# Chapter 34
# Generation of New Detection Codes for GPS Satellites Using NSGA-II

**J. Sosa, Tomás Bautista, Daniel Alcaraz, S. García-Alonso and Juan A. Montiel-Nelson**

**Abstract** In this paper we obtain new detection codes, to determine whether a GPS satellite in particular is visible, using NSGA-II as multi-objective optimization engine. Our approach takes into consideration the length of the code and the sampling frequency in comparison with other approaches found in the literature that fix those design parameters. The obtained new detection codes produce an improvement of the 19 % in terms of CPU execution time. Results demonstrate that both design parameters must be taken in consideration to obtain high quality detection codes.

**Keywords** Genetic algorithms · GNSS · Gold codes · Low computational effort · Multi-objective optimization

## 34.1 Introduction

Nowadays, the detection of GPS signals for performing location tasks is one of the most commonly demanded applications [1]. In particular, the fast expansion of the mobile telephony, the increasing of the CPU capabilities and the reduced

J. Sosa (✉) · T. Bautista · D. Alcaraz · S. García-Alonso · J.A. Montiel-Nelson
Department of Electronic Engineering and Automation (DIEA),
Institute for Applied Microelectronics (IUMA),
University of Las Palmas de Gran Canaria (ULPGC),
35015 Las Palmas de Gran Canaria, Spain
e-mail: jsosa@iuma.ulpgc.es

T. Bautista
e-mail: bautista@iuma.ulpgc.es

D. Alcaraz
e-mail: dalcaraz@iuma.ulpgc.es

S. García-Alonso
e-mail: sgarcia@iuma.ulpgc.es

J.A. Montiel-Nelson
e-mail: montiel@iuma.ulpgc.es



D. Greiner et al. (eds.), *Advances in Evolutionary and Deterministic Methods for Design, Optimization and Control in Engineering and Sciences*, Computational Methods in Applied Sciences 36, DOI 10.1007/978-3-319-11541-2_34

battery charging times provide a wide market where GPS applications are potentially explored [2, 3]. However, in this scenario, since the GPS receiver just becomes one application running concurrently with some others in a single device, the requirements of low computational effort and low power consumption are mandatory [4]. Reducing the computational effort to determine whether a GPS satellite is visible or not is a hot topic in this research area [5–7]. In the existing literature three basic approaches are presented to cope with this problem. The first one consists in reusing most of the computation with additional hardware. This approach is called *split-sum methodology* [8]. Other authors propose to obtain a single detection code that allows to know if more than one satellite is visible or not [9]. Finally, in another approach [10] authors present a methodology to obtain GPS detection codes of 341 bits achieving a lower computational effort. In this paper we explore the idea of obtaining reduced length detection codes for GPS satellites presented in [10] using a multi-objective approach and we introduce as new optimization variables the length of the reduced code and the sampling frequency.

## 34.2 Problem Definition

In order to detect whether a satellite is visible or not, receivers compare the incoming GPS identifiers (at a frequency L1 of 1,575.42 MHz), with all the possible GPS satellite identifiers [11]. A satellite identifier is a Pseudo Random Number (PRN). Basically, a PRN is an array of binary digits where each digit is called *chip*. The length of the PRN array is 1,023 chips [12]. Each satellite has assigned an unique PRN as identifier. Every satellite transmits its own PRN identifier every millisecond. The comparison function is as follows:

$$Comp(PRN^{sat}, D^{rx}) = [a_1, a_2, \ldots, a_L] \tag{34.1}$$
$$a_i = \sum_{j=1}^{L} PRN^{sat}_{mod_L(i+j)} * D^{rx}_i$$

where $L$ is the length of the PRN, that is, 1,023 chips for each GPS satellite. $D^{rx}$ is the incoming radio frequency data that is acquired by the GPS analog front-end receiver. $PRN^{sat}$ is the PRN identification of the GPS satellite (*sat*). This function takes in consideration all the possible alignments between the incoming GPS identification data and the compared PRN. Therefore, this is the reason to obtain an array of values and not only a single value. The size of array $Comp(PRN^{sat}, D^{rx})$ is $L$, one value for each possible alignment between the incoming data and the compared PRN.

