UNIVERSIDADE
BEIRA INTERIOR

Multi-objective meta-heuristic sleep scheduling in low-power devices

Versão final após defesa

Tiago André Gaspar Nunes Rodrigues

Dissertação para obtenção do Grau de Mestre em
Engenharia Eletrotécnica e de Computadores
(2º ciclo)

Orientador: Prof. Dr. António Eduardo Vitória do Espírito Santo

Abril de 2023

Declaração de Integridade

Eu, Tiago André Gaspar Nunes Rodrigues, que abaixo assino, estudante com o número de inscrição M11188 de/o Engenharia Eletrotécnica e de Computadores da Faculdade de Engenharias, declaro ter desenvolvido o presente trabalho e elaborado o presente texto em total consonância com o **Código de Integridades da Universidade da Beira Interior**.

Mais concretamente afirmo não ter incorrido em qualquer das variedades de Fraude Académica, e que aqui declaro conhecer, que em particular atendi à exigida referenciação de frases, extratos, imagens e outras formas de trabalho intelectual, e assumindo assim na íntegra as responsabilidades da autoria.

Universidade da Beira Interior, Covilhã 24/04 /2023

Agradecimentos

Agradeço à família por garantir todo o suporte necessário para poder percorrer o meu caminho até este ponto. Agradeço aos professores que de quaisquer maneiras contribuíram para a minha formação e também pelo seu importantíssimo ensino de valores que me fazem hoje quem eu sou. Ao Prof. Doutor António Espírito Santo, que esteve sempre à disposição, me auxiliou na escrita deste trabalho e me garantiu todos os recursos necessários para a sua finalização. Agradeço também a todos os meus amigos próximos que sempre acreditaram em mim e em especial à Matilde Lopes, pelo amparo e palavras motivadoras.

Resumo

Técnicas de colheita de energia e redes de sensores sem fios demonstram uma ótima simbiose, uma vez que o ambiente típico onde os nós estão dispostos, na maioria das vezes apresentam características não favoráveis à operação e manutenção humana, maioritariamente são localizados em lugares remotos de difícil alcance e com impraticabilidade de operação, isto torna a possibilidade de reciclar as baterias, ou qualquer outra peça de hardware, laboriosa ou até mesmo inexequível, o que requere um extensivo uso de recursos. Com a aplicação de módulos de colheita de energia e algoritmos de otimização, é possível alcançar a condição de operação neutra de energia, tornando os nós em dispositivos autónomos capazes de funcionar durante anos, sem interrupções ou intervenção humana. Algoritmos meta-heurísticos são aplicados nestes cenários, onde uma solução ótima global é necessária para o nó operar sob um esquema energeticamente eficiente, o que torna estes sistemas em completas black-boxes. Ao ter em conta o máximo possível de variáveis físicas relacionadas com o problema, e ao exercer um rigoroso esquema de controlo é possível apontar como saída do sistema um modo ideal de operação, comutando, por exemplo, o ciclo de operação, a variação de tensão e frequência ou até mesmo a inserção de metodologias que decompõem tarefas presentes no processador, em múltiplas tarefas menos complexas. Nesta dissertação um algoritmo de calendarização, baseado na meta-heurística, é estudado, criado e demonstrado, assim como um sistema é arquitetado com base em um microcontrolador LPC1768 de maneira a simular um nó de uma rede sensor sem fios, uma extensa revisão de técnicas do estado-da-arte usadas em dispositivos EH-WSNs é também realizada.

Palavras-Chave

Colheita de energia;meta-heurística;redes sensores sem fio;ciclo de operação;decomposição de tarefas;variação de tensão e frequência

Abstract

Energy Harvesting techniques and Wireless Sensor Networks evidence an optimal symbiosis, given that the typical environment where the motes are placed are most times remote, hard to reach, and with poor operational capabilities, this makes the possibility of changing batteries or another piece of hardware arduous and laborious, requiring great resource exhaustion. With the appliance of EH modules and optimization algorithms, it is possible to reach an Energy Neutral Operation condition, which turns these motes into self-autonomous devices capable of operating for years without disruption and human intervention. Meta-heuristic algorithms are applied in these scenarios, where a global optimum solution is required for the mote to operate under a rigorous and efficient energy expenditure scheme, turning these systems into complete black-boxes, inputting the maximum possible physics variables, and outputting the ideal mode of operation, commuting, for example, duty cycle, voltage and frequency variation or even devising methodologies that decompose one extensive task into multiple, easier to process, divided tasks. In this dissertation, a sleep scheduling algorithm, based on MHs, is studied, devised, and demonstrated. As well as a system being architected based on an LPC1768 microcontroller in order to simulate a mote of a WSN, an extensive revision of the state-of-art techniques, used in EH-WSNs is also carried out.

Keywords

Energy-harvesting;meta-heuristic;wireless sensor networks;duty cycle;task decomposition;dynamic voltage and frequency scaling

Index

List of Figures

List of Tables

Acronyms List

ACK	Acknowledgment Frame
ADC	Analog-Digital Converter
AI	Artificial Intelligence
CMOS	Complementary Metal-Oxide-Semiconductor
COS	Computing Oriented Scheduling
CQ	Custom Queue
DAC	Digital-Analog Converter
DC	Direct Current
DE	Differential Evolution
DMA	Direct Memory Acess
DPM	Dynamic Power Manager
DRAM	Dynamic Random-Access Memory
DVFS	Dynamic Frequency Voltage Scaling
DVS	Dynamic Voltage Scaling
EA	Evolutionary Algorithm
EDD	Earliest Due Data
EDF	Earliest Deadline First
ENO	Energy Neutral Operation
EPWSN	Environmentally Powered Wireless Sensor Network
FA	Firefly Algorithm
FIFO	First In First Out
FPGA	Field Programmable Gate Arrays
FQ	Fair Queuing
GPIO	General Purpose Input/Output
GSA	Gravitational Search Algorithm
HW	Hardware
IGSA	Intelligent Gravitational Search Algorithm
IoT	Internet of Things
ISR	Interrupt Service Routine
LAN	Local Area Network
MAC	Media Access Control
MH	Meta-Heuristic
MOEA	Multi-Objective Evolutionary Algorithm
MOO	Multiple Objective Optimization
MOP	Multiple Optimization Problem
MPP	Maximum Power-Point
NOP	No-Operation
NSGA	Non-Dominated Sorting Genetic Algorithm
OS	Operating System
p.d.u	Procedure defined unit
PLL	Phased-Lock Loop
PQ	Priority Queue
PSO	Particle Swarm Optimization
PSU	Power Supply Unit
PWM	Pulse Width Modulation
QoS	Quality of Service
R/W	Read/Write
RAM	Random Access Memory
RF	Radio Frequency
RM	Rate Monotonic
RTC	Real-Time Clock
SHM	Structural Health Monitoring

TCP	Transmission Control Protocol
TX/RX	Transmit/Receive
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
VEGA	Vector Evaluated Genetic Algorithm
WDT	Watchdog Timer
WEDF	Weighted Earliest Deadline First
WFQ	Weighted Fair Queuing
WSN	Wireless Sensor Network

Chapter 1

Introduction

Since ancient times, humans used energy from the environment to their advantage, from making fire to sailing on the seas, it was always present, side-by-side with evolution. In the last decades, especially since transistors appeared as a new technology, energy started to prove useful in ways humans never thought it could aid, energy was transformed into electricity to automate tedious day-to-day tasks and even eliminate the need for human labor, in some specific instances. This led to the fast progression of technology, and eventually to achieve the harvest of clean energy, without the use of fossil fuels, availing the energy directly derived from the sun and other natural sources.

Low-power devices benefit from energy harvesting since they can operate autonomously, for long periods without the need for technical intervention. In WSNs there are multiple low-power devices, named motes, composing the network itself, which improve when EH techniques are present. WSNs have a potential application in diverse areas, in environment health, for example tracking and monitoring fire-hazard zones; medicine, by identifying and monitoring diseases inside the body; military, by tracking and identifying military activity and hazard zones; urban, by tracking, for example, the quality of buildings infra-structures; industrial, by monitoring work capacity and execution, etc.

With the prospect to keep WSN motes within the energy neutral operation (ENO) condition, their lifetime is increased as well as the operation is kept 'green'. To achieve this condition, there is not only the necessity of having EH capabilities but there is also the need to keep processes, tasks, and operation modes, within the motes, optimized. MH algorithms help to achieve the ENO condition.

In the following dissertation, all these subjects will be focused on and studied thoroughly to present a valid implementation of a proposed sleep scheduling algorithm.

1.1. Work motivation

In this dissertation, one of the most important challenges is to identify the main variables involved in the concept of sleep schedule in a high-performance ARM Cortex-M3 core, located on an NXP LPC1768 MCU, using Mbed OS firmware. Finding these variables requires the need to identify the weight of each one related to the efficiency of execution in a running process. Nowadays there are several ways to optimize this process in order to have a higher energy load, extended operation lifetime, and a more trustable performance, the main objective is to use literature-reviewed meta-heuristic algorithms to insert these behaviors in a low-level choice of sleep scheduling and duty cycle variance, based on energy availability. Energy efficiency in

embedded systems is useful in a wide number of fields, from the creation of clusters in WSNs to the enormous field of IoT or even in areas where a trustable mode of execution is whereas time and energy are at play. In the course of the current dissertation, a critical evaluation will be done on the performance of the chosen MH algorithm, as well as a debate between all presented MH algorithms to filter which one is more compatible with the problem involved.

1.2. Main goals

The main goals of the present work are to architect a system based on the LPC1768 board, which simulates a typical WSN mote. A metaheuristic algorithm is also to be coded in order to establish an energy-efficient scheme based on the architected system, where it will be a target to schedule the sleep/awake routine of a perpetual process, the system will utilize resources to commutate between operation modes, based on output results from the algorithm. To follow this course of action, first, there needs to be a concise understanding of the wide variation of MH algorithms, to acknowledge which ones are viable for the application at hand and why. Given the high volume of scientifically reviewed MH algorithms, a cautious choice needs to be carried out, some perform great at a given problem and may fail at another instance, so the objective is to review, evaluate and compare various sets of these algorithms, and conclude their effectiveness on the optimization performance.

Then, to check where the MH algorithm output must act in the system, an extensive review of state-of-art scheduling methodologies that seek ENO conditions, are to be analyzed and discerned. When both subjects are grasped, a comprehension of the mbed OS is to be made in order to utilize its resources to the optimal and establish a bridge between the MH algorithm, running on a remote server, and the board, running on the field. Lastly, a validation of the MH algorithm implementation is to be done regarding a critical analysis to check where the optimization has played its role, and in which ways it can enhance the performance of the overall system.

Other relevant mentions, about energy saving in WSNs motes, are also discussed throughout the dissertation, there are several ways to optimize the energy usage in this kind of problem, other than commutation of sleep/awake routines. Literature reviews acknowledge different approaches, like multi-modal data fusion techniques [1] some traditional scheduling algorithms like FIFO, PQ, CQ, FQ, and WFQ used in multimedia WSN [2], amended dynamic round robin scheduling for timeshared systems [3] and many more, that will be discussed later. Applying MH algorithms at this level is a new approach that the academic field may benefit from, since there is insufficient work on the subject, various approaches are needed given that it is a volatile field, where there is not a single apex method of optimizing a mote operation within its network.

1.3 Documentation structure and organization

The structure of this dissertation is represented in the following figure, fig. 1.1. Chapter 1 introduces the reader to the structure and essence of the work while explaining the motivation and main goals behind it, chapter 2 summarizes metaheuristic concepts and reviews a wide range of algorithms, chapter 3 does an extensive review on state-of-art scheduling methodologies performed in typical WSNs motes as well as proposing an overall system architecture, chapter 4 introduces the reader to the Mbed OS platform while explaining how it interconnects with the whole system, in chapter 5, the reader will find the implementation of the selected algorithm, a brief overview of system low power capabilities along with energy tests and algorithm output results, in the last chapter, chapter 6, a conclusion of the present work is found, briefing on the main achievements and proposed future works.

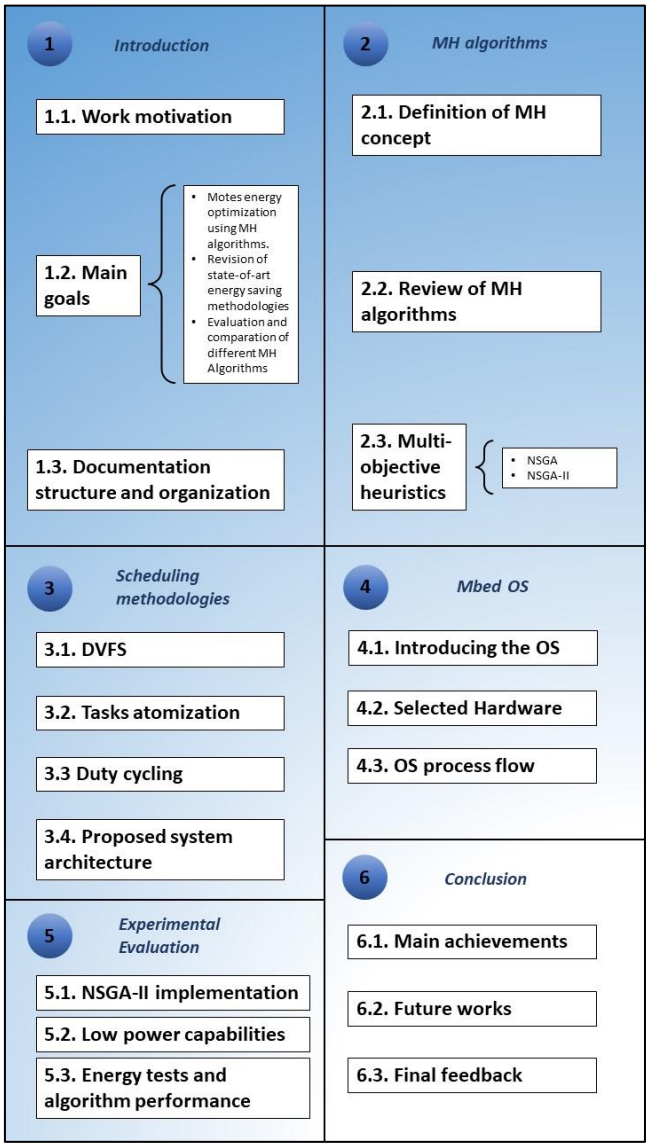

Figure 1.1 – Chapter roadmap.

Chapter 2

Meta-heuristic algorithms

All study fields that involve science and technology eventually require problem-solving, and for a problem to be solved there needs to be a set of adequate solutions. The less complex problems, with a low number of variables and minimum entropy, are easily solved by the human mind, resorting to the rational part of our brains, but when, in these fields, a problem with a high-level of complex variables, conditions and high entropy appears, an immediate solution is not easily found.

With the use of computing power, meta-heuristic algorithms help identify an appropriate set of approximate and non-deterministic solutions for the latter kind of problems, these algorithms compose strategies that seek to guide the search process in an objective space, which can range from a single dimension to infinite dimensions, depending on the type of problem and computational power at hand.

The word meta-heuristic was first mentioned in 1986 by Glover's work [4], but it has a much older presence, remarking works from 1951 [5], and even to these days it is still relevant to areas such as engineering, economics, informatics, biology, etc.

In the current chapter the concept of MH will be clarified, as well as some of the most popular algorithms will be explained and visualized. In the final section a multi-objective MH algorithm, NSGA-I, and NSGA-II will be also demonstrated since they are relevant to the present work.

2.1. Definition of MH concept

The concept of heuristic says that it is a pedagogical process where a certain element discovers for itself what it wants to teach and retain through questions to itself, meta-heuristic algorithms exercise the same concept, they are algorithms that vision the optimization of a determined mathematical process through the creation of a particle population, this population tries to find the optimal solution to the given problem. Usually, these algorithms are based on natural, social, or physical behavior [6]. As previously mentioned, optimization processes are essential, in the majority of fields, from engineering to economy or even to vacation planning and time-saving protocols, given that money and finite resources are limited to the human being it is therefore crucial that they be used optimally. Most real-life optimization problems are highly non-linear and multimodal, this is, they have various complex and stochastic restrictions that prevent the forecast of results, thus in those circumstances optimization may be hard to attain.

Firstly, for the resolution of these problems, there needs to be efficient research of the existent MH algorithms, this is because currently, there is a wide range of optimization algorithms that

are classified in various ways, depending on their focus and characteristics. The most common split in optimization algorithms is made in two types, deterministic or stochastic, in other words, they are predictable or randomized. Meta-Heuristic algorithms fall over the second type, that is, because all of them have one or more agents that generate randomness in their searches, enabling the particles to explore the search space without having the problem of converging into a local optimum. An analogy that is commonly used, is the wolf pack hunting analogy, imagine that a pack of wolves hunts within a 1km radius area, the pack can look for food and find potential prey near their base, which they decide to hunt upon, getting sufficient food for three wolves, or else, they can choose to search the whole area and find the global optimum, a wildebeest herd, for example, hunting upon the global best will now give food for the whole pack, in contrast to feeding only three wolves.

Some of the most used and popular MH algorithms are reviewed in the current chapter, these are PSO, FA, DE, GSA, and hybrid, these algorithms will be addressed in the next section based on their performance and utility, in section 2.3 the concept of multi-objective optimization is introduced as well as a discussion about how MH algorithms perform in scenarios where a trade-off is required, for example, by minimizing energy consumption in a WSN mote and maximizing task execution at the same time.

2.2. Review of MH algorithms

2.2.1. Gravitational search algorithm

GSA is a physics-based metaheuristic algorithm that was inspired by Newton's laws of gravity and motion, created back in 2009 by Rashedi et al. [7]. In GSA all the particles have a behavior based on their position, inertial mass, gravitational mass, and intrinsic velocity. Each new position of the particles represents a solution with a determined fitness value, given by the following equation (2.1).

$$P_i(it+1) = P_i(it) + V_i(it+1) \tag{2.1}$$

Where $P_i(it)$ is the particle at iteration it and $V_i(it+1)$ is the updated velocity of i^{th} particle, given by equation (2.2).

$$V_i(it+1) = rand \times V_i(it) + acc_i(it) \tag{2.2}$$

Here, $rand$ is a random value between 0 and 1, and $acc_i(it)$ is the acceleration of particle i at the it_{th} iteration, acceleration is given by the current force of the particle divided by its current mass. The force of a given particle at a given moment is calculated by equation (2.3).

$$F_{ij}(it) = G(it) \times \frac{M_{passive(i)}(it) \times M_{active(j)}(it)}{R_{ij}(it) + \varepsilon} \times (p_j(it) - p_i(it)) \quad (2.3)$$

G is a gravitational constant, $M_{passive}$ is the passive gravitational mass of the particle i, M_{active} is the active gravitational mass of particle j and R_{ij} is the Euclidian distance between particles i and j.

The fitness of a specific particle is weighted by its mass and the heaviest particle will be the one that has the best fitness value, thus creating attracting forces to all the other particles of the population, the particle position update will be the sum of all forces exercised on it, figure 2.1 demonstrates that effect.

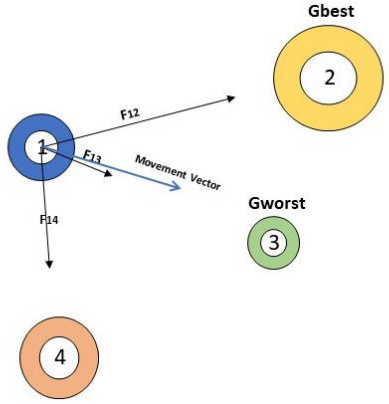

Figure 2.1 – Particle position update by GSA.

The pseudocode of this algorithm is introduced next.

