# Adaptive Embedded Systems
## *New Composed Technical Solutions for Feasible Low-Power and Real-time Flexible OS Tasks*

Hamza Chniter[1,2,3], Mohamed Khalgui[1,2] and Fethi Jarray[3,4]
[1]*INSAT Institute, University of Carthage, Carthage, Tunisia*
[2]*FST faculty, University of Elmanar, Elmanar, Tunisia*
[3]*ISI Institute, Mednine, Tunisia*
[4]*Laboratoire CEDRIC, CNAM, 292 rue St-Martin 75141, Paris cedex 03, France*

Keywords: Flexible Embedded System, Reconfiguration, Real-time and Low-power Scheduling, Integer Programming, Heuristic.

Abstract: The paper deals with low-power adaptive scheduling of synchronous and flexible real-time OS tasks. A software reconfiguration scenario is assumed to be any run-time operation allowing the addition-removal-update of OS tasks to adapt the system to its environment under well-defined conditions. The problem is that any reconfiguration can push the system to an unfeasible behavior where temporal properties are violated or the energy consumption is possibly high and unacceptable. A task in the system can change its characteristics at any time when a reconfiguration scenario is applied, it can also be stopped or replaced by another one. The difficulty is how to find the new temporal parameters of the systems tasks after any reconfiguration. We use a DVS processor which is with a variable speed to support run-time solutions that re-obtain the system's feasibility. The challenge is how to compute the best combinations between available processor speeds for a good compromise between execution time and energy consumption. We propose a combinatorial optimization method based on integer programming and heuristics. We propose also a solution when the available speeds do not allow the feasibility of the system. Both approaches include a mechanism to adjust the deadlines of tasks to satisfy the feasibility conditions and overcome the problem of rejected tasks. This mechanism makes the scheduling more flexible and able to react in accordance with its environment.

## 1 INTRODUCTION

An embedded system(ES) is a device with a dedicated function including hardware and software parts which form a unique component of a larger physical system which is expected to run without human intervention. The embedded systems should often run under real-time constraints that determine their reliability and correctness (Liu and Layland, 1973; Barr, 2007; Heath and Steve, 2003). ES are designed to control many devices in common use today to meet a large range of user requirements. Modern embedded systems are often based on microcontrollers and integrated in more complex systems to perform specific tasks (Heath and Steve, 2003). The goal is to optimize it in order to reduce the size, the cost of the product, and increase the reliability as well as performance. Some embedded systems also have real-time performance constraints that must be met for many reasons such as safety and usability. Others may have power constraints that can be violated after particular reconfiguration scenarios. A configuration scenario is assumed as a runtime software intervention which act on the system state to allow the addition-removal-update of OS tasks and consequently adapt the system to its environment under functional and extra-functional requirements. A reconfiguration can change the system behavior where temporal properties are violated or the energy consumption overcomes its limit and push the system to a unfeasible state. One challenge with embedded systems is delivering predictably good performance on monitoring and controlling. In fact, embedded systems have hard real-time requirements: if computations are not completed before a deadline, the system will fail, possibly injuring users and can causes many side effects. Hence the challenge for real-time system researchers is to develop approaches to design fast systems with


Chniter H., Khalgui M. and Jarray F..
Adaptive Embedded Systems - New Composed Technical Solutions for Feasible Low-Power and Real-time Flexible OS Tasks.
DOI: 10.5220/0005066000920101
In *Proceedings of the 11th International Conference on Informatics in Control, Automation and Robotics* (ICINCO-2014), pages 92-101
ISBN: 978-989-758-039-0

high predicted performance that must provide real-time response. When a processor is running, reconfigurable embedded systems can undergo different disturbances into their environments due to a reconfiguration scenario (Imran Rafiq Quadri et al., 2012). This can lead to the violation of temporal constraints such as deadlines, increasing in energy consumption and following a non-feasible system. Configuration scenarios can be a result of the addition-removal-update of the tasks in the system. To reach a system that gather between providing real-time responses and low-power consumption, modern embedded systems integrate new processor technology called DVS (Dynamic Voltage Scaling)(PARAIN et al., 2000) allowing to dynamically change the processor speed of OS tasks to make tradeoff between the energy consumption and the execution time. Each processor will have a set of available operating speeds which it can operate with. This technology tries to change the voltage of the chip which is related with the processor speed and the duration of tasks execution (He et al., 2012). The difficulty lies in determining the best scaling factor of voltage for the whole system at any instant when a reconfiguration occurs in order to achieve a new behavior or implementation of the system that meets all timing constraints and consumes less energy. To overcome the problem, we propose a combinatorial optimization approach based on integer programming (Hladik et al., 2008), and fast heuristic (Jeannenot et al., 2004)). The objective is to find the optimal scaling factor in order to obtain a new feasible system after any reconfiguration such as adding new tasks to the system. The approach tries to give additional solutions when the processor drains all the available scaling factors and the system lies unfeasible. Those solutions try to adjust the deadlines of the tasks by using fast optimization approaches to provide a more flexible system that can properly be adapted to its environment when any overload condition occurs. The remainder of this paper is organized as follows. In Section 2, we discuss the originality of this paper by studying the state of the art. In section 3, we expose the problem. We present in Section 4 some terminologies and the contribution dealing with integer program formulation and proposed heuristic to find the optimal scaling factors and adjusted deadlines. Finally, numerical results are presented and discussed in Section 5.

