Mini-Mémoire*

Pierre-Alexandre Blanc[1]

[1] Université libre de Bruxelles, Faculté des Sciences

June 1, 2018

Abstract

This article is a non-exhaustive comparative study of task sets generation algorithms.

1 Introduction

The goal of this article is to provide a non-exhaustive study of the most commons task sets generation algorithms, both for multiprocessors and for uniprocessors. Tasks are an abstraction of real time computing processes, constructed by focusing exclusively on the time required for the execution of that particular process, here via their utilizations, and the eventual deadline for that task. Those tasks are used to estimate how effective various real time scheduling algorithms are.

Task set generation is thus an important field of study for real time scheduling. Therefore a large number of algorithms have been created, with various goal(s), be it efficiency, lack of bias, limiting the length of the hyper-period, ... This paper present some of them, chosen for their relevance, and study them using the various procedures described in the methodology section.

The algorithms studied here are in order the Uuniform algorithm, the UuniFast algorithm, the UuniFastDiscard algorithm, Stafford's algorithm, Ripoll et Al algorithm and Goosens and Macq algorithm. They can be divided in two subsection, the algorithms limited to single processor task sets, Uuniform and UuniFast, and those that can produce multiple processor task sets. The last ones can once again be separated in the ones that produce task set with a set number of task, UuniFastDiscard and Stafford's algorithm, and the ones that do not, Ripoll et Al along with Goosens and Macq.

1.1 Glossary

Task : A task is a representation of a process. In this paper, we follow the sporadic task model common in the literature. A task is defined by their period T_i, their worst case execution time and their utilisation.

Number of task set : The number of task sets, m, is useful for algorithm that produce task sets for multiprocessor, and represent the number of those processor and thus the number of sub task sets within a task set.

Cardinality : The cardinality, n, of a task set is the number of tasks within that task set.

Period : A period or minimal interval time, T_i, is the time between the deadline and the start of the task.

Worst case execution time : The worst case execution time C_i, is the time the task would take in the worst possible case. It is inferior or equal to the period.

Utilization : The utilization of a task has been defined as $U_i = C_i/T_i$.

Total utilization : The total utilization of a task set is the sum of all task utilizations within that task set. It is defined as $u = \sum_{i=1}^{n} U_i$.

Hyper-period : The hyper-period is the length of the biggest period of the task set.

2 Methodology

2.1 Value for tests

Explanation : As some task sets groups are used multiple times in this paper, we will first present the arguments for them.

Uuniform 20 000 task set with the following parameters $n = 3$ and $u = 0.98$ compose this group.

UuniFast 20 000 task set with the following parameters $n = 3$ and $u = 0.98$ compose this group.

UuniFastDiscard 20 000 task set with the following parameters $m = 1$, $n = 3$ and $u = 0.98$ compose this group.

*Dans le cadre de l'unité d'enseignement INFO-F-308.

Stafford RandFixedSum 20 000 task set with the following parameters $m = 1$, $n = 3$ and $u = 0.98$ compose this group.

Ripoll Et Al 20 000 task sets with the following parameters, $m = 1$, a maximum compute time of 1, a maximum slack time of 1, a maximum delay of 0 and a target utilization of 0.98 compose this group.

Goossens and Macq This algorithm require a matrix M in addition to the various parameters discussed further.

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 & 4 & 4 & 4 & 8 & 1 \\ 1 & 3 & 3 & 3 & 3 & 9 & 9 & 27 & 27 \\ 1 & 5 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 7 & 7 & 7 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 13 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 17 & 17 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 19 & 1 & 1 & 1 & 1 \end{bmatrix}$$

This algorithm was run to produce 20 000 task sets with the following parameters, $u_1 = 0$, $u_2 = 0$, $d_1 = 0$, $d_2 = 0$, $o_1 = 0$, $o_1 = 0$.