Following the theory, when a satellite is visible the array $Comp(PRN^{sat}, D^{rx})$ has an unique maximum. This maximum value is called Detection Peak ($D_P$). The location/index where the $D_P$ is placed in the comparison array is called code-phase. The code-phase determines the starting chip of the PRN sequence. The other values

in this array are lower than the $D_P$ value. These lower values are called Noise ($N$). If the compared satellite ($PRN^{sat}$) is not visible, all values in array $Comp(PRN^{sat}, D^{rx})$ are noise.

Nowadays, in the literature there exist multiple approaches to implement the detection Eq. 34.1. For instance, it is quite easy to translate this Eq. 34.1 from the time domain to the frequency domain and use the Fast Fourier Transform (FFT) and its inverse (IFFT) to obtain the same results. However, in practical GPS receivers there exist only two basic detection techniques [13]. Their main key in comparison with other approaches are their implementation simplicity and the required computational effort.

In one of these approaches the L1 incoming signal is oversampled, that is, every chip of the PRN is sampled more than once, so the $D_P$ grows with the increasing sampling frequency. In terms of Eq. 34.1, implementing the oversampling only requires to set the correct value to $L$.

The other solution, instead of increasing the sampling frequency, increases the sampling period. As a consequence, the recorded incoming GPS data contains more than one complete PRN sequence. Therefore, the Detection Peak increases its value in proportion to the increment of the sampling period, that is, the signal recording time is increased.

Stepping up the sampling frequency and/or the sampling period increases the sensibility of a GPS receiver when the GPS signal-to-noise ratio is too low. However, ordinary applications like open-sky navigation systems, i.e., typical GPS receiver for car tracking, only takes one of both methodologies with reduced increasing factors.

In this research, we propose to obtain new detection codes to determine whether a satellite is visible or not. The main feature of those new codes is their reduced length in comparison with the original PRN. Proposed new codes require lower computational effort than the traditional PRN [15]. Moreover, we are based on the approach presented in [10] where authors introduce a novel methodology to obtain new PRN detection codes using GA. In this previous approach, the research is focused in determining what kind of multi/single objective algorithm is more suitable for this type of application. However, in order to narrow the search space of the problem, authors fix the length of the new detection codes to a submultiple of the original length. This submultiple is 341 chips ($341 \times 3 = 1{,}023$).

Our proposal in this work is redefining the problem presented in [10] with two new variables to optimize. The first is the length of the detection code and the second is the sampling ratio. We define the sampling ratio as the size of the new detection code divided by 1,023 chips. Since we reduce the sampling ratio below the unity, we will use the term dropping ratio as a more adequate concept or definition. The approach presented in [10] has a length of 341 and a dropping ratio of 3 (that is, to take 1 sample and drop 2 for every 3 samples of the incoming signal). This results in a sampling ratio of 1/3 as shown in Fig. 34.1.

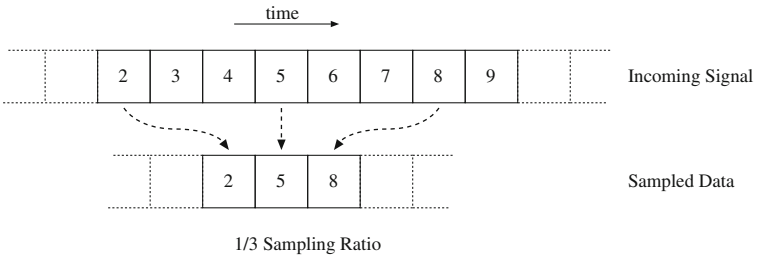


**Fig. 34.1** Example of 1/3 sampling ratio

## 34.3 Problem Codification

Based on the approaches found in the literature, we choose the NSGA-II [14] as GA multi-objective optimization engine. The codification for the individual is as follows. Each individual represents a new detection code. The individual is made by an array of Boolean values (see Fig. 34.2). The status of each Boolean value can be only a logic zero and a logic one. The unknown or error states are forbidden in this engine. The length of the individual array determines the length of the new detection code. Finally, each element of the individual array corresponds with an element of the new detection code; this makes that the index is the same in both arrays.

