Optimal pattern generation with priorities

Andrey Mokhov

andrey.mokhov@ncl.ac.uk

NCL-EEE-MSD-MEMO-2016-012

June 2016

Abstract

This memo discusses a formal approach to constructing custom patterns of light to be emitted using an LED array, applies the approach to a simple example, and discusses ideas for future work.

Optimal pattern generation with priorities

Andrey Mokhov
Newcastle University, UK

1 Motivation

The problem arises in the context of designing brain implants for suppressing epileptic seizures by emitting specially crafted patterns of light using an implanted LED array. See CANDO project website for more details [1]. This particular approach was inspired by a discussion of the µSystems research group on the topic of on-chip power switching.

2 Problem statement

Consider an array of n LEDs $\{L_1, \dots, L_n\}$.

A *pattern* of light with an integer *period* T is an assignment of ON or OFF states to each LED at each time moment $t \in \{1, \dots, T\}$. For example, if we denote ON or OFF states with symbols $\bullet$ and $\circ$, respectively, then the following is a pattern of light for 5 LEDs with period 8.

	1	2	3	4	5	6	7	8
L_1 :	○	○	○	○	○	○	○	○
L_2 :	●	●	●	●	●	●	●	●
L_3 :	○	○	○	●	●	●	●	●
L_4 :	●	●	○	○	○	●	●	●
L_5 :	○	●	○	●	○	●	○	●

In words, L_1 is always OFF, L_2 is always ON, L_3 is ON starting from $t = 4$, L_4 is OFF for $3 \leqslant t \leqslant 5$, and L_5 is ON whenever t is an even number.

A pattern is called *simple* if there are at most two changes between the ON or OFF states for each LED. Intuitively, this means that the pattern can be realised by switching an initially OFF LED ON (or, symmetrically, by switching an initially ON LED OFF) for some time *only once* within the period T. The above pattern is not simple, because L_5 has more than two state changes. If we drop L_5 the remaining pattern is simple. Indeed, L_1 and L_2 do not change their states at all; L_3 can be switched ON for the interval $t \in [4, 8]$, and L_4 can be switched OFF for the interval $t \in [3, 5]$.

In this memo we are interested in simple patterns that conform to the following requirements:

1. An LED L_k must be ON during exactly $a_k \leqslant T$ moments of time during the period T. This captures the integrating nature of the light-sensitive cells in the brain: the sequence of LED activations does not matter by itself, only the total amount of light emitted by L_k matters.

2. At most P LEDs can be ON at the same time. P represents the *power limit* of the system: we cannot afford to have more than P LEDs ON at the same time.

Note that the requirements are *feasible* if and only if the overall energy PT is enough for emitting the required amount of light, that is

$$\sum_{1 \leqslant k \leqslant n} a_k \leqslant PT.$$

We can now formulate **the problem of optimal pattern generation**.

Given: n, T, a_k, and P, such that $\sum_k a_k \leqslant PT$.

Result: a simple pattern satisfying the above requirements (1) and (2).

3 Solution

In this section we give a solution to the problem defined in Section 2. The solution is simple enough to be checked by an automated theorem prover, such as Coq [2] or Agda [3], which may be necessary for the certification of such safety critical systems as brain implants.

The solution is best explained by way of an example. Let $n = 5$, $T = 8$, $a = (5, 4, 2, 3, 6)$, and $P = 3$. Clearly, the feasibility condition is satisfied, as $5 + 4 + 2 + 3 + 6 = 20 \leqslant 24 = 3 * 8$.

Let us first attempt a simpler problem, where we do not have the limitation of the period T. Then we could trivially solve the problem by activating LEDs one at a time in sequence:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
L_1 :	●	●	●	●	●	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○
L_2 :	○	○	○	○	○	●	●	●	●	○	○	○	○	○	○	○	○	○	○	○
L_3 :	○	○	○	○	○	○	○	○	○	●	●	○	○	○	○	○	○	○	○	○
L_4 :	○	○	○	○	○	○	○	○	○	○	○	●	●	●	○	○	○	○	○	○
L_5 :	○	○	○	○	○	○	○	○	○	○	○	○	○	○	●	●	●	●	●	●

Note that the result is a simple pattern, that is we switch each LED On and Off only once, and furthermore the power limit is trivially satisfied (we always have exactly one LED On). Now our goal is to *fold* this pattern back into period T without violating its simplicity and power limit P. To do that we split the pattern into $P = 3$ subpatterns of period at most $T = 8$. Note that this is always possible in general due to the feasibility condition; indeed, the length of the unfolded pattern is $\sum_k a_k \leqslant PT$. The resulting subpatterns are shown below.

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \bullet & \bullet & \bullet & \bullet & \bullet & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ & \bullet & \bullet & \bullet \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ \end{bmatrix} \begin{bmatrix} 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\ \circ & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \bullet & \circ & \circ & \circ & \circ & \circ & \circ & \circ \\ \circ & \bullet & \bullet & \circ & \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \bullet & \bullet & \bullet & \circ & \circ \\ \circ & \circ & \circ & \circ & \circ & \circ & \bullet & \bullet \end{bmatrix} \begin{bmatrix} 17 & 18 & 19 & 20 \\ \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ \\ \circ & \circ & \circ & \circ \\ \bullet & \bullet & \bullet & \bullet \end{bmatrix}$$

We can now make the following key observation: subpatterns do not overlap, that is if we stack them on top of each other we will never see two On LED's at the same point. This is due to the fact that each LED can be On for at most T time moments (recall that $a_k \leqslant T$ according to the requirement (1) above). The result of stacking the subpatterns is shown below.

