Flavio Rubens Massaro Júnior

Configuring Mode Changes in Fixed-Priority Preemptively Scheduled Real-Time Systems

Configuração de Mudanças de Modo em Sistemas de Tempo Real Escalonados com Política Preemptiva de Prioridade Fixa

Limeira

2015

UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Tecnologia

Flavio Rubens Massaro Júnior

Configuring Mode Changes in Fixed-Priority Preemptively Scheduled Real-Time Systems

Configuração de Mudanças de Modo em Sistemas de Tempo Real Escalonados com Política Preemptiva de Prioridade Fixa

Dissertation presented to the School of Technology at the University of Campinas in partial fulfillment of the requirements for the Master's degree in Technology, in the area of Technology and Innovation.

Dissertação apresentada à Faculdade de Tecnologia da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Tecnologia, na área de Tecnologia e Inovação.

Supervisor/Orientador: Prof. Dr. Paulo Sérgio Martins Pedro

Co-supervisor/Co-orientador: Prof. Dr. Édson Luiz Ursini

Este exemplar corresponde à versão final da dissertação defendida pelo aluno Flavio Rubens Massaro Júnior, e orientada pelo Prof. Dr. Paulo Sérgio Martins Pedro.

Limeira

2015

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Faculdade de Tecnologia
Felipe de Souza Bueno - CRB 8/8577

M382c Massaro Júnior, Flávio Rubens, 1976-
Configuring mode changes in fixed-priority preemptively scheduled real-time systems / Flávio Rubens Massaro Júnior. – Limeira, SP : [s.n.], 2015.

Orientador: Paulo Sérgio Martins Pedro.
Coorientador: Edson Luiz Ursini.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Tecnologia.

1. Sistemas de tempo real. 2. Escalonamento de processos. 3. Algoritmos genéticos. I. Martins Pedro, Paulo Sérgio,1967-. II. Ursini, Edson Luiz,1951-. III. Universidade Estadual de Campinas. Faculdade de Tecnologia. IV. Título.

Informações para Biblioteca Digital

Título em outro idioma: Configuração de mudanças de modo em sistemas de tempo real escalonados com política preemptiva de prioridade fixa
Palavras-chave em inglês:
Real-time systems
Processor scheduling
Genetic algorithms
Área de concentração: Tecnologia e Inovação
Titulação: Mestre em Tecnologia
Banca examinadora:
Paulo Sérgio Martins Pedro [Orientador]
George Marconi de Araújo Lima
Alexandre Cláudio Botazzo Delbem
Data de defesa: 23-02-2015
Programa de Pós-Graduação: Tecnologia

DISSERTAÇÃO DE MESTRADO EM TECNOLOGIA

ÁREA DE CONCENTRAÇÃO: TECNOLOGIA E INOVAÇÃO

Configuring Mode Changes in Fixed-Priority Preemptively Scheduled Real-Time Systems

FLAVIO RUBENS MASSARO JÚNIOR

A Banca Examinadora composta pelos membros abaixo aprovou esta Dissertação:

[illegible]

PROF. DR. PAULO SÉRGIO MARTINS PEDRO
UNICAMP
Presidente

[illegible]

PROF. DR. GEORGE MARCONI DE ARAÚJO LIMA
UFBA

[illegible]

PROF. DR. ALEXANDRE CLÁUDIO BOTAZZO DELBEM
USP

Abstract

Modes of operation and mode-changes are a useful abstraction to enable configurable, flexible real-time systems. Substantial work on the fixed priority preemptive scheduling approach allowed tasks across a mode-change to be provided with real-time guarantees. However, the proper configuration of critical parameters such as task offsets, despite initial work, remains a gap in research. Without a method that automates this design step, while assuring that the basic requirements are met, the full adoption of mode-changes in real-time systems remains limited to relatively simple systems with limited task sets. We propose a method to assign offsets to tasks across a mode-change, using a metaheuristic approach (genetic algorithms). This method allows the configuration and/or the minimization of the worst-case latency of a mode-change. The latency of a mode change is a critical parameter to be minimized, since during the mode change the system offers limited functionality due to the fact that the task set is still incomplete. We also provide a classification of mode changes according to applications' requirements. This classification was useful, once applied to a number of case studies, both to validate the configuration approach and to a greater extent to show that the method is flexible in that it can accommodate a wide variety of types of mode-changes.

Keywords: real-time systems; schedulability analysis of mode-change; genetic algorithms; processor scheduling.

Resumo

Modos de operação e mudanças de modo são uma abstração útil para permitir que sistemas de tempo real sejam flexíveis e configuráveis. Trabalhos prévios em escalonamento preemptivo com prioridades fixas permitem que as tarefas passem de um modo de operação para outro provendo garantias de tempo real. No entanto, a configuração adequada dos parâmetros críticos, tais como o *offset* de uma tarefa, apesar de trabalhos anteriores terem abordado este assunto, permanece uma lacuna a ser explorada. Sem um método que automatize esta etapa do processo, garantindo ao mesmo tempo que os requisitos básicos sejam atendidos, a adoção plena de mudanças de modo em sistemas de tempo real permanece limitada a sistemas relativamente simples, com um conjuntos de tarefas limitado. Propomos um método para atribuir *offsets* às tarefas em uma mudança modo, através de uma abordagem Metaheurística (algoritmos genéticos). Este método permite a configuração e/ou a minimização da latência de pior caso de uma mudança modo. A latência de uma mudança de modo é um parâmetro crítico para ser minimizado, uma vez que durante a mudança de modo o sistema oferece funcionalidade limitada, uma vez que o conjunto de tarefas está parcialmente em operação. Também elaboramos uma classificação das mudanças de modo de acordo com as necessidades das aplicações. Esta classificação, quando aplicada a uma série de estudos de casos, permitiu validar a abordagem de minimização/configuração, estender a classificação anteriormente existente e demonstrar que o método é flexível, já que pode acomodar uma ampla variedade de tipos de mudanças de modo.

Palavras-chave: sistemas de tempo real; análise de escalonabilidade de mudança de modo; algoritmos genéticos; escalonamento de processos.

Contents

I dedicate this work to my wife Vanessa, to my daughters Livia and Lara and to my son Flávio Neto. They were my source of motivation for the conclusion of this journey.

Acknowledgements

I would like to thank both my supervisor Dr. Paulo Martins and my co-supervisor Dr. Edson Ursini for providing me with the opportunity to develop my dissertation at the University of Campinas, and for their continuing support and interest in my work. I also would like to thank them for the many hours of fruitful discussion and for reviewing the work in this dissertation. I am grateful to my colleagues from the College of Technology for their encouragement. Finally, I want to thank my wife Vanessa for her patience and support during difficult times.

Declaration

I declare that the research described in this dissertation is original work which I undertook between February 2013 and January 2015.

“Pray as though everything depended on God. Work as though everything depended on you.”
(Saint Augustine)

List of Figures

List of Tables

List of Acronyms and Abbreviations

Latin Characters

B	Block Time
C	Computation Time
Co	Active Constraints Vector
D	Deadline
k	Constant value dependent on the application (which is arbitrarily set to 30% in this work)
L	Latency
O	Offset Y or Z
Ob	Objective Values Vector
P	Priority
P_N	Set of priority from the new-mode tasks
P_O	Set of priority from the old-mode tasks
Q	Largest number of invocations of task τ_i that may occur within the busy period
R	WCRT of a task τ_i
R_N	Set of WCRT from the new-mode tasks
R_O	Set of WCRT from the old-mode tasks
T	Period between releases of a task τ_i
U	Utilization of Processor
w	Window Time
We	Objective Weights Vector
x	Interval of time between the mode-change request (MCR) and the activation of task τ_i
Y	Offset measured from the mode-change request MCR
Z	Offset measured from the end of the last task instance in the old-mode
$\#Tins_{(N)}$	Number of new-mode tasks inserted within significant interval δ
$\#Trmv_{(O)}$	Number of old-mode tasks completed within significant interval δ
$\#\tau_{(N)}$	Number of new-mode tasks
$\#\tau_{(O)}$	Number of old-mode tasks

Greek Characters

$\boldsymbol{\tau}$	Task
$\boldsymbol{\delta}$	Interval from the mode-change request (significant interval)
$\boldsymbol{\alpha}$	Relation between new-tasks inserted and old-mode tasks removed during interval δ
$\boldsymbol{\rho}$	Tolerance factor

Subscript

$_A$	Aborted Tasks
$_C$	Changed Tasks
$_N$	New Mode Tasks
$_O$	Old Mode Tasks (Completed)
$_U$	Unchanged Tasks
$_W$	Wholly New Tasks

Abbreviations

MIN	Minimum
MAX	Maximum

Acronyms

AOF	All-Old Mode Tasks First
ANF	All-New Mode Tasks First
BMC	Balanced Mode-Change
DMS	Deadline Monotonic Scheduling
DPGA	Elitist Distance-Based Pareto Algorithm
EA	Evolotionary Algorithm
EDF	Earliest Deadline First
EP	Evolutionary Programming
ES	Evolution Strategy

GA	Genetic Algorithm
GAs	Genetic Algorithms
IPCP	Immediate Priority Ceiling Protocol
LD	Latency Definition
MCD	Mode-Change Deadline
MCR	Mode-Change Request
MNF	Mostly-New Mode Tasks First
MOF	Mostly-Old Mode Tasks First
MOGA	Multi-Objective Genetic Algorithms
NDS	Nondominated Solution
NSGA-II	Elitist Nondominated Sorting Genetic Algorithm
RSS	WCRT of a Task in Steady-State Mode
SPEA-2	Strength Pareto Evolutionary Algorithm
WCRT	Worst Case Response Time
WECT	Worst Case Execution Time

Glossary

Busy Period - *"A level-i busy period is defined as the maximum time for which a processor executes tasks of priority greater than or equal to the priority of task τ_i"* (LEHOCZKY, 1990).

Cross interference - Interference between tasks from distinct modes.

Immediate Priority Ceiling Protocol (IPCP) - *"With IPCP, a resource has a ceiling priority assigned to it, not lower than that of the highest priority task that may use it. A task using a resource, immediately inherits its ceiling priority, thus avoiding unbounded priority inversion, deadlocks and transitive blocking"* (REAL; CRESPO, 2001).

Latency Definition I (LD I) - *"A window starting with the arrival of the mode change request to the system and ending when the set of new tasks have completed their first execution and the set of old tasks have completed their last execution"* (PEDRO; BURNS, 1998).

Latency Definition II (LD II) - *"The latency of mode-change has been considered as the time interval between MCR and completion of the first activation off all new-mode tasks"* (REAL; CRESPO, 2004).

Promptness - *"There may be new-mode tasks whose execution must be completed before a determinate time after the mode change was requested. This requirement models the need for a prompt response, specially useful when switching to an emergency mode"* (REAL; CRESPO, 2001).

Mode Change - The mode change occurs when one entity issues a mode-change request command (MCR). Once the system receives the MCR, its state changes to *transient-mode*. During the mode change the old-mode tasks will either be aborted or completed. New-mode tasks will start their execution. At the end of the transition the system changes back to *steady state/mode.*

Offset - Offset is a value that delays the introduction of a task during the mode change. Offsets are assigned to new-mode tasks to reduce the interference between tasks during the mode change.

Steady State - The system is in steady state when it executes a fixed task set (without mode changes).

Transition of Modes - Same as mode change.

1 Introduction

The new generation of real-time systems is required to be multifunctional and dynamically adaptable to the environment where they are deployed. One way of achieving multifunctionality and adaptability is by organizing the design of the system around modes of operation. Each mode implements a certain, well-defined system behavior and the system transitions from one mode to another in response to changes in the surrounding environment. The new active mode is customized and configured to the new operational phase and can thus deliver more performance than a general, monolithic (i.e. single mode) implementation of system functionality. In real-time systems, a mode of operation is defined by its behavior and implemented by a task set (schedule) (PEDRO, 1999). Changes in mode of operation thus involve changes in the task set, by adding, replacing or removing tasks from the schedule. In order to implement modal (or flexible) real-time systems, the transitions from mode to mode have to be guaranteed by offline (i.e. static) schedulability analysis.

This work is developed within the context of fixed-priority preemptively scheduled uniprocessor real-time operating systems (TINDELL *et al.*, 1994). As it is implied, these systems consist of multiple tasks with fixed priorities scheduled preemptively by the underlying operating system kernel. It offers a method to minimize the latency of a mode change in these systems. As we shall see in chapter 2 in more detail, in our model any new task is introduced in the system during a mode change (or transition) with an offset (or delay) O after the start of the transition to the new mode of operation. In one hand, offsets that are too small may increase the CPU utilization of the system at the start of the transition to a point where the system is no longer schedulable and therefore miss deadlines. On the other hand, if offsets are assigned with large values, the latency of a mode-change may increase to the point where the mode-change itself is not longer viable (i.e. assuming that the mode change must be completed within a certain deadline). This is due to the fact that, during a mode change, while the system is self-configuring its tasks, the system may be only partially delivering some of its critical functions. Therefore, it is of utmost importance that the latency of the system is reduced to the lowest possible value.

In this work we wish to find a method, based on genetic algorithms (GA), to automatically assign proper offsets to tasks in a mode-change so that the latency of the transition is minimized while real-time guarantees are also preserved. Real-time guarantees mean that some or all the tasks have real-time deadlines that must be fulfilled at design time, otherwise the system is deemed not schedulable leading to some sort of undesired behavior. This problem is a multi-objective optimization one that has to deal with tradeoffs as it will be discussed later.

To our knowledge, there is currently no work in the literature that addresses such concerns. The work that is closest to ours is the work by Real and Crespo (2001), whereby the authors tackle the issue of minimizing offsets. However, our goal in this work is completely different in that we wish to minimize the worst-case latency of a mode-change [1] (and not offsets). Notice also that most work using genetic algorithms in real-time systems fall outside the scope of this paper, as they deal with the issue of allocating tasks to multiprocessors, such as the work by Yoo (2009) and ManChon *et al.* (2011).

A secondary contribution of this work is a method combined with a software tool that allows the configuration of mode-changes in fixed-priority preemptively scheduled real-time systems. This method extends the mode-change classification proposed by Real and Crespo (2004) by introducing five new classes of mode changes according to the relative ordering of tasks in a mode change. Therefore, it is possible for the designer to specify the type of transition desired (which is application dependent feature), have it implemented and then minimized before deployment of the actual application.

In this work, was adopt the asynchronous mode-change model, where tasks from the new-mode begin execution in parallel with tasks from the old-mode, thus leading to shorter mode-change delays. Old-mode completed tasks are actually discarded from the system, leaving resources for tasks arriving from the new-mode. Therefore, the number of modes of operation is only limited by the available memory in the system. This approach allows systems to be schedulable, with a large number of modes and tasks. It also enables faster mode-changes due to the (pseudo) parallelism of old and new-mode tasks, and the early introduction of new-mode tasks at the beginning of a mode-change. This approach requires schedulability analysis, which is provided by existing work (REAL; CRESPO, 2001; PEDRO; BURNS, 1998).

1.1 Goals

In a broad sense, the goal of this work was to propose a method that allows the configuration of a mode-change based on an *evolutionary algorithm* in a way that meets the requirements of such transitions. These requirements include minimized mode-change latency, real-time guarantees, and the ability to express and configure various types of mode-changes as discussed later in this work. More specifically, in this work we wish to find a method

[1] whenever we refer to latency, we mean the worst-case latency

to assign proper offsets to tasks in a mode-change so that the latency of the transition is minimized while real-time guarantees are also preserved.

1.2 Dissertation Organization

The remainder of this dissertation is organized as follows:

Chapter 2 surveys the field of fixed priority preemptive scheduling, schedulability analysis of mode-change in fixed priority systems, the offset minimization algorithm, the evolutionary algorithms with an emphasis in single and multi-objective genetic algorithms as well as related work.

Chapter 3 introduces the approach chosen for minimization of mode-change latency using genetic algorithms. In addition, was presented a number of case studies to prove the efficacy of this method.

Chapter 4 focus on exploring a number of case studies that evidentiate the flexibility (or versatility) of the approach. Furthermore, a method for classification of mode changes is introduced in order to facilitate the proper configuration of mode-changes.

Finally, chapter 5 summarizes the main conclusions and proposals for future work.

2 Background and Literature Review

This work integrates topics from two distinct areas: Schedulability Analysis (real-time systems) and Evolutionary Algorithms (artificial intelligence). Therefore, in this chapter we present background and literature review related to this work such as schedulability analysis, schedulability analysis for mode change, offsets optimization in real-time systems and genetic algorithms.

2.1 Schedulability Analysis

The basic premise of a real-time system is that it provides a guarantee that it meets all timing requirements for a given configuration during its life-cycle.

One way to achieve real-time guarantees is by applying the schedulability analysis. The schedulability analysis uses the WCRT (Worst-Case Response Time) to determine if a task set meet its requirements. We calculate the worst-case response time (R) for each task in the system and compare it with its respective deadline (D). The system is deemed to be feasible if all tasks meet their deadlines (i.e. $R \leq D$). Otherwise, it is necessary to reconfigure the system and redo the analysis.

Throughout this chapter we will provide schedulability analysis that has been used to calculate the worst-case latency of a task across a mode change. For more details on the basic fundamentals of real-time schedulability analysis, the reader is referred to Burns and Wellings (2009).

2.2 Computational Model and Assumptions

We shall consider a set of periodic or sporadic tasks $\tau = \{\tau_1, \tau_2, \ldots \tau_i, \ldots \tau_p\}$ per mode. Each task τ_i is characterized by the tuple $S_i = \{T_i, D_i, C_i, P_i\}$, where: 1) T_i and D_i are respectively the period of task τ_i (or, if a sporadic task, the minimum inter-arrival time between successive tasks of the stream i) and the deadline; 2) C_i is the worst-case execution time (WCET) of the task τ_i. This value is deemed to contain the overheads due to context switching. Moreover, the values of C_i, D_i and T_i are such that $C_i < D_i \leq T_i$. We remove the restriction that$D_i \leq T_i$; 3) P_i represents the priority of task τ_i , assigned according to the Deadline Monotonic Scheduling algorithm.

Throughout this chapter, we use the notation $C_{i(O)}$,$C_{i(A)}$ and $C_{i(N)}$ when referring to the computational time of an old-mode completed task, an aborted task, and a new-mode task, respectively. τ_i denotes a task for which we are finding the WCRT and τ_j denotes a higher-priority task. We use the term *steady-state analysis* to refer to the body of schedulability analysis of single-mode systems, where the task set is fixed and there are no mode changes. We also use the notation:

- $\forall\ \tau_{j(O)}\ hp\ \tau_i$: set of old-mode tasks τ_j with priority higher than task τ_i;
- $\forall\ \tau_{j(A)}\ hp\ \tau_i$: set of aborted tasks τ_j with priority higher than task τ_i;
- $\forall\ \tau_{j(N)}\ hp\ \tau_i$: set of new-mode tasks τ_j with priority higher than task τ_i.

The mode-change model is based on the following assumptions:

- Tasks are executed in a uniprocessor system;
- Tasks are not permitted to voluntarily suspend themselves during an invocation (so, for example, tasks are not allowed to execute internal Ada-like delay statements);
- There are fixed task sets before and after the mode change;
- The worst-case response time of a generic task τ_i (WCRT), denoted R_i, is the longest time ever taken by that τ_i from the time it arrives until the time it completes its required computation;
- Tasks are scheduled with time offsets during the mode change only. This time phasing between tasks may or may not hold after the mode change.

Furthermore, we assume that there is to some extent a certain leeway in defining the tasks' real-time parameters such as the period, worst-case execution time and priorities. For example, the worst-case execution time depends at least on the processor speed, and on the code that implements the desired behavior for a task. A task may be divided in sub-tasks in case the value of C is unacceptably large. Similarly, the value of T may lie within a range: in one hand, the largest value (or upper bound) is dictated by the physical process variable monitored by the task, which must be scanned using a minimum sampling rate; the lower bound on T is the processor utilization, as a small periods substantially increase processor utilization. Within these limits, the application designer has some flexibility in choosing an appropriate value of T. Priorities can also be adjusted as long as the deadline monotonic policy (DMS) is ensured. This assumption is not unrealistic for most types of real world systems and it plays an important role in chapters 3 and 4 where we present our approach to latency minimization.

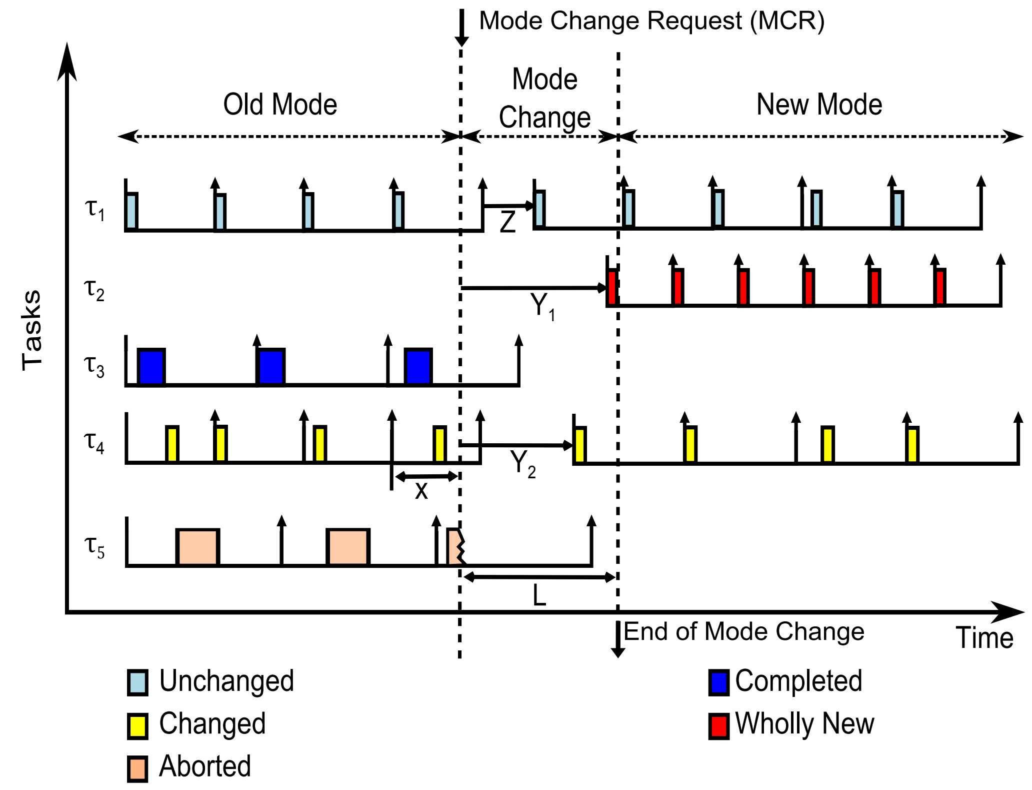


Figure 1 – Mode-Change Model.

A *mode-change request (MCR)* is the event that triggers a transition from an old-mode of operation to a new one. The window x is the phasing between the MCR and the activation of task τ_i. A MCR may not be preempted by another MCR. The mode-change model comprises of five types of tasks (Fig. 1):

- *Old-mode completed tasks*, $\tau_{i(O)}$: These tasks are released in the old-mode, i.e. before the arrival of the MCR. These tasks need to advance their execution in the transition window to finish execution. They cannot be simply aborted as they would leave the system in an inconsistent state. Once they complete during the transition, there are no further releases. They are used to model the behavior of the system in the old-mode that is no longer needed in the new-mode.

- *Old-mode aborted tasks*, $\tau_{i(A)}$: These tasks are also released prior to the MCR. They need to be immediately discarded after the MCR in order to release allocated resources back to the system. The functionality they implement is no longer needed in the new-mode of operation.

- *New-mode changed tasks*, $\tau_{i(C)}$: These tasks are released during the transition, with an offset Y from the MCR. This class models the behavior that is changed in the new-mode. Changed new-mode tasks have a modified timing parameter compared to their corresponding old-mode version, such as changed worst-case execution time (C), period (T), or priority (P).

- *New-mode unchanged tasks*, $\tau_{i(U)}$: These tasks are released during the transition window, with an offset Z, from the end of the period of their corresponding old-mode version. They model the behavior of the application that is not changed across the mode change and in the new-mode. Their timing parameters are the same as the preceding old-mode version.

- *Wholly new task*, $\tau_{i(W)}$: These tasks are released during the transition window with an offset Y. They are used to model the behavior that is totally new, i.e. has no equivalent in the old-mode of operation.

With respect to the way tasks are executed across a mode change, they are classified as: 1)Tasks with mode-change periodicity: these tasks are executed across the mode change and maintain their activation pace, and 2) Tasks without mode-change periodicity: These tasks do not preserve their activation pace across a mode change. The schedulability analysis of mode change is divided into two parts: analysis of the old-mode tasks and analysis of the new-mode tasks.

2.2.1 Analysis for Old-mode Tasks

The interference level of old-mode tasks is given in accordance with its classification of the types of tasks:

- *Interference from higher priority old-mode completed tasks $I_{hp(i)_O}$:* it is necessary to take into account the interference of higher priority old-mode tasks released in the interval x exclusively. Therefore, we have:

$$I_{hp(i)_O} = \sum_{\forall j \in hp(i)_O} \left\lceil \frac{x}{T_j} \right\rceil C_j \tag{2.1}$$

- *Interference from higher priority aborted tasks $I_{hp(i)_A}$:* while it is clearly not necessary to guarantee the schedulability of this class of tasks, we still need to consider their interference upon lower priority old-mode tasks. There are two components to this term: first we should consider the number of complete periods of the higher priority aborted task j that fit in in the interval x, which is given by $\lfloor x/T \rfloor$. For any given higher priority aborted task j, there are a number of complete executions in the interval by $\lceil x/T \rceil$. Also, we must consider for each higher priority aborted task the amount of interference in the remaining time before the start of the mode change. The remaining time can be great enough to either contain another complete execution of task j or great enough to fit only a partial execution of task j. This can be calculated by the following expression:

$$I_{hp(i)_A} = \sum_{\forall j \in hp(i)_A} \left(\left\lfloor \frac{x}{T_j} \right\rfloor C_j + \min\left(x - \left\lfloor \frac{x}{T_j} \right\rfloor T_j, C_j \right) \right) \tag{2.2}$$

- *Interference from higher priority new-mode tasks $I_{hp(i)_N}$:* new tasks (τ_N) have their first release at a time $(x + Y)$ after the start of the window w. Therefore, higher priority new tasks can preempt the execution of the old-mode task i from the remaining time in the window w, causing an interference that can be represented by:

$$I_{hp(i)_N} = \sum_{\forall j \in hp(i)_N} \left\lceil \frac{w_i - x - Y_j}{T_j} \right\rceil_0 C_j \tag{2.3}$$

- *Interference of unchanged tasks $hp(i)_U$:* tasks from the old-mode that remain unchanged in new-mode, using the same temporal behavior (period and deadline). Therefore, we have:

$$I_{hp(i)_U} = \sum_{\forall j \in hp(i)_U} \left\lceil \frac{x}{T_j} \right\rceil C_j + \left\lceil \frac{w_i - \lceil x/T_j \rceil T_j - Z_j}{T_j} \right\rceil_0 C_j \tag{2.4}$$

In summary, combining the analysis of interference of each task type we obtain the analysis model of the old-mode, given by the following equation:

$$\begin{gathered} w_i = C_i + B_i + \sum_{\forall j \in hp(i)_O} \left\lceil \frac{x}{T_j} \right\rceil C_j + \\ \sum_{\forall j \in hp(i)_A} \left(\left\lfloor \frac{x}{T_j} \right\rfloor C_j + \min\left(x - \left\lfloor \frac{x}{T_j} \right\rfloor T_j, C_j \right) \right) + \\ \sum_{\forall j \in hp(i)_N} \left\lceil \frac{w_i - x - Y_j +}{T_j} \right\rceil_0 C_j + \\ \sum_{\forall j \in hp(i)_U} \left\lceil \frac{x}{T_j} \right\rceil C_j + \left\lceil \frac{w_i - \lceil x/T_j \rceil T_j - Z_j}{T_j} \right\rceil_0 C_j \end{gathered} \tag{2.5}$$

The notation $\lceil z \rceil_0$ denotes a modified ceiling function that returns zero if $Z < 0$. Because w_i exists on both sides of equation (2.5), it is necessary to adapt the equation to predict the recurrence relation as follows:

$$\begin{gathered}
w_i^{n+1} = C_i + B_i + \sum_{\forall j \in hp(i)_O} \left\lceil \frac{x}{T_j} \right\rceil C_j + \\
\sum_{\forall j \in hp(i)_A} \left(\left\lfloor \frac{x}{T_j} \right\rfloor C_j + \min\left(x - \left\lfloor \frac{x}{T_j} \right\rfloor T_j, C_j \right) \right) + \\
\sum_{\forall j \in hp(i)_N} \left\lceil \frac{w_i^n - x - Y_j}{T_j} \right\rceil_0 C_j + \\
\sum_{\forall j \in hp(i)_U} \left\lceil \frac{x}{T_j} \right\rceil C_j + \left\lceil \frac{w_i^n - \lceil x/T_j \rceil T_j - Z_j}{T_j} \right\rceil_0 C_j
\end{gathered} \tag{2.6}$$

The initial value of w_i is set to zero. It can be shown that $w_i^{n+1} > w_i^n$, and hence the equation is guaranteed either to converge (i.e. $w_i^{n+1} = w_i^n$) or to exceed some threshold, such as D_i. However, the worst case for the response task Ri, which must then be compared with the respective deadline is given by $R_i = w_i + C_i$.

Therefore, the analysis should consider that the deadlines of the tasks may be greater than their respective periods, so it is necessary to identify which is the largest number of invocations of task i that may occur within the busy period, represented by Q_i. The Q_i for the task i is given by:

$$Q_i = \left\lceil \frac{t_i}{T_i} \right\rceil \tag{2.7}$$

where T_i is the period of task i and w_i, represents a recurrence relation, with the first invocation, $t_i^0 = C_i$ and last, $t_i^{n+1} = t_i^n$, with:

$$t_i^{n+1} = B_i + \sum_{\forall j \in hp(i) \cup i} \left\lceil \frac{t_i^n}{T_j} \right\rceil C_j \tag{2.8}$$

when $hp(i) \cup i$ is the set of tasks with priority equal or higher that the task i.