Unsurprisingly, the cost function is basically the Eq. 34.1. As before mentioned, this cost function is an array of comparison values. However we are not interested in all these values but our attention is focused on the Detection Peak and the Noise. Moreover, we know that the dropping ratio determines the total number of Detection Peaks in the comparison array. For example, if the length of the new detection code is set to 341 and the dropping ratio is set to 3, then there exist 3 different Detection Peaks. So the comparison array contains as many Detection Peaks as set the dropping ratio. Other values on this comparison array are Noise values.

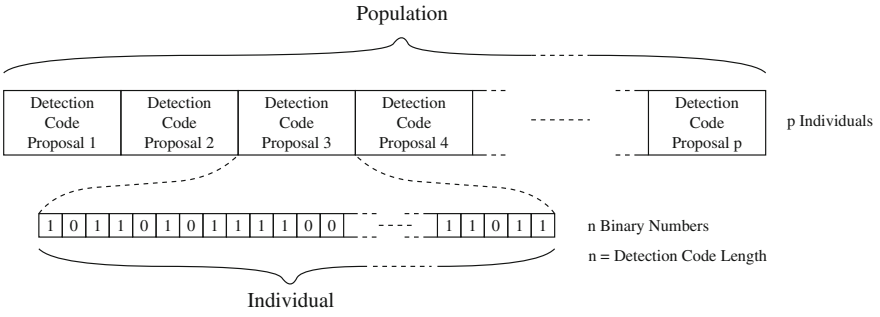


**Fig. 34.2** Problem codification

We are interested in obtaining new reduced GPS detection codes with a low Noise and high Detection Peaks. In order to obtain those values, after evaluating each individual with Eq. 34.1, the evaluation function extracts the minimum Detection Peak and the maximum Noise from the computed comparison array.

## 34.4 Experiments

We assume that the search space of this problem grows with the length of the new detection codes and dropping ratio used. If we provide all the degrees of freedom on all our problem variables, the required computational effort is increased in great matter. The magnitude of this computational effort produces non-practical CPU execution times. In this sense, in order to obtain the solutions in a practical CPU time, we include the following rule:

$$|Proposed\ length\ \times\ Dropping\ Ratio - 1{,}023| < \xi \tag{34.2}$$

where $\xi$ is the maximum allowed deviation between the original PRN and new proposed PRN.

Table 34.1 presents the NSGA-II optimization engine setup. We define two objective functions to optimize. The first one is to maximize $D_P$ (minimize $-D_P$) and the second is to minimize the Noise ($N$). The NSGA-II optimization engine is controlled by a program written in C that proposes lengths and sizes randomly using the rule introduced in Eq. 34.2.

In other words, our application (C program) proposes different lengths and dropping ratios and the NSGA-II optimization engine obtains new detection codes. That is, the application starts with a proposal of a new length and dropping ratio to obtain a new detection code. Then the NSGA-II optimization engine takes the control and produces a Pareto-Front curve for those parameters. Once the optimization is finished, our application extracts the best solution of the Pareto-Front. This procedure starts over and over with several lengths and dropping ratios. In addition, after extracting the best solution our application evaluates the convergence of the proposed

**Table 34.1** NSGA-II setup in our experiments

| Parameter | Value |
|---|---|
| Num. objectives | 2 |
| $OBJ_1$ | $-D_P$ |
| $OBJ_2$ | $N$ |
| Population | 40 |
| Generations | 10 k/100 k |
| Crossover | 0.8 |
| Mutation | 1/PRN length |
| Seed | Random/uniform distribution |

parameters. If the application determines that the solution can be refined/improved the NSGA-II is executed again with more generations (from 10 k to 100 k).

In order to measure the quality of new detection codes we introduce some concepts. We define the Detection Gap ($D_G$) as follows:

$$D_G = min(D_P) - max(N) \tag{34.3}$$

This means that the Detection Gap is the distance between the minimum Detection Peak ($min(D_p)$) and the maximum Noise ($max(N)$). Bigger Detection Gaps give better detection codes.

We also need to evaluate the required mathematical operations. For this purpose, we label the number of required multiplication/addition operations to execute Eq. (34.1) as CPU Operations ($CPU_O$). In this case, fewer CPU Operations require lower resources for hardware implementation.

