TELEGRAM SCHEDULING FOR THE PERIODIC PHASE OF THE MULTIFUNCTION VEHICLE BUS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MUSTAFA ÇAĞLAR GÜLDİKEN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

JANUARY 2020

Approval of the thesis:

TELEGRAM SCHEDULING FOR THE PERIODIC PHASE OF THE MULTIFUNCTION VEHICLE BUS

submitted by **MUSTAFA ÇAĞLAR GÜLDİKEN** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Date: **January 31, 2020**

ABSTRACT

TELEGRAM SCHEDULING FOR THE PERIODIC PHASE OF THE MULTIFUNCTION VEHICLE BUS

Güldiken, Mustafa Çağlar
M.S., Department of Electrical and Electronics Engineering
Supervisor : Prof. Dr. Klaus Werner Schmidt
Co-Supervisor : Prof. Dr. Ece Güran Schmidt

January 2020, 97 pages

Train communication network comprises different standards such as the Wire Train Bus (WTB) for the data exchange among different vehicles and the Multifunction Vehicle Bus (MVB) for the data communication within vehicles. Specifically, MVB is a highly robust real-time field bus specifically designed for control systems built into rail-vehicles. MVB supports both periodic process data and sporadic message data transfers in the form of telegrams.

In order to achieve timely and efficient data exchange on MVB, the available bandwidth has to be used efficiently. Accordingly, the main focus of this thesis is the development of systematic scheduling approaches for periodic telegrams on MVB. In this respect, the thesis provides four main contributions. First, the thesis proposes an original integer linear programming (ILP) formulation for the schedule computation on MVB. Second, the thesis develops 5 basic heuristic algorithms for the fast computation of feasible MVB schedules. Third, the thesis introduces several swap operations for improving the schedules obtained from the basic heuristics. Finally, the thesis presents a comprehensive evaluation of the developed scheduling methods.

This evaluation shows that, different from the proposed heuristics, the ILP formulation cannot provide solution schedules for large telegram sets with reasonable runtimes. Specifically, two of the proposed heuristics and two of the developed swap operations are found most suitable as a practical solution to the MVB scheduling problem.

Keywords: Train Communication Network, Multifunction Vehicle Bus, scheduling, integer linear programming, heuristics

ÖZ

ÇOK FONKSİYONLU ARAÇ VERİYOLU'NUN PERİYODİK FAZI İÇİN TELEGRAM ÇİZELGELEMESİ

Güldiken, Mustafa Çağlar
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Klaus Werner Schmidt
Ortak Tez Yöneticisi : Prof. Dr. Ece Güran Schmidt

Ocak 2020 , 97 sayfa

Tren haberleşme ağı, farklı araçlar arasındaki veri aktarımını sağlayan WTB (Wire Train Bus) ve araç içindeki veri iletişimini sağlayan MVB (Multifunction Vehicle Bus) gibi farklı standartları kapsamaktadır. MVB, özellikle demiryolu araçları içerisindeki kontrol sistemleri için tasarlanmış son derece sağlam, gerçek zamanlı bir veriyoludur. MVB, hem periyodik işlem verilerinin hem de aperiyodik mesaj verilerinin aktarımını telegramlar ile yapmaktadır.

MVB için, zamanında ve verimli veri alışverisi yapabilmek amacıyla mevcut bant genişliğinin etkili kullanılması gerekmektedir. Buna göre tez, esas olarak MVB üzerindeki periyodik telegramlar için sistematik çizelgeleme yaklaşımlarının geliştirilmesine odaklanmaktadır. Bu bağlamda, tezin dört temel amacı bulunmaktadır. İlk olarak tez, MVB çizelgeleme işlemi için özgün tamsayılı doğrusal programlama (ILP) formülasyonunu sunmaktadır. İkinci katkı olarak tez, uygun MVB çizelgeleme işleminin hızlı yapılabilmesi için 5 farklı temel buluşsal algoritma geliştirmektedir. Üçüncü olarak tez, temel buluşsal algoritmalar tarafından oluşturulan çizelgelemelerin geliștirilmesi için değişim işlemlerini tanıtmaktadır. Son olarak tez, geliştirilen çizelgeleme metotları için kapsamlı değerlendirmeleri sunmaktadır. Bu değerlendirmeler, ILP formülasyonunun geliştirilen buluşsal algoritmalardan farklı olarak büyük veri setleri için makul çalışma süreleri içerisinde uygun çizelgelemeyi yapamadığını göstermektedir. Özellikle, hem geliştirilen buluşsal algoritmaların hem de değişim işlemlerinden ikisi MVB çizelgeme probleminde en uygun pratik çözümlerdir.

Anahtar Kelimeler: Tren Haberleşme Ağı, Çok Fonksiyonlu Araç Haberleşmesi, çizelgeleme, tamsayılı doğrusal programlama, buluşsal algoritmalar

To my wife Elif and my family

ACKNOWLEDGMENTS

Foremost, I would like to express my deepest gratitude to my supervisor Prof. Dr. Klaus Schmidt and co-supervisor Prof. Dr. Ece Güran Schmidt, for their excellent guidance and continuous support of my research. I am thankful for their encouragement and helpful discussions which were very beneficial throughout the study.

I thank my company ASELSAN Inc. for giving me the opportunity of continuing my education.

I would like to thank my team leader and my colleagues in ASELSAN Inc. for their support. I also thank all my friends who never hesitated from giving their invaluable supports and trust to me in any condition.

I wish to thank my parents, my brother, and my sister who never hesitated from giving their supports to me throughout my life.

My deepest appreciation to my precious wife Elif, for her endless support, trust, patience and unconditional love during this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIONS

BP	Basic Period
MP	Macro Period
MF	Master Frame
SF	Slave Frame
BU	Bus Utilization
MVB	Multifunction Vehicle Bus
WTB	Wire Train Bus
TCN	Train Communication Network
PP	Periodic Phase
ILP	Integer Linear Programming
DV	Decision Variable
MAB	Minimum Accumulated Basic Period
CF	Cursor and Flag
R-MAB	Randomized Minimum Accumulated Basic Period
MLB	Minimum Longest Basic Period
SAB	Scaled Average Basic Period
FFD	First-Fit-Decreasing
SMB	Swap Operation According to Maximum Basic Period Duration
SSB	Swap Operation According to Sum Of Basic Period Duration
S2T	Swap Operation Looking At The Next Two Telegrams

S3T	Swap Operation Comparing Three Telegrams
TO	Timeout
σ	Standard Deviation
IEC	International Electrotechnical Committee
UIC	International Railways Union
LIN	Local Interconnect Network
CAN	Controller Area Network
CAN DF	CAN with Flexible Datarate
MOST	Media Oriented Systems Transport
AVB	Automotive Ethernet

CHAPTER 1

INTRODUCTION

Today's vehicles contain numerous electronic control units (ECUs), sensors and many other electrical/electronic components [1, 2, 3, 4, 5]. In order to perform the advanced functions of vehicles, robust, reliable and efficient in-vehicle communication is required between these ECUs. The specific requirements of the different car domains and vehicle types have led to the development of a large number of automotive networks such as LIN (Local Interconnect Network), CAN (Controller Area Network), CAN FD (CAN with Flexible Datarate), FlexRay, MOST (Media Oriented Systems Transport), AVB (automotive Ethernet), WTB (Wire Train Bus), MVB (Multifunction Vehicle Bus), etc. [6, 7, 8, 9].

As a particular vehicle type, trains have evolved from being the first practical forms of mechanized land transport to the extremely complex and sophisticated transportation systems we currently use [10]. A deterministic and robust fieldbus communication solution is necessary for safety-sensitive systems within trains which must operate in harsh and distributed environments [5, 11, 12]. For example, the control of railway vehicles necessitates data communication with very low latencies [13, 14]. Addressing the requirement of standardization [15], the Train Communication Network (TCN) [16] was developed as an international standard for data communication aboard rail vehicles with the collaboration of railway operators and manufacturers [17, 18]. TCN consists of MVB to connect the equipment within a vehicle and WTB to connect the vehicles [19, 14]. In this thesis, we focus on the real-time communication of low-latency periodic data using MVB.

MVB is a highly robust real-time field bus specifically designed for control systems built into rail-vehicles [12, 13, 20]. MVB supports both periodic process data and sporadic message data transfers in the form of telegrams. There are various functional and non-functional requirements [13] for MVB. On the one hand, non-functional requirements are mostly concerned with the reliability of the network. For example, it has to be the case that:

- Redundant communication lines should be used to achieve high robustness and to prevent the single point of failure,
- Redundant bus masters should be coordinated with each other in order to prevent the communication breaks down because of the device fails,
- Different media (twisted wire pair, optical fibres and RS485) should be used for reliable communication,
- Repeaters should be used to connect the media for a transition from one medium to another,
- The devices should be able to capture statistical information to detect the problems early.

On the other hand, functional requirements specify how and when data should be communicated among MVB nodes. Important requirements are that:

- the bit rate should be 1.5 Mbit/s to maximize the effective data throughput,
- process data should be delivered at fixed time slots,
- synchronization should be ensured among the devices.

The recent literature mostly investigates non-functional requirements of MVB such as robustness, reliability and security. [12] develops an algorithm for checking the health status of MVB using anomaly detection based on experiments. A formal model-driven design approach is proposed so as to build a secure implementation of an MVB bus controller in [20]. The forecast and analysis of the MVB network performance is a vital process for MVB design. Furthermore, [21] performs a simulation study on Matlab/Simulink to analyze the network efficiency of MVB with respect to data length and amount of devices.

Regarding the functional requirements, it has to be noted that MVB enables a concise definition of the MVB schedule for periodic message transfers in the form of telegrams [13]. This MVB schedule consists of consecutive basic periods (BPs) with a fixed duration and the telegrams have to be placed into these BPs based on their period. That is, given a telegram set to be transmitted on MVB, it is required to determine an MVB schedule that defines the exact time instants where each telegram should be transmitted.

The main focus of this thesis is the computation of MVB schedules for the transmission of periodic telegrams. The only related work for this topic is given by [22], which rather focuses on the co-design of scheduling and control on MVB without taking a formal view on the MVB scheduling problem. Accordingly, this thesis first formalizes the MVB scheduling problem and proposed several performance metrics for quantifying the quality of an MVB schedule. Then, the thesis develops an integer linear programming (ILP) formulation for computing optimal MVB schedules. This work is also published in [23]. Since optimal schedules cannot be computed for large telegram sets, the thesis further develops several original heuristics and swap operations for the schedule computation on MVB. A comprehensive evaluation with many test cases shows that the heuristics compute close-to-optimal schedules for small and medium-size telegram sets. Most importantly, feasible schedules can be determined for very large telegram sets even in cases where an optimal schedule cannot be found.

In summary, this thesis proposes telegram scheduling algorithms for periodic data transmitted on MVB. The main contributions of the thesis are listed as follows:

- First formalization of the MVB scheduling problem in the literature,
- ILP formulation of the optimal MVB scheduling problem,
- Development of different basic heuristic algorithms for the fast computation of MVB schedules for large telegram sets,
- Swap operations to improve the MVB schedules obtained from the basic heuristics and to generate close-to-optimal schedules,

- A comprehensive evaluation of the proposed algorithms based on a large number of randomly generated telegram sets with different properties.

The remainder of the thesis is organized as follows: Section 2 provides the necessary background information about MVB. Section 3 states the problem addressed in the thesis and proposes an ILP formulation for MVB scheduling. In addition, different basic heuristics are developed and their suitability is evaluated based on several performance metrics. Section 4 introduces different swap operations that are applied to the proposed basic heuristics in order to improve the obtained schedules. Moreover, a comprehensive performance comparison between the ILP solutions and the schedules from the proposed algorithms is presented. Conclusions are given in Section 5.

CHAPTER 2

MULTIFUNCTION VEHICLE BUS: BACKGROUND

This chapter gives background information about the train communication network (TCN). A general overview is presented in Section 2.1 and Section 2.2 introduces the multifunction vehicle bus (MVB).

2.1 Train Communication Network (TCN)

The TCN was adopted as the international standard IEC 61375 in 1999 with a joint effort by the International Railways Union (UIC), Utrecht, Netherlands, and the International Electrotechnical Committee (IEC), Geneva, Switzerland with the deputies from over 20 countries, including many European nations, the US, Japan, and China representing major railways operators and manufacturers [17]. It introduces a standard form of data for train control, diagnostics, and passenger information that is suitable for various train combinations such as metros, or suburban and international trains. Accordingly, the purpose of standardization is to define interfaces between programmable equipment, with the aim of achieving plug-compatibility.

The TCN architecture indicates all appropriate configurations used in rail vehicles. It consists of the Multifunction Vehicle Bus (MVB) that connects devices inside each vehicle and the Wire Train Bus (WTB) to connect the different vehicles as shown in Figure 2.1.

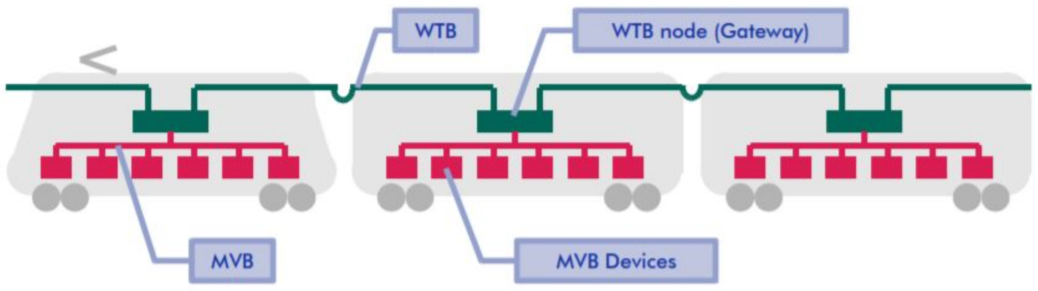


Figure 2.1: TCN Architecture

2.2 Multifunction Vehicle Bus (MVB)

MVB is a serial communication bus for railway vehicles that helps to connect devices within a vehicle for exchanging control, monitoring, and diagnosis information. According to [24], bus activity is divided into periods and the shortest period is denoted as the basic period (BP). The BP is the fixed time slot that is repeated and all BPs have the same duration T_{BP}. Following the MVB specification, the BP cycle time shall take a value as shown below:

$$1.0 \text{ ms} \leq T_{\text{BP}} \leq 2.50 \text{ ms}. \tag{2.1}$$

A BP is divided into three main phases as shown in Figure 2.2:

1. a Periodic Phase,
2. a Sporadic Phase,
3. a Guard Phase.

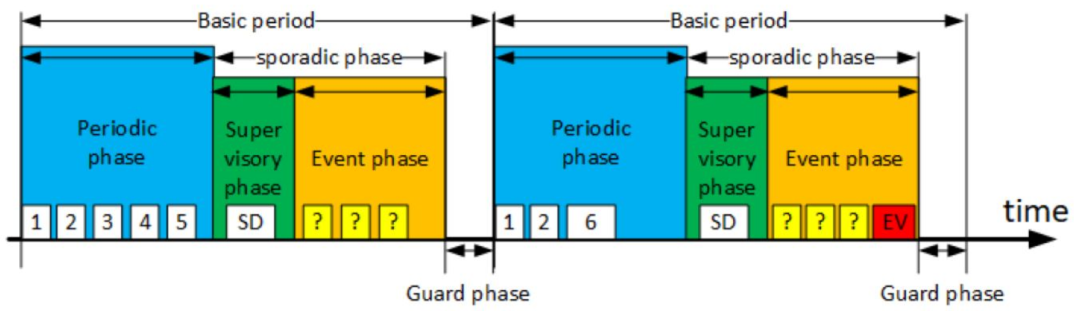


Figure 2.2: Basic Period

The Periodic Phase is reserved for periodic data and the Sporadic Phase is divided into a supervisory phase (for supervisory data) and an event phase (for event-triggered message data). The guard phase is introduced in order to separate consecutive BPs.

In principle, MVB supports both periodic data (for process variables) and sporadic data (for on-demand traffic) transfers in the form of telegrams. Process variables carry the state of the train such as the speed, motor current, and operator's commands. The Master polls the periodic data in sequence and periodic data are polled at their individual period (IP). Between periodic phases, the Master continuously polls the devices for events.

The IP is an interval between two successive transmissions of the same process data from the same source. An IP has to be equal to the BP duration T_{BP} multiplied by a power of 2 with a maximum value of $1024\,\text{ms}$. Then, the Macro Period (MP) is the longest IP, after which the periodic traffic returns to the same pattern. In this thesis, we denote the number of BPs in one MP as N_{MP} and the duration of the MP as $T_{\text{MP}} = N_{\text{MP}} \cdot T_{\text{BP}}$. If T_{BP} is equal to 1 ms, there can be at most 1024 BPs in one MP. In case that $T_{\text{BP}} = 2\,\text{ms}$ the maximum number of BPs is 512.

In MVB, data is sent via telegrams. Each telegram consists of a Master Frame (MF-request) and a Slave Frame (SF-response). The timing of a telegram anywhere on the bus is shown in Figure 2.3.

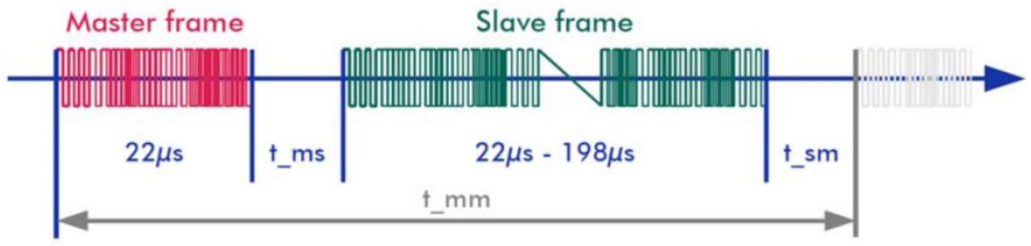


Figure 2.3: Telegram Structure and Timing

[25] indicates that an MF sent by the MVB Master has the format shown in Figure 2.4:

1. MF begins with the Master Start Delimiter (MSD),

2. MF is followed by 16 bits of frame Data,

3. MF is followed by the 8-bit Check Sequence (CS).

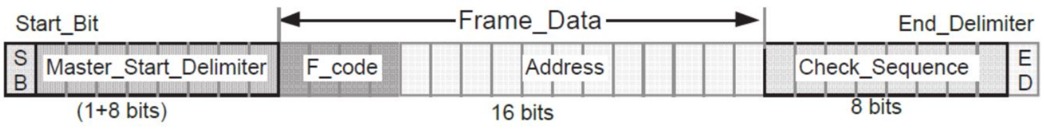


Figure 2.4: Master Frame Format

The length of the MF, which is 33 bits adding up the frame data, CS and MSD, is fixed to 22μs; this represents 16 bits of transmitted data according to the signaling speed which is defined as 1.5 Mbit/s $\pm 0.01\%$.

The time between an MF and its related SF (denoted as t_{ms}) is between 2 μs and 42.7 μs. Any slave node has to reply within 6 μs; in addition, time is allocated for possible delays of frames traveling along the line and crossing repeaters. In this thesis, $t_{\text{ms}} = 42.7\,\mu$s is used since it represents the worst-case scenario.

An SF sent by a Slave has the format shown in Figure 2.5:

1. SF begins with the Slave Start Delimiter (SSD),

2. SF is followed by 16, 32, 64, 128 or 256 bits of frame Data,

3. SF includes an 8-bit Check Sequence (CS) after each word of 64 data bits or appended to the frame in case of few data bits.

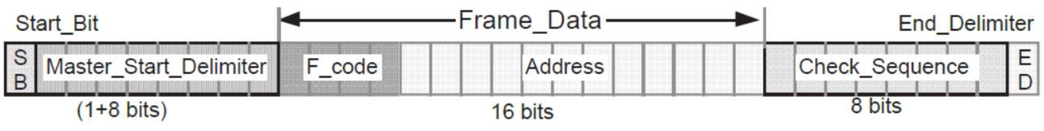


Figure 2.5: Slave Frame Format

The length of the SF depends on the type of data being transferred and is between 16 and 256 bits. Hence, the length of the SF changes between 22 μs and 198 μs according to the signaling speed which shall be 1.5 Mbit/s $\pm 0.01\%$.

The time from the end of an SF to the beginning of the next MF (t_{sm}) is at least 3 μs if it expects neither collision nor silence in response to its previous MF. The minimum value (3 μs) is used in this thesis.

The duration of each telegram equals the sum of the MF and SF duration, the propagation delay on the bus and possible processing delays in the MVB nodes as computed in (2.2) and as shown in Figure 2.3:

$$d_{\text{T}} = d_{\text{MF}} + d_{\text{SF}} + t_{\text{ms}} + t_{\text{sm}}. \tag{2.2}$$

Noting that the bit rate of MVB is 1.5 Mbit/s, using the telegram properties described above and writing k for the number of bit of the SF, the possible telegram durations are composed of:

$$d_{\text{MF}} = (16 \text{ bits} + 9 \text{ bits} + 8 \text{ bits})/1.5 \text{ Mb/s} = 22\mu s,$$
$$d_{\text{SF}} = (k \text{ bits} + 9 \text{ bits} + 2 \times 8 \text{ bits})/1.5\text{Mb/s},$$
$$t_{\text{ms}} = 22\,\mu s,$$
$$t_{\text{sm}} = 42.7\,\mu s,$$
$$d_{\text{T}} = (22 + d_{\text{SF}} + 42.7 + 3)\,\mu s.$$

Using all possible bit lengths of SFs, the duration of each telegram can be directly determined as in Table 2.1 according to (2.2).

Table 2.1: Telegram Durations

Master FS (bits)	16	16	16	16	16
Slave FS (bits)	16	32	64	128	256
Total Duration (μs)	89.7	100.37	121.7	169.7	265.7

CHAPTER 3

BASIC HEURISTIC ALGORITHMS FOR TELEGRAM SCHEDULING ON MVB

The main objective of this thesis is the computation of telegram schedules for MVB. In this chapter, the telegram scheduling problem is formally stated and different solution methods are proposed. Section 3.1 formulates the telegram scheduling problem on MVB and Section 3.2 develops a suitable integer linear programming (ILP) formulation in order to determine optimal MVB schedules. In addition, different basic heuristic algorithms for solving the telegram scheduling problem on MVB are proposed in Section 3.3. The different algorithms are evaluated in Section 3.4.

3.1 Problem Statement

Modern rail-vehicles exchange a large amount of periodic data over in-vehicle communication buses such as MVB. In order to achieve timely data exchange and to prevent data loss, the available bandwidth has to be used efficiently, which requires systematic scheduling approaches.

In order to formalize the scheduling problem on MVB, we define the set of t telegrams $T = \{T_1, ..., T_t\}$. Each telegram T_i has an individual period p_i and a duration d_i according to Table 2.1. In addition, we define the BP repetition as

$$r_i = \frac{p_i}{T_{\text{BP}}} \,. \tag{3.1}$$

to describe the difference between BPs where telegram T_i is repeated. Scheduling telegrams on MVB requires deciding about the offset o_i, which is the first BP in each MP, where T_i should be transmitted.

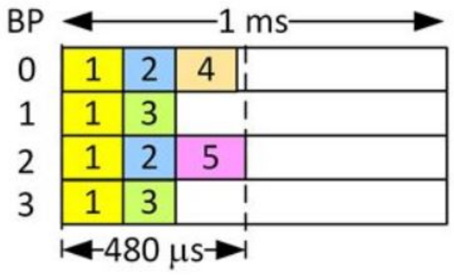
The example with five telegrams as shown in Table 3.1 helps to understand the scheduling problem.

Table 3.1: Sample Telegram Set For the MVB Scheduling Problem

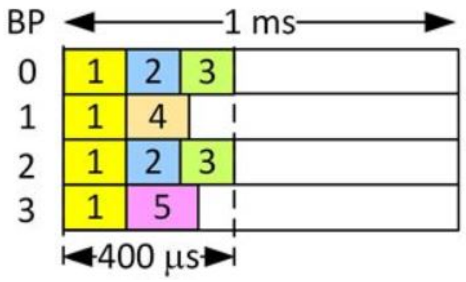
T_i	1	2	3	4	5
p_i [ms]	1	2	2	4	4
r_i	$1 = 2^0$	$2 = 2^1$	$2 = 2^1$	$4 = 2^2$	$4 = 2^2$
d_i [μs]	160	120	120	160	200

Figure 3.1 (a) shows a possible MVB schedule with the offsets $o_1 = 0$, $o_2 = 0$, $o_3 = 1$, $o_4 = 0$ and $o_5 = 2$. The required duration of the PP is $T_{\text{PP}} = 480\,\mu$s.

Figure 3.1 (b) shows an alternative schedule with the offsets like $o_1 = 0$, $o_2 = 0$, $o_3 = 0$, $o_4 = 1$ and $o_5 = 3$. In this case, the required duration of the PP is shorter with $T_{\text{PP}} = 400\,\mu$s.



(a) Possible Schedule



(b) Alternative Schedule

Figure 3.1: Offset Selection Importance

This example indicates that the choice of the offsets significantly affects the duration of the PP and hence the efficiency of the MVB schedule.

In order to evaluate the quality of an MVB schedule, we introduce several $U_{\text{BP}}^{\text{max}}$ is defined as the longest BP duration among all BPs. To this end, we write S_k for the set of telegrams that are scheduled in BP k. Then, U_{BP} is obtained as given in (3.3) based on each BP duration as stated in (3.2).

$$D_k = \sum_{T_i \in S_k} d_i, \quad k = 0, \dots, N_{\mathrm{MP}} - 1 \tag{3.2}$$

$$U_{\mathrm{BP}}^{\mathrm{max}} = \max_{k=0,\dots,N_{\mathrm{MP}}-1} \{D_k\} \tag{3.3}$$

It is desired to minimize the maximum BU $U_{\mathrm{BP}}^{\mathrm{max}}$ in order to achieve an efficient schedule. Specifically, decreasing $U_{\mathrm{BP}}^{\mathrm{max}}$ reduces the duration of the PP T_{PP}. That is, more time is left for the remaining phases in each BP. In addition, minimizing $U_{\mathrm{BP}}^{\mathrm{max}}$ makes it possible to determine if a given set of telegrams is schedulable on MVB or not. In particular, schedulability is given if and only if the minimum possible $U_{\mathrm{BP}}^{\mathrm{max}}$ is smaller than T_{BP}.

The performance metric for the evaluation is the minimum BU $U_{\mathrm{BP}}^{\mathrm{min}}$. Unlike the maximum BU, it is desired to maximize $U_{\mathrm{BP}}^{\mathrm{min}}$ because a small value of U_{BP} indicates that some BPs are not utilized well. In particular, a large difference between $U_{\mathrm{BP}}^{\mathrm{max}}$ and $U_{\mathrm{BP}}^{\mathrm{min}}$ means the MVB schedule is not balanced. The minimum BU is computed as

$$U_{\mathrm{BP}}^{\mathrm{min}} = \min\{D_k\}\,. \tag{3.4}$$

The standard deviation σ_{BP} of the BP durations is the third performance metric. It is the second most important performance metric for the evaluation after the maximum BU since it directly shows the balance of an MVB schedule. This performance metric is introduced based on the observation that an "ideal" MVB schedule would achieve equal BP durations.

$$D_{\mathrm{BP}}^{\mathrm{av}} = \sum_{i=1,\dots,t} \frac{d_i}{p_i} \cdot \frac{1}{N_{\mathrm{MP}}} \tag{3.5}$$

Since such ideal MVB schedule is generally not possible since telegrams have different durations and periods, σ_{BP} quantifies the deviation from $D_{\mathrm{BP}}^{\mathrm{av}}$ as

$$\sigma_{\mathrm{BP}} = \sum_{k=0}^{N_{\mathrm{MP}}-1} \frac{(D_k - D_{\mathrm{BP}}^{\mathrm{av}})^2}{N_{\mathrm{MP}}}\,. \tag{3.6}$$

Finally, the run-time of each proposed algorithm until finding a solution schedule is recorded to see which of the algorithms can produce a suitable MVB schedule in a practical time. Here, we note that MVB schedules are computed offline (that is, before system operation) such that run-times in the order of minutes are acceptable.

3.2 Optimal MVB Scheduling using Integer Linear Programming

In this section, we formulate the MVB telegram scheduling problem as an integer linear programming (ILP). That is, the objective function and constraints are linear also the variables are restricted to be integers. Hereby, we want to find optimal MVB schedules that minimize $U_{\text{BP}}^{\text{max}}$. ILP solution is the optimal solution if CPLEX terminates and is only the best solution until this point if CPLEX times out. We further note that the work in this section was published in [23].