2.2 Bias test

What is bias ? A task set generation algorithm is said to have bias when the repartition of task sets is not uniform, either in the number of tasks within a task set or within a set set cardinality.

Concept : According to the method described by P.Emberson, R.Stafford and R.I.Davis, a task set with a total utilization u containing n tasks can be plotted in a n-1 dimensional plane, using the utilization of each task as a coordinate.

Bias test for algorithm with known n : This part concern the algorithms with a set cardinality passed in argument, which are the following, Uuniform, UuniFast, UuniFastDiscard and Stafford's RandFixedSum. To verify the bias or lack of it within a specific algorithm we follow the concept previously described and create 20 000 task sets with $n = 3$ and $u = 0.98$. Because the task set cardinality is chosen, the uniformity and lack of bias of the task sets according to cardinality can be ensured simply by choosing the number of tasks within that task set according to an uniform distribution.

Bias test for algorithm with variable n : This part concern the Ripoll Et Al algorithm and the algorithm described by Gossens and Macq. First, this paper study whether the task set cardinality is an uuniform distribution by producing 20 000 task set with a total utilisation of $i = 0.98$ and studying the repartition of task sets cardinality. Secondly, knowing that to have no bias no part of the distribution of task set within the entire distribution may have bias, we can re-use the figure creation concept described higher, restricting it to the task set with a total utilization between 0.97 and 0.99 and discarding any task set with more or less than three tasks.

2.3 U-difference

The U-difference introduced by Bini and Buzzato[1] is a way to study the utilization disparity of a task set. It is defined as $\delta = (max\{U_i\} - min\{U_i\})/sum\{U_i\}$. To study it, we produced task sets group for every algorithm and plotted the U-difference of those task sets. The result is visible on figure 1.

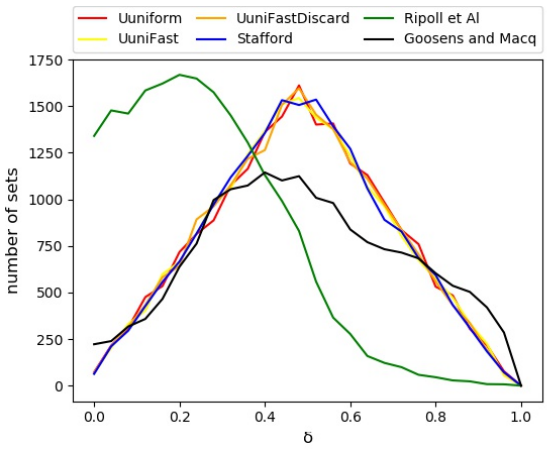

Figure 1: U-Difference result

The discussion for this result is found further in the paper.

3 Uuniform Algorithm

3.1 Goals and origin

The goal of this algorithm, developed by Bini and Buttazzo [2] is to create a set of task utilizations with no bias.

It is not used nowadays, as more efficient algorithms with no bias exist, and is only presented here for indicative reasons.

3.2 Hyper-period length

The length of the biggest period, the hyper-period, does not have an upper bound in this algorithm and thus as demonstrated by Goossens and Macq [6], this issue is liable to result in task sets that would be unlikely in reality.

3.3 Bias test

According to the methodology previously described, 20 000 task sets were produced using this algorithm and positioned. The result of it is visible on figure 2.

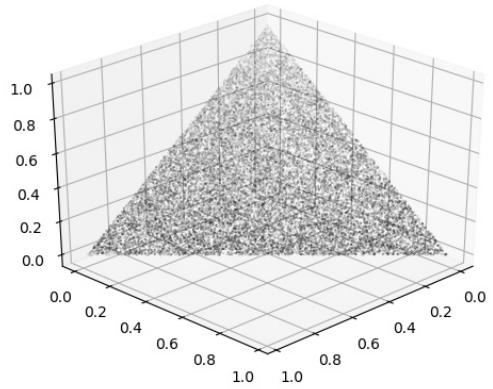

Figure 2: UuniForm algorithm

As can be seen the repartition of the task set is uniform, which indicate that there is no bias in the algorithm task set production. As was intended by the original creators of this algorithm.

