UNIVERSITY OF BREST
Lab-STICC

INTERNSHIP

MULTI-OBJECTIVE OPTIMIZATION BETWEEN REAL-TIME AND ENERGY OF A ROV

Students: Nikolina Amižić, Tina Baleta
Mentors: Tonko Kovačević, Marko Vukšić,
Laurent Lemarchand, Frank Singhoff

This work is funded by the ANR «Investissements d'avenir» number ANR-19-GURE-0001 in the framework of the ERASMUS+ SEA UE consortium

Brest, June 2021.

CONTENTS

1. INTRODUCTION

The problems of multi-objective optimization between real-time and energy of a ROV (Remotely Operated Vehicle) will be presented in this paper. ROV is a joint project of University Department of Professional Studies, University of Split and Lab-STICC, University of Brest.

The primary goal of this internship is learning about multi-objective optimization and real-time scheduling. The goal is to optimize energy and time constraints for different possible architectures. In order to achieve this, it is necessary to define a model that allows to evaluate good architectures. These architectures are found by the help of the optimization tools implemented into Cheddar. It is necessary to optimize energy consumption, task priorities and ensure execution in a given time. All of this is made to ensure the optimization of a ROV mission according to energy and time constraints.

First, key concepts such as multi-objective optimization (MOO) and real-time scheduling will be explained, as well as the development environment of the software Cheddar in which the evaluation will be held, and Ada programming language. In the next chapters the process will be thoroughly explained through steps with the display of results. Finally, there will be a conclusion of the work done in the assignment accompanied by comments and guidelines for future work.

2. BACKGROUND

2.1. Multi-objective optimization

Multi-objective optimization is an area of multiple-criteria decision making by concerning mathematical optimization problems involving more than one objective function to be optimized simultaneously [1]. In this assignment it is used to find the optimal task sets for the execution of a mission of a ROV diver.

In the Figure 2.1., the key steps of the multi-purpose optimization project task between real-time and energy are shown graphically.