The requirement for scheduling on MVB is to find a suitable offset o_i for each telegram T_i. In this context, it must hold that

1. $o_i \leq r_i - 1$, that is, the offset is smaller than the repetition,
2. telegram T_i is transmitted in all BP o_i + $k \cdot r_i$ for $k = 0, \ldots, \frac{N_{\text{MP}}}{r_i} - 1$.

In order to represent the offset of each telegram T_i, we introduce binary decision variables $x_{i,0}, \ldots, x_{i,r_i-1}$ such that the offset o_i has the value j if $x_{i,j} = 1$. That is, the selected offset for each telegram T_i is evaluated as

$$o_i = x_{i,0} \cdot 0 + x_{i,1} \cdot 1 + ... + x_{i,r_i-1} \cdot (r_i - 1). \tag{3.7}$$

Hereby, it must hold that exactly one of the decision variables for telegram T_i has the value 1, which is represented by the equality constraint:

$$\sum_{j=0}^{r_i-1} x_{i,j} = 1. \tag{3.8}$$

In addition, the integer decision variable (DV) T_{PP} is introduced for the duration of the PP. Any feasible schedule should have a maximum BU below the duration of the T_{BP} as a hard constraint so we get the following inequality constraint for all BPs:

$$\sum_{i=1}^{t} d_i \cdot x_{i,k \bmod r_i} \leq T_{\text{BP}} \ , \ 0 \leq k \leq N_{\text{MP}} - 1. \tag{3.9}$$

In (3.9), it is respected that a telegram which is scheduled with offset $o_i = j$ (implying that $x_{i,j} = 1$) appears in all BP k such that $j = k \bmod r_i$. The duration of each BP (T_{PP}) is then given by the sum of all telegrams that appear in that BP.

The decision vector x is given by the collection of all decision variables as shown below.

$$x = \begin{bmatrix} x_{1,0} \\ \vdots \\ x_{1,r_1-1} \\ \vdots \\ x_{t,0} \\ \vdots \\ x_{t,r_t-1} \\ T_{\text{BP}} \end{bmatrix}$$

Here, we recall that the variables $x_{i,j}$ for $i = 1, \ldots, t$ and $j = 0, \ldots, r_i - 1$ are binary, whereas T_{BP} is an integer variable that represents the used duration of the BP.

Since it is desired to minimize the duration of the PP, the objective function is given by:

$$J = \min_x \{T_{\text{PP}}\}. \tag{3.10}$$

Since, the duration of the PP must be smaller than the duration of the BP, we introduce the additional constraint:

$$T_{\text{PP}} \leq T_{\text{BP}}. \tag{3.11}$$

Together, the optimization problem can be formulated in the following vector/matrix form that can be used:

$$\min_x f \cdot x \tag{3.12}$$

such that

$$A_{\text{eq}} \cdot x = b_{\text{eq}}, \tag{3.13}$$

$$A \cdot x \leq b. \tag{3.14}$$

We next illustrate the ILP formulation using the telegram set in Table 3.1. For example the four decision variables $x_{4,0}, x_{4,1}, x_{4,2}, x_{4,3}$ are needed for telegram T_4 with repetition $r_4 = 4$ and the decision vector is

$$x = [x_{1,0}, x_{2,0}, x_{2,1}, x_{3,0}, x_{3,1}, x_{4,0}, ..., x_{4,3}, x_{5,0}, ..., x_{5,3}, T_{\text{BP}}]^T. \tag{3.15}$$

Then, the equality and inequality constraints in (3.13) and (3.14) are given as shown below.

$$\min_x \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot x$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} \cdot x = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 160 & 120 & 0 & 120 & 0 & 160 & 0 & 0 & 0 & 200 & 0 & 0 & 0 & -1 \\ 160 & 0 & 120 & 0 & 120 & 0 & 160 & 0 & 0 & 0 & 200 & 0 & 0 & -1 \\ 160 & 120 & 0 & 120 & 0 & 0 & 0 & 160 & 0 & 0 & 0 & 200 & 0 & -1 \\ 160 & 0 & 120 & 0 & 120 & 0 & 0 & 0 & 160 & 0 & 0 & 0 & 200 & -1 \end{bmatrix} \cdot x \leq \begin{bmatrix} 0 \\ 0 \\ 0 \\ T_{\text{BP}} \end{bmatrix}$$

Hereby, we note that the solution of the ILP is the decision vector x which needs to be translated to an actual schedule for MVB as in Figure 3.1. This task is performed by showing each telegram T_i with selected offset o_i in all the BP $o_i + k \cdot r_i$ for $k = 0, \ldots, \frac{N_{\text{MP}}}{r_i} - 1$, where the telegram appears.

3.3 Basic Heuristic Algorithms

As will be shown in the evaluation in Section 3.3, the ILP in the previous section cannot be solved when there is a large number of variables and a high BU. Because of this reason, a heuristic solution is required in order to be able to schedule such telegram sets. The defined problem in this work is similar to the bin packing problem. According to [26], this kind of problem can be solved by online and offline algorithms. Online algorithms, such as First Fit, Best Fit, etc. focus on the cases where items arrive back to back and each of them must be placed immediately, whereas the whole input is known before the placement process in the offline scheduling problem. Therefore, offline algorithms are investigated to achieve the scheduling goal in MVB.

In this context, the first-fit decreasing (FFD) algorithm operates by first sorting the items to be placed in decreasing order by their sizes and then inserting each item into the first available bin in the list with sufficient remaining space [27]. While FFD puts the incoming item in only one place, the item (that is, telegram) has to be placed to multiple places in the MVB scheduling problem as described in Section 3.1. This makes it much more difficult to solve the MVB scheduling problem since not only a single bin (that is, BP) but all BPs where a given telegram T_i will appear have to be evaluated. All the heuristics proposed in this thesis are constructed based on FFD but substantial modifications are made on the sorting of telegrams and the placement operations because each telegram has to be placed in different BPs. Instead of putting the item to the bin which has sufficient remaining space, the bins which have the maximum remaining space or which produce the most balanced schedule are searched in our heuristics.

All of the algorithms aim to decrease the maximum BU, to minimize the difference between the maximum and minimum BU in order to obtain a balanced and close-to-optimal MVB schedule. There are five different basic heuristic algorithms are proposed and each of them will be explained in detail.

For each algorithm, the same data set as shown in Table 3.2 will be used for illustration in order to compare their performance. Each telegram T_i has a unique number, the duration d_i in microseconds is specified according to the frame size and the period is p_i in milliseconds.

Table 3.2: Sample Telegram Data Set With Durations and Periods

T_i	1	2	3	4	5	6	7	8	9
p_i [ms]	1	2	2	2	4	4	4	4	4
r_i	$1 = 2^0$	$2 = 2^1$	$2 = 2^1$	$2 = 2^1$	$4 = 2^2$	$4 = 2^2$	$4 = 2^2$	$4 = 2^2$	$4 = 2^2$
d_i [μs]	89.7	121.7	265.7	169.7	169.7	100.37	265.7	89.7	100.37

As the common feature, all the algorithms take the set of t of telegrams $T = \{T_1, \ldots, T_t\}$ as input and produce an MVB schedule with the offsets o_i for each telegram T_i as an output.

3.3.1 Minimum Accumulated BP (MAB) Algorithm

As the main idea, the minimum accumulated BP (MAB) algorithm tries to minimize the accumulated duration of the BPs occupied by the current telegram T_i to be scheduled. At the beginning of the MAB, all BP durations are set to zero and offsets are cleared as initial settings. MAB sorts the telegrams in the list L according to increasing p_i and decreasing d_i in case of equal p_i. That is, it is desired to first schedule the telegrams which have a smaller p_i. The reason is that such telegrams have a considerable effect on the BP durations since they appear in many BPs. Less frequent telegrams are placed more easily without too much change in the whole schedule since they take place in a few BPs. Each telegram T_i from top of the sorted list is picked up and removed from L. Starting from the first offset, $o_i = 0$, the summation of BP durations in the MP is calculated, for each offset. Here, BP duration is checked whether it exceeds total allowed BP duration which is equal to T_{BP}. If one of them exceeds the allowed duration the current offset o_i is marked as a fail. Otherwise, the offset with the smallest sum C_j is selected and the telegram T_i is scheduled with that offset. Furthermore, D_j's and O are updated. The same procedure is repeated until there are no telegrams in the array, in sequence as in Algorithm 1.

In order to show the algorithm's ability in scheduling, consider the example with the 9 telegrams in Table 3.2. Each telegram T_i has individual period p_i and a duration d_i. Looking at the duration and individual period of the telegrams, the sorted list L turns out as shown in Table 3.3, for the given telegram set.

Table 3.3: Sorted List According to MAB

$L[i]$	1	2	3	4	5	6	7	8	9
T_i	1	3	4	2	7	5	6	9	8
d_i	89.7	265.7	169.7	121.7	265.7	169.7	100.37	100.37	89.7
p_i	1	2	2	2	4	4	4	4	4

Algorithm 1 MAB Pseudo Code

```
1: Input: $T$
2: Output: offset $o_i$ for each telegram $T_i$
3: Initialize: Set of assigned offsets $O = \emptyset$; $D_j = 0$ for $j = 0, \ldots, N_{\text{MP}} - 1$
4: Generate sorted list $L$ of telegrams according to increasing $p_i$ and decreasing $d_i$
   in case of equal $p_i$
5: while $L$ is not empty do
6:     Pick telegram $T_i$ from top of the list and remove $T_i$ from $L$
7:     for $0 \leq j \leq r_i - 1$ do
8:         $C_j = \sum_{k=0}^{N_{\text{MP}}/r_i - 1} D_{j+k \cdot r_i}$
9:         for $0 \leq k \leq \frac{N_{\text{MP}}}{r_i} - 1$ do
10:            if $D_{j+k \cdot r_i} + d_i > T_{\text{BP}}$ then
11:                $C_j = \infty$
12:    if $\min_{0 \leq j \leq r_i - 1}\{C_j\} < \infty$ then
13:        Select $o_i$ as the offset with the minimum $C_j$
14:        for $0 \leq k \leq \frac{N_{\text{MP}}}{r_i} - 1$ do
15:            $D_{j+k \cdot r_i} = D_{j+k \cdot r_i} + d_i$
16:    else
17:        Mark the algorithm as failed
```

Assuming the MP with 4 BPs, Figure 3.3, we next explain the first iterations of Algorithm 1 for the given telegram set. Firstly, the telegram T_1 is selected from L because it has the smallest p_i. Since the IP of the current telegram is equal to one, the telegram is placed to all BPs as shown in Figure 3.2 (left). All BPs are updated with the duration of the telegram. Secondly, MAB picks the telegram T_3 from L since it has the longest duration among the telegrams with $p_i = 2$. Since there are two possible offsets, $o_3 = 0$ and $o_3 = 1$, the algorithm calculates $\text{BP}_0 + \text{BP}_2$ and $\text{BP}_1 + \text{BP}_3$ back to back in order to find the minimum one. The sum of BUs are equal so MAB picks the smaller offset, $o_3 = 0$, which means T_3 is placed to BP_0 and BP_2 as shown in Figure 3.2 (middle). Again, the BP durations are updated. After that, the telegram T_4 is selected from L because it has the longest duration among the remaining telegrams that have $p_i = 2$. The offsets $o_4 = 0$ and $o_4 = 1$ are tried looking at the current BP durations. The smallest summation of BUs is achieved with offset $o_4 = 1$, hence T_4 is placed to BP$_1$ and BP$_3$ as shown in Figure 3.2 (right). All BP durations are updated. MAB continues for the remaining telegrams in the same way and the overall schedule shown in Figure 3.3 is obtained. For convenience, each telegram is assigned a color code for color representation in the graph. The resulting offsets are $o_1 = 0$, $o_2 = 1$, $o_3 = 0$, $o_4 = 1$, $o_5 = 2$, $o_6 = 1$, $o_7 = 0$, $o_8 = 1$ and $o_9 = 3$. The required duration of the periodic phase, T_{PP}, is equal to 621.1μs.

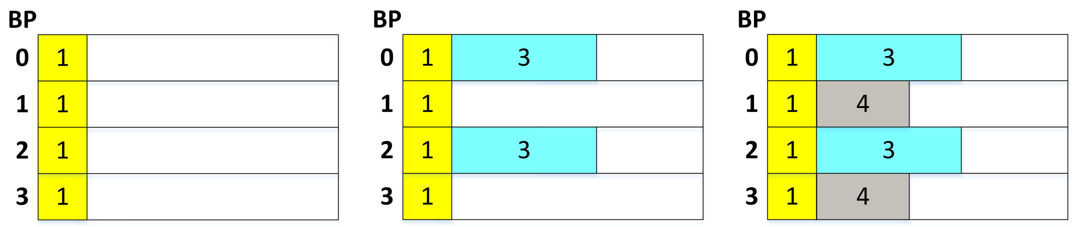


Figure 3.2: Placement of the First Three Telegrams in MAB

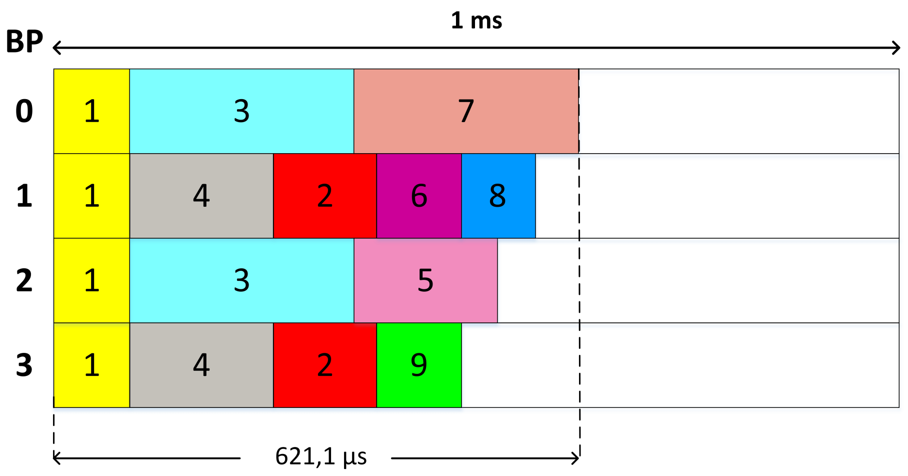


Figure 3.3: MVB Schedule with the MAB

3.3.2 Cursor and Flag (CF) Algorithm

CF sorts the telegrams in the list L according to increasing p_i and decreasing d_i in case of equal p_i. When a telegram T_i is picked up with period p_i, CF also picks the first two telegrams on the list T_x and T_y (if exist) which have the period $2 \times p_i$ in order to make the comparison between two telegrams depending on the duration. If (d_x+d_y) is larger than d_i and smaller than $2.1 \times d_i$, it holds that T_x and T_y together occupy a similar duration as T_i. In that case, starting from the first offset, $o_i = 0$, the summation of BP durations in the MP for each offset is calculated. Here, it is always checked if the BP duration exceeds the BP duration which is equal to T_{BP}. If one of them exceeds the allowed duration it is marked as a fail. Otherwise, the offset with the smallest value of the sums C_j is selected as the offset and the corresponding telegram(s), either T_i or T_x and T_y are scheduled with this offset. The same procedure is repeated until there are no telegrams in the array updating the D_j's and O, in sequence. If $d_x + d_y$ is smaller than d_i or larger than $2.1 \cdot d_i$, CF directly picks T_i from L and schedules it as in Algorithm 1.

In order to show the algorithm's ability in scheduling, consider the example with the 9 telegrams in Table 3.2. Each telegram T_i has IP p_i and a duration d_i. Looking at the duration and IP of the telegrams, the sorted list L is as shown in Table 3.4, for the mentioned telegram set.

Table 3.4: Sorted List According to CF

$L[i]$	1	2	3	4	5	6	7	8	9
T_i	1	3	4	2	7	5	6	9	8
d_i	89.7	265.7	169.7	121.7	265.7	169.7	100.37	100.37	89.7
p_i	1	2	2	2	4	4	4	4	4

Assuming the MP with 4 BPs, we next describe the first three iterations of the CF algorithm. Firstly, the telegram T_1 is selected from L because it has the smallest p_i. Then, the first two telegrams, T_3 and T_4, that have $2 \times p_i$ as IP are picked up. Because the sum of d_3 and d_4 is bigger than $2.1 \times d_i$, T_1 is directly selected as the telegram to be placed. Since the IP of the current telegram is equal to one, T_1 is placed to all BPs as shown in Figure 3.4 (left). BPs are updated with the duration of the telegram. Secondly, CF picks the telegram T_3 from L since it has the longest duration among the telegrams with $p_i = 2$. Then, the first two telegrams, T_7 and T_5, that have $2 \times p_i$ as IP are picked up. Since the sum of d_7 and d_5 is smaller than $2.1 \times d_i$, T_1 and greater than d_i, T_7 and T_5 are selected as the telegrams to be placed.

Algorithm 2 CF Pseudo Code

```
1:  **Input:** $T$
2:  **Output:** offset $o_i$ for each telegram $T_i$
3:  **Initialize:** Set of assigned offsets $O = \emptyset$; $D_j = 0$ for $j = 0, \ldots, N_{\text{MP}} - 1$
4:  Generate sorted list $L$ of telegrams according to increasing $p_i$ and decreasing $d_i$
    in case of equal $p_i$
5:  while $L$ is not empty do
6:    Pick telegram $T_i$ from top of the list and $T_x$, $T_y$ with $p_x = p_y = 2 \cdot p_i$
7:    if $T_i < (T_x + T_y) < 2.1 \times T_i$ then
8:      for $0 \leq j \leq r_x - 1$ do
9:        $C_j = \sum_{k=0}^{\frac{N_{\text{MP}}}{r_x}-1} D_{j+k \cdot r_x}$
10:       for $0 \leq k \leq \frac{N_{\text{MP}}}{r_x} - 1$ do
11:         if $D_{j+k \cdot r_x} + d_x + d_y > T_{\text{BP}}$ then
12:           $C_j = \infty$
13:     if $\min_{0 \leq j \leq r_x - 1}\{C_j\} < \infty$ then
14:       Select $o_i$ as the offset with the minimum $C_j$, remove $T_x$ and $T_y$ from $L$
15:       for $0 \leq k \leq \frac{N_{\text{MP}}}{r_x} - 1$ do
16:         $D_{j+k \cdot r_x} = D_{j+k \cdot r_x} + d_x + d_y$
17:     else
18:       Mark the algorithm as failed
19:   else
20:     for $0 \leq j \leq r_i - 1$ do
21:       $C_j = \sum_{k=0}^{\frac{N_{\text{MP}}}{r_i}-1} D_{j+k \cdot r_i}$
22:       for $0 \leq k \leq \frac{N_{\text{MP}}}{r_i} - 1$ do
23:         if $D_{j+k \cdot r_i} + d_i > T_{\text{BP}}$ then
24:           $C_j = \infty$
25:     if $\min_{0 \leq j \leq r_i - 1}\{C_j\} < \infty$ then
26:       Select $o_i$ as the offset with the minimum $C_j$, remove $T_i$ from $L$
27:       for $0 \leq k \leq \frac{N_{\text{MP}}}{r_i} - 1$ do
28:         $D_{j+k \cdot r_i} = D_{j+k \cdot r_i} + d_i$
29:     else
30:       Mark the algorithm as failed
```

Because there are two possible offsets, which are $o_5 = o_7 = 0$ and $o_5 = o_7 = 1$, the CF calculates $\text{BP}_0 + \text{BP}_2$ and $\text{BP}_1 + \text{BP}_3$ back to back in order to find the offset with the smallest sum of BPs. In this example, $\text{BP}_1 + \text{BP}_3$ is equal to $\text{BP}_0 + \text{BP}_2$. Hence, the selected telegrams are placed to $o_5 = o_7 = 0$, which means that T_7 and T_5 are placed to BP_0 and BP_2 as shown in Figure 3.4 (middle) also BP durations are updated. After that, the telegram T_3 is selected again from L because it was not placed in the previous placement. This time, the remaining two telegrams, T_6 and T_9, that have $2 \times p_i$ as IP are picked up. Because the sum of d_6 and d_9 is smaller than d_i, T_3 is selected as the telegram to be placed.There are two possible offsets, $o_3 = 0$ and $o_3 = 1$, the algorithm calculates $\text{BP}_0 + \text{BP}_2$ and $\text{BP}_1 + \text{BP}_3$ back to back in order to find the offset with the smallest sum of BP durations. Since sum of $\text{BP}_1 + \text{BP}_3$ is smaller than $\text{BP}_0 + \text{BP}_2$, T_3 obtains $o_3 = 1$ as shown in Figure 3.4 (right). Finally, all BP durations are updated.

The algorithm continues for the remaining telegrams in the same way and the schedule shown in Figure 3.5 is obtained. For convenience, each telegram is assigned a color code for color representation in the graph. The resulting offsets are $o_1 = 0$, $o_2 = 0$, $o_3 = 1$, $o_4 = 1$, $o_5 = 0$, $o_6 = 2$, $o_7 = 0$, $o_8 = 2$ and $o_9 = 2$. The required duration of the periodic phase is $T_{\text{PP}} = 646.8\,\mu$s.

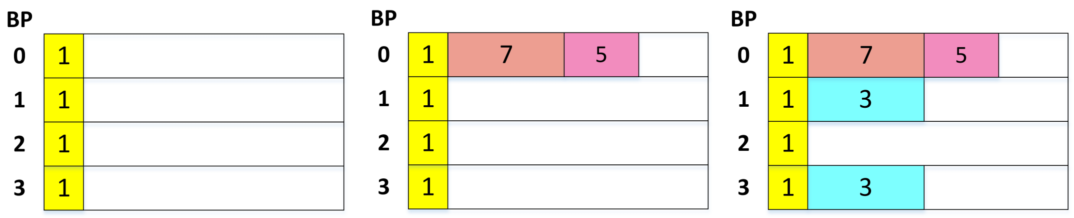


Figure 3.4: Placement of the First Three Telegrams in CF

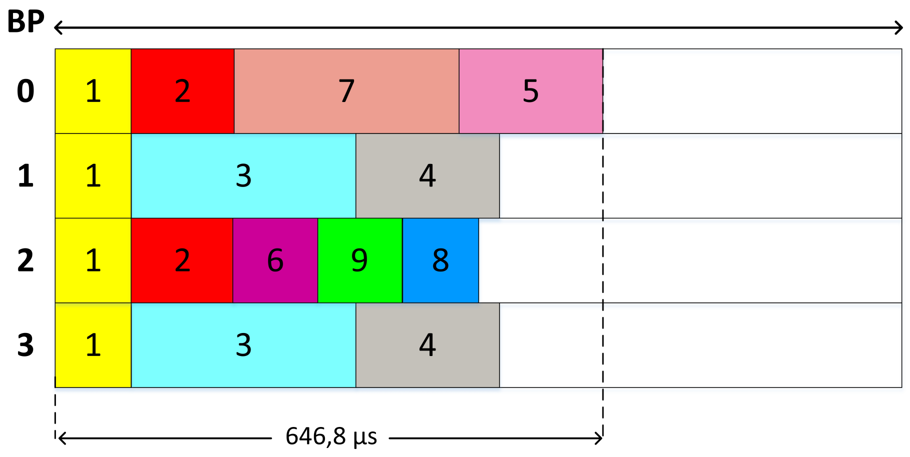


Figure 3.5: MVB Schedule with the CF

3.3.3 Randomized Minimum Accumulated BP (R-MAB) Algorithm

The randomized minimum accumulated BP (R-MAB) algorithm applies the same ideas as the MAB algorithm in Section 3.3.1 with a random organization of the list L. First, all telegrams in the data set are randomly shuffled and R-MAB puts them to the list L. It picks the first m telegrams in L, they are removed from the list and the algorithm sorts the m telegrams according to increasing p_i and decreasing d_i in case of equal p_i. The sorted telegrams are put on the list L_{shuffled} back to back. This process is repeated until there is no telegram in L. R-MAB tries to construct small sorted lists and combines all of the lists one after another. As a result, it is expected that scheduling will be more balanced. In light of this information, the algorithm is evaluated for $m = 5$ in this work.

MAB in Algorithm 1 is applied to L_{shuffled} as shown in Algorithm 3 and telegrams are scheduled according to that.

In order to show the algorithm's ability in scheduling, consider the example with the 9 telegrams in Table 3.2. Each telegram T_i has IP p_i and a duration d_i. Looking at the duration and IP of the telegrams sorted list L_{shuffled} occurs as shown in Table 3.5, for the mentioned telegram set.

Table 3.5: Sorted List According to R-MAB

$L_{\text{shuffled}}[i]$	1	2	3	4	5	6	7	8	9
T_i	3	4	2	5	9	1	7	6	8
d_i	265.7	169.7	121.7	169.7	100.37	89.7	265.7	100.37	89.7
p_i	2	2	2	4	4	1	4	4	4

Algorithm 3 R-MAB Pseudo Code

```
1: **Input:** $T$
2: **Output:** offset $o_i$ for each telegram $T_i$
3: **Initialize:** Set of assigned offsets $O = \emptyset$; $D_j = 0$ for $j = 0, \ldots, N_{\text{MP}} - 1$
4: Generate sorted list $L$ by randomly shuffling the telegrams and $L_{\text{shuffled}}$
5: **for** $1 \leq j \leq \lceil \frac{t}{5} \rceil$ **do**
6:     Get first five telegrams and remove all from $L$
7:     Sort the telegrams according to increasing $p_i$ and decreasing $d_i$ in case of equal $p_i$
8:     Put sorted telegrams to $L_{\text{shuffled}}$
9: Apply the MAB to $L_{\text{shuffled}}$
```

Assuming an macro period with 4 BPs, we illustrate the first steps of Algorithm 3. R-MAB sorts the telegrams after shuffling according to increasing p_i and decreasing d_i in case of equal p_i. R-MAB picks up each telegram in order. Firstly, the telegram T_3 is selected from L because it is the first telegram. Since the IP of the current telegram is equal to two, there are two possible offsets, $o_3 = 0$ and $o_3 = 1$. R-MAB calculates $\text{BP}_0 + \text{BP}_2$ and $\text{BP}_1 + \text{BP}_3$ back to back in order to find the offset with the smallest BP duration. Since all BP durations are initially equal to 0, R-MAB picks the smaller offset, $o_3 = 0$, which means T_3 is placed to BP_0 and BP_2 as shown in Figure 3.6 (left). Also, the BP durations are updated. Secondly, R-MAB picks the second telegram in L which is T_4. from L since it has the longest duration among the telegrams with $p_i = 2$. Because there are two possible offsets, $o_4 = 0$ and $o_4 = 1$, the algorithm calculates $\text{BP}_0 + \text{BP}_2$ and $\text{BP}_1 + \text{BP}_3$ back to back in order to find the offset. Since $\text{BP}_1 + \text{BP}_3$ is smaller than $\text{BP}_0 + \text{BP}_2$, the algorithm schedules T_4 with the offset

$o_4 = 1$. It indicates that T_4 is placed to BP_1 and BP_3 as shown in Figure 3.6 (middle) also BP durations are updated. After that, the third telegram T_2 in the L is picked up. Because the smallest sum of BUs is achieved with offset $o_2 = 0$, T_2 is placed to BP_0 and BP_2 as shown in Figure 3.6 (right) and all BP durations are updated. R-MAB continues for the remaining telegrams in the same way and the schedule shown in Figure 3.7 is obtained. For convenience, each telegram is assigned a color code for color representation in the graph. The resulting offsets are $o_1 = 0$, $o_2 = 1$, $o_3 = 0$, $o_4 = 1$, $o_5 = 0$, $o_6 = 3$, $o_7 = 1$, $o_8 = 2$ and $o_9 = 2$. The required duration of the periodic phase is $T_{PP} = 646.8\,\mu$s.