3.4 U-difference

As can be seen on the figure 1, the repartition of U-differences of this algorithm is not biased either toward task set with a large U-difference or a small one.

3.5 Advantages

It give a baseline to base all future algorithms on. It is useful as a worse case task generator. By definition and experience it has no bias.

3.6 Inconvenients

Uuniform does not restrict the length of the periods, which can lead to massive hyper-periods, that are unlikely in real conditions. This algorithm is very inefficient. The probably of having a correct sum term (the n-1 first utilization being inferior to u) is $1/(n-1)!$, which make it unsuitable for actual use.

4 UuniFast algorithm

4.1 Goals and origin

The goal of this algorithm, also developed by Bini and Buttazzo [2], is to create a version of the UuNiform algorithm that is efficient enough to be used.

4.2 Hyper-period length

The length of the biggest period, the hyper-period, does not have an upper bound in this algorithm and thus as demonstrated by Goossens and Macq [6], this issue is liable to result in task sets that would be unlikely in reality

4.3 Bias test

According to the methodology previously described, 20 000 task sets were produced using this algorithm and positioned. The result of it is visible on figure 3.

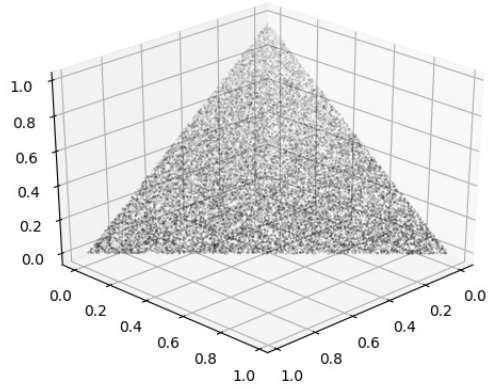

Figure 3: UuniFast algorithm

As can be seen the repartition of the task set is uniform, which indicate that there is no bias in the algorithm task set production. As was intended by the original creators of this algorithm.

4.4 U-difference

As can be seen on the figure 1, the repartition of U-differences of this algorithm is not biased either toward task set with a large U-difference or a small one.

4.5 Advantages

UUniFast is an efficient, unbiased task set generator for uniprocessor tasksets.

4.6 Inconvenients

UuniFast does not restrict the length of the periods, which can lead to massive hyper-periods, that are unlikely in real conditions. This algorithm is hard to expand to cases with a total utilisation for the taskset u greater than one, and thus for multiprocessor tasksets.

5 UuNiFast-Discard algorithm

5.1 Goals and origin

The goal of this algorithm, still developed by Bini and Buttazzo [5], is to create a version of the UuNiFast algo-

rithm that can be used to generate task sets for multiples processors.

5.2 Hyper-period length

The length of the biggest period, the hyper-period, does not have an upper bound in this algorithm and thus as demonstrated by Goossens and Macq [6], this issue is liable to result in task sets that would be unlikely in reality.

5.3 Bias test

According to the methodology previously described, 20 000 task sets were produced using this algorithm and positioned. The result of it is visible on figure 4.

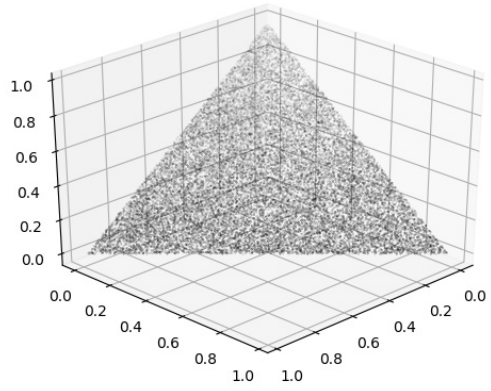

Figure 4: UuniFastDiscard algorithm

As can be seen the repartition of the task set is uniform, which indicate that there is no bias in the algorithm task set production. As was intended by the original creators of this algorithm.

