RAIRO-Oper. Res. 55 (2021) 873–897
https://doi.org/10.1051/ro/2021043

**RAIRO Operations Research**
www.rairo-ro.org

# AN OPTIMAL SOLUTION FOR THE BUDGETS ASSIGNMENT PROBLEM

Mahdi Jemmali[1,2,3,*]

**Abstract.** Municipalities are service organizations that have a major role in strategic planning and community development that consider the future changes and society developments, by implementing set of projects with pre-allocated budgets. Projects have standards, budgets and constraints that differ from one community to another and from one city to another. Fair distributing of different projects to municipalities, while ensuring the provision of various capabilities to reach developmental role is NP-Hard problem. Assuming that all municipalities have the same strategic characteristics. The problem is as follows: given a set of projects with different budgets, how to distribute all projects to all municipalities with a minimum budget gap between municipalities. To derive equity distribution between municipalities, this paper developed lower bounds and eleven heuristics to be utilized in the branch-and-bound algorithms. The performance of the developed heuristics, lower bounds and the exact solutions are presented in the experimental study.

**Mathematics Subject Classification.** 90C90, 90C59, 90C27.

Received March 8, 2019. Accepted March 18, 2021.

## 1. Introduction

The equitable distribution of development projects that ensures societal justice is important for every municipality and for every community. The good distribution of resources on municipalities makes the country more solidary and achieves the desired societal development. The main concern of this research is budget distribution. The problem arises when there are different projects to be distributed to several municipalities. The main objective is to search for a suitable and fair assignment of different projects to municipalities. A random distribution of these projects may lead to resources loss, unfair budget distribution and an undesired development results. These results may have economic and social impacts that are not preferable. This kind of problem can also be interesting for regional development. Given a set of projects, to be carried out in different cities or municipalities. Each project has its appropriate budget and will be entirely carried out in a chosen municipality. For the sake of equity, we apply some heuristics that can allocate different cities, so as to maximize the minimal total budget [9].

*Keywords.* Regional development, load balancing resources, equity distribution, heuristic, budgeting.

[1] Department of Computer Science and Information, College of Science at Zulfi, Majmaah University, Majmaah 11952, Saudi Arabia.

[2] MARS Laboratory, University of Sousse, Sousse, Tunisia.

[3] Department of Computer Science, Higher institute of computer Science and mathematics, University of Monastir, Monastir 5000, Tunisia.

[*]Corresponding author: m.jemmali@mu.edu.sa, mah_jem_2004@yahoo.fr

In this work, we assume that all municipalities are identical and have the same characteristics to allocate investments and with the same investment opportunities. Moreover, some of the projects are not applicable in other cities. This point of differentiation is very important for the distribution to be more realistic. For example, some projects cannot be assigned to some municipalities, nautical base projects will not be assigned to municipality of a non-coastal region. This kind of project has a regional characteristic. For this research, we consider only projects which can be assigned to any municipality and haven't any special related region characteristics. Heuristics and exact solutions will be developed to give an appropriate distribution of all projects to all municipalities.

Optimal allocation of resources, is developed with several activities that minimize the cost incurred by the allocation process. One constraint concerning the total amount of resources to be allocated, is to consider the minimization of the separable convex function. The problem can be viewed as a nonlinear programming problem or a nonlinear integer programming problem [15]. In [4] author presented a determination of optimal resource allocation for an invention which depends on the technological characteristics of the invention process and the nature of the market for knowledge.

Recently, authors in [1], developed lowers bounds, approximate solutions and an exact method for the problem investment project distribution on regions. The main objective is minimizing the maximum total number of newly created jobs.

The proposed and developed problem is cited for the first time in [11]. In the latter work, author proposed a probabilistic and randomized approximate solutions to the projects revenues assignment problem. Author in the latter research gives only algorithm to solve the proposed problem. No experimental results was presented in the latter work. However, author in [12], developed three heuristics for the same problem. Other domain to apply the equity distribution proposed in latter works is the gas turbines aircraft engines where authors in [13] proposed a mathematical modelling of the problem and two algorithms to solve approximately the problem. Other work related to gas turbines is developed in [14]. Author in [2] developed heuristics for the equity distribution of used space in storage supports.

The studied problem was solved by using or inspiring heuristics and algorithms from the main problem of $P||C_{\max}$ and $P||C_{\min}$. Heuristics and exact method developed in [8–10] can be largely exploited. In [8,10] authors developed a new efficient lower bounds and heuristics for the problem of minimizing makespan on identical parallel machines. A branch-and-bound algorithms were developed based on the lower and upper bounds yield a very effective exact algorithm. In addition, symmetry breaking branching strategy was presented. For the second problem, $P||C_{\min}$, an exact branch-and-bound algorithm using enhancement lower and upper bounds were developed in [9].

The $P||C_{\min}$ problem, also referred as machine covering was presented in [19] who proposed an exact branch-and-bound algorithm. Its most distinctive components are a different symmetry-breaking solution representation and enhanced lower and upper bounds.

This paper is structured as follows. In Section 2, we described the studied problem. While lower bound based on $P||C_{\min}$ will be explained in Section 3. Several heuristics based on dispatching rules, greedy algorithms, and complicated ones are described in Section 4. In Section 5, we develop an exact solution based on the branch and bound method. All developed algorithms will be discussed and assessed in the experimental results section.

## 2. Problem description and proprieties

The studied problem is described as follows. Let DP be a set of $n_{dp}$ development projects that will be distributed on $m_u$ different municipalities. Every project $p$ can be assigned to only one municipality $m$ with $m \in \{1, \ldots, m_u\}$. Each project $p$ with $p \in \{1, \ldots, n_{dp}\}$ is characterized by a predefined budget $b_p$. The cumulative budget will be denoted by $Cb_p$ for the municipality $m$ when project $p$ is assigned. The total number of assigned budgets to each municipality $m$ after finishing assignment process is denoted by $Tb_m$. The minimum (maximum) given budget after finishing distribution of all municipalities is denoted by $Tb_{\min}$ ($Tb_{\max}$). The budget of each

TABLE 1. Example of instance related to the studied problem.

| $p$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $b_p$ | 1320 | 5897 | 2354 | 3654 | 9632 | 3985 |

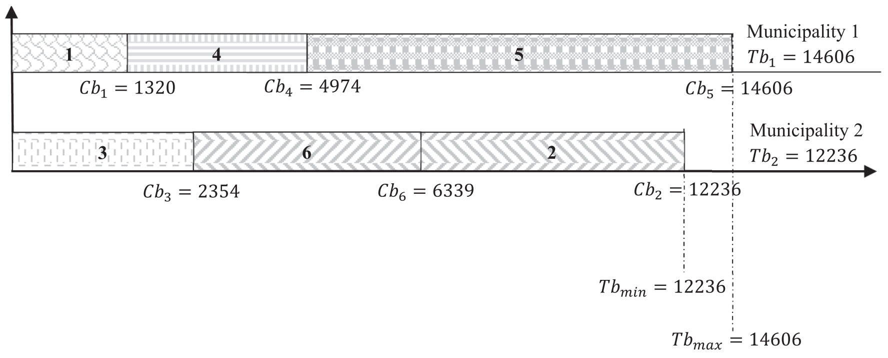

FIGURE 1. Budget municipality dispatching for example 1.

municipality is indexed as follows $Tb_1 \leq Tb_2 \leq \ldots \leq Tb_{m_u}$. The following example illustrates the studied problem.

**Example 2.1.** Let $n_{dp} = 6$ and $m_u = 2$. Table 1 represents the budget $b_p$ distribution for each project $p$.

We chose any algorithm to assign projects to municipalities. The chosen algorithm will give the schedule illustrated in Figure 1, which shows that municipality 1 has projects 1, 4, and 5, while municipality 2 has the projects 3, 6, and 2.

Based on Figure 1, municipality 1 has a total budget of 14 606. However, municipality 2 has a total budget of 12 236. The budget gap between municipality 1 and municipality 2 is equal to $Tb_1 - Tb_2 = 2370$. The objective here, is to reduce this gap. So, we need to find another schedule that is more efficient with a gap of less than 2370.

To evaluate the gap between municipalities, we can fix several indicators. For this work, the indicator that we propose for each municipality to reduce the budget gap will be $Tb_m - Tb_{\min}$. Therefore, considering the $m_u$ municipalities, the total budget gap which is denoted by $Bg_{\max}$ is given in equation (2.1).

$$Bg_{\max} = \sum_{m=1}^{m_u} \left[Tb_m - Tb_{\min}\right]. \tag{2.1}$$

The objective function for the studied problem is Minimize $Bg_{\max}$. Consequently, the objective function will be rewritten as

$$\text{Minimize } \left[\sum_{m=1}^{m_u} Tb_m - m_u Tb_{\min}\right]. \tag{2.2}$$

We denoted by $Bg^*_{\max}$ the optimal solution for the studied problem. Using the standard three-field notation of [17], this problem can be denoted as $P||Bg_{\max}$.

**Remark 2.2.** $Bg_{\max}$ can be written as: $Bg_{\max} = \sum_{p=1}^{n_{dp}} b_p - m_u Tb_{\min}$.

*Proof.* For any schedule we have: $\sum_{p=1}^{n_{dp}} b_p = \sum_{m=1}^{m_u} Tb_m$. Using this equality in equation (2.2), we prove remark 2.2. □

An important proposition of the studied problem is presented as follows.

**Proposition 2.3.** *$P||Bg_{\max}$ is equivalent to $P||C_{\min}$.*

*Proof.* Based on Remark 2.2, $Bg_{\max} = \sum_{p=1}^{n_{dp}} b_p - m_u Tb_{\min}$ , for the other side $\sum_{p=1}^{n_{dp}} b_p = \sum_{m=1}^{m_u} Tb_m$, then Minimize $Bg_{\max}$ is equivalent to Maximize $Tb_{\min}$. The latter problem is $P||C_{\min}$ □

**Corollary 2.4.** *$P||Bg_{\max}$ is NP-hard.*

*Proof.* According to the latter proposition $P||Bg_{\max}$ is equivalent to $P||C_{\min}$ which is shown to be NP-hard in [9]. □

## 2.1. Mixed Integer linear formulation

This subsection, presents a mathematical model that describes the studied problem. This mathematical model is based on the mixed-integer linear formulation. To derive the mixed-integer linear formulation, we introduce the binary variable $x_{pm}$ as follows.

$$x_{pm} : \begin{cases} 1 & \text{if project } p \text{ is assigned to municipality } m \\ 0 & \text{otherwise.} \end{cases}$$

The mixed integer formulation related to the studied problem is given as follows.

$$\text{Minimize} \left[\sum_{m=1}^{m_u} Tb_m - m_u Tb_{\min}\right] \tag{2.3}$$

Subject to:

$$\sum_{m=1}^{m_u} x_{pm} = 1, \qquad \forall p \in \{1,\cdots,n_{dp}\} \tag{2.4}$$

$$\sum_{p=1}^{n_{dp}} b_p x_{pm} \geq Tb_{\min}, \qquad \forall m \in \{1,\cdots,m_u\} \tag{2.5}$$

$$x_{pm} \in \{0,1\}, \qquad \forall p \in \{1,\cdots,n_{dp}\},\ \forall m \in \{1,\cdots,m_u\} \tag{2.6}$$

$$Tb_{\min} \geq 0. \tag{2.7}$$

The minimization of $Bg_{\max}$ is the objective function that is given by equation (2.3). While equation (2.4), is the constraint related to the obligation of the assignment of any project $m$ to exactly one municipality. The condition expressed in equation (2.5) is related to the constraint of the total assigned budget for each municipality. Whereas, the total assigned budget $Tb_m$ for a municipality $m$ is greater than or equal to $Tb_{\min}$ for all municipalities. Equation (2.6) is the constraint that declares $x_{pm}$ as a binary variable. Finally, equation (2.7) is the constraint that force the minimum total assigned budget for all municipalities to be positive.