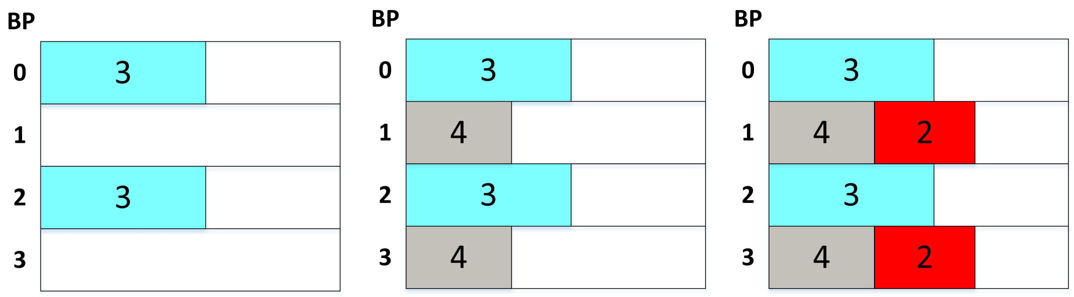


Figure 3.6: Placement of the First Three Telegrams in R-MAB

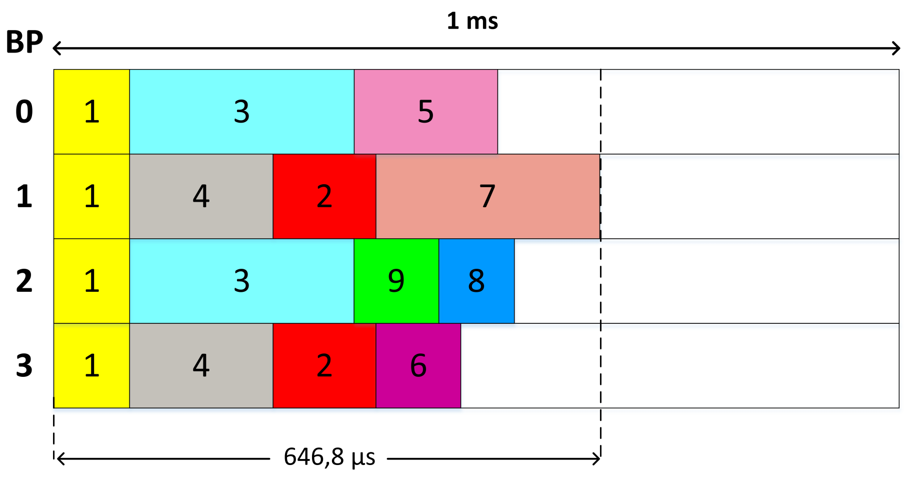


Figure 3.7: MVB Schedule with R-MAB

3.3.4 Minimum Longest BP (MLB) Algorithm

The previous algorithms decided on the most suitable offset based on the sum of the corresponding BP durations. Differently, the Minimum Longest BP (MLB) algorithm selects the offset that minimizes the maximum BP duration. Initially, MLB sorts the telegrams taking into account the value d_i / p_i from biggest to smallest and puts all of them into the list L. The reasoning is that the larger values of d_i / p_i occupy more space in the schedule looking at the overall duration. Hence, they affect the balance and required BP duration in the schedule more compared to smaller ones. Because of this reason, it is desired to place telegrams with a larger d_i / p_i when there is plenty of space in the schedule. Each telegram T_i from top of the sorted list is picked up. The maximum BP duration, D_j, is found in where the telegram should be repeated looking at the MP and repetition rate r_i for each offset. During the placement of each telegram, all the previous telegrams on the list are already scheduled. That is, each BP already can have telegrams and the duration of BPs varies depending on the duration of each telegram. At the end of the iteration, the offset o_i with the minimum D_j is selected and the telegram T_i is placed to this offset, D_j's and O are updated. The pseudo-code of the defined algorithm is shown in Algorithm 4.

In order to illustrate the algorithm's ability in scheduling, consider the example with the 9 telegrams in Table 3.2. Each telegram T_i has IP p_i and a duration d_i. Looking at the duration and IP of the telegrams sorted list L occurs as shown in Table 3.6, for the mentioned telegram set.

Table 3.6: Sorted List According to MLB

$L[i]$	1	2	3	4	5	6	7	8	9
T_i	3	1	4	7	2	5	6	9	8
Value $[\frac{d_i}{p_i}]$	132.85	89.70	84.85	66.43	60.85	42.43	25.09	25.09	22.43

Algorithm 4 MLB Pseudo Code

```
1: Input: $T$
2: Output: offset $o_i$ for each telegram $T_i$
3: Initialize: Set of assigned offsets $O = \emptyset$; $M_j = D_j = 0$ for $j = 0, \ldots, \frac{N_{\text{MP}}}{r_i} - 1$
4: Generate sorted list $L$ of telegrams according to decreasing $\dfrac{d_i}{p_i}$
5: while $L$ is not empty do
6:     Pick telegram $T_i$ from top of the list and remove $T_i$ from $L$
7:     for $0 \leq j \leq r_i - 1$ do
8:         $M_j = D_j$
9:         for $0 \leq k \leq \frac{N_{\text{MP}}}{r_i} - 1$ do
10:            if $D_{j+k \cdot r_i} > M_j$ then
11:                if $D_{j+k \cdot r_i} + d_i \leq T_{\text{BP}}$ then
12:                    $M_j = D_{j+k \cdot r_i}$
13:                else
14:                    $M_j = \infty$
15:    if $\min_{0 \leq j \leq r_i - 1}\{M_j\} < \infty$ then
16:        Select $o_i$ as the offset with the minimum $M_j$
17:        for $0 \leq k \leq \frac{N_{\text{MP}}}{r_i} - 1$ do
18:            $D_{j+k \cdot r_i} = D_{j+k \cdot r_i} + d_i$
19:    else
20:        Mark the algorithm as failed
```

Assuming the MP with 4 BPs, we describe the first steps of Algorithm 4. Firstly, the telegram T_3 is selected from L because it has the biggest d_i / p_i ratio. Due to the IP of the current telegram, there are two possible offsets, $o_3 = 0$ and $o_3 = 1$. The algorithm calculates the BPs durations; $D_0 = D_2 = 0\,\mu$s for $o_3 = 0$, $D_1 = D_3 = 0\,\mu$s for $o_3 = 1$. It picks the maximum BPs for each offsets as D_0, D_1 and $o_3 = 0$ is selected because of the maximum BP duration equality. So, T_3 is placed as shown in Figure 3.8 (left). BPs are updated with the duration of the telegram. Secondly, MLB picks the telegram T_1 from L since it has the second largest d_i / p_i ratio among the telegrams. Because the IP of T_1 is equal to one, the telegram is placed to all base cycles as shown in Figure 3.8 (middle). BPs are updated with the duration of the telegram. After that, the telegram T_4 is selected from L because it has third largest d_i / p_i ratio. Due to $p_i = 2$, the telegram can be scheduled with either $o_4 = 0$ or $o_4 = 1$. The algorithm calculates $D_0 = D_2 = 354.4\,\mu$s for $o_4 = 0$, $D_1 = D_3 = 89.7\,\mu$s for $o_3 = 1$. It picks the maximum values for each offsets as D_0 (354.4 μs), D_1 (89.7 μs) and selects the offset $o_4 = 1$ which has the minimum BP duration between D_0 and D_1. Therefore, T_4 is placed to BP$_1$ and BP$_3$ as shown in Figure 3.8 (right) and all BP durations are updated.

MLB continues for the remaining telegrams in the same way and the schedule shown in Figure 3.9 is obtained. For convenience, each telegram is assigned a color code for color representation in the graph. The resulting offsets are $o_1 = 0$, $o_2 = 0$, $o_3 = 0$, $o_4 = 1$, $o_5 = 3$, $o_6 = 3$, $o_7 = 1$, $o_8 = 2$ and $o_9 = 0$. The required duration of the periodic phase is T_{PP} = $577.47\,\mu$s.

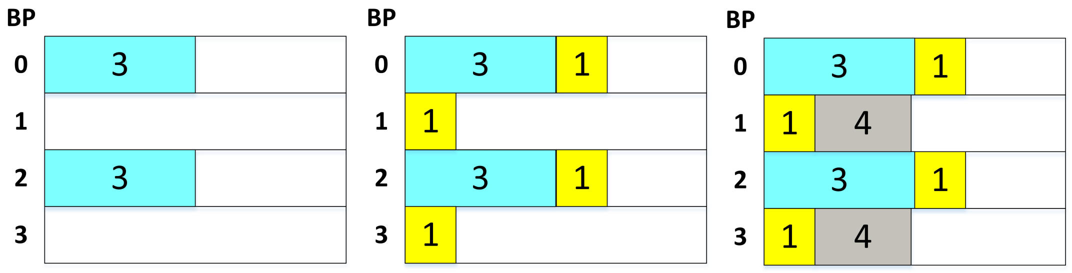


Figure 3.8: Placement of the First Three Telegrams in MLB

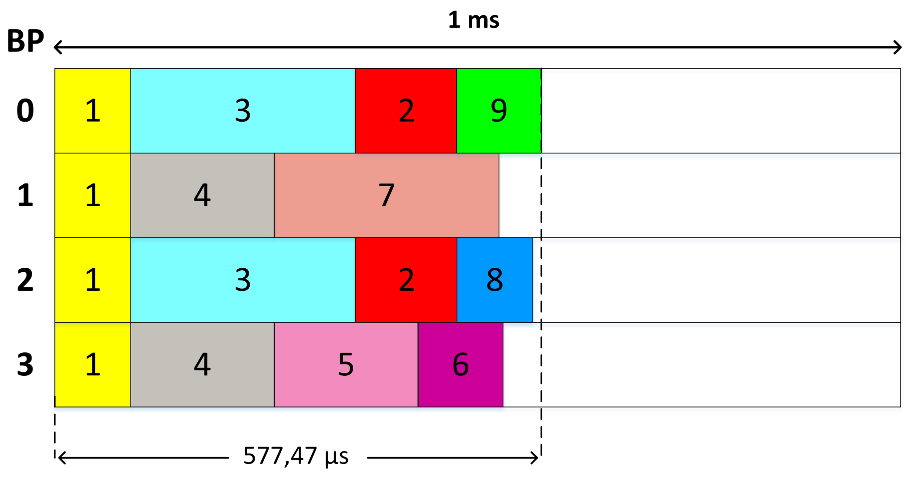


Figure 3.9: MVB Schedule with MLB

3.3.5 Scaled Average BP (SAB) Algorithm

The previous algorithms allow placing telegrams in BPs as long as the BP duration remains below the bound T_{BP}. Nevertheless, it was already observed in Section 3.1 that an ideal schedule for a given telegram set would fill all BPs up to an "average" duration of $D_{\text{BP}}^{\text{av}}$ in (3.5). Different from the other algorithms, SAB uses $D_{\text{BP}}^{\text{av}}$ as the target BP duration. Initially, SAB sorts all telegrams taking into account the value d_i / p_i from biggest to smallest with the same argument as before. Then, it puts all of them into a list L. Then, SAB performs operations according to a scaled average BP duration T_{av}, which is calculated with different coefficients γ in order to get the most balanced schedule as shown in the equation 3.16.

$$T_{\text{av}} = \frac{D_{\text{BP}}^{\text{av}}}{\gamma}, \quad \gamma \in \{0.75, 0.76, \ldots, 1.50\} \tag{3.16}$$

Telegram T_i from top of the sorted list L is picked up. Starting from the first offset, where T_{av} is not exceeded for all BPs where the selected telegram is repeated, the telegram is placed to that offset and the algorithm awaits the next telegram. If the SAB is not able to put the incoming telegram because of the average duration constraint, it does not place the telegram to the schedule and places it in another array L_2 that keeps the not-placed telegrams. After all the telegrams have gone through in the list L, SAB sorts the remaining telegrams in L_2. The telegrams in that list are sorted according to the smallest p_i, in the case of period equality the SAB checks d_i and the biggest duration takes the first place. After that, the algorithm calculates the sum of BP durations, where the incoming telegram is repeated for each offset option. The selected telegram is placed to the offset that has the minimum sum.

The algorithm is repeated for all values of T_{av} and the value that gives a more balanced schedule is selected as ideal value and the algorithm runs one more time with that value. SAB checks the schedule balance, calculating the standard deviation according to each BP duration for each scenario. The pseudo-code of SAB is shown in Algorithm 5.

Algorithm 5 SAB Pseudo Code

```
1: **Input:** $T$
2: **Output:** offset $o_i$ for each telegram $T_i$
3: **Initialize:** Set of assigned offsets $O = \emptyset$; $T_{\text{av}} = 0$, $B$, $\gamma$, $D_j = 0$ for $j = 0, \dots, N_{\text{MP}} - 1$
4: Generate sorted list $L$ of telegrams according to decreasing $\frac{d_i}{p_i}$
5: $T_{\text{av}} = (\sum_{i=1}^{t} \frac{p_i}{d_i \cdot \gamma})$
6: **while** $L$ is not empty **do**
7:     Pick telegram $T_i$ from top of the list and remove $T_i$ from $L$
8:     **for** $0 \leq j < r_i$ **do**
9:         $B =$ true
10:        **for** $0 \leq k \leq \frac{N_{\text{MP}}}{r_i} - 1$ **do**
11:            **if** $D_{j+k \cdot r_i} + d_i > T_{\text{av}}$ **then**
12:                $B =$ false
13:                **break**
14:        **if** $B =$ true **then**
15:            **break**
16:    **if** $B =$ false **then**
17:        Put telegram $T_i$ to another list $L_2$ and pick next telegram
18:    **else**
19:        Place telegram to current offset $o_i = j$ and pick the next telegram
20:        **for** $0 \leq k \leq \frac{N_{\text{MP}}}{r_i} - 1$ **do**
21:            $D_{j+k \cdot r_i} = D_{j+k \cdot r_i} + d_i$
22: **if** $L_2$ is not empty **then**
23:     Apply the MAB with $L_2$
```

In order to show the SAB ability in scheduling, consider the example with the 9 telegrams in Table 3.2. Each telegram T_i has IP p_i and a duration d_i. Looking at the duration and IP of the telegrams sorted list L occurs as shown in Table 3.7, for the mentioned telegram set.

Table 3.7: Sorted List According to SAB

$L[i]$	1	2	3	4	5	6	7	8	9
T_i	3	1	4	7	2	5	6	9	8
$[\frac{d_i}{p_i}]$	132.85	89.7	84.85	66.43	60.85	42.43	25.09	25.09	22.43

Assuming the MP with 4 BPs, we show for steps of Algorithm 5. Firstly, the telegram T_3 is selected from L because it has the biggest d_i / p_i ratio. p_3 is equal to 2 therefore there are two possible offsets, $o_3 = 0$ and $o_3 = 1$. SAB starts from the smaller one, and checks if the BP duration exceeds T_{av} or not. When d_3 is added to both BP$_0$ and BP$_2$, T_{av} is not exceeded for any BP. Then, T_3 is scheduled with the offset $o_3 = 0$ as shown in Figure 3.10 (1). BPs are updated with the duration of the telegram.

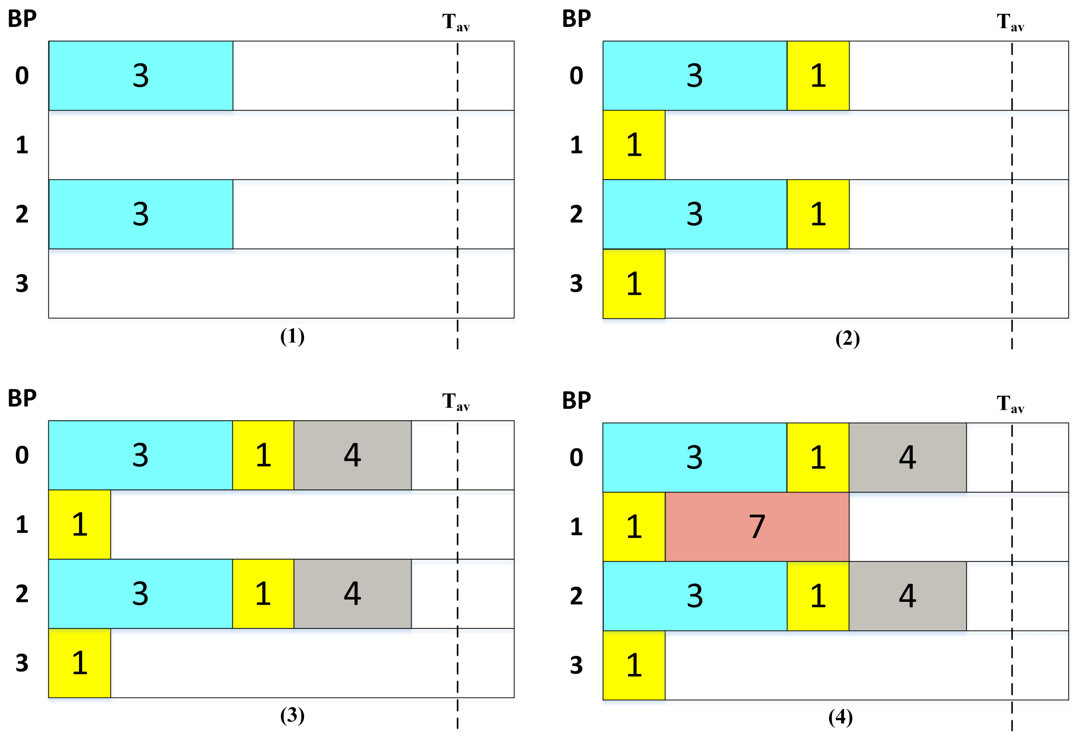


Figure 3.10: Placement of the First Three Telegrams in SAB

Secondly, SAB picks the telegram T_1 from L since it has the biggest d_i / p_i ratio among the remaining telegrams. Because the IP of T_1 is equal to one and none of the

BP duration exceeds T_{av}, the telegram is placed to all base cycles as shown in Figure 3.10 (2) also BP durations are updated. Third, SAB picks the telegram T_4 from L since it has the biggest d_i / p_i ratio among the remaining telegrams. Because the IP of T_4 is equal to two, SAB tries the two possible offsets back to back. For the first choice, $o_4 = 0$, the BP duration does not exceed T_{av}. Hence, the telegram is placed as shown in Figure 3.10 (3) also BP durations are updated. Fourth, SAB picks the telegram T_7 from L since it has the biggest d_i / p_i ratio among the remaining telegrams. Because the IP of T_7 is equal to four, SAB Tries the four possible offsets starting from the first one. When T_7 is placed with the offset $o_7 = 0$, BP$_0$ exceeds T_{av}. Then, SAB tries $o_7 = 1$ to schedule T_7. Because BP$_1$ does not exceed T_{av}, T_7 is placed as shown in Figure 3.10 (4) with the offset $o_7 = 1$.

Figure 3.11 shows the resulting MVB schedule that is constructed by Algorithm 5 with the offsets $o_1 = 0$, $o_2 = 1$, $o_3 = 0$, $o_4 = 0$, $o_5 = 3$, $o_6 = 1$, $o_7 = 1$, $o_8 = 2$ and $o_9 = 3$. The required duration of the periodic phase is T_{PP} = 577.47 μs.

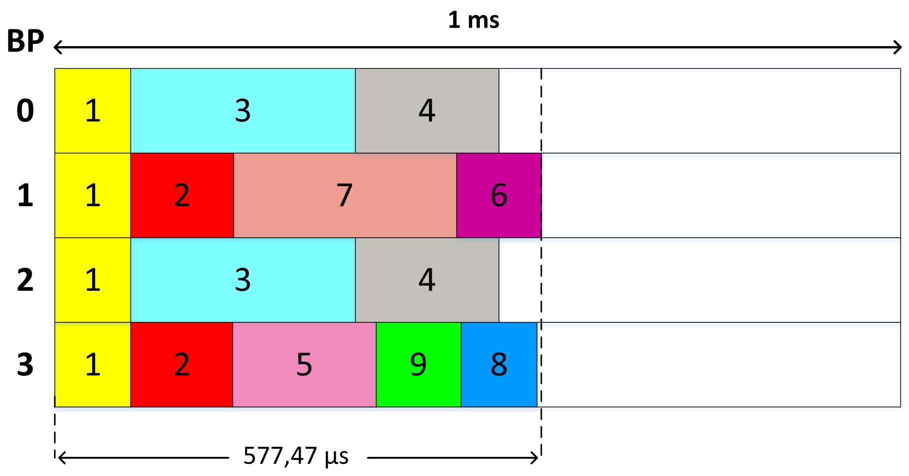


Figure 3.11: MVB Schedule with the SAB

For the given data set in Table 3.2, the best value of T_{av} is found for $\gamma = 0.9$. It gives the best σ_{BP} between all coefficient values γ from 0.75 to 1.50. Here, the T_{av} value is equal to 610.79 μs and it can be seen from Fig. 3.11 that none of the BPs exceeds T_{av}.

3.4 Evaluation of the Basic Heuristic Algorithms

There are five different basic heuristic algorithms as defined in Section 3.3. In this part of the thesis, the algorithms will be compared with each other in terms of the defined performance metrics in Section 3.1. Based on this evaluation, the algorithms with the best performance will be further refined in Chapter 4.

3.4.1 Environment

During this work, experiments are done on an HP ZBook 15 G3 workstation. The properties of the workstation are given below:

- Intel Core i7-6820HQ CPU @ 2.70GHz,
- 32 GB RAM,
- Windows 10 Pro OS, x64.

All the algorithms are written using MATLAB R2017a [28]. During the evaluation part, the ILP in Section 3.2 is solved by CPLEX and the performance of the basic heuristic algorithms will be compared with the optimal schedules obtained from the ILP formulation. ILOG CPLEX Optimization Studio, which is the product of IBM provides the fastest way to build efficient optimization models and state-of-the-art applications for the full range of planning and scheduling problems [29]. CPLEX v12.8 is used throughout the work. In CPLEX, the configurations below are applied as shown below:

- **options.MaxTime** which limits the run time is set to 1800 seconds,
- **options.Algorithm** is set to dual which is generally used for difficult problems.

The **cplexmilp** function, which is offered by CPLEX Optimization Studio is used for the evaluation. If the run time of CPLEX is equal to 1800 seconds, this test trial is counted as a time-out fail, since an optimal solution could not be found in a practical time. When the algorithm is not able to schedule the telegrams in the data set, the fail count is increased by one.

3.4.2 Data Set For Evaluation

Our test scenarios are based on randomly generated telegram sets that are constructed with the properties in Table 3.8. Telegram sets with different properties are obtained by taking into account the frequency of telegrams with certain individual periods. Four different frequency ranges are determined and results are obtained with reference to these.

Table 3.8: Data Set Classification and Telegram Sets

TS-1	More Frequent (r_i = 1, 2, 4)
TS-2	Normal (r_i = 8, 16, 32, 64)
TS-3	Less Frequent (r_i = 128, 256, 512, 1024)
TS-4	Equally distributed (according to percentage)

11 test cases for the more frequent telegrams, 13 test cases for the normal telegrams, 19 test cases for the less frequent telegrams and 9 test cases for the equally distributed telegrams are generated according to different values of the bandwidth utilization U_{MVB} are constructed. For each test case, there are twenty trials with the same BU but different telegrams looking at d_i and p_i. Hereby, the BU represents the part of the available duration that is used for transmitting periodic telegrams on MVB. For a given set of telegrams $T = \{T_1, \ldots, T_t\}$, BU is computed as

$$U_{\text{MVB}} = \sum_{i=1}^{t} \frac{d_i}{r_i}. \tag{3.17}$$

The performance of each algorithm will be evaluated according to data set classification. For each test case and every trial of it, maximum BU, minimum BU, average BU, standard deviation (σ_{BP}) and run-time parameters are recorded in order to use them in the evaluation.

All algorithms aim to schedule the telegrams in an optimal way but also they should not exceed T_{BP} for any basic period. If any of them is exceed the T_{BP}, it is assumed that the algorithm is not able to schedule telegrams for that trial according to test case parameters. In these test cases, the algorithm is marked as "failed" in the evaluation.

3.4.2.1 More Frequent Telegrams

The telegrams that have a small value of r_i are called More Frequent Telegrams. This concerns the telegrams with a small period such that a small number of telegrams already leads to a very high BU. Since the ILP formulation in this case only has to allocate a small number of telegrams, it is possible to determine an optimal solution despite the high BU.

In Table 3.9, the average telegram count and total BU for the different test cases (TC) is given.

Table 3.9: Average Telegram Counts and Total BUs in More Frequent Test Cases

TC-i	1	2	3	4	5	6	7	8	9	10	11
T Count	4	10	9	9	10	12	13	16	14	17	15
BU [%]	26.3	55.7	64.5	73.9	69.0	88.1	82.7	89.1	90.4	91.3	93.2

Table 3.10 shows the number of failed trials for the different algorithms and test cases. Minimizing the number of failed scheduling attempts is the second important goal after maximum BU reduction since a failed attempt means that no feasible MVB schedule could be found. Here, we note that a failed attempt of solving the ILP means that no solution exists for these test cases. In contrast, failed attempts of the heuristics potentially indicate the failure of finding a solution even a feasible MVB schedule exists.

Table 3.10: Fail Count for Each Algorithm in More Frequent Test Cases

Algorithm $\downarrow$ / TC $\rightarrow$	1	2	3	4	5	6	7	8	9	10	11
ILP	0	0	0	0	0	0	0	0	**2**	0	**1**
MAB	0	0	0	0	0	0	0	0	**3**	**7**	**5**
CF	0	0	0	0	0	**2**	0	0	**6**	**4**	**6**
R-MAB	0	0	0	0	0	**3**	0	**1**	**7**	**13**	**11**
SAB	0	0	0	0	0	0	0	0	**3**	**1**	**3**
MLB	0	0	0	0	0	0	0	0	**4**	**9**	**7**

After TC 5, both CF and R-MAB show failed attempts at a BU of approximately %88. When the BU is increased a little more, all algorithms including the ILP show fails in different trials although there are a few telegrams. Naturally, the ILP schedules the telegrams with a high rate of success according to results. Moreover, SAB schedules the telegram with a high rate of success. It performs the most similar performance among the basic heuristic algorithms to the ILP.

The maximum BU for each algorithm based on the TCs is given in Figure 3.12. The horizontal black lines in the plot show the lower bound for the maximum BP duration for each test case which is desired to be achieved in order to obtain the optimal MVB schedule. If at least one algorithm fails, all results in that trial are discarded among the twenty trials while calculating performance metrics.

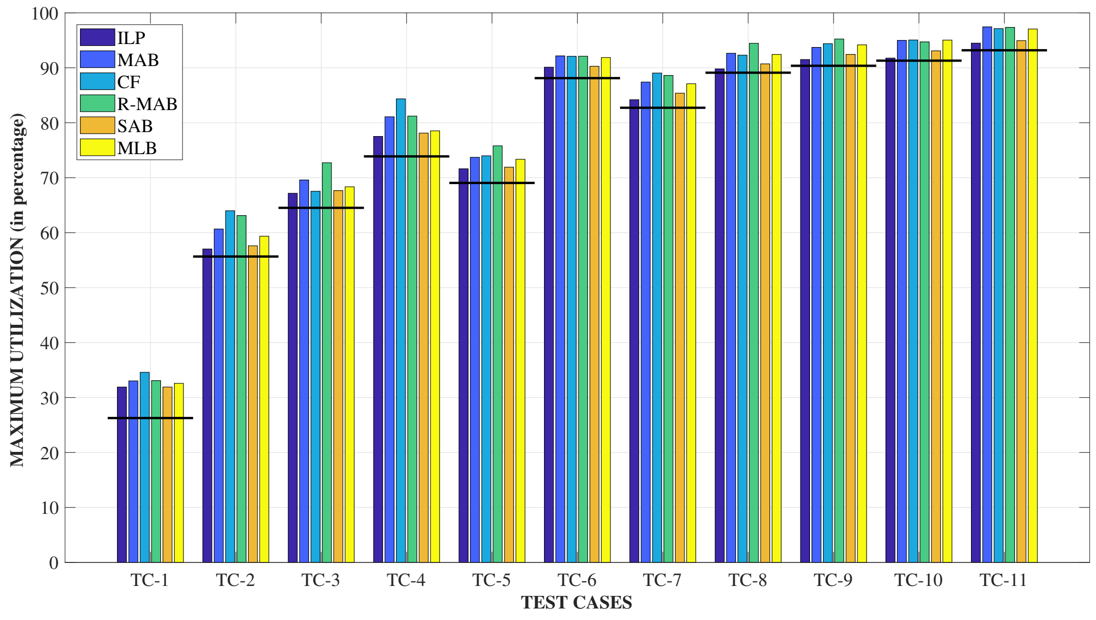


Figure 3.12: Maximum BU in More Frequent Test Cases

CF, MAB and R-MAB were never able to achieve the smallest maximum BU in any test case among the basic heuristics. Apart from the fact that these algorithms lead to a large number of failed attempts at high BUs, the also produced the worst maximum BU results compared to MLB and SAB. Here, it has to be noted that, in many cases, both MLB and SAB produce maximum BUs very close to the optimal ILP solution. Especially, SAB performs similar to or better than the ILP. For example, in TC 6, it can be seen that SAB has a smaller maximum BP duration than the ILP solutions.

This means that the ILP terminates without computing an optimal MVB schedule.