In summary, the worst window value w for i task using arbitrary deadlines is given by equation:

$$\begin{gathered}
w_i^{n+1}(q) = (q+1)C_i + B_i + \sum_{\forall j \in hp(i)_O} \left\lceil \frac{x}{T_j} \right\rceil C_j + \\
\sum_{\forall j \in hp(i)_A} \left(\left\lfloor \frac{x}{T_j} \right\rfloor C_j + \min\left(x - \left\lfloor \frac{x}{T_j} \right\rfloor T_j, C_j \right) \right) + \\
\sum_{\forall j \in hp(i)_N} \left\lceil \frac{w_i^n - x - Y_j}{T_j} \right\rceil_0 C_j + \\
\sum_{\forall j \in hp(i)_U} \left\lceil \frac{x}{T_j} \right\rceil C_j + \left\lceil \frac{w_i^n - \lceil x/T_j \rceil T_j - Z_j}{T_j} \right\rceil_0 C_j
\end{gathered} \tag{2.9}$$

Thus, the worst-case response time R of task i using arbitrary deadlines is given by:

$$R_i = \max_{q=0\ldots Q_i-1}(w_i(q) - qT_i + C_i) \tag{2.10}$$

2.2.2 Analysis for New-mode Tasks

Because new-mode tasks suffer interference from other higher priority new and other higher priority old-mode tasks, we need to guarantee their schedulability during the mode change. If, however, a new task i has an offset such that its first release occurs after all higher priority old-mode tasks have completed, its schedulability is guaranteed by steady-state analysis and we do not need to apply the following analysis to obtain its WCRT.

- *Interference from higher priority old-mode tasks $hp(i)_O$:* old-mode tasks may progress through the mode change until completion. In the worst-case scenario, all old tasks are released momentarily before the mode change, thus sharing a critical instant with the window w_i. Therefore, we have:

$$I_{hp(i)_O} = \sum_{\forall j \in hep(i)_O} C_j \tag{2.11}$$

- *Interference from higher priority new-mode tasks $I_{hp(i)_N}$:* the interference caused by all new higher priority tasks j (released at Y_j) upon a new-mode task i. This can be calculated by the following expression:

$$I_{hp(i)_N} = \sum_{\forall j \in hp(i)_N} \left\lceil \frac{w_i - Y_j}{T_j} \right\rceil_0 C_j \tag{2.12}$$

- *Interference of unchanged tasks $hp(i)_U$:* tasks from the old-mode that remain unchanged in new-mode, using the same temporal behavior (period and deadline). Therefore, we have:

$$I_{hp(i)_U} = \sum_{\forall j \in hp(i)_U} \left(C_j + \left\lceil \frac{w_i - T_j - Z_j}{T_j} \right\rceil_0 C_j \right) \tag{2.13}$$

The worst-case response time of a new task i across a mode change is therefore given by:

$$\begin{gathered} w_i = B_i + \sum_{\forall j \in hep(i)_O} C_j + \\ \sum_{\forall j \in hp(i)_N} \left\lceil \frac{w_i - Y_j}{T_j} \right\rceil_0 C_j + \\ \sum_{\forall j \in hp(i)_U} \left(C_j + \left\lceil \frac{w_i - T_j - Z_j}{T_j} \right\rceil_0 C_j \right) \end{gathered} \tag{2.14}$$

Because w_i exists on both sides of equation (2.14), it is necessary to adapt the equation to predict the recurrence relation as follows:

$$\begin{aligned} w_i^{n+1} = B_i + \sum_{\forall j \in hep(i)_O} C_j + \\ \sum_{\forall j \in hp(i)_N} \left\lceil \frac{w_i^n - Y_j}{T_j} \right\rceil_0 C_j + \\ \sum_{\forall j \in hp(i)_U} \left(C_j + \left\lceil \frac{w_i^n - T_j - Z_j}{T_j} \right\rceil_0 C_j \right) \end{aligned} \tag{2.15}$$

The initial value of w_i is set to zero. It can be shown that $w_i^{n+1} > w_i^n$, and hence the equation is guaranteed either to converge (i.e. $w_i^{n+1} = w_i^n$) or to exceed some threshold, such as D_i. However, the worst case for the response task Ri, which must then be compared with the respective deadline is given by $R_i = w_i$ - Y_i.

However, it should also be considered in the analysis of new tasks so that the deadlines of tasks may be greater than the respective periods (arbitrary deadlines), so the derived equation corresponds to:

$$\begin{aligned} w_i^{n+1}(q) = B_i + (q+1)C_{i(N)} + \sum_{\forall j \in hep(i)_O} C_j + \\ \sum_{\forall j \in hp(i)_N} \left\lceil \frac{w_i^n - Y_j}{T_j} \right\rceil_0 C_j + \\ \sum_{\forall j \in hp(i)_U} \left(C_j + \left\lceil \frac{w_i^n - T_j - Z_j}{T_j} \right\rceil_0 C_j \right) \end{aligned} \tag{2.16}$$

whereas the task i was released with an offset O_i, corresponding to Y offset, when the task belongs only to the old-mode and Z offset, when the task the old-mode continues its execution in the new-mode, to determine the worst response time of task R_i must be subtracted from the w_i window corresponding to the assigned offset value, so:

$$R_i(q) = w_i(q) - qT_i - O_i \tag{2.17}$$

2.3 Schedulability Analysis Algorithm

Fig. 2 shows an UML activity diagram modeling the flow of the algorithm that calculates the schedulability analysis using arbitrary deadlines. It consists of the following steps:

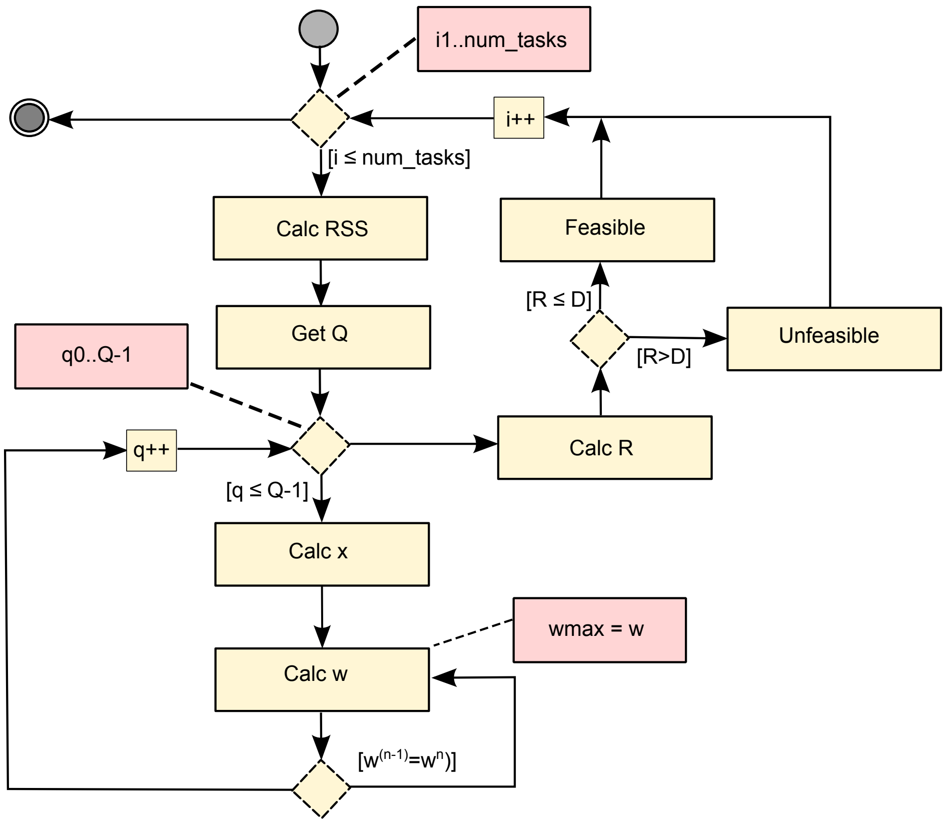


Figure 2 – Schedulability Analysis UML Diagram of Mode Change

- *CALC RSS:* The worst-case steady-state response time for each old-mode task τ_i (RSS) is calculated. This value is used to place a bound on the maximum value of x for each task;

- *Get Q:* In this step it is calculated the value of Q, i.e. the largest number of invocations of task τ_i that may occur within the busy-period (analysis with arbitrary deadline). This is the outermost loop in the analysis. It is repeated for values of q = 0,1,2,3 . . . and until $q \leq Q$.;

- *Calc x:* For each value of Q, and for each old-mode task, it is calculated the value of x, i.e. the arrival time of the old-mode task before the start of the transition (MCR). This is the second outermost loop in the analysis;

- *Calc w:* For each task in the system, the worst-case response time window w_i is calculated using a recurrence relation. This relation converges when $w^n = w^{n-1}$. This is the innermost loop in the system.

- *Calc R:* Once the previous loops are ended, the system then is ready to calculate the worst-case response time R_{mc} of a task across the mode change. This value takes into account the worst values of Q, x and w found in the previous steps. In case $R \leq D$ for all tasks, the system is deemed feasible, and otherwise unfeasible (or unschedulable).

2.4 Definition of Mode-Change Latency (L)

The mode-change latency is usually an important performance criteria when dealing with mode changes. We often seek to minimize the latency since during the mode change the system may deliver only partial functionality at the expenses of more critical services. The mode change latency is defined by Pedro and Burns (1998) and Real and Crespo (2004) as follows:

- *Definition I: "A window starting with the arrival of the mode-change request (MCR) and ending when the set of new-mode tasks have completed their first execution and the set of old-mode tasks have completed their last execution"* (PEDRO; BURNS, 1998). The worst-case latency is given by equation (2.18):

$$L = \max(R_{i(N)} + O_{i(N)}, R_{i(O)} - x_i) \ \forall \ i \ \epsilon \ \tau \tag{2.18}$$

  where $R_{i(N)}$ is the worst-case response time of new-mode task τ_i across the mode change, and $R_{i(O)}$ is the worst-case response time of the old-mode task τ_i in the mode change. This definition of latency is the default definition adopted for this work.

- *Definition II: "The latency of mode change has been considered as the time interval between MCR and completion of the first activation off all new-mode tasks"* (REAL; CRESPO, 2004). This definition is given by equation (2.19):

$$L = \max(R_{i(N)} + O_{i(N)}) \ \forall \ i \ \epsilon \ \tau \tag{2.19}$$

  where $R_{i(N)}$ is the worst-case response time of new-mode task τ_i across the mode change, $O_{i(N)}$ is the offset Y or Z assigned to new task τ_i.

2.5 Offset Minimization Algorithm

An algorithm for the optimization tasks' offsets across a mode change was proposed by Real and Crespo (2001). This algorithm reduces offsets for high-priority tasks and considers two scenarios in order to achieve consistency in the use of shared resources across an asynchronous mode change: increased blocking of new-mode tasks and violation of the IPCP (Immediate Priority Ceiling Protocol). The IPCP protocol ensures that lower-priority old-mode tasks sharing resources do not lock resources used by new-mode, higher-priority tasks (REAL; WELLINGS, 1999). Fig. 3 shows the algorithm used in Real and Crespo (2001).

There are three vectors used in this approach:

- Y_r is a vector with the values of the response time of each $\tau_{Li(O)}$ (set of lowest priority old-mode tasks that may use the conflicting resource), initialized with 0.
- Y_{min} is a vector holding the values of the minimum offsets at any point in the analysis, which has all its values initialized with 0.
- Y_{max} is a vector with the maximum offsets that can be used by new-mode tasks, initialized with an arbitrarily large offset for each task.
- Y_i is a vector that holds the values of the current repetition of the analysis, which may or not be minimum values. This vector is initialized with large values (step 5).

```
 1. $Y_r$ := (0,0,... 0)
 2. $Y_{max}$ := (∞,∞, ...∞)
 3. $Y_{min}$ := (0,0,... 0)
 4. **loop**
 5.   ∀$_i$, $Y_i$ := $Y_{i,max}$
 6.   **if** Feasible then
 7.      Reduce-Offsets
 8.      $Y'_r$ := ($R_{L1}$,$R_{L2}$,..., $R_{Ln}$)
 9.      **exit when** $Y'_r$ = $Y_r$
10.     $Y_r$ =$Y'_r$
11.     $Y_{min}$ =$Y_r$
12.   **else**
13.      -- Not feasible
14.   **endif**
15. **end loop**
```

Figure 3 – Algorithm to Calculate Offsets (REAL; CRESPO, 2001).

The offset of the higher priority task is reduced to a point where the system is closest to becoming unschedulable or it is zero. This procedure is repeated to the next lower priority task until all tasks in the task set are covered. All final offsets must be within a limited range of values $[Y_{i,min} \ldots Y_{i,max}]$.

In addition to the schedulability analysis work of (PEDRO; BURNS, 1998; REAL; CRESPO, 2004; TINDELL *et al.*, 1992) on mode changes in uniprocessor, fixed-priority preemptive real-time systems using the deadline monotonic policy, other authors have contributed to the theme employing other models and/or approaches, such as mode changes in 1) systems with the rate-monotonic policy (SHA *et al.*, 1988), 2) time-triggered real-time systems (FOHLER, 1993), 3) multiprocessor systems' scheduling (NÉLIS; GOOSSENS, 2008; YOMSI *et al.*, 2010), 4) uniprocessor using Earliest Deadline First (EDF) scheduling (ANDERSSON, 2008; STOIMENOV *et al.*, 2009), (NELIS *et al.*, 2011), and 5) multiprocessor systems with mixed-criticality constraints (NEUKIRCHNER *et al.*, 2013; NIZ; PHAN, 2014).

2.6 Evolutionary Algorithms (EA)

Evolutionary algorithms are based on *"Biologically Inspired Computing"* (BONGARD, 2009) and *"Natural Computing"* (CASTRO, 2006), i.e, inspired on biological processes and natural evolution to solve optimization problems. The genetic algorithm was introduced by Holland (1975) with the goal of applying concepts related to the law of evolution to find optimization solutions, assuming that the process of natural evolution could be adapted for application in the search for complex optimization solutions, due to its robustness and simplicity.

Although the exploration of the evolutionary algorithm approach started in the 30s, it was only in the 60s that its application was expanded. It was motivated by an increase of the availability of low-cost computers (JONG, 2006; RECHENBERG, 1965; FOGEL *et al.*, 1966).

"The initial specification and analysis of these simple evolutionary algorithms (EAs) in the 1960s left two major issues unresolved: 1) characterizing the behavior of implementable systems, and 2) better understanding how these system could be used to solve problems" (JONG, 2006). In the 70s, researchers were focused on finding answers to these questions by conceiving methods to apply the existing theory, which resulted in three types of EAs: Evolutionary Programming (EP), Evolution Strategies (ES) and genetic algorithms.

2.6.1 Genetic Algorithm (GA)

The genetic algorithm (GA) was introduced by Holland (1975), with the purpose of applying the concept related to the law of species evolution to find optimization solutions.

His approach was that the process of natural evolution could be adapted for application in the search of complex solutions of optimization due to its robustness and simplicity.

According to Jong (2006), different from EP and ES, the GAs allow the application of the algorithm in an independent form from the application. For that, they used a universal genetic representation for the individuals that simplify its utilization. The mutation operators are applied from a fixed probability, and the recombination selects the genetic sequence of a pair of progenitor's individuals randomly, generating new individuals that inherit genetic characteristics of both.

According to Michalewicz (1996), traditionally, the GAs uses the binary notation for the individual's representation. Although, in some cases it can cause a delay in the search of solutions with the degree of accuracy required. Thus, he conducted studies that demonstrate that the individuals representation from the numerical notation, using float point, allow to reduce the exploratory space, enabling to find solutions in a more effective way.

The taxonomy reported in Corte *et al.* (2012) divides metaheuristics frameworks into three classes:

1. Local search metaheuristics, which operate on a single complete solution and iteratively improve it by making small adjustments called moves;

2. Population-based metaheuristics, which operate on a set of solutions and find better solutions by combining solutions from that set into new ones;

3. Constructive metaheuristics, which build a solution by working with a single, unfinished, solution and adding one solution element at a time.

We found that methods belonging to the Local search metaheuristics risk to be trapped in unfeasible spaces, while methods belonging to constructive metaheuristics are promising but quite difficult to be calibrated and require further research.

GA methods belong to the population based metaheuristics are stable and produce a sure improvement without the need to calibrate parameters that do not have their own physical meaning, and which therefore must be estimated empirically.

2.6.2 Basic Terminology

The basic terminology presented was adapted based on Coello *et al.* (2007). Fig. 2.6.2 represents the key components used by evolutionary algorithms.

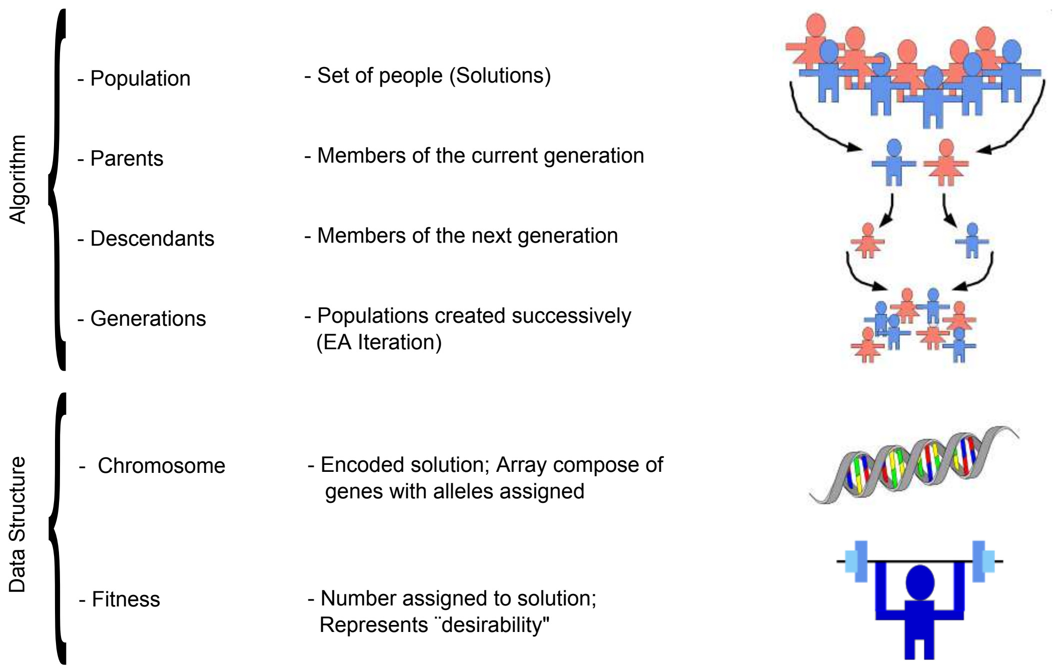

Algorithm
- Population
- Set of people (Solutions)
- Parents
- Members of the current generation
- Descendants
- Members of the next generation
- Generations
- Populations created successively (EA Iteration)
Data Structure
- Chromosome
- Encoded solution; Array compose of genes with alleles assigned
- Fitness
- Number assigned to solution; Represents ¨desirability"

Figure 4 – Key Components of EA. Adapted from (COELLO *et al.*, 2007)

Chromosomes, Genes and Alleles

In biology, a *chromosome* contains the genetic code that determines how the organisms are composed. The number of chromosomes can vary from one species to another. The set of chromosomes that compose an individual is called *genotype*. The genes are a set of symbols that represent the codification of a chromosome. The position of a gene inside a chromosome is called *locus*. The values assigned to a gene are called *alleles*. As an analogy, the values assigned to a chromosome are a candidate solution to a given optimization problem.

Fitness Function

The *fitness* function allows the calculation of the level of fitness of an individual inside a population.

Selection, Recombination and Mutation

The selection process selects the chromosomes with higher fitness for reproduction, i.e., the larger the fitness of a chromosome, the larger its probability of being chosen for reproduction. After the selection process, pairs of selected chromosomes are recombined using a crossover operator. Therefore, a new individual is created inheriting genetic characteristics from both parents.

Another genetic operator, the *mutation operator*, is used to increase the diversity of chromosomes in a population. The mutation operator randomly changes the data of a chromosome, following a mutation probability previously determined.

2.6.3 Multi-Objective Genetic Algorithms (MOGA)

At first, GA's were used in optimization problems involving only a single objective function. Multi-objective problems were solved by assigning different weights to each objective (weights-based multi-objective GA), thus converging to a single fitness. However, it is a simplistic approach which cannot achieve the requirements of real applications. This perception led to research towards another approach: optimization of competing or conflicting objectives, giving rise to the development of Multi-Objective Genetic Algorithms. The basic concepts on multi-objective optimization are presented in the following paragraphs.

Multi-Objective Problems

According to the Vilfredo Pareto's (1848-1923) concept, multi-objective problems have a set of optimal solutions: non-dominated (or Pareto optimal), and Pareto efficient (or non-inferior) (PARETO, 1896). However, this set of solutions raises another question: *"Which solution to choose?"*. This question can be answered using the multi-criteria analysis. Its main goal is to assist the decision-makers in articulating their preferences in the presence of ambiguities and uncertainties. Thus, it makes their decision more consistent with their interests (COELLO, 2000).

In real-world problems, in addition to objective optimization, it is normally necessary to meet application constraints. These constraints may be modeled using an inequality function such as $g_i(\vec{x}) \geq 0$. Therefore, $\vec{x} = (x, \ldots, x_k)$ is the vector of optimization variables. A multi-objective problem (MOP) is given by:

$$\begin{aligned} min\vec{F}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), \ldots, f_m(\vec{x})), \\ subject\ to\ g_i(\vec{x}) \geq 0, i = 1, 2, \ldots, l, \\ x_i \in [xmin_i, xmax_i], i = 1, 2, \ldots, k \end{aligned} \tag{2.20}$$

It is denominated objective space the coordinated space where are represented the obtained arrays from the objective function evaluation it is called the space of optimization variables, the coordinate space in which the axes represent each decision variable. The objective functions make, therefore, the mapping between each point $\vec{F}(\vec{x}) = (f_1, f_2, \ldots, f_m)$ in the objective space.

Pareto-Optimal Concept

The nonexistence of a global optimum that meet at the same time all conflict objectives, as it happens in mono-objective optimization, is expected to have a set of solutions for the multi-objective problems. This set can be defined using the concepts of the dominance relation and Pareto-optimal to the solutions that meet n constraints simultaneously. These concepts are shown below, assuming the same minimization problem.

Dominance Relation

In an optimization problem of minimization, using two arrays $\vec{y}_A$ e $\vec{y}_B \in \Re^m$ have been: $\vec{y}_A \prec \vec{y}_B$ if, at least one dimension j, $\vec{y}_A$ is less than $\vec{y}_B$ and in the other dimensions $i \neq j$, $\vec{y}_A$ is less than or equal $\vec{y}_B$. In this case, array $\vec{y}_B$ is dominated by $\vec{y}_A$. Applying this definition for the multi-objective optimization problem suggested, given two points $\vec{x}_A$ e $\vec{x}_B$, where $\vec{y}_A = (f_1(\vec{x}_A), f_2(\vec{x}_A), \ldots, f_m(\vec{x}_A))$ and $\vec{y}_B = (f_1(\vec{x}_B), f_2(\vec{x}_B), \ldots, f_m(\vec{x}_B))$, we have (VELDHUIZEN; LAMONT, 2000):

$$\begin{aligned} \vec{y}_A \prec \vec{y}_B \leftrightarrow \forall i \in \{1, 2, \ldots, m\}, f_1(\vec{x}_A) \leq f_1(\vec{x}_B) \wedge \\ \exists j \in \{1, 2, \ldots, m\}, f_j(\vec{x}_A) < f_j(\vec{x}_B) \end{aligned} \tag{2.21}$$

The dominance relation concept allows to introduce the condition for optimal solutions for multi-objective problems. Determining the feasible region Ω, each possible solution $\vec{x}_A \in \Omega$ is called Pareto-Optimal Solution if it doesn't exist another point $\vec{x}_B \in \Omega$ such as, $\vec{y}_B = (f_1(\vec{x}_B), f_2(\vec{x}_B), \ldots, f_m(\vec{x}_B))$ dominates $\vec{y}_A = (f_1(\vec{x}_A), f_2(\vec{x}_A), \ldots, f_m(\vec{x}_A))$.

Pareto-Optimal Set

The optimality concept for a multi-objective problem, define a set of efficient solutions, given by (VELDHUIZEN; LAMONT, 2000):

$P* := \{\vec{x}_A \in \Omega | \neg \exists \vec{x}_B \in \Omega : \vec{F}(\vec{x}_B) \prec \vec{F}(\vec{x}_A)\}$.

These set of Pareto-optimal solutions is the set of evaluation arrays of the objective functions that meet the space of front, i.e, a Pareto-Optimal front $PF*$ or simply Pareto -Front, given by:

$PF* := \{\vec{F}(\vec{x}) | \forall \vec{x} \in P*\}$

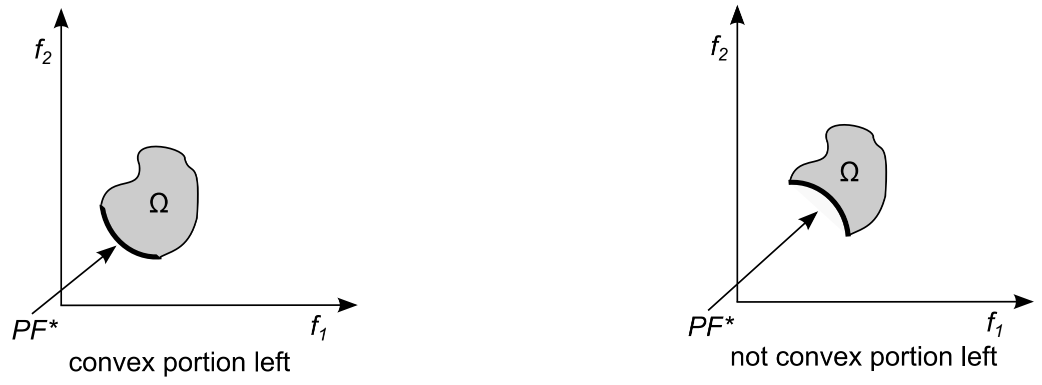


Figure 5 – Example of Pareto-Front.

Note: The Pareto front may delimit a convex or not convex portion of the feasible region (Fig.2.6.3).

Selection in Multi-Objective Genetic Algorithms (MOGA)

The selection criteria for MOGA has as a goal to guide the GA for the most viable regions where are found the Pareto front. Thus, beyond preserving the efficient solution and discarding the dominated ones it's necessary to guarantee a good distribution over the front. In the literature there are many MOGA's, which differences are the way the selection operator promotes the non-dominant solutions and the solution that are found in less populated areas (DEB, 2011).

Nowadays the most important MOGA are: Elitist Non-dominated Sorting Genetic Algorithm (NSGA-II), Strength Pareto Evolutionary Algorithm (SPEA-2) and Elitist Distance-Based Pareto Algorithm (DPGA)(DEB *et al.*, 2000; KIM *et al.*, ; DEB, 2011).

Elitist Non-dominated Sorting Genetic Algorithm (NSGA-II)

The NSGA-II is a multi-objective genetic algorithm developed by Deb *et al.* (2000) that uses the optimization concept presented in the preceding section. It is currently widely used to find solutions for multi-objective optimization problems. Its main characteristics are:

- Combined population;
- Fast selection operator for sorting by dominance;
- Use of two ordering algorithms:
  - *Non-dominated Sorting Algorithm:* Finds solution near the Pareto-Front;

– *Crowding-Distance Sorting:* Finds well-distributed solutions in space.

Thus, the selection operator for NSGA-II assigns for each individual two indexes. The first index, represented by nf, determines the front in which each individual belongs. To achieve this, it is necessary to classify the population in non-dominated front, where the first front should correspond to non-dominated points of the actual population; the second should correspond to points dominated by first front, and so on. Thus, for all individuals belonging to the first front is assigned the value $nf = 1$, for all individuals belonging to the second front is assigned the value $nf = 2$, and so on. The second index is the crowding distance that corresponds to measure the distance around of the a determined individual that not is occupied by another solution.

2.6.4 Use of GA in Real-Time Systems

An example of the application of GA in real-time systems is the work by Briand *et al.* (2006). GAs were used to provide automated analysis of the schedulability of tasks during the design process, as well as to test the systems' response time to events in an effective manner - once implemented. Their main goal was to automate stress testing, i.e. based on the system task architecture, the derivation of test cases that maximize the chances of critical deadline misses within the system. The second goal was to enable, at design time, an early but realistic analysis of tasks' schedulability. The authors have developed a specific solution based on genetic algorithms and implemented it in a tool. Case studies were run and results showed that the tool was effective at identifying test cases that would likely stress the system to such an extent that some tasks could miss deadlines. Furthermore, the case studies could identify scenarios that were deemed to be schedulable and, nevertheless, exhibited missed deadlines. Most work using genetic algorithms in real-time systems fall outside the scope of this paper, as they deal with the issue of allocating tasks to multiprocessors, such as the work by Yoo (2009) and ManChon *et al.* (2011).

3 Minimizing the Mode-Change Latency

In this chapter we present a method for latency and/or offsets minimization in fixed priority systems across a mode-change, using single and multi-objective GAs. To validate the efficacy of this method, we perform a number of case studies each illustrating distinct mode-change scenarios.

The focus of this chapter is to address this challenge and propose a workable solution, which involves the modeling of the approach and its validation. Therefore, it is not the aim of this research to experiment with different meta-heuristics or evolutionary algorithms and compare their performance. We have opted to use genetic algorithms due to their known efficiency and simplicity. More advanced approaches could certainly be adopted here, but the results obtained in this work have shown that the proposed approach is feasible, i.e. it is able to find solutions that satisfy the requirements defined for the problem. Therefore, we reserve the comparison analysis of meta-heuristics and evolutionary algorithms to future work, see chapter 5.

The choice of Genetic Algorithms may be strengthened by the taxonomy reported in (CORTE *et al.*, 2012), which divides metaheuristics frameworks into three classes:

1. Local search metaheuristics, which operate on a single complete solution and iteratively improve it by making small adjustments called moves;

2. Population-based metaheuristics, which operate on a set of solutions and find better solutions by combining solutions from that set into new ones;

3. Constructive metaheuristics, which build a solution by working with a single, unfinished, solution and adding one solution element at a time.

We found that methods belonging to the Local search metaheuristics risk to be trapped in unfeasible spaces, while methods belonging to constructive metaheuristics are promising but quite difficult to be calibrated and require further research.

