European Journal of Control (2010)5:419–430

DOI:10.3166/EJC.16.419–430

European Journal of Control

Why Is Resorting to Fate Wise? A Critical Look at Randomized Algorithms in Systems and Control

Marco C. Campi*

Department of Electronics for Automation, University of Brescia, Brescia, Italy

During recent years, the systems and control community has experienced a growing interest in randomized algorithms, and there appears to be a wide-spread excitement for methods incorporating randomized features since these methods seem to be capable of transcending some traditional difficulties inherent in deterministic approaches, especially in relation to computational issues. Upon reflection, a randomized algorithm is nothing but a procedure where one or more steps are based on a random rule, that is, when using a randomized algorithm, at some stage instead of making a decision ourselves we call on fate to choose for us. But then, a question arises naturally: why should resorting to fate be any wise? Fate is not an expert of anything, all it does is choosing by chance. So, why should randomized be any good? A conscious use of randomized methods demands to question ourselves about this issue, and this write-up contains my reflections on this matter.

Keywords: Randomized methods, Worst-case approach, Probabilistic robustness, Problem solvability, Computational complexity

1. Introduction

1.1. Some Historical Notes

Recent years have witnessed a flurry of activity related to randomized algorithms in systems and control.

The birth of randomized methods traces back to the introduction of the Monte Carlo (MC) approach, a scheme for estimating an expectation $E[f(X)]$ from randomly extracted samples x_k of the random variable X (randomization of X), [1]. Shortly after, in an attempt to surmount the difficulties attendant to generating the samples x_k, the Markov Chain Monte Carlo (MCMC) approach was conceived, [2], which eventually led to the Metropolis-Hastings algorithm by the work of Hastings, [3]. See [4–7] for comprehensive treatments of MC and MCMC.

Monte Carlo methods permit one to estimate an expectation value and they provide effective tools for the *analysis* of probabilistically robust control schemes. However, they have little potentials for *synthesis* problems.

An independent literature was meanwhile initiated in the late 1960s by the work of Vapnik and Chervonenkis, [8, 9], aiming at establishing conditions for the *uniform* convergence of empirical means to the corresponding expectations: instead of estimating $E[f(X)]$, the goal turned to the much more technically difficult problem of estimating $E[f(X, \gamma)]$ – where X is a random variable

* E-mail: marco.campi@ing.unibs.it

Received 10 May 2009; Accepted 15 March 2010
Recommended by M. Gevers

with respect to which expectation is taken and γ is a parameter – for each and every instance of γ simultaneously. It was Blumer et al. [10] that established a fundamental connection between this uniform convergence problem and *learning* problems: if convergence takes place uniformly, minimizing the empirical error with respect to γ returns a nearly minimizer of the corresponding expected error $E[f(X,\gamma)]$. This way, randomized methods broke up the scene of optimization and synthesis. See [11] and [12] for book-length presentations of the learning theory; in particular, reference [11] treats side by side in a comprehensive way uniform convergence and learnability.

Only recently by the seminal work of Vidyasagar, [13], methods from learning theory were imported into the robust control community, and this gave birth to a whole new application domain of randomized methods, that of *average performance* controller *synthesis*: given a set of plants X with an associated probability measure describing the chance of occurrence of the different plants, Vidyasagar noticed that the expected performance of a controller defined by a parameter γ could be written as $E[f(X,\gamma)]$. Consequently, following [10], controller optimization could be conducted by randomization over X provided uniform convergence holds.

The work of Vidyasagar concentrates on average performance. However, it has to be said that the use of randomization is not restricted to problems where the performance index is an expected value $E[f(X,\gamma)]$. Following a different route, randomization can also be applied to min-max problems provided that the max requirement is relaxed in a probabilistic sense. Precisely, one may be content with minimizing the cost $\max_{x\in\bar{X}} f(x,\gamma)$, where the set of plants $\bar{X} \subset X$ is required to have large enough probability, say $prob\{\bar{X}\} \geq 1-\epsilon$. Said differently, this is a "chance-constrained" problem where one is allowed to disregard plants in X up to a chance ϵ of occurrence; when ϵ is pushed down to 0, this problem goes back to the standard min-max robust control formulation. Randomized methods for this chance-constrained approach have been recently given a solid mathematical foundation in the articles [14–16].