## 3. Lower bounds

In this section, we present the developed lower bounds. These lower bounds are developed based on the upper bounds of the well-known problem $P||C_{\min}$. Indeed, from a given upper bound of $P||C_{\min}$ we develop a lower bound for the studied problem. Denoted by $\text{UB}_{cmin}$ a given upper bound for $P||C_{\min}$.

**Lemma 3.1.** *Given an instance $I$ of $P||Bg_{\max}$ and an upper bound $\text{UB}_{cmin}$ on the optimal makespan of the corresponding $P||C_{\min}$, then a valid lower bound on the optimal solution of the $P||Bg_{\max}$ instance is $\sum_{p=1}^{n_{dp}} b_p - m_u \text{UB}_{cmin}$.*

*Proof.* $\text{UB}_{cmin}$ is an upper bound for $P||C_{\min}$, so for any schedule we have: $Tb_{\min} \leq \text{UB}_{cmin}$. Multiplying by $m_u$, we have $m_u Tb_{\min} \leq m_u \text{UB}_{cmin}$. This is can be written as $-m_u Tb_{\min} \geq -m_u \text{UB}_{cmin}$. Now, adding $\sum_{p=1}^{n_{dp}} b_p$ we obtain the lower bound proposed in Lemma 3.1. □

For our study, we use the upper bounds $U_0$ and $U_3^*$ for $P||C_{\min}$ described in [9] to develop respectively $L_1$ and $L_2$ as lower bounds for our studied problem.

## 4. Heuristics

In this section, we present eleven developed heuristics based on different resolution categories. The first heuristic category is developed using the dispatching rules. The second one is the randomized method using two variants. Utilizing the lower bounds for the $P||C_{\min}$ problem with two variants is the third category. For the fourth category, we use the well-known multi-fit problem. On the other hand, the subset-sum problem is utilized to determine category 5, and the knapsack problem is imbricated to develop category 6.

### 4.1. Dispatching rules based-heuristics

These heuristics are based on the order given before scheduling projects to municipalities. We have to choose if we start with a non-decreasing order or a non-increasing order. The selection of the municipality will be the first municipality having the least total budget.

#### *4.1.1. Non-decreasing budget order heuristic (NDB)*

For this dispatching, all projects are arranged in a non-decreasing order based on projects budget. After that, we assign the project having the smallest budget to the municipality with the smallest total budget.

#### *4.1.2. Non-increasing budget order heuristic (NIB)*

In contrary to NDB, order all projects are arranged in a non-increasing order of its budgets. After that, we assign the project having the greatest budget to the municipality which has the smallest total budget.

### 4.2. Randomized based heuristics

For this category of heuristics, this study is based on assigning the project with the largest budget to the municipality which has the minimum budget total with some probabilistic method. The first probabilistic is to choose between the first largest budget project or the second largest budget project. However, the second probabilistic is to choose between the largest budget project or a set of fixed number of large budget projects.

#### *4.2.1. α-Randomized budget heuristic ($\text{RB}_\alpha$)*

The first step to apply this heuristic is to put all projects in non-increasing budget order. This heuristic is described as follows. Instead of selecting the project with the greatest budget value, we select one of the two projects with the greatest budget and with respective probability $\alpha$ (randomly and uniformly generated in [0,1]) and $1-\alpha$. Now, the question is how to select between the two largest remaining budgets. For that aim, an experimental study is conducted. For which parameter $\gamma$ is fixed. This $\gamma$ represents the bound for the selection between the two largest budgets. For example, if we fix $\gamma = 0.1$ this means if $\alpha$ is $\leq 0.1$ then the next selected budget is the largest one. Otherwise, if $\gamma > 0.1$ the selected budget is the second one. The choice of $\gamma$ can affect the performance of results. For this reason, we lead a study to choose the best value of $\gamma$. We fix $\gamma = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. For each value of $\gamma$, we ran the algorithm and note the corresponding heuristic value. A comparative study is made for 1700 instances with $n_{dp} = \{10, 20, 50, 100, 300, 500\}$. This heuristic consists of running the randomized-based heuristic more than once. The number of iterations used to calculate the corresponding value is 1000. Thus, the heuristic is repeated 1000 times and we choose the best value among these iterations. For each instance, we ran the algorithm with the given 9 values of $\gamma$ and determine the minimum value of $V_{\min}$. The number of instances that are equal to $V_{\min}$ is denoted by $nb$, and denoted by $Per$ the percentage among 1700 instances. Table 2 gives all percentages for each $\gamma$ value.

TABLE 2. Behavior of heuristic values according to $\gamma$.

| $\gamma$ | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| $nb$ | 1045 | 1074 | 1126 | 1155 | 1102 | 1097 | 1051 | 988 | 884 |
| $Per$ | 61.47 | 63.18 | 66.24 | 67.94 | 64.82 | 64.53 | 61.82 | 58.12 | 52.00 |

TABLE 3. Behavior of lonely participation in the minimum heuristic values according to $\gamma$.

| $\gamma$ | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 | 0.9 |
|---|---|---|---|---|---|---|---|---|---|
| $nb_{lo}$ | 216 | 33 | 32 | 27 | 15 | 18 | 11 | 9 | 0 |
| $Per$ | 12.71 | 1.94 | 1.88 | 1.59 | 0.88 | 1.06 | 0.65 | 0.53 | 0.00 |

TABLE 4. Selecting appropriate number of iterations.

| Iteration | 50 | 100 | 250 | 500 | 1000 |
|---|---|---|---|---|---|
| Best | 300 | 365 | 425 | 475 | 516 |
| Time | – | – | – | – | 0.01 |

Referred to Table 2, the first greatest percentage and the second one are obtained for $\gamma = 0.4$ and for $\gamma = 0.3$ respectively.

Another type of study is based on the lonely participation in $V_{\min}$. Which means to determine the number of instances ($nb_{lo}$) for each $\gamma$. The number $nb_{lo}$ is the number of instances when the heuristic value equals lonely to $V_{\min}$ (all others values number of elated to other values of $\gamma$ are not equal to $V_{\min}$). Table 3 presents the results given by each $\gamma$ values.

Based on Table 3, it is clearly that the maximum percentage is 12.71% and obtained for $\gamma = 0.1$. From Tables 2 and 3, we choose finally three $\gamma$ values as follows $\gamma = 0.1$, $\gamma = 0.3$ and $\gamma = 0.4$. For each value of $\gamma$ we calculate the heuristic value and after finishing all values, we pick the minimum one.

The 1000 iteration is selected based on the following experimental study. For this experimental study $n_{dp} = \{10, 20, 50\}$. We experimented the number of iteration equivalent to $\{50, 100, 250, 500, 1000\}$. We ran the heuristic $\mathrm{RB}_\alpha$ for all iteration values and we calculate the minimum value $\mathrm{Min}_\alpha$ for each instance. We denoted by *Best* the number of instances when $\mathrm{RB}_\alpha$ is equal to $\mathrm{Min}_\alpha$. Table 4 show the choice of the number of iteration. Based on the Table 4, 1000 iteration provides maximum number of best solutions. We stopped at 1000 because of the needed extra time.

### *4.2.2. $\beta$-Randomized budget heuristic ($\mathrm{RB}_\beta$)*

The first step in applying this heuristic is to put all projects in non-increasing budget order. For this heuristic, an adaptation of the NIB and $\mathrm{RB}_\alpha$ heuristics are performed. This adaptation is based on extending the selection interval of the candidate projects. In $\mathrm{RB}_\alpha$, the selection of the candidate project to be scheduled is between the two first unscheduled projects. Now, we extend the interval of selection from only two projects to reach $n_{dp}$. The obtained heuristic is denoted by $\mathrm{RB}_\beta$. The performed experiment was carried out over a data set of 1700 instances. Let $\mathrm{RB}_\beta^{\mathrm{Int}}$ be the obtained upper bound, if the random selection for NIB is applied to the Int first unscheduled projects ($2 \leq \mathrm{Int} \leq n_{dp}$). Let Perc to be the percentage of instances where $\mathrm{RB}_\beta^{\mathrm{Int}}$ is equal to the minimum of $\mathrm{RB}_\beta^{t}$, with $1 \leq t \leq \mathrm{Int}$. The set of selected projects will be chosen among Int first projects that have the largest budgets with a probability $\beta$ (uniformly). This probability is chosen as follows.

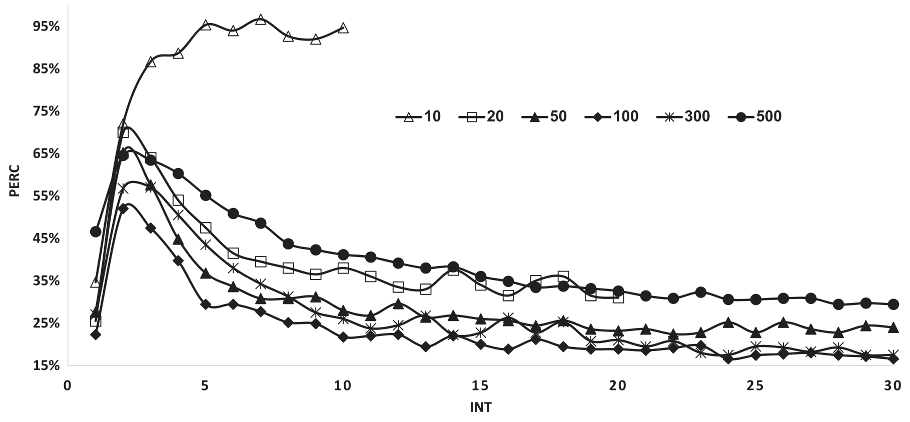

FIGURE 2. Experimental determination of the best interval.

TABLE 5. Percentage Perc(Int) for each interval Int and each $n_{dp}$ of RB$_\beta$ heuristic.

| $n_{dp}$ | Int | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 10 | 34.7% | 72.0% | 86.7% | 88.7% | 95.3% | 94.0% | 96.7% | 92.7% | 92.0% | 94.7% |
| 20 | 25.5% | 70.0% | 64.0% | 54.0% | 47.5% | 41.5% | 39.5% | 38.0% | 36.5% | 38.0% |
| 50 | 27.6% | 65.2% | 57.6% | 44.8% | 36.8% | 33.6% | 30.8% | 30.8% | 31.2% | 28.0% |
| 100 | 22.3% | 52.0% | 47.4% | 39.7% | 29.4% | 29.4% | 27.7% | 25.1% | 24.9% | 21.7% |
| 300 | 27.0% | 56.8% | 57.0% | 50.5% | 43.5% | 38.0% | 34.3% | 31.3% | 27.5% | 26.0% |
| 500 | 46.6% | 64.6% | 63.4% | 60.3% | 55.1% | 50.9% | 48.6% | 43.7% | 42.3% | 41.1% |

- We chose randomly a number $r$ between $[1, \text{Int}]$. The selected number will be the $r$th largest unscheduled project index which will be assigned to the municipality having the lower total budget.
- In case when the number of unscheduled projects $\text{UN}_p$ is less than Int, the randomly $r$ will be between $[1, \text{UN}_p]$.

An experimental study is carried out and a summary of the obtained results is displayed in Figure 2. According to the results shown in Figure 2, it is observed that a maximum is reached at Int $= 2$ or at Int $= 3$, while there still some residual percent for Int $= \{3, 4, 5, \ldots, 10\}$. For this reason, stopping at 5 intervals is worthy.