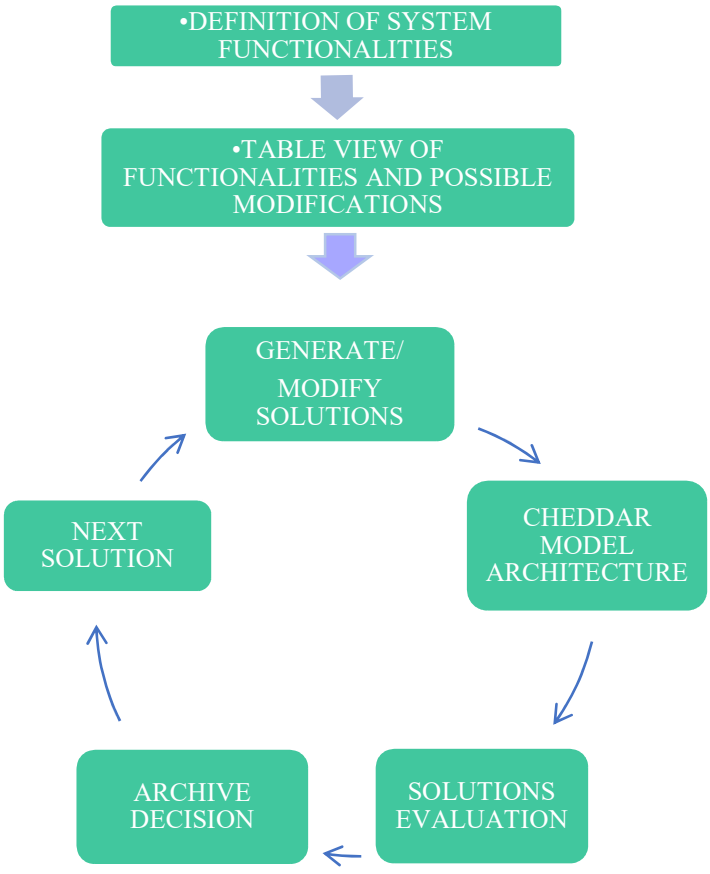

Figure 2.1.: Key steps of MOO

This process is iterative one and it explores different solutions by transforming randomly current one into another somewhat similar, with hopefully improved characteristics. Also, not every solution can be kept. There is dominance concept that is being followed in decision making process, meaning that non dominant solutions are those dominated by solutions with better values of objective functions. Non dominant solutions are thrown away while dominant solutions are kept in the archive. First step is to define all implemented functionalities and determine which are mandatory and which are optional. When this is done, a table or a list with all the functionalities can be made for easier managing. All possible modifications are defined for future evaluations. Finally, the model of the system can be designed and the first solution can be evaluated. The archive is filled with solutions according to the dominance concept. This loop is repeated for all possible solutions. The end product is an archive with a set of optimal solutions that can be used in future missions.

2.2. Real-Time scheduling

Real-Time Systems are those in which the accuracy of the system depends not only on the logical result of computation, but also on the execution time for the results. Therefore, real-time scheduling is a process of scheduling all tasks in a way that all tasks meet their deadlines to make sure the system is correct.

This is made with the help of scheduling algorithms which define how tasks are processed by the scheduling system. Every task must be assigned its description, capacity, deadline and priority so that the scheduler can determine the optimal schedule. Two main groups of schedulers are fixed priority schedulers and dynamic priority schedulers, and the optimal is chosen depending on the tasks specifications. To sum up, the schedule is valid only if all task deadlines are met, and the particular task set is feasible only if a valid schedule can be found for it.

3. DEVELOPMENT ENVIRONMENT

3.1. Cheddar tool

Cheddar is an open-source programming tool used for simulation of real-time scheduling. It was developed by a team of engineers from Lab STICC at the University of Brest in 2002.

It enables designing a model of a software architecture of real-time systems and the evaluation of different task sets with their constraints. Evaluations can be made by simulations or by feasibility tests. It can be done in the user interface or by implementing a part of code in the source code. It is mostly used for analysis during early stages of system development.

3.2. Ada programming language

Ada is a language commonly used when designing critical software systems due to its reliability, efficiency and simplicity. It is unique among programming languages for the help it offers in finding code bugs early in the process when the costs of fixing the bugs are very low. It was originally designed for embedded and real-time systems and for big software systems.

It is based on the code simplicity and automatic checks during coding that reduce the possibility of human errors. This is the reason why it is one of the most suitable languages for critical systems in which even the smallest mistakes can have catastrophic consequences such as aviation industry or space technologies, and many other.

4. PROJECT ASSIGNMENT

In this chapter the steps of the assignment will be introduced and explained.

4.1. Functionalities definition

The functionalities embedded in the Rover are:

- Energy conversion
- Control unit
- Lights
- Thrusters
- VLC communication (Visible Lights Communication)
- Ultrasound communication
- Ultrasound positioning
- CAS system (Collision Avoidance System)
- Sensors
- Object recognition
- Camera

Most of these functionalities are mandatory for the work and with them there is very little to optimize. However, there are two optional functionalities (lights and object recognition) which can be optimized. Also, DVFS (Dynamic Voltage and Frequency Scaling) policy is used in order to decrease energy consumption.

Other functions that need to be defined are the objective functions for optimization. In this case, those are energy and time constraints. Energy constraints are used to ensure the battery energy implemented on the Rover will be enough for the duration of a mission. Time constraints are used to ensure that all tasks will meet their deadlines and the system will work correctly.

4.2. Mission scenario

Mission scenario was made randomly in order to simulate the behavior of the Rover. It involves tasks such as movements to the right, left, straight, up and down, and stop to take an image. This is shown in the Figure 4.2.1.