## 2 RELATED WORKS

Nowadays, real-time reconfigurable systems need more and more solutions for flexible and adaptive scheduling of their tasks under power constraints. The problem of real-time scheduling is classically to ensure the execution of all the tasks at run-time without missing the deadlines where the total energy consumption is minimized (Letters, 1996). The use of maximum scaling factor of the processor can accelerate the execution time of all tasks and meet the temporal constraints. This can produce significant energy consumption that exceeds the system capacity, hence the fact to vary the scaling factor during execution becomes a need. A new technology known as DVS(Dynamic Voltage Scaling) (PARAIN et al., 2000) is integrated in the new processors for this purpose to dynamically change the processor speed. Choosing the suitable scaling factor for the tasks to ensure the best compromise between the execution time and the energy consumption remains the most desired constraint. Several studies have been performed in this context such as integer programming (Hladik et al., 2008), graph traverse (Heilmann, 2003), branch and bound (Xu, 1993). In (Fang and Lin, 2013), the authors presented a linear integer program to solve the problem by applying DVS technique for mobile computing platforms. In (ciçek and Celik, 2011) and (Fidanova, 2006), the low-power scheduling problem was studied for parallel processors architecture, a simulated annealing and a tabu search approaches were proposed to solve the problem. Each task can be divided into a number of parts called sub-tasks, and each part must be executed on a separate processor. In (Ying and Cheng, 2010), it was assumed that all the processors are available and each processor can handle work on time without preemption. In addition, each arriving job can be processed properly. The author in (He et al., 2012) tries to solve the problem by breaking down the processor to active and inactive state. He presents a mechanism to adjust the supply voltage depending on the load working system for low-power energy consumption. Genetic algorithms have been also applied to solve the scheduling problems for multiprocessor scheduling periodic dependent tasks such as in (Nossal, 1998; Dalfard and Mohammadi, 2012). Two approaches was proposed in (Chniter et al., 2014) to solve the scheduling problem in a reconfigurable real-time system. The objective is to determine the suitable processor scaling factor which meet the corresponding deadlines and to decrease the energy consumption. In another way and to reach a flexible system that react correctly with it environment, (Chantem et al., 2009; Dwivedi, 2012) present an elastic real-time model based on period and deadline adjustment. The objective is to find a solution for rejected tasks in the system by changing the period or deadline of OS tasks.

Among this category, there are those which try to solve the problem of real-time scheduling by fixing the adequate scaling factor of processor for a feasible system with a low-power energy consumption. Nevertheless, no one in related works addresses this problem by using integer programming and heuristics to allow optimal reconfiguration real-time scheduling with power constraints. In addition, they did not consider the case when all the available scaling factors with which the processor can operate, may not guarantee a solution for violated temporal constraints. In this case, the system may not react in accordance with its environment and can miss interesting tasks. Other works like (Buttazzo et al., 1998; Chantem et al., 2009; Dwivedi, 2012) try this problem but they did not take into account the energy constraints. They try to find the adjusted parameters without fixing the execution sequence of tasks.

In the present paper we propose an elastic method to determine the appropriate scaling factor of processor for a feasible reconfigurable system with a low-power energy consumption. If the system lies unfeasible, a proposed solution based in deadline adjustment is used to meet the new requested constraints after reconfiguration. The proposed method produces the optimal scaling factor of processor, new adjusted deadlines and the execution sequence of tasks.

## 3 PROBLEM AND NOTATIONS

We assume a reconfigurable real-time system to be composed of periodic independent synchronous tasks. A reconfiguration scenario is any run-time operation allowing the addition-removal-update of tasks to adapt the system to its environment. Nevertheless, the application of a scenario can increase the energy consumption or push some tasks to violate the corresponding deadlines. Our goal is to provide some solutions that will optimize the energy consumption and guarantee the respect of deadlines after each reconfiguration scenario. We propose an Integer Programming model and a heuristic to find the required solution by changing the scaling factor of the processor. We construct also a mechanism that will be applied when no available scaling factor can fulfill the system requirements. This mechanism is based on the deadlines adjustment to meet the corresponding constraints and it presents a solution for a more flexible and relaxed system which can react properly with its environment .

**Notation:**

Let us assume a reconfigurable system to be initially composed of $n$ periodic tasks $T_i, i = 1 \dots n$. We assume a reconfiguration scenario to add $m$ new tasks. Each task is characterized by four parameters according to (Liu and Layland, 1973). Firstly by its release (or arrival) time $r_i$, i.e each task $T_i$ cannot begin execution before $r_i$. Secondly by its absolute deadline constraint $d_i$, i.e. each task should finish before $d_i$. Thirdly by its computation time at the normalized processor frequency $C_{ni}$. Finally by its period which is equal to the deadline. It is assumed that the WCET(Worst Case Execution Time) is constant and that the tasks will be executed on a single processor with a variable operating frequency according to the EDF scheduling policy. We denote respectively by $f_n$ and $V_n$ the normalized frequency and the voltage of the system. the actual execution time (i.e. computational delay) of the task is prolonged when the voltage is decreased to save the energy. The reason is that the frequency of the processor is approximately linearly proportional to the supply voltage(Zhu, 2005). We see that reducing voltage cuts down the energy dissipation, but the operating frequency will decrease accordingly. We can see that the task execution time is inversely proportional to the voltage. In order to provide the required system performance, the supply voltage should be scaled as low as possible to minimize the energy consumption, while guaranteeing the temporal properties. We suppose that each task $T_i$ is executed at frequency $F_i$ and at voltage $V_i$. We denote by $\eta_i$ the reduction factor of voltage when $T_i$ is executed, $V_i = \frac{V_n}{\eta_i}$. So the WCET is equal to $C_i = C_{ni}\eta_i$.The power consumption is $P = CV^2F$ where $C$ is a constant related to the circuit type of the processor ensuring that $P_i$ has the dimension of a power(He et al., 2012). Hence, if the system is running over $x$ times, the energy consumption is $E = Px$. The problem is then to allow a low-power and real-time optimal scheduling of reconfigurable tasks after each reconfiguration scenario. We assume a simplified model of power, i.e. The power $P_i$ consumed by the task $T_i$ is :

$$P_i = CV_i^2F_i = C\frac{V_nf_n}{{\eta_i}^3}.$$

The energy $E_i$ consumed by the task $T_i$ is:

$E_i = P_iC_i = C\frac{V_nf_nC_{ni}}{{\eta_i}^2} = K\frac{C_{ni}}{{\eta_i}^2}$ with $K = CV_nF_n$.

So the total energy consumption of the system is:

$$E = \sum_{i=1}^{n} E_i = K\sum_{i=1}^{n}\frac{C_{ni}}{{\eta_i}^2} \quad (1)$$

the CPU charge factor $U$ is calculated by:

$$U = \sum_{i=1}^{n}\frac{C_i}{d_i}. \quad (2)$$

where we remember that $n$ denotes the number of tasks in the system and $C_i$, $d_i$ are respectively the execution time and the deadline of the task $i$.