The details of percentage values according to Int and $n_{dp}$ is shown in Table 5. For each fixed Int, we repeat the heuristic 1000 times. The minimum value will be selected over 1000 iterations for the 5 intervals.

## 4.3. $C_{\min}$ based heuristics

These heuristics are developed based on the lowers bounds of the well-known problem $P||C_{\min}$. Indeed, from a given lower bound of $P||C_{\min}$, we develop an upper bound for the studied problem. Denoted by $\text{LB}_{cmin}$ a given lower bound for $P||C_{\min}$.

**Lemma 4.1.** *Given an instance $I$ of $P||Bg_{\max}$ and a valid lower bound $\text{LB}_{cmin}$ on the optimal makespan of the corresponding $P||C_{\min}$, then a valid upper bound on the optimal solution of the $P||Bg_{\max}$ instance is $\sum_{p=1}^{n_{dp}} b_p - m_u \text{LB}_{cmin}$*

*Proof.* $\text{LB}_{cmin}$ is a lower bound for $P||C_{\min}$. So for any schedule, we have: $Tb_{\min} \geq \text{LB}_{cmin}$. Multiplying by $m_u$, we have $m_u Tb_{\min} \geq m\text{LB}_{cmin}$. This is can be written as $-m_u Tb_{\min} \leq -m_u \text{LB}_{cmin}$. Now, adding $\sum_{p=1}^{n_{dp}} b_p$ we obtain the upper bound proposed in Lemma 4.1. □

TABLE 6. Instance of budgets for heuristic BMF.

| $p$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| $b_p$ | 55 | 59 | 94 | 30 | 51 | 95 | 43 | 44 | 100 | 70 |

For our study, we use the lower bounds described for $P||C_{\min}$ in [9] to develop upper bounds for our studied problem. We denoted by $U^1_{cmin}$ and $U^2_{cmin}$ upper bounds derived from the subset-sum lower bound and the heuristic algorithm based lower bound, respectively.

## 4.4. Budgeting multi-fit based heuristic (BMF)

This heuristic is based essentially on searching for the minimum capacity (minimum total budget) such that all $n_{dp}$ projects will be assigned to all $m_u$ municipalities. This search is based on the utilization of the bin-packing algorithm (BPA) [3,7]. The municipality will be presented as a bin, while projects will be presented as items in the (BPA). For each fixed bin capacity, the First Fit Decreasing (FFD) method is used to fit projects to the bin [5,21]. Before running the algorithm we must order projects according to their budget such that $b_1 \geq b_2 \geq \dots b_{n_{dp}}$. The FFD method assigns the projects in succession to the lowest indexed municipality which can contain the project (regarding budgeting capacity) within the capacity. Let $\mathrm{LB}_{\max} = \max\left(b_1, b_{mu} + b_{mu+1}, \left\lceil \frac{\sum_{p=1}^{n_{dp}} b_p}{m_u} \right\rceil\right)$ and $\mathrm{UB}_{\max}$ is the value given by applying the LPT heuristic for $P||C_{\max}$. We fix an iterative number *ite* which is the number of iteration of FFD. We set $ite = 135$ and denote by $n_{bin}$ the number of bins used by applying FFD. Algorithm 1 describes all steps to calculate BMF.

**Algorithm 1.** Budgeting multi-fit algorithm BMF.

```
1: Set $I = 0$, $u_p = \mathrm{UB}_{\max}$ and $l_o = \mathrm{LB}_{\max}$.
2: Set mid $= \lfloor \frac{u_p + l_o}{2} \rfloor$, set $I = I + 1$.
3: Apply FFD with capacity $C$.
4: if we can assign all projects $n_{dp}$ into $m_u$ municipalities, then set $u_p = C$ and goto 5, otherwise set $l_o = C$ and goto 5.
5: if ($I = ite$) then
6:     STOP.
7: else
8:     Goto 2.
9: end if
10: if ($n_{bin} > m_u$) then
11:     The schedule given by NIB is taken.
12: else
13:     The schedule given by FFD is taken. The taken schedule noted by $\sigma$.
14: end if
15: Determine the $C_{\min}$ of schedule $\sigma$.
16: BMF $= \sum_{p=1}^{n_{dp}} b_p - m_u C_{\min}$.
17: Return BMF.
```

**Example 4.2.** Let $n_{dp} = 10$ and $m_u = 2$. Table 6 represents the budget of each project.

Considering the $P||C_{\max}$ problem, LPT heuristic give the upper bound value $\mathrm{UB}_{\max} = 323$ and the lower bound is $\mathrm{LB}_{\max} = 318$. Then, mid $= \lfloor \frac{323+318}{2} \rfloor = 322$. Applying FFD function with capacity 322 and the 10 items which is represented by projects in Table 6. The first municipality has projected $\{4, 3, 6, 9\}$ and the second municipality will have the projects $\{1, 2, 5, 7, 8, 10\}$. The first municipality) has a total budget of 319 however, the second one has a total budget of 322. Thus, $Bg_{\max} = 641 - 2 \times 319 = 3$. On the other hand, the $C_{\min}$

obtained by LPT is 318 which is meaning the NIB heuristic gives $Bg_{\max} = 5$. So, the result obtained by BMF is better than NIB.

## 4.5. Subset-sum based heuristics

This category of heuristic is based essentially on solving iteratively of a set of subset-sum problems.

### *4.5.1. Repeating of solving subset-sum problems heuristic (RSS)*

A greedy algorithm is used to develop this heuristic. This heuristic is based on solving iteratively several subset-sum problems (SSP) [20]. We denoted these problems by $(Ps)_k$ for $k = \{1, 2, \dots, m_u - 1\}$ as follows:

$$(Ps)_k : \begin{cases} \min \sum_{p \in S_k} b_p y_p \\ \text{subject to} \sum_{\mathrm{DP}_p \in S_k} b_p y_p \geq \mathrm{LB}(S_k, m_u - k + 1) \end{cases}$$

with $y_p \in \{0, 1\}$ for all $\mathrm{DP}_p \in S_k$. Where $S_1 = \mathrm{DP}$ and $S_{k+1} = S_k \setminus \mathrm{DP}_k$ where $\mathrm{DP}_k$ is an optimal subset sum for $Ps_k$ and $k = 1, 2, \cdots, m_u - 1$. $\mathrm{LB}(S, K)$ denote a valid lower bound on the makespan of a reduced instance defined on $k \leq m_u$ machines and a subset of jobs $S \subset \mathrm{DP}$.

Therefore, for the first municipality, we assign projects until reaching LB on $Ps_1$. The remaining projects with remaining municipalities will constitute the second problem $Ps_2$ to solve the new SSP until reaching LB and so on [10]. A pseudo-polynomial is used to solve a subset sum problem using the dynamic programming algorithm developed by [20].

After ending assignment of all projects, a $C_{\min}$ value will be calculated and we conclude the $Bg_{\max}$ value.

### *4.5.2. Most-least-repeating of solving subset-sum problems heuristic (MLRS)*

For the sake of clarity, we briefly recall the basic ideas derived from this heuristic. As shown in [10] the $P_2\|C_{\max}$ could be reformulated as a subset-sum problem. Based in this proposition a repeating local search method will be implemented, that requires repetitively solving a problem of two-municipalities. Given a feasible schedule $\sigma$ and assume that the municipalities are indexed such that $Tb_1 \leq Tb_2 \leq \cdots \leq Tb_{m_u}$. The most-least-repeating subset-sum (MLRS) heuristic requires the selection of two municipalities $m_{m_u}$ and $m_k$ and solving $P_2\|C_{\max}$ defined on $\mathrm{DP} = \mathrm{DP}_1 \cup \mathrm{DP}_k$. The number of iteration is fixed to $iter = 500$.

We define a binary variable $y_p$ that takes value 1 if project $\mathrm{DP}_p$ is assigned to $m_1$, and 0 otherwise. Then, $P_2\|C_{\max}$ applying the problem SSP will be solved interactively. The most-least-repeating subset-sum heuristic (MLRS) is described below:

After the termination assignment of all projects, a $C_{\min}$ value will be calculated and we conclude the $Bg_{\max}$ value.

## 4.6. Knapsack based heuristics

This category of heuristic is based essentially on solving iteratively of a set of knapsack problems [16] and [18].

### *4.6.1. Repeating of solving knapsack problems heuristic (RSK)*

This heuristic is based on the division of the problem in $m_u$ problems. Each problem searches the scheduling of the projects to municipalities. For the first municipality, we apply a knapsack problem to assign projects to municipality 1. After that, the remaining projects will be assigned to the remaining municipalities by applying a knapsack problem, and so on. So, for each municipality, a knapsack problem must be solved as follows.

$$(\mathrm{KN})_k : \begin{cases} \min \sum_{p \in Q_k} w_p y_p \\ \text{subject to} \sum_{p \in Q_k} b_p y_p \leq \left\lfloor \frac{\sum_{p \in Q_k} b_p y_p}{k} \right\rfloor \end{cases}$$

**Algorithm 2.** Most-least-repeating subset-sum heuristic MLRS.

```
1: Generate a starting feasible solution $\sigma$ and set $iter = 1$.
2: Compute $Tb_k$ for $k \in \{1, 2, \cdots, m_u\}$. Set $k = 1$.
3: Solve the $P2||C_{\max}$ instance that is defined by $m_k$ and $m_{m_u}$, denoted by $C^k_{\max}$ the optional makespan that is obtained
   after solving the corresponding subset-sum problems.
4: if ($C^k_{\max} < Tb_{m_u}$) then
5:     Update $\sigma$ and goto 2.
6: end if
7: if ($k < m_u - 1$) then
8:     Set $k = k + 1$ and goto 3.
9: end if
10: Output the best-found solution for $C_{\min}$.
11: Calculate $Bg_{\max}$.
12: if ($iter \leq 500$) then
13:     Goto 2
14: else
15:     STOP.
16: end if
```

where:

- $k$ is the index of the municipality.
- $F_1 = \text{DP}$ and $F_{k+1} = F_k \setminus Op_k$ where $Op_k$ is the set containing projects after solving $(\text{KN})_k$.
- $w_p = \frac{|F_k|}{k} \times b_p - 1$, where $|F_k|$ is the number of remaining projects and $k$ the remaining municipalities.

For the first municipality, we assign projects until reaching $\left\lfloor \frac{\sum_{p \in Q_k} b_p y_p}{m_u} \right\rfloor$ on $(\text{KN})_1$. The remaining projects with remaining municipalities constitute the second knapsack problem $(\text{KN})_2$ to solve the new knapsack problem until reaching $\left\lfloor \frac{\sum_{p \in Q_k} b_p y_p}{m_u - 1} \right\rfloor$ and so on.

After the termination assignment of all projects, a $C_{\min}$ value will be calculated and we conclude the $Bg_{\max}$ value.

### *4.6.2. Most-least-repeating of solving knapsack problems heuristic (MLRK)*

The $P_2||C_{\max}$ could be reformulated as a knapsack problem. Based on this proposition, a most-least-repeating local search method will be implemented, which requires repetitively solving a problem of two-municipalities. The MLRK heuristic is in the same vein as MLRS. The difference is localized in the problem solved in each iteration. Indeed, for MLRK, in each iteration, a knapsack problem (KP) is solved instead of a SSP. It is well known that KP could be efficiently solved in pseudo-polynomial time.