GSA Pseudocode

1. Declaration of all variables;

2. Initialization of population with particles (p);

3. **while** stopping criteria is not met, **do**

4. Calculation of fitness, G, and masses;

5. Calculate K_{best} (Best particle from current iteration);

6. Determine acceleration (acc) and velocity (V);

7. Update the particles by equation (1);

8. **end while**

Studies show that GSA is a highly relevant meta-heuristic algorithm due to its performance when dealing with complex optimization problems, nevertheless, it presents some flaws, more specifically convergence of particles and confinement at local minima [8].

2.2.2. Differential evolution

DE is a biology-based metaheuristic algorithm, that introduces the concept of evolution of its search particles, it does this by the resource of mutation and crossover agents, considered to be one of the most effective MH algorithms [9], DE is one of the MH algorithms with the most variants, varying in the type of mathematical selection of a given target vector, the number of differentials and the type of crossover used, in this dissertation, it will not be possible to synthesize all of them, so the discussion will be only about the implement of $DE/rand/1/bin$ algorithm, 'DE' standing for differential evolution, 'rand' makes a random selection of a given vector, '1' is the number of differentials and 'bin' indicates that the crossover agent utilizes binary search.

The efficiency of DE has been approved due to its universal appliance to any optimization problem, linear or nonlinear, combinatory or continuous, or even multi/mixed variable problems, even so, DE was originally meant for difficult optimization on continuous space, and it performs best in those cases [9]. In the next figure, fig. 2.2, the reader can observe all the areas where DE has been applied successfully.

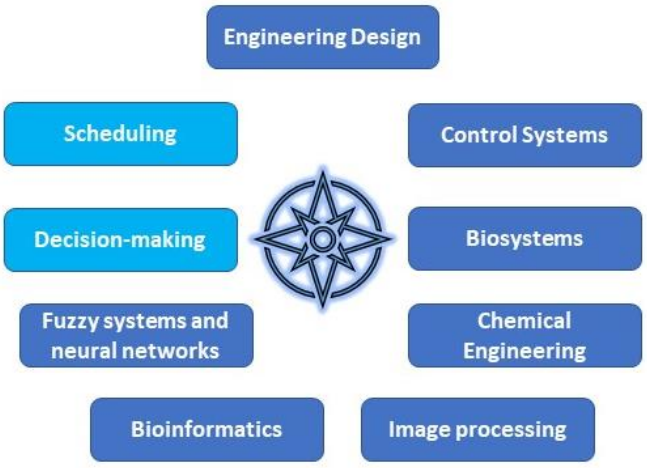

Figure 2.2 – Areas where DE was successfully applied.

The light blue rectangles are the areas where the final proposed algorithm on the current work will stand by.

A brief explanation, accompanied by a flowchart, of the DE MH algorithm is given in the forward text.

Before starting the DE algorithm, there is the need to establish some fixed variables that will have an important role at the elapse of the algorithm, these are, mutation constant F, crossover probability Cr_{Prob}, population size NP, problem dimension dim, number of iterations max_{it} and boundary limits lim_{inf} and lim_{sup}. The greater F is, the more weight is added to the mutation phase, and the greater the Cr_{Prob} is the more likely it is for the crossover process to be applied to a given particle.

Following the course of action of other MH algorithms, the DE algorithm starts with a randomized population in a determined search area given the objective function at hand, after this, it determines the fitness of each particle and updates their positions based on the movement formula, position update is carried by two operators, mutation and crossover, it is different from other evolutionary algorithms where the solution is mutated after the crossover, in this example the mutation of the particle is firstly applied, and only then, crossover.

The mutation operator is used to create a Trial Vector, for each particle (Parent Vector). It is done by mutating a Target Vector, equation no. (2.4), for each dimension.

$$Trial\ Vector = Target\ Vector + F \times (Randomly\ selected\ solution_1 - Randomly\ selected\ solution_2) \tag{2.4}$$

It should be noted that the indexes for randomly selected solutions, Target Vector and Parent Vector should be different from each other. After the mutation phase, the algorithm checks if the particles are within boundaries, and if not, it repositions them at the limited boundaries.

In the DE crossover phase, an offspring is generated by using the discrete recombination of Parent Vector and Trial Vector, given the crossover probability Cr_{Prob}. Before the generation of the offspring, the algorithm uses binomial crossover 'bin', where it selects some crossover points that are selected from the set $\{1, \dots, dim\}$ with Cr_{Prob}. This may lead to the situation where no point is selected, if this is to happen then there will not be any change in the offspring, and the particles will be equal to the parent, see equation (2.5), so I is always considered as non-empty set by including a random point from the set $\{1, \dots, dim\}$ initially.

$$Offspring\ X'_{1j} = \begin{cases} Trial_{1j}, & if\ j \in I \\ Parent_{1j}, & otherwise \end{cases} \tag{2.5}$$

Where X is a given particle, j is the current dimension and I is the crossover set.

After the generation of the offspring, the algorithm will compare each offspring's fitness with its parent relative and check which one of both is the fittest, the best ones are continuing to the next iteration, and the others are eliminated from the population, this step, in essence, is what gives optimization to the process.

In the next figure, fig. 2.3, the flowchart for $DE/rand/1/bin$ MH algorithm is presented.

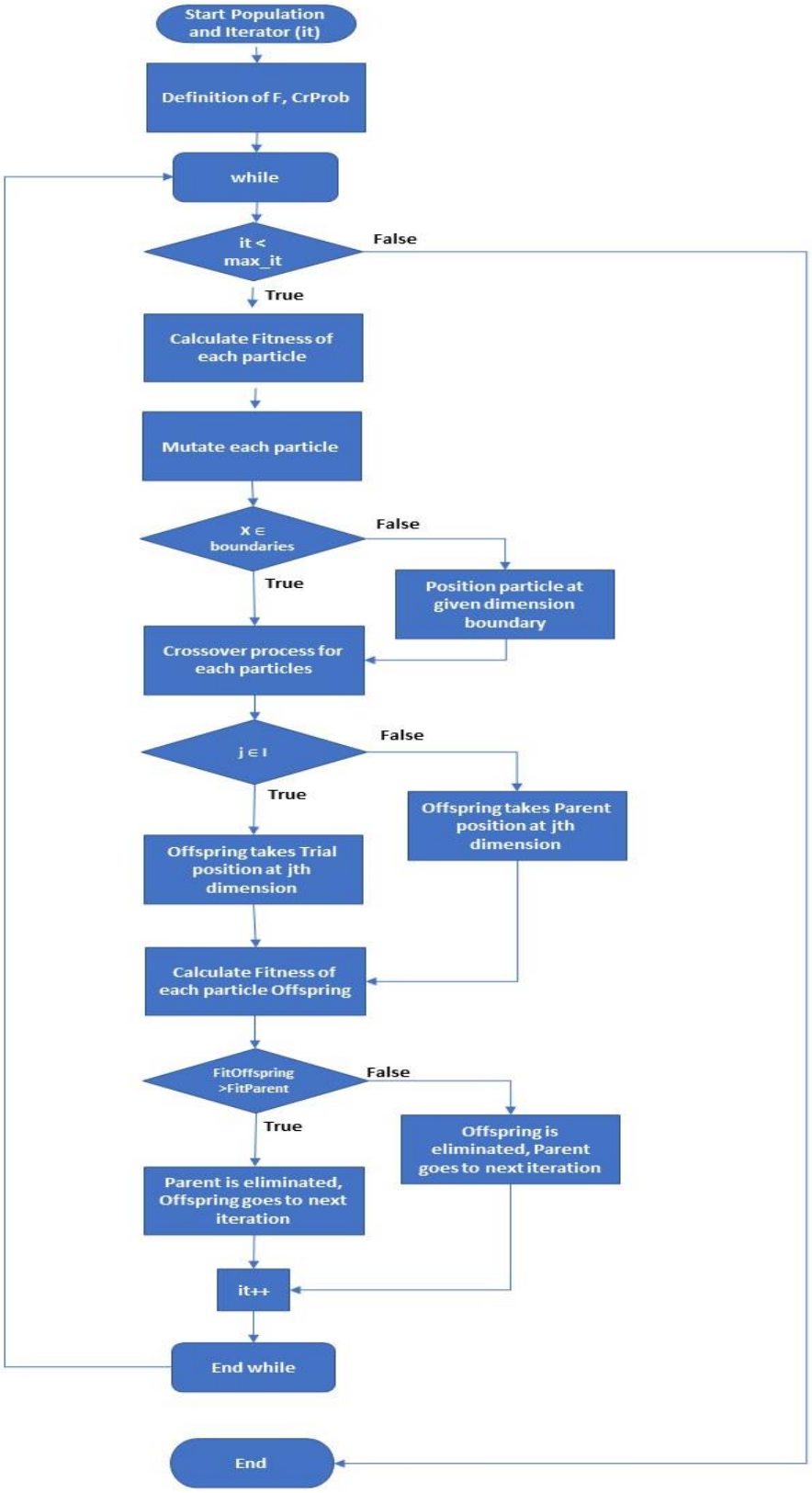

Figure 2.3 – Flowchart of DE/rand/bin/1.

2.2.3. Particle swarm optimization

PSO is one of the most well-known metaheuristic algorithms, that is because it was one of the pioneers, published in 1995 by Kennedy et al. [10], it showed that a simple algorithm could be effective in the optimization of a wide range of functions, an interesting fact about this study is that it was led by a social psychologist and an electrical engineer, that is because in this algorithm there is the presence of a flock social behavior, that influences where the particles are headed. After this paper was released, some years later, in 2007, it was updated by the same author [11]. In PSO all the particles communicate with each other by two main parameters, personal trust (c_1) and global trust (c_2), c_1 adds weight to the best particle of the current iteration and c_2 adds weight to the best particle of the whole set of iterations, usually, these parameters are used to calculate the velocity of a given particle, and have a random number attached to them to turn them into varying values, see equation (2.6).[12]

$$v_{k+1}^{i} = w \times v_k^i + (c_1 \times r_1 \times (Pbest^i - x_k^i)) + (c_2 \times r_2 \times (p_k^g - x_k^i)) \tag{2.6}$$

Where v is the vector velocity, w the inertial weight, that usually decreases over the iterations to values lower than unity, both r's are random generated numbers, P_{best} is the best particle of the current iteration, G_{best} is the best global particle and x is the position of i_{th} particle.

After attaining the velocity vector, the position of the particles must be updated by equation (2.7).

$$x_{k+1}^{i} = x_k^i + v_{k+1}^i \tag{2.7}$$

So, the following PSO original algorithm exhibits the following execution of steps.

PSO Pseudocode

1. Declaration of all variables;

2. Initialization of population with particles (p);

3. **while** stopping criteria is not met, **do**

4. Generation of r_1 and r_2 and update w;

5. Calculation of P_{best} and evaluation of G_{best};

6. Calculate a velocity vector for each particle in the population, using equation (6);

7. Update the position of each particle using equation (7);

8. **end while**

In venter et al. [12] the author claims that the results that were led by the PSO algorithm and variants were able to deal with both unconstrained and constrained, as well as the continuous and discrete design problems, and even showed great performance when dealing with problems with many local minima or maxima, but despite the fact that these results were very accurate, PSO suffered from a high computational cost, at least higher than that of traditional gradient-based optimizers.

2.2.4. Firefly algorithm

The FA, introduced in 2009 by Yang et al. is based on the behavior of fireflies, they emit a bright light via a process of bioluminescence, that it is used to mate with partners and repel potential prey, in addition, the flash may also serve as protective warning mechanism[13]. This phenomenon is used in this algorithm by making a direct proportionality between the brightness of a firefly with its fitness, this means that a less bright firefly will move towards a brighter firefly. Distance is a constraint to this approach, given that the farthest away a particle is from the other one, the less bright it will perceive, because of the diffusion of light through air. All these rules are presented in the following equations:

$$\beta(r) \text{ or } P_1 = \beta_0 \times e^{-\gamma r^2} \tag{2.8}$$

Equation (2.8) gives us the mathematical formula for how to calculate the attractiveness β, of a particle as a function of its attractiveness at 0 units of distance β_0, light absorption coefficient γ and distance r, the Euclidean distance between n particles $n \epsilon \{1, \dots, dim\}$.

$$P_2 = \alpha \times \left(rand - \frac{1}{2}\right) \times scale\,, \qquad \alpha \wedge r \epsilon [0,1] \tag{2.9}$$

Given that this algorithm is under the category of MH, it needs a random parcel, ruled by Gaussian distribution, α and r, as shown in equation (2.9), $scale$ is the upper bound of the search area problem minus the lower bound, this is essential because some problems vary greatly in search areas, thus the FA needs to adapt.

Presented equations (2.8) and (2.9) will influence the movement update of the fireflies given by equation (2.10):

$$x_{i+1}^{dim} = x_i^{dim} + P_1 \times \left(x_j^{dim} - x_i^{dim}\right) + P_2 \tag{2.10}$$

Where x is the position of a given particle at a given dimension.

FA pseudocode is nextly visualized.

FA Pseudocode

1. Declaration of all variables (light absorption coefficient γ, α);

2 .Initialization of population with particles (p);

3. **while** stopping criteria is not met, **do**

4. Calculate Euclidean distance between all particles, distance [$dim\ x\ dim$];

5. Compare the Brightness of i_{th} particle with all the other particles;

6. If particle i is less bright than particle j move i towards j;

7. Else if it is the same, then it moves randomly;

8. Evaluate new particles, based on eq. (2.10), and update brightness;

9. Rank the particles and find the current best;

10. **end while**

2.3. Multi-objective heuristic optimization

Most real-life problems like in engineering, economics, sociology, and nature have several objectives to be satisfied and these objectives sometimes may conflict with one another. In the previous section, meta-heuristics was approached as single-objective optimization problems by transforming one objective into mathematical constraints, however in the current section the objective is to outline the concept of an optimum solution when the exact solution has various variables and a trade-off is required, this is because in having several objective functions, the knowledge of optimum is unclear, in MOPs, there is a need to find the best compromise between all variables rather than a single optimum solution, as in single objective optimization. There is a very clear analogy that demonstrates this concept, suppose a racing car that is engineered to reach the most velocity capable of but, at the same time, it needs to save fuel as much as it can because the race criteria takes into account the efficiency of the vehicle, in this example there are two objectives that go against one another, so an optimal solution must be found that tries to not compromise one as a function of another.

Having both these objectives depending on one another, they must share mutual variables, this process starts with the mathematical formulation of the given problem, thereafter there is an assumption that all the variables are included in the theoretical formulation and that all the

feasible solutions can be represented in the plotting. Figure 2.4 shows the concept of optimization and the processes involved, viewed from a simpler perspective.

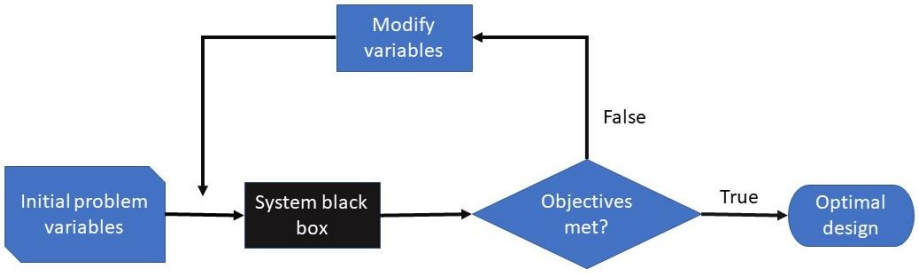

Figure 2.4 – Optimization process.

As mentioned, the process starts by formulating the problem and including all real-world variables into a mathematical system, after this, the formula is inputted into the system black-box where the MOO algorithm stands, when the algorithm ends, it ends by one of two possible reaches, 1) the maximum iteration of the algorithm was reached or 2) the algorithm has stopped optimizing because it achieved the global optimum. In the second case, an optimal design is reached and there is the possibility to translate the acquired solution to a real word scenario in order to be optimally applied in a system, in the first case the algorithm needs to ask itself if the objective was met even if maximum iterations were reached, if this is true, then it has to find an optimal design, if it is false it runs again, but now with modified variables, until an optimal design is reached in the next iterations.

The general MOP is mathematically defined as the following:

Find a vector $\vec{v} = [v_1, v_2, \dots, v_n]^T$ that satisfies the m inequality constraints:

$$g_i(\vec{v}) \leq 0, \qquad i = 1,2, \dots, m \tag{2.11}$$

The p equality constraints:

$$h_i(\vec{v}) = 0\,, \qquad i = 1,2, \dots, p \tag{2.12}$$

And that will optimize the following vector function:

$$\vec{f}(\vec{v}) = [f_1(\vec{v}), f_2(\vec{v}), \dots, f_k(\vec{v})]^T \tag{2.13}$$

When choosing an optimal solution that involves n objectives, the search space will be $\mathbb{R}^n$, in this search space it is possible to trace a set called Pareto set, this set is named after Vilfredo Pareto. A Pareto optimal exists when in a vector of decisions variables $\vec{v} \in \mathcal{F}$ doesn't exist another $\vec{v} \in \mathcal{F}$ such that $f_i(\vec{v}) \leq f_i(\vec{v}^*)$ for all $i = 1, \dots, k$ and $f_j(\vec{v}) < f_j(\vec{v}^*)$ for at least one j. This means that there is a Pareto optimal set if there exists no feasible vector of variables that would decrease some criteria without causing a simultaneous increase in at least one other criterion. Given this, the Pareto optimal set, called the Pareto front is visualized in the following figure, fig. 2.5. A product from the double objective function f_1 x f_2, is seen in fig. 2.6. The particles not seen in fig. 2.5 are the particles that are dominated by the Pareto set, thus its omission. Since none of the solutions in the Pareto set are mathematically better than any other, any of them is a good candidate solution.

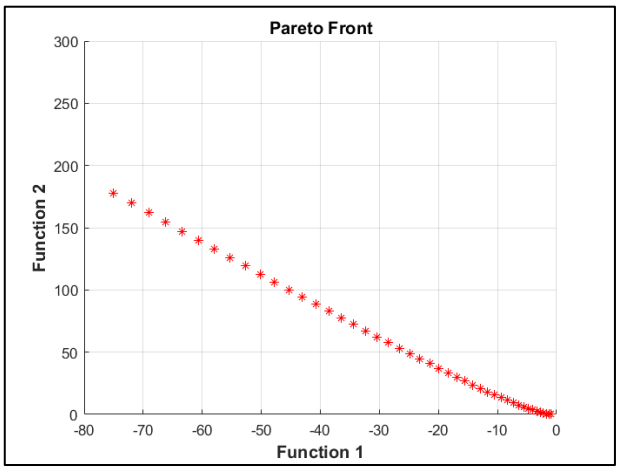

Figure 2.5 – Pareto Front.

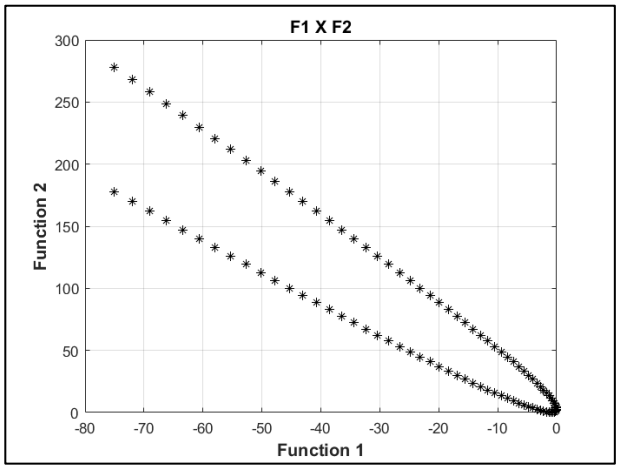

Figure 2.6 – Function 1 x function 2.

Figure 2.7 demonstrates a plotting scenario where all the feasible space is occupied by particles, thereafter the Pareto front is easily visualized, in the case of minimization, it is just a matter of looking at where the particles stop having minor values and tracing an imaginary curve.