Moreover, we define the CPU Effort ($CPU_E$) as the CPU Operations per Detection Gap, that is:

$$CPU_E = \frac{CPU_O}{D_G} = \frac{num.\ required\ mul/add}{min(D_P) - max(N)} \tag{34.4}$$

In our experiments we compare our proposal with the traditional methodology [15]. This traditional methodology consists of implementing Eq. (34.1) directly without any of our proposed improvements.

Table 34.2 presents some results of our application when we look for reduced codes of GPS satellite ID 1. In this experiment, we set $\xi$ to a maximum of 400 chips. The first and second columns of the table contain the proposed code length and the dropping ratio. The number of generations are shown in the third column (in times of $1k$ generations). The fourth column gives the value of the rule defined in Eq. (34.2). The following two columns exhibit the best optimized minimum Detection Peak ($min(D_P)$) and maximum Noise ($max(N)$). The seventh column indicates the Detection Gap ($D_G$). The following column measures the CPU Operations ($CPU_O$). Ninth column evaluates the CPU Effort $CPU_E$. Finally, the last column presents the difference ($CPU_{Diff}$) between our proposal and the traditional methodology for the CPU Effort ($CPU_E(our) - CPU_E(traditional)$). In addition, please note that the last two rows in this Table 34.2 show the best solution obtained in [10] and also using the traditional methodology, respectively.

As expected, results from Table 34.2 are better as greater is the total number of generations in terms of maximum Noise and/or minimum Detection Peak. Last column presents a comparison between the traditional methodology and our new reduced detection code. The comparison is done in terms of required computational effort and detection gap to determine whether a satellite is visible or not. The negative values in this column denote better solutions than using the traditional methodology.

From Table 34.2, we observed that, for instance, the combination $320 \times 3$ (length $\times$ dropping ratio) with 100 k generations and $212 \times 3$ with 10 k generations have similar values in last column, $-73$ and $-76$, respectively. The first one has the double

**Table 34.2** NSGA-II experiments for Satellite ID 1 and $\xi < 400$

| Code length | Dropping ratio | Generations ($\times$ 1 k) | Rule value | $min(D_P)$ | $max(N)$ | $D_G$ | $CPU_O$ | $CPU_E$ | $CPU_{Diff}$ |
|---|---|---|---|---|---|---|---|---|---|
| 384 | 2 | 10 | 768 | 206 | 64 | 142 | 147,456 | 1038.4 | −59 |
| 448 | 2 | 10 | 896 | 236 | 66 | 170 | 200,704 | 1180.6 | 84 |
| 480 | 2 | 10 | 960 | 250 | 70 | 180 | 230,400 | 1280.0 | 183 |
| 512 | 2 | 10 | 1,024 | 276 | 72 | 204 | 262,144 | 1285.0 | 188 |
| 212 | 3 | 10 | 636 | 90 | 46 | 44 | 44,944 | 1021.5 | −76 |
| 288 | 3 | 10 | 864 | 116 | 54 | 62 | 82,944 | 1337.8 | 241 |
| 304 | 3 | 10 | 912 | 126 | 56 | 70 | 92,416 | 1320.2 | 223 |
| 320 | 3 | 10 | 960 | 134 | 56 | 78 | 102,400 | 1312.8 | 216 |
| 336 | 3 | 10 | 1,008 | 140 | 60 | 80 | 112,896 | 1411.2 | 314 |
| 341 | 3 | 10 | 1,023 | 141 | 59 | 82 | 116,281 | 1418.1 | 321 |
| 256 | 4 | 10 | 1,024 | 90 | 52 | 38 | 65,536 | 1724.6 | 628 |
| 512 | 2 | 100 | 1,024 | 276 | 70 | 206 | 262,144 | 1272.5 | 176 |
| 288 | 3 | 100 | 864 | 136 | 54 | 82 | 82,944 | 1011.5 | −85 |
| 304 | 3 | 100 | 912 | 148 | 56 | 92 | 92,416 | 1004.5 | −92 |
| 320 | 3 | 100 | 960 | 156 | 56 | 100 | 102,400 | 1024.0 | −73 |
| 341 | 3 | 100 | 1,023 | 177 | 57 | 120 | 116,281 | 969.0 | −128 |
| 256 | 4 | 100 | 1,024 | 98 | 52 | 46 | 65,536 | 1424.7 | 328 |
| 341 | 3 | 40 | 1,023 | 177 | 89 | 88 | 116,281 | 1321.4 | 224[a] |
| 1,023 | 1 | – | 1,023 | 1,023 | 69 | 954 | 1,046,529 | 1097.0 | 0[b] |