# 4 CONTRIBUTION: FLEXIBLE RECONFIGURABLE REAL-TIME SCHEDULING WITH DEADLINE ADJUSTMENT

This section deals with the proposed methods to compute the scaling factor and estimate the deadline adjustment for the tasks after any reconfiguration scenario. The originality of our contribution consists in finding not only the optimal scaling factors to ensure temporal constraints but also the adjusted deadlines which represent a flexible proposed solution when the processor drains all the available scaling factors and the system lies unfeasible.

## 4.1 Mixed Integer Programming Model

The model tries to find the optimal scaling factors and minimize the total energy consumption of the system under various operating constraints. We assume that the tasks will be executed in a single processor system with variable scaling factors. It is assumed that the processor has a set of $p$ available scaling factor. we denote by $m_k : k = 1..p$ the $k^{th}$ available scaling factors. We introduce a binary variable to describe the combination between the scaling factor $k$ and tasks $i$.

$$Y_{ik} = \begin{cases} 1 \text{ If the task } i \text{ is executed with the scaling factor } k \\ \quad i = 1 \dots n, k = 1 \dots p \\ 0 \text{ Otherwise} \end{cases} \tag{3}$$

Let $t_i$ be the effective starting time of the task $T_i$. Our goal is to minimize the total consumed energy under the following operating constraints:

a) No simultaneously executed tasks:
To ensure a single executed task in a time, we should have either $t_j - t_i - Cn_i\, m_k\, Y_{ik} \geq 0$ or $t_i - t_j - Cn_j\, m_k\, Y_{jk} \geq 0$ or for every pair of tasks $T_i$ and $T_j$ . This condition can be rewritten as $t_i - t_j \geq Cn_j\, m_k\, Y_{ik} - M\alpha_{ij}$ and $t_j - t_i \geq Cn_i\, m_k\, Y_{jk} - M(1-\alpha_{ij})$ where $\alpha_{ij}$ is a binary variable and $M$ is a big constant. $\alpha_{ij} = 1$ means that $T_j$ is executed before $T_i$.

b) The deadline of each task should be respected

$$t_i + Cn_i\, m_k\, Y_{ik} \leq d_i \;\forall\, k \tag{4}$$

c) The the release time should be respected: $t_i \geq r_i \;\forall\, i$.

Thus the basic model is the following:

$$PS \begin{cases} Minimize\ \alpha_i + K \sum_{i=1}^{n} \frac{Cn_i}{f_i^2} \\ s.t. \\ t_i - t_j \geq Cn_i\, m_k\, Y_{ik} - M\alpha_{ij} \\ t_j - t_i \geq Cn_i\, m_k\, Y_{ik} - M\,(1-\alpha_{ij}) \\ \alpha i\ (t_i + Cn_i\, Y_{ij}\, m_k) \leq d_i \;\forall\, i \\ t_i \geq r_i \;\forall\, i \\ \alpha_i \leq 1 \\ \sum_{i=1}^{n} \frac{Cn_i\, m_k\, Y_{ik}\, \alpha_i}{dn_i} \leq 1 \\ d_i = \frac{dn_i}{\alpha_i} \\ \sum_{k=1}^{p} Y_{ik} = 1 \;\forall\, i \\ t_i \geq 0 \;\forall\, i \\ \alpha_{ij} \in \{0,1\} \;\forall\, i < j \end{cases} \tag{5}$$

With $f_i = \sum_{k=1}^{p} (m_k\, Y_{ik})$ is the scaling factor of the processor to execute the task $i$.

**Case Study**

We assume a real-time embedded system to be composed of 5 tasks as depicted in table 1. The

Table 1: Current system configuration.

| Tasks | Release time | WCET | deadline | period |
|---|---|---|---|---|
| $T_1$ | 0 | 13 | 80 | 80 |
| $T_2$ | 0 | 6 | 70 | 70 |
| $T_3$ | 0 | 30 | 90 | 90 |
| $T_4$ | 0 | 13 | 110 | 110 |
| $T_5$ | 0 | 26 | 100 | 100 |

current system is feasible since the CPU charge is equal to 0.959. The energy consumption is equal to $2.112J = 2112mW$. The CPU charge factor $U$ was calculated by equation (2) and the energy by the equation (1) previously presented. We assume a reconfiguration scenario by adding 3 additional tasks (table 2). The new system becomes infeasible because the tasks $(T_5, T_8, T_4)$ miss their deadlines and the CPU charge factor is equal to 1.327. The energy consumption is also increased and becomes $2,952J = 2952mW$.

The goal is to ensure that the feasibility of the eight tasks while satisfying the energy constraints. So

Table 2: New system configuration.

| Tasks | Release time | WCET | deadline | period |
|---|---|---|---|---|
| $T_1$ | 0 | 13 | 80 | 80 |
| $T_2$ | 0 | 6 | 70 | 70 |
| $T_3$ | 0 | 30 | 90 | 90 |
| $T_4$ | 0 | 13 | 110 | 110 |
| $T_5$ | 0 | 26 | 100 | 100 |
| $T_6$ | 0 | 10 | 85 | 85 |
| $T_7$ | 0 | 11 | 94 | 94 |
| $T_8$ | 0 | 14 | 105 | 105 |

Table 3: Applied model for WCET reconfiguration.

| Tasks | WCET | New WCET | Start time | Finish time | Deadline | Scaling factor |
|---|---|---|---|---|---|---|
| $T_1$ | 13.00 | 10.40 | 0.00 | 10.40 | 80.00 | 0.80 |
| $T_2$ | 6.00 | 4.80 | 10.40 | 15.20 | 70.00 | 0.80 |
| $T_3$ | 39.00 | 31.20 | 36.20 | 67.40 | 90.00 | 0.80 |
| $T_4$ | 13.00 | 10.40 | 99.60 | 110.00 | 110.00 | 0.80 |
| $T_5$ | 26.00 | 20.80 | 15.20 | 36.00 | 100.00 | 0.80 |
| $T_6$ | 10.00 | 10.00 | 67.40 | 77.40 | 85.00 | 1.00 |
| $T_7$ | 11.00 | 11.00 | 77.40 | 88.40 | 94.00 | 1.00 |
| $T_8$ | 14.00 | 11.20 | 88.40 | 99.60 | 105.00 | 0.80 |

Table 4: New system configuration.