The minimum BU should be close to the maximum BU in order to obtain a balanced schedule. Minimum BU that is desired to be maximized for each algorithm based on the test cases is given in Figure 3.13. As seen in the figure, MLB schedules the telegrams as well as the ILP looking at the minimum BU. In some test cases, it is seen that this algorithm produced better results compared to ILP. The remaining four algorithms except for MLB schedule the test cases with smaller minimum BU but still, there is not too much difference between them.

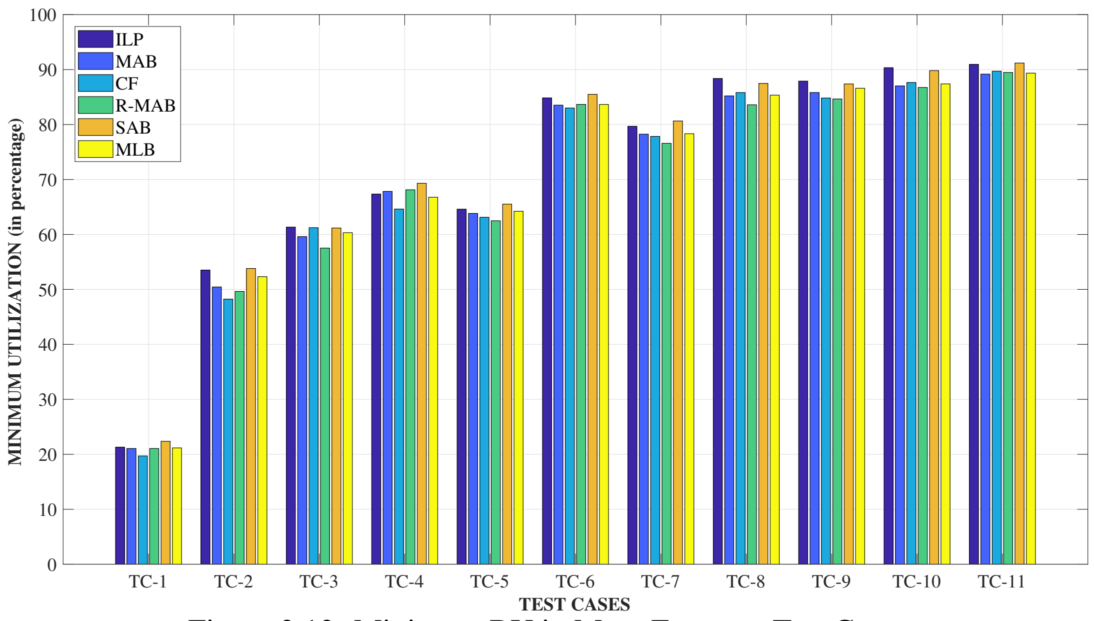


Figure 3.13: Minimum BU in More Frequent Test Cases

σ_{BP} directly states the balance of the schedule. If the duration of the BPs is close to each other, this parameter becomes smaller and $\sigma_{\mathrm{BP}} = 0$ is the best result for the schedule. In Figure 3.14, it is seen that CF, MAB and R-MAB have large values of σ_{BP}, which is not desired. SAB schedules the telegrams with the smallest σ_{BP} among heuristics and it produces balanced schedules similar to or better than the ILP. Additionally, MLB also gives good results compared to the remaining three heuristics even so it is not as successful as SAB.

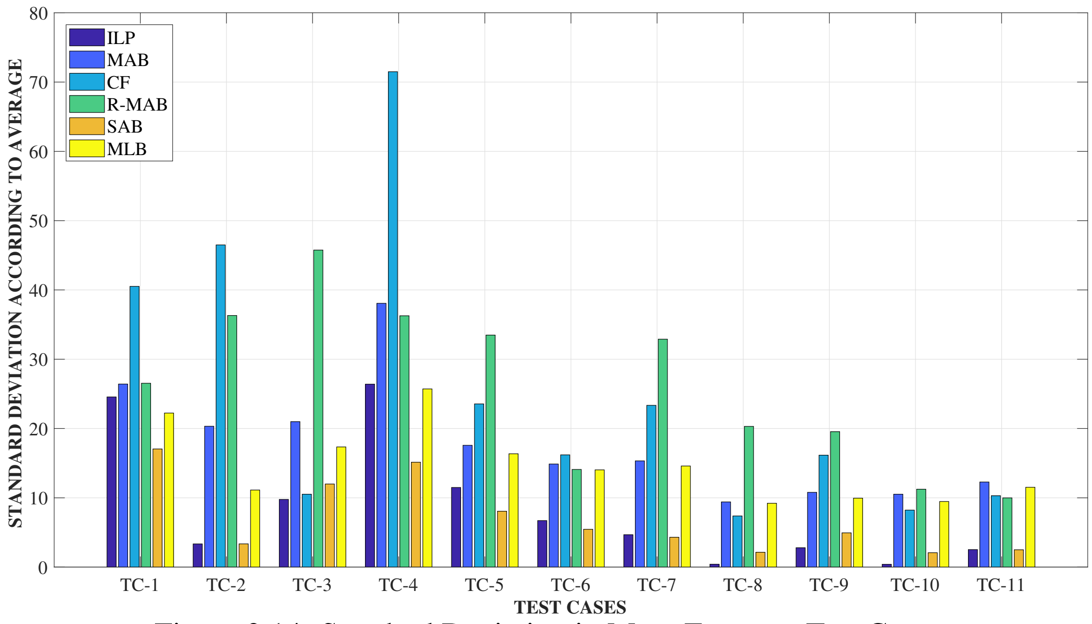


Figure 3.14: Standard Deviation in More Frequent Test Cases

The run-time is another important parameter that should be reduced by algorithms as one of the main goals. It is desired to minimize the run-time while decreasing the maximum BP and σ_{BP}. Depending on the combination of high BU and a large number of telegrams, the ILP starts to spend much more time to find an optimal solution. As seen in Table 3.11, all heuristic algorithms complete the scheduling in approximately the same run-time because of the small number of telegrams.

Table 3.11: Run-Time (ms) in More Frequent Test Cases

	TEST CASES										
	TC-1	**TC-2**	**TC-3**	**TC-4**	**TC-5**	**TC-6**	**TC-7**	**TC-8**	**TC-9**	**TC-10**	**TC-11**
ILP	2.7	38.8	36.5	63.7	43.5	92.0	395.0	119.0	76.2	173.0	54.7
MAB	0.9	0.3	0.3	0.3	0.3	0.3	0.5	0.3	0.3	0.3	0.3
CF	1.6	0.4	0.4	0.4	0.3	0.4	0.5	0.3	0.3	0.5	0.3
R-MAB	11.4	0.5	0.5	0.5	0.3	0.5	0.5	0.4	0.4	0.4	0.4
SAB	11.9	9.8	11.2	11.6	10.0	13.0	17.7	12.4	10.3	10.8	9.2
MLB	0.7	0.2	0.2	0.2	0.2	9.2	0.3	0.2	0.2	0.2	0.1

Total fail counts for each algorithm are also given in Table 3.12.

Table 3.12: Total Fail Counts in More Frequent Telegrams

Algorithm	**Fail Count**
MAB	15
CF	18
R-MAB	35
SAB	7
MLB	20

The winners for each test case according to different performance metrics are noted and the sum of them are given in Table 3.13 according to each heuristic.

Table 3.13: Winner of Test Cases in More Frequent Telegrams

	Performance Metrics			
	$\min\{U_{\text{BP}}^{\max} - U_{\text{BP}}^{\min}\}$	σ_{BP}	Run-Time	$\min U_{\text{BP}}^{\max}$
MAB	0	0	0	0
CF	1	1	0	1
R-MAB	0	0	0	0
SAB	10	10	0	10
MLB	0	0	11	0

Investigating Table 3.13 and 3.12, it is seen that SAB performs best and produces results closest to the ILP (sometimes better than the ILP). Except for run-time, it gets the highest score among the heuristics and for the run-time, it produces similar results as MLB. The remaining three algorithms show a considerably larger number of failed attempts and do not perform as well when looking at the different performance metrics.

3.4.2.2 Normal Telegrams

Normal telegrams are considered as telegrams with a medium period. Hence, a relatively large number of telegrams are needed to obtain a high BU. Nevertheless, the number of telegrams is still limited such that the ILP can be solved in many instances.

In Table 3.14, the average telegram count and total BU is given according to the 13 test cases for this test scenario.

Table 3.14: Average Telegram Counts and Total BUs in Normal Test Cases

TC-i	12	13	14	15	16	17	18	19	20	21	22	23	24
T Count	16	47	53	50	37	88	99	102	113	151	155	175	216
BU [%]	11.2	22.1	24.0	25.1	27.2	48.1	68.1	60.1	60.2	63.2	82.1	74.0	84.1

The count of the failed trials for each algorithm is given according to each test case in Table 3.15. Since the number of telegrams is still limited, most of the basic heuristics and the ILP can schedule all test instances. Only R-MAB shows unsuccessful trials in case of a high BU.

Table 3.15: Fail Count for Each Algorithm in Normal Test Cases

Algorithm $\downarrow$ / TC $\rightarrow$	12	13	14	15	16	17	18	19	20	21	22	23	24
ILP	0	0	0	0	0	0	0	0	0	0	0	0	0
MAB	0	0	0	0	0	0	0	0	0	0	0	0	0
CF	0	0	0	0	0	0	0	0	0	0	0	0	0
R-MAB	0	0	0	0	0	0	0	0	0	0	**6**	**3**	**16**
SAB	0	0	0	0	0	0	0	0	0	0	0	0	0
MLB	0	0	0	0	0	0	0	0	0	0	0	0	0

CF and R-MAB produce the highest maximum BU for most of the test cases as shown in Figure 3.15 which is not desired. MAB produces better results when the number of telegrams is increased to 150 and above. SAB and MLB are superior to the other algorithms looking at the results. These algorithms produce performance parameters very close to the ILP solution and they require only short run-times, which are negligible compared to the run-time of ILP as stated in Table 3.16.In particular, $U_{\text{BP}}^{\text{max}}$ for SAB in TC-24 is smaller than the corresponding value of the ILP solution, which indicates that the ILP cannot produce an optimal solution in case of a high BU.

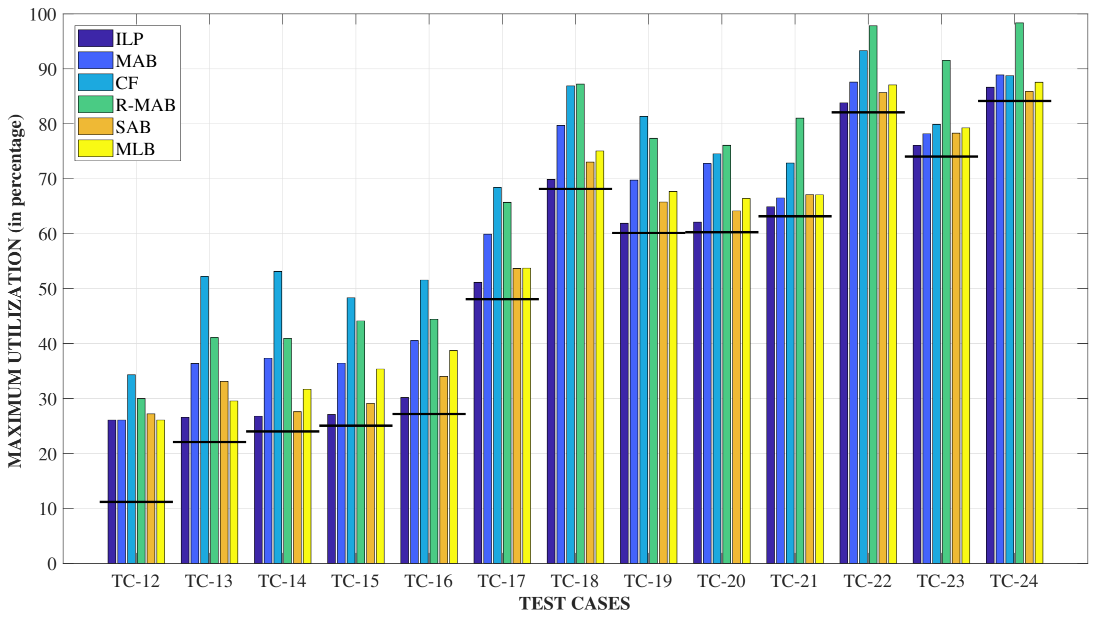


Figure 3.15: Maximum Utilization in Normal Test Cases

Looking at the minimum BU, all of the basic heuristic algorithms and ILP produce results that are very close to each other as shown in Figure 3.16.

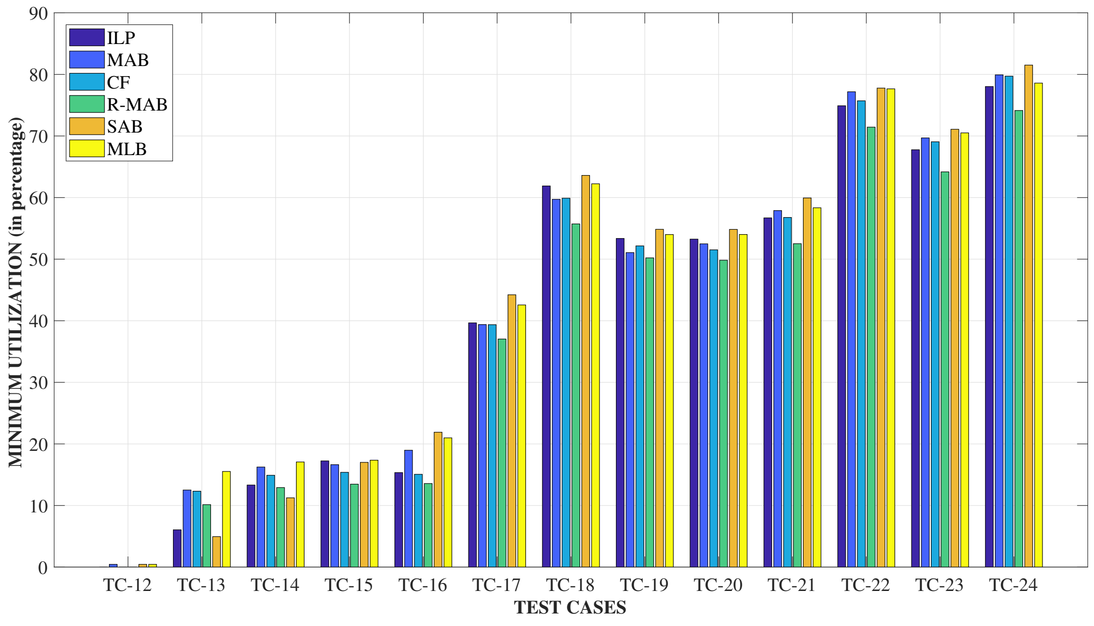


Figure 3.16: Minimum Utilization in Normal Test Cases

Because the minimum BU results are similar between all algorithms, σ_{BP} results for each test cases are directly related to maximum BU which is clearly seen looking at Figure 3.17.

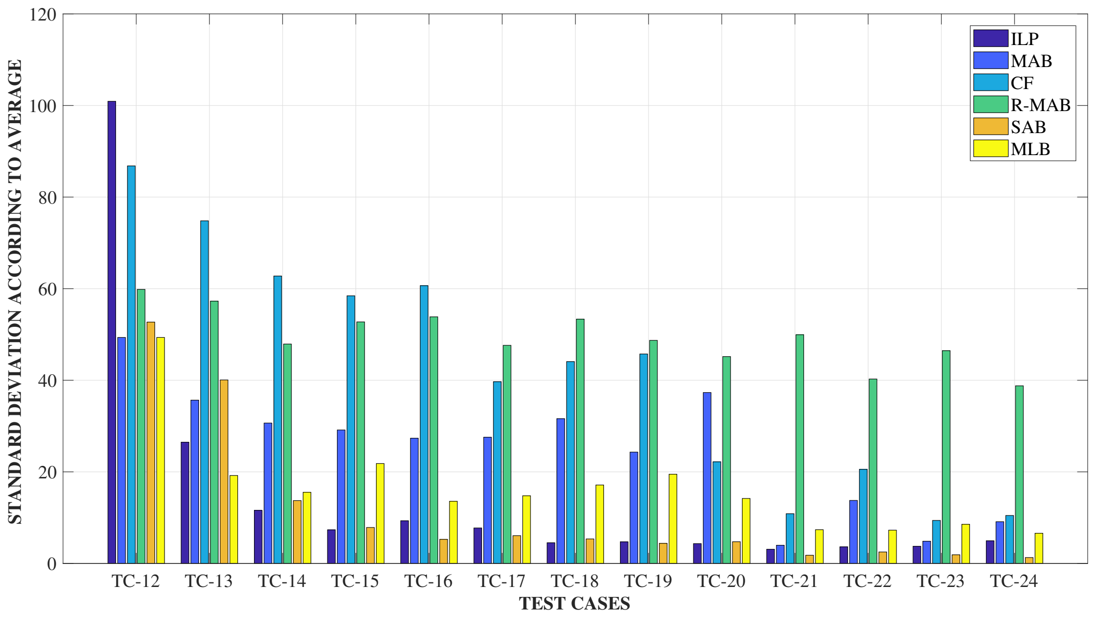


Figure 3.17: Standard Deviation in Normal Test Cases

As can be seen in Table 3.16, the basic heuristic algorithms run in a very short time, which is negligible compared to the ILP run-time. After TC-17, the run-time of the ILP is shown as timeout for the remaining test cases because the solver times out after 1800 seconds according to Table 3.17. When the run-time equals this value it indicates that an optimal solution is not found in the specified duration and it is counted as time failure. In the time-failure scenario, the trial is discarded in the evaluation for run-time. As the main result, Fig. 3.15 together with Table 3.16 indicate that the basic heuristic algorithms produce results close to the ILP solution in a much shorter time.

Table 3.16: Run-Time (ms) in Normal Test Cases

	TEST CASES												
	TC-12	**TC-13**	**TC-14**	**TC-15**	**TC-16**	**TC-17**	**TC-18**	**TC-19**	**TC-20**	**TC-21**	**TC-22**	**TC-23**	**TC-24**
ILP	$1 \cdot 10^3$	$6 \cdot 10^3$	$2 \cdot 10^3$	$66 \cdot 10^3$	$75 \cdot 10^3$	$380 \cdot 10^3$	TO	TO	TO	TO	TO	TO	TO
MAB	0.7	0.7	0.7	0.7	0.7	0.9	0.9	0.9	1.1	1.5	1.4	4.5	2.6
CF	0.7	0.8	0.8	0.8	0.7	1.0	1.1	1.1	1.3	1.5	1.4	5.4	3.4
R-MAB	0.8	0.9	1.0	0.9	0.8	1.3	1.3	1.4	1.6	1.9	1.9	8.4	4.3
SAB	33.3	33.4	35.0	34.0	32.7	43.8	46.1	46.1	49.2	59.1	59.0	131.0	138.9
MLB	0.6	0.6	0.5	0.5	0.5	0.7	0.7	0.7	0.8	0.9	0.9	2.7	2.1

Results show that ILP could not find the optimal solution when the BU is higher than %50 and telegram count is bigger than 100 according to test cases. Except for R-MAB which is failed for some test cases, other basic heuristics are very good at run-time performance compared to ILP.

Table 3.17: ILP Time Failure Count For Normal Test Cases

TC	12	13	14	15	16	17	18	19	20	21	22	23	24
Failures	0	0	**7**	**8**	**2**	**13**	**20**	**20**	**20**	**20**	**20**	**20**	**20**

The winners for each test case according to different performance metrics are noted and summation of them are given in Table 3.18 according to each heuristics.

Table 3.18: Winner of Test Cases in Normal Telegrams

	Performance Metrics			
	$\min\{U_{\text{BP}}^{\max} - U_{\text{BP}}^{\min}\}$	σ_{BP}	Run-Time	$\min U_{\text{BP}}^{\max}$
MAB	1	0	0	2
CF	0	0	0	0
R-MAB	0	0	0	0
SAB	8	9	0	8
MLB	3	2	11	2

Total fail counts for each algorithm are also given in Table 3.19.

Table 3.19: Total Fail Counts in Normal Telegrams

Algorithm	**Fail Count**
MAB	0
CF	0
R-MAB	25
SAB	0
MLB	0

Using Table 3.18 and Table 3.19, it is seen that SAB performs best and produces results closest to ILP as in Section 3.4.2.1. In addition, most of the cases, the ILP could not find an optimal solution and times out. On the other hand, MLB and SAB can always find a feasible schedule and they produce results close to ILP in a very short time. Except for the run-time, SAB gets the highest score. While MAB and CF do not fail, they could not produce results close to the ILP. R-MAB shows too many failed trials and performs the worst for the Normal Telegram Data Sets.

3.4.2.3 Less Frequent Telegrams

Less frequent telegrams consider telegrams with a small period such that a very large number of telegrams is needed to obtain a bus utilization above %20 as seen in Table 3.20, that gives the average telegram count and total BU. It is expected that finding an optimal MVB schedule is difficult for telegram sets with a very large number of telegrams with large periods since a large number of DVs are needed in the ILP formulation.

Table 3.20: Average Telegram Counts and Total BUs in Less Frequent Test Cases

TC-i	25	26	27	28	29	30	31	32	33	34
T Count	45	94	91	83	79	257	274	281	333	368
BU [%]	2.2	3.6	4.0	4.0	4.2	10.5	16.0	14.0	14.0	13.0
TC-i	35	36	37	38	39	40	41	42	43	
T Count	532	601	669	685	716	1931	2313	2571	3343	
BU [%]	21.0	22.0	23.0	24.0	25.0	60.0	72.0	80.0	70.0	

Table 3.21: Fail Count for Each Algorithm in Less Frequent Test Cases

Algorithm $\downarrow$ / TC $\rightarrow$	25	26	27	28	29	30	31	32	33	34
ILP	0	0	0	0	0	0	0	0	0	0
MAB	0	0	0	0	0	0	0	0	0	0
CF	0	0	0	0	0	0	0	0	0	0
R-MAB	0	0	0	0	0	0	0	0	0	0
SAB	0	0	0	0	0	0	0	0	0	0
MLB	0	0	0	0	0	0	0	0	0	0
Algorithm $\downarrow$ / TC $\rightarrow$	35	36	37	38	39	40	41	42	43	
ILP	0	0	0	0	0	0	0	0	0	
MAB	0	0	0	0	0	0	0	0	0	
CF	0	0	0	0	0	0	0	0	0	
R-MAB	0	0	0	0	0	0	0	**14**	0	
SAB	0	0	0	0	0	0	0	0	0	
MLB	0	0	0	0	0	0	0	0	0	

The fail count for each algorithm is given according to each test case in Table 3.21. Because the total BU is less than %25 for the test cases between TC-25 and TC-39 depending on the high r_i, no algorithm fails, although there is a large number of telegrams. After TC-39, the total BU is increased and all of the algorithms scheduled the telegrams in data set without any fail except R-MAB.

Between TC-40 and TC-43, ILP produces the worst result for the maximum BU minimization because it could not find an optimal solution in the selected maximum time because of the large number of telegrams (and hence DVs) as shown in Table 3.23. Discarding these test cases, CF and R-MAB produce worse schedules than the other algorithms, looking at the variables that are given in Figure 3.18. The remaining three algorithms schedule the telegrams close to ILP in terms of maximum BU.

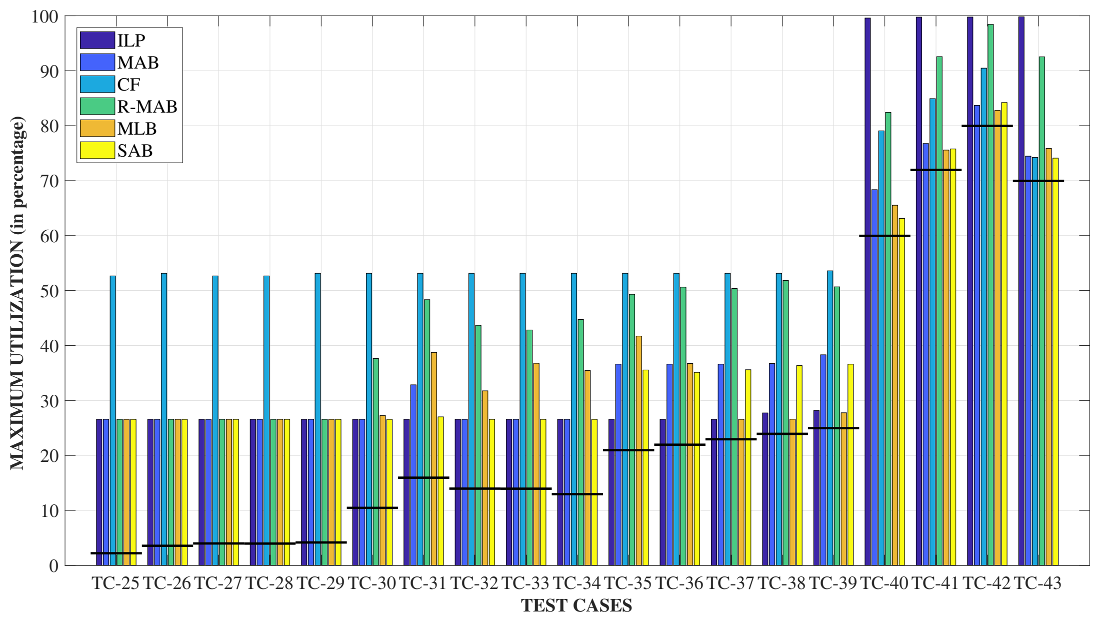


Figure 3.18: Maximum Utilization in Less Frequent Test Cases

Considering that a high minimum BU is desired, all of the basic heuristic algorithms perform better than ILP as seen in Figure 3.19. The basic heuristics schedule the telegrams very similar to each other in terms of minimum BU.

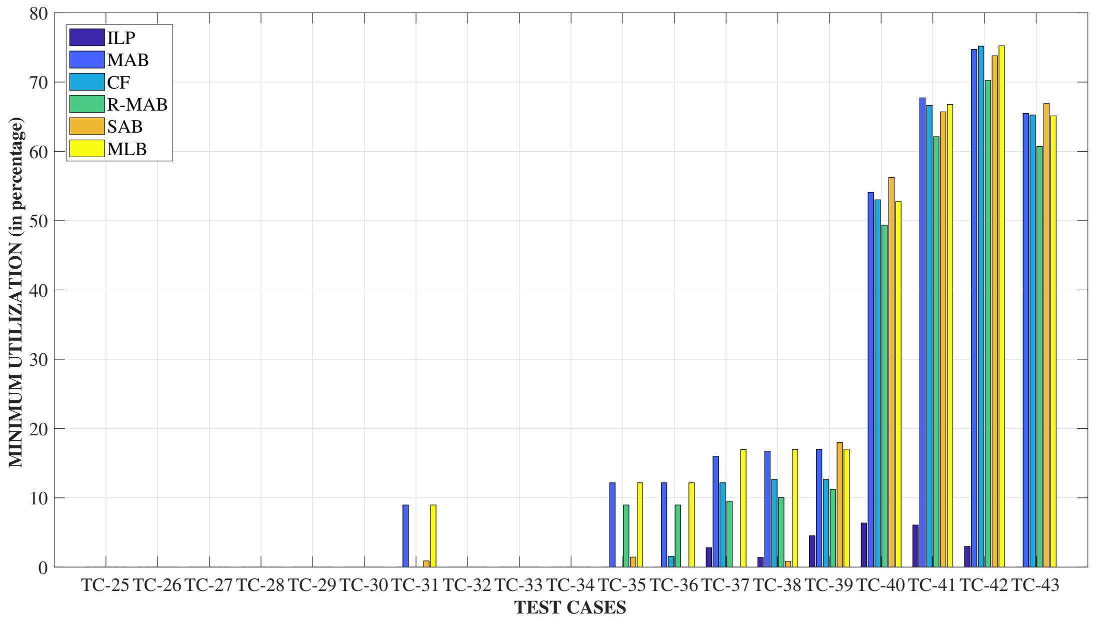


Figure 3.19: Minimum Utilization in Less Frequent Test Cases

Since r_i is high, the difference between the maximum and minimum BU is large. As a result of this, relatively large values of σ_{BP} are obtained as demonstrated in Figure 3.20. Depending on this difference, σ_{BP} is decreased as the test case number is increased. σ_{BP} for the ILP solution is getting significantly larger after TC-39 because the optimal solution is not found in the allocated time.