3.1 Model and Approach to Minimization

The flow of the optimization process using GA is illustrated in Fig. 6. Each one of the steps presented are explained in details as follows:

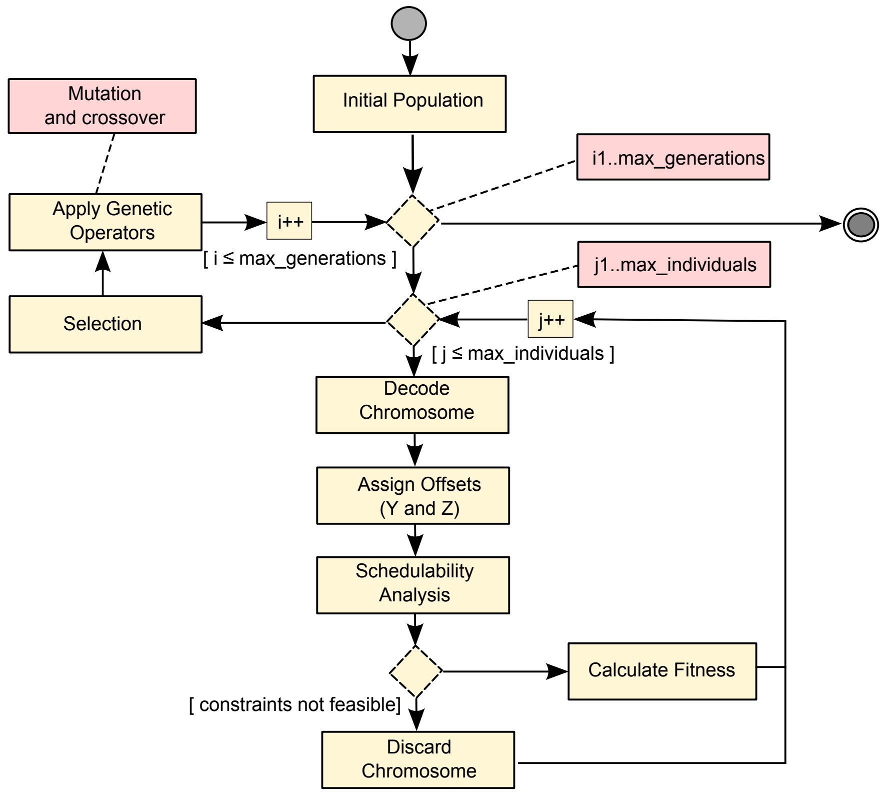


Figure 6 – Diagram of the Optimization Using GA

- *Initial Population:* The first step is to create the initial population, can be predetermined or randomly defined. We chose to adopt an initial random population. In this work, the population is the set of offsets from all new-mode tasks. Determining an appropriate initial population for this analysis is not straightforward. Small values for offsets are not good candidates, since small offsets lead to larger latencies. Large offsets are amenable to reduction to some extent, and there is not relationship that establishes a direct function between latency and offsets. Considering that offsets and latency are very application dependent.

- *Decode Chromosome:* The analysis of the problem to be optimized shows that each gene in a chromosome must correspond to an offset value O_i (Y or Z). The genetic algorithm adopted in this work uses a permutation-based representation (COELLO *et al.*, 2007). Each gene is expressed in hexadecimal notation and uses 4 characters that represents an integer value between 0 and 65535 (Fig. 7). A chromosome must have n genes, where n is the number of new-mode tasks across a mode-change.

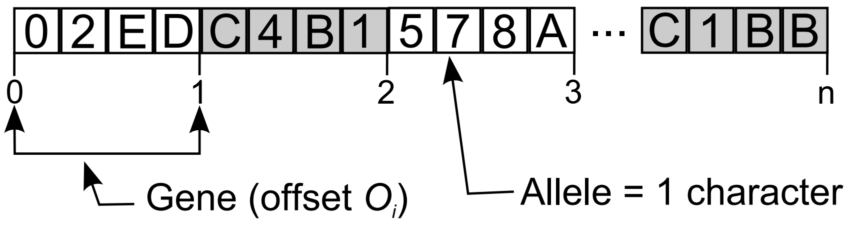


Figure 7 – Structure of a Chromosome (n equal to the number of new-mode tasks)

- *Assign Offsets (Y and Z):* Each chromosome belonging to a generation is decoded, and the value of its genes are assigned to the task offsets.

- *Schedulability Analysis:* Having assigned the offsets, the tasks set is submitted to the mode-change schedulability analysis, using equations (2.6) and (2.15).

- *Check Constraints:* The optimization process must meet constraints represented as Co. Co is a vector that uses binary notation[1]. In the case study described in the sections 3.2.1 to 3.2.4 two constraints were used:

  1. *System feasibility (constraint Co_1)*, i.e. all tasks meet their deadlines, where we apply the schedulability analysis (i.e. $R \leq D$) tests to discard unfeasible solutions. This analysis are based on the worst-case response time windows as shown in equations (2.15).
  2. *Resource sharing via IPCP* protocol (constraint Co_2). This objective is also used to discard solutions that do not comply with the IPCP protocol when resources are shared across a mode-change. In case tasks are independent, the objective is simply removed from the search (i.e. by assigning $Co_2 = 0$).

  At the end of the analysis process, we check the constraints (system feasible and compliance with the IPCP protocol, when necessary). Case the constraints were not feasible the chromosome is discarded.

[1] a 1 represents an active (or valid) constraint and a 0 an inactive one

- *Calculate Fitness:* Upon completion of the real-time analysis, if the constraints ate met, the algorithm proceeds by calculating the fitness function as follows:

$$fitness = \sum_{j=1}^{n} Ob_j.We_j \tag{3.1}$$

where Ob_j corresponds to the value of objective j, and We_j represents its weight (i.e. the relative importance of one objective j in comparison to the other ones). $Fitness$ is the degree of aptitude of a chromosome. The smaller the value of $fitness$, the better the individual within the population.

The minimization of the latency of mode-changes is a multi-objective optimization problem. For example, in the case studies described ahead we used two objectives to compose the fitness function:

1. *Reduction of latency (objective Ob_1)*, using equation (2.18). In this case the minimization of latency is modeled as the main, single/multi-objective GA.
2. *Minimization of offsets (objective Ob_2)*, where we use the summation of all task offsets (i.e. $\sum O_i$) as a measure of the degree of minimization of a candidate solution. Clearly, the lower the sum of offsets for a given solution, the better (or more fit is) a solution in regards to this objective.

However, other objectives can also been adopted such as minimization of the interference between tasks for a given value of latency, and minimization of offsets for higher priority tasks.

The rating or the quality of the solution is represented by the *fitness function*, which expresses the fitness of a particular chromosome within a population. The greater the fitness (smaller value), the higher are the chances of a chromosome adapting to its environment and passing its genes to its successors.

- *Selection:* The selection process chooses the chromosomes with the highest degree of suitability for reproduction, i.e., the larger the fitness of a chromosome, the greater are its chances of reproduction. The selection operator used in this work was *elitist selection* (COELLO *et al.*, 2007).

- *Genetic Operators:* After the selection process, the selected chromosomes are recombined to a certain extent using a crossover operator. Pairs of chromosomes from a population are randomly chosen, taking into account their fitness, and then recombined, generating a new individual that inherits the genetic characteristics of both parents. The crossover operator adopted was the *two-point crossover*(COELLO *et al.*, 2007; JONG, 2006). The *mutation operator* was also used to increase the diversity of a population of chromosomes. The mutation and crossover probabilities were fixed at 10% and 70% respectively, for all scenarios and case studies.

3.2 Case Studies

This section is subdivided into four case studies described as follows:

1. Minimization of offsets (single-objective),
2. Minimization of the latency (single-objective),
3. Minimization of both latency and offsets (weights-based multi-objective), and
4. Minimization of both the latency and the offsets (multi-objective optimization).

These case studies are set to according to scenarios that are likely to be encountered in real applications. They also provide a range of possibilities with enough coverage to allow us to understand the behavior of the search in the face of its objectives and constraints. Table 1 shows a summary of the cases studies introduced in this section.

Table 1 – Summary of the Case Studies

Case Study	**Goal**	**Algorithm**
1	Minimizing Offsets	Single Objective GA
2	Minimizing Latency	Single Objective GA
3	Latency and Offsets	Weights-Based Multi-Objective GA
4	Latency and Offsets	Multi-Objective NSGA-II

The following case studies were performed on a hardware/software platform consisting of 1) Processor: Intel(R) Core(TM) i7-2670QM CPU @ 2.20GHz; 2) Installed Memory (RAM): 6.00 GB; 3) Operating System: Windows 7 Home Premium 64 Bits; 4) RDBMS: Oracle 11g Standard Edition.

3.2.1 Case 1 - Minimizing Offsets

Definition: In this case study the optimization was performed on a set of six generic tasks (Table 2), in order to minimize offsets. The goal of this case is twofold: 1) demonstrate the flexibility of the approach, i.e. the weights-based multi-objective GA mode-change minimization approach can also be used to minimize offsets (in addition to minimization of latency); 2) compare the GA with the algorithm by Real and Crespo (2001). In case the two methods lead to the same values of offsets, this example validates both methods used to offset minimization.

Therefore, we used the same task set used by Real and Crespo (2001). Table 2 presents two modes of operation $M1$ and $M2$; each mode of operation comprises of six tasks, i.e. τ_1 through τ_6. The CPU utilization rate is 73.1% in $M1$ and 66.35% in $M2$. Tasks share two resources, $R1$ and $R2$, as represented in the second half of Table 2. As there are shared resources, the immediate priority ceiling protocol (IPCP) was used to protect against deadlocks and transitive blocking, due to the sharing of resource $R2$ across a mode-change. In mode $M1$, this resource is used by task τ_5 with ceiling priority 4. In mode $M2$ resource $R2$ is used by new-mode tasks τ_2 and τ_5 with ceiling priority 2.

Table 2 – Case Study 1 - Set of tasks

Mode M1						
Tasks	**P**	**C**	**B**	**T=D**	**R**	**TEST**
$\tau_{1(O)}$	1	10	0	100	10	OK
$\tau_{2(W)}$	not active in this mode					
$\tau_{3(O)}$	2	30	25	200	65	OK
$\tau_{4(O)}$	3	40	24	280	115	OK
$\tau_{5(O)}$	4	50	25	300	165	OK
$\tau_{6(O)}$	5	60	0	350	200	OK

Mode M2						
Tasks	**P**	**C**	**B**	**T=D**	**R**	**TEST**
$\tau_{1(U)}$	1	10	0	100	10	OK
$\tau_{2(W)}$	2	20	25	120	55	OK
$\tau_{3(C)}$	3	30	25	270	85	OK
$\tau_{4(U)}$	4	40	24	280	155	OK
$\tau_{5(C)}$	5	50	25	350	180	OK
$\tau_{6(O)}$	not active in this mode					

Resources	**C**	**Mode M1**		**Mode M2**	
		Users	**Ceiling**	**Users**	**Ceiling**
$R1$	25	τ_3,τ_5,τ_6	2	τ_3,τ_5	3
$R2$	25	τ_5	4	τ_2,τ_5	2

Note: The column "TEST" shows that the task is feasible in steady-state mode.

GA configuration: To determine the best parameters to be used in the optimization process using GA, eight distinct scenarios S_1, $S_2 \dots S_8$ were created (Table 3). Each scenario: 1) varies the size of the population and the number of generations, and 2) was subject to ten repetitions, to measure the accuracy of the data obtained.

Table 3 – Case Study 1 - Scenarios for GA Optimization

Parameters	**Scenarios**							
	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8
Population	100	100	250	250	500	500	1000	1000
Generations	25	50	25	50	25	50	25	50

Modeling: The minimization of offsets was treated as a single-objective minimization problem with two constraints, i.e., the schedulability of the task set across a mode-change and the compliance with the IPCP protocol (due to shared resources).

To achieve these objectives the fitness function was instantiated with the weights $We = \{0, 1\}$, thus canceling the first objective (reduction of latency), and the constraints used were $Co = \{1 ,1\}$, i.e., system feasibility and compliance with the IPCP protocol. The fitness function chosen was the *sum of the offsets of all new-mode tasks*, i.e., $fitness = \sum O_i \;\; \forall \; i \; \epsilon \; \tau$. The smaller value of the sum the better the solution found.

Results: Table 4 shows the summary of the mode-change latency sensitivity analysis for the set of six tasks, using the fitness function set to minimize offsets. The columns represent 1) the scenario used for optimization, 2) the average number of tests processed during the optimization, 3) the average time spent on each repetition of the scenario, 4) the lowest, highest and the average sum of offsets, and 5) the variation of the average in relation to the lowest value.

At it can be seen, scenario S_8 presented the smallest sum of offsets (265).

Table 4 – Case Study 1 - GA Sensitivity Analysis to Offset Minimization

Scen.	**# of Analysis**	**Time (Min)**	**Sum of Offsets**			
			Low	**High**	**AVG**	**Var.**
S_1	3418	0.81	265	281	272.9	3.0%
S_2	6110	1.16	265	289	274.8	3.7%
S_3	8364	1.39	265	281	267.9	1.1%
S_4	15053	2.55	265	267	267.9	0.3%
S_5	16504	2.89	265	267	265.7	0.3%
S_6	30083	4.86	265	268	265.3	0.1%
S_7	33224	5.58	265	266	265.1	0.0%
S_8	**60143**	**9.95**	**265**	**265**	**265.0**	**0.0%**

After optimization, the values of the best offsets obtained are shown in Table 5. The left table describes old-mode tasks across the mode-change. Likewise, the right table shows new-mode tasks during the transition.

The left table shows the deadlines D, the worst-case response time obtained with the genetic algorithm R^2, the worst-case response time R^1 obtained with the algorithm of Real and Crespo (2001), variable x, and the results of the schedulability test. The right table shows the same attributes for all new-mode tasks. It includes the offsets for each task.

Table 5 – Case Study 1 - Offsets Obtained ^{1}Algorithm of Real and Crespo (2001) and ^{2}Genetic Algorithm

Old-Mode Tasks M1 → M2						
Task	**T=D**	**R^1**	**R^2**	**x^1**	**x^2**	**Test**
τ_1	100	10	10	0	0	OK
τ_3	200	65	65	0	0	OK
τ_4	280	115	115	101	101	OK
τ_5	300	195	195	1	1	OK
τ_6	350	260	260	1	1	OK

New-Mode Tasks M1 → M2						
Task	**T=D**	**O^1**	**R^1**	**O^2**	**R^2**	**Test**
τ_1	100	0	10	0	10	OK
τ_2	120	195	55	195	55	OK
τ_3	270	0	70	0	70	OK
τ_4	280	70	240	70	240	OK
τ_5	350	0	350	0	350	OK

3.2.2 Case 2 - Minimizing Latency

Definition: In this case study the optimization was performed on a set of 21 tasks (Tables 6 and 7). This set of tasks was based on the Generic Avionics Platform (GAP), initially described by Locke *et al.* (1991). Table 6 presents mode $M1$ (Cruise Control) and Table 7 mode $M2$ (Defense). The CPU utilization rate is 76.58% in $M1$ and 85.01% in $M2$. The goal of this study was to search for the solutions that minimize the latency. There is no concern for the resulting configuration of task offsets in this case.

GA Configuration: We applied the basic GA, where We = {1, 0}. thus canceling the second objective (minimization of offsets), and the constraints used were $Co = \{1\ ,1\}$, i.e., system feasibility and compliance with the IPCP protocol.

Modeling: This search is modeled as a single-objective optimization problem, where latency minimization is the single objective, and system feasibility is treated as a constraint as in the preceding case.

Table 6 – Case Study 2 - GAP Tasks for Cruise Mode

Task	**Cruise Control Mode M1**						
	Description	**P**	**C**	**T**	**D**	**R**	**Test**
$\tau_{1(O)}$	Auto-Pilot	1	10	1000	50	10	OK
τ_2	not active in this mode						
$\tau_{3(O)}$	Radar Tacking Filter	12	200	2000	1200	742	OK
$\tau_{4(O)}$	RWR Contact Mgmt	13	5	2000	1400	747	OK
$\tau_{5(O)}$	Data Bus Pull Service	4	10	400	400	100	OK
τ_6	not active in this mode						
$\tau_{7(O)}$	Mission Advisor	5	20	600	450	120	OK
$\tau_{8(O)}$	Fueling Mgmt	6	50	800	500	170	OK
τ_9	not active in this mode						
$\tau_{10(O)}$	Nav Update	15	80	1100	1550	977	OK
$\tau_{11(O)}$	Display Graphic 1	16	40	1700	1600	1187	OK
$\tau_{12(A)}$	Display Hook Update	17	100	1700	1650	1397	OK
$\tau_{13(O)}$	Tracking Target Upd	10	30	2000	800	342	OK
$\tau_{14(O)}$	Display Graphic 2	11	90	3000	900	442	OK
τ_{15}	not active in this mode						
$\tau_{16(O)}$	Nav Steering Cmds	2	20	250	60	30	OK
$\tau_{17(O)}$	Display Stores Updates	3	60	250	120	90	OK
$\tau_{18(O)}$	Display Keyset	14	10	3000	1500	897	OK
$\tau_{19(O)}$	Display Stat Update	7	30	4000	590	200	OK
$\tau_{20(O)}$	BET E Status Update	8	15	20000	600	215	OK
$\tau_{21(O)}$	Nav Status	9	17	20000	700	232	OK

Table 7 – Case Study 2 - GAP Tasks for Defense Mode

Task	**Defense Mode M2**						
	Description	**P**	**C**	**T**	**D**	**R**	**Test**
τ_1	not active in this mode						
$\tau_{2(W)}$	Weapon Release	1	30	2000	50	30	OK
$\tau_{3(C)}$	Radar Tacking Filter	2	20	250	60	50	OK
$\tau_{4(C)}$	RWR Contact Mgmt	3	50	250	120	100	OK
$\tau_{5(C)}$	Data Bus Pull Service	4	10	400	400	110	OK
$\tau_{6(W)}$	Weapon Aiming	5	30	500	450	140	OK
τ_7	not active in this mode						
τ_8	not active in this mode						
$\tau_{9(W)}$	Radar Target Update	6	50	500	500	190	OK
$\tau_{10(C)}$	Nav Update	7	80	590	590	340	OK
$\tau_{11(C)}$	Display Graphic 1	8	90	800	600	440	OK
$\tau_{12(C)}$	Display Hook Update	9	20	800	700	460	OK
$\tau_{13(C)}$	Tracking Target Upd	10	50	1000	800	740	OK
τ_{14}	not active in this mode						
$\tau_{15(W)}$	Weapon Protocol	11	10	2000	900	750	OK
$\tau_{16(C)}$	Nav Steering Cmds	12	30	2000	1200	970	OK
$\tau_{17(C)}$	Display Stores Updates	13	10	2000	1400	980	OK
$\tau_{18(C)}$	Display Keyset	14	10	2000	1500	990	OK
$\tau_{19(C)}$	Display Stat Update	15	30	2000	1550	1380	OK
$\tau_{20(C)}$	BET E Status Update	16	10	10000	1600	1390	OK
$\tau_{21(C)}$	Nav Status	17	10	10000	1650	1400	OK

Results: After optimization, the values of the best offsets obtained are shown in Table 8. The left table represents the old-mode tasks across a transition, and their real-time attributes whereas the right table describe the new-mode tasks.

Table 8 – Case Study 2 - Offsets Obtained

Old-Mode Tasks M1 → M2					**New-Mode Tasks M1 → M2**			
Task	**D**	**R**	**x**	**Test**	**D**	**O**	**R**	**Test**
τ_1	50	10	0	OK	not active in this mode			
τ_2	not active in this mode				50	257	30	OK
τ_3	1200	742	601	OK	60	1185	50	OK
τ_4	1400	787	1	OK	120	1187	100	OK
τ_5	400	100	1	OK	400	1143	110	OK
τ_6	not active in this mode				450	973	140	OK
τ_7	450	120	1	OK	not active in this mode			
τ_8	500	170	1	OK	not active in this mode			
τ_9	not active in this mode				500	1026	190	OK
τ_{10}	1550	1007	1	OK	590	787	340	OK
τ_{11}	1600	1197	1101	OK	600	695	440	OK
τ_{12}	1650	1407	1251	OK	700	640	460	OK
τ_{13}	800	342	251	OK	800	500	740	OK
τ_{14}	900	442	401	OK	not active in this mode			
τ_{15}	not active in this mode				900	199	193	OK
τ_{16}	60	30	1	OK	1200	420	272	OK
τ_{17}	120	90	1	OK	1400	397	480	OK
τ_{18}	1500	897	801	OK	1500	0	897	OK
τ_{19}	590	200	1	OK	1550	961	126	OK
τ_{20}	600	215	1	OK	1600	260	877	OK
τ_{21}	700	232	1	OK	1650	136	1191	OK

In short, the processing time was 112.42 minutes using GA, the sum of offsets was 10766 and the latency was 1327. The values obtained using the algorithm of Real and Crespo (2001) will be presented in the case study 3.

Note: Sensitivity does not work in this case study, thus, the optimization was performed just with a population of 2000 individuals for 500 generations.

The results are going to be analyzed later in section 3.3.

3.2.3 Case 3 - Minimizing Latency and Offsets - Weights-Based Multi-Objective

Definition: In this case study the optimization was performed on the same set of tasks (Tables 6 and 7). The goal of this study was to search for the solutions that minimize the worst-case latency of a mode-change as the main priority, and also attempt to keep offsets at minimum values, as a second priority.

GA Configuration: As with the previous case, we have determined the best parameters for the optimization process based on nine distinct scenarios (Table 9). Each scenario was subject to ten repetitions in order to measure the accuracy of the resulting data.

Table 9 – Case Study 3 - Scenarios for Optimization with GAs

Parameters	**Scenarios**								
	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9
Population	500	500	500	1000	1000	1000	2000	2000	2000
Generations	100	250	500	100	250	500	100	250	500

Modeling: This search is modeled as a one-objective optimization problem, where solutions, that are schedulable (i.e. feasible) and with lower offsets, are seen as constraints. As there are no shared resources, the IPCP protocol was not necessary, thus, canceling constraint 2.

To achieve these goals, the objective function was instantiated with weights We = {1, 0.00001}. Thus, the fitness function chosen was the combination of 1) the latency of the mode-change (the fitness value is captured in a decimal part) (Ob_1), 2) the sum of the offsets of all new-mode tasks (Ob_2), where the fitness value is accounted for in the fractional part. The constraints used was $Co = \{1\ ,0\}$, i.e., system feasibility and no resource sharing.

Results: Table 10 shows the summary of the sensitivity analysis of the set of twenty-one tasks. The columns represent: 1) the scenario used for optimization, 2) the average number of tests processed during optimization, 3) the average time spent on each repetition of the scenario, 4) the lowest, high, average and variation between average and lowest worst-case latency, 5) the lowest sum of offset for lowest and 5) high worst-case latency. The minimum values were found in scenario 8.

Table 10 – Case Study 3 - GAs Sensitivity Analysis to Offset Minimization

Scen.	**# Analysis**	**Time (Min.)**	**Measured Latency**				**$\sum$ Offsets**	
			Low	**High**	**AVG**	**Var.%**	**Low**	**High**
1	51501	3.65	1397	1665	1475	5.6	8951	7971
2	126501	8.21	1397	1982	1479	5.9	7533	1397
3	251501	15.78	1397	1636	1464	4.8	7579	5600
4	103001	6.06	1397	1453	1403	0.4	8170	9389
5	253001	15.45	1397	1417	1399	0.1	7540	7696
6	503001	38.06	1397	1397	1397	0.0	7585	7585
7	503001	14.09	1397	1397	1397	0.0	7657	7657
8	506001	31.25	1397	1397	1397	0.0	7133	7693
9	1006001	64.43	1397	1412	1399	0.1	7487	7558

After optimization, the values of the best offsets obtained are shown in Table 11. The left table represents the old-mode tasks across a transition, and their real-time attributes whereas the right table describe the new-mode tasks. The parameters with superscript 1 are the ones from the algorithm of Real and Crespo (2001), whereas the ones with superscript 2 are the ones resulting from the GA.

Table 11 – Case Study 3 - Offsets Obtained ^{1}Algorithm of (REAL; CRESPO, 2001) ^{2}Genetic Algorithm

	Old-Mode Tasks M1 → M2						**New-Mode Tasks M1 → M2**					
Task	**D**	**R^1**	**R^2**	**x^1**	**x^2**	**Test**	**D**	**O^1**	**R^1**	**O^2**	**R^2**	**Test**
τ_1	50	10	10	0	0	OK	not active in this mode					
τ_2	not active in this mode						50	0	40	288	30	OK
τ_3	1200	1152	802	1	1	OK	60	119	50	1152	50	OK
τ_4	1400	1157	1017	1	1	OK	120	70	120	1152	100	OK
τ_5	400	200	100	1	1	OK	400	0	210	256	110	OK
τ_6	not active in this mode						450	0	260	400	140	OK
τ_7	450	230	120	1	1	OK	not active in this mode					
τ_8	500	310	170	1	1	OK	not active in this mode					
τ_9	not active in this mode						500	286	154	897	190	OK
τ_{10}	1550	1257	1137	1	1	OK	590	1546	340	816	340	OK
τ_{11}	1600	1417	1277	1	1	OK	600	1546	440	608	440	OK
τ_{12}	1650	1557	1397	1101	251	OK	700	1546	460	146	106	OK
τ_{13}	800	562	342	1	251	OK	800	1546	740	400	740	OK
τ_{14}	900	722	492	1	1	OK	not active in this mode					
τ_{15}	not active in this mode						900	1546	750	146	356	OK
τ_{16}	60	60	30	1	1	OK	1200	1546	970	146	866	OK
τ_{17}	120	120	90	1	1	OK	1400	1546	980	146	881	OK
τ_{18}	1500	1167	1037	1	1	OK	1500	1546	990	146	901	OK
τ_{19}	590	470	200	1	1	OK	1550	1546	1380	288	949	OK
τ_{20}	600	485	215	1	1	OK	1600	1546	1390	146	1141	OK
τ_{21}	700	532	232	1	1	OK	1650	567	1650	0	1397	OK

In summary, by using GA, the number of processed analysis (i.e. # iterations) was 506001, the total processing time was 31.25 minutes, the sum of offsets was 7133 and the worst-case latency was 1397. Using the algorithm Real and Crespo (2001) the number of analysis processed was 53145, the processing time was 4.35 minutes, the sum of offsets was 16502 and the worst-case latency was 2936.

The results are going to be analyzed later in section 3.3.

3.2.4 Case 4 - Minimizing Latency and Offsets - Multi-Objective

Definition: In this case study the optimization was performed on the same set of tasks (Tables 6 and 7). The goal of this study was to search for the solutions that minimize both the worst-case latency of a mode-change and task offsets using a multi-objective GA.

GA Configuration: We applied the *NSGA-II* algorithm, where We = {1, 1}. Thus, the objectives 1 and 2 have the same weights in the optimization process. The crossover and mutation operators used for NSGA-II are the same as the ones employed in the GA. The constraints used were $Co = \{1\ ,0\}$, i.e., system feasibility and no resource sharing.

Modeling: This search is modeled as a multi-objective optimization problem, where latency and offsets are the main objectives and schedulability is a constraint. As with the previous case, since there are no shared resources, the IPCP protocol was not necessary, thus, canceling constraint 2.

Results: After optimization, the values of the best offsets obtained are shown in Table 12. The left table represents the old-mode tasks across a transition, and their real-time attributes whereas the right table describe the new-mode tasks. The parameters with superscript 1 are the ones from the algorithm Real and Crespo (2001), whereas the ones with superscript 2 are the ones resulting from the GA and with superscript 3 are the ones resulting from the NSGA-II.

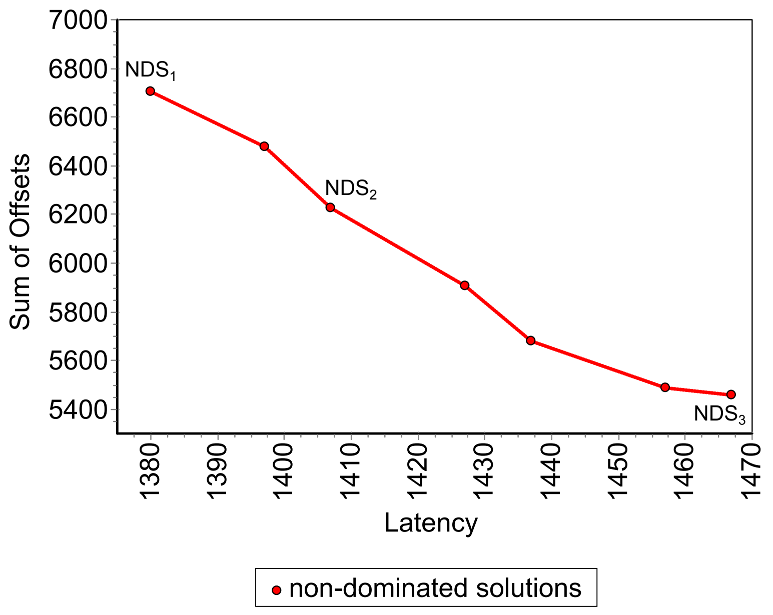


Figure 8 – Non-dominated Solutions Obtained using NSGA-II.

Fig. 8 shows the best frontier of non-dominated solutions found during the optimization process using NSGA-II. From the solutions illustrated, three were used as a comparison in Table 12: NDS_1, the solution that has the smallest value of mode-change latency; NDS_2, an intermediate solution and NDS_3, the solution that has the smallest sum of offsets. However, any of the solutions could be used, as the criteria for selection may vary according to the premises adopted by the system designer.

Table 12 – Case Study 4 - Offsets Obtained with NSGA-II [1] NDS_1 (Lower Latency) [2] NDS_2 (Intermediate) [3] NDS_3 (Lower Offsets)

	Old-Mode Tasks M1 → M2						**New-Mode Tasks M1 → M2**					
Task	$\mathbf{R}^1$	$\mathbf{R}^2$	$\mathbf{R}^3$	$\mathbf{x}^1$	$\mathbf{x}^2$	$\mathbf{x}^3$	$\mathbf{O}^1$	$\mathbf{R}^1$	$\mathbf{O}^2$	$\mathbf{R}^2$	$\mathbf{O}^3$	$\mathbf{R}^3$
τ_2	not active in this mode						0	40	0	40	0	40
τ_3	812	812	812	601	601	601	1248	50	1248	50	480	50
τ_4	847	847	1007	601	601	1	1280	100	1280	100	1280	100
τ_5	130	130	130	1	1	1	0	140	0	140	0	140
τ_6	not active in this mode						1024	140	512	140	512	140
τ_7	160	160	160	1	1	1	not active in this mode					
τ_8	210	210	210	1	1	1	not active in this mode					
τ_9	not active in this mode						1024	190	1024	190	1024	190
τ_{10}	1107	1137	1197	1	1	1	848	340	880	340	880	340
τ_{11}	1337	1337	1347	1101	1101	1	768	440	768	440	768	440
τ_{12}	1557	1557	1557	1251	1251	1251	0	292	0	292	0	292
τ_{13}	402	402	402	251	251	251	512	740	512	740	512	740
τ_{14}	502	502	502	401	401	401	not active in this mode					
τ_{15}	not active in this mode						0	432	0	432	0	432
τ_{16}	60	60	60	1	1	1	0	712	0	742	0	1002
τ_{17}	120	120	120	1	1	1	0	727	0	757	0	1097
τ_{18}	1007	1007	1107	801	801	1	0	747	0	977	0	1117
τ_{19}	240	240	240	1	1	1	0	1137	0	1167	0	1257
τ_{20}	255	255	255	1	1	1	0	1187	0	1227	0	1357
τ_{21}	272	272	272	1	1	1	0	1377	0	1407	0	1467

Table 13 – Case Study 4 - Comparison between Cases 1-4

Case	**Algorithm**	**# of Analysis**	**Time (Minutes)**	**$\sum$ Offsets**	**Latency of Mode**
Set 0f 6 Tasks					
1	Real et al. (2001)	-	< 1	265	350
1	Single-Objective GA	60143	9.95	265	350
Set of 21 Tasks					
2	Single-Objective GA	506001	112.42	10766	1327
2 and 3	Real et al. (2001)	53145	4.35	16502	2936
3	Weights-Based MOGA	506001	31.25	7133	1397
4	NSGA-II - NDS_1	1002001	395.08	6704	1380
4	NSGA-II - NDS_2	1002001	395.08	6224	1407
4	NSGA-II - NDS_3	1002001	395.08	5456	1467

Comparison: Table 13 presents comparison between cases 1-4 with latency and offset optimization using the algorithm of (REAL; CRESPO, 2001), GA, NSGA-II - S_1 (lowest latency), NSGA-II - S_2 (intermediate) and, NSGA-II - S_3 (lowest sum of offsets). Note:

Sensitivity does not work with NSGA-II. Thus, the optimization was performed just with a population of 2000 individuals for 500 generations.