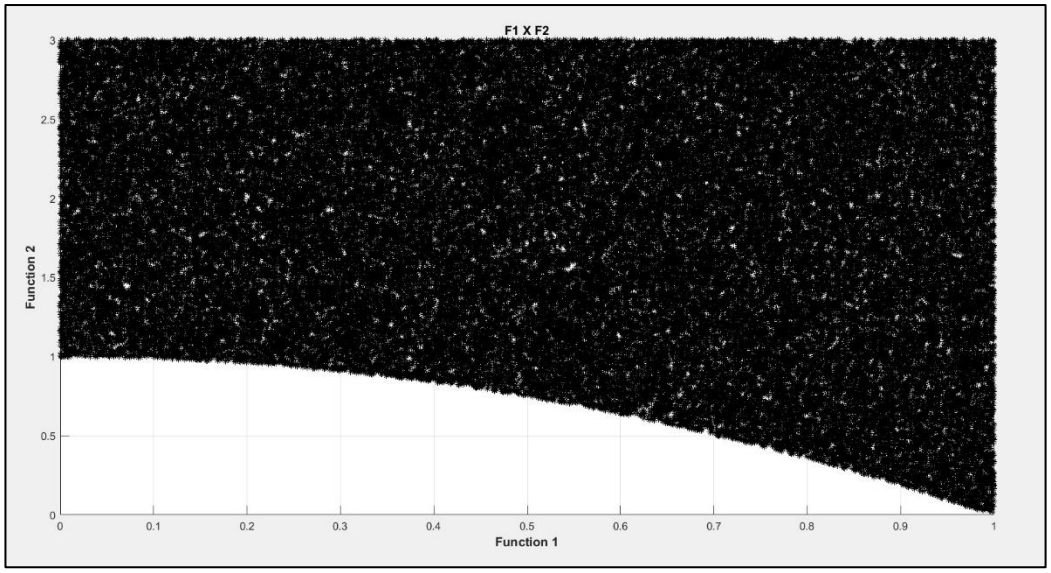

Figure 2.7 – Function 1 x function 2 feasible space.

The resort to metaheuristics in MOPs comes from the very particularly suitable appliance to solve these kinds of problems, MH is less susceptible to the continuity of the Pareto front and can deal with other more abstract forms of Pareto curves like discontinuous or concave. Given the fact that MH algorithms are population-based, it is possible to generate several particles in the Pareto optimal set in a single run and keep evaluating them iteration by iteration until a stopping criterion is met.

Chandra et al. [14] released a review study in mid-2021 that provided in-depth knowledge on nature-inspired MH models as well as hyper-heuristic models, pointing out that nature exhibits intelligent ways to decipher complex optimization problems. The author's approach to MOO

claims that this kind of optimization is at the use of evolutionary techniques, for example, the well-known GA, and that it is a black-box optimization with fine-tuning parameters involving trial and error methods.

The authors formalize the MOO conventional model and point out that in MOO there is not an optimal solution that optimizes all objective functions simultaneously, but there is the existence of a set that denotes the best array of a solution called the Pareto set, as already pointed in this section, that represent the best overall solutions involved, they also point the fact that the choosing or the selection of solutions from the Pareto set must be a human-made decision with knowledge in the problem field, in which it can be controversial, as there is always the option to apply an intelligent system at this level of decision-making. However, the choice of one solution over the other requires good problem knowledge and several problem-related factors, an expert in the area of domain or even AI algorithms are a good candidate for the decision-making phase.

In the next section NSGA will be introduced, one of the most known algorithms in order to solve MOPS, based on GA.

2.3.1 Non-dominated sorting genetic algorithm

Proposed by N.Srinivas and Kalyanmoy Deb in 1994 [15], NSGA is an evolutionary algorithm that existed in the first generation of MOEAS, characterized by the use of Pareto ranking and fitness sharing, it is based on several layers of classification of the population particles. Nondominated particles get a fitness value and are evaluated and possibly removed from the population, the process is repeated until the entire population was evaluated or a stop criterion is met, in order to maintain the diversity of the population, classified individuals are shared with their fitness values.

In their research published in 1994 [15], authors presented to the science community an algorithm that was capable of realizing MOO with population distribution over the Pareto-optimal regions, in contrast to other algorithms well knew back then like, VEGA, that scalarized the objective vector into a single objective and made the solution highly sensitive to the weight vector used in the scalarization. NSGA is based on the speciation method of the GA along with a niche formation and a contemporary method to find multiple Pareto-optimal points simultaneously. Authors say that a common difficulty with MOO comes from a phenomenon called objective conflict, where none of the feasible solutions allow simultaneous optimal solutions for all objectives, thus the preference for a Pareto-optimum, that offers less objective confliction.

Back in the time that the paper was published, the most common and classical methods for MOO were three: 1) Objective Weighting, 2) Distance Function, and 3) Min-Max formulation. The authors pointed out that all these methods concatenated the multiple objectives into only one, presenting the drawbacks of this action like, 1) Single point solution, in a real-world scenario, one solution is unthinkable, different alternatives are a must, 2) If objectives are

discontinuous variables these methods do not work effectively, 3) Requirement of individual optimum knowledge before optimization and 4) Sensitivity towards weight or demand levels, being one of the most profound drawbacks according to the authors.

The idea behind NSGA is a ranking selection method used to emphasize good solutions and a niche method to maintain a subpopulation of good particles, the only deviation from GA comes from the way the selection operator works, crossover and mutation remain the same. The following figure, fig. 2.8, demonstrates the flow chart of the first-generation NSGA.

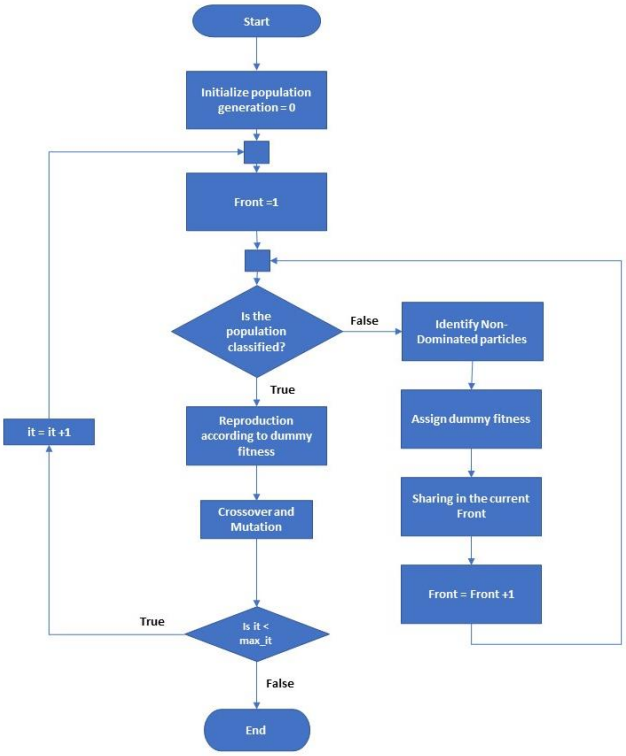

Figure 2.8 – NSGA flowchart.

The classification of nondominated fronts and the sharing operation is what differs from GA. The sharing in each front is achieved by calculating a sharing function value between two particles of the same front, as equation (2.14) shows:

$$Sh(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{share}}\right), & if\ d_{ij} < \sigma_{share} \\ 0, & otherwise \end{cases} \tag{2.14}$$

Where $\boldsymbol{d_{ij}}$ is the phenotypic distance between two individuals I and j, and σ_{share} is the maximum phenotypic distance allowed between any two individuals, to be members of a niche.

The efficiency of NSGA comes from the fact that it ensures the reduction of multiple objectives to a dummy fitness function using non-dominated sorting. Authors claim that any number of

objectives can be solved and that both min-max problems are suitable to be applied. In the final section, simulation results, NSGA is compared to VEGA in three benchmark problems and stands out in all three of them. This subsection represented the origin of generation one of NSGA, in the next subsection, its successor, NSGA-II, will be discussed.

2.3.2 Non-dominated sorting genetic algorithm II

Eight years after the release of the generation-I NSGA, Kalyanmoy Deb and his co-workers proposed a new, refreshed version of the NSGA, called the NSGA-II, which computationally speaking is more efficient than its predecessor. NSGA is considered to be a generation two EA given its recursion to the concept of elitism, it also uses a crowded comparison operator that maintains diversity without specifying any new parameters.

The main reason NSGA-II was developed was that at the time, MOEAs were criticized for three main reasons, with special emphasis on NSGA-I, which urged the formulation of a new and efficient algorithm, the first reason was that its computational complexity was in the order of $O(MN^3)$, M stands for the number of objectives and N for population number, this made the algorithm extremely expensive and inefficient for larger population sizes. Secondly, NSGA-I lacked an elitist approach, elitism can increase performance and prevent the loss of optimal solutions. The third and last reason was that NSGA-I, needed the specification of a sharing parameter, to ensure diversity in a population, the problem with this specification was that it was a fixed value introduced by the programmer and it did not have linear behavior, thus a parameter-less mechanism was desirable. Given this, NSGA-II was proposed, implemented, and tested against other MOEAs, it was able to find a better spread of solutions and better convergence near the Pareto front.

Like other MH algorithms, first, it is set a population region with a set of particles, making up, up to N particles. In NSGA-II the first action upon these particles is to sort them according to their dominance rank. Each particle of the population will be compared to the others to find its dominance, when the first front is found, the total complexity is $O(MN^2)$, to find the second front, the first front is temporally dismissed, and the process is repeated. Meanwhile this process, there are two variables for each particle, domination count, n_p, which represents the number of solutions that dominate the particle p, and s_p a set of solutions that the particle p dominates. The particles in the first front will all have a domination count equal to zero, and for each particle within a zero count the algorithm visits the set of particles within the s_p set, and reduces subsequently its domination count by one, doing this, the next particles that reach a zero count will be part of the next front. An example of all acquired fronts is visualized in the following figure, fig. 2.9.

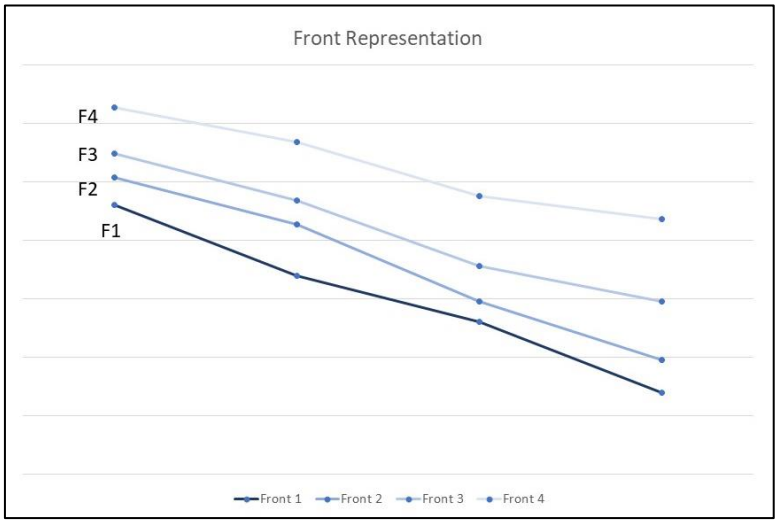

Figure 2.9 – Four fronts organization.

Regarding the preservation of diversity, NSGA-II does not use a sharing parameter like its predecessor, this shared parameter greatly impacted performance based on its chosen value and its function complexity of $O(N^2)$. In NSGA-II a crowded comparison approach is presented, capable of diminishing the previous constraints, this approach does not require any set of values and has a simpler computational complexity. First, the populational density of a given particle is estimated by finding the two nearest neighbors, on which side of the given particle, these neighbors will act as cuboid vertices, to calculate the perimeter of the cuboid, fig. 2.10, this value will then result in the crowding distance, and it will be performed on each objective.

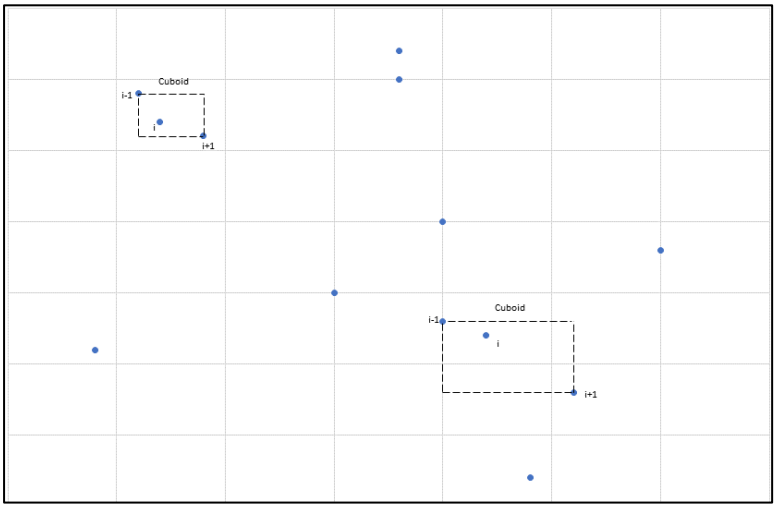

Figure 2.10 – Cuboid representation.

After the entirety of particles have a distance value assigned, the algorithm compares two solutions by their extent of proximity to other particles, if a particle has a smaller cuboid distance value, it will be more crowded, the higher this distance is, the more likely it is for a particle to be isolated. The next phase compares the particles and guides the selection process,

this process has criteria defined upon two attributes responsible for the selection, the crowding distance, and the non-domination rank. When comparing two solutions, the selection preference will lean towards the particle with the lower non-domination rank, if by any chance, both solutions are part of the same front then the less crowded solution will be selected, the computational complexity of this process is lower than the front assignment, thus it remains as $O(MN^2)$. After this selection process is done, crossover and mutation operators are introduced to the algorithm to establish a population mixed with parent and offspring particles, the population size, at this stage, will be $2N$. After this process, only N particles will survive to the next iteration.

To better comprehend the algorithms next phase, let's suppose that on the n^{th} generation there is an initial population consisting of the junction of both parents and offspring ($CM(it) \cup S(it)$), this population will then be sorted by their front ranks until the first n^{th} fronts make up the population of N particles, the rest will be discarded. The algorithm will consequently follow the crowding distance sorting and therefore variation operators, resulting in a new population $CM(it+1) \cup S(it+1)$, this process is then repeated until a finishing criterion is met or the Pareto front is formed, in the next figure, fig. 2.11, this methodology is visualized, followed by an upfront representation of the NSGA-II algorithm.

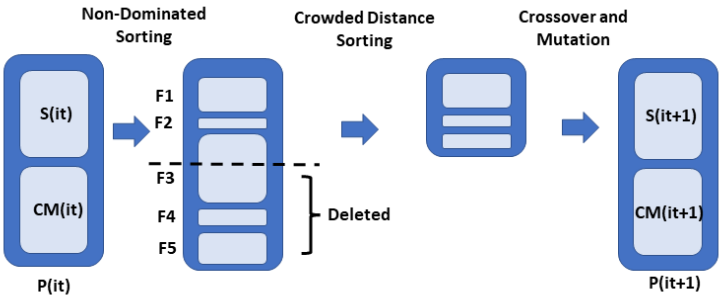

Figure 2.11 – NSGA-II visualization scheme.

Algorithm 1 NSGA-II

1: Representation of objectives and variables, it=0 (Iteration counter), Maximum allowed iteration = max_it;

2: Initialization of a random population (P(it));

3: Evaluate (P(it)) and assign ranks according to the non-dominated sorting method, diversify the population through measures of the agglomeration distance;

4: **While** $it < max\_it$ **do**

5: S(it)= Selection within (P(it)) (Selection based on the agglomeration tournament);

6: CM(it)= Variation within P(it) (With resource to the crossover and mutation methods);

7: Evaluate the current population according to fitness value;

8: Merge population $P_{temp(it)} = S(it) \cup CM(it)$;

9: Assign ranks according to the non-dominated sorting method, diversify the population through measures of the agglomeration distance;

10: $P(it+1) = Survivors(P_{temp(it)})$;

11: $it = it + 1$;

12: **end while**

In [16] NSGA-II was compared with two other algorithms on a set of test problems, the results had shown that NSGA-II can find a better spread of solutions than its rivals, it was also documented that it has a faster and better ideal solution convergence.

After the algorithm runs a simulation, there is a wide set of particles that can be part of a potential solution, the quest to find a possible applicable solution is performed in several different ways, one of the simplest ways is to select a random particle between the Pareto set, given that all the particles within the curve are mathematically ideal solutions. The most recommended way to choose a unique solution is to have an individual that is an expert within the problem area, visualize, analyze and choose the best solution from the Pareto set, given that its criteria are based on years of experience with similar problems. This step can also be taken by a mechanism of artificial intelligence, based upon a database of results of similar problems, making the decision more prone to success, given that two solutions do not converge to the same result. The typical process to reach a final solution takes the steps that are visualized in the next figure, fig. 2.12.

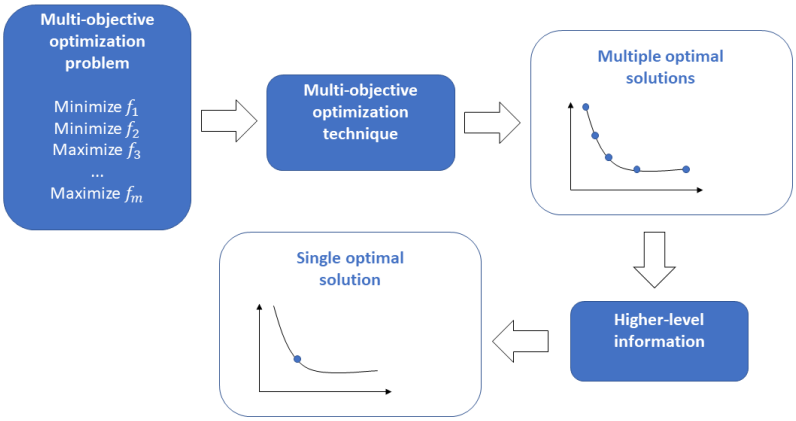

Figure 2.12 – Typical process of a unique solution search.

Chapter 3

Scheduling methodologies

Most of the peer-reviewed scheduling algorithms, in order to prosper energy and maintain QoS fall under one or more of these three categories:

- Dynamic Voltage and Frequency Scaling (DVFS)
- Decomposition or atomization of tasks
- Duty cycling

These methodologies play an important role in the big scheme of sleep scheduling, their main goal is to run the mote within the limited available energy of the mote's environment while trying to meet all the task deadlines and maximizing performance in terms of stimulus detection, for example, it would be interesting in a mote whose objective is to detect forest fires to maximize the number of times the stimulus is checked in order to achieve a quicker action response.

In addition to these three categories, there are also more reviewed ways of achieving energy neutrality and high operation runtimes, like the devise of intelligent algorithms that predict future energy harvested and the implementation of custom-made OSs that can complement these.

Sandhu et al. [17] in their research introduced the concept of ENO, and discuss the various routes, while dwelling on the previously mentioned methodologies, to achieve it in a conventional IoT-EH sensing mote, usually batteryless. ENO strictly means that the harvested energy is higher than the energy required for signal acquisition. Authors discuss the implementation of a dual purpose EH-node that captures the energy from the environment and consequently extracts data from these readings, therefore discarding conventional power-hungry sensors, some of the energy sources cited in their research involve: 1) solar, 2) kinetic, 3) thermal and 4) RF. The combination of a multi-EH-node, of different energy sources data, may also provide higher accuracy of environment variables.

There is also a mention that coupling these signal readings to a capacitor, adulterates the original signal waveform, thus spoiling the data extraction process, therefore open- voltage reading may be beneficial. Another critical point to reach the ENO state is the devise of task scheduling algorithms that consider the fact that the EH unit behaves as the sensing unit, these algorithms will ensure the efficient use of harvested energy, meaning minimization of energy consumption, deadline miss ratio, and maximization of EH and mote operation (tasks executed).

An intrinsic review is then done by the authors pointing out the major pros and cons of the given categories, as well as detailing the steps required to make all of them viable in their presented challenge, thus concluding that more research is required.