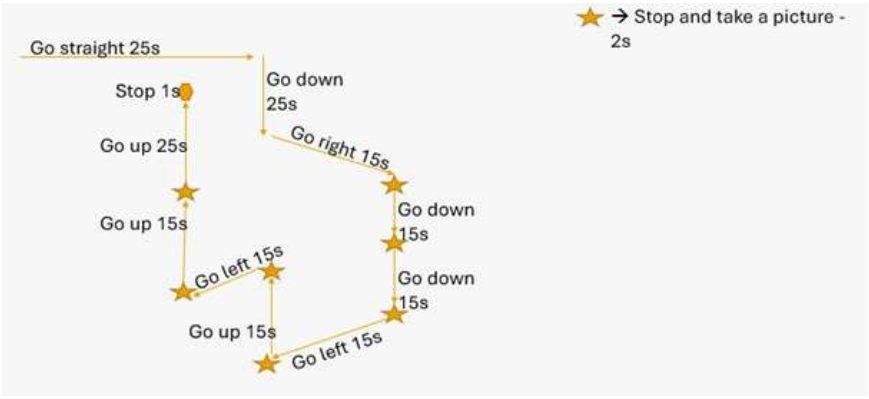

Figure 4.2.1.: Mission scenario used for simulation

According to this scenario, Cheddar model was made with all the tasks implemented in the mission, as shown in Figure 4.2.2. The tasks were defined with their types, capacities, deadlines, start times and priorities.

Name	Task Type	Processor Name	Address Space	Capacity	Deadline	Start time	Priority	Blocking Ti
GPS	Periodic	cpu1	ad1	2	10	0	1	0
data_encrypt	Periodic	cpu1	ad1	5	10	0	1	0
data_send	Periodic	cpu1	ad1	5	10	0	1	0
down1	Scheduling	cpu1	ad1	25	50	50	1	0
down2	Scheduling	cpu1	ad1	15	25	127	1	0
down3	Scheduling	cpu1	ad1	15	25	154	1	0
left	Scheduling	cpu1	ad1	15	25	181	1	0
left2	Scheduling	cpu1	ad1	15	25	234	1	0
object_rec	Periodic	cpu1	ad1	5	10	0	1	0
right1	Scheduling	cpu1	ad1	15	25	100	1	0
stop	Scheduling	cpu1	ad1	2	2	338	1	0
stop_pic	Scheduling	cpu1	ad1	2	2	125	1	0
stop_pic2	Scheduling	cpu1	ad1	2	2	152	1	0
stop_pic3	Scheduling	cpu1	ad1	2	2	179	1	0
stop_pic4	Scheduling	cpu1	ad1	2	2	206	1	0
stop_pic5	Scheduling	cpu1	ad1	2	2	232	1	0
stop_pic6	Scheduling	cpu1	ad1	2	2	259	1	0
stop_pic7	Scheduling	cpu1	ad1	2	2	286	1	0
straight1	Scheduling	cpu1	ad1	25	50	0	1	0
up	Scheduling	cpu1	ad1	15	25	208	1	0
up2	Scheduling	cpu1	ad1	15	25	261	1	0
up3	Scheduling	cpu1	ad1	25	50	288	1	0

Figure 4.2.2.: Task set for the mission defined in Cheddar

The simulation was made to test the feasibility and schedulability of this mission, and the result was that this mission scenario is schedulable. This is shown in Figure 4.2.3.