| Tasks | Release time | WCET | deadline | period |
|---|---|---|---|---|
| $T_1$ | 0 | 73 | 80 | 80 |
| $T_2$ | 0 | 65 | 70 | 70 |
| $T_3$ | 0 | 83 | 90 | 90 |
| $T_4$ | 0 | 103 | 110 | 110 |
| $T_5$ | 0 | 96 | 100 | 100 |
| $T_6$ | 0 | 75 | 85 | 85 |
| $T_7$ | 0 | 81 | 94 | 94 |
| $T_8$ | 0 | 100 | 105 | 105 |

we need to modify the scaling factor of the processor to find an optimal combination for the execution of the corresponding tasks. Our model was applied to the system recently presented to resolve the problem. It computes for each task, the start time, the finish time, the scaling factor of the processor , the previous deadline, the new deadline and the new WCET after changing the scaling factors. The results are presented in table 6. Now let us study the case when no available scaling factor can fulfill the needs. We assume that a reconfiguration scenario has been applied and the parameters of the tasks have been updated as depicted in Table 4. Our model tries to adjust the deadlines of tasks as a flexible solution to meet the corresponding requested constraints with a minimum energy consumption. The model was solved with the CPLEX solver Version 12.0. The computational results are shown in table 5.

## 4.2 Heuristic Approach

In this section, we present another method based on heuristic optimization approach that aim to find

Table 5: Applied model for WCET reconfiguration.

| Tasks | WCET | Start time | Finish time | scaling factor | Last dead-line | New dead-line | New WCET |
|---|---|---|---|---|---|---|---|
| $T_1$ | 73.00 | 95.40 | 110.00 | 0.2 | 80.00 | 119.80 | 14.60 |
| $T_2$ | 65.00 | 53.20 | 79.20 | 0.4 | 70.00 | 104.80 | 26.00 |
| $T_3$ | 83.00 | 00.00 | 33.20 | 0.4 | 90.00 | 134.80 | 33.20 |
| $T_4$ | 103.00 | 144.20 | 164.80 | 0.2 | 110.00 | 164.80 | 20.60 |
| $T_5$ | 96.00 | 125.00 | 144.20 | 0.2 | 100.00 | 149.80 | 19.20 |
| $T_6$ | 75.00 | 110.00 | 125.00 | 0.2 | 85.00 | 125.30 | 15.00 |
| $T_7$ | 81.00 | 79.20 | 95.40 | 0.2 | 94.00 | 140.80 | 16.20 |
| $T_8$ | 100.00 | 33.20 | 53.20 | 0.2 | 105.00 | 157.3 | 20.00 |

the appropriate scaling factor and meet a feasible system if possible, if not, the heuristic tries to adjust the deadlines of the tasks so that all the temporal constraints are respected without any energy lost. The processor has a set of available operating speeds such that each task can be executed with its own speed.

**Definition1:**
Let $S$ be a system of $n$ periodic tasks. We denote by $V = (V_i)_{1 \leq i \leq n}$ the vector of speeds where: $V_i$ the execution speed of the task $i$.

**Definition2:**
Let $A$ and $B$ be two vectors of size $n$. We denote $A \leq B$ (reads $A$ is smaller than $B$) if $A_i \leq B_i \forall i$.

**Proposition1 :**
Let $V$ and $V'$ be two vectors of speeds such as $V \leq V'$. If $S$ is feasible with $V$ then it is necessarily feasible with $V'$.

**Proof :**
Let $A$ be the feasible schedule of $S$ under $V$ that meets the following three constraints:

- Release time $r_i$ of the task $i$,
- The worst execution time $c_i$ of the task $i$,
- The deadline $d_i$ of the task $i$.

$A$ is also feasible under $V'$ because firstly, $S$ under $V$ and $S$ under $V'$ have the same release times and secondly, the execution times of the tasks of $S$ in $V'$ are smaller than those in $V$. We assume that the processor has $m$ levels of speeds $V^1 < V^2 < \cdots < V^m$. We denote by $V_{max} = (V^1, \ldots, V^m)$ the maximum execution speed of all the tasks.

**Proposition2 :**
If $S$ is not feasible under $V_{max}$, then $S$ is not feasible under any speed. We assume that all the tasks are activated at the time $t = 0$ i.e $r_i = 0 \forall i$.
The heuristic is based on the following idea: first, we set a maximum speed for all the $n$ tasks, if the system

is feasible (the CPU charge is equal or less than 1), we move to a lower speed and the operation is repeated until the system becomes unfeasible. In this case, the last value of the speed at which the system is feasible is fixed for the first task and the procedure is repeated with $n-1$ remaining tasks. If the system remains unfeasible (the CPU charge is greater than 1), the algorithm tries to increment the deadlines of all the tasks and repeat the last process until reaching the feasibility conditions.

**Algorithm 1:** Scheduling with deadline adjustment.

Inputs :
$n$ number of tasks in the system
$m$ Number of available levels of speed
Outputs :
A schedulable system
Begin :
**repeat**
  $V \leftarrow \{V^1, V^2, \ldots, V^m\}$: set of available speeds
  $T_{init} \leftarrow \{T_1, T_2, \ldots, T_n\}$: initial set of tasks
  $T \leftarrow T_{init}$
  Sort the tasks by increasing order of the priorities
  $i \leftarrow 1; j \leftarrow 1$
  **while** $T$ is not empty **do**
    **repeat**
      Assign speed $V^i$ to all tasks in $T$
      Compute the total charge $U$ of the processor
      Calculate the consumed energy $E$ of the system
      $i \leftarrow i+1$
    **until** $(U > 1$
    **if** $(i > 2)$ **then**
      Assign $V^{i-2}$ to $T_j$
      $i \leftarrow i-1$
      $T \leftarrow T/\{T_j\}$
      $j \leftarrow j+1$
    **else**
      System can not be schedulable
      $T \leftarrow \emptyset$
    **end if**
  **end while**
  Increase the deadline of all tasks
**until** (System is feasible)
End

## Case Study

Let we have a processor with the following set of scaling factor : $\{0.2, 0.4, 0.6, 0.8, 1\}$ and the nominal speed is equal to 1. The processor is charged to execute the following tasks (Table 6). The goal is to affect the suitable processor speed to execute each task in order to achieve a feasible system with low-power energy consumption according to the proposed algorithm. We compute in each iteration, the CPU charge of the processor and the consumed energy. If the available scaling factors of the processor cannot push to a feasible system, the algorithm tries to change the deadlines of the tasks as proposed solution.