One of the final sections of their research discusses the presence of algorithms that predict the future EH window, in order to make the task scheduling algorithm more “intelligent” and “aware”, some presented models employ statistical, probabilistic, and machine learning methods. Given the fact that the EH unit is batteryless, low-power capabilities are essential, thus the need for more research to devise fewer complex algorithms or to present other solutions. The possible solutions presented by the authors involve: exploring optimal sampling frequency of sensing signals, MPP tracking, employment of multi-source-EH as a simultaneous source of energy as well as context information, scheduling frameworks for EH-based sensing, predicting the harvested energy in EH-IoT and more research in batteryless IoT.

In his research, Delgado et al. [18] applied an energy-aware scheduling mechanism to improve the performance of successful application executions on devices without batteries. It was suggested that having prior knowledge of the expected energy to be consumed and the energy to be harvested, played a crucial role when optimizing tasks executability and efficiency, all of this while acknowledging task variables as 1) arrival time, 2) execution time, 3) priority rank, 4) deadline and 5) arrival order. The process to design the problem considered two of the previously mentioned methods, this is, duty cycling, allowing the control of the consumed energy by the nodes when they are not performing useful operations, and the division of a task into atomic subtasks, decomposing the application into a collection of interconnected atomic tasks to reduce overhead.

The OS kernel then keeps track of the active task, re-executing it after a possible power failure, note that the model used is batteryless, the output of the tasks needs to be stored in non-volatile memory to not lose its information. The proposed scheduler will then determine which task should be executed and when, while maximizing the number of tasks, prioritizing the most important ones for the situational area of action.

Examples of typical tasks are 1) sensing, 2) data sampling, 3) data processing, 4) data TX/RX, and 5) use of actuators, etc. Finally, the problem was classified as a MILP (Mixed Integer Linear Programming) and was optimized by Guroby Optimization. In the final section of the paper, results show that making the scheduler energy-aware would avoid power failures, allowing tasks to meet their deadlines and be successfully executed.

The implementation of an OS in these motes is a way of complementing the task scheduling mechanisms in order to design an ideal reactive kernel that adapts to different situations. In their research Yildirim et al. [19]proposed an innovative OS, named InK, that replaces

conventional static programming and welcomes batteryless EH reliable devices. InK runs on HW based on non-volatile memory to preserve important timers, registers, and variables in intermittent execution devices.

Their main objective is to create a dynamic OS capable of handling event-driven programming and a reactive run-time to adapt and respond to changing environments or HW interrupts to determine the next task to complete, and data to collect. To achieve these goals, InK, promises to include: 1) Forward progress of computation by executing restartable atomic tasks encapsulated with task threads each with unique priority, 2) time constraints of task threads by employing preemptive and power failure-immune scheduling and building a timer subsystem composed of a persistent timer, 3) ensuring memory consistency during event handling.

After presenting the proposed model, the authors compare their OS with other state-of-art OSs, by their 1) response to events, acclaiming that none of them can operate threads concurrently and that they would need a dynamic schedule mechanism to do it, 2) scheduling tasks, more precisely the inability to aforemention future events or to perform periodic sensing tasks, claiming, that without this ability, intermittent programming is doomed to oversample and event-misses, 3) handling interrupts, advising the appliance of ISR's that do not interrupt memory consistency, and at last 4) adaptation, by allowing a fluidic control flow.

In the final section of the article, InK is evaluated and compared to Alpaca and MayFly, non-event-driven OSs, by measuring the following metrics 1) success rate, 2) priority events execution, 3) missed event rate, 4) maximum power On time and 5) death rate, standing out in all five and showing a success rate 14 times greater than both OS's

3.1. Dynamic voltage frequency scaling

In any logical electronic circuitry, the power dissipated by its elements can come from static or dynamic sources, dynamic power is correlated with the switching of logic states caused by the computational tasks, on the other hand, static currents happen whenever the circuitry elements are switched on to maintain the logic states between events, even when computation tasks are not occurring. One of the major solutions to mitigate dynamic consumption is by using a technique called DVFS.

DVFS is a method that changes the voltage and frequency at a given device processor or peripherals to adjust power and speed capabilities, given a certain task. Having this change at the processor level ables the effective minimization of energy consumption, by realizing a task with the bare minimum power. This is an effective way to reduce power, since lowering the voltage has a squared result on active power consumption, see equation (3.1).

$$P = \frac{V^2}{R} \tag{3.1}$$

The basic idea is that if the clock frequency is cut in half, the time it takes to complete a certain process or task will double, if voltage keeps constant, then the total energy consumed would appear to stay the same. Thus, it is necessary to address both physical variables concurrently.

Nowadays most modern computers HW implement DVFS to save power and prolong the battery lifespan while keeping the computing power fully operational. It is important to highlight that DVFS also has the capability to enhance the processor power by increasing voltage and frequency, frequently called turbo-mode, boosting, or overclocking. Even so, in a typical mote, it is not realistic to throttle the CPU, the aim is to make the mote as energy efficient as possible, without compromising operation.

Some authors defend that DVFS may not be effectively applied in WSNs since it can have up to 10% area impact on HW and increase complexity, due to the inclusion of level shifters, power-up sequence requirements, and clock scheduling, which will inevitably impact the design turn-around time[20]. Thus, the authors defend that implanting DVFS in a mote would contradict the main goal of energy-aware systems, given the up-scaling of power consumption and mote HW complexity.

In the following section, a study review of DVFS techniques in typical WSNs is carried out.

A. Ravinagarajan et al. [21], proposed a task scheduling algorithm based on linear regression with the use of DVFS techniques, in order to apply an embedded system in an EH-SHM scenario, monitoring, over time, the structural damage of a given material.

The main challenge in this approach is to manage and conserve the energy budget while making available an acceptable level of QoS to the given deployment. This means, maximizing the number of times the system senses and actuates accordingly together with minimization of extra energy expenditure. The task scheduler will ensure that task allocation is in hand-to-hand with the available energy, and the DVFS technique will adjust the data processing unit, accordingly to the present workload.

The authors used an embedded system, called SHiMmer, that propagates ultrasonic waves throughout a structure and consequently detects and analyses any structural damage. The system is divided into three boards, 1) a digital board for data analysis and wireless networking with a working OS, 2) an analog board responsible for sensor activation and 3) a power management board in control of EH-unit and energy storage.

The working modes were divided into three sections: 1) Active high with the core running at 1.2V@300MHz, 2) Active low at 0.85V@150MHz and 3) Active low at 0.85V@75MHz, reducing the system power consumption to 50mW and allowing savings of 30%.

The primary goal of the task scheduler was to maximize the accuracy and recurrence of the measurements, respecting energy consumption and execution time. The algorithm is then shown in both steady state operation and external request operation, results indicate an increase of more than 50% in SHM measurements when compared with an iterative search method, 20% more average accuracy, and realization of up to 95% of external requests and overall reduction of energy needed due to DVFS application.

In [22], Hoang V. et al., published a paper that demonstrated the appliance of two energy-saving techniques in a wireless sensor node, embedded in a WSN, one of the techniques, DPM, lowers the static power dissipation, while the other, DVFS, aims for lowering the dynamic power losses. DPM turns off the unused elements of the circuitry in any action or switches them to lower power modes, DVFS, as already pointed out, adjusts frequency and voltage to required levels. The total power dissipation of an electronic logic circuitry is given by equation (3.2).

$$P_{total} = P_{static} + P_{dynamic} = \left(P_{leakage} + P_{bias}\right) + (P_{sc} + P_{sw}) \tag{3.2}$$

Static power has two components, leakage, and bias power, representing the power losses from circuitry leakage and the power drawn from the circuitry simply from being active, subsequently. Dynamic power comes from the short circuit, representing the power drainage from the current in a short circuit scenario, and the switching power descendant from the PSU. From these four sources of consumption, leakage power and switching power represent 95% of the total power.

The possible strategies presented by the authors to implement DVFS, include: 1) All tasks are supplied with the same low voltage to minimize energy dissipation, 2) Tasks are supplied with different voltages based on activity rate, and 3) The first task runs with the highest frequency/voltage and given the execution time, the variables are regulated accordingly, this strategy was also approached in [23].

The authors ended by increasing the complexity of the mote, by adding a Power Availability Manager, responsible for the DPM/DVFS techniques, and by adding an FPGA to enhance mote performance in a signal and video processing application. Given the fact that DVFS is a method that causes power overhead, the authors decided to address this issue by having only two values of supply voltage, 3V and 3.6V, the frequency adjusts itself by the type of received events, nominated normal or hazardous, these events will have an impact on the processor and memory and even on the TX bit throughput and range, based on equation (3.3).

$$E_{TX}(k, d) = E_{diss} \times k + \sigma_{amp} \times k \times d^2 \tag{3.3}$$

Where E_{diss} is the power dissipated by the radio module, k represents the number of bits to be transmitted, σ_{amp} the amplification coefficient and d, the distance range.

In the results section, the authors claim energy gains up to 80% for the processor, 53% for memory, and 31% for the radio module, leading to an increase in the battery lifetime.

In [24], Zhuo et al., proposed two dynamic task scheduling algorithms, relying on a DVS technique. The primary goal, when devising these algorithms was to minimize the system-level energy consumption, given the fact that when DVS acts to slow down the processor, it increases mote energy consumption. Both algorithms used an optimal speed setting, whose action fixes a speed value that would minimize overall system energy for a given task, and also used a limited preemption setting, which reduces the number of task interruptions.

Results show up to 43% energy savings when compared to other dynamic scheduling algorithms, and 12% in scenarios with no appliance of DVS, the authors also pinpoint that if the device power is much larger when compared to the CPU power, then it is not viable to use this technique, as results show that energy saving is greater with no DVS approaches.

Manzak et al. [25], proposed a variable voltage task scheduling algorithm, that considered both periodic and aperiodic tasks, whose objective was to minimize energy expenditure in a processing unit regarding its energy limitations.

The working model consisted of the given processing unit, capable of working under an array of different input voltages, and an off-chip DC/DC converter. The author's approach was to establish already existing scheduling algorithms, apply them, and after the feasible schedules were calculated an iterative voltage distribution to each task was performed, the algorithms used were: 1) EDF, 2) EDD, and 3) RM.

A Lagrange multiplier method was coupled to these algorithms to find a minimum energy configuration for both periodic and aperiodic tasks, correlating the operating voltage with the switching of activities, note that the authors decided to remain the operating voltage fixed during the execution of j_{th} task with $j \in N$, a subset of tasks, in order to keep simplicity.

The rest of the paper accounts for problem formulation, with each task supply voltage being updated after each iteration. After the formulation, each task scheduling algorithm is then presented by its definition, mechanics, and complexity. In the results section, each algorithm was put to the proof, coupled with the Lagrange method to assign the supply voltage to each task, followed by a brief performance summary on each heuristic, the highest energy saving was found in the coupling of EDF plus the Lagrange method, with different task switching values throughout the simulation, showing 65.4% energy saved, and an algorithm complexity of $O(n^2 \log k)$, n representing the number of tasks and k, the number of iterations.

The authors conclude by pointing out that a delay in voltage variation at the converter and a delay in clock frequency variation at the CPU must be addressed to achieve a more realistic simulation.

As mentioned at the beginning of the section, if the frequency is cutted in half the task execution time will naturally double, given that clock frequency translates to the number of cycles per unit of time. The problem with this formulation is that leakage is not considered, it stays present as long as the circuitry has current flowing in it, so if the mote stays active for more time, the leakage losses will also increase, and that cuts total energy savings, even though that peak power has been lowered[26]. When voltage comes into the equation, it is possible to reduce leakage losses, since leakage decreases linearly with voltage, even so, given this linear correlation the mote will start to consume a greater percentage of the total energy.

Another problem that needs to be addressed is the fact that energy saving should be looked in the perspective of the whole unit, instead of focusing the attention directly on the CPU, for example, memory voltage cannot be decreased by the same amount as the CPU, otherwise, it will have problems keeping data, and performing R/W commands.

DRAM, as it has been introduced more compactly into WSN's mote, has become more sought-after for memory purposes[27]. DRAM is another component that also needs to be continually refreshed and cannot have its voltage continuously changing, due to processing errors.

To be clear, the total energy to perform a stable performance across all the circuitry needs to be taken into account, and that means establishing a tradeoff between HW complexity and power and energy benefits. In [24] authors highlight the fact that their algorithms, and others similar, that control voltage and frequency, should be avoided when the device power out scales the CPU consumption, considering that most WSNs approaches avoid utmost CPU overhead, with time complexity notations being as basic as possible, it is rare to find a scenario where CPU power consumption out scales the circuitry segment as a whole.

In [25] it wouldn't be practical to annex a DC/DC converter in a mote as these energy converters devices, are economically unviable and add a great deal of complexity within a WSN scenario. In [20] it is also pointed out, as already mentioned, that applying these techniques to typical motes would translate into a larger mote due to HW implementation, increasing complexity, and total active power. It is important to acknowledge as well that most of the DVFS scheduling algorithms show time complexity that brings excessive processing within a mote's CPU in a typical WSN scenario, one of the solutions, is performing the processing off-grid. For example, the algorithm that performed better in Manzak et al. studies showed a time complexity of $O(n^2 \log k)$, when observing the following figure, fig. 3.1, its witnessed that this kind of time complexity is much more exhaustive to the CPU when compared to others demonstrated in the graph.

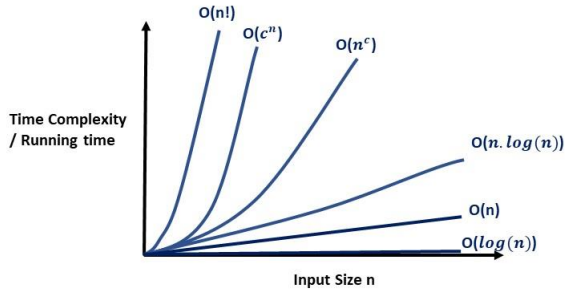

Figure 3.1 – Big O notation graph.

Summing up, the scientific community consensus that the use of DVFS methods comes at expense of power overhead due to its complex computation and hardware implementation [25][28] [29], given this, DVFS techniques will not be applied in the experimental section of this dissertation, the present section serves only for means of theoretical information about energy saving DVFS techniques.

3.2. Tasks atomization

When encountering a complex problem, it is prudent to decompose it into several uncomplicated problems, working on these easier problems, one by one will help in the bigger picture, given that, once all of them are finished, the realization of the complex problem is also dealt with. Task decomposition follows the same train of thought, since an EH-mote energy flow is limited to the surroundings, it is difficult to keep running the same power-intensive tasks unceasingly, making the system fragile to critical tasks and increasing the possibility to miss important deadlines.

Therefore, devising software for EH devices is problematic, since the mode of operation is intermittent, as energy is scarce and not always available, EH programs running on-chip often reboot due to power failure, which spoils program state and causes loss of important data like global variables, program counters, registers, etc. This prevents forward progress, thus, the implementation of task-based programming models allows for the continued execution of long-span applications.

A promising resolution to this kind of problem is to decompose or atomize the energy-intensive tasks into several subtasks which will eventually resource less computation and energy during their active running. For example, the sensing task of a given parameter, can be decomposed into several subtasks like 1) sensing the voltage drop across the sensor, 2) translating the voltage value into data bits, 3) reading data into the flash memory, and 4) writing data into the processor. The following figure, fig. 3.2 demonstrates an example of this decomposition. Note that when decomposing tasks, it is possible to execute the subtasks in different orders and periods from the original task, depending on the application scenario. These tasks will run to completion if they consume less energy than the present energy availability in the mote, thus,

these motes are dependent on the actual energy capacity and cannot fully rely on future external energy acquisition.

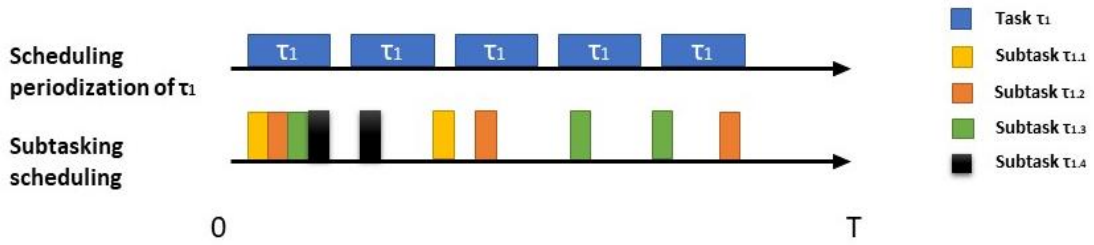

Figure 3.1 – Typical process of a unique solution search.

The decomposition into small tasks allows the occurrence of a successful partial task termination if a power failure happens and the data is allowed to be written in non-volatile memory, in the case of a large static task being performed, if a sudden power failure occurs under its active time, then the energy spent on the task would be meaningless and wasted, given the fact that all the task actions would not be fully accomplished until task finalization, the following reasoning is represented in the following figure, fig 3.3. Even so, small subtasks increase overhead due to the periodic act of writing data into the non-volatile memory, after each finished small task[30].

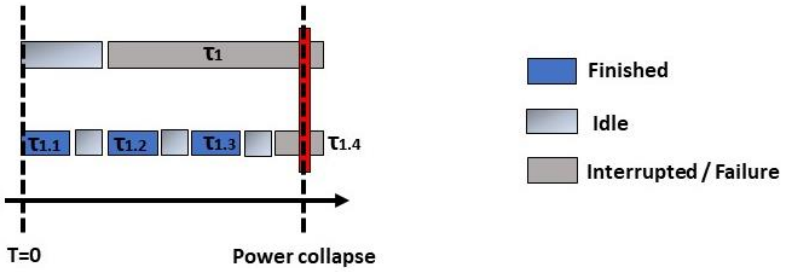

Figure 3.2 – Small subtasks vs unity task.

In addition to the overhead augmentation, task decomposition presents some other drawbacks, like the requirement of a programmer or a compiler to decompose a task in order to execute the desired model and to rearrange a task-to-task control flow, depending on the application scenario. It is also mentioned that subtasks can include arbitrary computation that despite power failures, should be finished, which causes disagreement since it also occurs when running unity tasks, not considering it an exclusive drawback from task decomposition.

In [30] Majid. A et al. conveyed some of the drawbacks of task decomposition, mentioned previously, and proposed a task-based system, Coala, that couples adaptive task size execution by performing task decomposition and grouping, based upon energy availability.

The author's main goal was to create a scheduler that could reduce overhead and transition costs while splitting tasks, it uses an energy prediction scheme formulated on the previous history, to the extent of finding criteria that considers task grouping order, and size. In the imminence of task non-termination, Coala incorporates a timer-based task-decomposing mechanism, that splits a task before its failure, and in consequence, it stores the remaining data in non-volatile memory, to second this mechanism it also incorporates a virtual memory manager to successfully achieve data storage in case of splitting imminence.

In the results section, Coala was compared with Alpaca, an algorithm based on intermittent execution without checkpoint listing [31], with six different benchmarks, results show a 25% to 70% increase in execution time, between all benchmarks. Coala has a lot of features and its formulation is quite extensive, more details about this dynamic scheduler are found at [30].

In [32] Zhu et al. proposed a Dynamic Energy Oriented Scheduling, that treats energy as a first-class schedulable resource and dynamically schedules tasks, accounting for real-time energy availability and tasks energy consumption. One of the methods used in their work consists of the decomposition and combination of subtasks, in order to avoid redundant operations, as well as concurrently executing subtasks that utilize the same resources.

To achieve this, their scheduler implements a feature that considers the mote as a single system instead of only the CPU, with regard to aggregating present tasks into the full range of circuitry components, allowing the system to combine them, if at least one of the tasks is redundant, for example, the task to wake up the radio does not need to be executed every time a variable is sensed, the data can be stacked overtime at the memory unit and then concatenated into a frame, only then the wake-up task may be executed in pursuance of TX the frame.