3.3 Discussion

Case study 1 (Reduction of offsets - single-objective GA) validates the results of the work by Real and Crespo (2001) and vice-versa, as both procedures resulted in the same values for minimum offsets for the new-mode tasks. For example, in Table 5 we notice that the offsets by Real and Crespo (2001) O^1 were $\{0, 195, 0, 70, 0\}$ and the ones from our approach O^2 were also the same.

In Case 2 (Reduction of latency - single-objective GA), it is shown that the reduction to minimum offsets does not necessarily lead to a minimum mode-change latency. This aspect of the analysis was unclear before this work commenced, and it has now been quantitatively corroborated. For example, the latency obtained with Real and Crespo (2001) was 2936, and the one by the GA was 1327, a gain of 54.8% in reduction in the worst-case latency.

In Case 3 (Reduction of latency and offsets using weights-based multi-objective GA), there is an increase in the latency from 1327 to 1397 (i.e. a 5.3% increase) in comparison with Case 2. However, there was a 33.74% reduction in offsets (from 10766 to 7133). Comparing this case with Real and Crespo (2001) algorithm, there is a 52.4% gain in latency and 34.8% gain in offsets (i.e. sum of offsets) reduction.

In Case 4 (Reduction of latency and offsets using a multi-objective NSGA-II algorithm), seven non-dominated solutions were found with NSGA-II (Fig. 8): The first solution had the lowest mode-change latency, but a higher sum of offsets. The last solution had a higher latency and a lower sum of offsets. Therefore, in this case, it is also possible to see that the reduction of the sum of offsets caused an increase in latency.

As we moved from a weights-based multi-objective GA (case 3) to NSGA II (case 4), there was a 1.22% decrease in the worst-case latency of the mode-change (i.e. in favor of the NSGA II). However, the reduction in the sum of offsets was 6.01%.

Based on the results and previous analysis, we ranked the solutions according to their ability to reduce worst-case mode change latency as follows:

1. *Single-Objective GA (Case Study 2):* This was the best solution as far as lowest worst-case latency is concerned. However, the sum-of-offsets was larger in comparison with the other approaches.

2. *Multi-Objective GA (NSGA-II) to reduce the worst-case latency and sum of offsets (Case Study 4):* NSGA-II showed to be the best choice where it is necessary to reduce worst-case latency and sum of offsets simultaneously, as it offers a set of well-balanced solutions (i.e. non-dominant solutions) to the system designer.

3. *Weights-Based Multi-Objective GA to reduce the worst-case latency and sum of offsets with weights (Case Study 3):* This approach has showed the worst results in comparison with NSGA-II and single-objective GA regarding the reduction of the worst-case latency only.

Note that our solution using GA can also accommodate the requirement of promptness as defined by Real and Crespo (2001). It would be required the definition of a fitness function whereby the objective is the minimization of the sum of offsets of the higher-priority tasks of interest (i.e. the ones that respond to alarms and have an mode-change deadline (MCD) (section 3.1). It is also possible to allow the fitness function to process a task's MCD, such as defined by Real and Crespo (2001). In this regard, the work presented in this research can be seen as a superset of the analysis of Real and Crespo (2001), i.e. it is the general case that can be instantiated through adjustments to the fitness function for more specialized requirements such as promptness.

As mentioned earlier in this work, our goal in this work is completely different from the work of Real and Crespo (2001) in that it lies in reducing the worst-case latency of a mode-change. The assumption is that the mode-change itself (and not the tasks across a mode-change) has a deadline, and the mode-change should be completed before this deadline. This is due to the fact that during a mode-change the system delivers its functionality and performance only partially as either the new-mode task set is not yet complete or both.

A priori analysis of the algorithm of Real and Crespo (2001) shows that it is not easily adaptable to accomplish the minimization of mode-change latencies due to the existence of old-mode tasks and consequently the window x in the analysis. Finding the latency is simple only if we consider changed and wholly new-mode tasks, because they all start with respect to the MCR, considering their offsets. The challenge is with the old-mode tasks and purely periodic tasks with an offset Z. With old-mode tasks, we have their R (worst-case response time (WCRT)) that depends on x. This WCRT for an old-mode task is not necessarily the time that goes farther into the transition, precisely because of x. For example, one can have an old-mode task with WCRT 100, but with an $x = 90$. It could be the case that for $x = 0$ one has a slightly lower WCRT, but the task would effectively complete later with respect to the MCR than it does with an $x = 90$. This issue also occurs with purely periodic tasks with an offset $Z > 0$.

Another point to emphasize is the software tool developed to perform this research, as it fully implements the presented approach. The tool has a high-level interface where the user can select the goals to be achieved when configuring the offsets. From an internal software design perspective, the program is also modular so that it can be easily extended to include more optimization objectives. This software tool is presented in details in Appendix A.

4 Configuring Mode Changes

In chapter 3, we have shown the feasibility of mode-change minimization using GAs. In this chapter we show that the approach is also flexible in that is can accommodate a wider variety of mode-change scenarios.

In order to do so, we classify the mode changes in five types and extend previous case studies to establish that it is possible to 1) generate these types of changes by adjusting tasks' parameters, and 2) minimize these mode changes using the proposed approach.

4.1 Types of Mode Change

Real and Crespo (2001) proposed two types of mode change: asynchronous and synchronous. In this chapter we adopt the asynchronous mode-change model, where tasks from the new-mode begin execution in parallel with tasks from the old-mode, thus leading to shorter mode-change delays. Old-mode completed tasks are discarded from the system, leaving resources for tasks arriving from the new-mode. Therefore, the number of modes of operation is only limited by the available memory in the system. This approach allows systems to be scalable, with a large number of modes and tasks. It also enables faster mode changes due to the (pseudo) parallelism of old and new-mode tasks, and the early introduction of new-mode tasks right at the beginning of the mode change. This approach requires schedulability analysis, which is provided by existing work (REAL; CRESPO, 2001; PEDRO; BURNS, 1998).

To allow the specification and the minimization of a wider flexibility of mode-change configuration, we classify mode changes in regards to

"the relative rate of completion of old-mode tasks in relation to the completion of new-mode tasks released within an interval δ from the start of the mode change"

The interval δ is called the *significant interval* and it is defined shortly within the next few paragraphs. Certain applications may impose an ordering of tasks during the start of the transition, e.g. old-mode tasks should be removed earlier (usually immediately at the start of a change) and new-mode tasks introduce later in order to fulfill specific functional requirements. In view of that, the following types of mode change can be identified:

1. *All-old-mode tasks first (AOF) (Also Synchronous) (Fig. 9)*: As the name implies, in this type of mode change the new-mode tasks are only introduced once all old-mode tasks have completed their execution. There is no overlapping or interference between the two tasks sets. To achieve this configuration, it is necessary to assign larger offsets for all new-mode tasks in regards to new-mode tasks' offsets. It is used when new-mode tasks do not carry emergency functions and the old-mode tasks need to complete soon to keep the system in a consistent state. It may also be used when the system's utilization is large in the old-mode and there is the need to release resources back to the system as soon as possible within the transition window. This type of transition is the one that has been traditionally used in many real-time systems. Since the two modes do not overlap, there is no interference between old and new-mode tasks and therefore no need for schedulability analysis. This category does not explore the potential for parallelism between tasks from distinct modes and therefore may lead to longer transition windows.

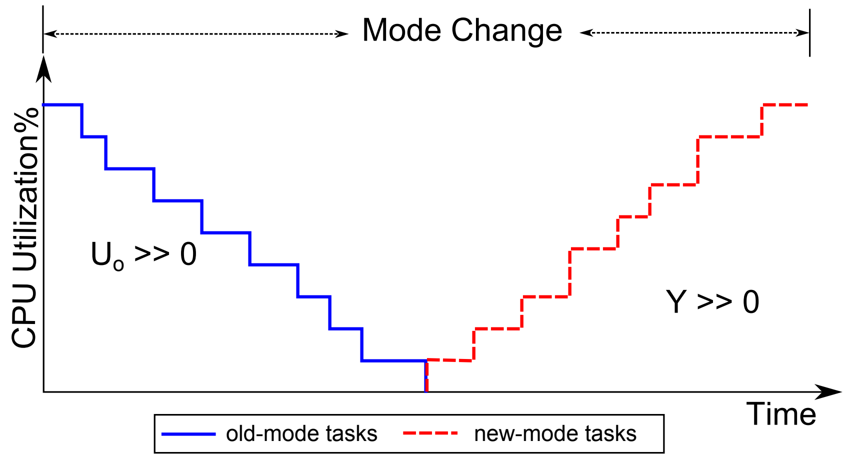


Figure 9 – All Old First (AOF)

2. *All-new-mode tasks first (ANF) (Fig. 10)*: In this configuration, all new-mode tasks precede the old-mode tasks in the transition. The old-mode tasks are preempted by the new-mode ones at the start of a mode change. The first old-mode task executes at the end of a busy period of the new-mode task set. Like the AOF approach, there is no interference between the old and the new task sets. This arrangement of tasks requires that new-mode tasks have small offsets and their priority should be higher than the old-mode tasks. The new-mode tasks are introduced at the beginning of the mode change, either all at once or in a staggered (step-wise) fashion. From the application perspective, this type of transition is required when the new-mode is an emergency mode, i.e. the new tasks are added to the system to reply (or service) an alarm or critical condition, whereas the old-mode does not represent a relatively high priority.

Note that it is not always possible to configure such type of change: it depends on the utilization and the deadlines of the old-mode tasks. If the utilization of the old-mode is relatively large, adding the new tasks immediately after the MCR may lead to a processor over utilization followed by missed deadlines.

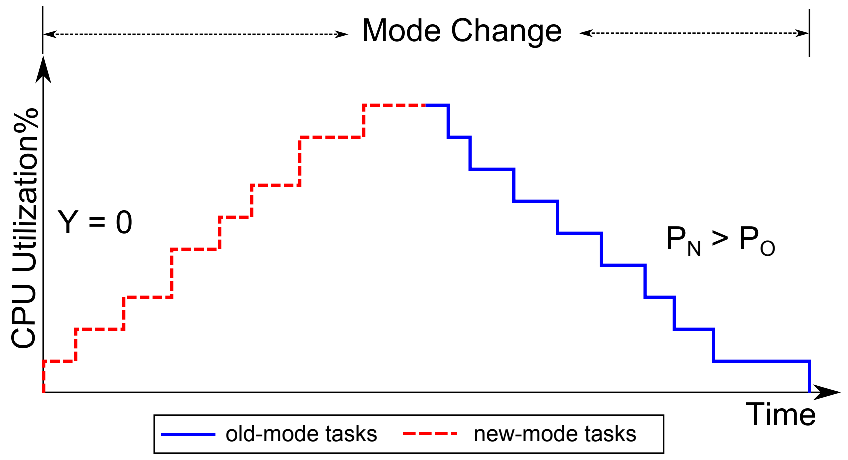


Figure 10 – All New First (ANF)

3. *Mostly-old-mode tasks first (MOF) (Fig. 11)*: Unlike the previous types of changes where the old and new task sets do not overlap in time. In this case most of the old-mode tasks are introduced first and suffer from interference from new-mode tasks that are gradually being introduced into the system. New-mode tasks have larger offsets allowing some old-mode ones to complete first. This model of mode change is recommended when the old-mode tasks need to be completed without restricting the introduction of some new-mode tasks. Due to cross interference, this type of change requires schedulability analysis.

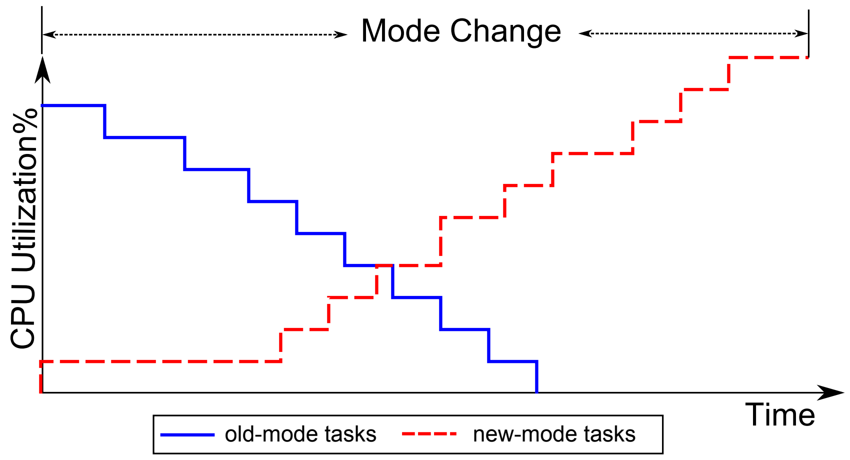


Figure 11 – Mostly Old First (MOF)

4. *Mostly-new-mode tasks first (MNF) (Fig. 12)*: This configuration is achieved by assigning relatively small offsets to the most new-mode tasks. Therefore, the start of a transition is characterized mostly by the insertion of new-mode tasks, and to a lesser extent by the removal of old-mode tasks. It occurs when there is the need to execute new-mode soon (e.g. emergency) and is necessary to allow some old-mode tasks to complete, avoiding missing the deadlines. It is not always possible to perform such type of mode change, since the old-mode tasks may miss their deadlines. There must be sufficient processor spare capacity in the old-mode to allow it. It requires that old-mode tasks have larger deadlines and the tasks set has fewer new-mode tasks.

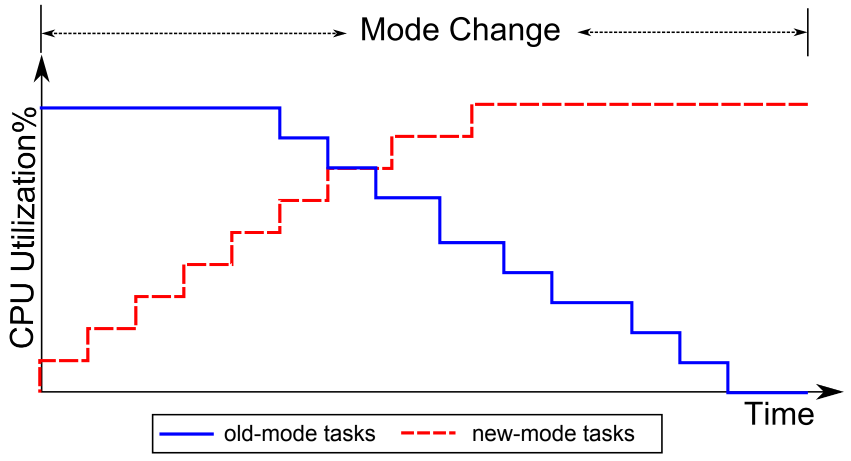


Figure 12 – Mostly New First (MNF)

5. *Balanced mode change (BMC) (Fig. 13)*: In this type of transition, old-mode tasks are removed from the system nearly at the same rate that the new-mode tasks are introduced into the system. Task's offsets and priorities have to be properly assigned to achieve the balanced mode change: For each old-mode task that completes, there must be a new-mode task with same (or close) priority. This type of transition is applied when the old-mode has to be executed in a pseudo-parallel fashion with the new-mode. If resource sharing is required, this must be the only way to allow synchronization of resources.

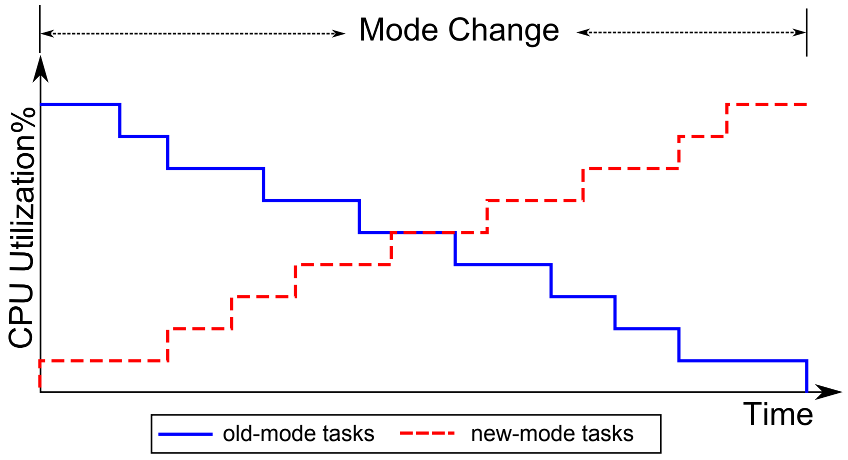


Figure 13 – Balanced Mode Change (BMC)

In essence, the underlying assumption in this model is that the applicant would choose to configure and execute:

- *the old-mode task set first*, when the requirement of releasing resources (memory, CPU utilization, etc) and keeping the consistency of the system (i.e. by closing files and connections) is more important than responding to alarm conditions (which is carried out by the new task set);

- *new-mode task set first*, when the requirement of responding to urgent alarm conditions by releasing and running new-mode tasks first is more critical than releasing resources back to the system and keeping the system in a consistent state;

- *a balanced mix of both task sets*, when both requirements are equally important; or

- *an unbalanced mix of both task sets*, when one requirement is more important than the other by still both should be considered.

Note that:

- *A tolerance factor* ρ is applied to the definition of a Balanced Mode Change. When ρ = 0, a change is balanced if and only if the rate of old to new completed is 50/50. For example, if $\rho = 5$, then a 45/65 rate is a balanced change, but a 0.4 to 0.6 is no longer balanced.

- Aborted tasks do not contribute to the definition.

- Task's priorities may be either increased or decreased. Nevertheless, the DMS priority assignment policy is maintained throughout this work.

The significant interval δ is defined as follows:

$$\delta = \min(L \times k, \max(R_{(O_i)} - x_i), \max(R_{(N_i)} + O_i)) \tag{4.1}$$

where L is worst-case latency of mode change; k is a constant value dependent on the application (which is arbitrarily set to 30% in this work), and it defines how far from the MCR we wish to extend the significant interval; R_O is the set of worst-case mode change response time of all old-mode tasks; x_i is the set of the values of variable "x" for each old-mode task τ_i; R_N is the set of worst-case mode change response time of all new-mode tasks and O_i is the set of offsets of all new-mode tasks.

The type of mode change can be found (or approximated) by the *rate of change* parameter (α), which is given following relation:

$$\alpha = \frac{\#\tau_{ins_{(N)}}}{\#\tau_{rmv_{(O)}} + \#\tau_{ins_{(N)}}} \tag{4.2}$$

where $\#\tau_{ins_{(N)}}$ is the number of new-mode tasks inserted within the significant interval δ from the MCR (i.e $O \leq \delta$), and $\#\tau_{rmv_{(O)}}$ is the number of old-mode tasks removed within the same interval (i.e $(R_O - x) \leq \delta$).

Note: When these exists a substantial difference between the number of new-mode and old-mode tasks, we can use the following equation:

$$\alpha = \frac{(\#\tau_{ins_{(N)}} / \#\tau_{(N)})}{(\#\tau_{rmv_{(O)}} / \#\tau_{(O)}) + (\#\tau_{ins_{(N)}} / \#\tau_{(N)})} \tag{4.3}$$

where $\#\tau_{(N)}$ is the number of new-mode tasks, $\#\tau_{(O)}$ is the number of old-mode tasks. This equation considers the relative percentage of tasks inserted and removed and not the absolute number of tasks. In absolute terms, a mode-change may be deemed of being one type, whereas in percentage terms it may be another type. The issue on whether one should adopt either the percentage or absolute values for the definition is application dependent.

The relationship between added new-mode tasks to old-mode tasks removed during the significant interval δ allows the identification the behavior of a mode change. The mode-change type, as defined according to the value of α, is described in Table 14.

Table 14 – Mode-Change Types

Acronym	**Type**	**Rate of Change**	**General Application**
BMC	Balanced	$0.4 \le \alpha \le 0.6$	used when a mix of old and new-mode tasks need to share the processor during the start of a transition.
AOF	All-old-first	$\alpha = 0$	used to complete old-mode task set *ASAP* to release resources back to the system.
MOF	Mostly-old-first	$\alpha < 0.4$	used when the old-mode tasks need to be completed quickly without restricting the release of some new-mode tasks.
ANF	All-new-first	$\alpha = 1$	used when the release of all the new-mode is a priority (e.g. due to an emergency/alarm condition and the completion of the old-mode tasks can be delayed).
MNF	Mostly-new-first	$\alpha > 0.6$	It is used when we need to execute the new-mode ASAP (e.g. emergency) and it is necessary to release some resources by completing a few old-mode tasks.

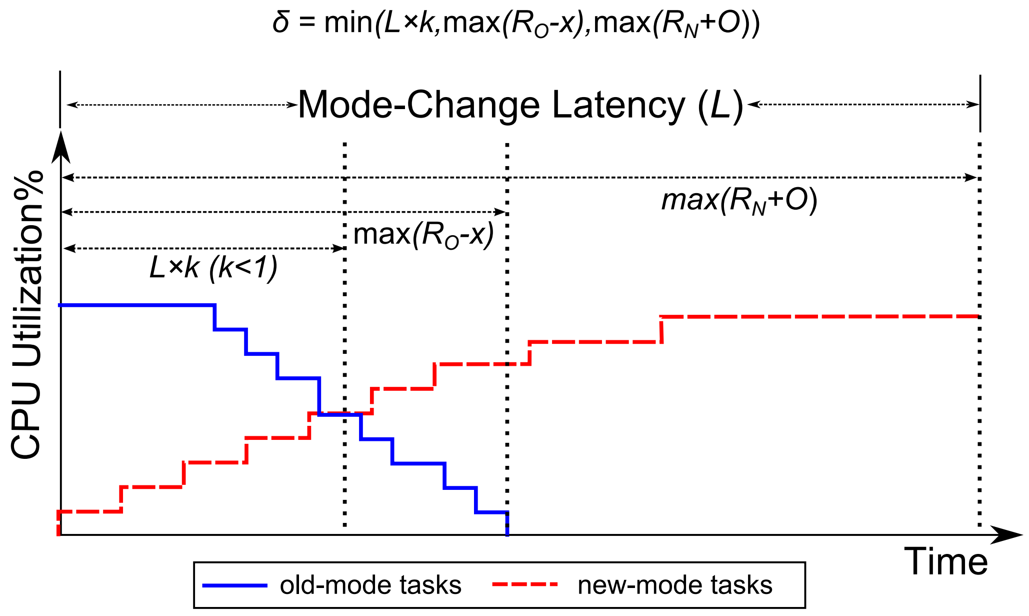


Figure 14 – $\delta = L \times k$ (BMC)

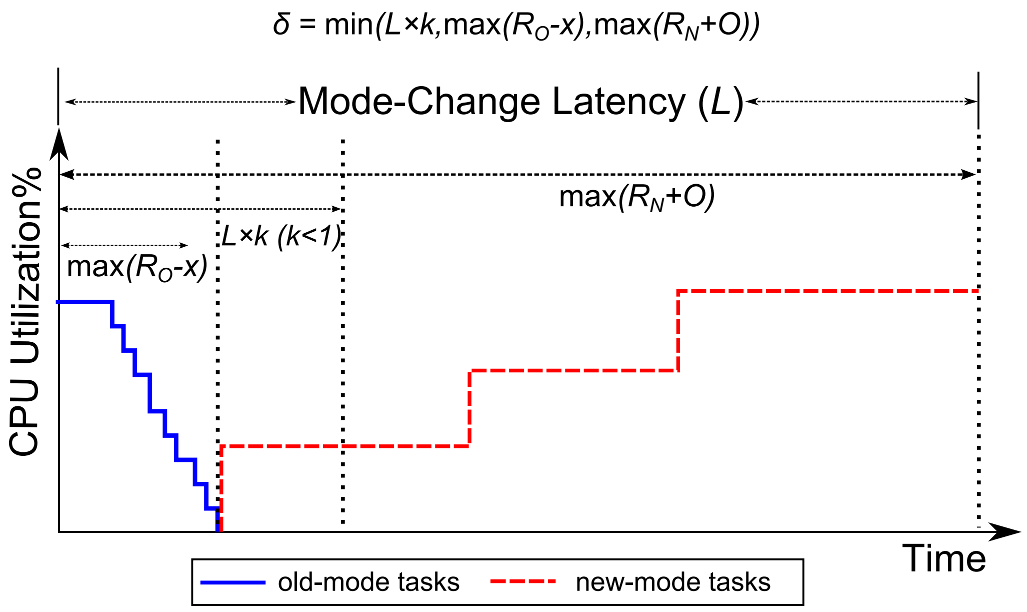


Figure 15 – $\delta = \max(R_O - x))$ (AOF)

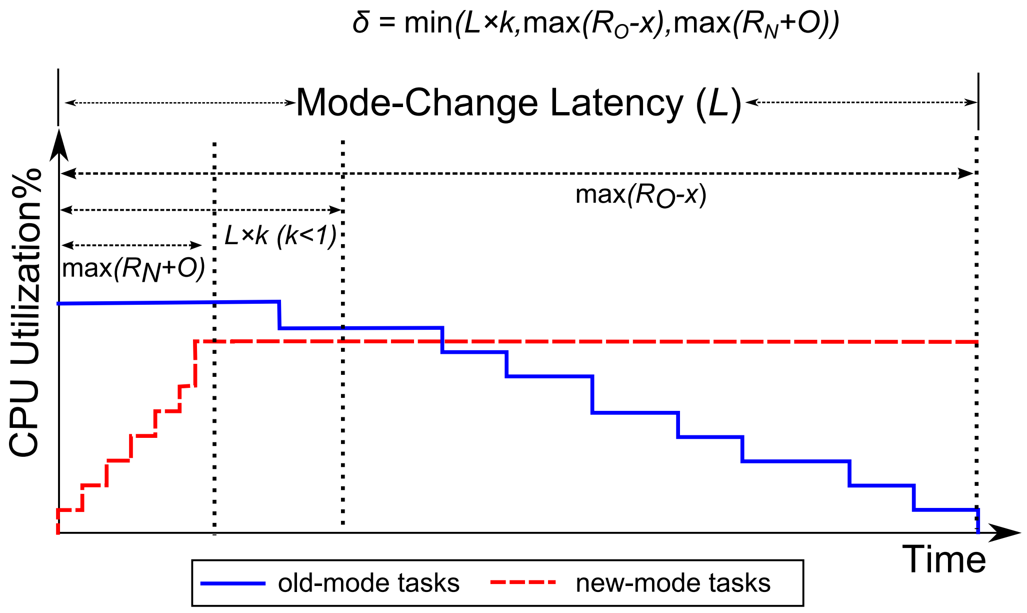


Figure 16 – $\delta = \max(R_N + O)$ (ANF)

Therefore, in one hand the rate of change α is maximum (i.e. 1) when the new-mode task set is all introduced first at the start of a transition. On the other hand, it is zero when all the old-mode task set is first completed before the introduction of any new-mode task. It is a balanced mode change when α tends to values around (or close to) 50%. Otherwise, for the cases not covered above, it falls to either a mostly-old-first or mostly-new-first depending upon which task set has more tasks completed within the significant interval.

Fig. 14 shows an example where $\delta = L \times k$ - the mode-change type is a BMC since five old-mode tasks are completed and five new-mode tasks are inserted. It could have been either a MNF or a MOF in case one task set outnumbered the other (in terms of releases and completions) within the δ interval. Fig. 15 illustrates the case where $\delta = \max(R_O)$ and the mode-change type is an AOF. In case we had a few releases new-mode tasks released within δ, another possible type of mode change would be MOF. At last, Fig. 16 provides an example where $\delta = \max(R_N + O)$ (i.e. an ANF mode change). Similarly, another possible type would have been a MNF in case we had a few old-mode tasks completed within δ. Note that α and k can be adjusted according to the application.

4.2 Model and Approach to Minimization

The approach to minimization used in this section employs the same features, i.e. chromosome structure, genetic operators, initial population and fitness function described in the section 3.1, except the constraints. The constraints adopted in this chapter are the following:

1. *System feasibility (Co_1)*, i.e. all tasks meet their deadlines, where we apply the schedulability analysis (i.e. $R \leq D$) tests to discard unfeasible solutions. This analysis is based on the worst-case response time windows as shown in equations (2.15).
2. *Offsets within a range (Co_2).* This constraint is used to select solutions where the offsets will be within a given range of values for some tasks.
3. *WCRT within a range (Co_3).* This constraint is used to select solutions where the WCRT will be within a range of values for some tasks.
4. *Latency within an acceptable range (Co_4).* This constraint is used to select solutions where the worst-case latency will be within an acceptable range of values.

4.3 Case Studies

- Cases 1 and 2 and 6 are similar to the ones from the previous chapter regarding the goals. The task set is new. The goal was to identify the type of mode change that is generated for each case.

- The remaining cases are new. They introduce new conditions and scenarios or possibilities of expressing a mode change. The goal is to show the versatility of the approach and that it is possible to configure a mode change.
- Cases 1-6 are spontaneous. Cases 7-12 were cases where a particular type of mode change was forced.

Considering that P_N is the set of priorities of all the new-mode tasks, P_O the set of priorities of the old-mode tasks, O the set of offsets of all the new-mode tasks (Y or Z) and $R_{(MC)_O}$ the set of worst-case response time during mode change of all the old-mode tasks, this section presents twelve case studies described as follows:

1. Minimizing Offsets and Latency (MOF)
2. Minimizing Latency and Offsets (AOF)
3. Minimizing Latency Imposing Offsets within a Range (MOF)
4. Minimizing Latency with WCRT within a Range (MOF)
5. Minimizing Offsets with Latency within an Acceptable Range (MOF)
6. Minimization Using the Algorithm of Real and Crespo (2001) (BMC)
7. Minimizing Latency with $P_N > P_O$ (MNF)
8. Minimizing Latency with $P_O > P_N$ (AOF)
9. Minimizing Offsets with $P_N > P_O$ (AOF)
10. Minimizing Offsets with $P_O > P_N$ (AOF)
11. Minimizing Latency with $P_O > P_N$ and $O > \max(R_O - x)$ (AOF)
12. Minimizing Latency with $P_N > P_O$ (ANF)

fictitious tasks were used to perform the minimization in all case studies that will be presented below.