Within the general frame sketched above, randomized methods have become widespread in the systems and control community over the past 15 years. Without distinguishing among the many different streams of thought and certainly without any claim of completeness, some significant contributions are recalled here: analysis and synthesis control problems are dealt with in [17–29]; estimation is the subject of [30]; yet other contributions in control are [31–34]; algorithms for random extraction is the subject of a number of articles among which [35, 23, 36]. The reader is also referred to the recent book [37] for a broad overview of randomized methods in control.

1.2. Objective of This Article

This article is not a survey on randomized algorithms.

The idea of generating this write-up stemmed from observing that the field of randomized methods in systems and control has been developing fast in a multitude of different streams, a number of far-reaching methods have been developed and indeed randomization has asserted itself as an emerging methodology, and yet there is a lack of awareness, at least among the laymen, of the mechanisms that make this approach effective and profitable. Through this article, I want to present my view and understanding of these mechanisms. My hope is that this article can contribute to future developments of randomized algorithms by providing a better understanding on the underlying "philosophical reasons of being."

A fundamental observation is at the root of this write-up: a randomized algorithm is simply an algorithm where some steps are based on a random choice. That is, instead of choosing ourselves, we cease the privilege of choosing to fate. If somebody comes to our office, somebody we credit for being more knowledgeable than us on a certain matter, we are certainly well available to listen to his or her advice. However, fate is not an expert of anything. So, one question that occurs naturally is

why should resorting to fate be any wise?

This question will be a central focus of attention in this article.

2. Randomized Algorithms: A Way to Explore Grey Hues

Suppose that an *uncertain* system S belongs to a class $\mathcal{S}$ and that our goal is to solve some analysis or design problem for S. We may want for instance to design a stabilizing controller, or we desire to build a predictor, or we want to identify the system parameters, etc. An *algorithm* is a sequence of steps through which a solution to the problem is determined[1].

In most cases, the algorithm steps are *deterministic*, that is each step is a deterministic rule that – using results from previous steps and information on the system – returns an

[1] An algorithm is normally designed to be applicable to different situations, that is to different systems. When at the beginning of this paragraph we speak of "uncertain system" we refer to the class of systems to which the algorithm has to be applied, and the difference between one system and another system in the class does not necessarily stay in its parameters, it can as well be associated to different operating conditions as determined by the inputs (disturbances) profiles.

output for use in subsequent steps of the algorithm. Such a standpoint, however, can be generalized leading to the following definition of *randomized* algorithms:

> **a randomized algorithm is an algorithm where one or more steps are based on a random rule, that is – among many deterministic rules – one rule is selected at random according to a probability P. P is an artificial element introduced in the algorithm to improve the problem solvability and selecting this probability P is part of the algorithm design.**

The following toy-example makes the idea of random rule more concrete. This example will be reconsidered in the next Section 2.1, where we discuss in which sense moving from a deterministic algorithm to a randomized one may improve solvability of the problem.

Example 1: *A system S has a 'tunable parameter' γ. Another parameter θ also influences the system behavior and this second parameter cannot be selected by us, it instead describes uncertainty in the system. We would like to select γ so as to make the system gain μ close to* 1*, say* $\mu \in [0.9, 1.1]$.

In concrete terms, suppose the system is

$$y_t = (0.99 - |\theta - \gamma|)y_{t-1} + |\theta - \gamma|u_t,$$

and that both γ and θ are discrete parameters taking a value among the 10 possible values: $0.1, 0.2, \ldots, 1$. *An inspection of the system immediately reveals that $\mu \in [0.9, 1.1]$ is achieved corresponding to the white boxes in Table 1, while black boxes correspond to $\mu \notin [0.9, 1.1]$.*

Selecting a deterministic algorithm simply corresponds to selecting a fixed value for the tunable parameter γ, that is a column in the table. However, any choice leads to $\mu \notin [0.9, 1.1]$ for one value of θ so that $\mu \in [0.9, 1.1]$ for any value of θ cannot be achieved. A randomized algorithm consists instead of choosing a probability P by which a γ is selected; the simplest choice is a uniform P which gives probability 10% *to each γ value. Given any θ, this random selection approach leaves us with a probability* 10% *only of having $\mu \notin [0.9, 1.1]$, and this is the probability of selecting that γ which falls in the black box in the table.* ∗