$min(D_P)$ minimum Detection Peak, $max(N)$ maximum noise, $D_G$ Detection Gap, $CPU_O$ CPU Operations, $CPU_E$ CPU Effort, $CPU_{Diff}$ CPU Effort Difference
[a]Data from [10]
[b]Traditional methodology [15]

of detection gap than the second one, but the second proposal requires a half of the CPU Operations to evaluate Eq. (34.1). Therefore, this comparison demonstrates that there exist several combinations of code lengths and dropping ratios that have similar ratios of CPU effort and detection gap.

The results obtained with our application, as shown on Table 34.2, are always better than the presented in a previous work [10] in terms of maximum Noise. In case of the Detection Peaks our approach obtains at least the same or better values than those in the referred work [10].

Table 34.3 presents the summary of the results for all checked lengths. The first column shows the PRN length. The second, the third and the fourth columns provide the maximum, minimum and average computational effort difference (CPU difference) between our proposal presented in this document and the traditional methodology. The last column in this table gives the obtained improvement in percentage. A negative value here means that our proposal is better than the traditional methodology in the case referred to.

**Table 34.3** Obtained results from experiments for all satellites

| PRN length | $CPU_{Diff}$ | | | Improvement (%) |
|---|---|---|---|---|
| | Max | Min | Avg | |
| 256 | 114.26 | −159.25 | −44.38 | 4.05 |
| 264 | 27.13 | −339.43 | −202.94 | 18.50 |
| 272 | −40.09 | −342.06 | −208.95 | 19.05 |
| 280 | −65.42 | −328.37 | −170.86 | 15.57 |
| 288 | 55.00 | −267.56 | −173.64 | 15.83 |
| 296 | 26.28 | −270.43 | −135.58 | 12.36 |
| 304 | 303.24 | −241.30 | −116.91 | 10.66 |
| 312 | −38.91 | −227.86 | −115.25 | 10.51 |
| 320 | 40.78 | −229.20 | −98.23 | 8.95 |
| 328 | 47.51 | −215.16 | −80.58 | 7.35 |
| 341 | 114.26 | −159.25 | −44.38 | 4.05 |

