$; whereas the optimal MCKP-BFS solution attains $143568.47ms$. Besides, the cost function achievement for these solutions is shown in the left part of this figure. It shows that the obtained results are the same for both solutions. Indeed, this experiment shows that the more is the number of SMDs the more is the execution time high, especially with an exponential variation for the optimal MCKP-BFS solution, and a stable execution times for BISSA.

C. Experiment 3

Now, we introduce the third experiment where we study the solutions' performance related to sub-problem $\mathscr{P}2\left(i, \beta_i\right)$. Thus, for one SMD only, we vary n_i between 2 and 30 while we record the average of the minimum of the cost function using the optimal BFS-TD method, the SA-TD, and the VFSA-TD heuristic methods. In each case we show the averaged execution time. Fig. 5 and 6 depict the experiment's results. They show the cost function besides the execution time w.r.t. n_i.

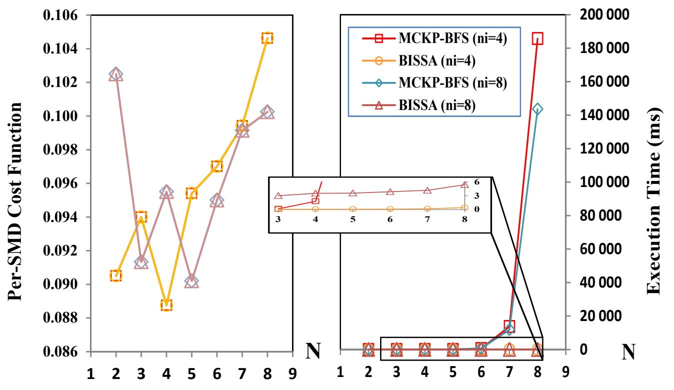

Fig. 4. Cost function and Execution Time with N; K=100, Kt=15, $n_i \in \{4; 8\}$

Hence, according to Fig. 5 that shows the obtained results in terms of Cost achievements, a small distance separating the results of all three methods is noticed.

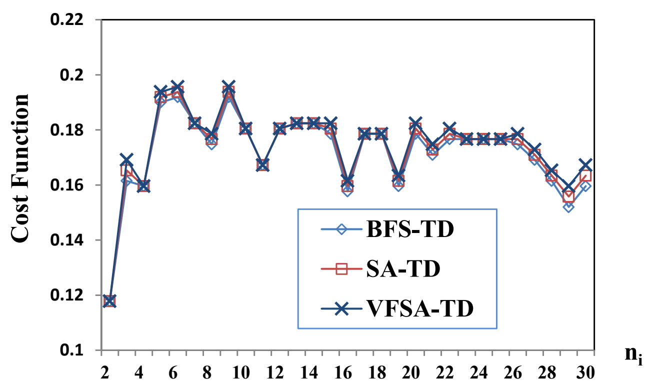

Fig. 5. Cost function performance $n_i \in [2; 30]$

Moreover, Fig. 6 shows the execution's time achievements. The obtained results illustrate important times' values for the BFS-TD method. Indeed, for $n_i = 20$ and 30 the execution time attains respectively 121.45 and $201544.20ms$. In this part of the figure and for clarity reason, we zoomed the figure to show the achievements of the heuristic solutions. as reported in the figure, for $n_i = 20$ and 30, both SA-TD and VFSA-TD heuristic methods give a stable averaged execution time that respectively attains only 0.030 and 0.086 ms for SA-TD, and 0.024 and 0.072 ms for VFSA-TD. Furthermore, as mentioned before, the BFS-TD achievements for small values of n_i between 2 and 8 in terms of execution time outrun both heuristic methods' performance. Consequently, the threshold $N_t = 8$ has to be set for a logical use of the heuristic methods.

VI. CONCLUSIONS

This paper considers a user-adaptive offloading problem with resource allocation in a multi-server Mobile Edge Com-