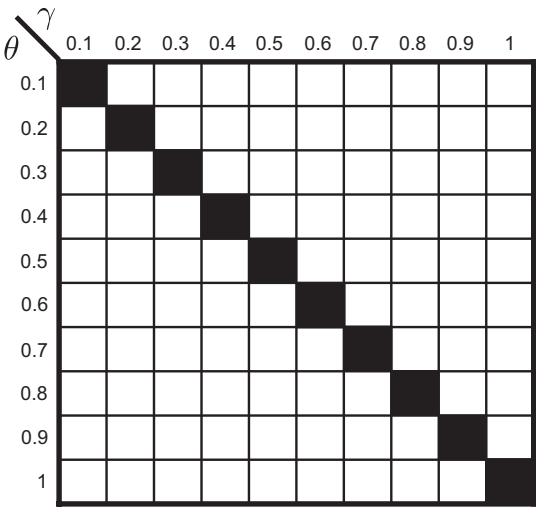

Table 1. Gain table: black boxes correspond to $\mu \notin [0.9, 1.1]$.

The above example, though very simple, contains some general elements that make the discussion to come more concrete.

2.1. "Successful" Algorithms

When applied to a given system S, an algorithm is successful if it achieves the result for which it has been designed. For instance, in the case of Example 1 the algorithm is successful if it attains $\mu \in [0.9, 1.1]$.

We want to see successfulness as a binary property: either YES, the algorithm was successful, or NO, it was not. Since a performance has often a continuum of possibilities, successfulness has at times to be judged on the basis of a minimum satisfaction level. If, for example, we desire short settling time in a control system, successfulness may be claimed when the settling time is below a chosen threshold time.

For deterministic algorithms, corresponding to a given system S, the algorithm is either successful or it is not. For instance, the algorithm corresponding to the 1st column of Table 1 is successful for all systems corresponding to the 2nd through the 10th row and it is unsuccessful for the system corresponding to the 1st row.

A deterministic algorithm is called a "successful algorithm" if it leads to success for all systems:

Definition 1 (successful deterministic algorithm): *A deterministic algorithm is* successful *if it is successful for all systems $S \in \mathcal{S}$.* ∗

Note that this definition is of worst-case nature, that is guarantee of success is required for any S.

Though appealing and natural, Definition 1 is very demanding and may lead to paralysis in the belief that the problem is not satisfactorily solvable: when a deterministic algorithm is designed for application to a variety of situations, a successful solution may be impossible to achieve for all situations simultaneously. In the case of Example 1, for example, no deterministic algorithm works in securing $\mu \in [0.9, 1.1]$ for all systems.

However, we have seen in Example 1 that a randomized approach can lead to that $\mu \in [0.9, 1.1]$ for all systems with

probability 90%. This observation suggests relaxing the deterministic Definition 1 and this leads to the following alternative:

Definition 2 (probabilistically successful algorithm): *A randomized algorithm is successful with probability p if, for all systems $S \in \mathcal{S}$, its probability to be successful is at least p.* ∗

This definition generalizes Definition 1 in that a deterministic algorithm is just a particular case of a randomized algorithm where probability P is concentrated on a single algorithmic choice.

Achieving a successful result *with high probability* as opposed to *deterministically* corresponds to accept a compromise and is often a wise choice:

> **when full-guarantee is not possible, one had better head for a** 90%**-guarantee instead.**

It is then a matter of personal sensitivity – also related to the nature of the problem – evaluating whether or not the selected chance of failure is tolerable. There are properties like robust stability where perhaps we are not ready to compromise on, but for many other properties a compromise makes perfect sense. Moreover, the chance of failure is for many randomized algorithms a parameter that can be tuned by the user, so that the user can push it below his or her level of tolerability. In Section 4, we provide examples of nontrivial problems that cannot be solved deterministically but that are amenable to a randomized solution carrying a high probability of success. Thus, parameter p in Definition 2 relaxes the deterministic notion of solvability and, according to this relaxed notion, problems that are not deterministically solvable may turn into probabilistically solvable problems.