## 5. Exact solution

We develop a branch-and-bound method, to find the optimal solution for the studied problem. Before developing the branch and bound it is important to give the relation between the optimal solution for the $P||C_{\min}$ and the optimal solution for the $P||Bg_{\max}$. The following corollary explain this relation.

**Corollary 5.1.** Minimize $[\sum_{m=1}^{m_u} Tb_m - m_u Tb_{\min}]$ *is equivalent to* Maximize $Tb_{\min}$.

*Proof.* As described in Remark 2.2, the objective function of the studied problem can be written as $Bg_{\max} = \sum_{p=1}^{n_{dp}} b_p - m_u Tb_{\min}$. The number of municipalities and the summation of all budgets $\sum_{p=1}^{n_{dp}} b_p$ are determined in advance and fixed. This is meaning Minimize $[\sum_{m=1}^{m_u} Tb_m - m_u Tb_{\min}]$ is equivalent to Minimize $-m_u Tb_{\min}$. Therefore, we have to Maximize $Tb_{\min}$. □

**Lemma 5.2.** *The optimal solution of the studied problem is* $\sum_{m=1}^{m_u} Tb_m - m_u Tb^*_{\min}$.

*Proof.* The solution given to Maximize $Tb_{\min}$ is $Tb^*_{\min}$. Based on Corollary 5.1, we deduce the optimal solution given in Lemma 5.2 □

For this paper and based on Lemma 5.2, we developed a branch and bound algorithm to solve $P||C_{\min}$ exactly. Then we use the found solution to obtain the exact solution to our budgeting problem.

Before we present the exact solution algorithm, we must define several notations. Let $I$ be an instance. We denoted by $U(I)$ and $L(I)$ the best upper bound and the best lower bound for the studied problem, respectively. We denoted by $U_m(I)$ and $L_m(I)$ the best upper bound and the best lower bound for the $P||C_{\min}$ problem, respectively. Dominance() is a function that test the dominance rules of $P||C_{\min}$ related to the instance given as input as cited in [9]. Algorithm 3 gives instructions to calculate the optimal solution for the studied problem.

**Algorithm 3.** Exact solution algorithm.

```
1: Calculate $U(I)$ and $L(I)$.
2: if ($U(I) = L(I)$) then
3:     Return $U(I)$
4: else
5:     Calculate $U_m(I)$ and $L_m(I)$.
6:     Store $L_m(I)$ the best solution found.
7:     Initialize a queue to hold a partial solution with none of the variables of
8:     the problem assigned.
9:     while (The is NOT empty) do
10:        Take a node $N$ off the queue.
11:        if ($N$ is a single candidate solution $\phi$ and $C_{\min}(\phi) > L_m(I)$) then
12:            $C_{\min}(\phi)$ is the best solution so far.
13:            Record the best solution and set $L_m(I) = C_{\min}(\phi)$.
14:        else
15:            Branch on $N$ to produce new node $N_i$ for each of these:
16:            if ($L_m(N_i) > U_m(N_i)$ OR Dominance($N_i$) = True) then
17:                Discard node
18:            else
19:                Store $N_i$ on the queue.
20:            end if
21:        end if
22:    end while
23:    Return $\sum_{m=1}^{m_u} Tb_m - m_u L_m(I)$
24: end if
```

## 6. Experimental results

In this section, we present the experimental results found after the execution of our algorithms. In order to assess the performance of the proposed algorithms, we coded them in Microsoft Visual C++ (Version 2013). All our experiments were obtained on an Intel(R) Xeon(R) CPU E5-2687W v4 @ 3.00 GHz and 64 GB RAM. The operating system used is Windows 10 with 64 bits. The proposed procedures are tested on a set of test problems that are displayed in the following subsection.

### 6.1. Test problems

Several types of instances are generated in this work. We tested results on a set of instances that was inspired as described in [6]. The project budget $b_p$ is generated according to different probability distributions. Each one

TABLE 7. Generation of $(n_{dp}, m_u)$.

| $n_{dp}$ | $m_u$ |
|---|---|
| 10 | $2, 3, 5$ |
| 20 | $2, 3, 5, 10$ |
| 50 | $2, 3, 5, 10, 25$ |
| 100 | $2, 3, 5, 10, 15, 25, 50$ |
| 300 | $2, 3, 5, 10, 15, 25, 50, 100$ |
| $500, 1000, 1500$ | $2, 10, 25, 50, 100, 250, 300$ |

of the probability distributions represents a class. The classes are:

- Class 1: $b_p$ is generated from the discrete uniform distribution $U[30, 100]$.
- Class 2: $b_p$ is generated from the discrete uniform distribution $U[50, 300]$.
- Class 3: $b_p$ is generated from the discrete uniform distribution $U[200, 500]$.
- Class 4: $b_p$ is generated from the normal distribution $N[50 - 150]$.
- Class 5: $b_p$ is generated from the normal distribution $N[25 - 500]$.

The overall instances is based on the choice of $n_{dp}$, $m_u$ and Class. The choice of the pair $(n_{dp}, m_u)$ is given in Table 7.

For each fixed triple $(n_{dp}, m_u, \text{Class})$, we generate 10 instances of budget project. Based on the choice of $(n_{dp}, m_u)$ referred to Table 7, the total number of instances is 2400.

We denoted by:

- UB the best (minimum) value obtained after the execution of all heuristics.
- $U$ the studied heuristic.
- Min the number of instances when the studied heuristic is equal to UB.
- $G_u = \frac{U-\text{UB}}{U}$.
- AG$_u$ is the average of $G_u$ for a fixed number of instances.
- LB the best (maximum) value obtained after the execution of all lower bounds.
- $L$ the studied lower bound.
- Max the number of instances when the studied lower bound is equal to LB.
- $G_l = \frac{L-\text{LB}}{L}$.
- AG$_l$ is the average of $G_l$ for a fixed number of instances.
- $G_l^u = \frac{\text{UB}-\text{LB}}{\text{LB}}$.
- AG$_l^u$ is the average of $G_l^u$ for a fixed number of instances.
- NN is the average number of nodes created in the branch and bound algorithm.
- US is the sum of instances till unsolved by the branch and bound algorithm.
- Time the time spent to execute heuristic in corresponding instances. This time will be in seconds and we denote by "–" if the time is less than 0.001 s.

## 6.2. Performance assessment

Obtained results in this paper will be analyzed based on several indicators. It is important to show the behavior of the gap according to $n_{dp}$, $m_u$, and Class. The results found in this research is very impressive. This can be seen through the general viewing of the results. The average total gap between Min and Max for the whole 2400 instances is 2.62 for average Time 51.76 s. These impressive results show the performance of heuristics and lower bounds. In line with this result, the number of times when the best lower bound value is equal to the best upper bound value was 2045 out of 2400 instances. Which represents 85.21% of the overall instances. In the other side the optimal solution using the branch-and-bound was successful only for 14.79%.

TABLE 8. Behavior of $AG_l$ and Time according to $n_{dp}$.

| $n_{dp}$ | $L_1$ | | $L_2$ | | Total | |
|---|---|---|---|---|---|---|
| | $AG_l$ | Time | $AG_l$ | Time | $AG_l$ | Time |
| 10 | 18.47 | – | 0.00 | – | 9.24 | – |
| 20 | 17.53 | – | 0.00 | – | 8.77 | – |
| 50 | 0.42 | – | 0.00 | – | 0.21 | – |
| 100 | 22.22 | – | 0.00 | 0.016 | 11.11 | 0.008 |
| 300 | 0.00 | – | 0.00 | 0.127 | 0.00 | 0.064 |
| 500 | 57.43 | – | 0.00 | 3.977 | 28.72 | 1.989 |
| 1000 | 16.76 | – | 0.00 | 4.054 | 8.38 | 2.027 |
| 1500 | 0.00 | – | 0.00 | 3.887 | 0.00 | 1.944 |

TABLE 9. Behavior of $AG_l$ and Time according to $m_u$.

| $m_u$ | $L_1$ | | $L_2$ | | Total | |
|---|---|---|---|---|---|---|
| | $AG_l$ | Time | $AG_l$ | Time | $AG_l$ | Time |
| 2 | 0.06 | – | 0.00 | 0.002 | 0.03 | 0.001 |
| 3 | 20.59 | – | 0.00 | 0.001 | 10.30 | 0.001 |
| 5 | 2.88 | – | 0.00 | 0.002 | 1.44 | 0.001 |
| 10 | 1.10 | – | 0.00 | 0.019 | 0.55 | 0.010 |
| 15 | 76.13 | – | 0.00 | 0.021 | 38.07 | 0.011 |
| 25 | 0.35 | – | 0.00 | 0.068 | 0.18 | 0.034 |
| 50 | 0.66 | – | 0.00 | 0.248 | 0.33 | 0.124 |
| 100 | 0.25 | – | 0.00 | 1.038 | 0.13 | 0.519 |
| 250 | 0.89 | – | 0.00 | 11.528 | 0.44 | 5.764 |
| 300 | 171.90 | – | 0.00 | 14.660 | 85.95 | 7.330 |

A comparison study between lower bounds is represented in Tables 8–11 and A.1. Table 8 presents the variation of $AG_l$ and Time according to $n_{dp}$. This table shows that, in 100% of cases $L_2$ is the better lower bound since $AG_l = 0$ for all values of $n_{dp}$. The maximum $AG_l$ value is obtained for $n_{dp} = 500$ with value 57.43 for $L_1$.

Table 9 presents the behavior of $AG_l$ and Time according to $m_u$. From this table we can't deduce any order of $AG_l$ according to $m_u$ values. Indeed, for $L_1$, when $m_u = 3$ the average gap is equal to 20.59, for $m_u = 10$ is equal to 1.10 and for $m_u = 15$ the average gap once again goes up to 76.13. The maximum $AG_l$ value is reached 171.90 for $L_1$ when $m_u = 300$.

Table 10 represents the difference levels of the problem-hardness according to classes. The given results shows that for the Class 5, the $AG_l$ has the largest value reaching 57.99 for $L_1$. This class is harder for $L_1$ comparing with Class 4 which the corresponding $AG_l$ is equal to 1.66. On the other hand the Time is increasing when the Class index increases.

We observe from Table 11 that the best lower bound is $L_2$ which has the maximum percentage of 100% with average gap zero and execution average time of 1.76s. However, the minimum percentage is 91.1% for $L_1$ with an average gap of 16.72 and Time less than 0.001s. The lower bound $L_1$ is faster than $L_2$. More details is presented in appendix in Table A.1. From this table we observe that the maximum average gap for the lower bound $L_1$ is 398.37 reached when $n_{dp} = 500$ and $m_u = 300$.