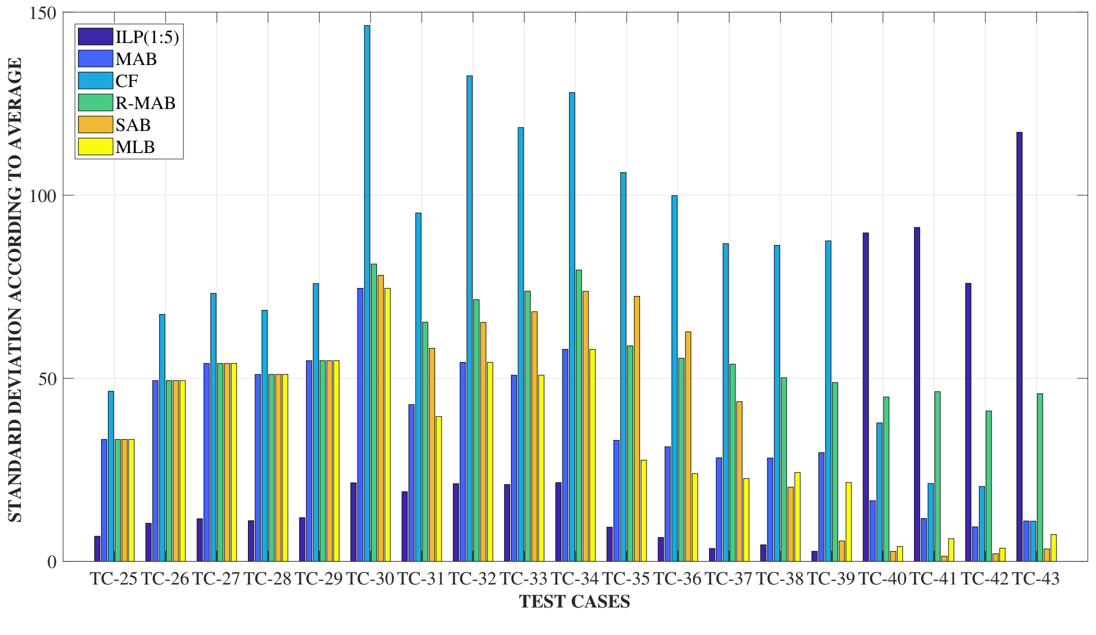


Figure 3.20: Standard Deviation in Less Frequent Test Cases

As seen in Table 3.22, the ILP consumes too much time when the BU is high and there is a large number of telegrams. Indeed, it can not find an optimal solution after some point. The described basic heuristics perform the schedule computation in a short time, below a second except for SAB whose run time can rise to 20 seconds, and their scheduling performances are better than or similar to the ILP.

Table 3.22: Run-Time (ms) in Less Frequent Test Cases

	TEST CASES									
	TC-25	**TC-26**	**TC-27**	**TC-28**	**TC-29**	**TC-30**	**TC-31**	**TC-32**	**TC-33**	**TC-34**
ILP	$6{\cdot}10^3$	$18\cdot10^3$	$17{\cdot}10^3$	$15{\cdot}10^3$	$9{\cdot}10^3$	$23{\cdot}10^3$	$19\cdot10^3$	$19\cdot10^3$	$25\cdot10^3$	$57\cdot10^3$
MAB	7.0	8.2	7.9	7.7	7.7	11.0	10.9	11.4	12.9	14.1
CF	6.9	8.1	8.1	8.0	7.4	11.2	11.5	12.2	13.7	14.7
R-MAB	7.4	8.7	8.3	8.1	7.9	12.3	12.4	13.6	14.4	15.7
SAB	390	571	558	554	513	740	627	647	762	900
MLB	5.7	6.6	6.6	6.6	6.2	9.0	9.1	9.2	10.5	11.4
	TC-35	**TC-36**	**TC-37**	**TC-38**	**TC-39**	**TC-40**	**TC-41**	**TC-42**	**TC-43**	
ILP	$112{\cdot}10^3$	$334\cdot10^3$	$527{\cdot}10^3$	TO	TO	TO	TO	TO	TO	
MAB	18.5	104.6	149.8	217.7	55.9	155.2	176.2	220.1	325.9	
CF	19.9	77.3	99.5	154.0	59.1	181.5	214.2	268.1	355.7	
R-MAB	20.7	173.3	181.9	201.5	62.4	166.5	193.5	244.3	315.7	
SAB	$1{\cdot}10^3$	$1{\cdot}10^3$	$2\cdot10^3$	$2\cdot10^3$	$3\cdot10^3$	$11\cdot10^3$	$12\cdot10^3$	$15\cdot10^3$	$21\cdot10^3$	
MLB	15.0	24.3	71.5	101.4	43.7	135.3	152.2	187.9	314.0	

Table 3.23: ILP Time Failure Count For Less Frequent Test Cases

TC	25	26	27	28	29	30	31	32	33	34
Failures	0	0	0	0	0	0	0	0	0	0
TC	35	36	37	38	39	40	41	42	43	
Failures	0	**1**	0	**20**	**20**	**20**	**20**	**20**	**20**	

It is observed that finding an optimal MVB schedule is difficult for telegram sets with a very large number of telegrams with large periods for ILP. The winners for each test case according to different performance metrics are noted and their summation is given in Table 3.24 according to each heuristic.

Table 3.24: Winner of Test Cases in Less Frequent Telegrams

	Performance Metrics			
	$\min\{U_{\text{BP}}^{\max} - U_{\text{BP}}^{\min}\}$	σ_{BP}	Run-Time	$\min U_{\text{BP}}^{\max}$
MAB	11	4	0	9
CF	0	0	0	0
R-MAB	5	1	0	5
SAB	7	6	0	10
MLB	16	10	19	14

The total fail counts for each algorithm are also given in Table 3.25.

Table 3.25: Total Fail Counts in Less Frequent Telegrams

Algorithm	Fail Count
MAB	0
CF	0
R-MAB	14
SAB	0
MLB	0

Using Table 3.24 and Table 3.25, it is seen that MLB performs best and produces results closest to ILP. In addition, it does not show any unsuccessful trials, whereas the ILP has too many time outs because of a large number of telegrams. The remaining heuristics also performs well and schedule the telegrams similar according to performance metrics. Except for R-MAB none of them fails in this test scenario.

3.4.2.4 Equal Distributed Telegrams

Equally distributed telegrams according to the percentage of occurrence consider telegram sets with telegrams of all possible periods and a similar BU per period. In Table 3.26, average telegram count and total BU is given for each test case.

Table 3.26: Telegram Count and Total BU for Equal Distributed Telegrams

TC-i	44	45	46	47	48	49	50	51	52
T Count	68	135	203	272	543	686	819	956	1095
BU [%]	2.7	6.4	11.2	14.9	33.9	47.7	57.0	66.8	75.8

Except for R-MAB and CF, all of the basic heuristics were able to schedule the telegrams sets in this test scenario without any failed attempt as stated in Table 3.27.

Table 3.27: Fail Count for Each Algorithm in Equal Distributed Test Cases

Algorithm $\downarrow$ / TC $\rightarrow$	44	45	46	47	48	49	50	51	52
ILP	0	0	0	0	0	0	0	0	0
MAB	0	0	0	0	0	0	0	0	0
CF	0	0	0	0	0	0	0	**3**	**17**
R-MAB	0	0	0	0	0	0	0	**5**	**18**
SAB	0	0	0	0	0	0	0	0	0
MLB	0	0	0	0	0	0	0	0	0

Even though the ILP solver could not find an optimal solution in most of the test cases according to Table 3.28, it still gives comparable results for the maximum BU as long as the BU is limited. MAB, MLB, and SAB give close BU to ILP and they achieve it in a significantly shorter run-time. CF and R-MAB produce much higher maximum BU which is an undesirable situation as shown in Figure 3.21.

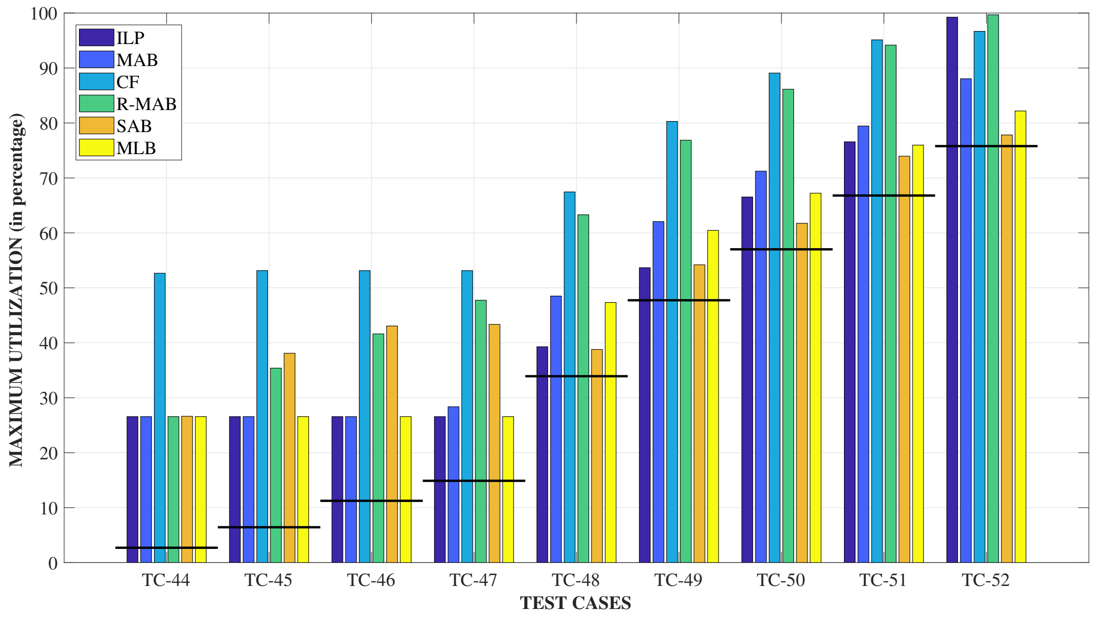


Figure 3.21: Maximum Utilization in Equally Distributed Test Cases

Figure 3.22 shows that all of the basic algorithms perform better than ILP in all of the test cases. Because of the time out of the ILP, it produces relatively small minimum BU values.

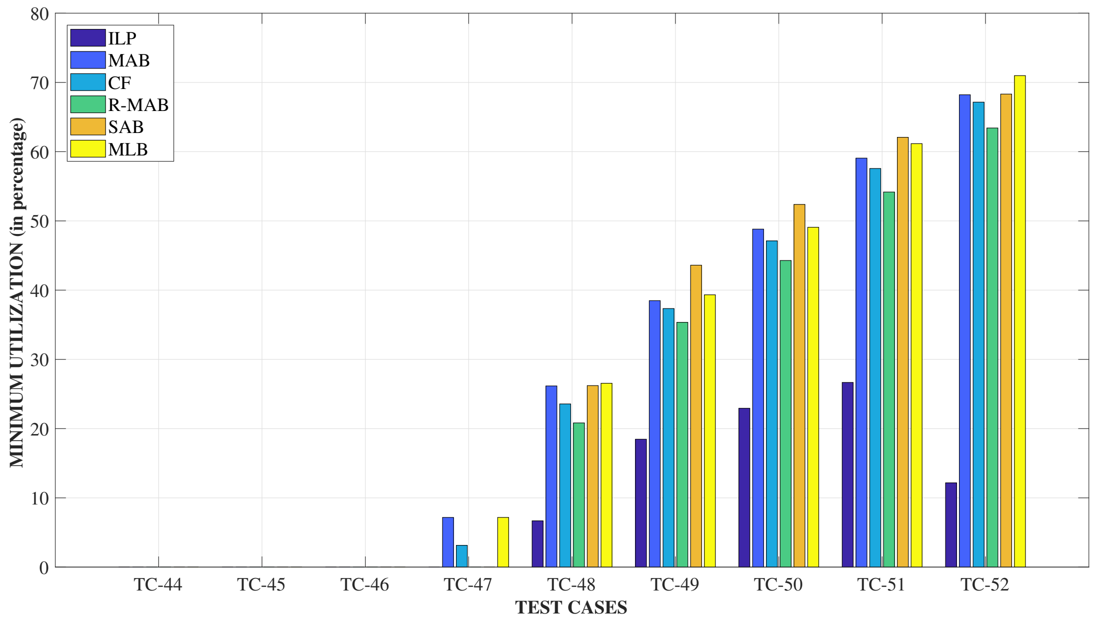


Figure 3.22: Minimum Utilization in Equal Distributed Test Cases

Except for R-MAB and CF, the remaining three basic heuristic algorithms give a smaller σ_{BP} compared to the ILP solution, which is desired. Especially SAB gives good results under high BU and a large number of telegrams with reference to TC-48 and later.

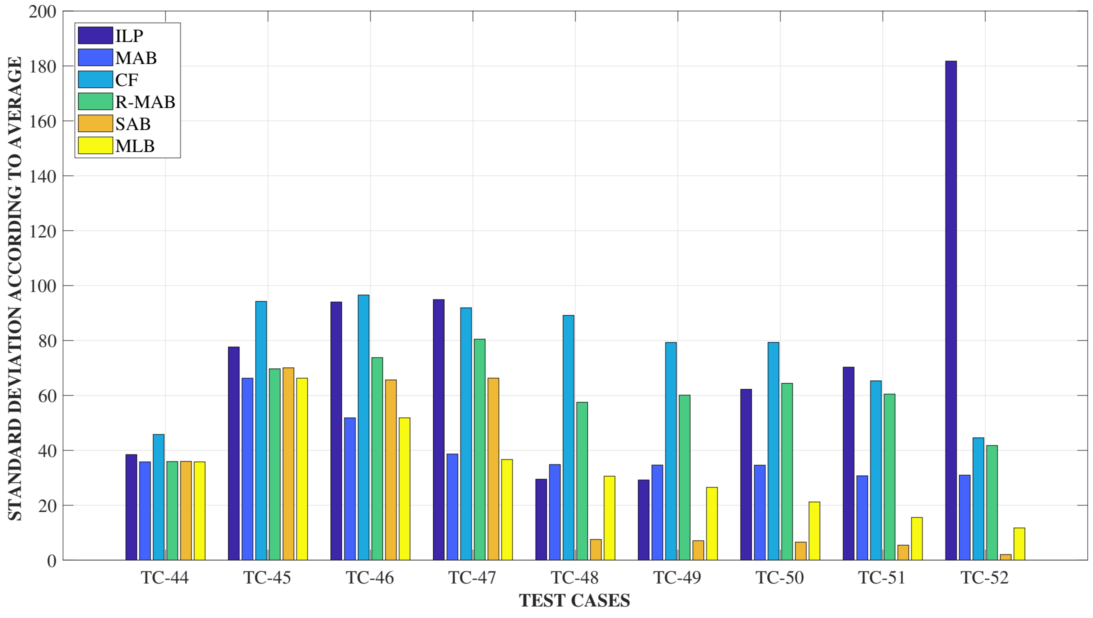


Figure 3.23: Standard Deviation in Equal Distributed Test Cases

Until TC-48, the total BU is smaller than %15 and under the small number of telegrams, the ILP solver can find the optimal solution in short run-time. When the total BU and number of telegrams increase, the ILP is not able to find an optimal solution in 1800 seconds (Timeout - TO) as shown in Table 3.28.

Table 3.28: Run-Time (ms) in Equal Distributed Test Cases

	TEST CASES								
	TC-44	**TC-45**	**TC-46**	**TC-47**	**TC-48**	**TC-49**	**TC-50**	**TC-51**	**TC-52**
ILP	$7 \cdot 10^3$	$13 \cdot 10^3$	$20 \cdot 10^3$	$32 \cdot 10^3$	TO	TO	TO	TO	TO
MAB	7.3	7.9	9.5	11.4	19.5	28.8	26.8	33.6	106.3
CF	7.5	8.2	10.0	11.8	21.4	29.9	29.7	35.8	75.2
R-MAB	7.8	8.7	10.6	12.6	21.5	31.0	29.7	36.6	75.0
SAB	520	569	640	681	$1 \cdot 10^3$	$2 \cdot 10^3$	$2 \cdot 10^3$	$2 \cdot 10^3$	$3 \cdot 10^3$
MLB	6.1	6.7	8.1	9.3	15.7	21.0	21.2	25.8	46.1

As given in Table 3.29, ILP could not find an optimal solution in scheduling for the test cases between TC-48 and TC-53 whereas run-time of all the basic heuristic algorithms is smaller than 5 seconds as stated in Table 3.28. Besides the run-time efficiency, looking at the maximum BU, minimum BU and σ_{BP} metrics SAB and MLB produce a result similar to or better than ILP.

Table 3.29: ILP Time Failure Count For Equal Distributed Test Cases

TC	44	45	46	47	48	49	50	51	52
Failures	0	0	0	0	**20**	**20**	**20**	**20**	**20**

The winners for each test case according to different performance metrics are noted and the sum of them is given in Table 3.30 according to each heuristic.

Table 3.30: Winner of Test Cases in Equal Distributed Telegrams

	Performance Metrics			
	$\min\{U_{\text{BP}}^{\max} - U_{\text{BP}}^{\min}\}$	σ_{BP}	Run-Time	$\min U_{\text{BP}}^{\max}$
MAB	3	2	0	3
CF	0	0	0	0
R-MAB	1	0	0	1
SAB	5	5	0	5
MLB	4	3	9	4

The total fail counts for each algorithm are also given in Table 3.31.

Table 3.31: Total Fail Counts in Equal Distributed Telegrams

Algorithm	**Fail Count**
MAB	0
CF	20
R-MAB	23
SAB	0
MLB	0

Using Table 3.30 and Table 3.31, it is seen that MLB and SAB perform best and produce results closest to the ILP solution. In addition, they do not get any fail whereas the ILP has too many time outs because of the large number of telegrams. CF and R-MAB could not win in any test cass and they could not schedule the telegrams in many cases.

3.4.3 Discussion of the Basic Heuristics

As explained in detail in the previous subsections, the different scheduling algorithms were studied according to defined performance metrics. Here, it is most important to minimize each BP duration in telegram scheduling. Besides, the run-time should be small to find an optimal solution fast.

Because of the small number of telegrams in TS-1, the ILP solver can find an optimal solution in all the test cases. According to the results, especially SAB performs similarly to the ILP, taking into account the maximum BUs and standard deviation σ_{BP}. In addition, the run-time is smaller than the ILP looking at Table 3.11. The remaining four heuristics perform faster than SAB and give results close to it for the maximum BUs. Further, all algorithms show a similar performance according to minimum BUs. In addition, according to σ_{BP}, MLB is the closest performing algorithm to SAB among the basic heuristics but the fail count of the MLB under high BUs is higher than SAB which makes it inferior to SAB as given in Table 3.10. Because MAB, CF and R-MAB show more failed trials, SAB is the best algorithm for the

TS-1 and produces schedules close to or better than ILP.

In TS-2, there are telegrams with a medium period and a relatively large number of telegrams are required to achieve high BUs compared to TS-1. Since the number of telegrams is still limited in TS-2, the ILP solver can find an optimal solution in most test cases when it does not time out. Despite time outs, the ILP still produces suitable results. According to the fail count, all the algorithms except for R-MAB do not show failed attempts. In general, R-MAB gives the worst results among the algorithm taking into account all the performance metrics. Looking at the maximum BUs, both MLB and SAB perform close to ILP as shown in Figure 3.15. Comparing σ_{BP}, SAB achieves a smaller σ_{BP} than ILP and MLB performs close to the ILP. CF and MAB approach MAB and SAB for the minimum BUs but they are inferior regarding the remaining performance metrics. Most importantly, MLB and SAB generate schedules close to or better than ILP but with a significantly reduced run-time. Specifically, the ILP times out for the test cases with more than 100 telegrams and 60% BUs.

TS-3 considers telegrams with a small period such that a very large number of telegrams is needed to obtain a BU above 20%. Because of the very large number of telegrams, an optimal solution cannot be obtained by the ILP solver. Different from TS-2, when the ILP times out, it does not produce suitable results according to maximum BU and σ_{BP}. Both SAB and MLB make the schedule close to or better than ILP in different cases according to the maximum BU as given in Figure 3.18. Looking at the minimum BU, all algorithms perform similar or better than ILP. When σ_{BP} is examined, the ILP performs much better than the heuristic algorithms if it does not time out as shown in Figure 3.20. MLB and MAB produce the closest results in cases where the ILP finds an optimal solution. When the ILP times out, all the heuristics perform better than the ILP, whereby SAB and MLB are superior to the other heuristics. Even the run-time of SAB is longer than for MLB and MAB, it is much shorter than the run-time of the ILP solver as given in Table 3.22.

Telegram sets with all possible periods and a similar BU per period exist in TS-4. As in the TS-3, it can be seen that the ILP cannot be solved if the number of telegrams is large. For the maximum BUs, MLB and SAB give better results than the ILP for the different test cases as seen in Figure 3.21. According to σ_{BP}, SAB gives more balanced schedules compared to ILP in all the test cases. In addition, MLB shows the closest performance to SAB and it is better than ILP in most of the cases. When the run-times are examined, all of the heuristics are much faster than the ILP.

The main and major difference between MLB-SAB and the remaining three heuristics is the telegram sorting approach in L. While MLB-SAB sort the telegrams according to d_i / p_i, the other algorithms consider the increasing p_i and decreasing d_i. CF, MAB, and R-MAB look at the value of C_j for placement. On the other hand, MLB checks D_j and SAB checks T_{av}. Therefore, it is seen that SAB performs the best and MLB follows SAB among the basic heuristics according to all test scenarios. These algorithms produce results that are very close to ILP according to all performance metrics. Even the ILP could not find an optimal solution in case of a large number of telegrams or a high BU, these two algorithms schedule the telegrams very fast and with high performance. Therefore, SAB and MLB will be used in Chapter 4 and improved with different swap operations.

The basic heuristics give suitable schedules but there is still some gap to the ILP, which leaves room for improvement. For the test cases that have a small number of telegrams, schedules are compared between heuristics and ILP. Here, it is seen that the heuristics place the larger telegrams to the longest BPs in some cases because they generally pick up telegrams in that order from L. As a result, it is possible that the schedules are not as balanced as desired, leading to longer BP durations. To overcome this shortcoming, different swap operations are developed in the next chapter in order to exchange the locations of telegrams in the schedule in case the maximum BP duration can be reduced.

CHAPTER 4

SWAP OPERATIONS FOR TELEGRAM SCHEDULING ON BASIC ALGORITHMS

This chapter develops different procedures for improving the schedules obtained by the basic heuristics in the previous chapter. Hereby, improvements include shortening the maximum BP duration, making schedules more balanced as well as making infeasible schedules feasible. Section 4.1 defines the different swap operations developed in this thesis and Section 4.2 performs a comprehensive evaluation of these swap operations.

4.1 Swap Operations Definition

We assume that an initial schedule is determined using one of the basic heuristics in Chapter 3 before applying any of the swap operations. Hereby, the constraint in (3.11) is discarded for the basic heuristic algorithms in order to always obtain a schedule that includes all the telegrams. This means that there can be infeasible schedules with BP durations greater than T_{BP}. The goal of the swap operations is then to reduce the BP durations in order to obtain a feasible schedule with the shortest possible maximum BP duration. If the constraint in (3.11) is still not satisfied, the algorithm is marked as unsuccessful.

Assume that we computed offsets o_i for all telegrams $T_i \in T$. Then, we define the parameters as shown below.

$$U_{\text{BP}}^{\max} = \max\{D_k\}, \quad \text{according to } (3.3) \tag{4.1}$$

$$U_{\text{BP}}^{\text{min}} = \min\{D_k\}, \quad \text{according to } (3.4) \tag{4.2}$$

$$D_{i,\text{max}} = \max\{D_{i,k}\}, \quad D_{i,k} = D_{o_i + k \cdot r_i}, \quad \text{where k} = 0, \ldots, \frac{N_{\text{MP}}}{r_i} - 1 \tag{4.3}$$

$U_{\text{BP}}^{\text{max}}$ and $U_{\text{BP}}^{\text{min}}$ characterize the maximum and minimum duration of a BP in the computed schedule, respectively. $D_{i,\text{max}}$ is the maximum duration of a BP, where telegram T_i is scheduled and σ_{BP} is the standard deviation of the BP durations which is calculated as shown in (3.6).

Since the swap operations update an existing schedule, we further use two sets of offsets. One set that is used for the currently assigned telegrams which are expressed with O and the other one is used to keep changes after the swap operations that is expressed by $\hat{O}$.

Since SAB and MLB were determined as the basic heuristics with the best performance, the swap operations are separately applied to these basic heuristics. We next define four different swap operations.

4.1.1 Swap Operation According to Maximum Basic Period Duration (SMB)

The main idea of the swap operation in this section is to exchange the offset values of two telegrams with the same period and different durations. Hereby, the aim is to reduce the maximum BP duration after swapping the telegrams.

The detailed procedure is given in Algorithm 6. After the initial schedule is constructed omitting the constraint control (3.11), the swap operation is applied separately for each possible repetition $r_k = 2^k$ back to back where $k = 0, \ldots, 10$. In each iteration, the operation counts the telegrams whose repetition is $r_k = 2^k$, as $|T_k|$. If there is no telegram or only one telegram (that is, there is nothing to be swapped), the operation proceeds to the next r_k. When at least two telegrams are found, it starts to check if swapping will be beneficial. The algorithm compares any two telegrams T_i and T_j with the offsets o_i and o_j bi-directional which means that both T_i-T_j and T_j-T_i are compared.

If the offsets of the two telegrams are equal to each other or $d_i < d_j$, T_i and T_j are not swapped. The reason is that there will be no gain when the two telegrams with the same offsets are exchanged. Otherwise, $D_{i,\text{max}}$, $D_{j,\text{max}}$, $U_{\text{BP}}^{\text{max}}$, and $U_{\text{BP}}^{\text{min}}$ are calculated by the algorithm in order to use them in comparison according to N_{MP}.

If $D_{i,\text{max}}$ is greater than $D_{j,\text{max}}$ and the difference between d_i and d_j is smaller than or equal to the difference between $D_{i,\text{max}}$ and $D_{j,\text{max}}$, the offsets are exchanged for i and j in order not to exceed $D_{i,\text{max}}$ after the swap. The new offsets are put in $\hat{O}$ and the basic periods are updated, while the algorithm keeps the parameters before the exchange until the iteration is complete. Otherwise, telegrams are discarded and the algorithm continues taking the next telegram.

When the offsets are exchanged, the algorithm calculates $\hat{D}_{i,\text{max}}$, $\hat{D}_{j,\text{max}}$, $\hat{U}_{\text{BP}}^{\text{max}}$, and $\hat{U}_{\text{BP}}^{\text{min}}$ for the modified offset assignment $\hat{O}$. Here, it is desired to decrease the maximum BP duration and to increase the minimum BP duration after the swap operation. That is, the algorithm accepts $\hat{O}$ if (i) $\hat{U}_{\text{BP}}^{\text{max}}$ is smaller than or equal to $U_{\text{BP}}^{\text{max}}$, (ii) $\hat{U}_{\text{BP}}^{\text{min}}$ is bigger than or equal to $U_{\text{BP}}^{\text{min}}$ and (iii) the absolute value of the difference between $\hat{D}_{i,\text{max}}$, $\hat{D}_{j,\text{max}}$ is smaller than $D_{i,\text{max}}$ and $D_{j,\text{max}}$. Otherwise, the swap operation is cancelled. If the conditions are satisfied, it is checked if $\hat{O}$ leads to a balanced schedule by computing σ_{BP} for both O and $\hat{O}$ as σ_{BP} and $\hat{\sigma}_{\text{BP}}$, respectively. If $\hat{\sigma}_{\text{BP}}$ is smaller than $\sigma_{\text{BP}} - 0.01$, and none of BP durations $\hat{D}_l$ is greater than T_{BP}, the swap operation is completed successfully and the modified values are assigned: $O = \hat{O}$ and $D_l = \hat{D}_l$. Here, it is desired that $\hat{\sigma}$ is smaller than σ_{BP} in order to increase the run-time and performance. If the conditions are not satisfied, the swap operation is cancelled and the previous values of the variables are used.