Table 6: System configuration.

| Tasks | Release time | WCET | deadline | period |
|---|---|---|---|---|
| $T_1$ | 0 | 73 | 80 | 80 |
| $T_2$ | 0 | 65 | 70 | 70 |
| $T_3$ | 0 | 83 | 90 | 90 |
| $T_4$ | 0 | 103 | 110 | 110 |
| $T_5$ | 0 | 96 | 100 | 100 |
| $T_6$ | 0 | 75 | 85 | 85 |
| $T_7$ | 0 | 81 | 94 | 94 |
| $T_8$ | 0 | 100 | 105 | 105 |

**Iteration1:** We affect the maximum speed $V^1 = 5$ for all the assumed tasks (scaling factor=0.2).

| $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ | $T_7$ | $T_8$ |
|---|---|---|---|---|---|---|---|
| 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |

$U_1 = \frac{1}{5}(\frac{13}{80} + \frac{6}{70} + \frac{30}{90} + \frac{13}{110} + \frac{26}{100} + \frac{10}{85} + \frac{11}{94} + \frac{14}{105}) = 0.265 \leq 1$
► feasible system $E_1 = 67.800mW$

**Iteration2:** We affect the speed $V^2 = \frac{5}{2}$ to all the tasks : (scaling factor=0.4 ).

| $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ | $T_7$ | $T_8$ |
|---|---|---|---|---|---|---|---|
| $\frac{5}{2}$ | $\frac{5}{2}$ | $\frac{5}{2}$ | $\frac{5}{2}$ | $\frac{5}{2}$ | $\frac{5}{2}$ | $\frac{5}{2}$ | $\frac{5}{2}$ |

$U_2 = \frac{2}{5}(\frac{13}{80} + \frac{6}{70} + \frac{30}{90} + \frac{13}{110} + \frac{26}{100} + \frac{10}{85} + \frac{11}{94} + \frac{14}{105}) = 0.530 \leq 1$ ► feasible system $E_2 = 16.950J$

**Iteration3:** We affect the speed $V^3 = \frac{5}{3}$ to all the tasks: (scaling factor=0.6 ). $V^3 = \frac{5}{3}$ for all task :

| $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ | $T_7$ | $T_8$ |
|---|---|---|---|---|---|---|---|
| $\frac{5}{3}$ | $\frac{5}{3}$ | $\frac{5}{3}$ | $\frac{5}{3}$ | $\frac{5}{3}$ | $\frac{5}{3}$ | $\frac{5}{3}$ | $\frac{5}{3}$ |

$U_3 = \frac{3}{5}(\frac{13}{80} + \frac{6}{70} + \frac{30}{90} + \frac{13}{110} + \frac{26}{100} + \frac{10}{85} + \frac{11}{94} + \frac{14}{105}) = 0.769 \leq 1$
► feasible system
$E_3 = 7.533J$

**Iteration4:** We affect the speed $V^4 = \frac{5}{4}$ to all the tasks: (scaling factor=0.6).

| $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ | $T_7$ | $T_8$ |
|---|---|---|---|---|---|---|---|
| $\frac{5}{4}$ | $\frac{5}{4}$ | $\frac{5}{4}$ | $\frac{4}{2}$ | $\frac{5}{4}$ | $\frac{5}{4}$ | $\frac{5}{4}$ | $\frac{5}{4}$ |

$U_4 = \frac{4}{5}(\frac{13}{80} + \frac{6}{70} + \frac{30}{90} + \frac{13}{110} + \frac{26}{100} + \frac{10}{85} + \frac{11}{94} + \frac{14}{105}) = 1.061 \geq 1$
► unfeasible system, ► we allocate the speed$V^3 = \frac{5}{3}$ to $T_1$
$E_4 = 4.237J$

**Iteration5:** We affect the speed $V^4 = \frac{5}{4}$ to the tasks $\{T_2,\ldots,T_8\}$(scaling factor=0.8).

| $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ | $T_7$ | $T_8$ |
|---|---|---|---|---|---|---|---|
| $\frac{5}{3}$ | $\frac{5}{4}$ | $\frac{5}{4}$ | $\frac{4}{2}$ | $\frac{5}{4}$ | $\frac{5}{4}$ | $\frac{5}{4}$ | $\frac{5}{4}$ |

$U_5 = \frac{3}{5}\frac{13}{80} + \frac{4}{5}(\frac{6}{70} + \frac{30}{90} + \frac{13}{110} + \frac{26}{100} + \frac{10}{85} + \frac{11}{94} + \frac{14}{105}) = 1.029 \geq 1$
► unfeasible system, ► we allocate speed $V^3 = \frac{5}{3}$ to $T_2$, $E_5 = 4.200J$

**Iteration6:** We affect the speed $V^4 = \frac{5}{4}$ to the tasks $\{T_3,\ldots,T_8\}$(scaling factor=0.8).

| $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ | $T_7$ | $T_8$ |
|---|---|---|---|---|---|---|---|
| $\frac{5}{3}$ | $\frac{5}{3}$ | $\frac{5}{4}$ | $\frac{4}{2}$ | $\frac{5}{4}$ | $\frac{5}{4}$ | $\frac{5}{4}$ | $\frac{5}{4}$ |

$U_6 = \frac{3}{5}(\frac{13}{80} + \frac{6}{70}) + \frac{4}{5}(\frac{30}{90} + \frac{13}{110} + \frac{26}{100} + \frac{10}{85} + \frac{11}{94} + \frac{14}{105}) = 1.0119 \geq 1$
► unfeasible system, ► we affect the speed $V^3 = \frac{5}{3}$ to $T_3$, $E_6 = 4.207J$

**Iteration7:** We the affect speed $V^4 = \frac{5}{4}$ to the tasks $\{T_4,\ldots,T_8\}$(scaling factor=0.8)
$V^4 = \frac{5}{4}$ for tasks $\{T_4,\ldots,T_8\}$

| $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ | $T_7$ | $T_8$ |
|---|---|---|---|---|---|---|---|
| $\frac{5}{3}$ | $\frac{5}{3}$ | $\frac{5}{3}$ | $\frac{4}{2}$ | $\frac{5}{4}$ | $\frac{5}{4}$ | $\frac{5}{4}$ | $\frac{5}{4}$ |