The performance of the studied heuristics can be shown by the behavior of $AG_u$ according to $n_{dp}$, $m_u$, and Class. An overview of heuristics comparison is given in Table 12. This table shows that the best *perc* value which represents the percentage of $Min$ among all 2400 instances is obtained for $U^2_{cmin}$ with 99.8% in an average time of 5.847 s. However, the minimum percentage is 14.4% and obtained for NDB heuristic in an average time less

TABLE 10. Behavior of $AG_l$ and Time according to Class.

| Class | $L_1$ | | $L_2$ | | Total | |
|---|---|---|---|---|---|---|
| | $AG_l$ | Time | $AG_l$ | Time | $AG_l$ | Time |
| 1 | 2.16 | – | 0.00 | 1.004 | 1.08 | 0.502 |
| 2 | 3.27 | – | 0.00 | 1.071 | 1.64 | 0.536 |
| 3 | 18.52 | – | 0.00 | 1.724 | 9.26 | 0.862 |
| 4 | 1.66 | – | 0.00 | 2.274 | 0.83 | 1.137 |
| 5 | 57.99 | – | 0.00 | 2.736 | 28.99 | 1.368 |

TABLE 11. Recapitulating comparison between lower bounds.

| | $L_1$ | $L_2$ |
|---|---|---|
| Max | 2187 | 2400 |
| Perc | 91.1% | 100.0% |
| $AG_l$ | 16.72 | 0.00 |
| Time | – | 1.76 |

TABLE 12. Heuristics comparison.

| | NIB | $U^1_{cmin}$ | $U^2_{cmin}$ | NDB | BMF | RSS | RSK | MLRS | MLRK | $RB_\alpha$ | $RB_\beta$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Min | 910 | 1866 | **2395** | 345 | 859 | 1755 | 1866 | 2112 | 2114 | 1159 | 1241 |
| *perc* | 37.9% | 77.8% | **99.8%** | 14.4% | 35.8% | 73.1% | 77.8% | 88.0% | 88.1% | 48.3% | 51.7% |
| $AG_u$ | 51.05 | 12.41 | **0.05** | 255.39 | 247.73 | 8.43 | 12.41 | 0.82 | 0.86 | 38.87 | 38.18 |
| Time | 0.000 | 0.025 | **5.847** | 0.000 | 0.078 | 0.023 | 0.031 | 53.406 | 330.302 | 0.219 | 0.300 |

than 0.001s and an average gap of 255.39 s. It is worthy to note that, the most time-consuming heuristic is MLRK which has an average time of 330.302 s.

The average gap of overall heuristics for each $n_{dp}$ is shown in Figure 3.

Based on Table 13, it is easy to observe that the heuristic $U^2_{cmin}$ has almost a zero-gap for all values of $m_u$. However, heuristics BMF and NDB have for most of the cases the largest gap comparing with the rest of heuristics. The heuristic MLRK consumes more time than others reaching to 2603.30 s for $m_u = 300$. Table 13 shows that there is no dominance between heuristics. For more details of $AG^u$ and Time for all studied heuristics, Table A.2 is given in the appendix.

Now our experimental study is focused on the gap between Min and Max. The noted time is the average time between lower bounds and heuristics. In Table 14, the maximum gap is 6.18 and obtained for $n_{dp} = 500$ in 19.428 s. It is important to note that for $n_{dp} = 1500$ the gap is 0 and the corresponding time is less than the corresponding time related to $n_{dp} = 500$.

The behavior of $AG^u_l$ according to $n_{dp}$ is depicted in Figure 4.

The behavior of $AG^u_l$ according to $m_u$ is shown in Table 15. It is clear to observe that zero-gap is obtained for $m_u = 2$ in 0.236 s. However, the maximum gap of 14.21 is obtained for $m_u = 250$ in 50.960 s.

From Table 16, we conclude that classes 2, 3 and 4 are slightly difficult than others with a gap around 3. On the other hand, the Class 5 is more time-consuming with an average time of 42.842 s for the all instances. For more details of $AG^u_l$, Table A.3 is given in the appendix.

Now, we show the results of the optimal solution produced by the developed branch-and-bound method. The results show that there are only 282 instances among 2400 still unsolved. Table 17, presents the NS, NN

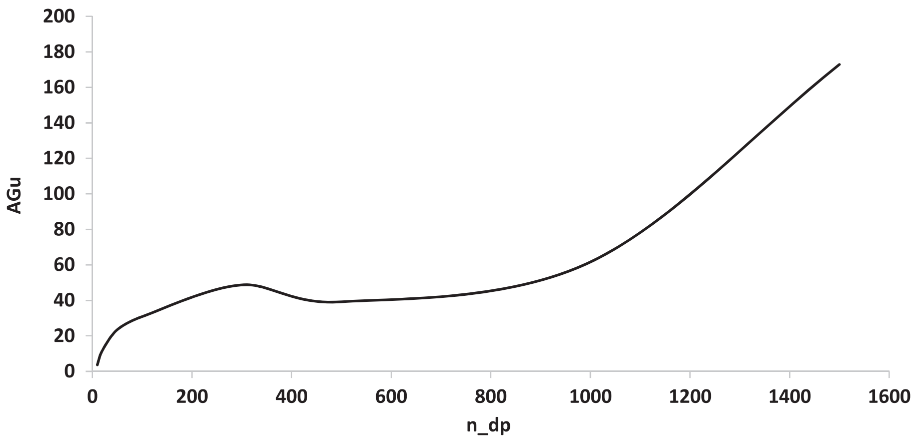

FIGURE 3. The average gap of overall heuristics for each $n_{dp}$.

TABLE 13. Behavior of $\mathrm{AG}_u$ and Time according to $m_u$.

| $m_u$ | NIB | | $U^1_{cmin}$ | | $U^2_{cmin}$ | | NDB | | BMF | | RSS | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\mathrm{AG}_u$ | Time | $\mathrm{AG}_u$ | Time | $\mathrm{AG}_u$ | Time | $\mathrm{AG}_u$ | Time | $\mathrm{AG}_u$ | Time | $\mathrm{AG}_u$ | Time |
| 2 | 2.94 | – | 0.00 | – | 0.00 | 0.05 | 46.38 | – | 1.74 | 0.02 | 1.01 | – |
| 3 | 67.52 | – | 0.17 | – | 0.00 | 0.11 | 149.12 | – | 54.10 | 0.01 | 0.55 | – |
| 5 | 19.28 | – | 0.91 | – | 0.00 | 0.13 | 126.56 | – | 19.17 | 0.01 | 0.63 | – |
| 10 | 53.84 | – | 0.64 | 0.01 | 0.00 | 0.33 | 281.37 | – | 64.71 | 0.06 | 1.33 | – |
| 15 | 112.35 | – | 0.13 | – | 0.00 | 0.88 | 412.40 | – | 192.32 | 0.03 | 2.33 | – |
| 25 | 40.93 | – | 4.48 | 0.01 | 0.00 | 0.97 | 304.44 | – | 97.30 | 0.07 | 1.72 | – |
| 50 | 65.20 | – | 4.42 | 0.02 | 0.00 | 1.80 | 356.21 | – | 339.81 | 0.09 | 1.88 | 0.01 |
| 100 | 86.79 | – | 25.10 | 0.05 | 0.61 | 7.48 | 381.33 | – | 522.53 | 0.13 | 6.49 | 0.01 |
| 250 | 63.95 | – | 93.23 | 0.12 | 0.00 | 11.56 | 397.26 | – | 188.63 | 0.30 | 13.80 | 0.06 |
| 300 | 93.76 | – | 54.46 | 0.14 | 0.00 | 56.16 | 465.14 | – | 1888.26 | 0.24 | 20.39 | 0.12 |

| $m_u$ | RSK | | MLRS | | MLRK | | $\mathrm{RB}_\alpha$ | | $\mathrm{RB}_\beta$ | | Total | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $\mathrm{AG}_u$ | Time | $\mathrm{AG}_u$ | Time | $\mathrm{AG}_u$ | Time | $\mathrm{AG}_u$ | Time | $\mathrm{AG}_u$ | Time | $\mathrm{AG}_u$ | Time |
| 2 | 0.81 | – | 0.13 | 0.18 | 0.13 | 8.34 | 1.58 | 0.05 | 0.46 | 0.07 | 5.02 | 0.79 |
| 3 | 0.39 | – | 0.05 | 0.05 | 0.05 | 6.59 | 1.92 | – | 0.17 | – | 24.91 | 0.61 |
| 5 | 0.45 | – | 0.08 | 0.06 | 0.08 | 6.92 | 1.91 | – | 0.30 | – | 15.40 | 0.65 |
| 10 | 0.99 | – | 0.24 | 0.61 | 0.22 | 13.40 | 15.66 | 0.07 | 14.73 | 0.09 | 39.43 | 1.32 |
| 15 | 0.62 | – | 0.23 | 0.08 | 0.20 | 7.54 | 2.10 | – | 1.19 | – | 65.81 | 0.78 |
| 25 | 1.09 | 0.01 | 0.32 | 1.53 | 0.30 | 19.77 | 4.42 | 0.09 | 4.21 | 0.12 | 41.75 | 2.05 |
| 50 | 1.46 | 0.02 | 0.39 | 7.21 | 0.37 | 54.34 | 30.13 | 0.20 | 30.00 | 0.26 | 75.44 | 5.81 |
| 100 | 6.22 | 0.05 | 0.88 | 17.59 | 0.85 | 121.41 | 34.17 | 0.31 | 34.67 | 0.41 | 99.97 | 13.40 |
| 250 | 89.86 | 0.12 | 1.13 | 77.20 | 1.62 | 512.10 | 62.30 | 0.66 | 62.53 | 0.89 | 88.57 | 54.82 |
| 300 | 54.65 | 0.14 | 3.53 | 448.19 | 3.33 | 2603.30 | 90.19 | 0.65 | 90.47 | 1.00 | 251.29 | 282.72 |

and Time according to $n_{dp}$. The maximum number of unsolved instances is 101 for $n_{dp} = 500$. The maximum average explored node of 2 679 348 is obtained for $n_{dp} = 15$.

Some algorithms are performing better for large instances than for small ones as the branch-and-bound. This is explained by the fact that the developed lower bound and upper bound are too far small instances which makes the convergence of branch-and-bound algorithm more difficult. In Table 18, we can notice that the minimum NU is obtained for $m_u = 100$ and the maximum running time is 544.593 s for $m_u = 500$.

Referred to Table 19, we can observe that all classes probably have the same difficulty of resolution.

TABLE 14. $AG_l^u$ according to $n_{dp}$.

| $n_{dp}$ | $AG_l^u$ | Time |
|---|---|---|
| 10 | 2.09 | 0.308 |
| 20 | 4.89 | 0.629 |
| 50 | 3.47 | 1.608 |
| 100 | 3.69 | 3.337 |
| 300 | 0.90 | 8.715 |
| 500 | 6.18 | 19.428 |
| 1000 | 0.89 | 76.474 |
| 1500 | 0.00 | 13.811 |

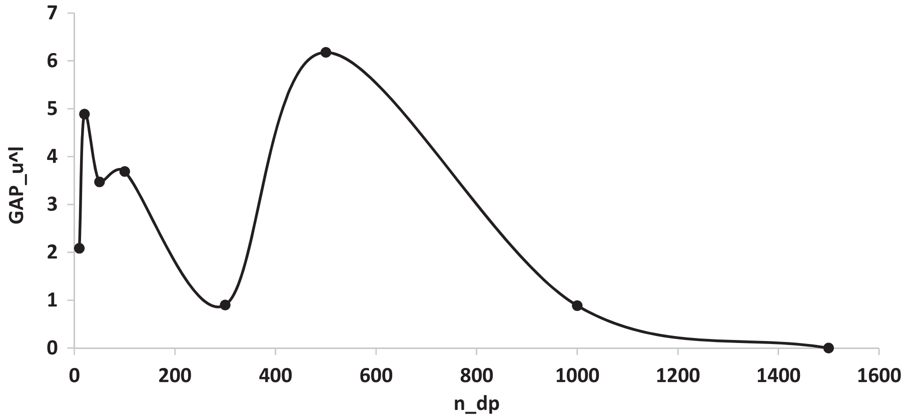

FIGURE 4. Behavior of $AG_l^u$ according to $n_{dp}$.