Algorithm 6 SMB Pseudo Code

```
1: **Input:** $T$
2: **Output:** offset $o_i$ for each telegram $T_i$
3: **Initialize:** Set of assigned offsets $O = \hat{O} = \emptyset$; define list $T_k \subseteq T$ of telegrams
   with repetition $2^k$; $D_l = \hat{D}_l = 0$ for $j = 0, \ldots, N_{\text{MP}} - 1$
4: Apply basic heuristic without 3.9 constraint; obtain offset $o_i$ for each telegram $T_i$
   and $D_l$ for each basic period
5: **for** $0 \leq k \leq 10$ **do**
6:    **if** $|T_k| == 1$ **then**
7:       **continue**
8:    **for** $1 \leq i \leq |T_k|$ **do**
9:       **for** $1 \leq j \leq |T_k|, j \neq i$ **do**
10:         Take telegrams $T_i$ and $T_j$
11:         **if** $o_i \neq o_j$ **and** $d_i > d_j$ **then**
12:            Compute $D_{i,\text{max}}$, $D_{j,\text{max}}$, $U_{\text{BP}}^{\text{max}}$, $U_{\text{BP}}^{\text{min}}$
13:            **if** $(D_{i,\text{max}} > D_{j,\text{max}})$ **and** $(d_i + D_{j,\text{max}} \leq d_j + D_{i,\text{max}})$ **then**
14:               Exchange $o_i$ and $o_j$ in $\hat{O}$ for $T_i$ and $T_j$
15:               Compute $\hat{D}_{i,\text{max}}, \hat{D}_{j,\text{max}}, \hat{U}_{\text{BP}}^{\text{max}}, \hat{U}_{\text{BP}}^{\text{min}}$, $\hat{\sigma}$, $\sigma_{\text{BP}}$, $\hat{D}_l$ for $0 \leq l \leq N_{\text{MP}} - 1$
16:               **if** $\hat{U}_{\text{BP}}^{\text{max}} \leq U_{\text{BP}}^{\text{max}}$ **and** $\hat{U}_{\text{BP}}^{\text{min}} \geq U_{\text{BP}}^{\text{min}}$ **and** $|\hat{D}_{j,\text{max}} - \hat{D}_{i,\text{max}}| < |D_{j,\text{max}} -$
                  $D_{i,\text{max}}|$ **and** $\hat{\sigma}_{\text{BP}} < (\sigma_{\text{BP}} - 0.01)$ **and** $\hat{D}_l \leq T_{\text{BP}}$ for $0 \leq l \leq N_{\text{MP}} - 1$
                  **then**
17:                  Swap operation is successfully completed, $O = \hat{O}$ and $D_l = \hat{D}_l$
18:               **else**
19:                  Don't swap. No change in $O$ and $D_l$ for $0 \leq l \leq N_{\text{MP}} - 1$
20:            **else**
21:               Don't swap. No change in $O$ and $D_l$ for $0 \leq l \leq N_{\text{MP}} - 1$
22:         **else**
23:            Don't swap. No change in $O$ and $D_l$ for $0 \leq l \leq N_{\text{MP}} - 1$
```

In order to show how swap operation proceeds and what is the result of it, consider the example with the 9 telegrams in Table 4.1. Each telegram T_i has individual period p_i and a duration d_i.

Table 4.1: Telegram Set For Swap Operation

T_i	1	2	3	4	5	6	7	8	9
d_i	89.7	265.7	89.7	121.7	89.7	265.7	100.37	265.7	121.7
p_i	1	2	2	2	2	4	4	4	4

Looking at the duration and individual period of the telegrams, the sorted list L is as shown in Table 4.2. Using MLB, the resulting initial schedule is shown in Figure 4.1.

Table 4.2: Sorted List according to MLB for Swap Operation

$L[i]$	1	2	3	4	5	6	7	8	9
T_i	2	1	6	8	4	3	5	9	7
d_i	265.7	89.7	265.7	265.7	121.7	89.7	89.7	121.7	100.37
p_i	2	1	4	4	2	2	2	4	4

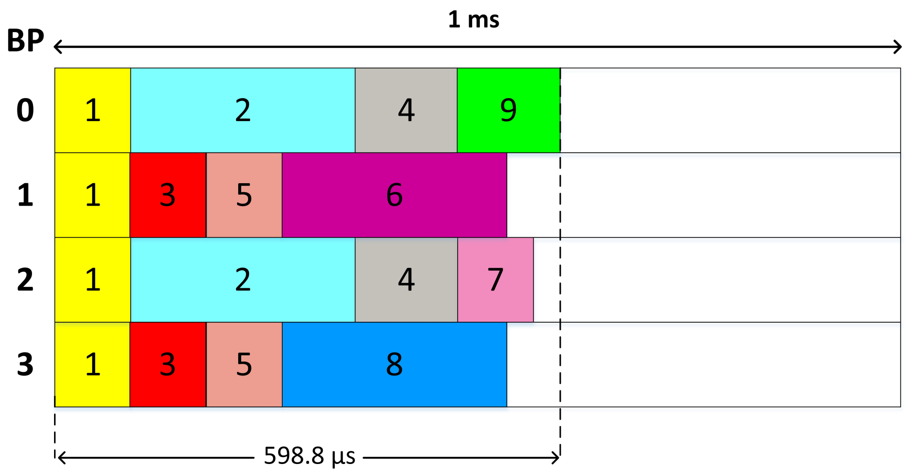


Figure 4.1: MVB Schedule with MLB for SMB

According to Figure 4.1, $\mathrm{BP}_0 = 598.8\,\mu\mathrm{s}$, $\mathrm{BP}_1 = 534.8\,\mu\mathrm{s}$, $\mathrm{BP}_2 = 577.47\,\mu\mathrm{s}$ and $\mathrm{BP}_3 = 534.8\,\mu\mathrm{s}$. When T_4 and T_3 are compared, it is seen that $o_4 = 0, o_3 = 1$. Since d_4 is larger than d_3 and the offsets of the two telegrams are not equal, the swap operation continues. The operation finds $D_{4,\max} = 598.8\,\mu$s and $D_{3,\max} = 534.8\,\mu$s. Also, $U_{\mathrm{BP}}^{\max} = 598.8\,\mu$s and $U_{\mathrm{BP}}^{\min} = 534.8\,\mu$s are noted. Due to the fact that $D_{3,\max} = 534.8\,\mu$s is smaller than $D_{4,\max} = 598.8\,\mu$s and $d_4 - d_3 = 32\,\mu$s is smaller than $D_{4,\max} - D_{3,\max} = 64\,\mu$s, the algorithm exchanges the offsets like $o_4 = 1$, $o_3 = 0$ in $\hat{O}$ and $\hat{D}_1, \hat{D}_2, \hat{D}_3$ and $\hat{D}_4$ updated.

After that, the algorithm finds $\hat{D}_{3,\max} = 566.8\,\mu$s and $\hat{D}_{4,\max} = 566.8\,\mu$s. Also $\hat{U}_{\mathrm{BP}}^{\max} = 566.8\,\mu$s and $\hat{U}_{\mathrm{BP}}^{\min} = 545.47\mu$s are noted. Since $|\, D_{4,\max} - D_{3,\max}| = 64\,\mu$s is larger than $|\hat{D}_{3,\max} - \hat{D}_{4,\max}| = 0$, $U_{\mathrm{BP}}^{\max} = 598.8\,\mu$s is larger than $\hat{U}_{\mathrm{BP}}^{\max} = 566.8\,\mu$s and $U_{\mathrm{BP}}^{\min} = 534.8\,\mu$s is smaller than $\hat{U}_{\mathrm{BP}}^{\min} = 545.47\,\mu$s, the algorithm continues and calculates $\sigma_{\mathrm{BP}} = 7.68\,\mu$s and $\hat{\sigma}_{\mathrm{BP}} = 0.85\,\mu$s, respectively. Due to the fact that $\hat{\sigma}_{\mathrm{BP}}$ is smaller than the $\hat{\sigma}_{\mathrm{BP}}$ - 0.01, the swap operation is successfully completed and the updated schedule is obtained as shown in Figure 4.2. That is, the maximum BP duration can be decreased from $598.8\,\mu$s to $566.8\,\mu$s in this example.

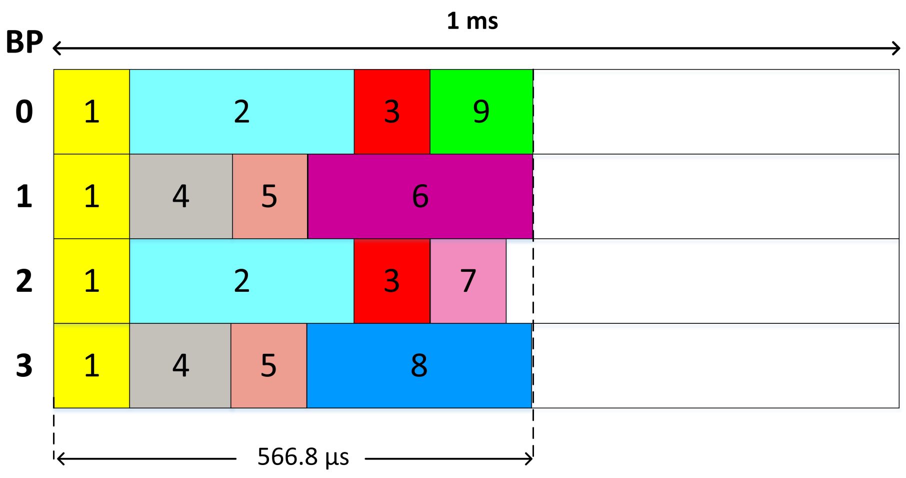


Figure 4.2: MLB schedule after SMB

4.1.2 Swap Operation According to Sum Of Basic Period Duration (SSB)

The difference of the swap operation in this section to the one in the previous section is to use the sum of the BP durations instead of the maximum BP duration. Assume that we computed offsets o_i for all telegrams $T_i \in T$. Then, we define the parameter

$$C_j = \sum_{k=0}^{N_{\text{MP}}/r_i - 1} D_{j+k \cdot r_i}, \quad \text{where } j = 0, \ldots, r_i - 1. \tag{4.4}$$

C_j characterizes the summation of the basic period durations in the computed schedule, where T_i transmitted according to $o_i = j$.

After the schedule is constructed omitting the constraint in (3.11), this process is run separately for each r_i back to back. In each iteration, the algorithm counts the telegrams that have the current r_i. If there is no telegram or only one telegram, the algorithm proceeds to the next r_i. When at least two telegrams are found, the algorithm starts to attempt the swap operation. The algorithm compares the two telegrams with each other bi-directional which means that both T_i-T_j and T_j-T_i are compared differently.

The operation finds the offset of the two telegrams, o_i and o_j, from O for relocation. If offsets of the two telegrams are equal to each other or d_i is smaller than d_j, these are discarded, the operation continues taking the next telegram. Because there will be no gain when the two telegrams with the same offsets are exchanged. Otherwise, C_{o_i} and C_{o_j} are computed by the algorithms in order to use them in comparison according to N_{MP}.

If C_{o_i} is larger than $C_{o_j} + 10$, 10 is added to C_{o_j} in order to eliminate the small difference according to experimental results for run-time performance, the operation continues and simulates the swap operation taking into account the d_i and d_j with changing o_i and o_j. As shown below, temporary summations are computed.

$$\hat{C}_{o_i} = C_{o_i} + \frac{N_{\text{MP}}}{r_i} - 1 \cdot (d_i - d_j) \tag{4.5}$$

$$\hat{C}_{o_j} = C_{o_j} + \frac{N_{\text{MP}}}{r_i} - 1 \cdot (d_j - d_i) \tag{4.6}$$

The operation goes on if the summation difference is decreased. It checks F and $\hat{F}$ that is calculated as shown below.

$$F = C_{o_i} - C_{o_j} \tag{4.7}$$

$$\hat{F} = |\hat{C}_{o_j} - \hat{C}_{o_i}| \tag{4.8}$$

If $\hat{F}$ is smaller than F, the operation proceeds to the new control mechanism because the difference should be decreased in order to get a more balanced schedule. Otherwise, a swap operation is canceled. When the condition is satisfied, $U_{\text{BP}}^{\text{max}}$ and $U_{\text{BP}}^{\text{min}}$ are computed by the algorithm. $\hat{D}_j$ is updated according to the offsets o_i and o_j using the definition:

$$\hat{D}_{o_i + k \cdots r_i} = D_{o_i + k \cdot r_i} - d_i + d_j, \text{ for } k = 0, \ldots, \frac{N_{\text{MP}}}{r_i} - 1, \tag{4.9}$$

$$\hat{D}_{o_j + k \cdot r_j} = D_{o_j + k \cdot r_j} - d_j + d_i, \text{ for } k = 0, \ldots, \frac{N_{\text{MP}}}{r_j} - 1. \tag{4.10}$$

Also, $\hat{O}$ and is updated. $\hat{U}_{\text{BP}}^{\text{max}}$ and $\hat{U}_{\text{BP}}^{\text{min}}$ are computed according to $\hat{D}_j$. If $\hat{U}_{\text{BP}}^{\text{max}}$ is smaller than or equal to $U_{\text{BP}}^{\text{max}}$ and $\hat{U}_{\text{BP}}^{\text{min}}$ is greater or equal than $U_{\text{BP}}^{\text{min}}$, the algorithm calculates σ_{BP} according to D_j, O and $\hat{\sigma}_{\text{BP}}$ according $\hat{D}_j$, $\hat{O}$. If the difference between $\hat{\sigma}_{\text{BP}}$ is smaller than $\sigma_{\text{BP}} - 0.01$, the algorithm checks all BP durations. If none of them is greater than T_{BP}, a swap operation is completed successfully and BPs are updated. A smaller σ_{BP} is desired after the operation in order to avoid unnecessary swaps. If the conditions are not satisfied, the swap operation is canceled and the variables are returned to the pre-processed state as recorded before.

Algorithm 7 SSB Pseudo Code

```
1: Input: $T$
2: Output: offset $o_i$ for each telegram $T_i$
3: Initialize: Set of assigned offsets $O = \hat{O} = \emptyset$; define list $T_k \subseteq T$ of telegrams with repetition $2^k$; $D_l = \hat{D}_l = 0$ for $j = 0, \ldots, N_{\text{MP}} - 1$
4: Apply basic heuristic without the constraint in (3.9); obtain offset $o_i$ for each telegram $T_i$ and $D_l$ for each basic period
5: for $0 \leq k \leq 10$ do
6:   if $|T_k| == 1$ then
7:     continue
8:   for $1 \leq i \leq |T_k|$ do
9:     for $1 \leq j \leq |T_k|, j \neq i$ do
10:      Take telegrams $T_i$ and $T_j$
11:      if $o_i \neq o_j$ and $d_i > d_j$ then
12:        Compute $C_i$ and $C_j$
13:        if $(C_i - C_j) < 10$ then
14:          Compute $\hat{C}_i$, $\hat{C}_j$, $F$ and $\hat{F}$
15:          if $\hat{F} < F$ then
16:            Exchange $o_i$ and $o_j$ in $\hat{O}$ for $T_i$ and $T_j$
17:            Compute $U_{\text{BP}}^{\max}$ , $U_{\text{BP}}^{\min}$, $\hat{U}_{\text{BP}}^{\max}$, $\hat{U}_{\text{BP}}^{\min}$ and $\hat{D}_l$ for $0 \leq l \leq N_{\text{MP}} - 1$
18:            if $\hat{U}_{\text{BP}}^{\max} \leq U_{\text{BP}}^{\max}$ and $U_{\text{BP}}^{\min} \leq \hat{U}_{\text{BP}}^{\min}$ then
19:              Compute $\sigma_{\text{BP}}$ and $\hat{\sigma}_{\text{BP}}$
20:              if $\hat{\sigma}_{\text{BP}} < \sigma_{\text{BP}} - 0.01$ and $\hat{D}_l \leq T_{\text{BP}}$ for $0 \leq l \leq N_{\text{MP}} - 1$ then
21:                Swap operation is successfully completed, $O = \hat{O}$ and $D_l = \hat{D}_l$
22:              else
23:                Don't swap. No change in $O$ and $D_l$ for $0 \leq l \leq N_{\text{MP}} - 1$
24:            else
25:              Don't swap. No change in $O$ and $D_l$ for $0 \leq l \leq N_{\text{MP}} - 1$
26:          else
27:            Don't swap. No change in $O$ and $D_l$ for $0 \leq l \leq N_{\text{MP}} - 1$
28:        else
29:          Don't swap. No change in $O$ and $D_l$ for $0 \leq l \leq N_{\text{MP}} - 1$
```

Looking at the duration and individual periods of the telegrams, the sorted list L for the example telegram set is shown in Table 4.3 and the initial schedule using MLB is as shown in Figure 4.3

Table 4.3: Sorted List according to MLB for SSB

$L[i]$	1	2	3	4	5	6	7	8	9	10	11
T_i	2	1	3	5	4	9	11	6	10	7	8
d_i	265.7	89.7	121.7	121.7	89.7	169.7	169.7	121.7	121.7	100.37	100.37
p_i	2	1	2	2	2	4	4	4	4	4	4

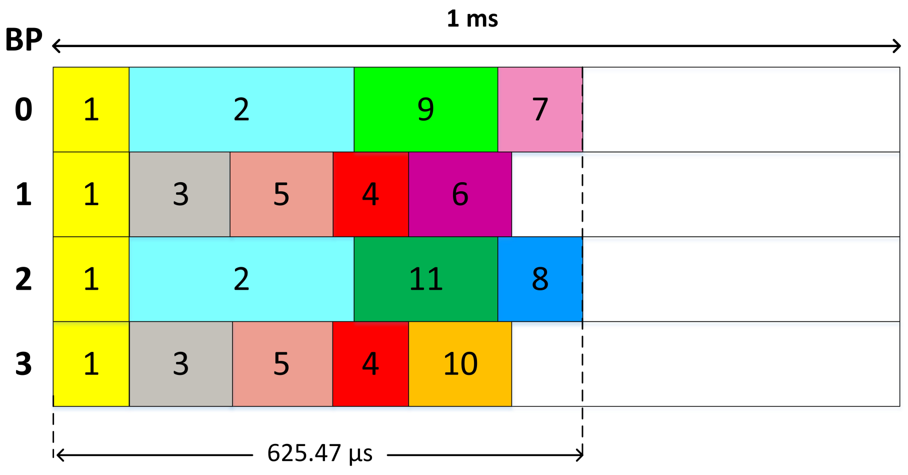


Figure 4.3: MVB Schedule with MLB for SSB

According to Figure 4.3, $\mathrm{BP}_0 = \mathrm{BP}_2 = 625.47\,\mu$s, $\mathrm{BP}_1 = \mathrm{BP}_3 = 544.5\,\mu$s. When T_9 and T_6 are compared, it is seen that $o_9 = 0, o_6 = 1$. Because d_9 is greater than d_6 and offsets are not equal for the two telegrams, the swap operation continues. The algorithm finds $C_0 = 625.47\,\mu$s and $C_1 = 544.5\,\mu$s. Since C_0 is greater than $C_1 + 10$, the offsets are exchanged and become $o_6 = 0, o_9 = 1$. $\hat{C}_0$ is computed as $577.47\,\mu s$ and $\hat{C}_1$ is computed as $592.5\,\mu$s. $\hat{O}$ is updated according to the offsets and $\hat{D}_l$ is updated for $0 \leq l \leq 3$. Then, $F = 80.97\,\mu s$ and $\hat{F} = 15.03\,\mu s$ are obtained.

Because $\hat{F} < F$ the condition for swapping is satisfied. Hence, $\hat{U}_{\mathrm{BP}}^{\max}$, $\hat{U}_{\mathrm{BP}}^{\min}$, $U_{\mathrm{BP}}^{\max}$, $U_{\mathrm{BP}}^{\min}$ are calculated. $U_{\mathrm{BP}}^{\max} = \hat{U}_{\mathrm{BP}}^{\max}$ is noted as $625.47\,\mu$s and $U_{\mathrm{BP}}^{\min} = \hat{U}_{\mathrm{BP}}^{\min}$ is noted as $544.5\,\mu$s. Due to $\hat{U}_{\text{BP}}^{\max} = U_{\text{BP}}^{\max}$ and $\hat{U}_{\text{BP}}^{\min} = U_{\text{BP}}^{\max}$, the algorithm calculates $\sigma_{\text{BP}} = 16.39\,\mu$s and $\hat{\sigma}_{\text{BP}} = 8.48\,\mu$s. Since $\hat{\sigma}_{\text{BP}}$ is smaller than $\hat{\sigma}_{\text{BP}}$ - 0.01, the swap operation is successfully completed and an updated schedule is obtained as shown in Figure 4.4. In this example, the maximum BP duration cannot be decreased, but a more balanced schedule is achieved after the swap operation.

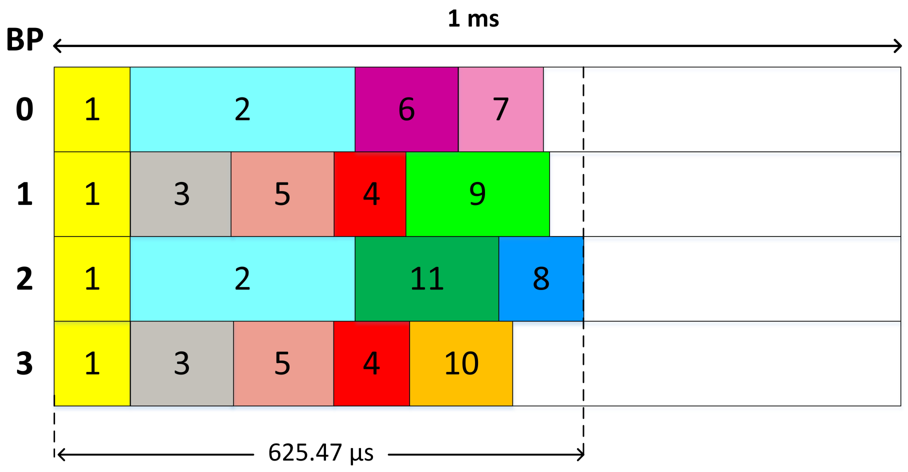


Figure 4.4: MLB schedule after SSB

4.1.3 Swap Operation Looking At The Next Two Telegrams (S2T)

Different from the swap operations in the previous sections, that compare telegrams with the same period, the swap operation in this section looks at telegrams with neighboring periods. After the schedule is constructed omitting the constraint in (3.11), the swap operation counts the telegrams that have the current repetition r_i, as $|T_k|$, and counts the telegrams that have the repetition $2 \cdot r_i$, as $|T_m|$. For each telegram T_i with repetition r_i, the algorithm picks up the two telegrams T_j and T_n with $2 \cdot r_i$. Here, each T_i is compared with T_j and T_n bi-directional.

The swap operation computes o_j and o_n for the selected telegrams. If o_n is different from o_j and $|o_j - o_n| \mod r_i$ is equal to 0, the algorithm continues. The reason is that, in order to swap T_j and T_n with T_i, the offset difference should be equal to r_i according to the repetition of T_i. Otherwise, T_i with r_i is not placed correctly, replacing the telegrams T_j and T_n in different BPs. If o_i is equal to o_j or o_n, $o_i + r_i$ is equal to o_j or o_n the operation stops to work for these telegrams and takes new telegrams. The reason is that any two telegrams should not be in the same BP for successful and meaningful exchange. Otherwise, C_{o_j} and C_{o_n} are computed according to o_j and o_n and their sum is assigned to C_{sum}. Also, C_{o_i} is computed and compared with C_{sum}.

The absolute value of the difference between C_{sum} and C_{o_i} is assigned to F, which is described in (4.7) and σ_{BP} is calculated at the same time. If C_{sum} is greater than C_{o_i}, d_j + d_n is greater than $2 \cdot d_i$ or C_{sum} is smaller than C_{o_i} and d_j + d_n is smaller than $2 \cdot d_i$, the algorithm goes on. In order to decrease σ_{BP} the algorithm makes the defined control, because at the end of this comparison C_{sum} and C_{o_i} are decreased. Because the T_j and T_n are repeated with $2 \cdot r_i$, in the comparison the algorithm checks $2 \cdot d_i$. Otherwise, the swap operation is canceled.

When the conditions are satisfied, $\hat{C}_{\text{sum}}$ and $\hat{C}_{o_i}$ are computed assuming that T_i and T_j, T_n are switched. According to this hypothesis, $\hat{F}$ is computed as given in (4.8) using $\hat{C}_{\text{sum}}$ and $\hat{C}_{o_i}$. If $\hat{F}$ is smaller than F, the algorithm updates $\hat{O}$ with O and $\hat{D}_j$ with D_j. The algorithm places T_i to the smaller offset between o_j and o_n instead of o_i because r_i is equal to the difference between o_j and o_n, $\hat{O}$ and D_j is updated according to that process.

The algorithm computes the $D_{i,\text{max}}$ according to r_i and finds the value k that gives $D_{i,\text{max}}$. If d_j is greater than d_n, the algorithm puts T_n to the offset where $D_{i,\text{max}}$ is computed and T_j is placed with the offset $o_i + r_i$ or $o_i - r_i$ and according to $D_{i,\text{max}}$. If the offset where $D_{i,\text{max}}$ is computed is greater than r_i, the offset $o_i - r_i$ is selected for T_j because it is desired to put the smallest telegram to $D_{i,\text{max}}$ in order to get a balanced schedule. $\hat{O}$ and $\hat{D}_j$ are updated according to the changed offsets.

Otherwise, if d_j is smaller than d_n, the algorithm puts T_j to the offset where $D_{i,\text{max}}$ is computed and T_n is placed with the offset $o_i + r_i$ or $o_i - r_i$ and according to $D_{i,\text{max}}$. If the offset, where $D_{i,\text{max}}$ is computed is greater than r_i, the offset $o_i - r_i$ is selected for T_n. $\hat{O}$ and $\hat{D}_j$ according to the changed offsets.

Then, the algorithm computes $\hat{\sigma}_{\text{BP}}$ using $\hat{D}_j$ and $\hat{O}$. If $\hat{\sigma}_{\text{BP}}$ is smaller than σ_{BP}, the swap operation completed successfully. If the condition is not satisfied, the swap operation is canceled and the variables are returned to their previous state as recorded.

Algorithm 8 S2T Pseudo Code

```
 1: Input: $T$
 2: Output: offset $o_i$ for each telegram $T_i$
 3: Initialize: Set of assigned offsets $O = \hat{O} = \emptyset$; define list $T_k \subseteq T$ of telegrams
    with repetition $2^k$; $D_l = \hat{D}_l = 0$ for $j = 0, \ldots, N_{\text{MP}} - 1$
 4: Apply basic heuristic without 3.9 constraint; obtain offset $o_i$ for each telegram $T_i$
    and $D_l$ for each basic period
 5: for $0 \leq k \leq 10$ do
 6:     if not $|T_k| == 1$ and $|T_{k+1}| == 2$ then
 7:         continue
 8:     for $1 \leq i \leq |T_k|$ do
 9:         for $1 \leq j \leq |T_{k+1}|$ do
10:             for $1 \leq n \leq |T_{k+1}|, n \neq j$ do
11:                 Take telegrams $T_i$, $T_j$ and $T_n$
12:                 if $o_i \neq o_j$ and $o_i \neq o_n$ and $o_j \neq o_n$ and $|o_j - o_n| = 2^k$ then
13:                     Compute $C_{\text{sum}}$, $C_{o_i}$, $F$, $\sigma_{\text{BP}}$
14:                     if $C_{\text{sum}} > C_{o_i}$ and $d_j + d_n > 2 \cdot d_i$ or $C_{\text{sum}} < C_{o_i}$ and $d_j + d_n < 2 \cdot d_i$
                        then
15:                         Compute $\hat{C}_{\text{sum}}$, $\hat{C}_{o_i}$, $\hat{F}$
16:                         if $\hat{F} < F$ then
17:                             Exchange $o_i$, $o_j$, $o_n$ in $\hat{O}$ for $T_i$, $T_j$, $T_n$
18:                             Compute $\hat{\sigma}_{\text{BP}}$, $\hat{D}_l$ for $0 \leq l \leq N_{\text{MP}} - 1$
19:                             if $\hat{\sigma}_{\text{BP}} < \sigma_{\text{BP}}$ and $\hat{D}_l \leq T_{\text{BP}}$ for $0 \leq l \leq N_{\text{MP}} - 1$ then
20:                                 Swap operation is successfully completed, $O = \hat{O}$ and $D_l =$
                                    $\hat{D}_l$
21:                             else
22:                                 Don't swap. No change in $O$ and $D_l$ for $0 \leq l \leq N_{\text{MP}} - 1$
23:                         else
24:                             Don't swap. No change in $O$ and $D_l$ for $0 \leq l \leq N_{\text{MP}} - 1$
25:                     else
26:                         Don't swap. No change in $O$ and $D_l$ for $0 \leq l \leq N_{\text{MP}} - 1$
27:                 else
28:                     Don't swap. No change in $O$ and $D_l$ for $0 \leq l \leq N_{\text{MP}} - 1$
```

In order to show how the swap operation is run, consider the example with the 14 telegrams in Table 4.4. Each telegram T_i has individual period p_i and a duration d_i.