$U_7 = \frac{3}{5}(\frac{13}{80} + \frac{6}{70} + \frac{30}{90}) + \frac{4}{5}(\frac{13}{110} + \frac{26}{100} + \frac{10}{85} + \frac{11}{94} + \frac{14}{105}) = 0.945 \leq 1$
► feasible system, $E_7 = 4.243J$

**Iteration8:** We affect the speed $V^5 = 1$ to the tasks $\{T_4,\ldots,T_8\}$(scaling factor=1).
$U_8 = \frac{3}{5}(\frac{13}{80} + \frac{6}{70} + \frac{30}{90}) + (\frac{13}{110} + \frac{26}{100} + \frac{10}{85} + \frac{11}{94} + \frac{14}{105}) =$

| $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ | $T_7$ | $T_8$ |
|---|---|---|---|---|---|---|---|
| $\frac{5}{3}$ | $\frac{5}{3}$ | 1 | 1 | 1 | 1 | 1 | 1 |

$1.095 \geq 1$
► system unfeasible, ► we affects the speed $V^4 = \frac{5}{4}$ to $T_4$, $E_8 = 3.522J$

**Iteration9:** We affect the speed $V^5 = 1$ to the tasks $\{T_5,\ldots,T_8\}$(scaling factor=1).

| $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ | $T_7$ | $T_8$ |
|---|---|---|---|---|---|---|---|
| $\frac{5}{3}$ | $\frac{5}{3}$ | $\frac{5}{3}$ | $\frac{5}{4}$ | 1 | 1 | 1 | 1 |

$U_9 = \frac{3}{5}(\frac{13}{80} + \frac{6}{70} + \frac{30}{90}) + \frac{5}{4}\frac{13}{110} + (\frac{26}{100} + \frac{10}{85} + \frac{11}{94} + \frac{14}{105}) =$
$1.0713 \geq 1$
► system unfeasible, ► we affect the speed $V^4 = \frac{5}{4}$ to $T_5$, $E_9 = 4.730J$

**Iteration10:** We affect the speed $V^5 = 1$ to the tasks $\{T_6,\ldots,T_8\}$(scaling factor=1).

| $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ | $T_7$ | $T_8$ |
|---|---|---|---|---|---|---|---|
| $\frac{5}{3}$ | $\frac{5}{3}$ | $\frac{5}{3}$ | $\frac{5}{4}$ | $\frac{5}{4}$ | 1 | 1 | 1 |

$U_{10} = \frac{3}{5}(\frac{13}{80} + \frac{6}{70} + \frac{30}{90}) + \frac{5}{4}(\frac{13}{110} + \frac{26}{100}) + (\frac{10}{85} + \frac{11}{94} + \frac{14}{105}) = 1.0193 \geq 1$
► unfeasible system, ► we affect the speed $V^4 = \frac{5}{4}$ to $T_6$, $E_{10} = 4.745J$

**Iteration11 :** We affect the speed $V^5 = 1$ to the tasks $\{T_7,\ldots,T_8\}$(scaling factor=1).

| $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ | $T_7$ | $T_8$ |
|---|---|---|---|---|---|---|---|
| $\frac{5}{3}$ | $\frac{5}{3}$ | $\frac{5}{3}$ | $\frac{5}{4}$ | $\frac{5}{4}$ | $\frac{5}{4}$ | 1 | 1 |

$U_{10} = \frac{3}{5}(\frac{13}{80} + \frac{6}{70} + \frac{30}{90}) + \frac{5}{4}(\frac{13}{110} + \frac{26}{100} + \frac{10}{85}) + (\frac{11}{94} + \frac{14}{105}) = 0.9958 \leq 1$
► feasible system, ► we affect the speed $V^5 = 1$ to $T_7 and T_8$, $E_{10} = 4.710J$. ► Final solution:

| $T_1$ | $T_2$ | $T_3$ | $T_4$ | $T_5$ | $T_6$ | $T_7$ | $T_8$ |
|---|---|---|---|---|---|---|---|
| $\frac{5}{3}$ | $\frac{5}{3}$ | $\frac{5}{3}$ | $\frac{5}{4}$ | $\frac{5}{4}$ | $\frac{5}{4}$ | 1 | 1 |

Table 7: System configuration.

| Tasks | WCET | deadline | period |
|---|---|---|---|
| $T_1$ | 18 | 100 | 100 |
| $T_2$ | 18 | 100 | 100 |
| $T_3$ | 18 | 100 | 100 |
| $T_4$ | 18 | 100 | 100 |
| $T_5$ | 18 | 100 | 100 |

# 5 NUMERICAL RESULTS

We note that the implemented approaches in this paper to solve the problem of non feasibility of a reconfigurable real-time embedded system, provides more efficient results compared to those in (Chetto and Chetto, 1990; Dwivedi, 2012). Our models allow to compute the scaling factor more than the execution sequence of tasks, the start and the finish time of each task. Our approaches give also good results in the case when no available scaling factor can fulfill the system requirements. To compare our developed approaches(Integer programming approaches IP and Heuristic) to theme presented in (Chetto and Chetto,

Table 8: Comparation between IP ,Heuristic and (Dwivedi, 2012) models for Deadline adjustment.

| | Last WCET | New WCET | | | Last Deadline | New Deadline | | |
|---|---|---|---|---|---|---|---|---|
| Task | IP & Heuristic & (Dwivedi, 2012) | IP | Heuristic | (Dwivedi, 2012) | IP & Heuristic & (Dwivedi, 2012) | IP | Heuristic | (Dwivedi, 2012) |
| $T_1$ | 18 | 7.2 | 7.2 | 18 | 50 | 90 | 85 | 50 |
| $T_2$ | 18 | 7.2 | 10.8 | 18 | 50 | 90 | 93 | 80 |
| $T_3$ | 18 | 7.2 | 10.8 | 18 | 50 | 90 | 102 | 110 |
| $T_4$ | 18 | 7.2 | 7.2 | 18 | 50 | 90 | 64 | 138 |
| $T_5$ | 18 | 7.2 | 7.2 | 18 | 50 | 90 | 108 | 150 |