5.4 U-difference

As can be seen on the figure 1, the repartition of U-differences of this algorithm is not biased either toward task set with a large U-difference or a small one.

5.5 Advantages

UUniFast-Discard is an unbiased task set generator for multiprocessors task sets.

5.6 Inconvenients

UuniFastDiscard does not restrict the length of the periods, which can lead to massive hyper-periods, that are unlikely in real conditions. The efficiency of this algorithm is low for large value of n, if u is close to n/2, as it will then discard a very large number of task sets.

6 RandFixedSum/Stafford algorithm

6.1 Goals and origin

The goal of this algorithm, discovered by Stafford [3], is the production of a multiprocessor task sets with no bias that do not require to discard parts of the result, thus ensuring its efficiency.

6.2 Hyper-period length

The length of the biggest period, the hyper-period, does not have an upper bound in this algorithm and thus as demonstrated by Goossens and Macq [6], this issue is liable to result in task sets that would be unlikely in reality.

6.3 Bias test

According to the methodology previously described, 20 000 task sets were produced using this algorithm and positioned. The result of it is visible on figure 5.

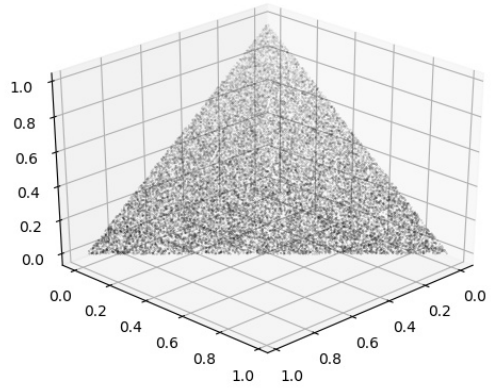

Figure 5: Stafford's RandFixedSum algorithm

As can be seen the repartition of the task set is uniform, which indicate that there is no bias in the algorithm task set production. As was intended by the original creator of this algorithm.

6.4 U-difference

As can be seen on the figure 1, the repartition of U-differences of this algorithm is not biased either toward task set with a large U-difference or a small one.

6.5 Advantages

This algorithm is an efficient multiprocessor taskset generator.

6.6 Inconvenients

Stafford's RandFixedSum does not restrict the length of the periods, which can lead to massive hyper-periods, that are unlikely in real conditions. This algorithm is also relatively complex to implement correctly, and slower than UuniFast for single processor task sets.

7 Rippoll et al algorithm

7.1 Goals and origin

The goal of this algorithm, discovered by Ripoll et al [4] [3], is the production of an efficient multiprocessor task set generator.

7.2 Hyper-period length

The length of the biggest period, the hyper-period, does not have an upper bound in this algorithm and thus as demonstrated by Goossens and Macq [6], this issue is liable to result in task sets that would be unlikely in reality.

7.3 Bias test

Knowing that this algorithm produce task set with a random cardinality, we need to study the uniformity or the lack of it, as described in the methodology. The result for this is the figure 6.