In case studies 1 to 11 the minimization was applied to the same task set, (Table 15), composed of modes $M1$ (old-mode) and $M2$ (new-mode). Each mode of operation is composed of 10 tasks ($\tau_1 \ldots \tau_{10}$). The CPU utilization rate in both modes is 78.22%. Case study 12 was performed with a modified task set.

Table 15 – Task Set Used in Cases 1 through 11

Mode M1						
Tasks	**P**	**C**	**T**	**T**	**R**	**Test**
$\tau_{1(O)}$	6	10	450	450	170	OK
$\tau_{2(W)}$	not active in this mode					
$\tau_{3(O)}$	4	30	300	300	140	OK
$\tau_{4(O)}$	2	20	200	200	45	OK
$\tau_{5(O)}$	7	25	500	500	195	OK
$\tau_{6(O)}$	5	20	400	400	160	OK
$\tau_{7(O)}$	1	25	100	100	25	OK
$\tau_{8(O)}$	3	40	250	250	85	OK
$\tau_{9(W)}$	not active in this mode					
$\tau_{10(O)}$	8	30	600	600	365	OK

Mode M2						
Tasks	**P**	**C**	**T**	**T**	**R**	**Test**
$\tau_{1(O)}$	not active in this mode					
$\tau_{2(W)}$	1	25	100	100	25	OK
$\tau_{3(C)}$	2	20	150	150	45	OK
$\tau_{4(C)}$	3	30	200	200	75	OK
$\tau_{5(C)}$	4	20	300	300	95	OK
$\tau_{6(U)}$	5	20	400	400	140	OK
$\tau_{7(C)}$	6	25	450	450	185	OK
$\tau_{8(C)}$	7	30	500	500	270	OK
$\tau_{9(W)}$	8	10	600	600	280	OK
$\tau_{10(O)}$	not active in this mode					

The case studies were performed on a hardware/software platform consisting of 1) Processor: Intel(R) Core(TM) i7-2670QM CPU @ 2.20GHz; 2) Memory (RAM): 6.00 GB; 3) Operating System: Windows 7 Home Premium 64 Bits; 4) RDBMS: Oracle 11g Standard Edition.

The case studies are divided to two groups: Cases 1 through 6 are cases where we minimize the latency and/or offsets without imposing a particular type of mode change. The goal is to analyze the resulting type of mode change. Cases 7 through 12 are cases where a particular type of mode change is enforced (or configured) by assigning proper values to task's parameters (such as offsets or priority). The particular type of mode change is also minimized using the proposed approach. By default it is used the "latency definition I", excluding cases 7 and 8 that use both latency definitions.

The case studies are introduced and discussed in details as follows.

4.3.1 Study Case 1 - Minimizing Offsets and Latency (MOF)

Definition: In this case study the minimization was performed to find the smallest sum of offsets for new-mode tasks with the smallest worst-case latency possible The goal of this minimization process is to identify what is the type of mode change when there is low sum of offsets.

GA Configuration: To determine the best parameters to be used in the minimization process using GA, ten distinct scenarios S_1, $S_2 \ldots S_{10}$ were created (Table 16). Each scenario was subjected to ten repetitions, to validate the accuracy of the data obtained. Moreover, these scenarios will be used to perform the minimization in the case studies 2, 3, 4, and 5.

Table 16 – Case Study 1 - Scenarios for GA Minimization

Parameters	**Scenarios**									
	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}
Population	100	100	200	200	500	500	1000	1000	2000	2000
Generations	25	50	50	100	125	250	250	500	500	1000

Modeling: The minimization of offsets and latency was treated as a weights-based multi-objective GA minimization problem with just system feasibility as the constraint. To achieve these objectives the fitness function was instantiated with the weights $We = \{0.0000001{,}1\}$, i.e., $fitness = \sum O_i + (L \times 0.0000001)\ \forall\ i \in \tau$. Thus, the main objective is offset minimization and the second one is latency minimization. The constraints used were $C_o = \{1, 0, 0, 0\}$, i.e, just system feasibility.

Results: Table 17 shows the GAs sensitivity analysis of each scenario submitted to the minimization process using GA. In this table, the columns represent successively: 1) the scenario used for minimization, 2) the average number of tests processed during minimization, 3) the average time spent on each repetition of the scenario, 4) the lowest, 5) highest and 6) average sum of offsets, 7) the variation of the average in relation to lowest value of sum of offsets, 8) the lowest latency for lowest and 9) high worst-case sum of offsets.

Table 17 – Case Study 1 - GAs Sensitivity Analysis

Scen.	**# of Analysis**	**Time (Min)**	**Sum of Offsets**				**Latency**	
			Low	**High**	**AVG**	**Var.**	**Low**	**High**
S_1	2801	0.07	795	1585	1057	170.9%	715	550
S_2	5301	0.16	504	1024	684	75.4%	600	746
S_3	10601	0.31	514	722	614	57.5%	595	585
S_4	20601	0.63	400	688	517	32.6%	600	553
S_5	64001	1.96	391	480	427	9.4%	595	595
S_6	126501	4.01	392	504	427	9.6%	595	500
S_7	253001	8.39	390	456	408	4.7%	595	595
S_8	503001	17.28	392	519	421	8.1%	595	520
S_9	**1006001**	**33.79**	**390**	**441**	**400**	**2.6%**	**595**	**595**
S_{10}	2006001	53.28	390	481	404	3.7%	595	470

Sensitivity analysis: From S_9 onwards we notice that the algorithm presents more stable results, i.e., variance around 2.6 %, and S_{10} with a variance of 3.7%. A population around 2000 individuals is the recommended value for this case.

After minimization, the values of the best offsets obtained are shown in Table 18. The left table describes old-mode tasks across the mode change. Likewise, the right table shows new-mode tasks during the transition. The left table shows the deadlines D, the worst-case response time obtained with the genetic algorithm, variable x, and the results of the schedulability test. The right table shows the same attributes for all new-mode tasks. It includes the offsets for each task. This same table format is adopted in similar tables throughout the remainder of this chapter, and therefore (to avoid repetition) this description is omitted in the following cases.

Table 18 – Case Study 1 - Offsets Obtained

Old-Mode Tasks M1 $\rightarrow$ M2				
Tasks	**T=D**	**R**	**x**	**Test**
τ_1	450	265	1	OK
τ_2	not active in this mode			
τ_3	300	190	101	OK
τ_4	200	45	1	OK
τ_5	500	380	1	OK
τ_6	400	255	1	OK
τ_7	100	25	0	OK
τ_8	250	105	1	OK
τ_9	not active in this mode			
τ_{10}	600	585	1	OK

New-Mode Tasks M1 $\rightarrow$ M2				
Tasks	**T=D**	**R**	**O**	**Test**
τ_1	not active in this mode			
τ_2	100	25	295	OK
τ_3	150	65	0	OK
τ_4	200	135	0	OK
τ_5	300	235	0	OK
τ_6	400	255	0	OK
τ_7	450	290	0	OK
τ_8	500	460	95	OK
τ_9	600	595	0	OK
τ_{10}	not active in this mode			

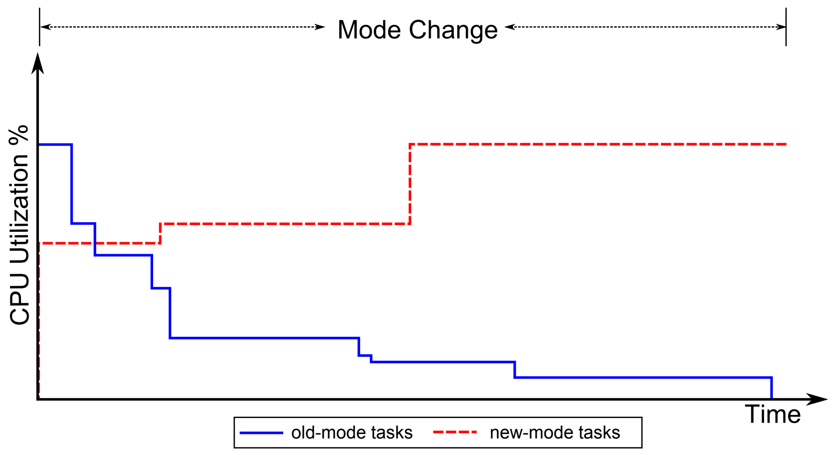


Figure 17 – Case Study 1 - Utilization Chart (MOF)

The graph shown in Fig. 17 shows the processor utilization during the mode-change period. The continuous line represents the old-mode and the dashed line the new-mode (this same scheme is adopted in similar graphs throughout the remainder of this chapter). Clearly, as the old tasks are being completed, the utilization of the old-mode task set is decreased, Likewise, as the new tasks are being introduced the utilization of the new-mode is increased. The processor utilization rate is given by Burns and Wellings (2009):

$$U_i = C_i/T_i \text{ for periodic tasks, or}$$

$$U_i = C_i/Min_i \text{ for sporadic tasks,}$$

where, C_i is the maximum computational time, T_i is the period and Min_i is the minimal interval between releases of new instances of the task i. The processor utilization rate U for a tasks set $\{\tau_1,\tau_2,\tau_3 \ldots \tau_n\}$ is given by:

$$U = \sum_{i}^{n} U_i \tag{4.4}$$

Discussion: Case study 1 reinforces the fact that smaller offsets do not necessarily lead to smaller latencies. This is in line with case 1 from chapter 3. In our example, if the sum of offsets is 481, the latency is 470. However, if offsets are down to 390, there is an increase in the worst-case latency to 595. We can also observe in Table 18 the following values: $\max(R_N+O) = 595$ (τ_9), and $\max(R_O - x) = 586$ (τ_{10}). The latency value was $L = 595$ (from equation (2.18)) and $L \times 30\% = 178.5$, thus, the significant interval was $\delta = min(178.5, 586, 595) = 178.5$. The number of new-mode tasks completed during the significant interval was 2 (τ_3,τ_4) and the number of old-mode tasks removed was 4 (τ_3,τ_4, τ_7,τ_8), resulting in $\alpha = \frac{2}{4+7} = 0.33$. This value indicates a mostly-old-first (MOF) type of mode change. Fig. 17 shows a visual representation that assists in the interpretation of the type of mode change.

This type of mode change has occurred due to the assignment of offsets (by the minimization algorithm) to tasks τ_1 and τ_8 that have the highest priority in new-mode , leaving most of the old-mode tasks preempting the new-mode tasks at the start of the mode change. However, we cannot claim that the minimization used in this case will always provide this type of mode change since it depends on the specific real time parameters of the tasks set.

4.3.2 Case Study 2 - Minimizing Latency and Offsets (AOF)

Definition: In this case study the minimization was performed to find the smallest worst-case latency with the smallest sum of offsets possible for new-mode tasks. The goal of this case study is to identify the type of mode change using the mode-change classification introduced, once the minimization process is completed.

Modeling: The minimization of latency and offsets was treated as a weights-based multi-objective GA minimization problem. We applied weights for each objective with just system feasibility as a constraint (thus, the constraints vector used was set to $C_o = \{1, 0, 0, 0\}$). To achieve these objectives, the fitness function was instantiated with the weights $We =$ {1,0.0000001}, i.e., $fitness = \sum(O_i \times 0.0000001) + L \;\forall\; i \in \tau$.

Results: Table 19 shows the summary of the minimization of the set of ten tasks using the fitness function set to minimize worst-case latency and offsets. In this table, the columns represent successively: 1) the scenario used for minimization, 2) the average number of tests processed during the minimization process, 3) the average time spent on each repetition of the scenario, 4) the lowest, 5) highest and the 6) average measured latency,7) the variation of the average in relation to the lowest value, the sum of offsets for 8) lowest and 9) highest worst-case latency.

Table 19 – Case Study 2 - GAs Sensitivity Analysis

Scen.	**# of Analysis**	**Time (Min)**	**Latency**				**Sum of Offsets**	
			Low	**High**	**AVG**	**Var.**	**Low**	**High**
S_1	2801	0.03	540	1053	713	98.1%	1424	4662
S_2	5301	0.07	385	799	522	44.9%	1187	3495
S_3	10601	0.14	360	410	378	5.0%	971	1100
S_4	20601	0.29	360	385	365	1.3%	753	882
S_5	64001	1.31	360	385	365	1.4%	725	732
S_6	126501	3.26	360	385	365	1.4%	740	805
S_7	253001	6.81	360	360	360	0.0%	692	944
S_8	503001	13.63	360	385	363	0.7%	694	773
S_9	1006001	25.13	360	360	360	0.0%	691	736
S_{10}	**2006001**	**56.69**	**360**	**360**	**360**	**0.0%**	**690**	**740**

Sensitivity analysis: From S_9 onwards we notice that the algorithm presents more stable results, i.e., variance around 0.0 %. A population around 2000 individuals is the recommended value for this case. The same values of the case study 1, showing that this is a general case or trend.

After minimization, the values of the best offsets obtained are shown in Table 20.

Discussion: In case study 2 the objectives were inverted in comparison to case 1, i.e. minimization of the worst-case latency is the major objective followed by the minimization of the sum of offsets. The results obtained demonstrate that whereas the latency was reduced, the offsets were increased instead (compared to case 1). The latency was reduced from to 595 to 360 and the sum of offsets increased from 390 to 690. However, larger offsets may lead to a smaller latencies due to the reduction of the amount of task interference during the mode change. This approximated this type of mode change to a synchronous mode-change since

Table 20 – Case Study 2 - Offsets Obtained

Old-Mode Tasks M1 → M2				
Tasks	**T=D**	**R**	**x**	**Test**
τ_1	450	195	1	OK
τ_2	not active in this mode			
τ_3	300	140	101	OK
τ_4	200	45	1	OK
τ_5	500	290	1	OK
τ_6	400	160	101	OK
τ_7	100	25	0	OK
τ_8	250	85	1	OK
τ_9	not active in this mode			
τ_{10}	600	460	301	OK

New-Mode Tasks M1 → M2				
Tasks	**T=D**	**R**	**O**	**Test**
τ_1	not active in this mode			
τ_2	100	25	260	OK
τ_3	150	45	210	OK
τ_4	200	75	160	OK
τ_5	300	75	60	OK
τ_6	400	155	0	OK
τ_7	450	240	0	OK
τ_8	500	320	0	OK
τ_9	600	360	0	OK
τ_{10}	not active in this mode			

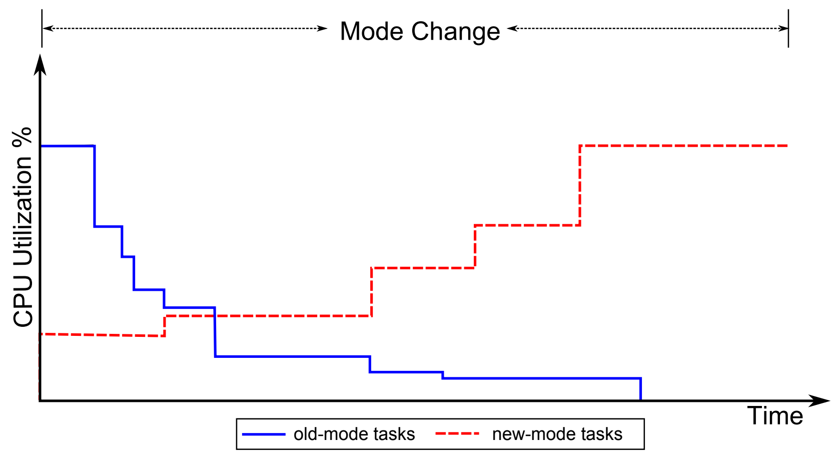


Figure 18 – Case Study 2 - Utilization Chart (AOF)

the new-mode tasks were shifted ahead with larger offsets. We can also observe in Table 20 the following values: $\max(R_N + O) = 360$ (τ_9) and $\max(R_O - x) = 289$ (τ_5). The latency value was $L = 360$ (from equation (2.18)) and $L \times 30\% = 180$. Therefore, the significant interval was $\delta = min(180, 289, 360) = 180$. The number of new-mode tasks completed during the significant interval was 0 and the number of old-mode tasks removed was 5, resulting in $\alpha = 0$. This value indicates an all-old-first (AOF) type of mode change (Fig. 18). This type of mode change occurred because all new-mode tasks were introduced with low priority at the beginning of the mode change (τ_6,τ_7,τ_8,τ_9), and have completed their execution after the significant interval.

4.3.3 Case Study 3 - Minimizing Latency Imposing Offsets within a Range (MOF)

Definition: In this case study the minimization was performed to find the smallest worst-case latency with the smallest possible sum of offsets possible for new-mode tasks while limiting the offsets within a range for some tasks.

Modeling: The minimization of latency and offsets was treated as a weights-based multi-objective minimization problem with two constraints: system feasibility and offsets within a range for some tasks. To achieve these objectives the fitness function was instantiated with the weights $We = \{1{,}0.0000001\}$, i.e., $fitness = \sum(O_i \times 0.0000001) + L \; \forall \; i \in \tau$. The constraints were set to $C_o = \{1, 1, 0, 0\}$, i.e., system feasibility and offsets within a range for some tasks. Table 21 shows the range of offsets used to perform the minimization in this case study.

Table 21 – Case Study 3 - Range for Offsets

Tasks	**Offset Minimum**	**Offset Maximum**
τ_2	366	1000
τ_3	400	600
τ_4	100	300

Results: Table 22 shows the summary of the minimization of the set of ten tasks using the fitness function set to minimize worst-case latency and offsets. This table is formatted with the same columns as table 19.

Table 22 – Case Study 3 - GAs Sensitivity Analysis

Scen.	**# of Analysis**	**Time (Min)**	**Valid Iter.**	**Latency**				**Sum of Offsets**	
				Low	**High**	**AVG**	**Var.**	**Low**	**High**
S_1	2801	0.04	30%	593	948	733	64.6%	2438	3482
S_2	5301	0.07	40%	538	783	623	39.9%	1805	3216
S_3	10601	0.13	100%	445	641	559	25.7%	1377	2409
S_4	20601	0.27	90%	445	557	521	17.0%	1069	1168
S_5	64001	0.84	90%	445	557	492	10.6%	868	1136
S_6	126501	1.73	90%	445	557	480	7.9%	869	1152
S_7	253001	3.50	100%	445	557	488	9.6%	868	992
S_8	503001	19.08	100%	445	557	506	13.8%	867	982
S_9	**1006001**	**14.61**	**100%**	**445**	**445**	**445**	**0.0%**	**866**	**880**
S_{10}	2006001	28.50	100%	445	557	456	2.5%	867	980

Sensitivity analysis: In scenario 1 only 30 % of repetitions were valid. Only after scenario five there were 90% of valid - and only in scenario seven they are all greater than 80% and mostly 100%. This case generates many unfeasible solutions because the constraints discard the chromosomes when the offsets are outside of the established range. A better approach is to modify the GA so that it only assign values belonging to the established range.

After minimization, the values of the best offsets obtained are shown in Table 23.

Table 23 – Case Study 3 - Offsets Obtained

Old-Mode Tasks M1 → M2					**New-Mode Tasks M1 → M2**				
Tasks	**T=D**	**R**	**x**	**Test**	**Tasks**	**T=D**	**R**	**O**	**Test**
τ_1	450	195	1	OK	τ_1	not active in this mode			
τ_2	not active in this mode				τ_2	100	25	367	OK
τ_3	300	145	1	OK	τ_3	150	45	400	OK
τ_4	200	45	1	OK	τ_4	200	15	100	OK
τ_5	500	270	101	OK	τ_5	300	165	0	OK
τ_6	400	185	1	OK	τ_6	400	185	0	OK
τ_7	100	25	0	OK	τ_7	450	220	0	OK
τ_8	250	85	1	OK	τ_8	500	275	0	OK
τ_9	not active in this mode				τ_9	600	365	0	OK
τ_{10}	600	490	301	OK	τ_{10}	not active in this mode			

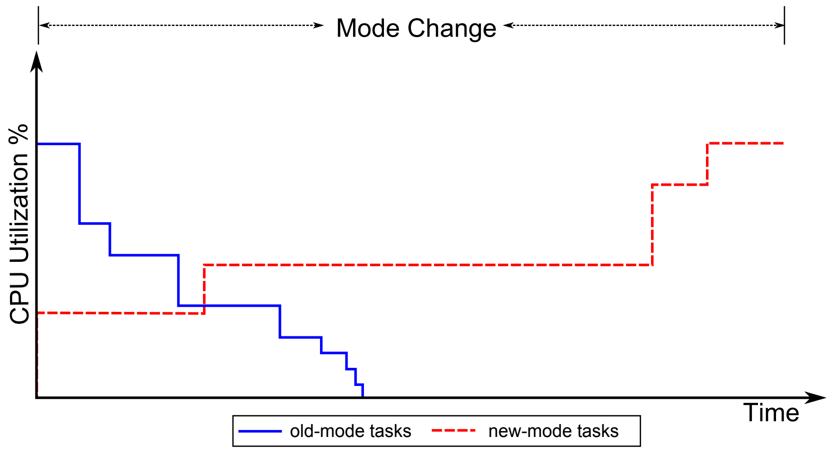


Figure 19 – Case Study 3 - Utilization Chart (MOF)

Discussion: Case study shows that is possible perform the minimization restricting the offsets within a range for some tasks. We can also observe in Table 23 the following values: $\max(R_N + O) = 445$ (τ_3) and $\max(R_O - x) = 289$ (τ_1). The latency value was $L = 445$ (from equation (2.18)) and $L \times 30\% = 133.5$. The significant interval was $\delta = min(133.5, 194, 445) = 133.5$. The number of new-mode tasks completed during the significant interval was 1 and the number of old-mode tasks removed was 3, resulting in $\alpha = 0.25$. This value indicates a mostly-old-first (MOF) type of mode change. Fig. 19 shows the processor utilization across the mode change, which assists in the interpretation of this type of mode change.

4.3.4 Case Study 4 - Minimizing Latency with WCRT within a Range (MOF)

Definition: In this case study the minimization was performed to find the smallest worst-case latency with the smallest sum of offsets possible for new-mode tasks, this time limiting the WCRT to a range of values for a subset of old and new-mode tasks.

Configuration of the GA: Twelve distinct scenarios S_1, $S_2 \dots S_{12}$ were created (Table 24), which allowed us to determine the best parameters to be used in the minimization process using GA. The mutation and crossover probabilities were fixed at 10% and 70% respectively, for all scenarios. Each scenario was subject to ten repetitions in order to validate the accuracy of the data obtained.

Table 24 – Case Study 4 - Scenarios for GA Minimization

Parameters	**Scenarios**											
	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}
Population	100	100	200	200	500	500	1000	1000	2000	2000	3000	3000
Generations	25	50	50	100	125	250	250	500	500	1000	750	1500

Modeling: The minimization of latency and offsets was treated as a weights-based multi-objective GA minimization problem with two constraints: system feasibility and WCRT within a range for some tasks. To achieve these objectives, the fitness function was instantiated with the weights $We = \{1{,}0.0000001\}$, i.e. $fitness = \sum(O_i \times 0.0000001) + L \ \forall \ i \in \tau$. The constraints used were $C_o = \{1, 0, 1, 0\}$, i.e. system feasibility and WCRT within a range for some tasks. Table 25 shows the range of WCRT used to perform the minimization in this study case.

Results: Table 26 shows the summary of the minimization of the set of ten tasks for each optimization scenario. This table is formatted with the same columns as table 19. The results evaluated after the minimization are presented in Table 27.

Table 25 – Case Study 4 - Range for WCRT

Tasks	**WCRT (old-mode)**		**WCRT (new-mode)**	
	Minimum	**Maximum**	**Minimum**	**Maximum**
τ_1	250	300	not defined	
τ_5	300	400	200	300
τ_8	not defined		300	400

Table 26 – Case Study 4 - GAs Sensitivity Analysis

Scen.	**# of Analysis**	**Time (Min)**	**Valid Iter.**	**Latency**				**Sum of Offsets**	
				Low	**High**	**AVG**	**Var.**	**Low**	**High**
S_1	2801	0.03	0%	-	-	-	-	-	-
S_2	5301	0.06	0%	-	-	-	-	-	-
S_3	10601	0.13	20%	513	520	517	14.8%	742	1481
S_4	20601	0.27	20%	473	537	505	12.2%	738	563
S_5	64001	0.87	20%	510	520	515	14.4%	962	759
S_6	126501	1.76	0%	-	-	-	-	-	-
S_7	253001	3.50	30%	540	653	593	31.9%	1127	786
S_8	503001	7.48	60%	480	571	522	16.0%	520	780
S_9	**1006001**	**15.11**	**100%**	**450**	**557**	**508**	**13.0%**	**482**	**769**
S_{10}	2006001	43.98	70%	451	545	516	14.6%	484	901
S_{11}	2259001	34.25	70%	457	570	509	13.1%	489	1085
S_{12}	4509001	69.01	60%	457	540	497	10.5%	486	896

Sensitive analysis: Results were satisfactory from beyond 10 cases (S9), where 100% of repetitions were valid.

Table 27 – Case Study 4 - Offsets Obtained

Old-Mode Tasks M1 → M2				
Tasks	**T=D**	**R**	**x**	**Test**
τ_1	450	265	1	OK
τ_2	not active in this mode			
τ_3	300	190	101	OK
τ_4	200	45	1	OK
τ_5	500	355	1	OK
τ_6	400	230	101	OK
τ_7	100	25	0	OK
τ_8	250	105	1	OK
τ_9	not active in this mode			
τ_{10}	600	560	301	OK

New-Mode Tasks M1 → M2				
Tasks	**T=D**	**R**	**O**	**Test**
τ_1	not active in this mode			
τ_2	100	25	432	OK
τ_3	150	65	0	OK
τ_4	200	108	27	OK
τ_5	300	205	0	OK
τ_6	400	198	27	OK
τ_7	450	290	0	OK
τ_8	500	385	0	OK
τ_9	600	425	0	OK
τ_{10}	not active in this mode			

Discussion: Case study 4, shows that is possible restrict the WCTR for some tasks within a range. We can also observe in Table 27 the following values: $\max(R_N + O) = 457$ (τ_2) and $\max(R_O - x) = 354$ (τ_5). The latency value was $L = 457$ (from equation (2.18)) and $L \times 30\% = 137.1$. The significant interval calculated was $\delta = min(137.1, 354, 457) = 137.1$.

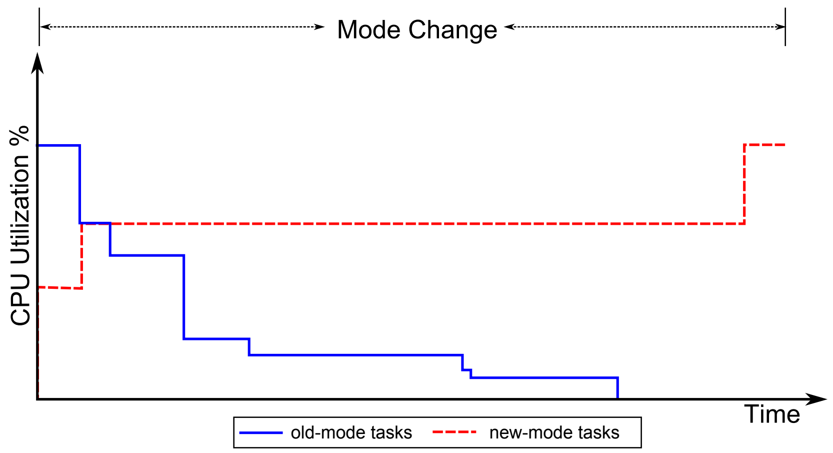


Figure 20 – Case Study 4 - Utilization Chart (MOF)

The number of new-mode tasks completed during the significant interval was 2 and the number of old-mode tasks removed was 5, resulting in $\alpha = 0.29$. This value of indicates a mostly-old-first (MOF) type of mode change (Fig. 20).

4.3.5 Case Study 5 - Minimizing Offsets with Latency within an Acceptable Range (MOF)

Definition: In this case study the goal was the minimization of offsets, limiting the worst-case latency within an acceptable range.

Modeling: The minimization of offsets and latency was treated as a weights-based multi-objective GA minimization problem with two constraints: system feasibility and worst-case latency within an acceptable range. Therefore, the constraint vector was set to $C_o = \{1, 0, 0, 1\}$. To achieve these objectives the fitness function was instantiated with the weights $We = \{0.0000001{,}1\}$, i.e., $fitness = \sum O_i + (L \times 0.0000001)\ \forall\ i \in \tau$. The latency range used in the minimization was arbitrarily set from 400 to 450.

Results: Table 28 shows the GAs sensitivity analysis of each scenario submitted to the minimization process using GA. In this table, the columns 1-9 represent respectively: 1) the scenario used for minimization, 2) the average number of tests processed during minimization, 3) the average time spent on each repetition of the scenario, 4) the lowest, 5) highest and 6) average sum of offsets, 7) the variation of the average in relation to to lowest value of sum of offsets, 8) the lowest latency for lowest, and 9) high worst-case sum of offsets. After minimization, the results obtained are shown in Table 29.

Table 28 – Case Study 5 - GAs Sensitivity Analysis

Scen.	**# of Analysis**	**Time (Min)**	**Valid Iter.**	**Sum of Offsets**				**Latency**	
				Low	**High**	**AVG**	**Var.**	**Low**	**High**
S_1	2801	0.04	60%	957	1490	1255	178.9%	425	450
S_2	5301	0.09	70%	534	832	667	48.2%	450	450
S_3	10601	0.17	90%	541	736	629	39.8%	450	405
S_4	20601	0.33	90%	452	723	574	27.5%	450	410
S_5	64001	1.01	100%	452	677	534	18.7%	450	405
S_6	126501	2.13	100%	452	560	480	6.7%	450	445
S_7	253001	4.24	100%	450	536	468	3.9%	450	445
S_8	503001	8.52	100%	450	549	489	8.6%	450	445
S_9	**1006001**	**17.07**	**100%**	**450**	**452**	**451**	**0.3%**	**450**	**450**
S_{10}	2006001	51.76	100%	450	475	454	0.9%	450	450

Sensitivity analysis: From S_9 onwards we notice that the algorithm presents more stable results, i.e., variance around 0.3 %. A population around 2000 individuals is the recommended value for this case.

Table 29 – Case Study 5 - Offsets Obtained

Old-Mode Tasks M1 → M2				
Tasks	**T=D**	**R**	**x**	**Test**
τ_1	450	240	101	OK
τ_2	not active in this mode			
τ_3	300	190	101	OK
τ_4	200	45	1	OK
τ_5	500	380	1	OK
τ_6	400	230	101	OK
τ_7	100	25	0	OK
τ_8	250	105	1	OK
τ_9	not active in this mode			
τ_{10}	600	530	301	OK

New-Mode Tasks M1 → M2				
Tasks	**T=D**	**R**	**O**	**Test**
τ_1	not active in this mode			
τ_2	100	25	350	OK
τ_3	150	65	0	OK
τ_4	200	85	50	OK
τ_5	300	205	0	OK
τ_6	400	175	50	OK
τ_7	450	290	0	OK
τ_8	500	410	0	OK
τ_9	600	450	0	OK
τ_{10}	not active in this mode			

Discussion: Case study 5 shows that it is possible to restrict the worst-case latency within an acceptable range. We can also observe in Table 29 the following values: $\max(R_N + O) = 450$ (τ_2) and $\max(R_O - x) = 379$ (τ_5). The latency value was $L = 450$ (from equation (2.18)) and $L \times 30\% = 135$. The significant interval was $\delta = min(135, 379, 450) = 135$. The number of new-mode tasks completed during the significant interval was 2 and the number of old-mode tasks removed was 5, resulting in $\alpha = 0.29$. This value indicates a mostly-old-first (MOF) type of mode change (Fig. 21).