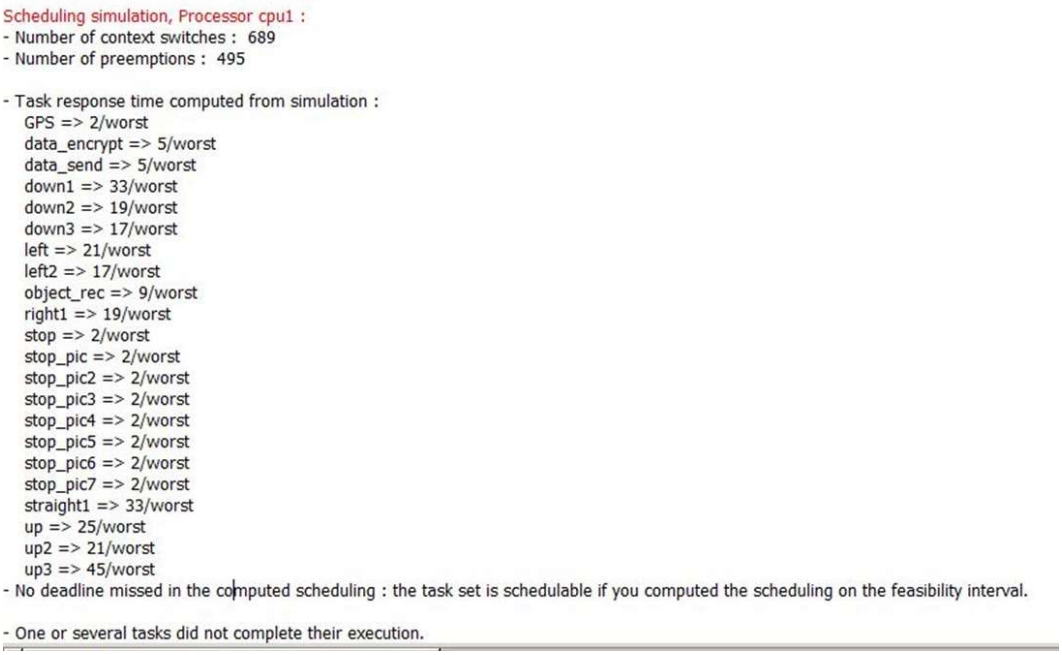

Figure 4.2.3.: Simulation result for this mission scenario

4.3. Energy consumption

The program for energy consumption was written in ADA programming language. It consists of the total of four files, two of which are definition files (.adb) and two are implementation files (.ads). Functionalities with their consumptions according to different frequencies are defined in the definition files, as well as the loop used to calculate the total energy consumption. Implementation files have PAES algorithm and the definition of all possible functions, as well as definition which functions are turned on.

Figure 4.3.1. shows the part of the code used for generating the solution. Types of functions are defined here, as well as all used frequencies. PAES algorithm is shown in Figure 4.3.2.

```
type optional_function is (on, off);
type allowed_frequency is (frequency_0_9, frequency_1_2, frequency_1_6, frequency_2_0, frequency_2_4,

type solution_energy is new generic_solution with record  -- definition of embedded functions

      frequency : allowed_frequency;
      power_conversion : optional_function;
      processor : optional_function;
      lights : optional_function;
      thrusters : optional_function;
      VLC : optional_function;
      ultrasound_comm : optional_function;
      ultrasound_position : optional_function;
      CAS : optional_function;
      sensors : optional_function;
      camera : optional_function;
      object_recognition : optional_function;
    end record;
```

Figure 4.3.1.: Function types and allowed frequencies

```
package paes is

   type generic_solution is abstract tagged
      record
         grid_loc : integer;
         hyperperiod:Integer;
      end record;


end paes;
```

Figure 4.3.2.: PAES algorithm

In the following figures the definition of a single solution (4.3.3.) and the record with energy consumption values for the given frequency (4.3.4.) are shown. In the Figure 4.3.5. the command to call function for energy consumption is shown.

```
begin
        s.power_conversion := on ;
        s.processor := on ;
        s.lights := on ;
        s.thrusters := on;
        s.VLC := on ;
        s.ultrasound_comm := on ;
        s.ultrasound_position := on;
        s.CAS := on;
        s.sensors := on ;
        s.camera := on ;
        s.object_recognition := on;
```

Figure 4.3.3.: Definition of functions turned on

```
rov0_9.power_conversion := 3 ;
rov0_9.processor := 28;
rov0_9.lights := 40;
rov0_9.thrusters := 30;
rov0_9.VLC := 50;
rov0_9.ultrasound_comm := 10;
rov0_9.ultrasound_position := 5;
rov0_9.CAS := 10;
rov0_9.sensors := 10;
rov0_9.camera := 12;
rov0_9.object_recognition := 40;
```

Figure 4.3.4.: Record with energy consumption values for the frequency 0.9 GHz

```
result:=compute_consumption(s, rov0_9);
put_line("energy = " & result'img);
```