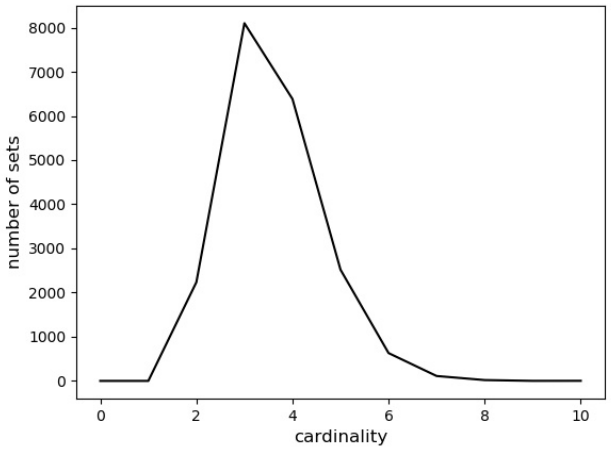

Figure 6: Ripoll et Al algorithm

As can be seen, the repartition of the set cardinality is not uniform. Therefore we need to verify if that issue can be rectified by dismissing all task sets outside of a chosen cardinality (to be able to use afterward an uniform distribution of cardinality). For this we use the previously discussed methodology for $n = 3$, with the afore mentioned characteristics for the 20 000 tasks. The result of this is the figure 7.

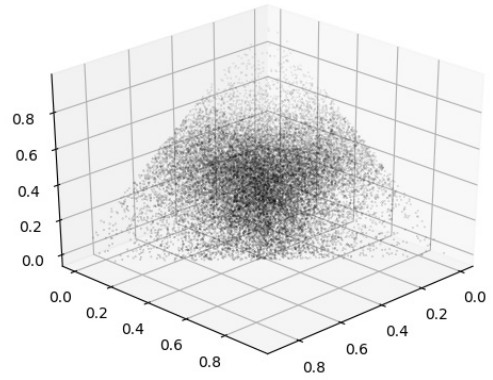

Figure 7: Ripoll et Al algorithm

As can be seen the repartition of the task sets is not uniform, being focused over the center of the distribution.

7.4 U-difference

As can be seen on the figure 1, the repartition of U-differences of this algorithm is biased toward task set with a low U-difference.

7.5 Advantages

This algorithm is an efficient computation wise multiprocessor taskset generator.

7.6 Inconvenients

Ripoll et al does not restrict the length of the periods, which can lead to massive hyper-periods, that are unlikely in real conditions. This algorithm confound task set cardinality and total utilization, which can lead to biased task sets, as seen in the bias test. This algorithm produce task sets with a small difference in length between the biggest and smallest period.

8 Goosens and Macq algorithm

8.1 Goals and origin

The goal of this algorithm, invented by Goosens and Macq [6], is to minimize the length of the hyper-period.

8.2 Hyper-period length

The length of the biggest period, the hyper-period, has an upper bound in this algorithm and thus the algorithm is more likely to result in task sets that are close to ones present in actual conditions.

8.3 Bias test

Knowing that this algorithm produce task set with a random cardinality, we need to study the uniformity or the lack of it, as described in the methodology. The result for this is the figure 8.

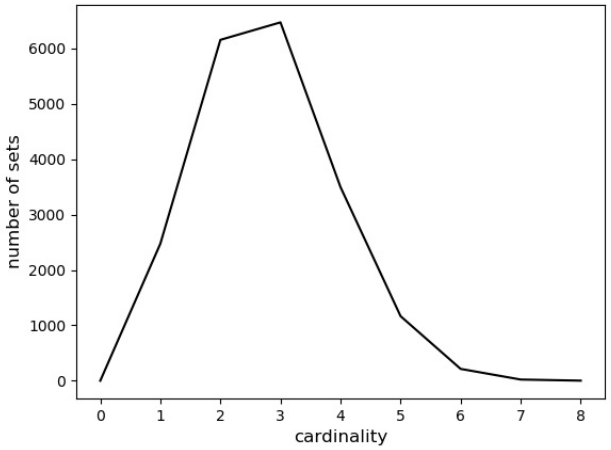

Figure 8: Goossens and Mack algorithm

As can be seen, the repartition of the set cardinality is not uniform. Therefore we need to verify if that issue can be rectified by dismissing all task sets outside of a chosen cardinality (to be able to use afterward an uniform distribution of cardinality). For this we use the previously discussed methodology, considering $n = 3$, with the afore mentioned characteristics for the 20 000 tasks. The result of this is the figure 9.