Table 9: Comparation between integer programming and Heuristic approach.

| | Heuristic approach | | | Integer programming approach | | |
|---|---|---|---|---|---|---|
| basic system+added tasks | Time | Energy | CPU charge | Time | Energy | CPU charge |
| 5+5 | 54ms | 63.31 | 0,9853 | 2.77s | 24.075 | 0,9776 |
| 15+5 | 68ms | 121.78 | 0,8821 | 11.76s | 43.74 | 0,9799 |
| 20+10 | 71ms | 169.26 | 0,99703 | 34.11s | 54.18 | 0,9991 |
| 30+10 | 56ms | 257.72 | 0,9885 | 387.52s | 63.19 | 0,9836 |
| 40+10 | 87ms | 269.13 | 0,9867 | 1706.09s | 153.04 | 0,9831 |
| 45+15 | 97ms | 364.05 | 0,9714 | 1787.23s | 168.17 | 0,9362 |
| 50+20 | 124ms | 353.25 | 0,9809 | 1792.42s | 167.11 | 0,8929 |
| 60+20 | 142ms | 451.93 | 0,9794 | 1801.65s | 163.68 | 0,9652 |
| 70+20 | 178ms | 467.49 | 0,9744 | 1831.46s | 239.24 | 0,9310 |
| 80+20 | 193ms | 557.91 | 0,9827 | 1807.96s | 315.62 | 0,9440 |
| 150+50 | 308ms | 1181.43 | 0,9720 | 1973.39s | 756,81 | 0,9915 |
| 250+50 | 742ms | 1737.40 | 1 | 2051.71s | 1244.15 | 0,8791 |
| 300+100 | 3s | 2201.21 | 0,9251 | 2408.15s | 1423.52 | 0,8636 |



1990; Dwivedi, 2012), we consider a period selection with da deadline to be equal to the corresponding period. The parameters of the task system are depicted in Table 7: By applying our developed approaches to this task system, we obtain the following results: Table 8 shows that our approaches (IP and Heuristic) give results better than works presented in (Dwivedi, 2012) according to the deadline adjustment. In fact, the gap between the last deadlines and the new deadlines is less for our applied methods. In addition, our approaches try to modify the WCET by acting on the scaling factor of the processor. In our experimentation, we have also randomly generated instances with 10 to 400 jobs. The numerical results are depicted in table 9. The first column shows the size of the problem i.e the number of jobs. The sub-column labeled "time" indicates the running time in milliseconds for each method. The sub-column labeled "Energy" gives the total energy consumption. The sub-column labeled "CPU charge" gives the total charge of the processor. For example in Table 9 line 1, for a system composed of 10 tasks(5 initial tasks with other 5 added after a reconfiguration scenario) we obtain $63.31J$ as a consumed energy and a CPU charge equal to $0,9853$ in a time of 54ms by using the heuristic approach. According to the integer programming approach, we obtain for the same OS tasks 24.075 as the consumed energy and a CPU charge equal to $0,9776$ in a time of 2.77s. Table 9 shows that the energy consumption result of the applied integer program is lower than that of the heuristic. However for the large size instances, the heuristic is much faster. We conclude that the integer programming is more efficient for the small instances. Moreover the two approaches guarantee that all the constraints are respected. Figures 1 and 2 present a graphic comparation between the heuristic and the integer programming in term of the CPU charge and the energy consumption. According to the energy consumption, we

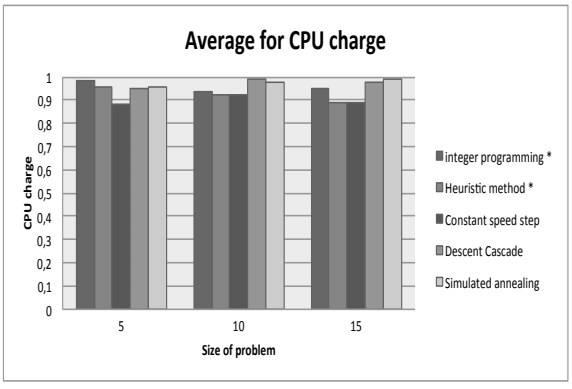


Figure 1: Comparation from CPU charge.

observe in Figure 2 that integer programming is more effective as the number of instances increases because it allows to explore more the search space of solutions and can give a fairly optimal solution. We can observe also that the heuristic is too fast than the integer programming mainly for the large instances Figure 3. In Figure 4, we compare the average CPU charge for the

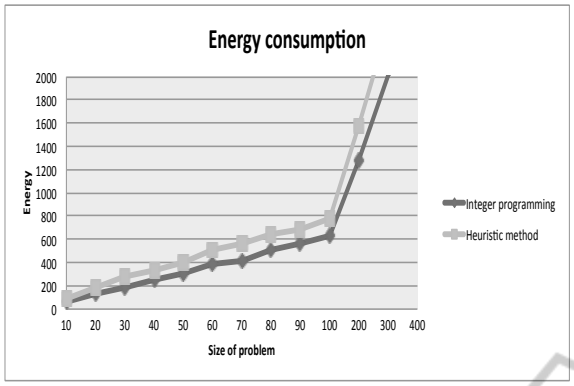


Figure 2: Comparation from energy consumption.

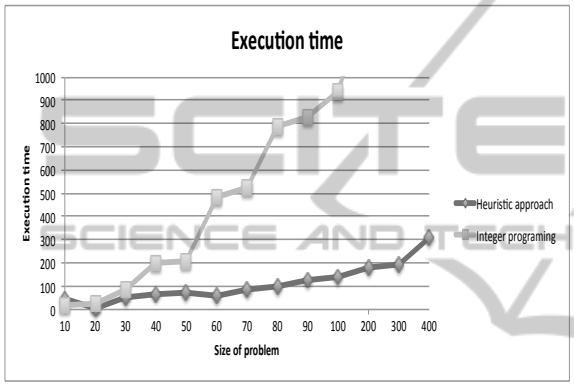


Figure 3: Comparation from energy consumption.

two proposed approaches and those presented as follows in (Jeannenot et al., 2004) on instances of 5 to 15 tasks. The solutions marked with '*' correspond to our proposed approach in this paper, the rest refer to (Jeannenot et al., 2004). The proposed models in (Jeannenot et al., 2004) and(Chniter et al., 2014) try to determine the correspondent scaling factors to ensure a feasible system, yet they they doesn't take into account the energy constraints, in addition, the proposed model may not provide a solution in the case where scaling factors do not allow a feasible system. Our approaches try to exploit the flexibility of the processor to meet the new deadlines of tasks and to minimize the energy cost because in our contribution approaches will work in a reconfigurable real-time embedded system so that the feasibility constraint after a reconfiguration scenario requires more resources of processors.