TABLE 15. $AG_l^u$ according to $m_u$.

| $m_u$ | $AG_l^u$ | Time |
|---|---|---|
| 2 | 0.00 | 0.236 |
| 3 | 0.52 | 0.453 |
| 5 | 0.88 | 0.611 |
| 10 | 2.69 | 1.390 |
| 15 | 2.89 | 2.683 |
| 25 | 5.16 | 4.657 |
| 50 | 6.45 | 7.388 |
| 100 | 3.06 | 23.908 |
| 250 | 14.21 | 50.960 |
| 300 | 0.59 | 182.090 |

TABLE 16. $\mathrm{AG}_l^u$ according to Class.

| Class | $\mathrm{AG}_l^u$ | Time |
|---|---|---|
| 1 | 1.09 | 8.806 |
| 2 | 3.24 | 10.875 |
| 3 | 3.56 | 16.126 |
| 4 | 3.30 | 12.270 |
| 5 | 1.89 | 42.842 |

TABLE 17. Exact solution according to $n_{dp}$.

| $n_{dp}$ | $Bg^*_{\max}$ | | |
|---|---|---|---|
| | NU | NN | Time |
| 10 | 0 | 502 | 0.011 |
| 15 | 58 | 2 679 348 | 355.113 |
| 50 | 50 | 156 547 | 200.002 |
| 100 | 50 | 975 761 | 200.081 |
| 200 | 0 | 1 | 0.004 |
| 300 | 19 | 71 074 | 76.377 |
| 500 | 101 | 53 603 | 404.016 |
| 1000 | 4 | 706 | 61.939 |
| 1500 | 0 | 1 | 0.118 |

TABLE 18. Exact solution according to $m_u$.

| $m_u$ | $Bg^*_{\max}$ | | |
|---|---|---|---|
| | NU | NN | Time |
| 2 | 0 | 1 | 0.001 |
| 3 | 0 | 43 | 0.005 |
| 5 | 9 | 569 080 | 48.893 |
| 10 | 49 | 1 124 756 | 168.004 |
| 15 | 0 | 1 | 0.209 |
| 25 | 50 | 156 547 | 200.017 |
| 50 | 50 | 1 170 913 | 240.014 |
| 100 | 5 | 112 498 | 120.570 |
| 250 | 50 | 57 925 | 400.196 |
| 300 | 54 | 43 079 | 544.593 |

TABLE 19. Exact solution according to Class.

| Class | $Bg^*_{\max}$ | | |
|---|---|---|---|
| | NU | NN | Time |
| 1 | 50 | 13 542 701 | 6004.299 |
| 2 | 55 | 19 451 110 | 6727.588 |
| 3 | 64 | 23 545 731 | 7687.297 |
| 4 | 56 | 18 348 314 | 6723.951 |
| 5 | 57 | 16 440 958 | 8545.672 |

TABLE 20. Exact solution according to $\frac{n_{dp}}{m_u}$.

| Index | $n_{dp}/m_u$ | NU | NN | Time |
|---|---|---|---|---|
| 1 | 1.6 | 50 | 127 639 | 1200.000 |
| 2 | 2 | 199 | 2 967 769 | 955.206 |
| 3 | 3 | 19 | 426 440 | 458.174 |
| 4 | 3.33 | 4 | 904 | 216.602 |
| 5 | 4 | 9 | 949 149 | 81.622 |
| 6 | 5 | 1 | 5888 | 6.170 |
| 7 | 6 | 0 | 1 | 0.123 |
| 8 | 6.6 | 0 | 1 | 0.209 |
| 9 | 10 | 0 | 1 | 0.010 |
| 10 | 12 | 0 | 1 | 0.011 |
| 11 | 15 | 0 | 1 | 0.010 |
| 12 | 16.6 | 0 | 1 | 0.003 |
| 13 | 20 | 0 | 1 | 0.008 |
| 14 | 25 | 0 | 1 | 0.001 |
| 15 | 30 | 0 | 1 | 0.005 |
| 16 | 33.33 | 181 | 20 | 20.926 |
| 17 | 40 | 0 | 1 | 0.002 |
| 18 | 50 | 0 | 1 | 0.004 |
| 19 | 60 | 0 | 1 | 0.002 |
| 20 | 100 | 0 | 1 | 0.001 |
| 21 | 150 | 0 | 1 | 0.001 |
| 22 | 250 | 0 | 1 | 0.002 |
| 23 | 500 | 0 | 1 | 0.001 |
| 24 | 750 | 0 | 1 | 0.000 |

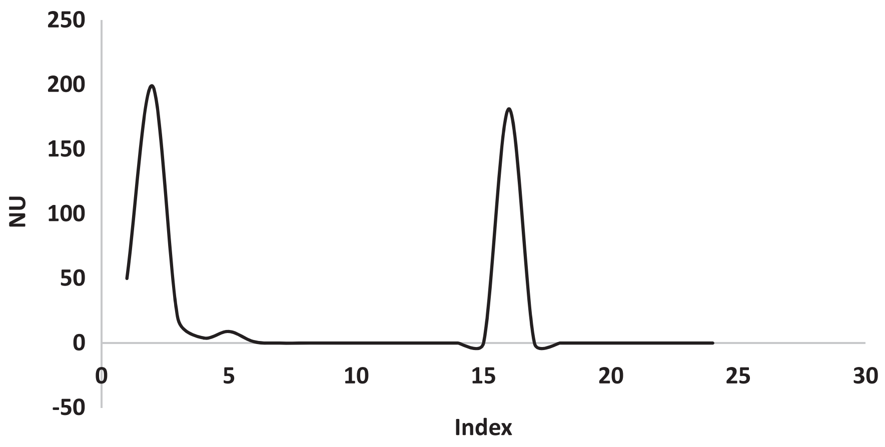

FIGURE 5. NU according to *index*.

We denote by $Index$ the number of the obtained $n_{dp}/m_u$ from all combination of pairs $n_{dp}$ and $m_u$. We have in total 24 values. We calculate the corresponding sum of NU and average of NN and Time. The results is given in Table 20.

Excluding when $\frac{n_{dp}}{m_u} = 33.33$, the unsolved instances is concentrated for $\frac{n_{dp}}{m_u} = \{1.6, 2, 3, 3.33, 4, 5\}$. The maximum average node of 2 967 769 is found for $\frac{n_{dp}}{m_u} = 2$. Figure 5 presents the NU according to the *index*.

For more details of the optimal results, Table A.4 is given in the appendix.

## 7. Conclusions

The application of the studied problem is very interesting in finance and regional development cases. The application of operational research and computer science, simulates the results to derive an experimental result that can be analyzed to present the required advantages of the found solutions. This paper shows the performance of the heuristics that were utilized to solve optimally the proposed problem. Experimental results showed that, among 2400 instances only 14.79% needed to apply the branch and bound. The developed heuristics were capable to yield the optimal solution of 88.25% out of the 2400 used instances. The remaining 11.75% instances were given a time limit of 1100 s to reach the optimal solution, after that the system is configured to choose near optimal solutions. Unsolved difficult instances such as the instances of $\frac{n_{dp}}{m_u} = \{1.6, 2, 3, 3.33, 4, 5\}$ will be studied in future work. This problem may serve as a solid background to study more complex problems. For example, the cases when we assign to each allocated budget a penalty if a deadline is exceeded without properly spending it. In addition, each budget might be subject to a release time after which this budget will be assigned to another municipality. This problem covers a range of simplified assumptions to real-life scenarios. In this research, the studied problem is intended to address a subset of municipalities with close requirements. In a future research works, the heterogeneous requirements municipalities will be considered. Such problems, will be modeled as parallel uniform machines which is an NP-hard problem.

## Appendix A.

Table A.1. Lower bounds detailed results.

| $n_{dp}$ | $m_u$ | $L_1$ | | $L_2$ | |
|---|---|---|---|---|---|
| | | AG$_l$ | Time | AG$_l$ | Time |
| | 2 | 0.48 | – | 0.00 | – |
| 10 | 3 | 40.56 | – | 0.00 | – |
| | 5 | 14.38 | – | 0.00 | – |
| | 2 | 0.00 | – | 0.00 | – |
| 20 | 3 | 62.40 | – | 0.00 | – |
| | 5 | 0.00 | – | 0.00 | 0.001 |
| | 10 | 7.72 | – | 0.00 | – |
| | 2 | 0.00 | – | 0.00 | – |
| | 3 | 0.00 | – | 0.00 | 0.001 |
| 50 | 5 | 0.00 | – | 0.00 | 0.001 |
| | 10 | 0.00 | – | 0.00 | 0.002 |
| | 25 | 2.11 | – | 0.00 | 0.007 |
| | 2 | 0.00 | – | 0.00 | – |
| | 3 | 0.00 | – | 0.00 | 0.001 |
| | 5 | 0.00 | – | 0.00 | 0.003 |
| 100 | 10 | 0.00 | – | 0.00 | 0.006 |

Table A.1. continued.

| $n_{dp}$ | $m_u$ | $L_1$ | | $L_2$ | |
|---|---|---|---|---|---|
| | | AG$_l$ | Time | AG$_l$ | Time |
| | 15 | 152.26 | – | 0.00 | 0.012 |
| | 25 | 0.00 | – | 0.00 | 0.022 |
| | 50 | 3.28 | – | 0.00 | 0.071 |
| 300 | 2 | 0.00 | – | 0.00 | 0.001 |
| | 3 | 0.00 | – | 0.00 | 0.004 |
| | 5 | 0.00 | – | 0.00 | 0.007 |
| | 10 | 0.00 | – | 0.00 | 0.016 |
| | 15 | 0.00 | – | 0.00 | 0.030 |
| | 25 | 0.00 | – | 0.00 | 0.069 |
| | 50 | 0.00 | – | 0.00 | 0.206 |
| | 100 | 0.00 | – | 0.00 | 0.684 |
| 500 | 2 | 0.00 | – | 0.00 | 0.003 |
| | 10 | 0.00 | – | 0.00 | 0.023 |
| | 25 | 0.00 | – | 0.00 | 0.080 |
| | 50 | 0.00 | – | 0.00 | 0.273 |
| | 100 | 1.00 | – | 0.00 | 0.872 |
| | 250 | 2.66 | – | 0.00 | 15.285 |
| | 300 | 398.37 | – | 0.00 | 11.304 |
| 1000 | 2 | 0.00 | – | 0.00 | 0.004 |
| | 10 | 0.00 | – | 0.00 | 0.044 |
| | 25 | 0.00 | – | 0.00 | 0.099 |
| | 50 | 0.00 | – | 0.00 | 0.363 |
| | 100 | 0.00 | – | 0.00 | 1.410 |
| | 250 | 0.00 | – | 0.00 | 9.550 |
| | 300 | 117.34 | – | 0.00 | 16.906 |
| 1500 | 2 | 0.00 | – | 0.00 | 0.006 |
| | 10 | 0.00 | – | 0.00 | 0.042 |
| | 25 | 0.00 | – | 0.00 | 0.134 |
| | 50 | 0.00 | – | 0.00 | 0.325 |
| | 100 | 0.00 | – | 0.00 | 1.187 |
| | 250 | 0.00 | – | 0.00 | 9.748 |
| | 300 | 0.00 | – | 0.00 | 15.770 |

TABLE A.2. Heuristics detailed results.