Another feature implemented is concurrent execution, in the interest of reducing CPU idle time and increasing sleep time. The basic design is divided into four phases: 1) Decomposition of tasks, the process of creating subtasks from an original task, 2) Combination, which involves the combination of tasks to save energy, it does this by following the two features previously mentioned, 3) Admission control, consisting of a schedulability test to check if energy is sufficient to schedule all the present tasks, and 4) Optimization, maximizing the total number of occurring tasks while respecting several linear restrictions and inequalities.

In the results section, the scheduler was implemented and evaluated under indoor and outdoor scenarios, the comparison metrics were the number of tasks executed and the number of missed deadlines, and it was tested against two meta schedulers, WEDF and COS. Task execution was

~30% higher with the proposed scheduler in both environments and missed deadlines were ~50% in both environments as well, following similar results against both rivals.

Tracking volatile states makes tasks restartable, but it won't ensure that a re-execution, after power loss, will have sufficient energy to achieve completion. With task decomposition this problem is mitigated, nevertheless, it is difficult to reason on how probable a task is to drain all present energy, for example, if a compiler divides extensively a task into several tasks, it will cause energy waste due to time overhead, in contrast, if a compiler uses too few divisions, the program may never finish completion, continuously restarting and failing.

In [33] CleanCut is proposed, a debugging tool that can monitor and account for non-terminating tasks, as well as decompose applications into efficient, completed tasks, it helps the programmer by placing task boundaries in a program. Two subdivisions of CleanCut are proposed, the checker and the placer.

The checker is responsible for checking a task boundary assignment and reporting any tasks that do not terminate, if the checker finds a path that consumes a higher energy level than the established threshold, it reports the path along with the boundaries of the given task. The placer subdivides an entire program into multiple tasks that are not capable of stalling in execution, it works iteratively, evaluating and identifying possible non-terminating paths, as well as submitting new task boundaries. The junction of both permits overhead minimization and elimination of non-terminating paths. When compared with a manual task decomposition strategy, it outperforms it with a mean speedup of up to 4.5 times [33].

3.3. Duty cycling

Duty cycling is one of the most used mechanisms applied on WSNs motes, to reduce the overall energy consumption, normally caused by idle listening. When referring to duty cycling, the most important factor is to define a given duty cycle, this is a percentual value that determines how much of the time the mote is ready to TX/RX and asleep. Let's say, for example, that a mote operates, for 24 hours with a duty cycle equal to 5%, that means that the mote is asleep for 95% of those 24 hours, to be more precise the mote is asleep for 22 hours and 48 minutes, and for 1 hour and 12 minutes is idle or ready to TX/RX.

Most studies found on duty cycle protocols define a fixed value across the network in order to attain synchronization between motes, but when referring to EH-motes, mostly batteryless, this becomes a problem because the current and future energy acquisition may be hard to predict, hence the need for a dynamic duty cycle, that keeps changing in conformity of the environment, mote, and network variables, like predicted energy, energy stored, tasks in need of execution, current topology changes, etc.

In the current section, a presentation is carried out about taxonomy studies referring to the duty cycle and also a broad view of its application in EH scenarios.

In 2014 a team of researchers from Brazil [34] gathered to publish an extensive survey study on duty cycling in WSNs, considering duty cycling to be one of the most effective ways to spare energy, with the goal to make the duty cycle value as minimal as possible, for example, as low as 0.1%.

The authors organized various proposals into the taxonomy and provided direction into their vantages and disadvantages, making attention to the fact that no WSN is equal to another, so all the variables must be well-known and specific to a given scenario. As previously said the main objective of duty cycling is to reduce energy consumption, going off when the mote is not needed, this means, minimizing active time, idle listening, and avoiding over-hearing.

The duty cycle is divided into 3 main categories in a typical WSN, 1) Synchronous, 2) Asynchronous and 3) Semi-synchronous. From these 3 categories arise some problems that affect in some way all of them like: 1) End-to-end message delay, the unrequested wait in a multi-hop scenario where a mote must wait for the next one to wake up, promoting network latency. 2) Collision rates, given the shortening of TX/RX time windows caused by duty-cycling. 3) Control packet overhead, because of the intrinsic need for more control traffic packages in order to synchronize motes. The taxonomy is divided as presented in the following figure, fig 3.4, adapted from [34].

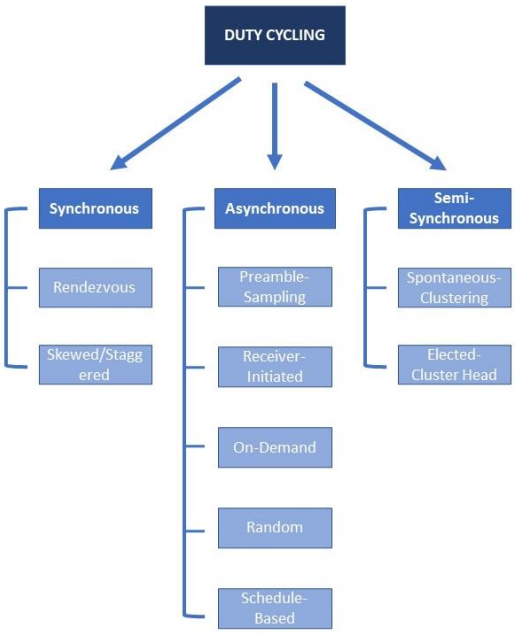

Figure 3.4 – Duty cycle taxonomy.

In summary, synchronous motes share a clock and are time synchronized, varying the degree of synchronization. Asynchronous methods do not share a clock and semi-synchronous form clusters in the network that have synchronous motes inside, and then there is an interaction between the clusters.

In the final section of the paper, the authors debate the impact of applications by, 1) data flow types, 2) frequency of transmissions, 3) end-to-end delay requirements, 4) security requirements, and 5) synchronization requirements. By the impact of the present hardware by, 1) CPU, Program memory and storage, 2) Clock, and 3) hardware synchronization. And finally, by the impact of deployment variables like 1) Terrain, 2) Density, 3) Replacement and addition of motes, 4) Topology types, 5) Topology stability, and 6) Longevity.

Concluding with a brief section on future directions approaching topics like the increasing power and complexity of motes, the upcoming models that predict latency, the increase of popularity in multi-purpose WSNs, and improvements in the PSU subsystem.

The next section is adapted from this taxonomy paper, where it will be presented the protocols shown in figure 3.4, detailed by a brief description and advantages/disadvantages of each one.

In the next figure, fig. 3.5, the two types of synchronous schemes are demonstrated.

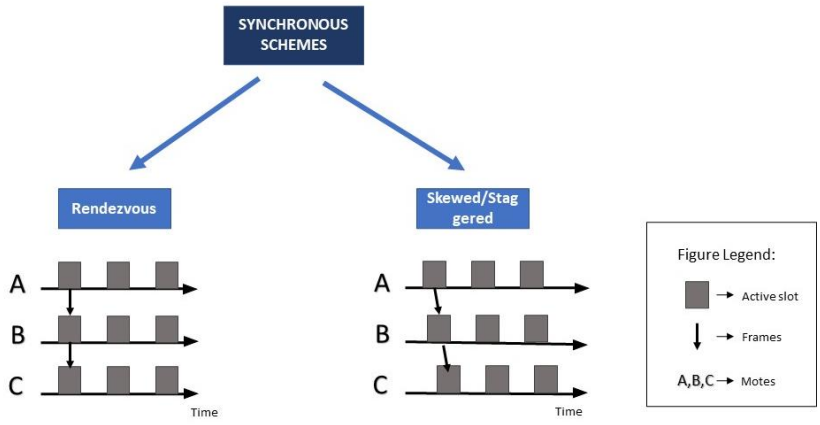

Figure 3.5 – Synchronous schemes.

In the left part of the scheme, the rendezvous protocol, all motes turn their radios on and off at the same instance, it is a relatively intuitive and straightforward protocol to adapt, and it supports broadcast, a method of simultaneous messaging to all WSN receptors. In contrast, it requires a synching mechanism that can be exhaustive to the WSN motes, and guard times are necessary to deal with sync errors, incrementing the duty cycle.

In the right part of the scheme, it is demonstrated the skewed/staggered protocol where motes wake up in a staircase pattern according to the variation in the network topology, it reduces sleep waiting, but in contrast, it is topology dependent, it requires a synching mechanism and guard times are also necessary.

In the following figure, fig. 3.6, the approached asynchronous methods are demonstrated.

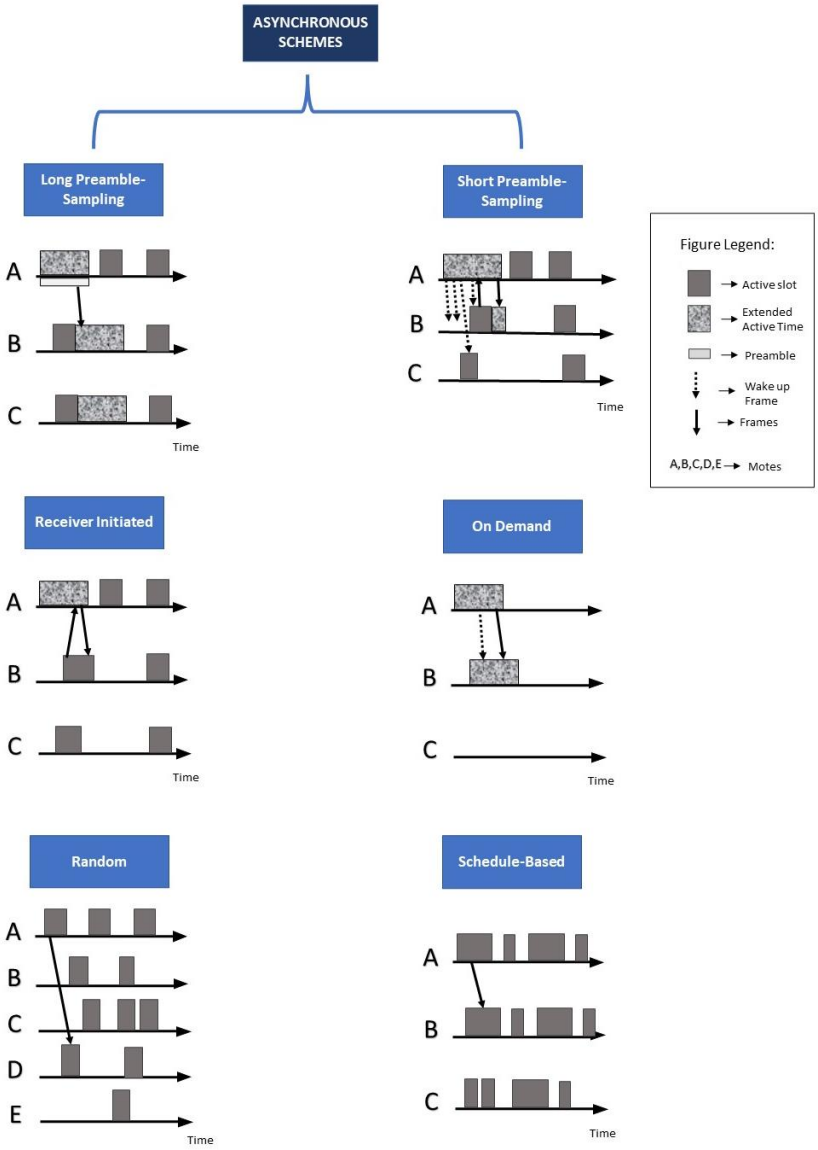

Figure 3.6 – Asynchronous schemes.

About Preamble Sampling, there are two types, long and short, in the long preamble, frames are preceded by long preambles that guarantee that motes which wake up periodically will have time to wake up, detect the frame and stay on to receive it, it shows some advantages like the transfer of energy expenditure from many receivers to one sender, but in contrast, it creates a long appropriation of channels that can block other motes, it has a high end-to-end delay and high probability of an excessive overhearing. In the short preambles, instead of the long ones the sender transmits a train of short packages with the receiver address, the receiver may ACK the first frames in the train and can reduce signaling time as well as minimize over-hearing, in this situation the unused motes can go to sleep. However, it is inefficient for broadcast traffic.

About the Receiver-initiated protocol, the sender waits for a periodic beacon from the receiver and transmits the frame back only after hearing the beacon, it shows advantages like the receiver beacon not needing to occupy medium for as long as preambles occur and the adaptation to the traffic load. However, it shows a high end-to-end delay.

The On-Demand protocol is a wake-up schedule that guarantees that all the active times overlap in order to be resilient to topology changes and not need the presence of control messages, but it has a particularity that these motes need an embedded wake-up radio, this additional interface will consume more power.

The random protocol consists in acquiring a dense deployment phase into the network, all the motes can go to sleep and wake up with random and variable duty cycles, relying on a high probability that there will be active neighbors when needed. This protocol shows a low end-to-end delay, an equal distribution of traffic load may extend the network lifetime, but in contrast, it is restricted to dense deployments, it shows low delivery rates and needs complex mechanisms to adapt the duty cycle to current density.

The last asynchronous protocol, Schedule-Based, consists in a wake-up schedule that also guarantees that all the active times overlap, so it is resilient to topology changes and does not need the presence of control messages, however, it shows a lower duty cycle resulting in high latency.

Lastly, in the next figure, fig. 3.7, the semi-synchronous schemes are shown.

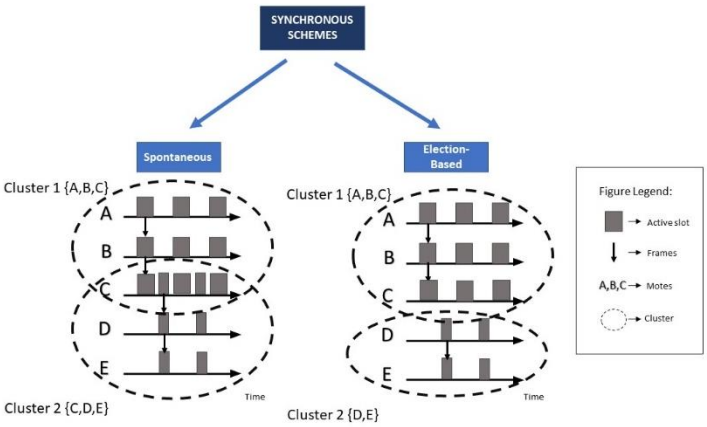

Figure 3.7 – Semi-synchronous schemes.

In the prior figure, fig. 3.7, on the left side, there is the Spontaneous protocol, which consists of the formation of clusters spontaneously as a consequence of synchronization among neighbors, results show that neighbor synchronization is easier than global synchronization, it is simpler to maintain than election-based clusters, but motes need to participate in many clusters, thus increasing the duty cycle.

About the Election Based, it is observable that a cluster-head is elected, and neighbors synchronize to it, as well as the previous protocol, this neighbor synchronization is easier than the global synchronization, but it has some downsides like the cluster maintenance demanding control traffic, the inadequate use for dynamic topologies and the requirement for additional mechanisms to achieve inter-cluster communication.

Similar to [34] a team of authors assembled a survey based on state-of-art scheduling methods, addressing duty cycling [35]. In contrast to [34], this paper addresses these methods based on EPWSNs with or without the use of batteries, the research started by demonstrating the term "energy-neutrality", a condition where the energy demand does not exceed the energy supply, just like ENO, which opens a possibility to a batteryless mote, thus addressing the problem of a limited lifetime.

Authors claim that the reach of this state is an intriguing challenge, given the fact that energy availability is uncertain, and that high volatility is present, so the main objective is to efficiently and adaptively utilize the available energy to optimize for example throughput or latency issues, aiming for the balance of energy supply and energy demand. In order to achieve this state, it is necessary the presence of algorithms that can dynamically change the duty cycle dependent on energy supply.

The authors then briefly write about EH methods as well as energy storage technologies and the most common challenges like 1) highly dynamic network topology due to failure-prone motes, 2) wireless links, 3) limited memory and 4) processing power. Bearing in mind wake-up schedules protocols, there are a set of challenges that need to be affronted like the 1) adaption to environment dynamics by having algorithms that are aware of the majority of the volatile variables, 2) minimization of latency, 3) duty cycle range, 4) processing complexity via a centralized processor and the 5) correlation between overhead and scalability. The rest of the paper follows the approach of [34] by assembling a taxonomy of state-of-art wake-up scheduling protocols.

In [36] Has et al. presents an adaptive duty cycle algorithm that allows a typical mote to autonomously adjust its duty cycle regarding present and future energy availability. The algorithm was developed aiming the fulfilling of three objectives: 1) achieving energy neutrality, 2) maximizing performance, and 3) adapting algorithm behavior to different energy schemes, they achieve these objectives by merging future energy prediction methods with duty cycle adaptation. Authors point out that most power management methods are based on battery status, often neglecting the dynamics of the energy supply, accounting only for the dynamics of the energy consumer, like the CPU, radio, and most peripherals. It is noted that giving importance to the latter point may even help EH systems to provide enhanced performances in contrast to operating at the lowest performance level.

Adapting duty cycle, as already seen in prior referenced papers, regarding a prediction of future energy availability is often done, in [37] authors point out that in conventional topologies, a fixed duty cycle can negatively influence the multi-hop data transmission cost, in contrast making this value dynamic and heterogenous will shift the cost accordingly to the duty cycle, minimizing negative influence. If the transmission cost is uncertain, the selection of the forwarding path will also become uncertain, resulting in poor results and low EH efficiency. Given this, the authors propose a novel communication scheme interconnected with a duty cycle adjustment method. The proposed scheme helps forward appropriate paths and avoid offline motes, while the duty cycle adjustment is based on an artificial intelligence algorithm that predicts future energy slots, balancing energy acquisition and consumption.

Vigarito et al. [38], discards the assumption of predicting the future energy profile and proposes an algorithm based on control theory, with the objective to find the optimal duty cycle, introducing a tunable mechanism for reducing the variance of the duty cycling. Making the duty cycle's variance minimized is beneficial in certain instances, for example when applied in sensor networks, in order to respect the criteria for some communication protocols, like MAC. Also, there are events in real scenarios that have an intrinsic periodic behavior, and the network must be awake, at the right cycle, to fetch data from these events, continuously and without interruption.

This approach stands out from other works referenced, since it makes no assumption about the nature and dynamics of the energy source, taking only into account the battery profile, turning it into an assured implementation when it comes to real-life systems, where energy profile is frequently stochastic and unpredictable.

The authors do this by creating an objective function centered on the mote's battery and other tasks performance, minimizing one and maximizing the other, subsequently. An optimal linear quadric tracking methodology is applied to maintain an optimal trade-off condition between both. When compared to another referenced algorithm, results have shown, in every data set, a 0% dead time, a lower duty cycle variance, and higher energy efficiency [38].

3.4. Proposed system architecture

During the exploration of the prior subjects of this chapter, the author has decided that by regulating the duty cycle there is the possibility to increase the mote work and lifetime, by reducing the energy consumption through periodic sleep mechanisms. The main objective is to simulate and mimic a typical mote from a WSN. A TCP/IP socket connection between the server (MATLAB) and the client (LPC1768) is established, and through this connection a channel is formed in which the MH algorithm optimization can act, to calculate and return to the client the optimal sleep scheduling pattern, enabling a power-continuous monitorization. The overall architecture of the system is visualized in the next figure, fig. 3.8.

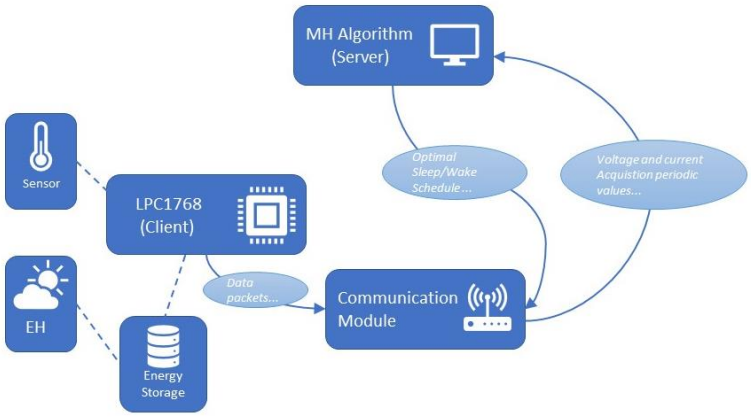

Figure 3.8 – System diagram.