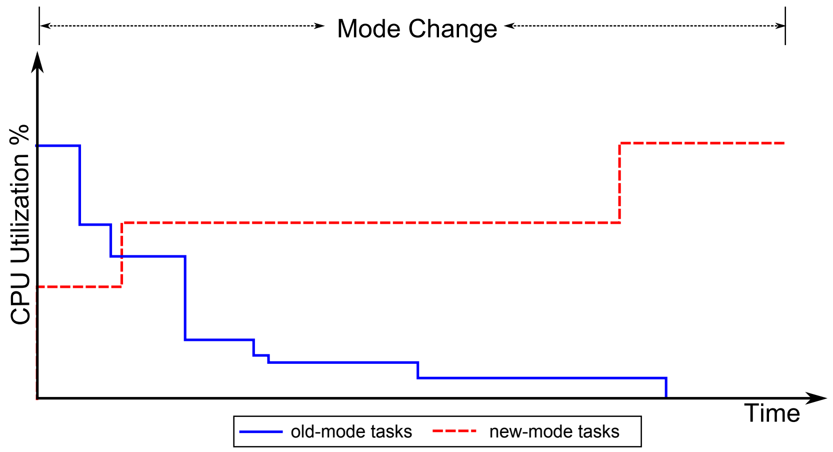


Figure 21 – Case Study 5 - Utilization Chart (MOF)

4.3.6 Case Study 6 - Minimization Using the Algorithm of Real and Crespo (2001) (BMC)

Definition: To provide a comparison analysis, in this case study the minimization was performed to find the smallest offsets for high priority tasks using the algorithm of Real and Crespo (2001).

After minimization, the results obtained are presented in Table 30. The processing time was 0.33 minutes, the sum of offsets was 1602, the worst-case latency was 754 and the number of analysis processed was 31173.

Table 30 – Case Study 6 - Offsets Obtained

Old-Mode Tasks M1 → M2				
Tasks	**T=D**	**R**	**x**	**Test**
τ_1	450	385	1	OK
τ_2	not active in this mode			
τ_3	300	290	1	OK
τ_4	200	70	1	OK
τ_5	500	485	1	OK
τ_6	400	375	1	OK
τ_7	100	25	0	OK
τ_8	250	175	1	OK
τ_9	not active in this mode			
τ_{10}	600	580	1	OK

New-Mode Tasks M1 → M2				
Tasks	**T=D**	**R**	**O**	**Test**
τ_1	not active in this mode			
τ_2	100	50	0	OK
τ_3	150	90	0	OK
τ_4	200	200	30	OK
τ_5	300	171	184	OK
τ_6	400	290	85	OK
τ_7	450	185	579	OK
τ_8	500	26	579	OK
τ_9	600	600	145	OK
τ_{10}	not active in this mode			

Discussion: In this case study, the sum of offsets and the worst-case latency obtained are both larger than those obtained in cases previously showed. The sum of offsets was 1602 and the latency 764. In case study 1 the sum of offsets was 390 and the latency was 595. In case study 2 the sum of offsets was 690 and the latency was 360. We can also observe in Table

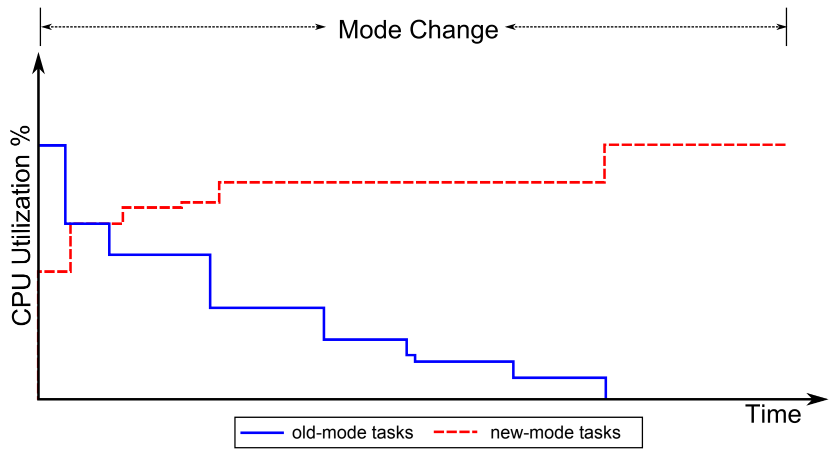


Figure 22 – Case Study 6 - Utilization Chart (BMC)

30 the following values: $\max(R_N + O) = 764$ (τ_7), and $\max(R_O - x) = 579$ (τ_{10}). The latency value was $L = 764$ (from equation (2.18)) and $L \times 30\% = 229.2$. The significant interval was $\delta = min(229.2, 579, 764) = 229.2$. The number of new-mode tasks completed during the significant interval was 2 and the number of old-mode tasks removed was 3, resulting in $\alpha = 0.4$. This value indicates a balanced type of mode change (BMC) (Fig. 22). This type of mode-change occurs because the minimization using the algorithm of Real and Crespo (2001) assigns small offsets for higher priority tasks and the relative lower utilization rate of old-mode tasks allows that new-mode tasks run in (pseudo) parallel with the old-mode tasks.

4.3.7 Case Study 7 - Minimizing Latency with $P_N > P_O$ (MNF)

Definition: This case study uses the same set of ten tasks used by case studies previously presented. We have decreased the priority of the all old-mode tasks (P_O) and maintained the priority of the new-mode ones (P_N). Therefore, P_O=$\{11, 12, 13, 14, 15, 16, 17, 18\}$ for tasks $\{\tau_2, \tau_3, \tau_4, \tau_5, \tau_6, \tau_7, \tau_8, \tau_9\}$. Applying this configuration, we seek to find the smallest possible worst-case latency with the smallest sum of offsets for new-mode tasks. This is carried out for both definitions of mode-change latency described above.

GA Configuration: Based on the sensibility analysis performed in the studies cases above, we assigned the value 2000 to the population and ran 1000 generations. The mutation and crossover probabilities were fixed at 10% and 70% respectively, for all scenarios. This scenario was subjected to ten repetitions in order to measure the accuracy of the data obtained. This GA configuration was used in all the remaining case studies in this chapter.

Modeling: The minimization of latency and offsets was treated as a weights-based multi-objective GA minimization problem with just one constraint, i.e. system feasibility. To achieve these objectives, the fitness function was instantiated with weights $We = \{1{,}0.0000001\}$. The fitness function was $fitness = \sum(O_i \times 0.0000001) + L \ \forall \ i \in \tau$. The constraints' vector was set to system feasibility, i.e. $C_o = \{1, 0, 0, 0\}$.

Minimization Using Latency Definition I

Table 31 shows the results of the minimization process using "latency definition I". The worst-case latency was 329 and the sum of offsets was 700.

Table 31 – Case Study 7 - Latency Definition I - Offsets Obtained

Old-Mode Tasks M1 → M2					New-Mode Tasks M1 → M2				
Tasks	**T=D**	**R**	**x**	**Test**	**Tasks**	**T=D**	**R**	**O**	**Test**
τ_1	450	305	1	OK	τ_1	not active in this mode			
τ_2	not active in this mode				τ_2	100	25	274	OK
τ_3	300	250	1	OK	τ_3	150	45	192	OK
τ_4	200	130	1	OK	τ_4	200	75	129	OK
τ_5	500	330	1	OK	τ_5	300	95	35	OK
τ_6	400	275	101	OK	τ_6	400	20	0	OK
τ_7	100	35	0	OK	τ_7	450	185	35	OK
τ_8	250	220	1	OK	τ_8	500	270	35	OK
τ_9	not active in this mode				τ_9	600	10	0	OK
τ_{10}	600	520	301	OK	τ_{10}	not active in this mode			

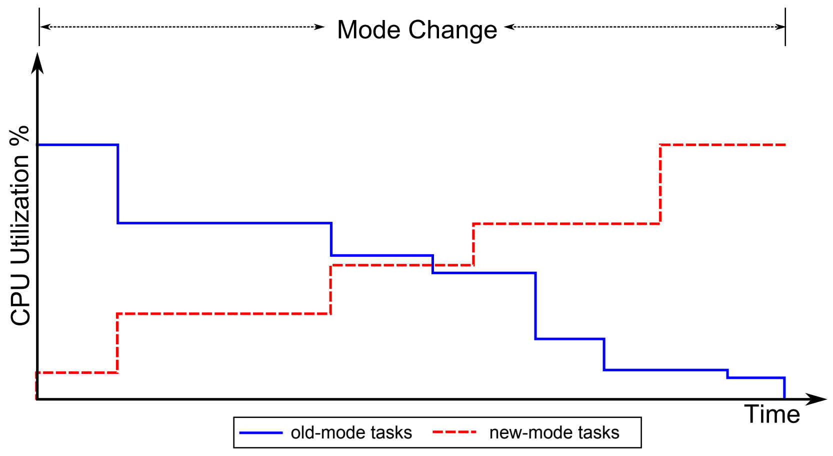


Figure 23 – Case Study 7 - Utilization Chart Using Latency Definition I (MNF)

Discussion: We can observe in Table 31 the following values: $\max(R_N + O) = 305$ (τ_8), and $\max(R_O - x) = 329$ (τ_5). The latency value was $L = 329$ (from equation (2.18)) and

$L \times 30\% = 98.7$. The significant interval was $\delta = min(98.7, 329, 305) = 98.7$. The number of new-mode tasks completed during the significant interval was 2 and the number of old-mode tasks removed was 1, resulting in $\alpha = 0.67$. This value of indicates a mostly-new-first type (MNF) of mode change. Fig. 23 shows a visual representation that allows the interpretation of the type of mode change. This type of mode change has occurred due to the fact that the priority of the new-mode tasks were higher than the old-mode tasks, leading to the preemption (and delaying) of old-mode tasks.

Minimization Using Latency Definition II

Table 32 shows the results after the minimization process using "latency definition II ". The worst-case latency obtained was 224 and the sum of offsets was 652. Using "latency definition I", the worst-case latency calculated was 479.

Table 32 – Case Study 7 - Latency Definition II - Offsets Obtained

Old-Mode Tasks M1 → M2				
Tasks	**T=D**	**R**	**x**	**Test**
τ_1	450	380	1	OK
τ_2	not active in this mode			
τ_3	300	275	1	OK
τ_4	200	160	1	OK
τ_5	500	450	1	OK
τ_6	400	370	1	OK
τ_7	100	90	0	OK
τ_8	250	245	1	OK
τ_9	not active in this mode			
τ_{10}	600	480	1	OK

New-Mode Tasks M1 → M2				
Tasks	**T=D**	**R**	**O**	**Test**
τ_1	not active in this mode			
τ_2	100	25	179	OK
τ_3	150	45	179	OK
τ_4	200	75	91	OK
τ_5	300	95	91	OK
τ_6	400	140	80	OK
τ_7	450	185	32	OK
τ_8	500	30	0	OK
τ_9	600	65	0	OK
τ_{10}	not active in this mode			

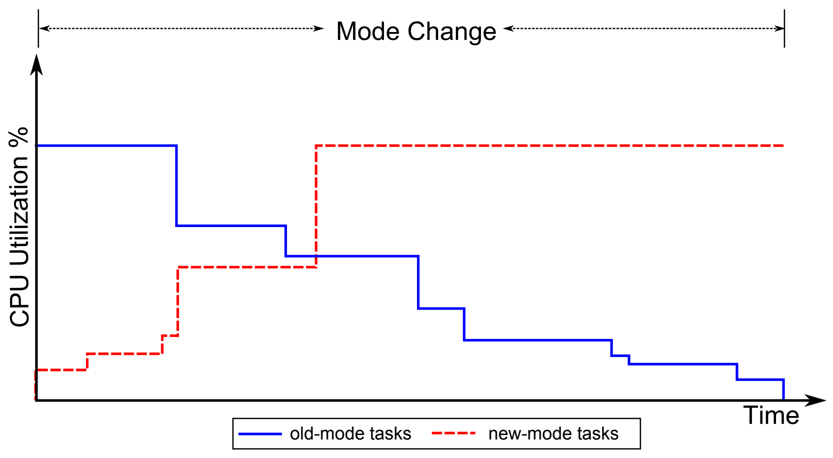


Figure 24 – Case Study 7 - Utilization Chart Using Latency Definition II (MNF)

Discussion: We can observe in Table 32 the following values: $\max(R_N + O) = 224$ (τ_3), and $\max(R_O - x) = 479$ (τ_{10}). The latency value is $L = 479$ (from equation (2.18)) and $L \times 30\% = 143.7$, thus, the significant interval is $\delta = min(143.7, 479, 224) = 143.7$. The number of new-mode tasks completed during the significant interval was 2 and the number of old-mode tasks removed was 1, resulting in $\alpha = 0.67$. This value indicates a mostly-new-first type (MNF) mode change. Fig. 24 shows a visual representation that assists in the interpretation of the type of mode change. This type of mode change has occurred due to the priority of the new-mode tasks being higher than the old-mode tasks and the minimization process using "latency definition II". Thus, the old-mode tasks are preempted, slowing their execution. Note: The value assigned to L used "latency definition I".

4.3.8 Case Study 8 - Minimizing Offsets with $P_N > P_O$ (MNF)

Definition: This case study uses the same set of tasks employed in *Case Study 7*. The major goal was to perform the minimization of offsets for new-mode tasks; the secondary goal was to minimize the worst-case latency. We employed both mode-change latency definitions.

Modeling: The minimization of offsets and latency was treated as a weights-based multi-objective GA minimization problem and system feasibility as the constraint. To achieve these objectives, the fitness function was instantiated with the weights $We = \{0.0000001, 1\}$, i.e. $fitness = \sum O_i + (L \times 0.0000001)\ \forall\ i \in \tau$. The constraints used were $C_o = \{1, 0, 0, 0\}$.

Table 33 – Case Study 8 - Offsets Obtained

Old-Mode Tasks M1 → M2				
Tasks	**T=D**	**R**	**x**	**Test**
τ_1	450	395	1	OK
τ_2	not active in this mode			
τ_3	300	295	101	OK
τ_4	200	200	1	OK
τ_5	500	465	1	OK
τ_6	400	315	101	OK
τ_7	100	100	0	OK
τ_8	250	240	1	OK
τ_9	not active in this mode			
τ_{10}	600	595	1	OK

New-Mode Tasks M1 → M2				
Tasks	**T=D**	**R**	**O**	**Test**
τ_1	not active in this mode			
τ_2	100	25	294	OK
τ_3	150	6	14	OK
τ_4	200	75	100	OK
τ_5	300	40	0	OK
τ_6	400	60	0	OK
τ_7	450	51	14	OK
τ_8	500	270	100	OK
τ_9	600	75	0	OK
τ_{10}	not active in this mode			

Results: Table 33 shows the results obtained after the minimization process. The worst-case latencies found were 594 and 370 according to "definitions I and II" respectively. The sum of offsets resulted in 522 (conforming to both definitions).

Fig. 25 shows a chart of the processor utilization rate during the mode-change period.

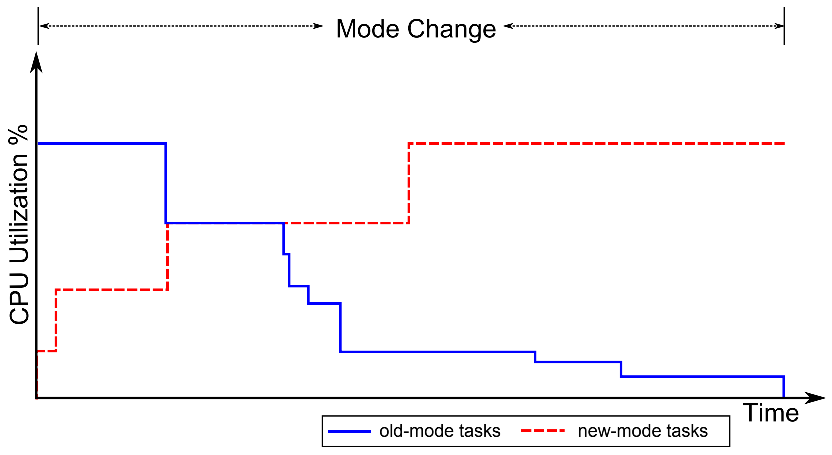


Figure 25 – Case Study 8 - Utilization Chart (MNF)

Discussion: We can observe in Table 33 the following values: $\max(R_N + O) = 370$ (τ_8), and $\max(R_O - x) = 594$ (τ_{10}). The latency value is $L = 594$ (from equation (2.18)) and $L \times 30\% = 178.2$. This leads to a significant interval $\delta = min(178.2, 594, 370) = 178.2$. The number of new-mode tasks completed during the significant interval was 6 and the number of old-mode tasks removed was 1, resulting in $\alpha = 0.86$. This value indicates a mostly-new-first (MNF) type of mode change (Fig. 25). This type of mode change has occurred due to the fact that the priority of the new-mode tasks were higher than the old-mode tasks and the main objective is to minimize offsets. This causes the first few new-mode tasks with higher priorities to have shorter offsets. At the end of the busy period of this small task set old-mode tasks execute and complete.

For a small new-mode task set and under the conditions set (i.e. $P_N > P_O$), the behavior of the GA resembles that of Real and Crespo (2001) since 1) both algorithms are configured to minimize offsets and 2) new-mode tasks have higher priorities than old-mode tasks.

4.3.9 Case Study 9 - Minimizing Latency with $P_O > P_N$ (AOF)

Definition: This case study uses the same set of ten tasks used by case studies previously presented. The goal is to configure an AOF type of mode change with minimum latency. Offsets are minimized as a secondary objective. For that purpose, we decreased the priority of the all new mode tasks (P_N) and maintained the priority of the old-mode set (P_O). Therefore, P_N=$\{16, 14, 12, 17, 15, 11, 13, 18\}$ for tasks $\{\tau_1, \tau_3, \tau_4, \tau_5, \tau_6, \tau_7, \tau_8, \tau_{10}\}$.

Modeling: The minimization of latency and offsets was treated as a weights-based multi-objective GA minimization problem. To achieve these objectives, the fitness function was instantiated with the weights $We = \{1{,}0.0000001\}$, i.e. $fitness = \sum(O_i \times 0.0000001) + L \; \forall \; i \in \tau$. The only constraint was system feasibility, therefore $C_o = \{1, 0, 0, 0\}$.

Results: Table 34 shows the results of the minimization process. The worst-case latency found (for both latency definitions) was 350, and the sum of offsets was 990. Fig. 26 shows a chart of the processor utilization across the mode-change period.

Table 34 – Case Study 9 - Offsets Obtained

Old-Mode Tasks M1 → M2				
Tasks	**T=D**	**R**	**x**	**Test**
τ_1	450	170	101	OK
τ_2	not active in this mode			
τ_3	300	140	101	OK
τ_4	200	45	1	OK
τ_5	500	195	101	OK
τ_6	400	160	101	OK
τ_7	100	25	0	OK
τ_8	250	85	1	OK
τ_9	not active in this mode			
τ_{10}	600	365	301	OK

New-Mode Tasks M1 → M2				
Tasks	**T=D**	**R**	**O**	**Test**
τ_1	not active in this mode			
τ_2	100	25	250	OK
τ_3	150	20	200	OK
τ_4	200	90	160	OK
τ_5	300	245	50	OK
τ_6	400	315	0	OK
τ_7	450	320	0	OK
τ_8	500	20	330	OK
τ_9	600	330	0	OK
τ_{10}	not active in this mode			

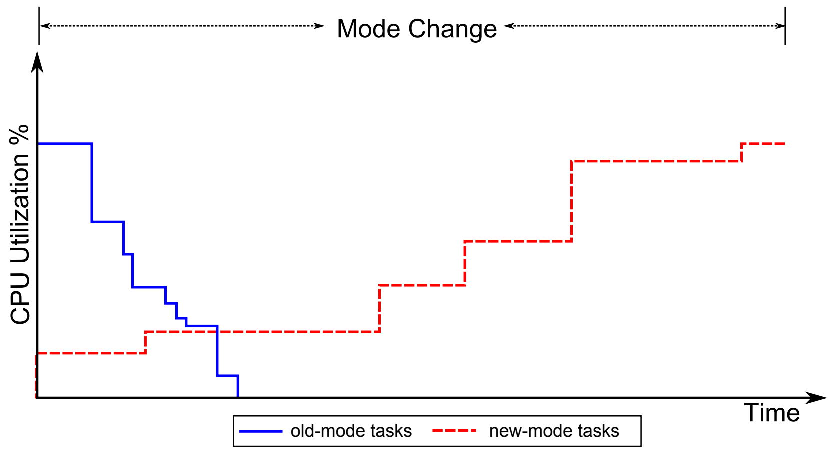


Figure 26 – Case Study 9 - Utilization Chart (AOF)

Discussion: We can observe in Table 34 the following values: $\max(R_N + O) = 350$ (τ_8), and $\max(R_O - x) = 94$ (τ_5). The latency value was $L = 350$ (from equation (2.18)) and $L \times 30\% = 105$. Thus, the significant interval was $\delta = min(105, 94, 350) = 94$. The number of new-mode tasks completed during the significant interval was 0 and the number of old-mode tasks removed was 8, resulting in $\alpha = 0$. This value indicates an all-old-first (AOF) type of mode change. Fig. 26 shows a visual representation that assists in the interpretation of the type of mode change. This type of mode change has occurred due to the fact that the priority of the old-mode tasks were higher that their new-mode counterparts and relatively large offsets were assigned to the new-mode set, thus delaying their completion. Note that, in this case study, the value assigned to δ was $\max(R_O - x)$.

4.3.10 Case Study 10 - Minimizing Offsets with $P_O > P_N$ (AOF)

Definition: This case study uses the same set of tasks used by *Case Study 9.* The aim of this case was to produce an AOF with all offsets minimized (as a first objective) and latency minimized as a second objective. In order to do so, we set all old-mode priorities larger than the priorities of the new-mode (i.e $P_O > P_N$).

Modeling: The minimization of offsets and latency was treated as a weights-based multi-objective GA minimization problem with just one constraint, system feasibility. To achieve these objectives the fitness function was instantiated with the weights $We = \{0.0000001{,}1\}$, i.e., $fitness = \sum O_i + (L \times 0.0000001)\ \forall\ i \in \tau$, thus, the main objective is offsets minimization and the secondary is latency minimization. The constraints used were $C_o = \{1, 0, 0, 0\}$, i.e, just system feasibility.

Results: Table 35 shows the values of the best obtained after the minimization process. The worst-case latency obtained using both latency definitions was 495, and the sum of offsets was 520.

Table 35 – Case Study 10 - Offsets Obtained

Old-Mode Tasks M1 → M2				
Tasks	**T=D**	**R**	**x**	**Test**
τ_1	450	170	101	OK
τ_2	not active in this mode			
τ_3	300	140	101	OK
τ_4	200	45	1	OK
τ_5	500	195	101	OK
τ_6	400	160	101	OK
τ_7	100	25	0	OK
τ_8	250	85	1	OK
τ_9	not active in this mode			
τ_{10}	600	365	301	OK

New-Mode Tasks M1 → M2				
Tasks	**T=D**	**R**	**O**	**Test**
τ_1	not active in this mode			
τ_2	100	25	295	OK
τ_3	150	135	85	OK
τ_4	200	175	95	OK
τ_5	300	290	0	OK
τ_6	400	385	0	OK
τ_7	450	410	45	OK
τ_8	500	485	0	OK
τ_9	600	495	0	OK
τ_{10}	not active in this mode			

Fig. 27 shows a chart of the processor utilization rate during the mode-change period.

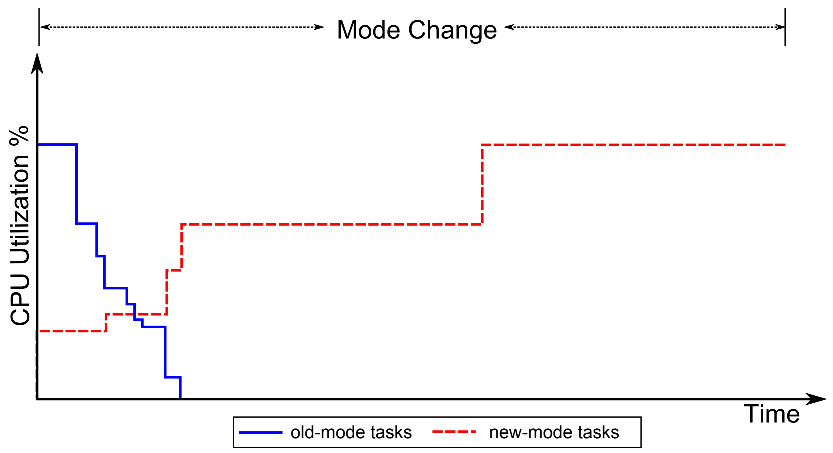


Figure 27 – Case Study 10 - Utilization Chart (AOF)

Discussion: In this case the worst-case latency value independs on the definition used, i.e. the values of latency found using definition I was the same value obtained using definition II. This is because old-mode tasks complete their execution before the new ones complete their execution. Comparing it with the case study 9, there was an increase of the latency from 350 to 495 and a substantial decrease of the sum of offsets from 990 to 520. Therefore, the algorithm delivered the results specified for this case.

The inspection of Table 35 shows the following values: $\max(R_N + O) = 495$ (τ_9) and $\max(R_O - x) = 94$ (τ_5). As the latency value is $L = 495$ (from equation (2.18)), then $L \times 30\% = 148.5$ and the significant interval is $\delta = min(148.5, 94, 495) = 94$. The number of new-mode tasks completed during the significant interval was 0 and the number of old-mode tasks removed was 8, resulting in $\alpha = 0$. This value of indicates an all-old-first (AOF) type of mode change (Fig. 27). Therefore, we can assert that when the priorities of old-mode tasks are higher that the new-mode tasks there is a tendency that, in most of cases, δ is equal to $\max(R_O - x)$. Clearly, the exception is when $\delta = L \times K$.

4.3.11 Case Study 11 - Minimizing Latency with $O > \max(R_O - x)$ (AOF)

Definition: The goal of this case study is to configure an AOF type of mode change. This case study uses the same task set used in *Case Study 1*. This type of mode change is accomplished by having the new-mode tasks starting after all the old-mode ones have completed. This last condition is enforced when the offsets of all new-mode tasks are larger or equal to maximum $WCRT$ of all old-mode tasks, i.e. $O > \max(R_O - x)$.

Modeling: The minimization of latency and offsets was treated as a weights-based multi-objective GA minimization problem and system feasibility as the sole constraint. To achieve these objectives the fitness function was instantiated with the weights $We = \{1{,}0.0000001\}$, i.e., $fitness = \sum(O_i \times 0.0000001) + L \ \forall \ i \in \tau$. Therefore, the main objective is latency minimization and the secondary one is offsets minimization. The allow feasibility as the only constraint, the constraint vector was set to $C_o = \{1, 0, 0, 0\}$.

Results: Table 36 shows the main results for this case study. The worst-case latency obtained (according to both latency definitions) was 350. The sum of offsets obtained was 1347.

Table 36 – Case Study 11 - Offsets Obtained

Old-Mode Tasks M1 → M2				
Tasks	**T=D**	**R**	**x**	**Test**
τ_1	450	170	101	OK
τ_2	not active in this mode			
τ_3	300	140	101	OK
τ_4	200	45	1	OK
τ_5	500	195	101	OK
τ_6	400	160	101	OK
τ_7	100	25	0	OK
τ_8	250	85	1	OK
τ_9	not active in this mode			
τ_{10}	600	365	301	OK

New-Mode Tasks M1 → M2				
Tasks	**T=D**	**R**	**O**	**Test**
τ_1	not active in this mode			
τ_2	100	25	325	OK
τ_3	150	12	208	OK
τ_4	200	125	125	OK
τ_5	300	174	96	OK
τ_6	400	194	96	OK
τ_7	450	199	96	OK
τ_8	500	20	305	OK
τ_9	600	209	96	OK
τ_{10}	not active in this mode			

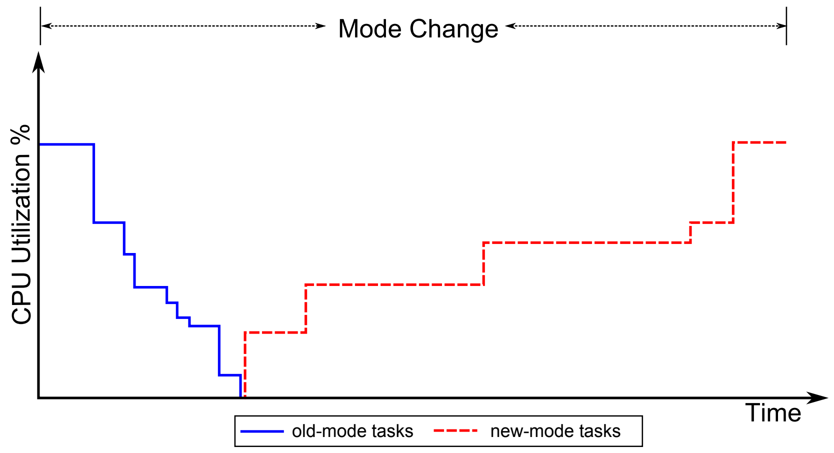


Figure 28 – Case Study 11 - Utilization Chart (AOF)

Fig. 28 shows a chart of the processor utilization rate during the mode-change period.

Discussion: This type of mode change is called of synchronous mode change. In this case, latency was relatively low (350) since many old-mode tasks have completed while still in the old-mode (i.e. before the MCR), which is indicated by variable x having larger values (e.g. t $\{\tau_1, \tau_3, \tau_5, \tau_6, \tau_{10}\} = \{101, 101, 101, 101, 301\}$). We can observe also in Table 36 the following values: $\max(R_N + O) = 350$ (τ_2), and $\max(R_O - x) = 94$ (τ_5). The latency value was $L = 350$ (from equation (2.18)) and $L \times 30\% = 105$. The significant interval was therefore $\delta = min(105, 94, 350) = 94$. Hence, the number of new-mode tasks introduced during the significant interval was 0 and the number of old-mode tasks removed was 8, resulting in $\alpha = 0$ (the desired mode change AOF) (Fig. 28).

4.3.12 Case Study 12 - Minimizing Latency with $P_N > P_O$ (ANF)

Definition: In this case study the minimization was performed with the task set described in Table 37. Each mode of operation $M1$ (old-mode) and $M2$ (new-mode) is composed of 10 tasks ($\tau_1 \ldots \tau_{10}$). The CPU utilization rate is 41.61% in $M1$ and 78.22% in $M2$.