	1	2	3	4	5	6	7	8
L_1 :	●	●	●	●	●	○	○	○
L_2 :	●	○	○	○	○	●	●	●
L_3 :	○	●	●	○	○	○	○	○
L_4 :	○	○	○	●	●	●	○	○
L_5 :	●	●	●	●	○	○	●	●

The second observation is that the resulting stacked pattern trivially satisfies the power limit requirement (2). Indeed, by stacking P subpatterns, each with power consumption of 1 unit, we can never exceed the budget of P units.

The final and perhaps the most subtle observation is that the resulting pattern is simple. To prove this we consider two cases:

- If the interval of activity of a LED L_k falls completely into a single subpattern then the LED is either always ON or can be controlled by switching it ON and OFF only once, because the original unfolded pattern was simple (this case holds for L_1, L_3 and L_4 above).
- If the interval of activity of a LED L_k spans two neighbouring subpatterns, then the LED can be controlled by switching it OFF and ON only once, because there are exactly two changes of its state (this case holds for L_2 and L_5 above).

The above two cases are exhaustive, since the interval of activity of a LED cannot span more than two subpatterns (because $a_k \leqslant T$).

To conclude, the above procedure generates a simple pattern for LED activation which satisfies requirements (1) and (2).

4 Priorities

One can imagine a scenario when the feasibility constraint ($\sum_k a_k \leqslant PT$) is not satisfied, yet it is still desirable to generate a simple pattern for LED activation which is allowed to underdeliver on the set targets of LED activity a_k. The following value is called the *energy deficit*:

$$D = \sum_{1 \leqslant k \leqslant n} a_k - PT.$$

Energy deficit is positive when the feasibility constraint is not met.

One simple strategy for deciding how to save up enough energy to fit into the energy budget is to use LED *priorities*. Without loss of generality we assume that priorities are encoded in the order of LEDs, that is L_1 has the highest priority and L_n has the lowest priority. Now given a set of activity targets a_k we can start lowering them until the energy deficit is reduced to 0. If the deficit D does not exceed a_n then it is sufficient to reduce the lowest priority target a_n only: $a'_n = a_n - D$. If $D > a_n$ then we reduce a_n to 0 and start reducing target a_{n-1}, and so forth.

Eventually, we will obtain a feasible set of requirements and will be able to apply the procedure presented in Section 3.

There are other sensible strategies, for example, one can reduce the required activities simultaneously according to given LED *weights* w_k, etc. All such strategies can be implemented as a preprocessing stage whose task is to reduce energy deficit to 0.

5 Discussion

This memo was inspired by the discussion on a general framework for the specification, verification and synthesis of controllers that can generate signal patterns according to a set of requirements. The specific problem considered in this memo highlights a number of important issues that such a framework should be able to address:

- The state of signals in a given moment of time, called *code*, is a key notion for both functional and non-functional characteristics of the controller. In particular, in the studied problem it was important that a code does not contain too many On values (the power limit). It was also important that codes integrate to a desired target value (LED activity requirements).
- There may be constraints on *transitions between the codes*. In the studied problem we were interested in simple code patterns where signals do not change too often. Another issue which is worth mentioning is that since different signals cannot be perfectly synchronised, we could violate the power limit for a very short period of time when two signals were changing in opposite directions. This may be unacceptable for certain applications, in which case more complex transition strategies will have to be devised.
- The system may operate in a *dynamically changing environment*, in which case the controller will have to be able to adapt accordingly. For example the energy budget of a system may change unpredictably, and one may prioritise codes or individual signals within the codes to deal with the energy deficit.
- It is essential that the framework allows *formal reasoning and verification* in order to design safety critical systems, such as brain implants.
- The presented solution heavily relied on the *composition of code patterns* which allowed building complex patterns from simpler ones. We envisage that real-life systems will demand the ability to incrementally construct complex solutions from a library of simple primitives. Possible mathematical models for describing and composing code patterns are Petri Nets [4], Conditional Partial Order Graphs [5], and Parameterised Graphs [6].

References

[1] Cando project website: `http://www.cando.ac.uk/`.

[2] Coq proof assistant webpage: `https://coq.inria.fr/`.

[3] Agda proof assistant webpage: `http://wiki.portal.chalmers.se/agda/`.

[4] Tadao Murata. *"Petri nets: Properties, analysis and applications",* Proceedings of the IEEE 77.4, pp. 541-580, 1989.

[5] Andrey Mokhov. *"Conditional Partial Order Graphs"*, PhD thesis, Newcastle University, 2009.

[6] Andrey Mokhov and Victor Khomenko. *"Algebra of Parameterised Graphs"*, ACM Transactions on Embedded Computing Systems, 13 (4s), 2014.