The communication module presented in fig. x, represents the WiFly RN171, a wireless LAN module, fully qualified and certified for 2.4 GHz Wi-Fi. The choice of this module is due to its capabilities to work under an ultra-low-power regime, which is inevitable for energy consumption minimization, the module accepts a 3.3V power supply, it consumes 4μA sleep, 35 mA on RX, and 185 mA on TX at 12dBm, noting that the TX power is programmable, it also can implement a programmable wake-up due to an RTC in its hardware. It possesses an onboard TCP/IP stack that will be fundamental to establishing the socket connection to the server.

When referring to the EH module, DC2080 is a viable choice, a demo board suitable for low power EH that converts thermal, solar, and piezoelectric stimulus into current. A visual representation of the board, with emphasis on the photovoltaic cells, is seen in the figure. 3.9.

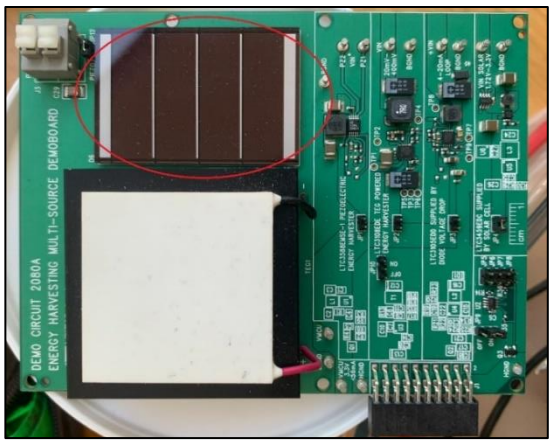

Figure 3.9 – Photovoltaic cells from DC2080 .

DC2080 board has optional energy storage, which consists of an array of capacitors, that includes 15 small capacitors, with a capacity of 100µF each, totalizing a capacity of 1.5 mF.

As pointed out earlier in this section MATLAB will be used to create a TCP/IP server, this interface is based on the transport protocol covered on top of the internet protocol. MATLAB has an Instrument Control Toolbox that supports TCP/IP communication and enables the creation of a network socket for communication with a single client. For this, it will be important that a user-defined port is established as well as the IP addresses of both the client and server.

The client will receive from the WiFly a trigger sent by the server that changes the sleep schedule of both the MCU and the WiFly, at the same time it receives a time delay signal, T_{ch}, that starts a timer which causes the communication module to be turned back on. The MH application programs the delay time, based on the last known predicted energy harvested curve and the jitter between the comms module clock and MCU clock. The mathematical formula that represents the model is explicit in the following paragraphs, it was transacted from [39], where the current dissertation writer was one of the authors.

In the current communication frame that occurs at the instant time i, the MH application must know the communication to establish at the next frame at the instant time i +1. This assumption makes it possible to determine the ideal communication time window, T_{Δ_U}, and the energy, E_{Δ_C}, required for communication. This calculation is performed based on the impact of uncertainty introduced by imprecision and jitter of the clock signal responsible for synchronizing both clocks of the LPC1768 and WiFly RN171. It is assumed that the introduced uncertainty has a time width of T_{Δ_J}, which is added to the useful communication frame window, T_{Δ_U}, and establishes the communication frame time T_{Δ_C}.

$$T_{\Delta_C} = T_{\Delta_U} + T_{\Delta_J}. \tag{3.4}$$

As the energy harvesting time, T_{ch}, increases, the uncertainty time, T_{Δ_j}, also increases due to the sum of errors increasing, as time passes by, in both clocks' synchronization. To maintain the necessary communication time frame T_{Δ_U}, it is required to increase the communication time T_{Δ_C}, that guarantees the completion of communications.

The energy required for the communication is given by E_{Δ_C}.

$$E_{\Delta_C} = E_{\Delta_U} + E_{\Delta_j}. \tag{3.5}$$

The energy corresponding to the uncertainty time interval, T_{Δ_j}, is a function of the load time, T_{ch}, being given by

$$T_{\Delta_j} = \alpha(t, T_{temp})T_{ch}. \tag{3.6}$$

Where α is a linear value that derives from the characteristics of the WiFly clock and LPC1768 core clock. It is considered that in the period T_{Δ_j} the communication module is on, requiring the operating power in reception mode, P_{RX}. The energy of this interval is given by E_{Δ_j}.

$$E_{\Delta_j} = P_{RX}T_{\Delta_j}. \tag{3.7}$$

The energy required to carry out communications within the useful time frame T_{Δ_U}, must consider the following WiFly's operating states and corresponding powers:

- Sleep – WiFly idle and MCU on power management - P_{sleep}
- Idle – WiFly idle and MCU active - P_{idle}
- RX – WiFly RX and MCU active - P_{RX}
- TX – WiFly TX and MCU active - P_{TX}

Considering the previous parameters, the energy E_{Δ_U} of the useful communication interval T_{Δ_U}, will be given by,

$$E_{\Delta_U} = P_{TX}T_{TX} + P_{RX}T_{RX} + P_{idle}T_{idle} + P_{sleep}T_{sleep.} \tag{3.8}$$

What corresponds to the useful communication interval T_{Δ_U}, given by,

$$T_{\Delta_U} = T_{\Delta_{TX}} + T_{\Delta_{RX}} + T_{\Delta_{idle}} + T_{\Delta_{sleep.}} \tag{3.9}$$

The energy harvested in the time interval T_{ch}, considering the harvested power P_{EH}, is given by,

$$E_{\Delta_{T_{ch}}} = P_{EH}(t)T_{ch.} \tag{3.10}$$

From the analysis of the previous model, it is seen that by increasing the charging time T_{ch} more energy is harvested, but on the other hand, an increase in uncertainty time T_{Δ_j} will cause an increase in the energy E_{Δ_C}. Therefore, an optimal operating point that ensures an optimal tradeoff is required, this point ensures that the energy harvested $E_{\Delta_{T_{ch}}}$ is greater than or equal to E_{Δ_C}, and on the other hand, the energy E_{Δ_j} has a minimum possible value, hence the utilization of the MOO MH algorithm, NSGA-II.

The energy that the server would ideally count on at the time frame i+1, is dependent on the extrapolation values calculated, these values are read from the voltage across the capacitor and determine the energy stored in the capacitor given by E_{cap}.

$$E_{cap} = \frac{1}{2} C V^2. \tag{3.11}$$

Chapter 4

Mbed operating system

As pointed out, at the beginning of this work, an NXP LPC1768 MCU, based on the 32-bit ARM Cortex-M3 design, is perceived as the central unit of the system, the MCU runs on an OS called Mbed, which is an open-source embedded operating system targeted specifically for IoT.

4.1. Introducing the OS

Mbed OS was released in 2009 with the marketing target of microcontrollers, IoT, and wearable devices, the applications for the Mbed devices can be developed online, only a browser is needed. All the building, linking and compilation processes are realized on a remote server. The firmware is written in C/C++ and it is currently supervised by ARM®. To develop and create firmware running on the LPC1768 the Mbed online IDE was used, it is a free online code editor that compiles the code in a remote server, using the ARMCC C/C++ compiler.

The Mbed OS beholds some core libraries that can provide peripheral drivers, networking, RTOS, building tools, testing frameworks, debugging scripts, and power management actions.

The latter is of great relevance in the work, and it will be further explored in the following sections of this chapter.

The Mbed compiler being a lightweight online compiler, permits the writing of programs, without many resources, only a browser, internet connection, and a USB cable to connect to the MCU via on-chip bootloader software are required. This makes the writing of code more convenient and quicker.

As it is seen in the following figure, fig. 4.1, the IDE is similar to a conventional IDE and straightforward to utilize, it has a tab responsible for the management of projects, program workspace, noted as number 1, it has a tab where you can interact with the IDE, noted as number 2, to create, import, save, build, compile, commit via version control and to revise code. In number 3 it is seen the program details, number 4 is where you can see all the building errors and warnings, and last, number 5 is where you can see all the files and libraries included in your open project, as well as being the window where program coding happens.

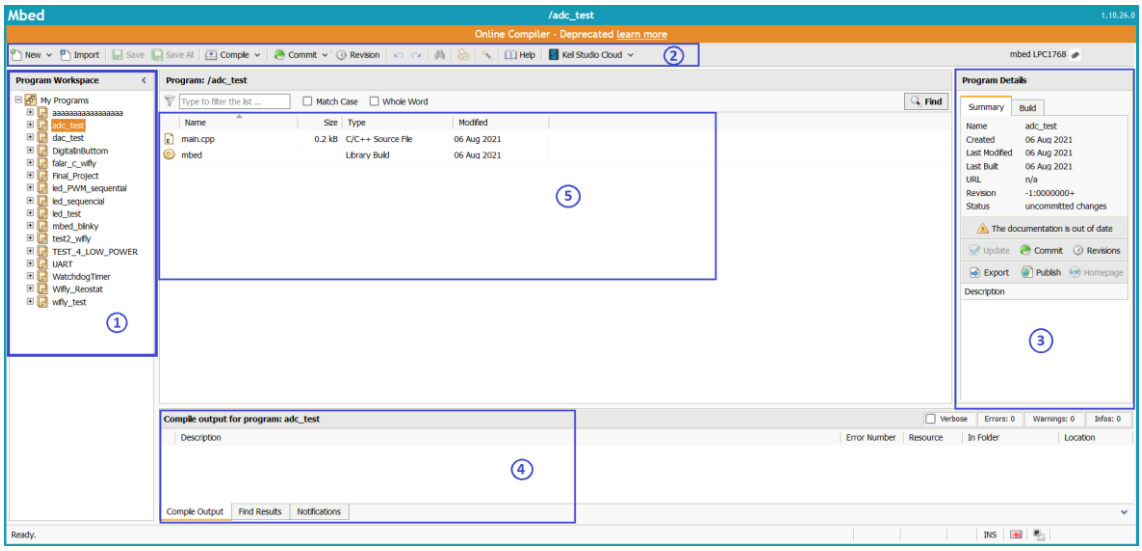

Figure 4.1 – Mbed online IDE.

To start a project, first, the compiler needs to know which mbed board it will target, in this work, as already mentioned, there is a subjecting of the theoretical work into a mbed LPC1768. Once the IDE has the tools necessary to build and compile programs on the target board, the programmer is ready to start implementing some extern libraries, to import some necessary features into the program.

In the developed program on the mbed compiler, mainly 2 libraries were used, 1) mbed and 2)WiflyInterface, a link to the documentation of these libraries is found in the appendix. The mbed library is responsible for the definition of some important macros, and to permit the function of some board features throughout code, like activation of drivers and I/O, detection of events, usage of clocks, memory, and other internal peripheries. The WiflyInterface library permits the exchange between the communication module and the main board, it is also used to create a TCP socket connection, which will be relevant as will be seen ahead.

4.2. Selected hardware

Within the LPC176x family, exists the Arm LPC1768 board based on the Cortex-M3 processor, this processor was thought for high-performance, up to 100MHz clock frequency, and low-cost, relatively cheap when compared with other marketed units, embedded applications within microcontrollers. The board features a high level of integration, low power consumption and some enhanced debug features.

The peripheral components within the board, include a 512 kB flash memory, a 64 kB SRAM with separate access paths, ethernet MAC, USB device/host interface, DMA controller, 4 UARTs, 2 CANs, 2 SSP controllers, SPI, I2C bus, ADC, DAC, PWM control, WDT, 4 independent

timers, an ultra-low power RTC with separate energy supply and up to 70 GPIO pins. It supports both In-System programming and In-Application programming. A photograph of the target board is seen in the following figure, fig 4.2.

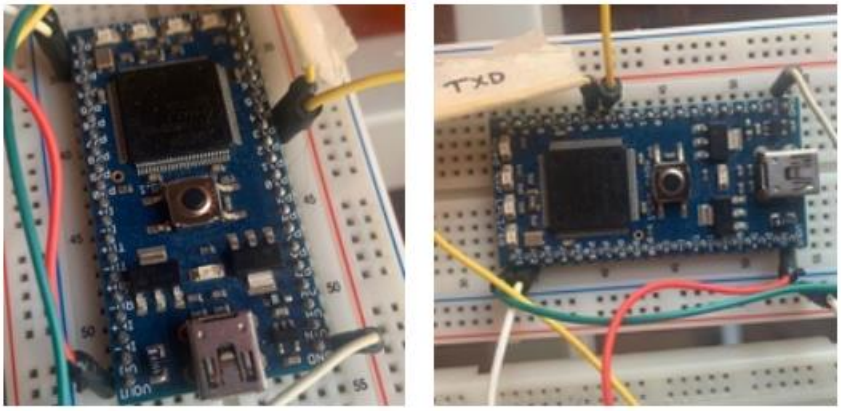

Figure 4.2 – Side and front view of the LPC1768 board.

The LPC1768 board stands out in various aspects, one of them, as already mentioned, is its relatively low cost, which makes it accessible within the microcontroller market, other important features are the ability to differentiate SRAM in various independent blocks allowing for higher throughput, all the GPIO pins have annexed a pull-up or down resistor and even an open-drain operating mode, RTC has a separate power supply and a dedicated oscillator, each peripheral has its own clock divider allowing further power saving, there is the presence of an integrated power management unit that adjusts internal regulators to also minimize power expenditure, it is capable of permutating between four different modes: 1) sleep, 2)deep sleep, 3) power-down and 4) deep power-down. A 3.3V power supply that can go as low as 2.4V and a wake-up interrupt controller that allows the CPU to wake up if an urgent interrupt occurs, even if all clocks are not functioning. The power consumption can go as low as 50mA, even though is not on the scale of state-of-art ultra-low power MCUs it can reach a current saving of up to 74% when compared to the normal operation mode. The sum of these features makes it a solid choice for the present work [40].

The communication module, WiFly RN-171, seen in fig 4.3. is not directly related to the Mbed OS, even though there is a header file in the mbed database, called ‘WiflyInterface.h’, that programs some functions to help the communication module interface with the board, the link to the library is found in the appendix, meanwhile, a description of the WiFly RN-171 is found in section 3.4.

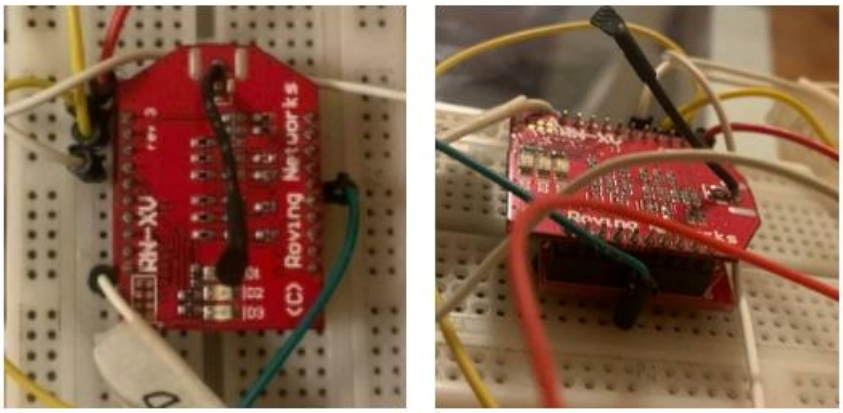

Figure 4.3 – Side and front view of WiFly RN-171.

4.3. Mbed OS process flow

As seen in section 3.4, the server (MATLAB) will need to establish a connection with the communication module (WiFly RN-171), and therefore with the main board (LPC1768), this connection is possible with the use of a TCP socket connection.

A TCP socket connection uses the Transmission Control Protocol and it requires three divisions or frames to set up a successful connection, 1) the SYN frame, 2) the SYN-ACK frame and 3) the ACK frame. The variables needed to perform a socket connection are the IP of both the server and the client as well as the port connection, this type of socket guarantees that all data is received acknowledged, and sent, thus being reliable. This type of connection works by an in-order data delivery. This process is visualized in the following figure, fig. 4.4. [41]

As soon as the connection is established, the client can write the periodic fetched data, for example, a temperature reading, into the buffer for the server to fetch. This kind of socket is also advantageous because it does not need to keep refreshing the connection status of the channel, unlike, for example, an HTTPS connection, where multiple requests must be made to keep the data flowing, this makes the communication less energy-hungry and more reliable in a real-time scenario.

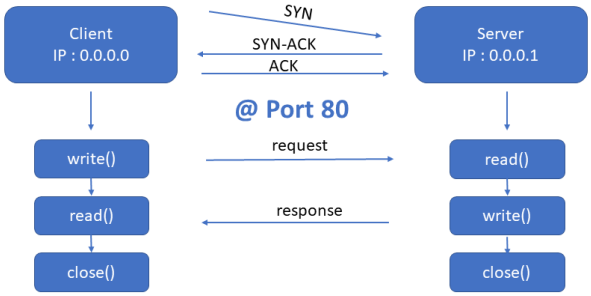

Figure 4.4 – TCP socket connection.

As pointed out, there is the need to create two instances to establish the socket, one entity must be the server, which provides the connection, and the other entity must be the client, which is present within the channel, ready to receive and send data. On the server side, within MATLAB, official MathWorks documentation was used to help establish the server [42], a little code snippet is found in the appendix for further detail.

On the other side, within the Mbed OS, it is required to know the server IP, and the connection port, and to be connected within the same network. With the meeting of these conditions and variables, and with the resources provided by the WiflyInterface library, it is possible to connect the client to the server, keeping the flow of data, a code snippet can also be found, within an online repository, stated in the appendix.

Other libraries, on the mbed OS, are also relevant to the program, like the 'WiFlyLowPower.h' and the 'PowerControl.h', which will be discussed and further analyzed in the next chapter within the low power capabilities section.

The complete process flow within the OS would be as followed:

1- Connect the communication module to the grid;
2- Send a request to join the server TCP/IP socket;
3- SYN and ACK frames are received and sent, establishing a successful connection;
4- The board sends data to the buffer, of a physic metric, read from an analog sensor;
5- Within the same frame, the board sends the acquired voltage levels, fetched from the environment in the last period instances.
6- The board and communication module go to idle mode, waiting for information back from the server;
7- The board fetches, from the server, a result of the algorithm, the duration of the sleep period, T_{Ch};
8- The board and communication module wake up, after the imposed duration;
9- Go back to step 1.

Chapter 5

Experimental evaluation

In the present chapter, a discussion of the work methodology is carried out, addressing the implementation of the selected meta-heuristic algorithm, NSGA-II, as well as a review on the low power capabilities of the modules used, lastly, a definition and performance of energy tests is established in order to do a critical analysis and evaluation.

The activities done in this dissertation were organized and executed in the following steps:

- Survey of the state of art in energy efficiency and metaheuristic algorithms.
- Survey of the power saving methodologies in a typical mote.
- Research and read up on the Mbed OS.
- Architecture proposition of the system.
- Formulation of the mathematic model including restrictions and boundaries.
- System prototyping.
- Evaluation of the most adequate meta-heuristic algorithm.
- Codification of the chosen algorithm for the multi-objective problem at hand, using MATLAB.
- Validation of the solution, based on the tests performed.

5.1. NSGA-II implementation

As stated previously, to ensure that the proposed synchronization can operate in devices with energy restrictions, such as those that obtain energy from energy harvesting, there is a need to find an optimal point of operation, establishing a commitment between the energy needed to perform communication in the assigned timeslot, ΔE_C, and the time available to collect energy, T_{ch}, needed to obtain the quantity of energy $E\Delta_j$, by maximizing one and minimizing the other, respectively, and simultaneously.

The system faces a problem whose optimization objective is not unique, given so, a multi-objective function is required. Using heuristics, it is possible to implement an algorithm that is based on a pedagogical process, where some elements discover what they want to learn and keep, by posing questions among them, aiming to find one or more optimal solutions to the particular problem.