The goal of this case study is to perform an ANF type of mode change. Therefore, the following changes were applied to the task set presented in case 11: 1) we decreased the priority of all old-mode tasks (P_O), and 2) we increased the deadlines and periods of the old-mode tasks, while maintaining the deadline monotonic policy for priority assignment.

These changes allow the preemption of the old-mode tasks during the introduction of new-mode tasks. After applying this configuration, we performed the minimization of offsets to find the smallest worst-case latency with the smallest sum of offsets for new-mode tasks possible, using the latency definition II and a weights-based multi-objective GA.

Table 37 – Case Study 12 - Set of Ten Tasks

Mode M1						
Tasks	**P**	**C**	**T**	**T=D**	**R**	**Test**
$\tau_{1(O)}$	16	10	900	900	145	OK
$\tau_{2(W)}$	not active in this mode					
$\tau_{3(C)}$	14	30	600	600	115	OK
$\tau_{4(C)}$	12	20	400	400	45	
$\tau_{5(C)}$	17	25	1000	1000	170	OK
$\tau_{6(U)}$	15	20	400	400	135	OK
$\tau_{7(C)}$	11	25	200	200	25	OK
$\tau_{8(C)}$	13	40	500	500	85	OK
$\tau_{9(W)}$	not active in this mode					
$\tau_{10(O)}$	18	30	1200	1200	200	OK

Mode M2						
Tasks	**P**	**C**	**T**	**T=D**	**R**	**Test**
$\tau_{1(O)}$	not active in this mode					
$\tau_{2(W)}$	1	25	100	100	25	OK
$\tau_{3(C)}$	2	20	150	150	45	OK
$\tau_{4(C)}$	3	30	200	200	75	OK
$\tau_{5(C)}$	4	20	300	300	95	OK
$\tau_{6(U)}$	5	20	400	400	140	OK
$\tau_{7(C)}$	6	25	450	450	185	OK
$\tau_{8(C)}$	7	30	500	500	270	OK
$\tau_{9(W)}$	8	10	600	600	280	OK
$\tau_{10(O)}$	not active in this mode					

Modeling: The minimization of latency offsets was treated as a weights-based multi-objective GA minimization problem with just one constraint, i.e. system feasibility. To achieve these objectives, the fitness function was instantiated with weights $We = \{1{,}0.0000001\}$, i.e., $fitness = \sum(O_i \times 0.0000001) + L \; \forall \; i \in \tau$. Thus, the main objective is latency minimization and the secondary is offsets minimization. The constraints used were $C_o = \{1, 0, 0, 0\}$, i.e, just system feasibility.

Table 38 shows the results (i.e. R, L, O, x) after the minimization process using Latency Definition II. The worst-case latency measured was 329, and the sum of offsets was 700.

Table 38 – Case Study 12 - Offsets Obtained Using Latency Definition II

Old-Mode Tasks M1 → M2				
Tasks	**T=D**	**R**	**x**	**Test**
τ_1	900	685	1	OK
τ_2	not active in this mode			
τ_3	600	365	1	OK
τ_4	400	175	1	OK
τ_5	1000	755	1	OK
τ_6	400	385	1	OK
τ_7	200	155	0	OK
τ_8	500	335	1	OK
τ_9	not active in this mode			
τ_{10}	1200	785	1	OK

New-Mode Tasks M1 → M2				
Tasks	**T=D**	**R**	**O**	**Test**
τ_1	not active in this mode			
τ_2	100	25	84	OK
τ_3	150	45	84	OK
τ_4	200	30	0	OK
τ_5	300	50	0	OK
τ_6	400	70	0	OK
τ_7	450	75	0	OK
τ_8	500	150	0	OK
τ_9	600	130	0	OK
τ_{10}	not active in this mode			

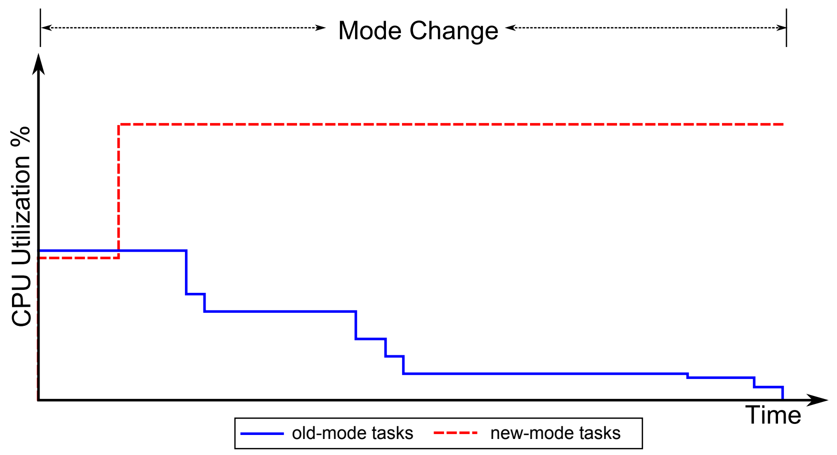


Figure 29 – Case Study 12 - Utilization Chart Using Latency Definition II

Fig. 29 shows a chart of the processor utilization rate during the mode-change period.

Discussion: Case study 12 was performed to force a specific mode-change, i.e., all-new -first (ANF). It is interesting to note that this particular mode-change type can only be minimized when using “latency definition II”, since the end of the transition is marked by the first execution of the all new-mode tasks. Therefore, to achieve low latency, the algorithm pushes all new mode tasks to the start of the transition, ahead of the old-mode tasks.

As the introduction of the new-mode tasks with higher priority and short offsets could lead an system overload (and therefore missed deadlines), we forced a lower utilization of the old-mode task set by both increasing old-mode tasks periods and deadlines.

We can also observe in Table 38 the following values: $\max(R_N + O) = 150$ (τ_8), and $\max(R_O - x) = 784$ (τ_{10}). The latency value is $L = 784$ (from equation (2.18)) and $L \times 30\% = 235.2$. Therefore, the significant interval is $\delta = min(235.2, 784, 150) = 150$. The number of new-mode tasks completed during the significant interval was 8 and the number of old-mode tasks removed was 0, resulting in $\alpha = 8 \div (0+8) = 1$, i.e. an ANF mode change. Fig. 29 shows a visual representation that assists in the interpretation of the type of mode change.

4.4 Discussion

In this section we present a global discussion of the results for all case studies, which are summarized in Table 39. This table describes for each case study, the definition of latency used in the minimization process, the type of mode change, the worst-case latency using both definitions , the sum of offsets, the significant interval δ and the rate of change parameter α.

Table 39 – Summary of Case Studies

#	**Case Study**	**LD for Minim.**	**Type**	**LD I**	**LD II**	**Sum Offsets**	δ	α
1	Minimizing Offsets and Latency	LD I	MOF	595		390	178.5	0.33
2	Minimizing Latency and Offsets	LD I	AOF	360		690	180	0
3	Minimizing Latency Imposing Offsets within a Range	LD I	MOF	445		866	133.5	0.25
4	Minimizing Latency with WCRT within a range	LD I	MOF	457		486	137.1	0.29
5	Minimizing Offsets with Latency within an Acceptable Range	LD I	MOF	450		450	135	0.29
6	Minimization Using the Algorithm of Real and Crespo (2001)	LD I	BMC	764		1602	229.2	0.4
7	Minimizing Latency with $P_N > P_O$	LD I	MNF	329	305	700	98.7	0.83
		LD II	MNF	479	224	652	143.7	0.67
8	Minimizing Offsets with $P_N > P_O$	LD I, II	MNF	594	370	522	178.2	0.86
9	Minimizing Latency with $P_O > P_N$	LD I, II	AOF	350		990	94	0
10	Minimizing Offsets with $P_O > P_N$	LD I, II	AOF	495		520	94	0
11	Minimizing Latency with $P_O > P_N$ and and $O > \max(R_O - x)$	LD I, II	AOF	350		1347	94	0
12	Minimizing Latency with $P_N > P_O$	LD II	ANF	784	150	168	150	1

Before proceeding further, some notes on the definition of latency used for minimization are in order:

- In cases 1 through 6, we used the definition of latency I to guide the minimization process. The goal of these cases was to show the flexibility of the minimization algorithm in addressing both offset and latency minimization requirements, as well as to compare our approach to the one of Real and Crespo (2001).

- In cases 7 through 11, the goal was to minimize latency and we also sought to test the sensitivity of the algorithm to the definition of latency employed. Therefore, each case was minimized using latency definition I and II. However, only case 7 produced different results depending on the definition. The remaining cases were not affected by the type of definition of latency.

- In case 12, the goal was specifically to configure a ANF mode change. Therefore, latency definition II was used. As it is clarified in the next few paragraphs, the definition II is the one to be used when configuring ANF changes.

The utilization is often not used as a schedulability analysis criteria in real-time systems, let alone when dealing with mode changes. However, within the context of this work, it is a useful concept to better understand the behavior of the transition. The ordering of the mix of old and new mode tasks during the mode change may have an substantial impact on both the functionality and performance of the application. Therefore, it was taken in this chapter as a roadmap to study the types of mode changes and their configuration (including minimization of the latency).

It is possible to establish, after the analysis of the case studies presented in this chapter (also chapter 3), that there is an inverse relationship between the minimization of offsets and latency. Clearly, small offsets tend to result in greater latencies and vice versa.

Case study 6 is the only balanced mode change (BMC). However, recall that case 6 used the algorithm of Real and Crespo (2001) which mostly reduces the offsets of high-priority new-mode tasks. As the additional introduction of lower-priority new-mode tasks within the significant interval leads to unschedulability (i.e. missed deadlines), the configuration algorithm allows old-mode tasks to execute and complete after the completion of the high-priority new-mode tasks. Hence, the algorithm of Real and Crespo (2001) tends to a balanced type of mode change.

From all case studies, cases 7, 8 and 12 were sensitive to the definition of latency. These cases presented different values of latency depending on the definition of latency (I or II) used. This occurred since in these cases new-mode tasks were assigned larger priorities than the old-mode tasks. Therefore, the new-mode tasks completed before the old-mode tasks. Thus, the value of latency by definition II is less than the value of latency definition I, because the latency definition II only takes into account the worst-case response time of the new-mode tasks. This validate the mostly-new-first and/or the all-new-first classification of mode changes, i.e. it was possible to specify and minimize these types of changes.

Cases 3, 4 and 5 show that is possible to restrict offset values within a range for some variables and still have the latency minimized. This approach provides the application designer with some level of control of the behavior of tasks across the mode change, allowing, for example, the use of precedence relations between them.

The analysis of the case studies showed that it is important to distinguish between two separate issues or challenges at stake: 1) Configuring a particular type of mode change, and 2)

Minimizing the latency of the configuration (type of mode change). Whereas we may be able to perform the configuration, it does not necessarily mean that it is also possible to minimize a particular configuration, specially if the system is stressed with high utilization. As a matter of fact, both the configuration and the minimization are susceptible to the utilization of the system. In general, larger processor utilization reduce the range of possibilities in working with types of mode changes, limiting the system to synchronous AOF (or MOF) types of changes, such as case studies 1, 2, 9, 10, 11, as well as cases 3 through 5 . This is because these are the only changes that alleviate the processor load during the transition.

In essence, the aim of this chapter was demonstrate the flexibility of the proposed approach: it was not limited to the case studies discussed in the previous chapter. We added 12 case studies that show that we can enforce a wide range of mode-change scenarios for which we wish to minimize the latency. The approach was not limited to these cases, and more scenarios can be identified for minimization. In addition to that, we provide a classification of mode-changes and study what type of mode-change results after the minimization of a mode-change in each case study. This classification can assist the system designer providing a method to configure a particular type of mode-change in accordance with the requirements demanded.

Table 40 – Requirements for Each Type of Mode-Change

Type	**Requirements**
ANF	$P_N > P_O$ minimization: using LD II and not high utilization
AOF	$P_O > P_N$ minimization: $O_{min} = \max(R_O - x)$
BMC	Strict-Alternation: Pairwise assignment of P Random: Random P Uniform: Uniform P
MOF	$P_N > P_O$ for a few tasks
MNF	$P_N > P_O$ and $Y \simeq 0$ and not high utilization

It was possible by means of these case studies to the type of mode change depends on the tasks priorities, the utilization of the system, and the type of definition of latency used. Therefore, we summarize these requirements for each type of mode change as follows (Table 40):

- An *ANF* type of mode change can be obtained if the priorities of all new mode tasks are larger than the priorities of the old-mode tasks (i.e. $P_N > P_O$). This type of mode change can also be minimized when we apply the latency definition II. The minimization algorithm forces all offsets to be close to the MCR in order to fulfill the objective of low latency.

- An AOF mode change is likewise obtainable if all old-mode tasks have a higher priority than the new-mode ones. To minimize the latency of this type of mode change, it is also required that the smalls offset O is larger than the maximum completion time of all old-mode tasks.

- A BMC can occur in strict alternation, randomly or uniformly. Strict alternation means that tasks alternate during the significant interval, i.e. one new mode tasks is released after the old-mode tasks followed by a new-mode tasks again and so on. Another possibility is that there is no pattern in the release of old and new tasks, however, the total number of new-mode tasks released is the same as the number of old-mode tasks. Uniform release means that a pattern of task releases can be identified, such as n-new followed by n-old number of tasks (n is an integer > 1). This types of mode changes can be accomplished if tasks priorities are assigned in accordance with the desired release pattern. If strict alternation of tasks is the goal, then priorities should likewise be assigned in strict alternation. In this case, the highest priority level in the system is assigned to both a new and and old-mode tasks. The next priority level is assigned to another pair of old and new tasks and so on. This can be a challenge if priorities are restricted somehow, e.g. by a priority-ceiling protocol (PCP).

- A MOF can be configured by having a few new-mode tasks with larger priorities than the old-mode tasks with lower offsets. To be able to minimize this type, it is advised that *“latency definition I”* must be used.

- A MNF type of mode change can be configured if a few old-mode tasks have priority larger than the new-mode tasks while new-mode tasks are released close to the MCR. To minimize this configuration, it is required (additionally) no mode of operation has a high utilization.

In one hand, AOF is the easiest type of mode change to accomplish (from the schedulability analysis perspective), since it is always releasing resources back to the system and allowing more system utilization (i.e. it creates more spare capacity). It is then easy for the minimization approach to find a feasible (i.e. schedulable) solution. On the other hand, the most difficult type to achieve is the ANF. Unlike AOF, this type requires substantial CPU utilization since the old and the new mode both compete with each other for CPU utilization. It is clearly then more difficult to the minimization algorithm to find a feasible solution. Within these extremes lie the balanced mode change BMC, where nearly half of the tasks executed during the significant interval are new and the other half are old. The MNF is relatively more difficult than the MOF for the same reason mentioned earlier. The more new-mode tasks introduced during the significant interval, the harder it is for the configuration tool to find a feasible solution. Another factor that impacts on the difficulty in achieving one type of mode change or another is the utilization of the old and the new-mode task set. If either the old or the new-mode has a high processor utilization, then few or no new-mode tasks are able to be introduced without causing missed deadlines. Therefore, heavily loaded modes of operation call for either an AOF or a MOF type of mode change. In essence, Table 40 is a guideline to configure the set of types of mode changes. Whether or not the method will be able to achieve such type with minimized latency will depend on the *configurability* of a mode of operation. The configurability itself will largely depend on the processor utilization. In general, modes of operation that are lightly loaded will be amenable to be configured in most types of mode changes, whereas heavily loaded modes cause the minimization algorithm to produce either a minimized AOF or MOF types of mode changes.

5 Summary and Conclusions

Mode changes are a structured way to dynamically configure real-time systems. During a mode change, new tasks are introduced and old-mode ones are removed, where some other tasks are simply changed according to the applications' demands. Although there has been significant work presented for mode changes in the realm of fixed-priority scheduled systems, work on the assignment of offsets, and specifically work on the minimization of the latency of mode changes has remained largely an open issue in real time systems. The latency of a mode change is clearly a crucial parameter since during the transition the system only partially delivers its functionality and desired performance. Hence, the shorter the latency, the better are the chances of the system coping with its mission and pre-specified requirements. In the absence of algorithms, methods and tools that automate and facilitate minimization, offset assignments have to be accomplished manually. The drawbacks of manual assignment of offsets are at least the following: 1) It is manually tedious and may require detailed knowledge of the inner workings of mode change protocols; 2) It may not lead to results that fulfill real-time systems requirements, and 3) It may not be feasible for large task sets and more complex mode-change scenarios. Therefore, the introduction of mode changes in fixed-priority preemptively scheduled real-time systems depends on the availability of higher-level tools and methods that allow practitioners to configure the mode change, including the minimization of the worst-case mode change latency.

This work was introduced as a first step to overcome this challenge and research gap, by presenting a model and approach to minimization of the latency of mode changes in real time systems. The approach chosen is based on evolutionary algorithms due to their known ability to solve relatively complex optimization scenarios. The approach was validated by test cases, each of which considered a different scenario to the optimization of latency.

Future work may include one or more of the following avenues: 1) Use of other evolutionary algorithms and finding their suitability to this sort of problem: As the focus of this work was not on artificial intelligence, but instead on modeling and finding a solution to the challenge of minimization, we have left for future work issues such as which is the most appropriate metaheuristic to the challenge at hands. We have adopted single and multi-objective GAs, but clearly other approaches such as simulated annealing, particle swarm intelligence (PSO) and other methods could have clearly been used and compared among each other; 2) Use of evolutionary algorithms to configure the parameters of the task set (C, T, D, P, B) to meet real-time goals: This work has focused on the latency, but other real-time parameters can also be tweaked to improve system performance. For example, periods, deadlines and execution times are parameters that usually can be adjusted and configured (through evolutionary approaches) within a range in order to achieve better results; 3) Minimization using a large number of tasks: This work used a small set of tasks, but, in future work it is possible increase the performance using parallel computing to allows it for use with a large number of tasks; 4) Use with multicore/multiprocessor analysis: The schedulability analysis model used in this work is to uniprocessor systems. In future work this approach may be expanded for use in multicore/multiprocessor systems; 5) Finally, in this work we focused on demonstrating the approach. There was not much effort dedicated to fine tuning the parameters of the GA algorithms, where there might be potential for improving the results even further.

To sum up, the major conclusions and/or results of this work were:

- *Identification of latency as a key requirement*: Before this work commenced, previous work have primarily dealt with the assignment of offsets as a critical parameter to be minimized. This work has recognized and brought up the importance of the latency of a mode change as well as an important parameter to be optimized.

- *Minimization of offsets does not necessarily lead to the minimization of latency*: The state-of-the-art before this work was such that it was not formally clear what was the impact of offsets on the duration of a mode change. It was intuitively established among practitioners that a reduction of offsets would lead to shorter latencies. This work raised and analyzed the issue by showing a counter- example: a reduction in offsets may lead to an increase in the latency in some scenarios, especially when the CPU utilization is relatively large at the start of the mode change.

- *Feasibility of the approach (use of metaheuristics)*: There were many questions raised earlier at the inception of this work, as to how to model the minimization problem in terms of metaheuristics, and whether or not it would be computationally feasible to execute the approach. We have shown that the approach is viable and the results for a variety of scenarios are significant. These considerations led us to the conclusion that the issues of latency minimization and mode-change configuration are best dealt with metaheuristic approaches such as genetic algorithms. Furthermore, the use of metaheuristics has previously been confined to the assignment of tasks to processors in multi-processor real-time systems. This work has expanded the application of such methods in the field of real-time systems.

- *Flexibility of the approach (based on metaheuristics)*: In general, we found that the approach presented is a general and flexible tool in that it can be easily customized to deal with various mode-change scenarios. These scenarios may vary from simple ones, such as the bare reduction of offsets, to more complex scenarios which may include multiple optimization objectives, allowing the configuration of the mode changes. Before this work started, there were no tools or methods to deal with the reduction of mode-latency in real-time systems. The closest method available was the one by Real and Crespo (2001). However, this method was exclusively applicable to the reduction of offsets. Whereas the reduction of offsets is ideal to improve promptness (the requirement that some tasks need to execute soon at the start of a mode change to fulfill urgent, e.g. alarm functions), reducing latency requires another algorithm. [1]

- *Framework for identification and management of mode changes (understand, configure and minimize)*: Before this dissertation, the classification of mode changes was based on a two-tiered system comprised of synchronous and asynchronous changes. Whereas this was a significant step towards identifying and understanding the behavior of changes, it was somehow limited to recognizing and modeling the motivation of a mode change from the application perspective. Therefore, this work has expanded on this issue, as well by identifying and describing five types of mode changes. This classification brings up the issue of which task set must execute first across a mode change, which is ultimately dictated by the application requirements. Whereas this classification scheme can be extended to deal with more complex scenarios, we deem it satisfactory to deal with most of the applications in real time systems. The types of mode change uncovered not only allow one to better understand the behavior of the application during a mode change: with the framework provided (classification and tool) it is now possible for a designer to specify the type of mode change desired, the configuration of the mode change parameters to achieve it, and ultimately the minimization of the latency for that type of mode change.

[1] Our experience has shown that it is not an easy task to modify Real and Crespo (2001) algorithm to deal with latency reduction.

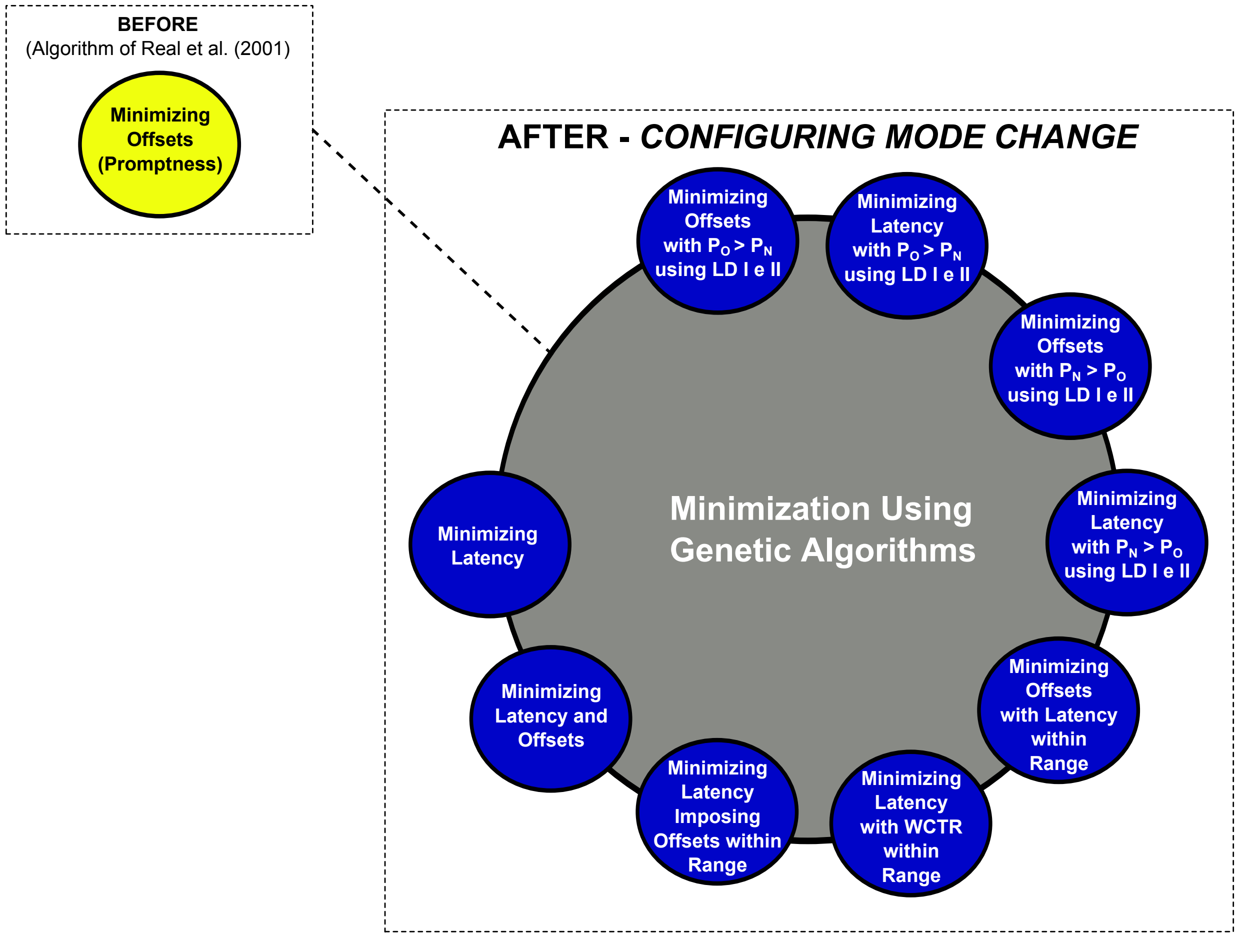

BEFORE
(Algorithm of Real et al. (2001)
Minimizing Offsets (Promptness)
AFTER - CONFIGURING MODE CHANGE
Minimization Using Genetic Algorithms
Minimizing Offsets with $P_O > P_N$ using LD I e II
Minimizing Latency with $P_O > P_N$ using LD I e II
Minimizing Offsets with $P_N > P_O$ using LD I e II
Minimizing Latency with $P_N > P_O$ using LD I e II
Minimizing Offsets with Latency within Range
Minimizing Latency with WCTR within Range
Minimizing Latency Imposing Offsets within Range
Minimizing Latency and Offsets
Minimizing Latency

Figure 30 – Research Before and After this Work.

Fig. 30 shows an overview of the state-of-the-art on offset minimization before and after this work. In summary, there was previously only one algorithm to minimize offsets aiming at promptness. This work approached different ways of configuring the minimization of mode changes.

Bibliography

ANDERSSON, B. Uniprocessor edf scheduling with mode change. In: *Proceedings of the 12th International Conference on Principles of Distributed Systems*. Berlin, Heidelberg: Springer-Verlag, 2008. (OPODIS '08), p. 572–577. ISBN 978-3-540-92220-9. Available from Internet: <http://dx.doi.org/10.1007/978-3-540-92221-6_43>.

BONGARD, J. Biologically inspired computing. *Computer*, v. 42, n. 4, p. 95–98, 2009. ISSN 0018-9162.

BRIAND, L. C.; LABICHE, Y.; SHOUSHA, M. Using genetic algorithms for early schedulability analysis and stress testing in real-time systems. *Genetic Programming and Evolvable Machines*, Kluwer Academic Publishers, Hingham, MA, USA, v. 7, n. 2, p. 145–170, jun. 2006. ISSN 1389-2576. Available from Internet: <http://dx.doi.org/10.1007/s10710-006-9003-9>.

BURNS, A.; WELLINGS, A. *Real-Time Systems and Programming Languages: Ada, Real-Time Java and C/Real-Time POSIX*. 4th. ed. USA: Addison-Wesley Educational Publishers Inc, 2009. ISBN 0321417453, 9780321417459.

CASTRO, L. de. *Fundamentals of Natural Computing: Basic Concepts, Algorithms, and Applications*. Taylor & Francis, 2006. (Chapman & Hall/CRC Computer & Information Science Series). ISBN 9781584886433. Available from Internet: <http://books.google.com.br/books?id=N6iYpNVP9RgC>.

COELLO, C. A. C. Handling preferences in evolutionary multiobjective optimization: a survey. In: . [s.n.], 2000. v. 1. Available from Internet: <http://dx.doi.org/10.1109/CEC.2000.870272>.

COELLO, C. A. C.; LAMONT, G. B.; VELDHUIZEN, D. A. V. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

CORTE, A. D. *et al. Optimisation of water distribution network design: a critical review.* [S.l.], 2012.

DEB, K. Multi-Objective Optimization Using Evolutionary Algorithms : An Introduction. *KanGAL Report Number 2011003*, p. 1–24, february 2011.

DEB, K.; AGRAWAL, S.; PRATAP, A.; MEYARIVAN, T. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. In: . [S.l.]: Springer, 2000. p. 849–858.

FOGEL, L.; OWENS, A.; WALSH, M. *Artificial Intelligence Through Simulated Evolution.* John Wiley & Sons, 1966. Available from Internet: <http://books.google.com.br/books?id=QMLaAAAAMAAJ>.

FOHLER, G. *Changing Operational Modes in the Context of Pre Run-Time Scheduling.* 1993.

HOLLAND, J. H. *Adaptation in Natural and Artificial Systems.* [S.l.]: The University of Michigan Press, 1975.

JONG, K. D. *Evolutionary Computation: A Unified Approach.* Mit Press, 2006. (Bradford Book). ISBN 9780262041942. Available from Internet: <http://books.google.com.br/books?id=OIRQAAAAMAAJ>.

KIM, M.; HIROYASU, T.; MIKI, M.; WATANABE, S. Spea2+: Improving the performance of the strength pareto evolutionary algorithm 2. In: *Parallel Problem Solving from Nature - PPSN VIII.* Springer Berlin Heidelberg, (Lecture Notes in Computer Science, v. 3242). p. 742–751. ISBN 978-3-540-23092-2. Available from Internet: <http://dx.doi.org/10.1007/978-3-540-30217-9_75>.

LEHOCZKY, J. Fixed priority scheduling of periodic task sets with arbitrary deadlines. *Real-Time Systems Symposium, 1990. Proceedings., 11th*, p. 201 –209, december 1990.

LOCKE, C.; VOGEL, D.; MESLER, T. Building a predictable avionics platform in ada: a case study. In: *Real-Time Systems Symposium, 1991. Proceedings., Twelfth.* [S.l.: s.n.], 1991. p. 181–189.

MANCHON, U.; HO, C.; FUNK, S.; RASHEED, K. Gart: A genetic algorithm based real-time system scheduler. In: *Evolutionary Computation (CEC), 2011 IEEE Congress on.* [S.l.: s.n.], 2011. p. 886–893. ISSN Pending.

MICHALEWICZ, Z. *Genetic Algorithms + Data Structures = Evolution Programs (3rd Ed.).* London, UK, UK: Springer-Verlag, 1996. ISBN 3-540-60676-9.

NELIS, V.; ANDERSSON, B.; MARINHO, J.; PETTERS, S. Global-edf scheduling of multimode real-time systems considering mode independent tasks. In: *Real-Time Systems (ECRTS), 2011 23rd Euromicro Conference on.* [S.l.: s.n.], 2011. p. 205–214. ISSN 1068-3070.

NÉLIS, V.; GOOSSENS, J. Mode change protocol for multi-mode real-time systems upon identical multiprocessors. *CoRR*, abs/0809.5238, 2008. Available from Internet: <http://arxiv.org/abs/0809.5238>.

NEUKIRCHNER, M.; LAMPKA, K.; QUINTON, S.; ERNST, R. Multi-mode monitoring for mixed-criticality real-time systems. In: *Proceedings of the Ninth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis.* Piscataway, NJ, USA: IEEE Press, 2013. (CODES+ISSS ’13), p. 34:1–34:10. ISBN 978-1-4799-1417-3. Available from Internet: <http://dl.acm.org/citation.cfm?id=2555692.2555726>.