## 6 CONCLUSIONS

In this paper, We have presented two combinatorial optimization approaches to solve the scheduling

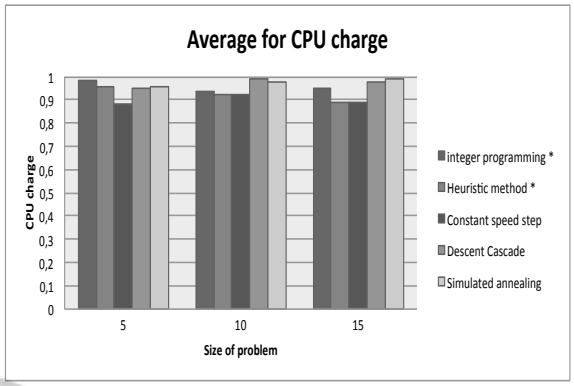


Figure 4: Average CPU charge from each approach.

problem in a reconfigurable real-time embedded system while minimizing the energy consumption. The numerical results show that the integer programing model provides more relevant results than the heuristic approach. However, the heuristic is faster to execute large instances. Globally, the methods give more chance to meet the timing requirements and overcome the failure caused by the rejected tasks. As a future work, our proposed models can be extended to include other constraints such as multiprocessor systems and other criteria such as minimization of the communication between the tasks and can include other categories of tasks such as sporadic and aperiodic.

## REFERENCES

Barr, M. (2007). "embedded systems glossary". *Neutrino Technical Library*, 4(21).

Buttazzo, G. C., Lipari, G., and Abeni, L. (1998). Elastic task model for adaptive rate control. *RTSS*, pages 286–295.

Chantem, T., Hu, X. S., and Lemmon, M. D. (2009). Generalized elastic scheduling for real-time tasks. *IEEE Trans*, Computers 58(4):480–495.

Chetto, H. and Chetto, M. (1990). A feasibility test for scheduling tasks in a distributed hard real-time system. *APII*, pages 239–25.

Chniter, H., Jarray, F., and Khalgui, M. (2014). Combinatorial approaches for low-power and real -time adaptive reconfigurable embedded systems. *International Conference on Pervasive and Embedded Computing and Communication Systems(PECCS)*.

ciçek, I. S. and Celik, C. (2011). Two meta-heuristics for parallel processor scheduling with job splitting to minimize total tardiness. *Applied Mathematical Modelling*, 35(8).

Dalfard, V. M. and Mohammadi, G. (2012). Two meta-heuristic algorithms for solving multi-objective flexible job-shop scheduling with parallel processor and

maintenance constraints. *Computers & Mathematics with Applications*, 64(6):2111–2117.

Dwivedi, S. P. (2012). Adaptive scheduling in real-time systems through period adjustment. *CoRR abs/1212*, (3502).

Fang, K.-T. and Lin, B. M. T. (2013). Parallel-processor scheduling to minimize tardiness penalty and power cost. *Computers & Industrial Engineering*, 64(1):224–234.

Fidanova, S. (2006). Simulated annealing for grid scheduling problem. *In JVA '06: Proceedings of the IEEE John Vincent Atanasoff 2006 International Symposium on Modern Computing*, pages 41–45.

He, C., Zhu, X., Hui Guo, A. Q., and Jiang, J. (2012). Rolling-horizon scheduling for energy constrained distributed real-time embedded systems. *Journal of Systems and Software*, 85(4):780–794.

Heath and Steve (2003). Embedded systems design. *EDN series for design engineers (2 ed.)*, Newnes. p. 2. ISBN 978-0-7506-5546-0.

Heilmann, R. (2003). A branch-and-bound procedure for the multi-mode resource-constrained project scheduling problem with minimum and maximum time lags. *European Journal of Operational Research*, 144(2):348–365.

Hladik, P.-E., Cambazard, H., Deplanche, A.-M., and Jussien, N. (2008). Solving a real-time allocation problem with constraint programming. *J. Syst. Softw*, 81(1):132–149.

Imran Rafiq Quadri, A. G., Boulet, P., Meftali, S., and Dekeyser, J.-L. (2012). Expressing embedded systems configurations at high abstraction levels with uml marte profile: Advantages, limitations and alternatives. *Journal of Systems Architecture - Embedded Systems Design*, 58(5):178–194.

Jeannenot, S., RICHARD, P., and RIDOUARD, F. (2004). Ordonnancement temps réel avec profils variables de consommation d'énergie. *Real-Time Embedded Systems*.

Letters, I. P., editor (1996). *A note on scheduling on a single processor with speed dependent on a number of executed jobs*, volume 297-300.

Liu, C. L. and Layland, J. W. (1973). Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM*, 20(1):46–61.

Nossal, R. (1998). An evolutionary approach to multiprocessor scheduling of dependent tasks. *In 1st International Workshop on Biologically Scheduling of Dependent Tasks, Orlando, Florida, USA*.

PARAIN, F., BANATRE, M., CABILIIC, G., HIGUERA, T., ISSARNY, V., and LSEOT, J. (2000). Tenchiques de réduction de la consommation dans les systèmes embarqués temps réel. *INRIA Research report*, (3932).

Xu, R. (1993). Multiprocessor scheduling of processes with release times, deadlines, precedence, and exclusion relations. *EEE Transactions*, 19(2).

Ying, K.-C. and Cheng, H.-M. (2010). Dynamic parallel processor scheduling with sequence-dependent setup times using an iterated greedy heuristic. *Expert Syst. Appl*, 37(4):2848–2852.

Zhu, Y. (2005). Dynamic voltage scaling with feedback edf scheduling for real-time embedded systems. Master's thesis, North Carolina State University.