Table 4.4: Telegram Set For S2T

T_i	1	2	3	4	5	6	7
d_i	89.7	89.7	89.7	265.7	169.7	169.7	89.7
p_i	1	1	1	2	2	2	2
T_i	8	9	10	11	12	13	14
d_i	100.37	265.7	265.7	265.7	100.37	100.37	89.7
p_i	4	4	4	4	4	4	4

Looking at the duration and individual periods of the telegrams, the sorted list L is as shown in Table 4.5. The resulting schedule for the example telegram set using MLB is shown in Figure 4.5.

Table 4.5: Sorted List according to S2T

$L[i]$	1	2	3	4	5	6	7
T_i	4	1	2	3	5	6	9
d_i	265.7	89.7	89.7	89.7	169.7	169.7	265.7
p_i	2	1	1	1	2	2	4
$L[i]$	8	9	10	11	12	13	14
T_i	10	11	7	8	12	13	14
d_i	265.7	265.7	89.7	100.37	100.37	100.37	89.7
p_i	4	4	2	4	4	4	4

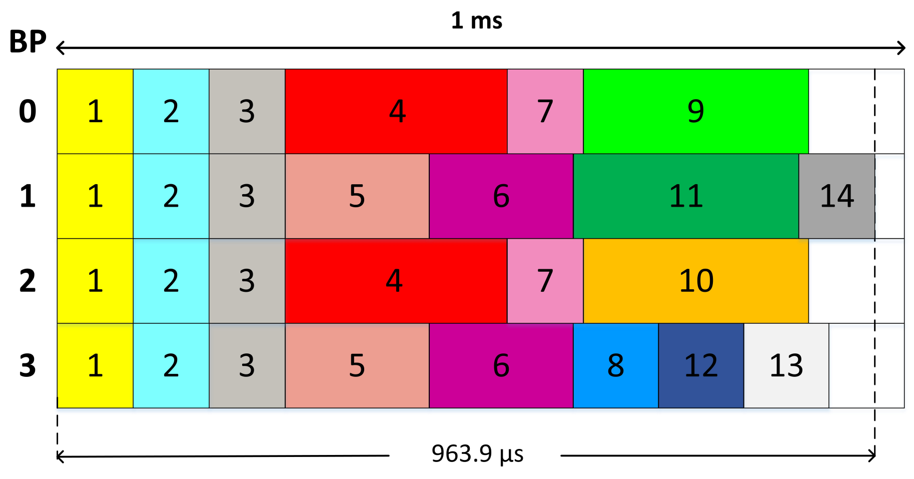


Figure 4.5: MVB Schedule with MLB for S2T

According to Figure 4.5, $\mathrm{BP}_0 = 890.2\,\mu$s, $\mathrm{BP}_1 = 963.9\,\mu$s, $\mathrm{BP}_2 = 890.2\,\mu$s and $\mathrm{BP}_3 = 909.6\,\mu$s.

S2T picks T_7 with the offset $o_7 = 0$ and $r_7 = 2$. Then, the algorithm looks for two telegrams with $2 \cdot r_7$. According to that, S2T picks up T_8 with $o_8 = 3$ and T_{14} with the offset $o_{14} = 1$. Because of their offsets are different from each other and $|o_{14} - o_8| = r_7$, C_{sum} is calculated summing up C_3 and C_1 which is equal to $1873.5\,\mu$s. C_0 is computed as $1780.4\,\mu$s and F is found as $93.1\,\mu$s from $C_{\text{sum}} - C_0$ and σ_{BP} is calculated as $9.10\,\mu$s. Since C_{sum} is greater than C_0 and $d_{14} + d_8$ is greater than $2 \cdot d_7$, S2T computes $\hat{C}_{\text{sum}} = 1862.8\,\mu$s assuming that T_8 and T_{14} exchanged their offsets with T_7. $\hat{C}_0$ is also calculated as $1791.1\,\mu$s and $\hat{F}$ is obtained as $71.77\,\mu$s. Because $\hat{F}$ is smaller than F, the offset of T_7 is changed to $o_7 = 1$. Because d_8 is smaller than d_{14} and BP_0 is equal to BP_2, T_8 is scheduled with offset $o_8 = 0$ and T_{14} is scheduled with offset $o_{14} = 2$. After the swap operation, $\hat{\sigma}_{\text{BP}}$ is calculated as $8.64\,\mu$s. The swap operation is completed successfully because $\hat{\sigma}_{\text{BP}}$ is smaller than σ_{BP}. The resulting schedule is shown in Figure 4.6. The maximum BP is not decreased but the schedule become more balanced bringing duration of basic periods closer together. This is achieved since the standard deviation σ_{BP} is decreased after the swap operation.

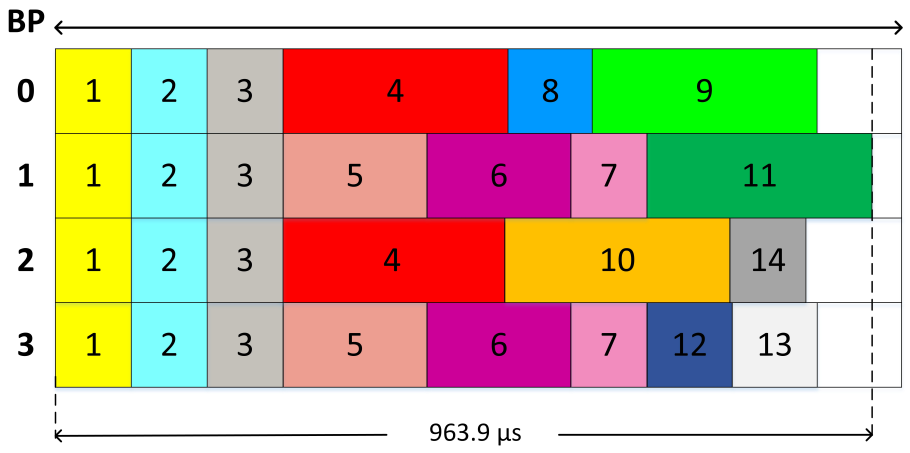


Figure 4.6: MLB schedule after S2T

4.1.4 Swap Operation Comparing Three Telegrams (S3T)

The main idea of this swap operation is to select three telegrams with the same repetition and to check if one of the telegrams can be replaced by the other two telegrams. After the schedule is constructed omitting the constraint in (3.11), the algorithm counts the telegrams whose repetition rate is 2^k, as $|T_k|$. The algorithm picks three telegrams, T_i, T_j and T_n, at the same time with r_i. If $|T_k|$ is smaller than three, the algorithm stops and proceeds to the next repetition r_i.

If o_i and o_j are different, the algorithm continues and picks a third telegram with o_n. If o_n is equal to o_j and d_i is greater than d_j + d_n, the swap operation proceeds to next step. C_{o_i} and C_{o_j} (or C_{o_n}) are computed, which indicate the summation of the BP duration for the different offsets. Also, $U_{\mathrm{BP}}^{\max}$ and $U_{\mathrm{BP}}^{\min}$ are calculated as done in (4.1) and (4.2).

If C_{o_i} is greater than $C_{o_j} + 50$ and $d_i - (d_j + d_n)$ is smaller than or equal to $C_{o_i} - C_{o_j}$, the swap operation continues. Because d_i is greater than d_j + d_n, it is desired to evaluate C_{o_i} greater than C_{o_j} to decrease the maximum BP duration. C_{o_j} + 50 is used to eliminate unnecessary swaps. When the conditions are satisfied, $\hat{C}_{o_i}$, $\hat{C}_{o_j}$, F and $\hat{F}$ are computed back to back by the algorithm. Otherwise, the swap operation cancelled and next telegrams are examined.

If $\hat{F}$ is smaller than F, $\hat{D}_l$ is updated for the offset o_i and o_j. Then, $\hat{U}_{\text{BP}}^{\text{max}}$ and $\hat{U}_{\text{BP}}^{\text{min}}$ are computed according to $\hat{D}_l$. If $\hat{U}_{\text{BP}}^{\text{max}}$ is smaller than $U_{\text{BP}}^{\text{max}}$ and $\hat{U}_{\text{BP}}^{\text{min}}$ is greater than $U_{\text{BP}}^{\text{min}}$, the offsets are exchanged among o_i and o_j, o_n. Then, $\hat{O}$ is updated. After that, σ_{BP} and $\hat{\sigma}_{\text{BP}}$ are calculated. If $\hat{\sigma}_{\text{BP}}$ is smaller than σ_{BP}, the swap operation completed successfully and the assignments $\hat{O} = O$ and $\hat{D}_l = D_l$ are done.

Algorithm 9 S3T Pseudo Code

```
1: **Input:** $T$
2: **Output:** offset $o_i$ for each telegram $T_i$
3: **Initialize:** Set of assigned offsets $O = \hat{O} = \emptyset$; define list $T_k \subseteq T$ of telegrams
   with repetition $2^k$; $D_l = \hat{D}_l = 0$ for $j = 0, \ldots, N_{\text{MP}} - 1$
4: Apply basic heuristic without 3.9 constraint; obtain offset $o_i$ for each telegram $T_i$
   and $D_l$ for each basic period
5: for $0 \leq k \leq 10$ do
6:    if $|T_k| < 3$ then
7:       continue
8:    for $1 \leq i \leq |T_k|$ do
9:       for $i \leq j \leq |T_k|$ do
10:         for $j \leq n \leq |T_k|, i \neq j, j \neq n, i \neq n$ do
11:            Take telegrams $T_i$ and $T_j$
12:            if $o_i = o_j$ then
13:               continue
14:            Take the telegram $T_n$
15:            if $o_j = o_n$ and $d_i > d_j + d_n$ then
16:               Compute $U_{\text{BP}}^{\max}$, $U_{\text{BP}}^{\min}$ , $C_{o_i}$ and $C_{o_j}$ ( $C_{o_n}$ )
17:               if $C_{o_i} > C_{o_j} + 50$ and $d_i - (d_j + d_n) \leq C_{o_i} - C_{o_j}$ then
18:                  Compute $\hat{C}_{o_i}$, $\hat{C}_{o_j}$ , $F$ and $\hat{F}$
19:                  if $\hat{F} < F$ then
20:                     $\sigma_{\text{BP}}$, $\hat{\sigma}_{\text{BP}}$, $\hat{U}_{\text{BP}}^{\max}$ and $\hat{U}_{\text{BP}}^{\min}$
21:                     if $\hat{\sigma}_{\text{BP}} < \sigma_{\text{BP}}$ and $\hat{U}_{\text{BP}}^{\max} \leq U_{\text{BP}}^{\max}$ and $U_{\text{BP}}^{\min} \leq \hat{U}_{\text{BP}}^{\min}$ then
22:                        Swap operation is successfully completed, $O = \hat{O}$ and $D_l =$
                           $\hat{D}_l$
23:                     else
24:                        Don't swap. No change in $O$ and $D_l$ for $0 \leq l \leq N_{\text{MP}} - 1$
25:                  else
26:                     Don't swap. No change in $O$ and $D_l$ for $0 \leq l \leq N_{\text{MP}} - 1$
27:               else
28:                  Don't swap. No change in $O$ and $D_l$ for $0 \leq l \leq N_{\text{MP}} - 1$
29:            else
30:               Don't swap. No change in $O$ and $D_l$ for $0 \leq l \leq N_{\text{MP}} - 1$
```

For illustration, the same data set is used as in 4.1.2. The sorted list L is shown in Table 4.3 and the resulting initial schedule using MLB for the example telegram set using MLB is shown in Figure 4.3.

We consider that some iterations were already performed and the algorithm picks up the telegrams T_2 and T_3. $o_2 = 0$ is different than $o_3 = 1$ so the algorithm continues and picks the third telegram T_5 with $o_5 = 1$. Due to $o_3 = o_5 = 1$ and $d_2(265.7\,\mu s)$ is greater than $d_3 + d_5 = 211.4\,\mu$s, the swap operation proceeds to the next step. C_0 for o_2 and C_1 for $o_3 = o_5$ are computed as $1250.94\,\mu$s and $1089\,\mu$s respectively. Also, $U_{\text{BP}}^{\text{max}}$ and $U_{\text{BP}}^{\text{min}}$ are calculated as $625.47\,\mu$s and $544.5\,\mu$s.

Since C_0 is greater than $C_1 + 50$ and $d_2 - d_3 - d_5 = 54.3\,\mu$s is smaller than $C_0 - C_1 = 161.94\,\mu$s, the offsets are exchanged and $\hat{O}$ is updated. o_2 becomes 1 and $o_3 = o_5$ becomes 0, also D_l is updated. Then, $\hat{C}_0$ and $\hat{C}_1$ is calculated according to $\hat{O}$, as $\hat{C}_0 = 1194.64\,\mu$s and $\hat{C}_1 = 1143.3\,\mu$s. According to that, $F = 161.94\,\mu$s and $\hat{F} = 55.27\,\mu$s is obtained. Because $\hat{F}$ is smaller than F, the algorithm calculates $\hat{U}_{\text{BP}}^{\text{max}} = 603.17\,\mu$s and $\hat{U}_{\text{BP}}^{\text{min}} = 566.8\,\mu$s.

Because the conditions are satisfied, the algorithm calculates σ_{BP}, $\hat{\sigma}_{\text{BP}}$ as $16.39\,\mu$s and $1.91\,\mu$s respectively. Due to $\hat{\sigma}_{\text{BP}}$ is smaller than the $\hat{\sigma}_{\text{BP}}$ - 0.01, the swap operation is successfully completed and the schedule is obtained as shown in Figure 4.7. In this case, it is possible to reduce the maximum BP duration to $603.17\,\mu$s.

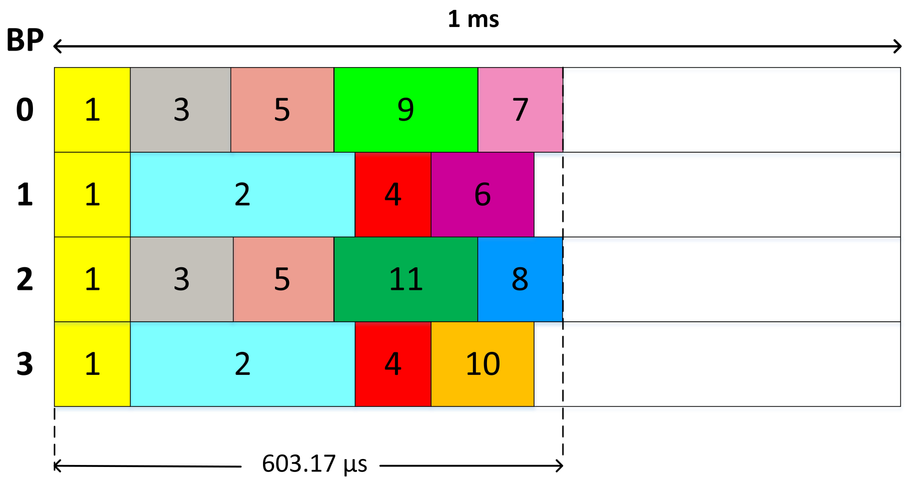


Figure 4.7: MLB schedule after S3T

4.2 Swap operations Evaluation

The defined swap operations help to obtain schedules close to or better than the ILP solution, improving the performance metrics BUs and σ_{BP}. Besides, the swap operations help to decrease the number of unsuccessful scheduling attempts of the basic heuristics.

For the evaluation of the swap operations, different algorithm combinations will be compared and the following terms will be used to define each of them.

$$\text{MLB*} = \text{MLB} + \text{SMB}$$

$$\text{MLB+} = \text{MLB} + \text{SSB}$$

$$\text{MLB++} = \text{MLB} + \text{SSB} + \text{S2T}$$

$$\text{MLB+++} = \text{MLB} + \text{SSB} + \text{S2T} + \text{S3T}$$

$$\text{SAB*} = \text{SAB} + \text{SMB}$$

$$\text{SAB+} = \text{SAB} + \text{SSB}$$

$$\text{SAB++} = \text{SAB} + \text{SSB} + \text{S2T}$$

$$\text{SAB+++} = \text{SAB} + \text{SSB} + \text{S2T} + \text{S3T}$$

4.2.1 More Frequent Telegrams

In this section, we evaluate the improvement achieved when applying the proposed swap operations to selected test cases with more frequent telegrams. We recall that these test cases have a small number of telegrams such that the optimal solution can always be found using the ILP formulation in Section 3.2. Hence, the first aim of this section is to evaluate the possible reduction of the number of unsuccessful (failed) instances of the heuristics in case of a very high bandwidth utilization close to 100%. The second aim of this section is to determine the quality of the proposed heuristics with respect to the known optimal solution.

When the more frequent telegrams are scheduled by the basic heuristics at a very high bandwidth utilization as in TC-10, the fail count per test case is as shown in Table 3.10.

Table 4.6: Fail Counts in Test Case TC-10

Algorithm	**Fail Count**
MLB	9
MLB*	3
MLB+	3
MLB++	3
MLB+++	3

According to the result of TC-10, SAB and MLB get 1 and 9 fails respectively while there is no fail when using the ILP. Because of the high BU and more frequent telegrams, MLB fails too much compared to SAB. When the swap operations are applied to MLB, the fail count is decreased drastically as stated in Table 4.6.

Consider the second test trial in TC-10 with the defined telegrams.

Table 4.7: Telegrams in TC-10

T_i	1	2	3	4	5	6	7	8	9
p_i [ms]	1	1	2	2	2	2	2	4	4
d_i [μs]	100.37	89.7	121.7	169.7	265.7	100.37	121.7	89.7	169.7
T_i	10	11	12	13	14	15	16	17	18
p_i [ms]	4	4	4	4	4	4	4	4	4
d_i [μs]	169.7	100.37	121.7	100.37	265.7	100.37	169.7	169.7	121.7

When the telegrams of TC-10 are scheduled by MLB, the last telegram T_8 in the ordered list L could not be placed by MLB because of the constraint defined in (3.11). The resulting basic period durations are shown below just before the placement of T_8

(recall that the maximum BP duration is $1000\,\mu$s):

$$\begin{aligned}\text{BP}_0 &= 969.23\,\mu s,\\ \text{BP}_1 &= 947.90\,\mu s,\\ \text{BP}_2 &= 969.23\,\mu s,\\ \text{BP}_3 &= 921.23\,\mu s.\end{aligned}$$

If the swap operations are applied during the schedule computation, the basic heuristic algorithms discard the maximum BP duration constraint in (3.11) and schedule all telegrams, potentially violating (3.11). When MLB is used with SMB or SSB, T_8 is placed with offset $o_8 = 3$ and the resulting basic period durations are as shown below:

$$\begin{aligned}\text{BP}_0 &= 969.23\,\mu s,\\ \text{BP}_1 &= 947.90\,\mu s,\\ \text{BP}_2 &= 969.23\,\mu s,\\ \text{BP}_3 &= 1010.93\,\mu s.\end{aligned}$$

Since BP$_3$ exceeds T_{BP}, it is considered as a failed scheduling attempt for the basic heuristic algorithm. Nevertheless, when the SMB is applied to that schedule, the algorithm first swaps T_9 ($d_9 = 169.7\,\mu s$) and T_{12} ($d_{12} = 121.7\,\mu s$) which have the offsets $o_9 = 3$ and $o_{12} = 1$. After swapping these telegrams, the resulting basic period durations are as shown below:

$$\begin{aligned}\text{BP}_0 &= 969.23\,\mu s,\\ \text{BP}_1 &= 995.90\,\mu s,\\ \text{BP}_2 &= 969.23\,\mu s,\\ \text{BP}_3 &= 962.93\,\mu s.\end{aligned}$$

That is, for this example, both SMB and SSB are able to determine a feasible schedule. In general, it holds that SMB and SSB help to decrease the fail counts from 9 to 3 for this test case. With the help of S2T and S3T, the fail counts could not be decreased anymore. When the swap operations are applied to SAB, fail count which already equals 1 is not changed.

Comparing SAB and MLB, it turns out that SAB still performs better than MLB even after implementing the swap operation. This can be observed looking at the performance metrics as shown in Figure 4.8 and Figure 4.9. When the MLB and SAB are supported by SSB, SMB and S2T, the maximum BU is decreased and the minimum BU is increased approaching the ILP which computes the optimal solution. It is seen that, despite the S3T operation, no further improvement is seen in the schedule.

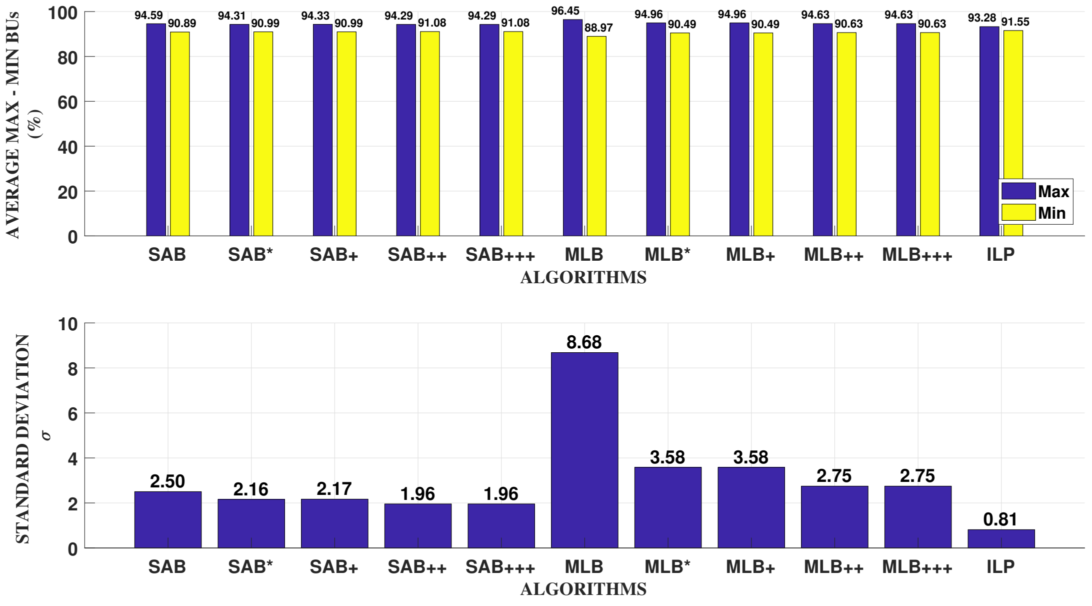


Figure 4.8: Performance Metrics for TC-10

When the swap operations are applied to basic heuristics MLB and SAB, σ_{BP} is improved with SMB, SSM, and S2T. Especially in TC-11, almost the same σ_{BP} obtained by SAB with the swap operations. As in BUs, S3T does not make any improvement according to σ_{BP}.

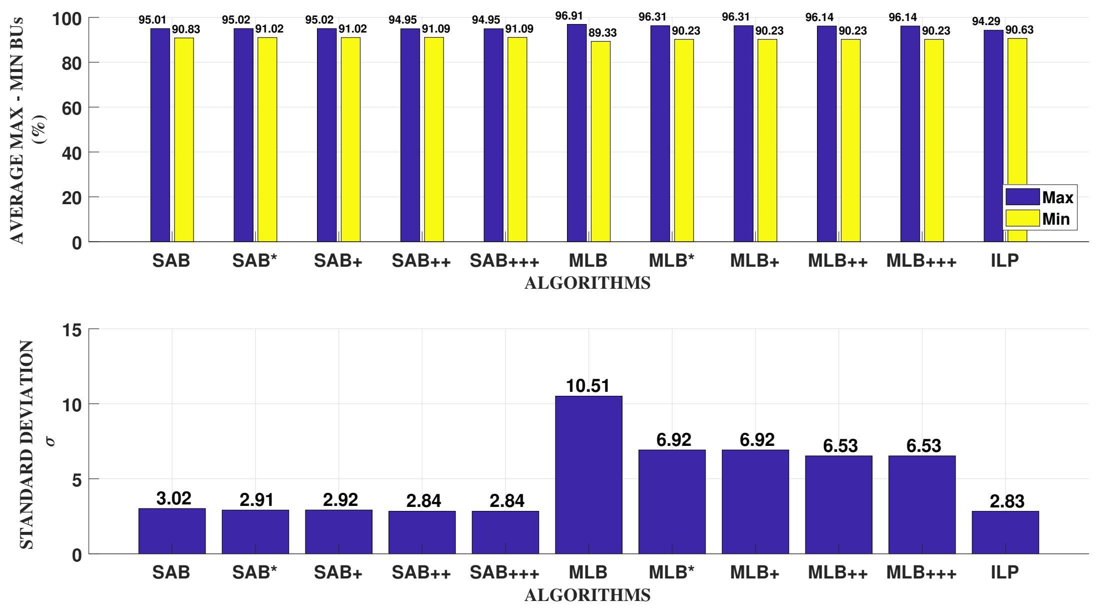


Figure 4.9: Performance Metrics for TC-11

4.2.2 Normal Telegrams

In this section, we evaluate the improvement achieved when applying the proposed swapping operations to selected test cases with normal telegrams that have medium r_i. We recall that despite these test cases that do not have a large number of telegrams, the optimal solution cannot always be found using the ILP formulation within the limited time in Section 3.2. While basic heuristics find a solution in a short time, ILP performs better according to performance metrics although timed out. Hence, the aim of this section is to see how basic heuristics are improved and approach the ILP best result when the swap operations are applied according to performance metrics.

As given in Table 3.17, ILP could not find an optimal solution in 30 minutes for the test cases TC-18 to 24 in any trial although the number of telegrams is not too high as shown in Table 4.8. ILP gives the best result within a limited time.

For TC-22, without any swap operation, SAB produces a better minimum BU and σ_{BP} as given in 4.10 compared to ILP. After all swap operations are applied, both MLB and SAB are improved and produce closer results to the best result of ILP for the BUs and σ_{BP}.

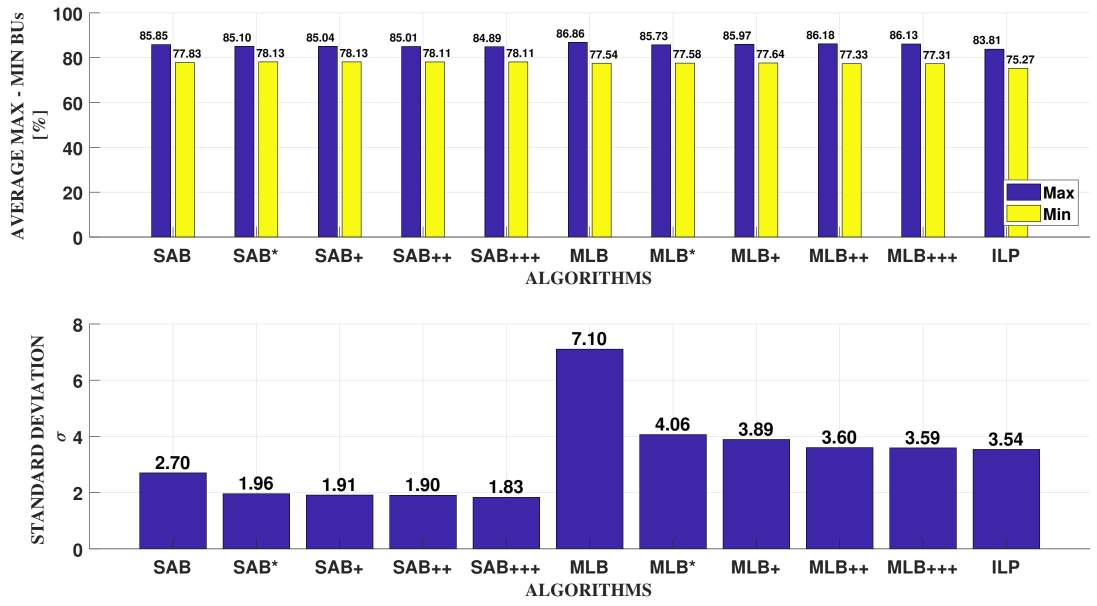


Figure 4.10: Performance Metrics for TC-22

Regarding the TC-23, the number of telegrams is not too large and BU is around %74. ILP could not find an optimal solution in any trial. Nevertheless, it still performs results well enough according to performance metrics as shown in Figure 4.11.