One point which should be remarked is that the degree of freedom provided by p cannot be exploited as long as we stick to deterministic algorithms: for a deterministic algorithm, the probability of success can only be 0 or 1. So to say, the situation is either black or white. Moving to randomized algorithms instead gives full sense to the notion of "probabilistically successful" algorithm. For randomized algorithms, p becomes a continuous parameter ranging over the $[0, 1]$ interval and

> **when a grey hue is acceptable, a randomized algorithm provides us with an extraordinary opportunity to satisfactorily solve problems otherwise deemed intractable.**

Remark 1: *One thing that should be clearly recognized is that deterministic vs. randomized is not the dichotomy of worst-case vs. something else: both Definitions 1 and 2 require that the result holds for all possible systems in $\mathcal{S}$, that is both are of worst-case nature.* ∗

2.2. A Game-Theory Interpretation

A *deterministic* worst-case approach can be interpreted as a 1-step 2-players game: we first select the algorithm, and then "devil"[2] selects the system. Devil selects second, so that it can calibrate its selection to hit our weaknesses.

When moving to a randomized method, this state of things changes in that the game becomes 2-steps: we select the randomized algorithm first and devil selects the system in turn. At this first step, the algorithm is known to devil, but the actual random selection is not. Then, the 2nd step takes place and it is our move again: by random extraction, we select a rule. In this way, devil cannot counteract us as effectively as in the case of deterministic algorithms since it has not full knowledge of our random choices at the time it selects the system. The net result is that – even when a successful deterministic algorithm does not exist – a randomized algorithm can often be designed that is successful with high probability.

2.3. An Alternative: Bayesian Approach

A different probabilistic approach – of Bayesian nature – is also possible.

Let us take a look at Table 1. Suppose that the system (i.e. the row in the table) is selected according to a probability, say Q. If we now concentrate on a deterministic algorithm (e.g. the one represented by the first column in the table), then we can conclude that this algorithm is successful with high probability: if, for example, Q is uniform, the probability of success is 90%.

Two important differences between this Bayesian perspective and the randomized approach need be highlighted:

(1) in the Bayesian approach, the conclusion is no longer worst-case, it only holds with high probability with respect to the system.
(2) Q in the Bayesian approach and P in the randomized approach have two completely different meanings: in the Bayesian approach, Q describes how probable a system is as compared to the other systems; thus selecting Q is part of the modeling of the problem. In contrast, P in the randomized approach is something *we artificially select and use*. It only exists in our algorithm and, therefore, we cannot run into the problem of poor modeling, a fact that can happen with Q in the Bayesian approach.

[2] We use the word "devil" to signify that the opponent is not neutral, it selects with the intention to damage us.

2.4. An Intrinsic Limit to the Probability of Success

There exists an intrinsic limit to the probability of success of any randomized algorithm and this limit is related to the probability of success in a Bayesian framework. Precisely, let $\mathcal{R}$ be the set of rules R among which the randomized algorithm selects and further let Q be any Bayesian probability over $\mathcal{S}$. Then,

$$p \leq 1 - \min_R E_Q[I(R,S)], \tag{1}$$

where $I(R,S) = 1$ if R is unsuccessful for S and it is 0 otherwise, and expectation $E_Q[I(R,S)]$ is with respect to S for any fixed R. In words, (1) says that the probability of success of any randomized algorithm cannot be better than 1 minus the probability of failure of the best deterministic rule R applied in a Bayesian context. Therefore, if a Bayesian probability Q exists such that no deterministic rule performs satisfactorily (that is $\min_R E_Q[I(R,S)] =: \alpha$ is unsatisfactorily large), then the probability of success of a randomized algorithm is also bounded away from 1 by this α. This fact immediately follows from the well-known Yao's principle, see [38] for details.