Figure 4.3.5.: Command for calling the compute consumption function

In the Figure 4.3.6. the function used to compute total energy is shown. Energy is calculated according to the functions which are on at the given solution, and the functions which are off are not taken in the calculation.

```
function compute_consumption(s : in  solution_energy;
                             r : in rov_consumption) return float is
 result : float := 0;
begin
      if (s.power_conversion = on) then
      result := result + r.power_conversion;
end if;
      if (s.processor = on) then
      result := result + r.processor;
end if;
      if (s.lights = on) then
      result := result + r.lights;
end if;
      if (s.thrusters = on) then
      result := result + r.thrusters;
end if;
      if (s.VLC = on) then
      result := result + r.VLC;
end if;
      if (s.ultrasound_comm = on) then
      result := result + r.ultrasound_comm;
end if;
      if (s.ultrasound_position = on) then
      result := result + r.ultrasound_position;
end if;
      if (s.CAS = on) then
      result := result + r.CAS;
end if;
      if (s.sensors = on) then
      result := result + r.sensors;
end if;
      if (s.camera = on) then
      result := result + r.camera;
end if;
      if (s.object_recognition = on) then
      result := result + r.object_recognition;
end if;

 return result;

 end compute_consumption;
```

Figure 4.3.6.: Function for computing total consumption

The written algorithm is tested in Linux Ubuntu OS with GNAT Programming System. The test is shown in the Figure 4.3.7.

```
tp@ubuntu-VirtualBox:~/proba$ gnatmake test_proba.adb
gcc -c paes.ads
gnatbind -x test_proba.ali
gnatlink test_proba.ali
tp@ubuntu-VirtualBox:~/proba$ gnatbind test_proba
tp@ubuntu-VirtualBox:~/proba$ gnatlink test_proba
tp@ubuntu-VirtualBox:~/proba$ ./test_proba
energy =  238
energy =  755
energy =  1281
energy =  1802
energy =  2324
energy =  2844
energy =  3366
energy =  3372
```

Figure 4.3.7.: Algorithm test in Linux Terminal

4.4. Extreme cases

Extreme cases represent the cases in which the optimization is most important because if those cases are optimized, then all the cases which are in between these will be optimized too. The cases chosen in this case represent those with all optional functionalities off for the lower extreme, and those with all optional functionalities on for the higher extreme. The frequencies used for testing these cases are all defines frequencies: 0.9GHz, 1.2GHz, 1.6GHz, 2GHz, 2.4GHz, 2.9GHz, 3.5GHz and 4GHz.

Total number of these cases is 16, and energy consumption and WCET (worst case execution time) were calculated for all of them. With those calculations, .csv file were made in order to draw diagrams with the results. Example of such diagram for 4 frequencies (0.9GHz, 1.6GHz, 2.4GHz, 4 GHz) and 8 cases is shown in Figure 4.4.1.