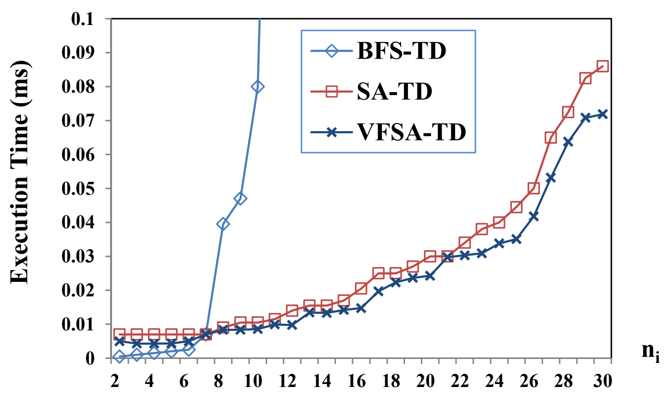

Fig. 6. Execution time performance $n_i \in [2; 30]$

puting network. We considered N smart mobile devices possessing many computation-intensive tasks each. The resulting optimization problem jointly optimizes the energy, the processing delays, and the unsatisfied processing workloads. Due to its combinatorial nature that leads to high complexity solutions, we decomposed it using a sub-problem in a first phase to build a matrix. Then, the optimal resource allocation is decided by solving a second sub-problem. To evaluate the components of the general solution, we designed a set of experiments to evaluate their performance using simulations. Accordingly, the first sub-problem exact solution is time consuming for experiments with large number of tasks. Consequently, we proposed an approximate solution based on the Very Fast Simulated Annealing algorithm. This solution is very efficient in terms of its execution time as well as its results. On the other hand, the BISSA solution for the obtained MCKP special case is much efficient and gives, in reasonable execution time, comparable result to the exact resolution. In perspectives, we plan to study services migration with backhauls' delays optimization in this proposed system.

REFERENCES

[1] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[2] G. Premsankar, M. Di Francesco, and T. Taleb, "Edge computing for the internet of things: A case study," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 1275–1284, 2018.

[3] M. Kadhum, S. Manaseer, and A. L. A. Dalhoum, "Cloud-edge network data processing based on user requirements using modify mapreduce algorithm and machine learning techniques," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 12, pp. 307–320, 2019.

[4] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 2, pp. 89–103, 2015.

[5] X. Xu, Y. Li, T. Huang, Y. Xue, K. Peng, L. Qi, and W. Dou, "An energy-aware computation offloading method for smart edge computing in wireless metropolitan area networks," *Journal of Network and Computer Applications*, vol. 133, pp. 75–85, 2019.

[6] M.-H. Chen, B. Liang, and M. Dong, "Multi-user multi-task offloading and resource allocation in mobile cloud systems," *IEEE Transactions on Wireless Communications*, vol. 17, no. 10, pp. 6790–6805, 2018.

[7] H. Li, "Multi-task offloading and resource allocation for energy-efficiency in mobile edge computing," *International Journal of Computer Techniques*, vol. 5, no. 1, pp. 5–13, 2018.

[8] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 974–983, 2015.

[9] K. Zhang, Y. Mao, S. Leng, Q. Zhao, L. Li, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, "Energy-efficient offloading for mobile edge computing in 5g heterogeneous networks," *IEEE access*, vol. 4, pp. 5896–5907, 2016.

[10] P. Zhao, H. Tian, C. Qin, and G. Nie, "Energy-saving offloading by jointly allocating radio and computational resources for mobile edge computing," *IEEE Access*, vol. 5, pp. 11255–11268, 2017.

[11] B. Hayat, M. Nauman, S. Yousaf, M. Mehmood, and H. Saleem, "Efficient energy utilization in cloud fog environment," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 04, pp. 617–623, 2019.

[12] T. Chanyour, M. El Ghmary, Y. Hmimz, and M. O. Cherkaoui Malki, "Energy-efficient and delay-aware multitask offloading for mobile edge computing networks," *Transactions on Emerging Telecommunications Technologies*. doi: 10.1002/ett.3673.

[13] B. Gu and V. S. Sheng, "A robust regularization path algorithm for ν-support vector classification," *IEEE Transactions on neural networks and learning systems*, vol. 28, no. 5, pp. 1241–1248, 2017.

[14] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2015.

[15] A. Agra and C. Requejo, "The linking set problem: a polynomial special case of the multiple-choice knapsack problem," *Journal of Mathematical Sciences*, vol. 161, no. 6, pp. 919–929, 2009.

[16] P. Sinha and A. A. Zoltners, "The multiple-choice knapsack problem," *Operations Research*, vol. 27, no. 3, pp. 503–515, 1979.

[17] E. M. Bednarczuk, J. Miroforidis, and P. Pyzel, "A multi-criteria approach to approximate solution of multiple-choice knapsack problem," *Computational Optimization and Applications*, vol. 70, no. 3, pp. 889–910, 2018.

[18] D. Pei, J. A. Quirein, B. E. Cornish, D. Quinn, and N. R. Warpinski, "Velocity calibration for microseismic monitoring: A very fast simulated annealing (vfsa) approach for joint-objective optimization," *Geophysics*, vol. 74, no. 6, pp. WCB47–WCB55, 2009.