NIZ, D. de; PHAN, L. Partitioned scheduling of multi-modal mixed-criticality real-time systems on multiprocessor platforms. In: *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2014 IEEE 20th.* [S.l.: s.n.], 2014. p. 111–122. ISSN 1080-1812.

PARETO, V. *Cours d’Economie Politique.* Genève: Droz, 1896.

PEDRO, P. *Schedulability of Mode Changes in Flexible Real-Time Distributed Systems*. Tese (Doutorado) — "The University of York", 1999.

PEDRO, P.; BURNS, A. Schedulability analysis for mode changes in flexible real-time systems. *Proceeding. 10th EUROMICRO Workshop on Real-Time Systems (Cat. No.98EX168)*, IEEE Comput. Soc, p. 172–179, june 1998. Available from Internet: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=685082>.

REAL, J.; CRESPO, A. Offsets for scheduling mode changes. *Real-Time Systems, 13th Euromicro Conference on, 2001.*, p. 3–10, 2001.

REAL, J.; CRESPO, A. Mode change protocols for real-time systems: A survey and a new proposal. *Real-Time Systems*, v. 26, n. 2, p. 161–197, march 2004. ISSN 0922-6443. Available from Internet: <http://link.springer.com/10.1023/B:TIME.0000016129.97430.c6>.

REAL, J.; WELLINGS, A. Dynamic ceiling priorities and ada 95. *Ada Letters, XIX(2)*, p. 41–48, 1999.

RECHENBERG, I. *Cybernetic solution path of an experimental problem*. [S.l.], 1965.

SHA, L.; SHA, L.; RAJKUMAR, R.; RAJKUMAR, R.; LEHOCZKY, J.; LEHOCZKY, J.; RAMAMRITHAM, K.; RAMAMRITHAM, K. Mode change protocols for priority-driven preemptive scheduling. *Real-Time Systems*, v. 1, p. 243–264, 1988.

STOIMENOV, N.; PERATHONER, S.; THIELE, L. Reliable mode changes in real-time systems with fixed priority or EDF scheduling. *2009 Design, Automation & Test in Europe Conference & Exhibition*, Ieee, p. 99–104, abr. 2009. Available from Internet: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5090640>.

TINDELL, K.; BURNS, A.; WELLINGS, A. Mode changes in priority preemptively scheduled systems. In: *Real-Time Systems Symposium, 1992*. [S.l.: s.n.], 1992. p. 100–109.

TINDELL, K.; BURNS, a.; WELLINGS a.J. An extendible approach for analysing fixed priority hard real-time systems. *Journal of Real-Time Systems*, p. 133–152, march 1994.

VELDHUIZEN, D. A. V.; LAMONT, G. B. *Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art*. 2000.

YOMSI, P.; NELIS, V.; GOOSSENS, J. Scheduling multi-mode real-time systems upon uniform multiprocessor platforms. In: *Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on*. [S.l.: s.n.], 2010. p. 1–8. ISSN 1946-0740.

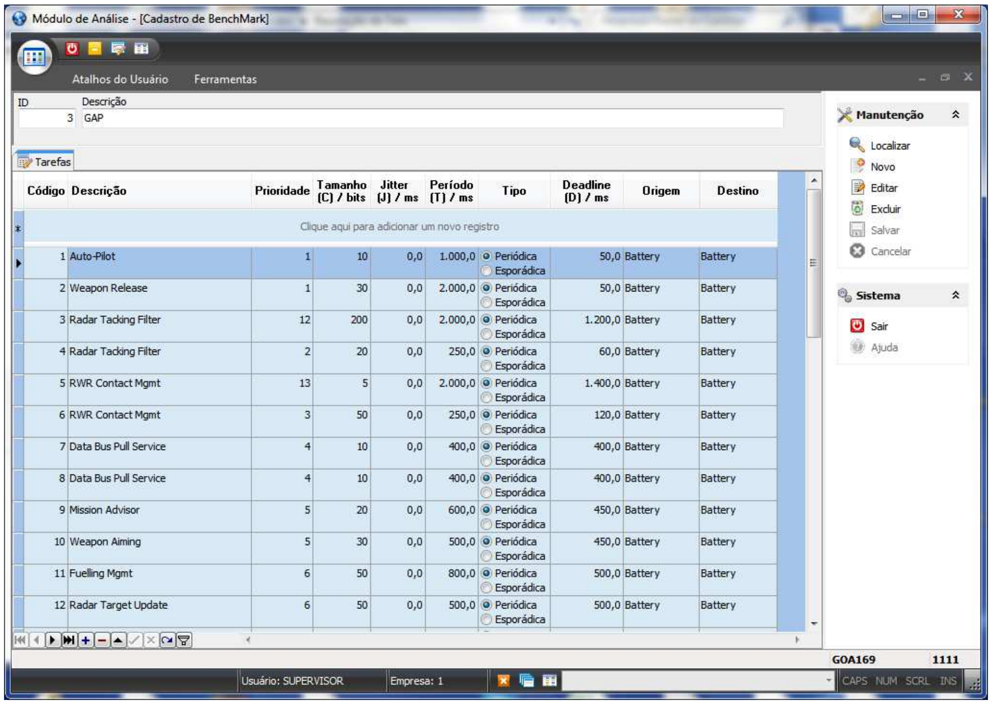
YOO, M. Real-time task scheduling by multiobjective genetic algorithm. *Journal of Systems and Software*, v. 82, n. 4, p. 619 – 628, 2009. ISSN 0164-1212. Special Issue: Selected papers from the 2008 {IEEE} Conference on Software Engineering Education and Training (CSEET08). Available from Internet: <http://www.sciencedirect.com/science/article/pii/S0164121208002070>.

Appendix

APPENDIX A – Software Tool

In this appendix we illustrate a software tool developed within the context of this dissertation to allow the configuration of mode changes. This tool has a high-level interface where the user can select the goals to be achieved when configuring the offsets. From an internal software design perspective, the program is also modular so that it can be easily extended to include more optimization objectives. Besides the usage in operation systems, this software was designed to be used for the analysis of messages across a mode change in a Controller Area Network (CAN). This tool was developed using Oracle Datatabase and Delphi programming language. The main features of the software tool are as follows:

1. *Tasks descriptor:* This feature allows the definition of the tasks that compose the various modes of operation. The tasks are described through the following fields: id, description, size, priority, periodicity, deadline and type (periodic or sporadic), as it can be observed in the Fig. 31.

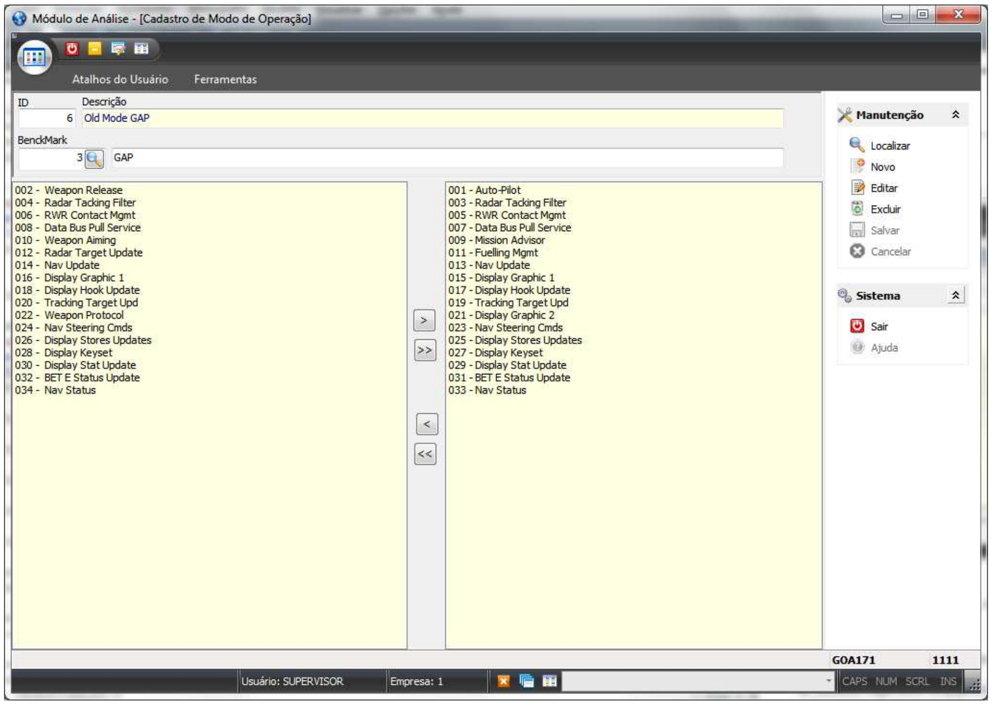


ID	Descrição
3	GAP

Código	Descrição	Prioridade	Tamanho (C) / bits	Jitter (J) / ms	Período (T) / ms	Tipo	Deadline (D) / ms	Origem	Destino
Clique aqui para adicionar um novo registro									
1	Auto-Pilot	1	10	0,0	1.000,0	[✓ Periódica] Esporádica	50,0	Battery	Battery
2	Weapon Release	1	30	0,0	2.000,0	[✓ Periódica] Esporádica	50,0	Battery	Battery
3	Radar Tacking Filter	12	200	0,0	2.000,0	[✓ Periódica] Esporádica	1.200,0	Battery	Battery
4	Radar Tacking Filter	2	20	0,0	250,0	[✓ Periódica] Esporádica	60,0	Battery	Battery
5	RWR Contact Mgmt	13	5	0,0	2.000,0	[✓ Periódica] Esporádica	1.400,0	Battery	Battery
6	RWR Contact Mgmt	3	50	0,0	250,0	[✓ Periódica] Esporádica	120,0	Battery	Battery
7	Data Bus Pull Service	4	10	0,0	400,0	[✓ Periódica] Esporádica	400,0	Battery	Battery
8	Data Bus Pull Service	4	10	0,0	400,0	[✓ Periódica] Esporádica	400,0	Battery	Battery
9	Mission Advisor	5	20	0,0	600,0	[✓ Periódica] Esporádica	450,0	Battery	Battery
10	Weapon Aiming	5	30	0,0	500,0	[✓ Periódica] Esporádica	450,0	Battery	Battery
11	Fuelling Mgmt	6	50	0,0	800,0	[✓ Periódica] Esporádica	500,0	Battery	Battery
12	Radar Target Update	6	50	0,0	500,0	[✓ Periódica] Esporádica	500,0	Battery	Battery

Figure 31 – Screen of Task Descriptor

2. *Definition of Operational Modes:* Having defined the tasks, it is necessary to associate them to an operational mode. Fig. 32 shows that, using the tasks registered previously, it is possible to create a mode of operation by just selecting the task set intended. The operational modes created may then be used in multiple analysis/optimization processes.



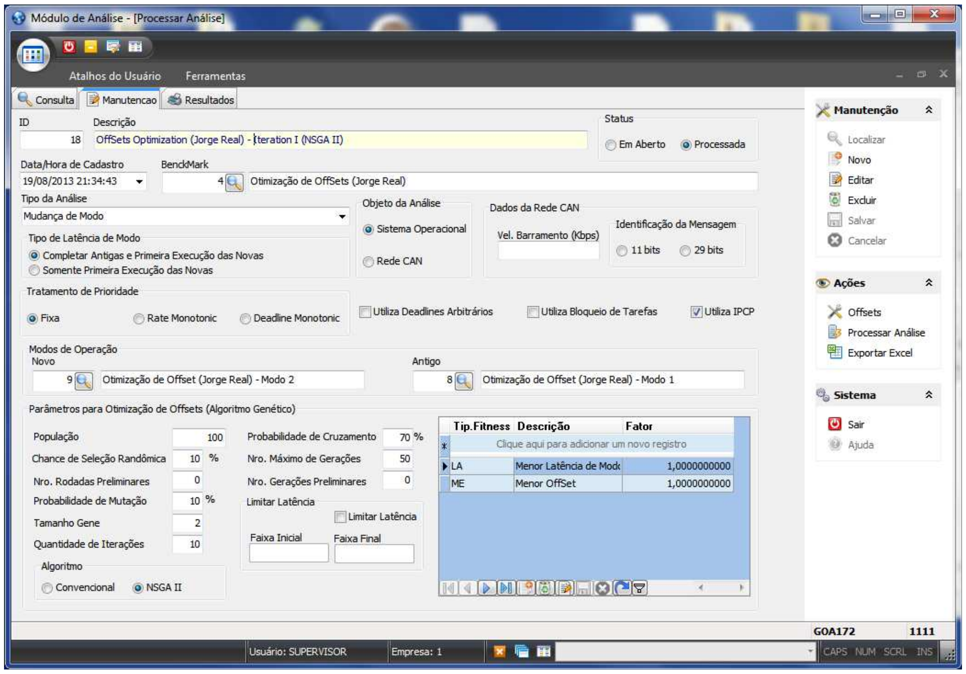
ID: 6
Descrição: Old Mode GAP
BenckMark: 3 GAP

```
002 -  Weapon Release              001 - Auto-Pilot
004 -  Radar Tacking Filter        003 - Radar Tacking Filter
006 -  RWR Contact Mgmt            005 - RWR Contact Mgmt
008 -  Data Bus Pull Service       007 - Data Bus Pull Service
010 -  Weapon Aiming               009 - Mission Advisor
012 -  Radar Target Update         011 - Fuelling Mgmt
014 -  Nav Update                  013 - Nav Update
016 -  Display Graphic 1           015 - Display Graphic 1
018 -  Display Hook Update         017 - Display Hook Update
020 -  Tracking Target Upd         019 - Tracking Target Upd
022 -  Weapon Protocol             021 - Display Graphic 2
024 -  Nav Steering Cmds           023 - Nav Steering Cmds
026 -  Display Stores Updates      025 - Display Stores Updates
028 -  Display Keyset              027 - Display Keyset
030 -  Display Stat Update         029 - Display Stat Update
032 -  BET E Status Update         031 - BET E Status Update
034 -  Nav Status                  033 - Nav Status
```

G0A171 1111
Usuário: SUPERVISOR
Empresa: 1

Figure 32 – Screen of Operation Modes Register

3. *Schedulability analyzer engine:* To begin the process of schedulability analysis, it is required to set the configuration parameters. These parameters are: object of analysis (task's in an operating system or CAN network), tasks set for new and old-mode, type of latency definition, priority assignment policy (deadline monotonic, rate monotonic or user defined), use of arbitrary deadlines, use of blocking and use of IPCP (Fig. 33). When the object to be analyzed are messages in a CAN, one must also associate the bus rate and the message header size. If IPCP is checked, the user is prompted to enter the shared resources and to which tasks they need to be allocated (Fig. 34).


Módulo de Análise - [Processar Análise]

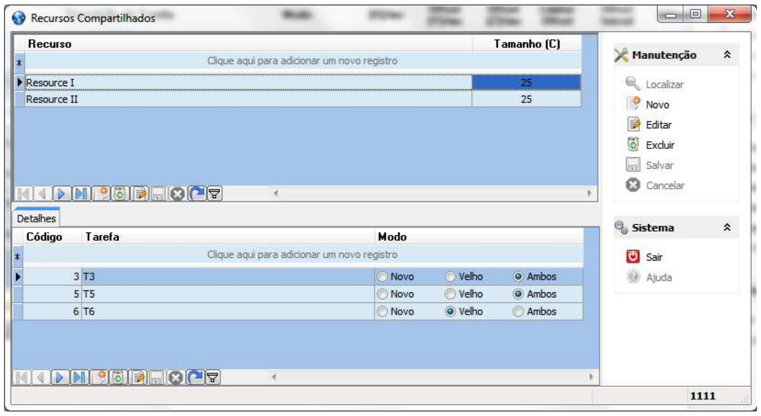
ID: 18
Descrição: OffSets Optimization (Jorge Real) - Iteration I (NSGA II)
Status: Em Aberto [✓ Processada]
Data/Hora de Cadastro: 19/08/2013 21:34:43
BenckMark: 4 Otimização de OffSets (Jorge Real)
Tipo da Análise: Mudança de Modo
Tipo de Latência de Modo: [✓ Completar Antigas e Primeira Execução das Novas] Somente Primeira Execução das Novas
Objeto da Análise: [✓ Sistema Operacional] Rede CAN
Dados da Rede CAN: Vel. Barramento (Kbps):
Identificação da Mensagem: 11 bits 29 bits
Tratamento de Prioridade: [✓ Fixa] Rate Monotonic Deadline Monotonic
Utiliza Deadlines Arbitrários ☐ Utiliza Bloqueio de Tarefas ☐ Utiliza IPCP ✓
Modos de Operação
Novo: 9 Otimização de Offset (Jorge Real) - Modo 2
Antigo: 8 Otimização de Offset (Jorge Real) - Modo 1

Parâmetros para Otimização de Offsets (Algoritmo Genético)
População: 100
Chance de Seleção Randômica: 10 %
Nro. Rodadas Preliminares: 0
Probabilidade de Mutação: 10 %
Tamanho Gene: 2
Quantidade de Iterações: 10
Algoritmo: Convencional [✓ NSGA II]
Probabilidade de Cruzamento: 70 %
Nro. Máximo de Gerações: 50
Nro. Gerações Preliminares: 0
Limitar Latência: ☐ Limitar Latência
Faixa Inicial:
Faixa Final:

Tip.Fitness	Descrição	Fator
Clique aqui para adicionar um novo registro		
LA	Menor Latência de Modo	1,0000000000
ME	Menor OffSet	1,0000000000

GOA172 1111
Usuário: SUPERVISOR Empresa: 1 CAPS NUM SCRL INS

Figure 33 – Screen of the Schedulability Analysis and GA Configuration


Recursos Compartilhados

Recurso	Tamanho (C)
Clique aqui para adicionar um novo registro	
Resource I	25
Resource II	25

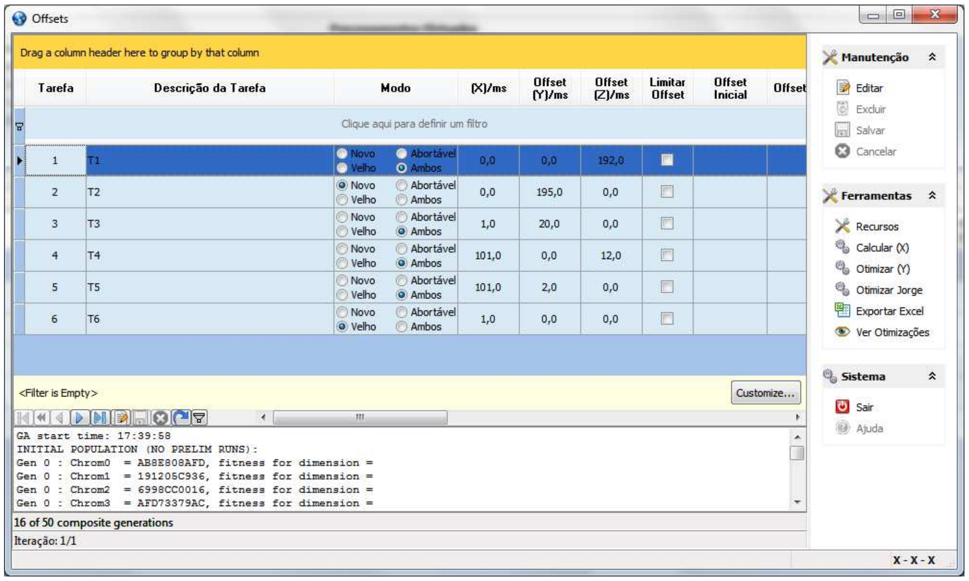
Código	Tarefa	Modo
Clique aqui para adicionar um novo registro		
3	T3	Novo Velho [✓ Ambos]
5	T5	Novo Velho [✓ Ambos]
6	T6	Novo [✓ Velho] Ambos

1111

Figure 34 – Screen of Shared Resources

4. *Genetic Algorithm engine:* The parameters used by the genetic algorithm for offsets optimization are initialized on the same screen provided by the schedulability analyzer (Fig. 33). These parameters are: population size, crossover probability, chance of random selection, number of generations, mutation probability, size of gene , number of iterations, kind of GA (conventional or NSGA II), latency limits (when applicable), the fitness functions and their weight.

5. *Offsets optimization process:* After completing the setting of the analysis / GA, the offsets/latency optimization process (Fig. 35) is ready to start. During this process, the system stores all the results (partial and complete ones) in a database for further (post-mortem) analysis. Note that, before starting the optimization process, the data related to any tasks can be adjusted without having to change the primary register.



Drag a column header here to group by that column

Tarefa	Descrição da Tarefa	Modo	(X)/ms	Offset (Y)/ms	Offset (Z)/ms	Limitar Offset	Offset Inicial	Offset
	Clique aqui para definir um filtro							
1	T1	Novo Velho Abortável [✓ Ambos]	0,0	0,0	192,0	☐		
2	T2	[✓ Novo] Velho Abortável Ambos	0,0	195,0	0,0	☐		
3	T3	Novo Velho Abortável [✓ Ambos]	1,0	20,0	0,0	☐		
4	T4	Novo Velho Abortável [✓ Ambos]	101,0	0,0	12,0	☐		
5	T5	Novo Velho Abortável [✓ Ambos]	101,0	2,0	0,0	☐		
6	T6	Novo [✓ Velho] Abortável Ambos	1,0	0,0	0,0	☐		

<Filter is Empty>

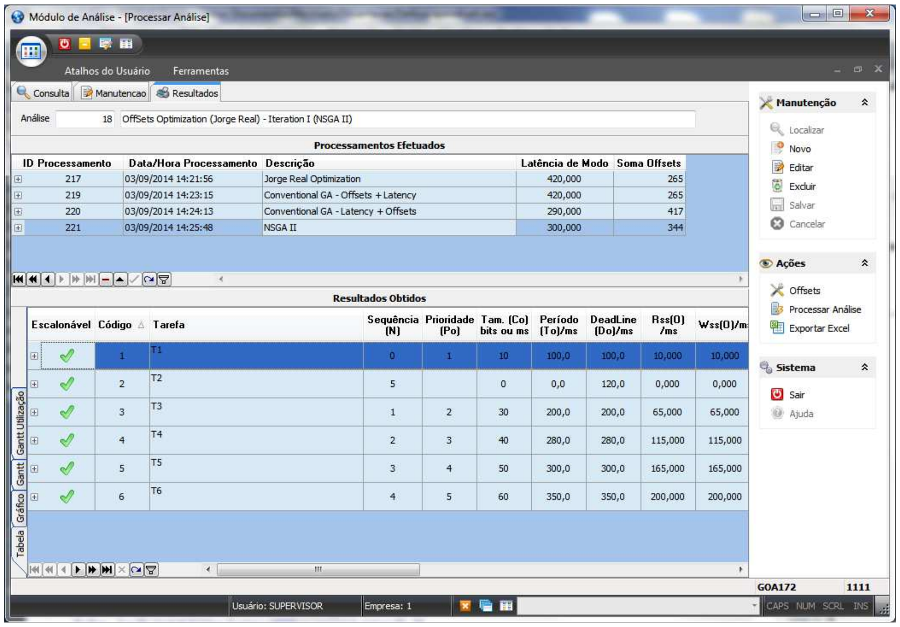
```
GA start time: 17:39:58
INITIAL POPULATION (NO PRELIM RUNS):
Gen 0 : Chrom0  = AB8E808AFD, fitness for dimension =
Gen 0 : Chrom1  = 191205C936, fitness for dimension =
Gen 0 : Chrom2  = 6998CC0016, fitness for dimension =
Gen 0 : Chrom3  = AFD73379AC, fitness for dimension =
```

16 of 50 composite generations

Iteração: 1/1

Figure 35 – Screen of Offsets Optimization

6. *Analysis of results:* The values obtained from the optimization process can be easily analyzed using the charts and graphs made available by the software tool. Fig. 36 shows the values of latency, sum of offsets, WCRT, etc., for each run (or cycle/iteration) of the optimization process. Fig. 37 shows a chart with the optimal Pareto front for an optimization process using NSGA II. Fig. 38 shows the utilization chart for a specific system configuration.



Análise 18 OffSets Optimization (Jorge Real) - Iteration I (NSGA II)

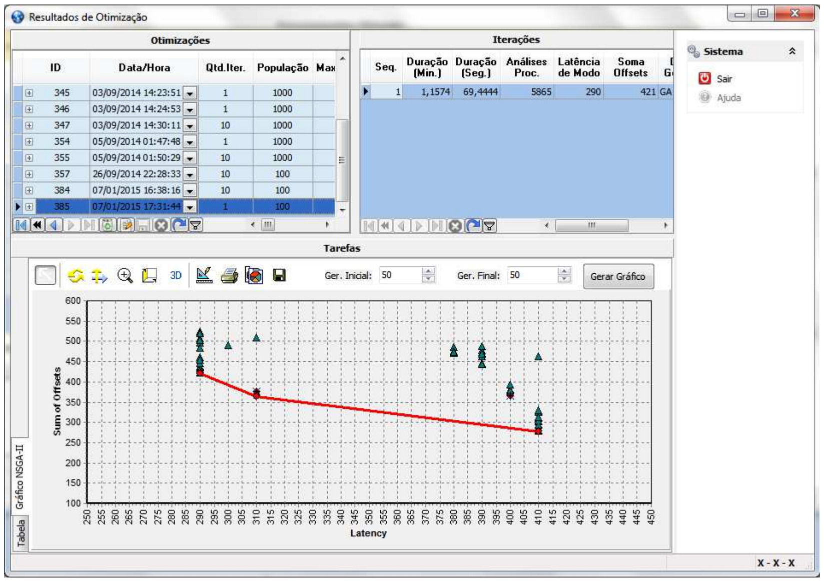
Processamentos Efetuados

ID Processamento	Data/Hora Processamento	Descrição	Latência de Modo	Soma Offsets
217	03/09/2014 14:21:56	Jorge Real Optimization	420,000	265
219	03/09/2014 14:23:15	Conventional GA - Offsets + Latency	420,000	265
220	03/09/2014 14:24:13	Conventional GA - Latency + Offsets	290,000	417
221	03/09/2014 14:25:48	NSGA II	300,000	344

Resultados Obtidos

Escalonável	Código	Tarefa	Sequência (N)	Prioridade (Po)	Tam. (Co) bits ou ms	Período (To)/ms	DeadLine (Do)/ms	Rss(O) /ms	Wss(O)/m
✓	1	T1	0	1	10	100,0	100,0	10,000	10,000
✓	2	T2	5		0	0,0	120,0	0,000	0,000
✓	3	T3	1	2	30	200,0	200,0	65,000	65,000
✓	4	T4	2	3	40	280,0	280,0	115,000	115,000
✓	5	T5	3	4	50	300,0	300,0	165,000	165,000
✓	6	T6	4	5	60	350,0	350,0	200,000	200,000

Figure 36 – Screen of Analysis of Result



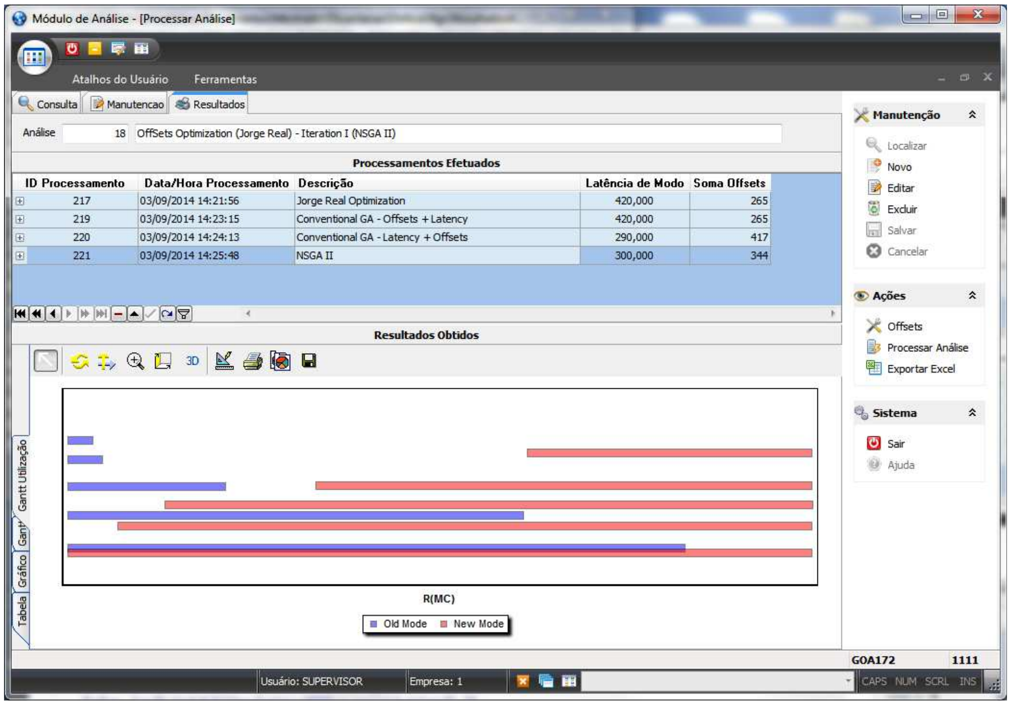
Otimizações

ID	Data/Hora	Qtd.Iter.	População	Max
345	03/09/2014 14:23:51	1	1000	
346	03/09/2014 14:24:53	1	1000	
347	03/09/2014 14:30:11	10	1000	
354	05/09/2014 01:47:48	1	1000	
355	05/09/2014 01:50:29	10	1000	
357	26/09/2014 22:28:33	10	100	
384	07/01/2015 16:38:16	10	100	
385	07/01/2015 17:31:44	1	100	

Iterações

Seq.	Duração (Min.)	Duração (Seg.)	Análises Proc.	Latência de Modo	Soma Offsets	G
1	1,1574	69,4444	5865	290	421	GA

Figure 37 – Screen of Pareto Font for NSGA Optimization


Módulo de Análise - [Processar Análise]

Análise 18 OffSets Optimization (Jorge Real) - Iteration I (NSGA II)

Processamentos Efetuados

ID Processamento	Data/Hora Processamento	Descrição	Latência de Modo	Soma Offsets
217	03/09/2014 14:21:56	Jorge Real Optimization	420,000	265
219	03/09/2014 14:23:15	Conventional GA - Offsets + Latency	420,000	265
220	03/09/2014 14:24:13	Conventional GA - Latency + Offsets	290,000	417
221	03/09/2014 14:25:48	NSGA II	300,000	344

Resultados Obtidos

Figure 38 – Screen of Utilization Gantt

7. *Export results:* Another feature is the possibility to export data related to the results obtained to a spreadsheet format , as well as the charts to image format.

The software showed in this appendix was an important by-product of this dissertation. It was used as a support tool to achieve the results throughout the research process, due to its versatility under different usage scenarios. Also, it is an asset and contribution to the implementation of future work.