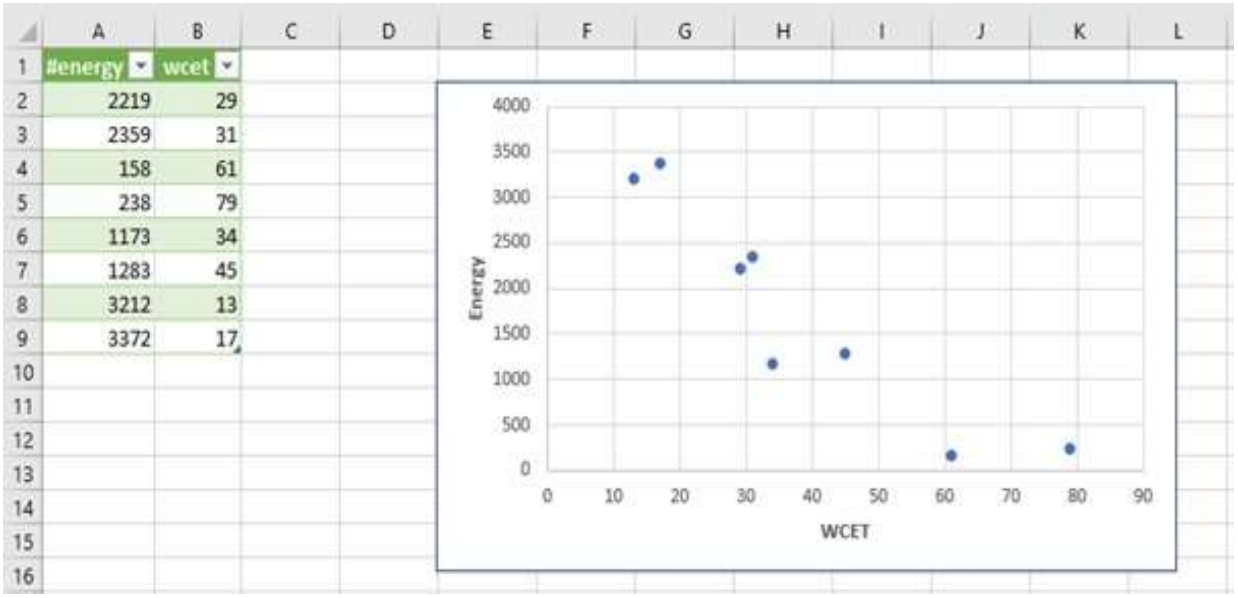

Figure 4.4.1.: Example of a .csv file with calculations and diagram for 4 frequencies

In this diagram, Pareto front formed with these cases is visible (marked in Figure 4.4.2.), and it shows the dominant solutions.

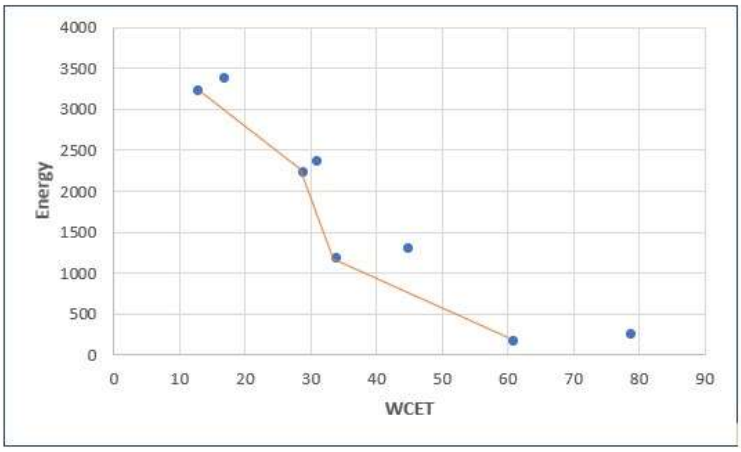

Figure 4.4.2.: Pareto front for 4 frequencies

5. CONCLUSION

During this internship, the basics of multi-objective optimization and real-time scheduling have been covered, with the focus being on PAES algorithm. Skills in Ada programming language and modelling with Cheddar simulation tool were achieved through the assignments of this project. The work that has been done is explained through writing a research paper.

The main objective of this project was multi-objective optimization between real-time and energy constraints. Real-time constraints were dealt with by modelling ROV diver architecture using Cheddar tool and evaluating possible mission solutions with the same tool, in order to get an archive with optimal solutions for future missions. ADA program was written in order to explore energy constraints of solutions and make sure that total energy consumption stays within the hardware limits.

The process of automating the evaluation into an optimization process using PAES is not yet finished. The part of the algorithm which automatically evaluates both energy and time constraints is still in the development process and should be ready to summarize this project in the near future.

LITERATURE

[1] MOO, https://www.sciencedirect.com/topics/engineering/multiobjective-optimization, [Last approach 15.4.2021.]

[2] Cheddar, http://beru.univ-brest.fr/cheddar/, [Last approach 15.4.2021.]

[3] Ada, https://www.adacore.com/about-ada, [Last approach 15.4.2021.]

[4] Real-Time scheduling, http://beru.univ-brest.fr/~singhoff/ENS/USTH/sched.pdf , [Last approach 20.4.2021.]

[5] Pareto Front, https://www.researchgate.net/figure/Dominated-non-dominated-and-Pareto-front-solution-set_fig2_310835567 [Last approach 28.5.2021.]