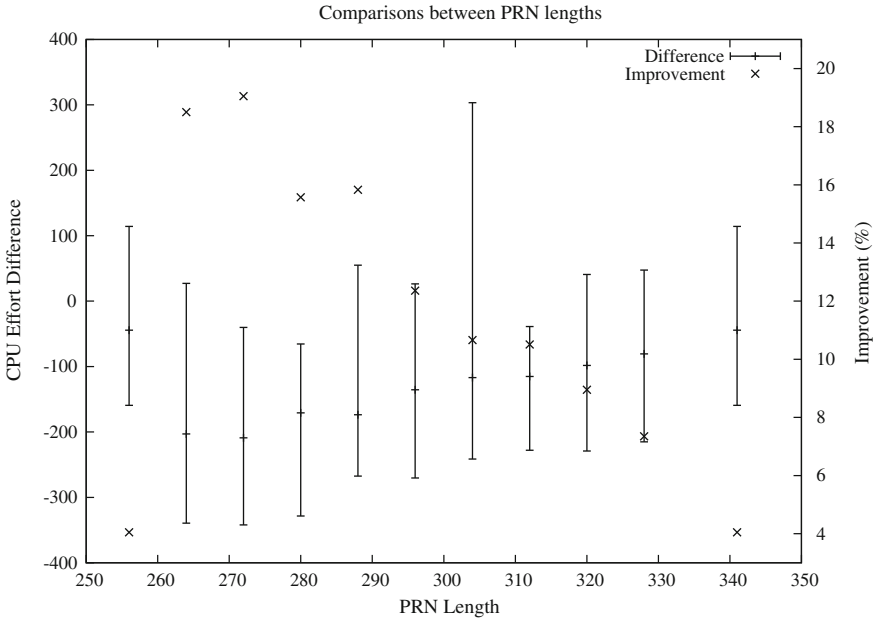


**Fig. 34.3** Results comparison in terms of PRN length and CPU effort improvement

Figure 34.3 display graphically the data on Table 34.3. On the left vertical axis is represented the computational effort difference units. In addition, on the right vertical axis the average improvement in percentage is shown. The horizontal axis gives the detection code length (proposed PRN length). In this Fig. 34.3, each vertical bar represents the maximum, minimum and average obtained improvement (left vertical axis). Finally, the × symbol identifies the improvement in percentage (right vertical axis).

The ideal goal in our application is to obtain a set of detection codes where the obtained improvement is equal for all satellites. In addition, due to hardware restrictions, all detection codes must have with the same length. These are moved to Fig. 34.3 looking for a bar (set of detection codes with same length) and its maximum and minimum as close as possible to the bottom of the figure. In this sense, there are two solutions very close with similar results, lengths 264 and 272, where the improvement in average is 18.50 and 19.05 % respectively. The advantage of length 272 set is not only the better percentage, but also this set provides improvements in all its detection codes (maximum, minimum and average improvements are negative values).

## 34.5 Conclusions

In this paper a methodology is proposed to take into account new design parameters to obtain reduced GPS detection codes to determine whether a satellite is visible or not. Those new design parameters are the length of the reduced detection code and the dropping ratio. New optimal detection codes are obtained using NSGA-II as optimization engine. In addition, we present a new metric to evaluate the performance of new detection codes in terms of required CPU effort. Results demonstrate that new detection codes exist that exhibit both similar or better performance in terms of CPU effort, detection gap, code length and/or dropping ratio.

**Acknowledgments** This work is patent pending and was funded under project BATTLEWISE (TEC2011-29148-C02-01) of the Ministry of Economy and Competitiveness.

## References

1. USAF Navstar GPS (2003) Where am I? Are we there yet? Air Space Power J 6(2):182–197
2. Kaplan E, Hegarty C (1996) Understanding GPS principles and applications. Artech House, Norwood
3. Duncan MJ, Badland HM, Mummery WK (2009) Applying GPS to enhance understanding of transport-related physical activity. J Sci Med Sport 12(5):549–556
4. Prasad R, Ruggieri M (2005) Applied satellite navigation using GPS, GALILEO, and augmentation systems. Artech House, Norwood
5. Borre K (2006) A software defined gps and galileo receiver: a single-frequency approach. Birkhuser, Boston
6. Hyoungmin S, Haeyoung J, Changdon K (2007) A new GNSS signal acquisition algorithm based on cross-correlation sequence with reduced signal-receiving time. In: International conference on control, automation and systems (ICCAS'07), pp 2563–2567
7. Chih-Hung W, Wei-Han S (2011) A study on GPS GDOP approximation using support-vector machines. IEEE Trans Instrum Meas 60(1):137–145
8. Gunawardena S, van Graas F (2006) Split-sum correlator simplifies range computations in GPS receiver. Electron Lett 42(25):1469–1471
9. Jan S-S, Lin Y-C (2009) A new multi-C/A code acquisition method for GPS. GPS Solutions 13(4):293–303

10. Sosa J, Montiel-Nelson JA, Nooshabadi S (2011) Low power GPS pseudo random numbers using genetic algorithms. In: Evolutionary and deterministic methods for design, optimisation and control with applications to industrial and societal problems (EUROGEN11), Capua, Italy
11. Lee S-W, Kim J, Jeong M-S, Lee YJ (2011) Monitoring atomic clocks on board GNSS satellites. Adv Space Res 47(10):1654–1663
12. Michalski A, Czajewski J (2004) The accuracy of the global positioning systems. IEEE Instrum Meas Mag 7(1):56–60
13. Hamza G, Motawie I (2009) Implementation of a complete GPS receiver using Simulink. IEEE Circuits Syst Mag 9(4):43–51
14. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comp 6:182–197
15. Dunn MJ (2012) Navstar GPS space segment/navigation user interfaces (IS-GPS-200). U.S. Coast Guard Navigation Center