| $n_{dp}$ | $m_u$ | NIB | | $U^1_{cmin}$ | | $U^2_{cmin}$ | | NDB | | BMF | | RSS | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | AG$_u$ | Time | AG$_u$ | Time | AG$_u$ | Time | AG$_u$ | Time | AG$_u$ | Time | AG$_u$ | Time |
| 10 | 2 | 10.10 | – | 0.00 | – | 0.00 | – | 43.62 | – | 6.10 | – | 0.00 | – |
| | 3 | 10.08 | – | 0.85 | – | 0.00 | 0.01 | 34.85 | – | 8.04 | – | 0.04 | – |
| | 5 | 0.01 | – | 0.83 | – | 0.00 | 0.01 | 3.04 | – | 0.28 | – | 0.46 | – |
| 20 | 2 | 3.72 | – | 0.00 | – | 0.00 | 0.00 | 32.88 | – | 2.08 | – | 0.71 | – |
| | 3 | 74.10 | – | 0.00 | – | 0.00 | 0.02 | 157.59 | – | 44.54 | – | 0.99 | – |
| | 5 | 27.37 | – | 3.72 | – | 0.00 | 0.02 | 116.33 | – | 26.20 | – | 0.99 | – |
| | 10 | 0.02 | – | 2.56 | – | 0.00 | 0.03 | 3.44 | – | 0.65 | – | 1.23 | – |
| 50 | 2 | 3.56 | – | 0.00 | – | 0.00 | 0.02 | 48.64 | – | 1.56 | 0.01 | 1.06 | – |
| | 3 | 186.51 | – | 0.00 | – | 0.00 | 0.20 | 288.39 | – | 140.40 | – | 1.05 | – |
| | 5 | 17.16 | – | 0.00 | – | 0.00 | 0.21 | 147.38 | – | 17.27 | 0.01 | 1.19 | – |
| | 10 | 55.56 | – | 1.95 | – | 0.00 | 0.43 | 295.02 | – | 59.76 | 0.01 | 1.35 | – |
| | 25 | 0.01 | – | 5.26 | – | 0.00 | 0.82 | 6.33 | – | 1.79 | 0.01 | 1.07 | – |
| 100 | 2 | 0.92 | – | 0.00 | – | 0.00 | 0.05 | 31.44 | – | 0.80 | 0.01 | 0.82 | – |
| | 3 | 64.47 | – | 0.00 | – | 0.00 | 0.21 | 159.24 | – | 76.02 | – | 0.58 | – |
| | 5 | 9.97 | – | 0.00 | – | 0.00 | 0.21 | 173.03 | – | 10.28 | 0.02 | 0.16 | – |
| | 10 | 30.15 | – | 0.00 | – | 0.00 | 0.39 | 232.43 | – | 38.49 | 0.02 | 0.00 | – |
| | 15 | 195.27 | – | 0.12 | – | 0.00 | 1.21 | 508.76 | – | 339.13 | – | 0.00 | – |
| | 25 | 54.55 | – | 21.63 | – | 0.00 | 1.38 | 268.48 | – | 135.48 | 0.01 | 0.00 | – |
| | 50 | 0.01 | – | 5.73 | – | 0.00 | 1.66 | 7.50 | – | 4.57 | 0.01 | 0.00 | – |
| 300 | 2 | 1.44 | – | 0.00 | – | 0.00 | 0.06 | 45.40 | – | 1.00 | 0.02 | 0.06 | – |
| | 3 | 2.46 | – | 0.00 | – | 0.00 | 0.12 | 105.54 | – | 1.50 | 0.04 | 0.07 | – |
| | 5 | 41.88 | – | 0.00 | – | 0.00 | 0.19 | 193.03 | – | 41.82 | 0.04 | 0.34 | – |
| | 10 | 37.40 | – | 0.00 | – | 0.00 | 0.35 | 361.20 | – | 47.75 | 0.05 | 2.83 | – |
| | 15 | 29.43 | – | 0.15 | – | 0.00 | 0.55 | 316.04 | – | 45.50 | 0.05 | 4.66 | – |
| | 25 | 54.37 | – | 0.00 | 0.01 | 0.00 | 0.90 | 385.26 | – | 102.91 | 0.05 | 5.92 | – |
| | 50 | 110.91 | – | 7.10 | 0.01 | 0.00 | 1.77 | 517.04 | – | 669.05 | 0.05 | 6.58 | – |
| | 100 | 100.81 | – | 47.89 | 0.02 | 2.44 | 17.97 | 412.11 | – | 499.10 | 0.05 | 7.25 | – |
| 500 | 2 | 1.44 | – | 0.00 | – | 0.00 | 0.07 | 61.92 | – | 0.88 | 0.04 | 5.22 | – |
| | 10 | 61.02 | – | 0.00 | – | 0.00 | 0.34 | 322.47 | – | 85.27 | 0.09 | 3.92 | – |
| | 25 | 70.02 | – | 0.00 | 0.01 | 0.00 | 0.94 | 396.84 | – | 178.71 | 0.09 | 3.31 | – |
| | 50 | 59.13 | – | 8.25 | 0.02 | 0.00 | 1.82 | 401.85 | – | 234.14 | 0.10 | 2.83 | – |
| | 100 | 100.84 | – | 33.85 | 0.02 | 0.00 | 4.57 | 371.90 | – | 543.86 | 0.09 | 1.90 | – |
| | 250 | 0.01 | – | 11.41 | 0.05 | 0.00 | 9.70 | 13.37 | – | 2.84 | 0.23 | 1.44 | – |
| | 300 | 0.00 | – | 0.06 | 0.05 | 0.00 | 4.98 | 1.08 | – | 0.02 | 0.29 | 0.91 | – |
| 1000 | 2 | 1.64 | – | 0.00 | 0.00 | 0.00 | 0.09 | 59.24 | – | 1.12 | 0.04 | 0.19 | – |
| | 10 | 88.00 | – | 0.00 | 0.01 | 0.00 | 0.34 | 379.13 | – | 103.34 | 0.09 | 0.00 | – |
| | 25 | 40.02 | – | 0.00 | 0.02 | 0.00 | 0.87 | 395.72 | – | 93.21 | 0.10 | 0.00 | – |
| | 50 | 53.88 | – | 0.00 | 0.03 | 0.00 | 1.92 | 404.04 | – | 229.07 | 0.11 | 0.00 | 0.01 |
| | 100 | 55.41 | – | 1.72 | 0.07 | 0.00 | 3.59 | 364.57 | – | 285.47 | 0.14 | 16.78 | 0.02 |
| | 250 | 49.33 | – | 91.82 | 0.11 | 0.00 | 16.48 | 365.89 | – | 157.39 | 0.27 | 39.58 | 0.06 |
| | 300 | 73.15 | – | 78.45 | 0.16 | 0.00 | 149.69 | 231.20 | – | 423.64 | 0.12 | 38.84 | 0.22 |
| 1500 | 2 | 0.72 | – | 0.00 | 0.00 | 0.00 | 0.13 | 47.88 | – | 0.36 | 0.07 | 0.00 | – |
| | 10 | 104.75 | – | 0.00 | 0.02 | 0.00 | 0.41 | 375.89 | – | 117.74 | 0.16 | 0.00 | – |
| | 25 | 26.59 | – | 0.00 | 0.03 | 0.00 | 0.92 | 374.01 | – | 71.70 | 0.16 | 0.00 | 0.01 |
| | 50 | 102.08 | – | 1.01 | 0.04 | 0.00 | 1.84 | 450.60 | – | 562.24 | 0.18 | 0.00 | 0.02 |
| | 100 | 90.09 | – | 16.95 | 0.08 | 0.00 | 3.80 | 376.73 | – | 761.70 | 0.22 | 0.03 | 0.03 |
| | 250 | 142.53 | – | 176.46 | 0.20 | 0.00 | 8.52 | 812.52 | – | 405.65 | 0.39 | 0.38 | 0.10 |
| | 300 | 208.13 | – | 84.87 | 0.20 | 0.00 | 13.82 | 1163.14 | – | 5241.13 | 0.32 | 21.42 | 0.13 |

Table A.2. continued.

| $n_{dp}$ | $m_u$ | RSK |  | MLRS |  | MLRK |  | RB$_\alpha$ |  | RB$_\beta$ |  |
|---|---|---|---|---|---|---|---|---|---|---|---|
|  |  | AG$_u$ | Time | AG$_u$ | Time | AG$_u$ | Time | AG$_u$ | Time | AG$_u$ | Time |
| 10 | 2 | 0.00 | – | 0.00 | 0.03 | 0.00 | 2.77 | 0.14 | – | 0.00 | – |
|  | 3 | 0.02 | – | 0.05 | 0.03 | 0.05 | 3.52 | 0.54 | – | 0.00 | – |
|  | 5 | 0.43 | – | 0.08 | 0.03 | 0.08 | 4.32 | 0.84 | – | 0.00 | – |
| 20 | 2 | 0.54 | – | 0.13 | 0.04 | 0.13 | 5.14 | 2.06 | – | 0.00 | – |
|  | 3 | 0.80 | – | 0.14 | 0.05 | 0.14 | 5.96 | 3.13 | – | 0.00 | – |
|  | 5 | 0.85 | – | 0.15 | 0.05 | 0.15 | 6.49 | 3.16 | – | 0.00 | – |
|  | 10 | 1.21 | – | 0.12 | 0.05 | 0.12 | 7.17 | 2.76 | – | 0.00 | – |
| 50 | 2 | 0.89 | – | 0.09 | 0.06 | 0.09 | 8.14 | 2.46 | – | 0.00 | – |
|  | 3 | 1.05 | – | 0.04 | 0.06 | 0.04 | 9.00 | 1.24 | – | 0.00 | – |
|  | 5 | 0.86 | – | 0.03 | 0.06 | 0.03 | 9.88 | 0.03 | – | 0.00 | – |
|  | 10 | 0.83 | – | 0.02 | 0.07 | 0.02 | 10.89 | 0.00 | – | 0.00 | – |
|  | 25 | 0.45 | – | 0.00 | 0.06 | 0.00 | 9.45 | 0.00 | – | 0.00 | – |
| 100 | 2 | 0.37 | – | 0.00 | 0.05 | 0.00 | 7.68 | 0.00 | – | 0.00 | – |
|  | 3 | 0.10 | – | 0.00 | 0.04 | 0.00 | 6.01 | 0.00 | – | 0.00 | – |
|  | 5 | 0.02 | – | 0.00 | 0.03 | 0.00 | 4.32 | 0.00 | – | 0.00 | – |
|  | 10 | 0.00 | – | 0.00 | 0.02 | 0.00 | 2.79 | 0.00 | – | 0.00 | – |
|  | 15 | 0.00 | – | 0.00 | 0.03 | 0.00 | 3.57 | 0.03 | – | 0.00 | – |
|  | 25 | 0.00 | – | 0.00 | 0.05 | 0.00 | 4.80 | 0.60 | – | 0.03 | – |
|  | 50 | 0.00 | – | 0.00 | 0.07 | 0.00 | 6.25 | 3.27 | – | 0.75 | – |
| 300 | 2 | 0.00 | – | 0.00 | 0.08 | 0.00 | 7.33 | 4.59 | – | 0.81 | – |
|  | 3 | 0.00 | – | 0.00 | 0.10 | 0.00 | 8.44 | 4.69 | – | 0.86 | – |
|  | 5 | 0.08 | – | 0.15 | 0.11 | 0.15 | 9.59 | 5.51 | – | 1.51 | – |
|  | 10 | 0.50 | – | 0.44 | 0.12 | 0.35 | 10.65 | 6.05 | – | 2.53 | – |
|  | 15 | 1.25 | – | 0.46 | 0.13 | 0.40 | 11.52 | 4.18 | – | 2.39 | – |
|  | 25 | 1.82 | – | 0.85 | 0.14 | 0.79 | 12.62 | 4.36 | – | 3.57 | – |
|  | 50 | 3.72 | – | 1.25 | 0.14 | 1.14 | 13.49 | 5.35 | – | 4.55 | – |
|  | 100 | 4.53 | – | 1.10 | 0.17 | 0.99 | 16.88 | 4.50 | – | 3.90 | – |
| 500 | 2 | 4.49 | – | 0.83 | 0.19 | 0.80 | 20.82 | 3.39 | – | 2.85 | – |
|  | 10 | 4.40 | – | 0.81 | 0.21 | 0.76 | 24.25 | 2.59 | – | 2.28 | – |
|  | 25 | 4.29 | – | 0.42 | 0.22 | 0.37 | 27.03 | 1.09 | – | 1.03 | – |
|  | 50 | 2.56 | – | 0.02 | 0.24 | 0.02 | 30.04 | 0.00 | – | 0.00 | 0.01 |
|  | 100 | 1.67 | – | 0.02 | 0.20 | 0.02 | 24.72 | 0.00 | – | 0.00 | 0.01 |
|  | 250 | 1.29 | – | 0.00 | 0.15 | 0.00 | 18.49 | 0.00 | – | 0.00 | 0.01 |
|  | 300 | 0.63 | – | 0.00 | 0.11 | 0.00 | 12.73 | 0.00 | – | 0.00 | 0.01 |
| 1000 | 2 | 0.17 | – | 0.00 | 0.07 | 0.00 | 7.78 | 0.00 | 0.01 | 0.00 | 0.01 |
|  | 10 | 0.00 | – | 0.00 | 0.11 | 0.00 | 3.28 | 0.00 | 0.01 | 0.00 | 0.01 |
|  | 25 | 0.00 | – | 0.00 | 0.26 | 0.00 | 4.23 | 0.00 | 0.01 | 0.00 | 0.01 |
|  | 50 | 0.00 | 0.04 | 0.39 | 18.13 | 0.39 | 109.95 | 45.35 | 0.37 | 46.89 | 0.49 |
|  | 100 | 1.72 | 0.09 | 2.40 | 34.57 | 2.40 | 226.73 | 50.62 | 0.47 | 52.23 | 0.62 |
|  | 250 | 91.82 | 0.13 | 2.87 | 150.51 | 2.90 | 965.36 | 47.10 | 0.78 | 47.64 | 1.04 |
|  | 300 | 78.45 | 0.19 | 2.27 | 1215.05 | 2.23 | 7124.98 | 72.24 | 0.87 | 72.24 | 1.16 |
| 1500 | 2 | 0.00 | – | 0.00 | 0.96 | 0.00 | 7.07 | 0.00 | 0.41 | 0.00 | 0.53 |
|  | 10 | 0.00 | 0.02 | 0.30 | 3.71 | 0.30 | 34.74 | 98.20 | 0.45 | 98.34 | 0.61 |
|  | 25 | 0.00 | 0.04 | 0.63 | 8.47 | 0.63 | 60.48 | 20.47 | 0.53 | 20.64 | 0.70 |
|  | 50 | 1.01 | 0.06 | 0.30 | 17.45 | 0.30 | 111.99 | 96.70 | 0.62 | 97.79 | 0.82 |
|  | 100 | 16.95 | 0.09 | 0.00 | 35.41 | 0.00 | 217.31 | 81.56 | 0.76 | 82.53 | 1.02 |
|  | 250 | 176.46 | 0.23 | 0.51 | 80.94 | 1.98 | 552.45 | 139.81 | 1.21 | 139.97 | 1.62 |
|  | 300 | 84.87 | 0.24 | 8.33 | 129.41 | 7.76 | 672.17 | 198.34 | 1.09 | 199.16 | 1.83 |