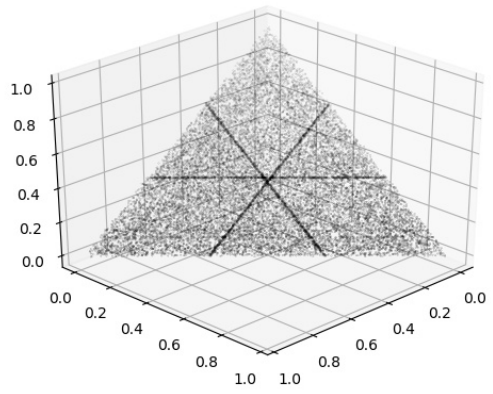

Figure 9: Goossens and Mack algorithm

As can be seen the repartition of the task set is relatively uniform. One using this algorithm need nonetheless to keep in mind the small bias.

8.4 U-difference

As can be seen on the figure 1, the repartition of U-differences of this algorithm has a relatively small bias toward task sets with a large U-difference.

8.5 Advantages

Goosens and Macq algorithm restrict the hyper-period length. This algorithm permit to explore the number of interruptions according to each schedulers and depending on the load.

8.6 Inconvenients

This algorithm show bias cardinality wise that can be solved via discarding task sets outside of a cardinality chosen via a random uniform distribution. This algorithm has a small tendency to produce task sets with a large U-difference. length

9 Conclusion

To conclude this paper, we would like to make a small recapitulation of the various algorithms and their uses.

Uuniform is a good algorithm with no bias to establish a baseline, but can be very slow as it discard sets and its period length is not restricted.

UuniFast is an efficient task sets producer with no bias for single processor, but its period length is not restricted which can lead to task set that are unlikely to occur in real conditions and it cannot produce task sets for multi processor.

UuniFastDiscard is an expansion of UuniFast, and therefore doesn't have bias either, but as it discard some task sets it can be very slow, and as its predecessor its period length is not restricted.

Stafford's random fixed sum is an algorithm with no bias for multi processor task sets but it has the issue of not restricting its period length which can lead to unlikely task sets.

Ripoll Et Al algorithm is a multi processor taskset production algorithm that is relatively fast but it can be heavily biased, both cardinality wise and within a set cardinality, and it doesn't constrict the period length. It has a tendency to produce task sets with a small U-difference.

Goosens and Macq algorithm is a multi processor that restrict period length, but has a little bias if we discard all task set outside of a cardinality chosen via an uniform distribution. It has a tendency to produce task sets with a large U-difference.

10 Appendix

10.1 Code

The code used to generate the figures is open-source and available on github. $https://github.com/RaisedDjed/mini\_memoire\_2018$

References

[1] BINI, E., AND BUTTAZZO, G. C. Biasing effects in schedulability measures. In *Proceedings of the 16th Euromicro Conference on Real-Time Systems* (Washington, DC, USA, 2004), ECRTS '04, IEEE Computer Society, pp. 196–203.

[2] BINI, E., AND BUTTAZZO, G. C. Measuring the performance of schedulability tests. *Real-Time Syst. 30*, 1-2 (May 2005), 129–154.

[3] CHERAMY, M. Simso. https://github.com/MaximeCheramy/simso, 2018.

[4] CHÉRAMY, M., HLADIK, P.-E., AND DÉPLANCHE, A.-M. SimSo: A Simulation Tool to Evaluate Real-Time Multiprocessor Scheduling Algorithms. In *5th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS)* (Madrid, Spain, July 2014), p. 6 p.

[5] EMBERSON, P., STAFFORD, R., AND DAVIS, R. *Techniques For The Synthesis Of Multiprocessor Tasksets*. 7 2010, pp. 6–11.

[6] GOOSSENS, J., AND MACQ, C. Limitation of the hyper-period in real-time periodic task set generation. In *In Proceedings of the RTS Embedded System (RTS'01* (2001), pp. 133–147.