Example 2 (Example 1 continued): *To illustrate Yao's principle, consider again Example 1. According to the Bayesian framework, pick a probability Q over the system class, that is a discrete probability over the possible values* $0.1, 0.2, \ldots, 1$ *for* θ. *Since there are 10 possible values for* θ, *if Q is not uniform there certainly exists a* θ *value such that* $Q(\theta) < 10\%$. *Say that this* θ *value is* $\theta = 0.3$. *Referring to equation (1), by selecting* $R = \gamma = 0.3$ *we have:* $\min_R E_Q[I(R,S)] \leq E_Q[I(0.3,S)] = Q(0.3) < 10\%$. *If, on the other hand, Q is uniform over* θ, *it is easy to see that* $\min_R E_Q[I(R,S)] = 10\%$. *Thus, (1) leads to the conclusion that:*

$$p \leq 1 - 10\% = 90\%. \tag{2}$$

Indeed, $p = 90\%$ *is attainable for, as we saw, a uniform P over* γ *has a probability of success of* 90%. *Seeking to improve this probability is hopeless since (2) establishes that* $p = 90\%$ *is the best possible result.* ∗

2.5. Randomized Algorithms and Random Data

An algorithm – either deterministic or randomized – is a sequence of steps through which a solution to a problem is determined. One or more of these steps can use specific information on the system $S \in \mathcal{S}$ at hand. Such information can come in two different forms:

(i) a priori information, that is we a priori know that S belongs to a certain *subset* of $\mathcal{S}$;

(ii) a posteriori information, that is data collected from the system.

For instance, in Example 1 our a priori information was that the system parameter θ was in the set $\{0.1, 0.2, \ldots, 1\}$, while no a posteriori information was available, that is we did not run any experiment on the system to reduce uncertainty on θ.

When the system is stochastic, a posteriori information accrued through data is random. Note that use of random information does not qualify an algorithm as a randomized algorithm: even though data are random, if the rule through which the solution is found is deterministic the algorithm is not randomized. If, for example, we estimate a system parameter using least squares, data can well be stochastic, but the rule for processing the data is deterministic so that least squares is not a randomized algorithm. What qualifies an algorithm for being randomized is the presence of an artificial probability P through which some algorithmic choices are made.

3. Drawing Some Mid-Paper Conclusions

Extracting the fundamental elements of the discussion so far, we can say that, at an abstract level, a randomized algorithm is just a collection of deterministic rules among which one specific rule is selected according to a random choice dictated by an artificial probability P. Using a random choice gives sense to the notion of probability of success and, when no successful deterministic algorithm exists, a randomized algorithm can still carry a high probability of success for all systems S.

Said with different words,

> **adding a randomized element to an algorithm gives us the possibility to simultaneously consider more deterministic algorithms (with a probability attached to each of them) with the result that the probability of success for each single S is given by an average among many different algorithms. This way, the black boxes in a table like Table 1 are averaged by the rows turning into greys. It is just a matter of perspective, but, if we are sensitive to the worst, greys may be more palatable than an alternation of blacks and whites. A randomized algorithm just offers us the possibility to choose this more palatable opportunity of doing things.**

Importantly, the level of grey, that is the probability of success, is computed with respect to P, a probability that we have no reason to doubt about since it was manufactured by us. P is not a descriptor of natural elements of the problem.

In the next section, all the points of the discussion so far are made more concrete through a non-trivial example.

4. An Example: Randomized Set-Membership Identification

The following example is based on the work reported in [39] and [40].

Consider the system

$$y_t = \theta u_t + n_t,$$

where u_t is input, θ is an unknown parameter, and n_t is noise; more general systems are considered in [39, 40].

We assume that we have no prior knowledge on θ, that is θ is a generic unknown real parameter; $\theta \in \mathbb{R}$. The noise n_t describes all other sources of variation in y_t besides u_t and we do not want to make any assumption on n_t, so that n_t can be any deterministic time signal. This reflects the fact that in many applications the noise characteristics are not known and, in any case, devising algorithms that work without noise assumptions makes the theory more generally applicable.

The system is accessible for experiments: we are asked to inject an input signal u_t, and, based on the observed output y_t, we have to provide an interval $[\theta_{min}, \theta_{max}]$ for θ. To make things as simple as possible, we assume in the following that we can access the system for 7 instants only, so that we inject $u_1, \ldots, u_7$ and collect $y_1, \ldots, y_7$. Thus, an algorithm is in two steps, where the first step is input selection (i.e. selection of $u_1, \ldots, u_7$), and the second step is a map from the input signal $u_1, \ldots, u_7$ and the collected output signal $y_1, \ldots, y_7$ into $[\theta_{min}, \theta_{max}]$.