Table A.3. $\mathrm{AG}_l^u$ detailed results.

| $n_{dp}$ | $m_u$ | $\mathrm{AG}_l^u$ |
|---|---|---|
| 10 | 2 | 0.00 |
| | 3 | 2.60 |
| | 5 | 3.66 |
| 20 | 2 | 0.00 |
| | 3 | 0.00 |
| | 5 | 0.73 |
| | 10 | 18.83 |
| 50 | 2 | 0.00 |
| | 3 | 0.00 |
| | 5 | 0.00 |
| | 10 | 0.00 |
| | 25 | 17.37 |
| 100 | 2 | 0.00 |
| | 3 | 0.00 |
| | 5 | 0.00 |
| | 10 | 0.00 |
| | 15 | 0.00 |
| | 25 | 0.00 |
| | 50 | 25.81 |
| 300 | 2 | 0.00 |
| | 3 | 0.00 |
| | 5 | 0.00 |
| | 10 | 0.00 |
| | 15 | 0.00 |
| | 25 | 0.00 |
| | 50 | 0.00 |
| | 100 | 7.21 |
| 500 | 2 | 0.00 |
| | 10 | 0.00 |
| | 25 | 0.00 |
| | 50 | 0.00 |
| | 100 | 5.04 |
| | 250 | 37.62 |
| | 300 | 0.58 |
| 1000 | 2 | 0.00 |
| | 10 | 0.00 |
| | 25 | 0.00 |
| | 50 | 0.00 |
| | 100 | 0.00 |
| | 250 | 5.00 |
| | 300 | 1.20 |
| 1500 | 2 | 0.00 |
| | 10 | 0.00 |
| | 25 | 0.00 |
| | 50 | 0.00 |
| | 100 | 0.00 |
| | 250 | 0.00 |
| | 300 | 0.00 |

Table A.4. Exact solution detailed results.

| $n_{dp}$ | $m_u$ | $Bg^*_{\max}$ | | |
|---|---|---|---|---|
| | | NU | NN | Time |
| 10 | 2 | 0 | 1 | 0.002 |
| | 3 | 0 | 212 | 0.010 |
| | 5 | 0 | 1293 | 0.023 |
| 20 | 2 | 0 | 1 | 0.001 |
| | 3 | 0 | 1 | 0.008 |
| | 5 | 9 | 2 844 103 | 244.437 |
| | 10 | 49 | 7 873 288 | 1176.007 |
| 50 | 2 | 0 | 1 | 0.001 |
| | 3 | 0 | 1 | 0.003 |
| | 5 | 0 | 1 | 0.002 |
| | 10 | 0 | 1 | 0.006 |
| | 25 | 50 | 939 275 | 1200.000 |
| 100 | 2 | 0 | 1 | 0.001 |
| | 3 | 0 | 1 | 0.003 |
| | 5 | 0 | 1 | 0.002 |
| | 10 | 0 | 1 | 0.003 |
| | 15 | 0 | 1 | 0.411 |
| | 25 | 0 | 1 | 0.065 |
| | 50 | 50 | 5 854 560 | 1200.000 |
| 300 | 2 | 0 | 1 | 0.001 |
| | 3 | 0 | 1 | 0.002 |
| | 5 | 0 | 1 | 0.003 |
| | 10 | 0 | 1 | 0.004 |
| | 15 | 0 | 1 | 0.006 |
| | 25 | 0 | 1 | 0.011 |
| | 50 | 0 | 1 | 0.023 |
| | 100 | 19 | 426 440 | 458.174 |
| 500 | 2 | 0 | 1 | 0.002 |
| | 10 | 0 | 1 | 0.007 |
| | 25 | 0 | 1 | 0.019 |
| | 50 | 0 | 1 | 0.038 |
| | 100 | 1 | 23 549 | 24.091 |
| | 250 | 50 | 170 428 | 1200.000 |
| | 300 | 50 | 127 639 | 1200.000 |
| 1000 | 2 | 0 | 1 | 0.001 |
| | 10 | 0 | 1 | 0.001 |
| | 25 | 0 | 1 | 0.00 |
| | 50 | 0 | 1 | 0.00 |
| | 100 | 0 | 1 | 0.00 |
| | 250 | 0 | 3345 | 0.37 |
| | 300 | 4 | 1596 | 433.20 |
| 1500 | 2 | 0 | 1 | 0.00 |
| | 10 | 0 | 1 | 0.00 |
| | 25 | 0 | 1 | 0.00 |
| | 50 | 0 | 1 | 0.01 |
| | 100 | 0 | 1 | 0.01 |
| | 250 | 0 | 1 | 0.22 |
| | 300 | 0 | 1 | 0.58 |

*Acknowledgements.* The authors would like to thank the Deanship of Scientific Research at Majmaah University for supporting this work under Project no. RGP-2019-13.

## References

[1] M. Alharbi and M. Jemmali, Algorithms for investment project distribution on regions. *Comput. Intell. Neurosci.* **2020** (2020) 3607547.

[2] H. Alquhayz, M. Jemmali and M.M. Otoom, Dispatching-rule variants algorithms for used spaces of storage supports. *Discrete Dyn. Nat. Soc.* **2020** (2020).

[3] A.C. Alvim and C.C. Ribeiro, A hybrid bin–packing heuristic to multiprocessor scheduling. In: International Workshop on Experimental and Efficient Algorithms. Springer (2004) 1–13.

[4] K.J. Arrow, Economic Welfare and the Allocation of Resources for Invention. Macmillan Education UK, London (1972) 219–236.

[5] B.S. Baker, A new proof for the first-fit decreasing bin-packing algorithm. *J. Algorithms* **6** (1985) 49–70.

[6] M. Dell'Amico and S. Martello, Optimal scheduling of tasks on identical parallel processors. *ORSA J. Comput.* **7** (1995) 191–200.

[7] M. Haouari and A. Gharbi, Fast lifting procedures for the bin packing problem. *Discrete Optim.* **2** (2005) 201–218.

[8] M. Haouari and M. Jemmali, Tight bounds for the identical parallel machine-scheduling problem: Part II. *Int. Trans. Oper. Res.* **15** (2008) 19–34.

[9] M. Haouari and M. Jemmali, Maximizing the minimum completion time on parallel machines. *4OR* **6** (2008) 375–392.

[10] M. Haouari, A. Gharbi and M. Jemmali, Tight bounds for the identical parallel machine scheduling problem. *Int. Trans. Oper. Res.* **13** (2006) 529–548.

[11] M. Jemmali, Approximate solutions for the projects revenues assignment problem. *Commun. Math. App.* **10** (2019) 653–658.

[12] M. Jemmali, Budgets balancing algorithms for the projects assignment. *Int. J. Adv. Comput. Sci. App.* **10** (2019) 574–578.

[13] M. Jemmali, L.K.B. Melhim, S.O.B. Alharbi and A.S. Bajahzar, Lower bounds for gas turbines aircraft engines. *Commun. Math. App.* **10** (2019) 637–642.

[14] M. Jemmali, L.K.B. Melhim and M. Alharbi, Randomized-variants lower bounds for gas turbines aircraft engines. In: World Congress on Global Optimization. Springer (2019) 949–956.

[15] N. Katoh and T. Ibaraki, Resource Allocation Problems. Springer US, Boston, MA (1999) 905–1006.

[16] H. Kellerer, U. Pferschy and D. Pisinger, Multidimensional knapsack problems. In: Knapsack problems. Springer (2004) 235–283.

[17] E.L. Lawler, J.K. Lenstra, A.H.R. Kan and D.B. Shmoys, Sequencing and scheduling: algorithms and complexity. In: Vol. 4 of *Handbooks in Operations Research and Management Science* (1993) 445–522.

[18] S. Martello, D. Pisinger and P. Toth, Dynamic programming and strong bounds for the 0-1 knapsack problem. *Manage. Sci.* **45** (1999) 414–424.

[19] R. Walter, M. Wirth and A. Lawrinenko, Improved approaches to the exact solution of the machine covering problem. *J. Scheduling* **20** (2017) 147–164.

[20] D. Pisinger, Dynamic programming on the word ram. *Algorithmica* **35** (2003) 128–145.

[21] B. Xia and Z. Tan, Tighter bounds of the first fit algorithm for the bin-packing problem. *Discrete Appl. Math.* **158** (2010) 1668–1675.