However, it is crucial to understand that the optimal point is uncertain within the context of multi-objective optimization. What is feasible is not one unique solution, but the best

harmonization from a set of optimal solutions, this is known as the optimal set, localized in the Pareto curve. More information regarding the subject is discussed in section 2.3 where MH algorithms and NSGA-II are introduced.

The objective functions that will be optimized in the server are,

$$\min_{\alpha \in \mathbb{R}} \; f_1(\alpha, T_{ch}) = \alpha(t) T_{ch},$$

$$\max_{P_{EH} \in \mathbb{R}} \; f_2(P_{EH}, T_{ch}) = P_{EH}(t) T_{ch}, \qquad 0 \leq T_{ch<\infty},$$

In the effort of achieving faster running times on the server-side and simpler computation complexities, both functions are simultaneously minimized by,

$$\min_{\alpha \in \mathbb{R}} \; f_1(\alpha, T_{ch}) = \alpha(t) * T_{ch},$$

$$\min_{P_{EH} \in \mathbb{R}} \; f_2(P_{EH}, T_{ch}) = -(P_{EH}(t) * T_{ch}), \quad 0 \leq T_{ch<\infty},$$

The variable α, a linear value that derives from the characteristics of the WiFly clock, due to the requirement of extensive testing, will be considered a fixed value for demonstration purposes, this value has a direct proportion to the amount of time passed since a working cycle begins. The standard procedure for measuring a clock jitter suggests an extensive reading by the random measurement between two clock periods and by taking the absolute difference between the result, it requires a high number of samples to acquire a viable value, so the chosen value was chosen arbitrarily, within realist parameters.

The function $\alpha(t)$ will then be represented by,

$$\alpha(t) = 0.02.t\,, \qquad 0 \leq t < \infty, t \in \mathbb{R} \tag{5.1}$$

Ideally the function $P_{EH}(t)$, would be extrapolated, according to the latest values of voltage across the capacitor, read from the EH module. However, in the present work, a fixed EH curve will be used, given by the following function

$$P_{EH}(t) = sin\left(t + e^{\cos(\sqrt{7}+t)}\right) + 1.73, \qquad 0 \leq t < \infty, t \in \mathbb{R} \tag{5.2}$$

After the server simulates all iterations, given the objective functions, it will return to the client one of the feasible solutions containing the optimal T_{ch}.

5.2. Low power capabilities

Reducing the power when it is not required has a lot of advantages that can directly impact the lifetime, performance, sustainability, and cost of a determined application, ideally an application of any embedded system, when possible, should operate without the need to store energy, and to be fully capable of performing under a continuous EH regime. In this work, is crucial that the minimization of the energy consumption is performed in any way possible to achieve the maximum operation time of the system, making it more prone to capture recurrent data from the environment and respond accordingly to a real-world scenario.

In embedded systems, the current technology used for CMOS integrated circuits, the power consumption comes in a simple form as the next formula.

$$Power = CapLoad * V^2 * ClockFreq \tag{5.3}$$

To reduce power, any of those terms need to be minimized, with special emphasis on voltage, given that it possesses a squared exponent.

Reducing the capacitive load of a CMOS IC depends primarily on the way the gate technology of transistors works, and clearly on the number of transistors per area. It is extremely impractical to change the gate technology, but even so, it is possible in a typical MCU to turn off various peripherals that are not being used [43].

Reducing the voltage levels is impractical for a customer because they are fixed to the retail product and it is not feasible to change the interiors of a logic board to change the voltage applied in all peripherals, what the customer can do is look for units that have lower supply voltages, like 3V3 instead of the traditional 5V. Some processors can even vary the voltage at the core depending on the existing computational complexity [43].

To reduce the clock frequency the processor must be able to adjust its frequency according to the tasks being executed and the readiness execution time of the given tasks. A faster clock will boost performance, but it will increase power levels. Turning off the clock or even slowing it down whenever speed is not relevant is an adequate approach to save power [43]. Even so, it is important to not fully turn off the board as there is important data stored in volatile memory or registers, that cannot be permanently lost.

5.2.1. Mbed

The low power capabilities present in the LPC1768, were put to the test by resorting to the "PowerControl/EthernetPowerControl.h" library written by Michael Wei [40], this library takes control of the board power management and Cortex-M3 power control functions. The functions are meant to turn off some peripherals including, most clocks, PLLs, flash memory, brown-out detection circuit, and PHY. There is also a method, involving clock control, that can be used to change clock cycles to save energy, yet it was not employed in the work, clock cycles were left

running in the normal operation mode at around 96 MHz. The Cortex-M3 possesses 4 different energy-saving modes, sleep, deep sleep, power down, and deep power-down.

All peripherals are to be turned off by the PHY layer, and the sleep mode is the preferable induce mode. When sleep mode is entered, the main clock is stopped, execution of instructions and tasks is put on halt until either a reset or an interrupt comes up in ISR, peripheral functions continue operation unless they are coded to turn off, these peripherals can be used to generate interrupts to resume execution of the processor, it is important to also note that all dynamic power is eliminated.

To wake up the board, WDT is programmed, the timer fetches the T_{ch}, a value that is led by the meta-heuristic algorithm running on the server side, after T_{ch} a kick is generated into the watchdog and the board resets, resuming operation, the flow of the process is visualized in the following figure, fig. 5.1.

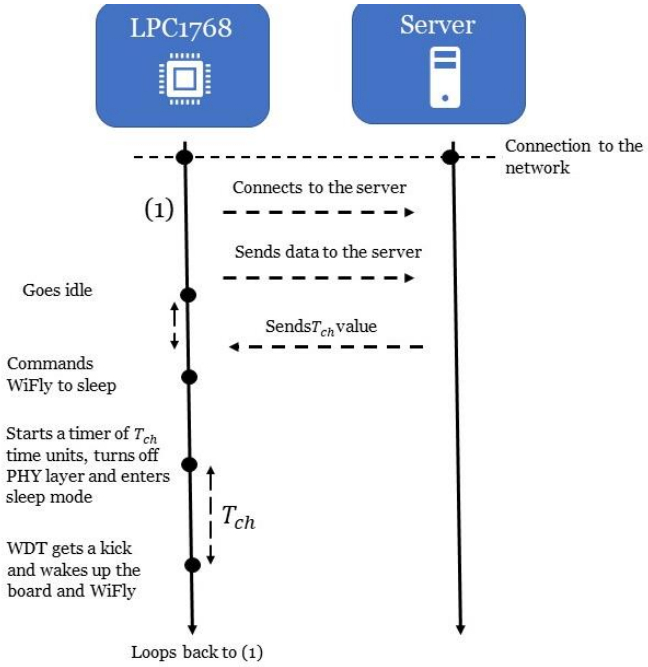

Figure 5.1 – Sleep/wake schedule.

Note that the data sent to the server, would not only be the data related to the harvested energy but also data related to the analog readings from the sensor.

5.2.2. Wifly

The communication module used in the present work is the WifLy RN171 wireless LAN module, it is possible to manage the power usage of this unit through an ultra-low-power sleep mode that consumes approximately 4μA, a battery boost control circuitry, and an RTC capable of time stamping, auto-sleep and auto-awake. It also has a low-current consumption mode that

consumes 40mA on RX and 120mA at 0 dBm on TX. It is capable of transitioning from sleep to CPU-active in 1.7 ms, and CPU.active to network connection in less than 35ms.

There are a set of commands that can be sent to the communication module, via UART, to change its operation metrics. Regarding the transmission power, in RN171 this variable is configurable by setting forth the *'set WLAN tx <value>'* command to the module. Allowing the TX to vary its power from 0 to 12 dBm.

To induce the module to sleep three main ways are used by the programmer, a sleep command through UART, a sleep time controlled by the internal RTC, and interfacing with GPIOs.

In this work, the sleep induction is being performed through a command via UART and the force awake is being triggered by a system reset controlled by the program, the state machine diagram, regarding this commutation, is visualized in the following image, fig 5.2.

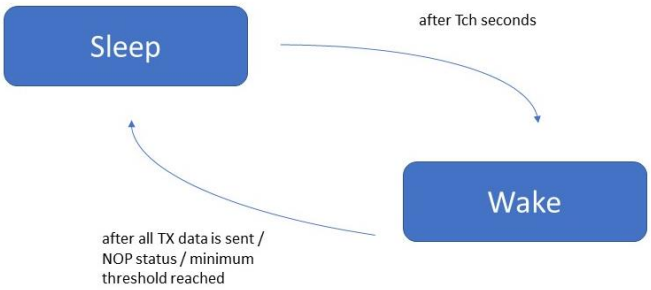

Figure 5.2 – State machine diagram of sleep/wake system.

The variable T_{ch}, as discussed earlier, will be returned by the NSGA-II, and will permit the system to function in a dormant mode, to harvest the ideal quantity of energy. Considering the induction of sleep, one of three conditions is met for this action to enroll: 1) all TX data is sent, 2) the unit is doing nothing relevant (NOP status), or, 3) the minimum energy threshold of capacitor energy storage was reached. Regarding 3), this threshold would be established, in a way that the system would rarely encounter a situation where energy available is scarce. During the transition between wake and sleep, the module must reconnect to the network. A library, 'WiFlyLowPower.h' was written and used by the author of the present work to implement functions that test, send commands through the UART, reset, connect and do other features, a link to an online repository is found in the appendix.

5.3. Energy tests and algorithm performance

The presenting section reveals energy tests performed on the system, while commutating between different operation modes, in order to annotate the difference in power consumption. Algorithm performance is also tested and simulated in the last subsection.

5.3.1. Energy tests execution

The current subsection presents the energy readings from energy mode commutation in both the board and communication module, the readings were made using an oscilloscope and a +5v power supply, by measuring the voltage drop in a 1Ω resistor between GND and the modules. The experimental schematic for the readings is visualized in the following figure, fig 5.3.

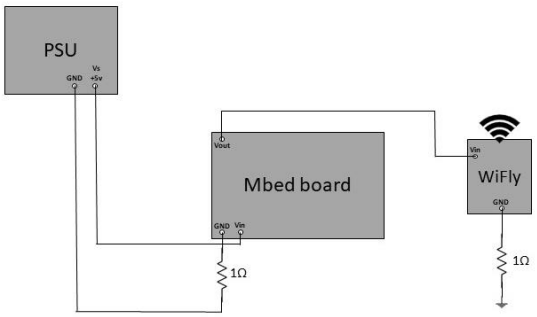

Figure 5.3 – Experimental schematic for energy readings.

In the following figure, fig 5.4, on the left side it is represented the commutation between normal mode and off (0mA), of the Mbed LPC1768 board, the normal mode consists of an infinite while loop, while executing defined tasks, in this mode, current drawn is approximately 200mA, with consumption of 1W. On the right, with a 140mA offset, the board starts in PowerDown mode, where all clocks are disabled as well as flash memory, exhibiting a current drawn of approximately 130mA, consuming 0.65W, going after to normal mode with a peak value of 205mA, after around 5s the board is induced to sleep, on this operation mode the clock to the core is disabled and code execution is halted, a current fall is observed to approximately 180mA consuming 0.9W.

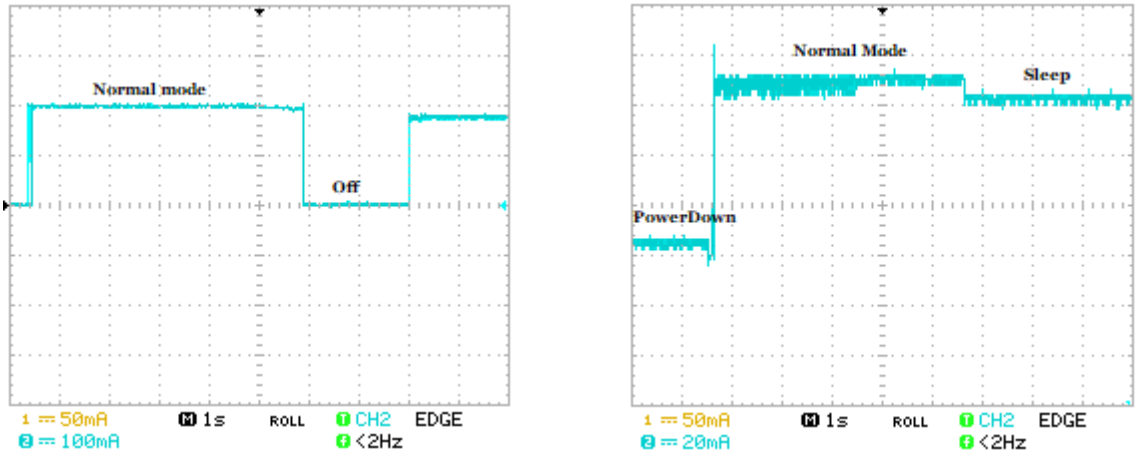

Figure 5.4 – Operation modes commutation pt.1.

In the next figure, fig.5.5, on the left side is represented the commutation between normal to DeepSleep mode, with a 140mA offset, when the board is induced into DeepSleep, the majority of clocks are turned off as well as the PLLs circuit are disabled, at this mode, there is a current drawn fall to approximately 140mA with a consumption of 0.7W. On the right side, normal mode is commuted with DeepPowerDown, a mode where only the RTC and its registers are functional as well as the RESET pin, at this mode, it is observed a current drawn of 120mA with a consumption of 0.6W.

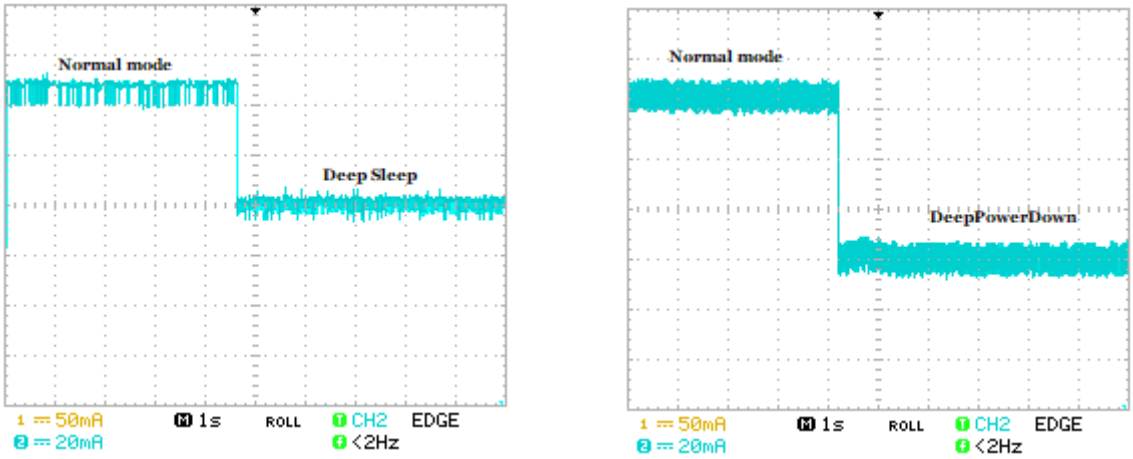

Figure 5.5 – Operation modes commutation pt2.

In the final readings figure, fig. 5.6, the reader can observe, on the left side, the board going from DeepSleep mode into turning off the PHY circuit, disabling all peripherals, and then eventually going into DeepPowerDown, note that by doing this sequence of actions, the lowest current drawn recorded value is obtained, presenting a current draw of 89 mA with consumption of 0.45W. On the right side, regarding the communication module, the WiFly RN-171 is seen turning on and connecting to both the network and server socket, peaking at a current draw value of 400mA with a 2W consumption, after the module is connected it goes to idle mode until TX is requested, idle mode represents a current draw of 30mA consuming

0.15W. When frames are transmitting the current draw peaks at approximately 120mA with a 0.6W consumption. Note that the TX power bit value is set as 0, on the module chip, from a scale from 0 to 12, where 0 represents around 12dBm, according to the manufacturer.

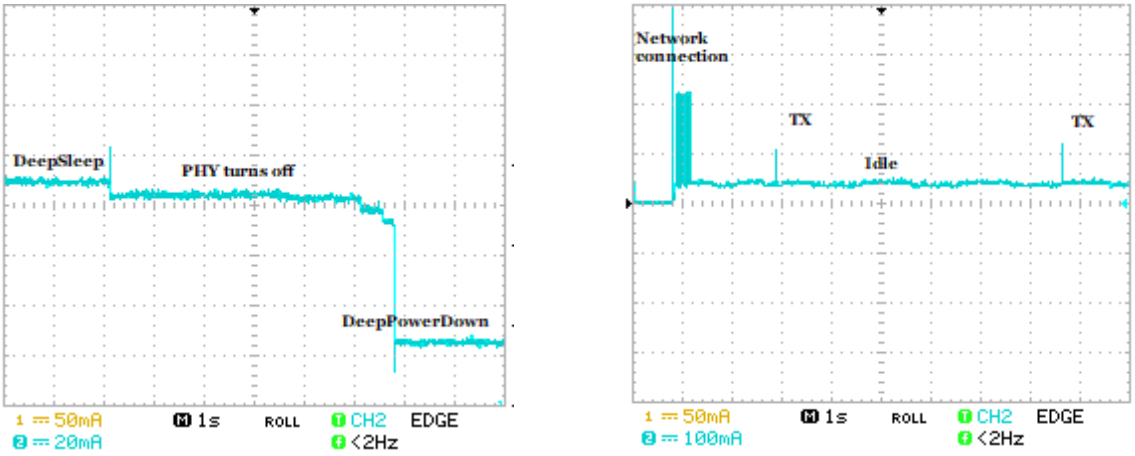

Figure 5.6 – Operation modes commutation pt3.

Table 5.1 shows the comparison between all present operation modes and the % of normal mode consumption for both the mbed LPC1768 board and WiFly RN-171. Note that the values shown are the read peak values instead of the RMS value.

Table 5.1.: Compared values between operation modes

mbed LPC1768	**Current drawn (mA)**	**Power consu. @5V (W)**	**% of normal mode**
Normal mode	205	1.03	100%
Sleep	180	0.9	87.4%
DeepSleep	140	0.7	68.0%
PowerDown	131	0.65	63.1%
DeepPowerDown	120	0.6	58.3%
PHY off + DPD	89	0.45	43.7%
WiFly RN-171	**Current drawn (mA)**	**Power consu. @5V (W)**	**% of normal mode**
Idle	30	0.15	100%
Network/socket connection	400	2	1333%
TX	120	0.6	400%

5.3.2. Algorithm performance

In the current subsection, the devised algorithm, based on the NSGA-II explored during the resolution of this work, will be demonstrated based on its performance and output results. As seen in section 5.1. the algorithm will run the simulation, with two objective functions at the core of its assignment, f_1 and f_2, where $f_1(T_{ch}) = \alpha(t) * T_{ch}$ and $\alpha(t) = 0.02 * t$, standing for $T_{\Delta j}$, and $f_2(T_{ch}) = P_{EH}(t) * T_{ch}$ and $P_{EH}(t) = sin\left(t + e^{\cos(\sqrt{7}+t)}\right) + 1.73$, standing for $E_{\Delta T_{ch}}$. Both functions, $\alpha(t)$ and $P_{EH}(t)$ represent theoretical values, $\alpha(t)$ being based on theoretical clock jitter between the board and the communication module, and $P_{EH}(t)$ based on theoretical fetched energy readings.

The simulation seen in fig. 5.7 was performed with a population of 100 particles for a maximum of 25 iterations. All particles were submitted and restricted to the condition $E_{\Delta T_{ch}} > E_{\Delta c}$, as explained in section 3.4, the boundaries set suppose a period not greater than 6.5s for T_{ch}, but when regarding the variable t, the boundary is set within a maximum arbitrary chosen value depending on the $T_{\Delta c}$ maximum established duration, this value was calculated by the following expression $t_{max} = \max(T_{ch}) + \max(T_{ch}) * \frac{1}{3} = 8.67s$, therefore $0 \leq T_{ch} \leq 6.5[s] \wedge 0 \leq t \leq 8.67[s]$. Note that T_{ch} is the EH period and t is the period that starts counting between every operation cycle, this is $t = (T_{ch}(it) + T_{\Delta c}(it)) - t_i\,[s]$, t_i is always zero. Note, that $E_{\Delta c}$ was initialized with a value of 0.3J.