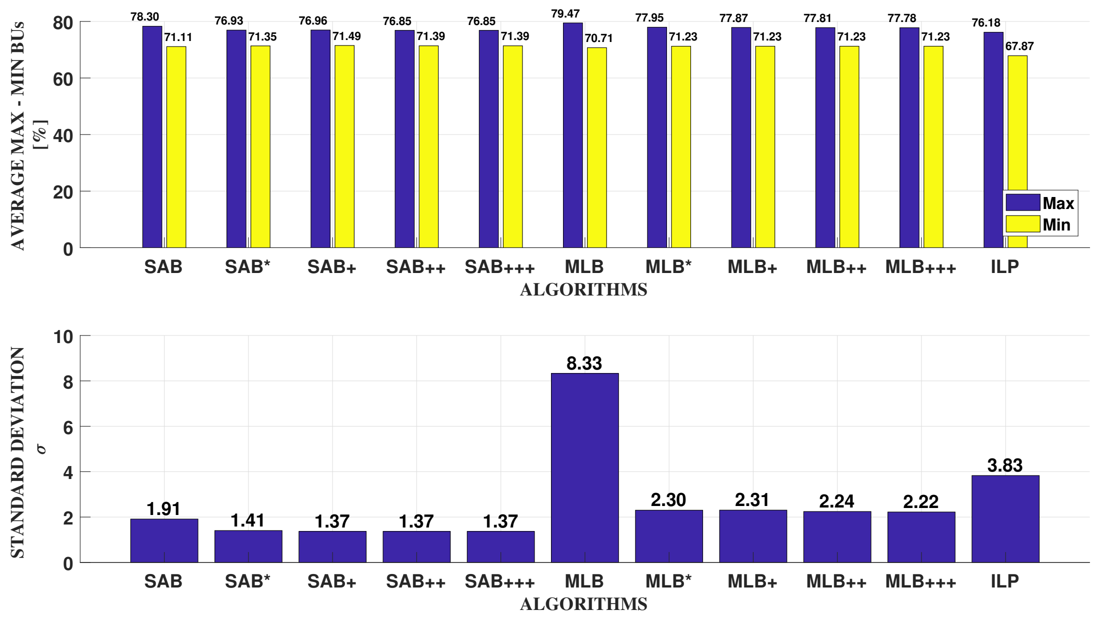


Figure 4.11: Performance Metrics for TC-23

On the other hand, MLB and SAB with the help of swap operations give close results for the maximum BU and better results for minimum BU compared to ILP. When σ_{BP} is examined, with the help of swap operations MLB and SAB schedules the telegrams more balanced. Especially in MLB, SMB and SSB help to reduce σ_{BP} drastically.

In TC-24, the total BU is increased to %84 and the number of telegrams is increased to 216. The ILP could not find an optimal solution but still makes the scheduling well-performed. MLB and SAB produce better σ_{BP} and minimum BU results compared to ILP without any swap operations. When SSB and SMB are implemented to SAB and MLB, the maximum BU value for SAB becomes better than ILP and MLB approaches the ILP as shown in Figure 4.12.

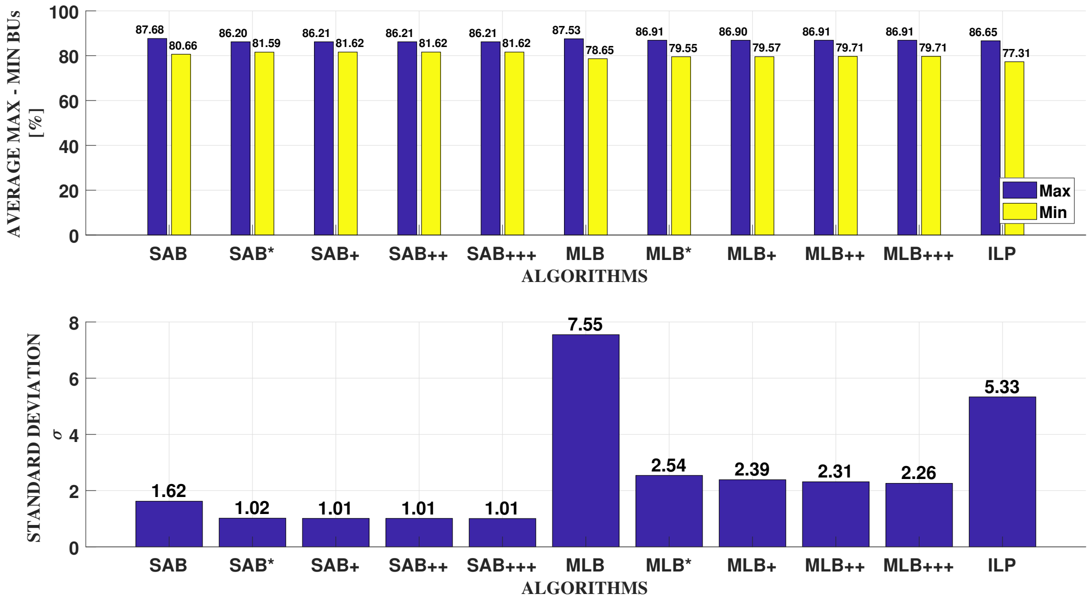


Figure 4.12: Performance Metrics for TC-24

The numbers in Table 4.8 indicate the algorithms as given in Figure 4.12 with the same order. Although the ILP times out for all the test cases, MLB and SAB schedules the telegram less than a second. Even if the swap operations are applied, the run time does not exceed 3 seconds, which is negligible compared to the ILP run time.

Table 4.8: Run Time (in seconds) For Normal Telegrams

	1	2	3	4	5	6	7	8	9	10	11
TC-22	0.03	0.07	0.07	0.09	0.5	0.001	0.05	0.04	0.7	0.48	TO
TC-23	0.03	0.1	0.09	0.1	1.1	0.001	0.09	0.08	0.09	1.09	TO
TC-24	0.04	0.15	0.13	0.14	2.81	0.002	0.13	0.12	0.12	2.79	TO

4.2.3 Less Frequent Telegrams

In this section, we evaluate the improvement achieved when applying the proposed swap operations to selected test cases with less frequent telegrams. We recall that these test cases have a large number of telegrams despite the low BU such that the optimal solution cannot be found using the ILP formulation in Section 3.2 when the BU above 20% and a number of telegrams is approximately higher than 600. Consequently, σ_{BP} becomes large, which means that the schedule is not balanced. Hence, the aim of this section is to evaluate the performance of the swap operations how they improve the basic heuristics which already find good results with respect to the best ILP solution according to performance metrics as in Section 4.2.2.

For TC-40 and TC-41, MLB and SAB produce much better schedules compared to the ILP looking at both BUs and σ_{BP} because of the large number of telegrams. In order to show results more comprehensible in graphs, σ_{BP} value for ILP is divided by 100.

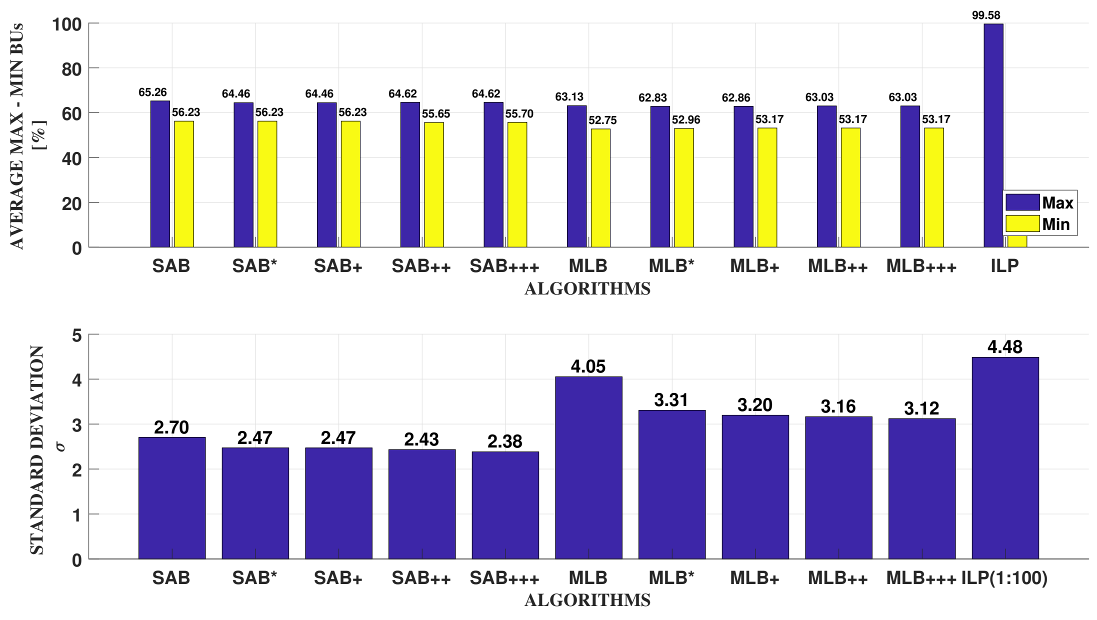


Figure 4.13: Performance Metrics for TC-40

In TC-40, the average telegram count is 1931 and BU is approximately %60. Swap operations help to improve the performance metrics but run time is increased when S2T and S3T are applied which is not desired as stated in Table 4.9. When the SSB is applied to both MLB and SAB, the best schedule is obtained according to the performance metrics.

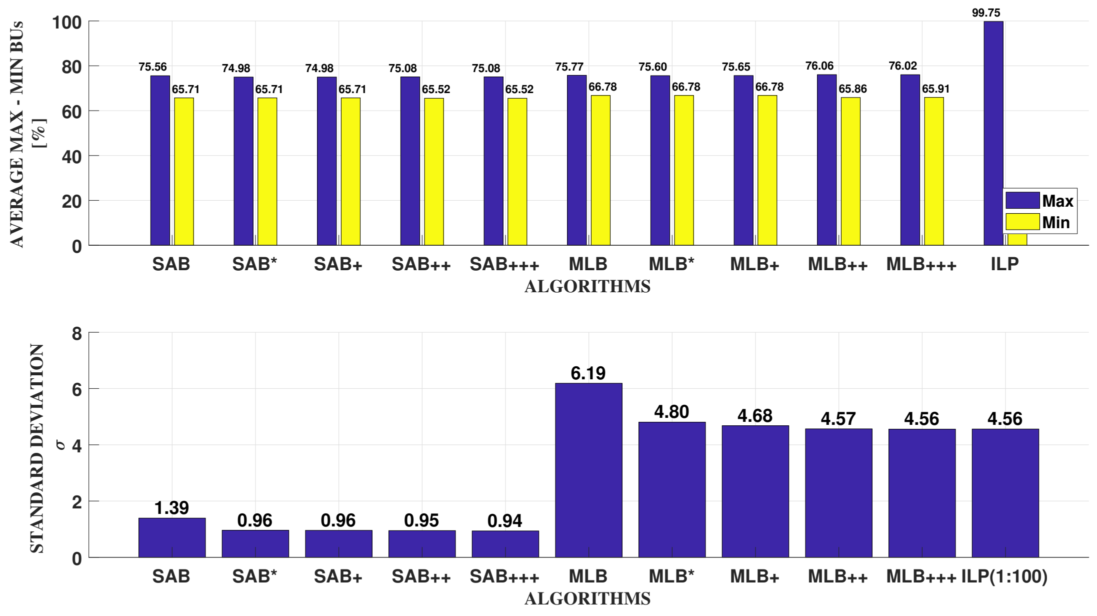


Figure 4.14: Performance Metrics for TC-41

Under a large number of telegrams (2313) and high BU (%72), MLB and SAB schedule the telegrams in a short time. When the swap operations are applied, small improvements are seen in maximum BU and σ_{BP}. If either S2T or S3T is applied, the run time is increased drastically as shown in Table 4.9.

Table 4.9: Run Time (in seconds) For Less Frequent Telegrams

	1	2	3	4	5	6	7	8	9	10	11
TC-40	2.62	211	74	561	850	0.3	185	111	505	792	TO
TC-41	3.12	253	137	1517	2052	0.04	434	370	1109	1645	TO

4.2.4 Equal Distributed Telegrams

In this section, we evaluate the improvement achieved when applying the proposed swap operations to selected test cases with equal distributed telegrams. We recall that these test cases consider the telegram sets with all possible BPs and a similar BU per period. Similar to Section 4.2.3, ILP cannot be solved if the number of telegrams is too large. Consequently, the schedule that is produced by ILP shows poor performance according to the performance metrics. The basic heuristics already find a solution in a short time but the aim of this section is to see how basic heuristics are improved when the swap operations are applied according to performance metrics.

A large number of telegrams causes a high BU and prolongs the run-time of the algorithms. As mentioned in Table 3.29 ILP could not find an optimal solution after TC-47 in 30 minutes. Consequently, the maximum utilization increased and minimum utilization decreased. As a result, σ_{BP} is increased due to the balance between basic periods.

When TC-52 with 1095 telegrams is scheduled by the algorithms, the results are obtained as shown in Figure 4.15. Both MLB and SAB produce better schedules than ILP but SAB makes the schedule more balanced compared to MLB in a short time as given in Table 4.10. When SSB and SMB are applied to MLB and SAB, the maximum BU is decreased and the minimum BU is increased as desired also σ_{BP} is improved. In this case, S2T and S3T are not able to improve the schedule.

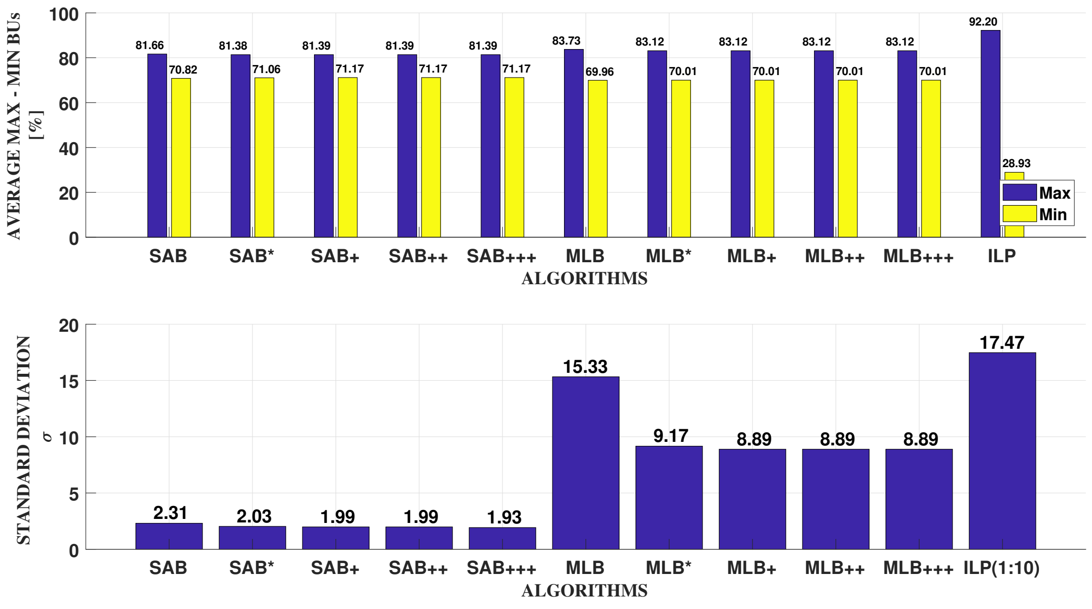


Figure 4.15: Performance Metrics for TC-52

In TC-51, the ILP could not find an optimal solution in 30 minutes for any trials whereas MLB and SAB schedule the telegrams in the set within 2 seconds achieving small values of σ_{BP} and maximum BU. When SMB and SSB are applied after the basic heuristics, the performance metrics are improved as shown in Figure 4.16. Although S2T and S3T are applied to basic heuristics, performance metrics are not improved any more.

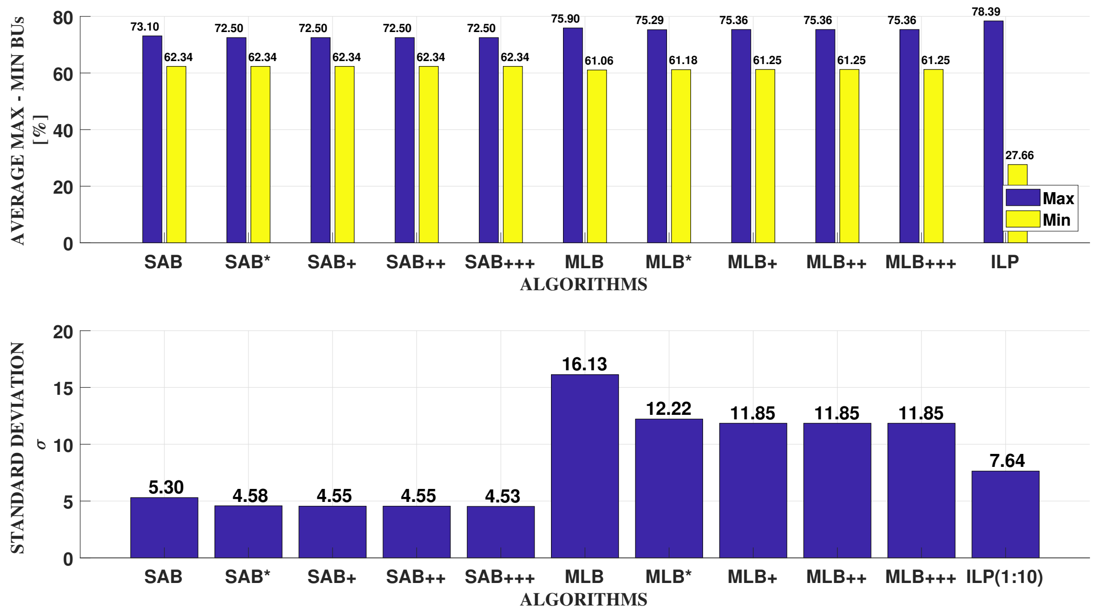


Figure 4.16: Performance Metrics for TC-51

The numbers in Table 4.10 indicate the algorithms as given in Figure 4.16 with the same order. While ILP times out, basic heuristics and swap operations schedule the telegrams in less than 2 minutes.

Table 4.10: Run Time (in seconds) For Normal Telegrams

	1	2	3	4	5	6	7	8	9	10	11
TC-51	1.39	45	33	33	56	0.02	66	41	41	64	TO
TC-52	1.58	37	23	23	58	0.02	105	68	68	103	TO

4.3 Discussion

In this section, we summarize the obtained results. As discussed in Section 3.4.3, SAB and MLB performed better than other basic heuristics in almost all the test cases according to the defined test scenarios and performance metrics. In addition, these heuristics generally find a schedule that is close to or better than the ILP solution taking into account the maximum BUs and standard deviation. Moreover, these heuristics generally have a much smaller run-time than the ILP.

According to our experiments, MLB performs well because it sorts the telegrams according to the d_i/p_i. Telegrams with a larger value of d_i/p_i occupy more space in the schedule looking at the overall duration and hence affect the balance and required BP duration more compared to telegrams with smaller values. SAB uses the same sorting of telegrams as MLB. In addition, it fills each basic period until a scaled average value that is calculated with different coefficient γ in order to get the most balanced schedule.

Although our evaluation shows that the basic heuristics can find schedules for large telegram sets, our results for small numbers of telegrams indicate that the basic heuristics can be improved in order to find close-to-optimal solutions. Since the basic heuristics place telegrams one-by-one picking up from the L, the following issues arise in the solution schedules:

1. Telegrams with a small duration can be scheduled in BP with a small duration, whereas telegram with a larger duration and the same r_i are scheduled in BP with a larger duration,
2. Two telegrams with the same offset are scheduled in a short BP while a telegram with the same r_i and a duration that is greater than the summation of the two telegram durations are scheduled in a longer BP.

For these situations, it would be beneficial to swap the positions of these telegrams in the schedule in order to obtain a better schedule with a smaller maximum BP duration. In order to address the states issues, four different swap operations are defined as mentioned. In addition to improving the performance metrics, these swap operations also help to convert unsuccessful test trials of the basic heuristic algorithms to successful test trials.

Regarding the swap operations, it is seen that SMB and SSB show a similar performance when applied to initial schedules from SAB or MLB for all the test scenarios. Hereby, SSB works faster than SMB if there is a very large number of telegrams as shown in Table 4.9 and Table 4.8. Both SMB and SSB are able to produce feasible schedules from infeasible initial schedules as shown in Table 4.6. While swap operations provide moderate improvements for SAB (which already produces very good schedules), they manage to bring MLB closer to SAB. As a result, the combination of SAB or MLB with the swap operations quickly computes schedules that are better than the schedules from the ILP for very large telegram sets. Finally, we note that SMB and SSB achieve greater improvements compared to S2T and S3T according to maximum BU and σ_{BP} as shown in Figure 4.15. The main reason for this observation is that SMB and SSB are directly applied to the initial solution of the basic heuristics, whereas S2T and S3T are applied to the resulting schedules from SMB or SSB. That is, according to our comprehensive evaluation it is sufficient to apply swap operations with only two telegrams.

CHAPTER 5

CONCLUSION

The Multifunction Vehicle Bus (MVB) is a highly robust real-time field bus for the data communication of control systems in rail-vehicles. MVB supports both periodic process data and sporadic message data transfers in the form of telegrams. Hereby, the schedule is organized in the form of basic periods (BPs) with a fixed length. The main focus of this thesis is on periodic telegrams. These periodic telegrams occur in the periodic phase (PP) of the MVB schedule and are scheduled in BPs according to their *period* starting from given *offset*. As a result, the fraction of a certain BP that is occupied for periodic data transmission is determined by summing up the duration of the telegrams that appear in that BP. Furthermore, the length of the PP is given by the duration of the maximum BP in the MVB schedule.

When computing an MVB schedule for a given set of telegrams, it is desired to minimize the longest BP duration in order to get a short PP and to leave sufficient time for the transmission of sporadic telegrams. In addition, MVB schedules should be balanced in the sense that the durations of different BPs should be close to each other. That is, the standard deviation σ_{BP} of the BP durations should be small. Finally, practical methods should be able to compute MVB schedules with run-times in the order of seconds or minutes.

The main objective of this thesis is the computation of MVB schedules for periodic telegrams. To this end, the first contribution of the thesis is the formulation of the optimal MVB scheduling problem for the PP of MVB as an integer linear programming problem (ILP). In addition, the schedules produced by ILP are investigated with respect to performance metrics such as the maximum BP duration and σ_{BP}. Hereby, different test scenarios are constructed according to the telegram periods. This evaluation shows that the ILP cannot be solved in cases with a large number of telegrams and a large bandwidth utilization. That is, it is not always possible to obtain an optimal MVB schedule based on the ILP formulation.

Consequently, as the second contribution of the thesis, five basic heuristics that schedule the telegrams in the cases that ILP could not find the optimal solution are developed. Moreover, the ILP solution and the schedules generated by the basic heuristics are evaluated, taking into account the described performance metrics. As a result of this evaluation, two basic heuristics, which are denoted as Scaled Average BP (SAB) and Minimum Longest BP (MLB) are found as the algorithms with the best performance. In particular, these algorithms determine MVB schedules with a small run-time even in cases where the ILP cannot be solved. Nevertheless, it is also observed that these heuristics cannot always find an optimal solution and might fail for very high bandwidth utilizations.

As the third contribution of the thesis, we proposed additional swap operations in order to find better schedules and to decrease the fail count. To this end, four different swap operations, Swap Operation According to Maximum BP Durations (SMB), Swap Operation According to Sum of BP Durations (SSB), Swap Operation Looking at the Next Two Telegrams (S2T) and Swap Operation Comparing Three Telegrams (S3T) are defined and evaluated. The comparison between ILP, basic heuristic algorithms and swap operations shows that especially SSB and SMB help to decrease the maximum BU and σ_{BP}. Besides, the number of fails is decreased. In summary, our comprehensive computational evaluation shows that the proposed basic heuristics and heuristics with swap operations outperform the ILP based optimal scheduling algorithm, that times out in the case of large telegram sets.

REFERENCES

[1] Insup Lee, Joseph YT Leung, and Sang H Son. *Handbook of real-time and embedded systems*. CRC Press, 2007.

[2] Ugur Keskin. In-vehicle communication networks: a literature survey. *Computer Science Report*, 10, 2009.

[3] Shane Tuohy, Martin Glavin, Edward Jones, Mohan Trivedi, and Liam Kilmartin. Next generation wired intra-vehicle networks, a review. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 777–782. IEEE, 2013.

[4] Shane Tuohy, Martin Glavin, Ciarán Hughes, Edward Jones, Mohan Trivedi, and Liam Kilmartin. Intra-vehicle networks: A review. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):534–545, 2014.

[5] Richard Zurawski. *Embedded Systems Handbook: Embedded systems design and verification*. CRC press, 2018.

[6] Steve C Talbot and Shangping Ren. Comparision of fieldbus systems can, ttcan, flexray and lin in passenger vehicles. In *2009 29th IEEE International Conference on Distributed Computing Systems Workshops*, pages 26–31. IEEE, 2009.

[7] Nicolas Navet and Françoise Simonot-Lion. In-vehicle communication networks-a historical perspective and review. Technical report, University of Luxembourg, 2013.

[8] Computer Network & Telematics. Introduction to in-vehicle networking: Generic protocols. 2007.

[9] Weiying Zeng, Mohammed AS Khalid, and Sazzadur Chowdhury. In-vehicle networks outlook: Achievements and challenges. *IEEE Communications Surveys & Tutorials*, 18(3):1552–1571, 2016.

[10] Xabier Iturbe, Jaime Jiménez, Aitzol Zuloaga, Jesús Lázaro, and José Luis Martín. The train communication network: Standardization goes aboard. In *2010 IEEE International Conference on Industrial Technology*, pages 1667–1672. IEEE, 2010.

[11] Mathieu Grenier, Lionel Havet, Nicolas Navet, et al. Scheduling messages with offsets on controller area network-a major performance boost. In *The automotive embedded systems handbook*. Taylor & Francis, 2008.

[12] Z. Li, L. Wang, Y. Yang, X. Du, and H. Song. Health evaluation of MVB based on SVDD and sample reduction. *IEEE Access*, 7:35330–35343, 2019.

[13] G. A. zur Bonsen. The multifunction vehicle bus (MVB). In *Proceedings 1995 IEEE International Workshop on Factory Communication Systems*, pages 27–34, Oct 1995.

[14] D. Ludicke and A. Lehner. Train communication networks and prospects. *IEEE Communications Magazine*, 57(9):39–43, Sep. 2019.

[15] Richard Zurawski. *Industrial communication technology handbook*. CRC Press, 2014.

[16] British-Standard-Institution. Iec std. iec 61 375-1:2012-06: Electronic railway equipment — train communication network (tcn) — part 1: General architecture. Technical report, British-Standard-Institution, June, 2012.

[17] Hubert Kirrmann and Pierre A Zuber. The iec/ieee train communication network. *IEEE Micro*, 21(2):81–92, 2001.

[18] Philip Koopman and Tridib Chakravarty. Analysis of the train communication network protocol error detection capabilities. *Institute for Software Research*, 2001.

[19] Train communication network - kunbus gmbh.

[20] Y. Jiang, H. Liu, H. Song, H. Kong, R. Wang, Y. Guan, and L. Sha. Safety-assured model-driven design of the multifunction vehicle bus controller. *IEEE Transactions on Intelligent Transportation Systems*, 19(10):3320–3333, Oct 2018.

[21] Zhu Jun, Li Fang, Wang Lifang, and Li Yong. Study on network dynamic performance of multifunction vehicle bus based on simulation model. In *2014 IEEE Conference and Expo Transportation Electrification Asia-Pacific (ITEC Asia-Pacific)*, pages 1–5, Aug 2014.

[22] Yizhao Wang, Lide Wang, Xiang Yan, and Ping Shen. Fuzzy immune particle swarm optimization algorithm and its application in scheduling of MVB periodic information. *Journal of Intelligent & Fuzzy Systems*, 32(6):3797–3807, 2017.

[23] Mustafa Çağlar Güldiken, Klaus Werner Schmidt, and Ece Güran Schmidt. Optimal telegram scheduling for the periodic phase of mvb. In *Digital Transformation & Smart Systems*, pages 1–5, Oct. 2019.

[24] British-Standard-Institution. Electronic railway equipment — train communication network (tcn) part 3-1: Multifunction vehicle bus (mvb). Technical report, British-Standard-Institution, August, 2012.

[25] Schlage. *MVB System User's Guide*. Duagon.

[26] Kalpana Rajagopal. Simulation of online bin packing in practice. *UNLV THESES, DISSERTATIONS, PROFESSIONAL PAPERS, AND CAPSTONES*, 2016.

[27] Nurul Afza Hashim, Faridah Zulkipli, Siti Sarah Januri, and S Sarifah Radiah Shariff. An alternative heuristics for bin packing problem. In *Proceedings of the International Conference on Industrial Engine*, page 1560, 2014.

[28] The Mathworks, Inc., Natick, Massachusetts. *MATLAB version 9.2.0.538062 (R2017a)*, 2017.

[29] IBM. Cplex optimization studio v12.8, 2018.