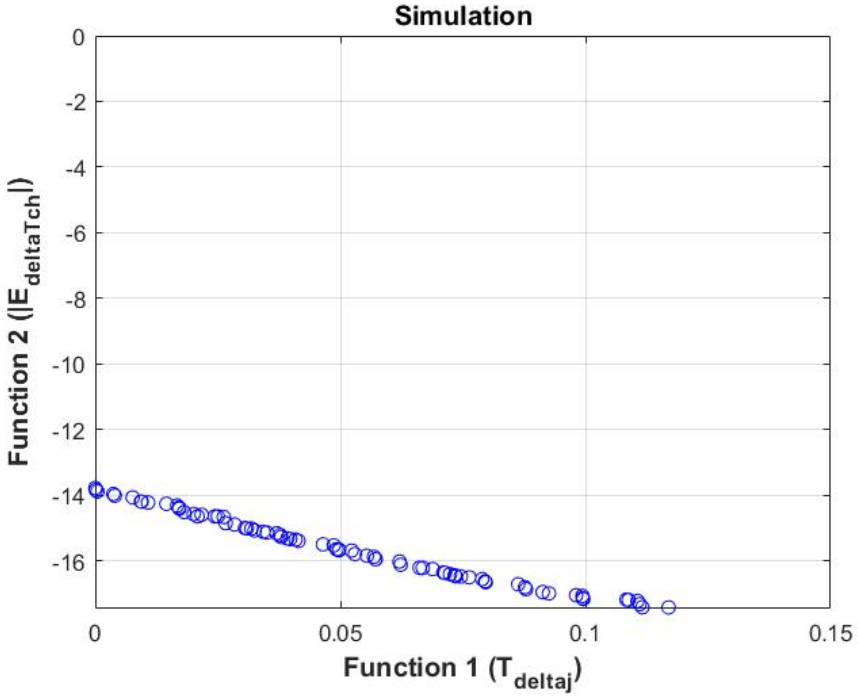

Figure 5.7 – Particle distribution across Pareto front.

Note that the y-axis, shows a negative value, but in reality, the value is absolute, this happens because f_2 or $E_{\Delta T_{ch}}$, is inverted, due to its original maximization, as discussed in section 5.1. This

function requires to be minimized because both input objective functions, on the algorithm, must be simultaneously minimized or in alternative maximized, f_2 was the chosen function to be inverted, even though f_1 could also be inverted to generate a max-max optimization.

As seen in prior chapters on multi-objective problems, the proposed solutions are all considered mathematically acceptable, if within the Pareto curve, in fig. 5.7. Pareto curve is observed, the curve was established after 25 iterations and any of the presented solutions are viable, therefore a random choice from the total population is performed.

In solution 94, it is verified an optimization output of $T_{ch} = 6.437[\text{s}]$, which results in an uncertainty time interval of $T_{\Delta_j} = 0.111[s]$. Variable t is returned, by the particle no.94, with $t = 0.859 + T_{\Delta c}(it)\ [s]$, $T_{\Delta c}(it)$ mean value is assumed as a third of the maximum EH period. This means that regarding the last known energy harvested curve, at $t = 3.025[s]$ the instantaneous power read from the EH module, $E\Delta T_{ch}(7.296) = 17.208\ [p.d.u]$, it is at the optimal point, for the given objective function results, this reasoning is visualized in figure 5.8. Note that if another particle is chosen from the population, other than no. 94, results can slightly vary.

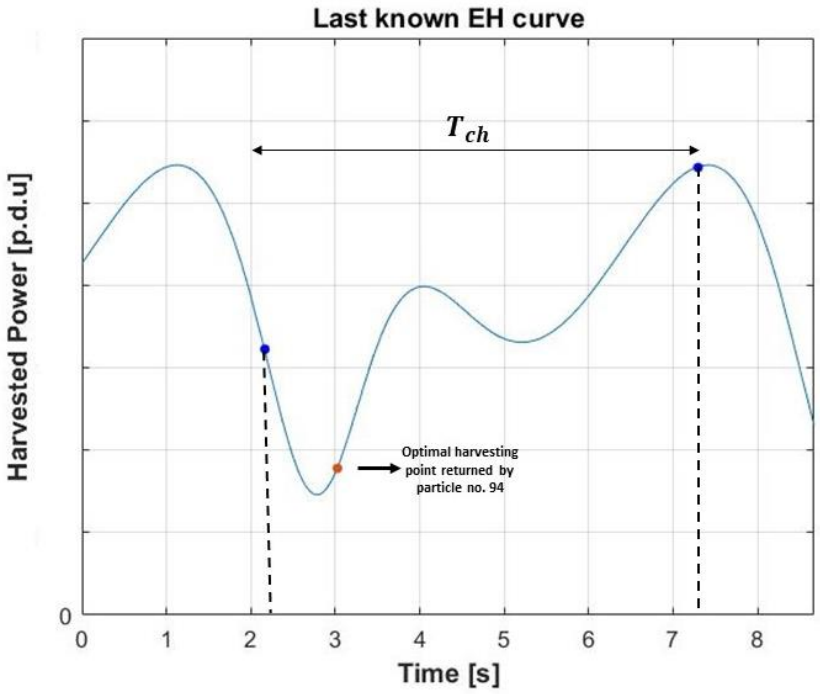

Figure 5.8 – EH curve visualization with optimal outputted t, for particle no.93.

An illustration based on the outputted variables is seen in figure x, the illustration serves as a support to visualize what happens within the system.

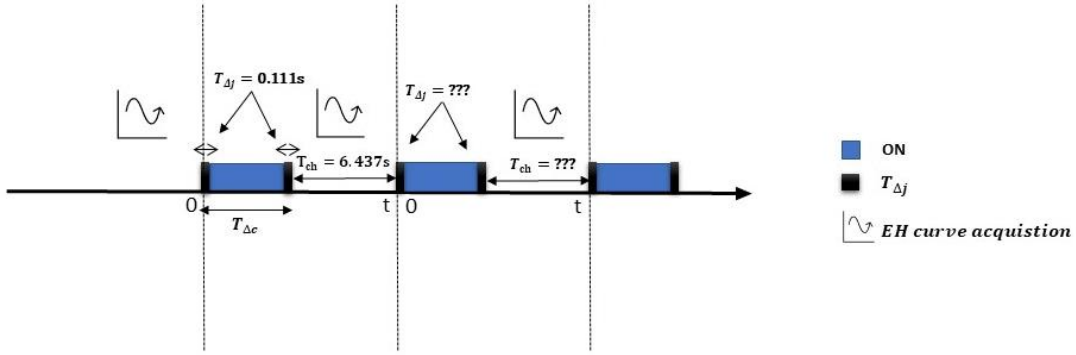

Figure 5.9 – System cycles.

Note that when a harvesting cycle ends, the board is waked up, resetting clock jitter and thus, starting again from $t = 0$, restarting the process over again.

Source code for the algorithm and workspace variables from the current simulation is given in the repository annexed by the appendix.

CHAPTER 6

Conclusion

6.1. Main achievements

At the beginning of this work, an assortment of goals was set, the final achievements go according to what was established, a wide review was performed on state-of-art scheduling methodologies, from techniques performed on hardware, like DVFS to duty cycling algorithms running on external servers. Meta-heuristic algorithms were also extensively reviewed, discussing from single objective to multi-objective and addressing heuristics from different inspiration areas, like social, physics, and biology-based algorithms. After the state-of-art techniques were assessed, NSGA-II was chosen in order to base the devised presented algorithm, alternating the board duty cycle, its performance was documented and analyzed demonstrating the expected output while alternating the energy harvesting period as a function of prior fetched energy values. An experimental setup was also proposed, in which its architecture is similar to one of a typical WSN mote.

6.2. Future work

Having in mind that the input data, inserted into the algorithm, was based on theoretical values, it would be interesting to, in future works, assess this data as practical values read from the environment, for example $P_{EH}(t)$ instead of being a fixed curve, it could be the voltage values fetched periodically from the EH module. As well as $\alpha(t)$, the jitter calculation between module clocks, could be practically demonstrated by testing and sampling various times. In a post hypothetical work, it would also be interesting the insertion of another state-of-art algorithm or method to compare the energy efficiency differences.

6.3. Final feedback

The NSGA-II inspired algorithm, has shown how it can guarantee that the proposed synchronization can operate in devices with energy restrictions, with the calculation of a set of possible optimal solutions, the Pareto curve, it is possible to establish a commitment between the energy needed to perform communication as well as analogic data fetching, ΔE_C , and time duration, T_{ch}, available to harvest sufficient energy, $E\Delta_J$, without the system going offline, by consistently meeting the $E_{\Delta T_{CH}} > E_{\Delta C}$ condition. This, in addition to the low power capabilities presented on the system board, and communication module make the system closer to reaching the ENO condition.

Bibliography

[1] K. Akkaya, I. Guvenc, R. Aygun, N. Pala, and A. Kadri, "IoT-based occupancy monitoring techniques for energy-efficient smart buildings," in *2015 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, 2015, pp. 58–63. doi: 10.1109/WCNCW.2015.7122529.

[2] H. Attar, M. R. Khosravi, S. S. Igorovich, K. N. Georgievan, and M. Alhihi, "Review and performance evaluation of FIFO, PQ, CQ, FQ, and WFQ algorithms in multimedia wireless sensor networks," *Int J Distrib Sens Netw*, vol. 16, no. 6, p. 1550147720913233, Jun. 2020, doi: 10.1177/1550147720913233.

[3] U. Shafi *et al.*, "A Novel Amended Dynamic Round Robin Scheduling Algorithm for Timeshared Systems 1," *International Arab Journal of Information Technology*, vol. 16, Mar. 2019, doi: 10.34028/iajit/17/1/11.

[4] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Comput Oper Res*, vol. 13, no. 5, pp. 533–549, 1986, doi: https://doi.org/10.1016/0305-0548(86)90048-1.

[5] H. Robbins and S. Monro, "A Stochastic Approximation Method," *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, Sep. 1951, doi: 10.1214/aoms/1177729586.

[6] M. Abdel-Basset, L. Abdel-Fatah, and A. K. Sangaiah, "Chapter 10 - Metaheuristic Algorithms: A Comprehensive Review," in *Intelligent Data-Centric Systems*, A. K. Sangaiah, M. Sheng, and Z. B. T.-C. I. for M. B. D. on the C. with E. A. Zhang, Eds. Academic Press, 2018, pp. 185–231. doi: https://doi.org/10.1016/B978-0-12-813314-9.00010-4.

[7] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "GSA: A Gravitational Search Algorithm," *Inf Sci (N Y)*, vol. 179, no. 13, pp. 2232–2248, 2009, doi: https://doi.org/10.1016/j.ins.2009.03.004.

[8] D. Shen, T. Jiang, W. Chen, Q. Shi, and S. Gao, "Improved chaotic gravitational search algorithms for global optimization," in *2015 IEEE Congress on Evolutionary Computation (CEC)*, 2015, pp. 1220–1226. doi: 10.1109/CEC.2015.7257028.

[9] V. Feoktistov, Ed., "Differential Evolution BT - Differential Evolution: In Search of Solutions," Boston, MA: Springer US, 2006, pp. 1–24. doi: 10.1007/978-0-387-36896-2_1.

[10] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, 1995, vol. 4, pp. 1942–1948 vol.4. doi: 10.1109/ICNN.1995.488968.

[11] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[12] G. Venter and J. Sobieszczanski-Sobieski, "Particle Swarm Optimization," *AIAA Journal*, vol. 41, no. 8, pp. 1583–1589, Aug. 2003, doi: 10.2514/2.2111.

[13] X.-S. Yang, "Firefly Algorithms for Multimodal Optimization BT - Stochastic Algorithms: Foundations and Applications," 2009, pp. 169–178.

[14] V. C. S. S. and A. H. S., “Nature inspired meta heuristic algorithms for optimization problems,” *Computing*, vol. 104, no. 2, pp. 251–269, 2022, doi: 10.1007/s00607-021-00955-5.

[15] N. Srinivas and K. Deb, “Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms,” *Evol Comput*, vol. 2, no. 3, pp. 221–248, 1994, doi: 10.1162/evco.1994.2.3.221.

[16] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multi-objective genetic algorithm: NSGAII,” *IEEE Transactions on Evolutionary Computation*, vol. 6, Aug. 2002.

[17] M. M. Sandhu, S. Khalifa, R. Jurdak, and M. Portmann, “Task Scheduling for Simultaneous IoT Sensing and Energy Harvesting: A Survey and Critical Analysis,” *arXiv: Signal Processing*, 2020.

[18] C. Delgado and J. Famaey, “Optimal Energy-Aware Task Scheduling for Batteryless IoT Devices,” *IEEE Trans Emerg Top Comput*, vol. 10, no. 3, pp. 1374–1387, 2022, doi: 10.1109/TETC.2021.3086144.

[19] K. S. Yldrm, A. Y. Majid, D. Patoukas, K. Schaper, P. Pawelczak, and J. Hester, “InK: Reactive Kernel for Tiny Batteryless Sensors,” in *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*, 2018, pp. 41–53. doi: 10.1145/3274783.3274837.

[20] C. D. Systems, “Dynamic Voltage and Frequency Scaling (DVFS).” https://semiengineering.com/knowledge_centers/low-power/techniques/dynamic-voltage-and-frequency-scaling/ (accessed Feb. 26, 2022).

[21] A. Ravinagarajan, D. Dondi, and T. Rosing, *DVFS based task scheduling in a harvesting WSN for Structural Health Monitoring*. 2010. doi: 10.1109/DATE.2010.5457052.

[22] V. Hoang, N. Julien, and P. Berruet, “Increasing the autonomy of wireless sensor node by effective use of both DPM and DVFS methods,” in *2013 IEEE Faible Tension Faible Consommation*, 2013, pp. 1–4. doi: 10.1109/FTFC.2013.6577766.

[23] M. K. Bhatti, C. Belleudy, and M. Auguin, “An inter-task real time DVFS scheme for multiprocessor embedded systems,” in *2010 Conference on Design and Architectures for Signal and Image Processing (DASIP)*, 2010, pp. 136–143. doi: 10.1109/DASIP.2010.5706257.

[24] J. Zhuo and C. Chakrabarti, “System-level energy-efficient dynamic task scheduling ,” *42nd Design Automation Conference, DAC 2005* . pp. 628-631 BT-Proceedings-Design Automation Con, 2005. [Online]. Available: http://www.scopus.com/inward/record.url?scp=27944458086&partnerID=8YFLogxK

[25] A. Manzak and C. Chakrabarti, “Variable voltage task scheduling algorithms for minimizing energy/power,” *IEEE Trans Very Large Scale Integr VLSI Syst*, vol. 11, no. 2, pp. 270–276, 2003, doi: 10.1109/TVLSI.2003.810801.

[26] B. Bailey, “Is DVFS Worth The Effort?,” 2020. https://semiengineering.com/is-dvfs-worth-the-effort/ (accessed Mar. 02, 2022).

[27] W. Liu, Z. Zhang, M. Li, and Z. Liu, “A Trustworthy Key Generation Prototype Based on DDR3 PUF for Wireless Sensor Networks,” in *2014 International Symposium on Computer, Consumer and Control*, 2014, pp. 706–709. doi: 10.1109/IS3C.2014.188.

[28] P. Schmitz, Marcus T., Al-Hashimi, Bashir M. and Eles, *System-Level Design Techniques for Energy-Efficient Embedded Systems*. Boston: Kluwer Academic Publishers, 2004.

[29] M. Sandhu, S. Khalifa, R. Jurdak, and M. Portmann, *Task Scheduling for Energy Harvesting-based IoT: A Survey and Critical Analysis*. 2020.

[30] A. Y. Majid *et al.*, "Dynamic Task-Based Intermittent Execution for Energy-Harvesting Devices," *ACM Trans. Sen. Netw.*, vol. 16, no. 1, 2020, doi: 10.1145/3360285.

[31] K. Maeng, A. Colin, and B. Lucia, "Alpaca: Intermittent Execution without Checkpoints," *Proc. ACM Program. Lang.*, vol. 1, no. OOPSLA, 2017, doi: 10.1145/3133920.

[32] T. Zhu, A. Mohaisen, Y. Ping, and D. Towsley, "DEOS: Dynamic energy-oriented scheduling for sustainable wireless sensor networks," in *2012 Proceedings IEEE INFOCOM*, 2012, pp. 2363–2371. doi: 10.1109/INFCOM.2012.6195625.

[33] A. Colin and B. Lucia, "Termination Checking and Task Decomposition for Task-Based Intermittent Programs," in *Proceedings of the 27th International Conference on Compiler Construction*, 2018, pp. 116–127. doi: 10.1145/3178372.3179525.

[34] R. C. Carrano, D. Passos, L. C. S. Magalhaes, and C. V. N. Albuquerque, "Survey and Taxonomy of Duty Cycling Mechanisms in Wireless Sensor Networks," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 181–194, 2014, doi: 10.1109/SURV.2013.052213.00116.

[35] A. C. Valera, W.-S. Soh, and H.-P. Tan, "Survey on wakeup scheduling for environmentally-powered wireless sensor networks," *Comput Commun*, vol. 52, pp. 21–36, 2014, doi: https://doi.org/10.1016/j.comcom.2014.05.004.

[36] J. Hsu, S. Zahedi, A. Kansal, M. Srivastava, and V. Raghunathan, *Adaptive Duty Cycling for Energy Harvesting Systems*, vol. 2006. 2006. doi: 10.1109/LPE.2006.4271832.

[37] X. Zhang, C. Wang, and L. Tao, "An Opportunistic Packet Forwarding for Energy-Harvesting Wireless Sensor Networks With Dynamic and Heterogeneous Duty Cycle," *IEEE Sens Lett*, vol. 2, no. 3, pp. 1–4, 2018, doi: 10.1109/LSENS.2018.2849366.

[38] C. M. Vigorito, D. Ganesan, and A. G. Barto, "Adaptive Control of Duty Cycling in Energy-Harvesting Wireless Sensor Networks," in *2007 4th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, 2007, pp. 21–30. doi: 10.1109/SAHCN.2007.4292814.

[39] H. Rocha, J. Pereira, T. Rodrigues, A. Santo, and A. Salvado, "An Energy-Efficient Process for Optimal Communication Synchronization in Low Power Wireless Smart Sensors," *IEEE International Symposium on Measurements & Networking (M&N)*, 2022.

[40] Michael Wei, "mbed Power Control/Consumption," 2010. https://os.mbed.com/users/no2chem/notebook/mbed-power-controlconsumption/ (accessed Apr. 10, 2022).

[41] howtouselinux, "Understanding TCP Socket With Examples," 2017. https://www.howtouselinux.com/post/tcp-socket (accessed Mar. 07, 2022).

[42] Inc. The MathWorks, "TCP/IP Server Sockets official MathWorks ® documentation." https://www.mathworks.com/help/instrument/communicate-using-tcpip-server-sockets.html (accessed Mar. 10, 2022).

[43] Arm, “Power Management.” https://os.mbed.com/cookbook/Power-Management (accessed Jul. 31, 2022).

Appendix

Online repository on GitHub where the reader can find the source code for the proposed algorithm coded on MATLAB, workspace variables from presented simulation, MATLAB socket creation, and mbed header files devised by the current work's author.

https://github.com/tiago98r/DissertationTiagoRodrigues

The link to the mbed OS library, containing all functions present in the 'mbed.h' header file

https://github.com/ARMmbed/mbed-os

The link to the WiFlyInterface library, containing all functions present in the 'WiFlyInterface.h' header file

https://os.mbed.com/users/samux/code/WiflyInterface/

The link to the LowPower library, containing all functions present in the 'PowerControl.h' header file

https://os.mbed.com/users/no2chem/code/PowerControl/