Let us formalize the problem in the notational set-up of Section 2. We see the noise as a vector in $\mathbb{R}^7$: $n := [n_1, \ldots, n_7] \in \mathbb{R}^7$, and we want to find an algorithm that is successful in providing an interval $[\theta_{min}, \theta_{max}]$ such that $\theta \in [\theta_{min}, \theta_{max}]$ for any $\theta \in \mathbb{R}$ and for any $n \in \mathbb{R}^7$. Thus, a (θ, n) couple is analogous to a row in Table 1 of Example 1, and the algorithm is required to work for all $(\theta, n) \in \mathbb{R} \times \mathbb{R}^7$.[3]

At first glance, this problem looks unsolvable: since the noise can be whatever, it seems that the observed data are unable to give us a hand in constructing a confidence region for θ! It is a fact that *a successful deterministic algorithm does not exist*, as we can argue through a simple reasoning: consider any deterministic input string $[u_1, \ldots, u_7]$, say $[u_1, \ldots, u_7] = [2, \ldots, 2]$, and any map $M : ([u_1, \ldots, u_7], [y_1, \ldots, y_7]) \to [\theta_{min}, \theta_{max}]$. The input $[u_1, \ldots, u_7] = [2, \ldots, 2]$ together with the map M form the deterministic algorithm. Then, consider a specific output string, for instance $[y_1, \ldots, y_7] = [1, \ldots, 1]$, and construct the interval that corresponds in the map M to the input $[2, \ldots, 2]$ and the output $[1, \ldots, 1]$: $M([2, \ldots, 2], [1, \ldots, 1])$ and let $\bar{\theta}$ be any number that does not belong to $M([2, \ldots, 2], [1, \ldots, 1])$. Now, the system with this $\bar{\theta}$ as parameter and noise given by $\bar{n}_t = 1 - \bar{\theta} u_t = 1 - \bar{\theta} \cdot 2$ generates $[1, \ldots, 1]$ as output. If the algorithm defined by the input string $[u_1, \ldots, u_7] = [2, \ldots, 2]$ and the map M is applied to $(\bar{\theta}, \bar{n})$, it returns $[\theta_{min}, \theta_{max}]$, so that if $\bar{\theta}$ is the correct value the algorithm misclassifies this true $\bar{\theta}$ value and it is unsuccessful[4].

A natural question to ask is: is the above a "hopeless" problem, or can we solve it in some generalized sense?

Indeed, the problem is solvable within the framework of randomized algorithms, in the sense that we can provide an algorithm that returns an interval $[\theta_{min}, \theta_{max}]$ containing θ with a prescribed probability level whatever θ and the noise signal are.

A randomized algorithm can be constructed as follows.

4.1. A Randomized Algorithm

STEP 1: input selection

Let u_t, $t = 1, \ldots, 7$, be independent and identically distributed with distribution

$$u_t = \begin{cases} 1, & \text{with probability } 0.5 \\ -1, & \text{with probability } 0.5. \end{cases}$$

(Input randomization like the u_t introduced here has been used in other contributions, see e.g. [41]). Using a random input represents the random part of the algorithm. Here, each input string $[\pm 1, \ldots, \pm 1]$ has the same probability to occur, that is P gives probability $1/2^7$ to each one of the possible 2^7 strings.

STEP 2: map from $[u_1, \ldots, u_7]$, $[y_1, \ldots, y_7]$ *to* $[\theta_{min}, \theta_{max}]$

After collecting the 7 outputs $y_1, \ldots, y_7$, consider $y_t \cdot u_t - x$, $t = 1, \ldots, 7$, where x is a real variable; in other words, consider the -45 degrees lines that intersect the x-axis in $y_t \cdot u_t$. Next, we take the average of some of these lines in a few different ways. Precisely, we form 7 averages of the form:

$$\text{average}_i(x) = \frac{1}{4} \sum_{t \in \text{set}_i} [y_t \cdot u_t - x], \quad i = 1, \ldots, 7,$$

[3] Note that a successful algorithm in the described context is as well successful in a stochastic context, whatever the noise characteristics are, that is it is successful pathwise on all noise realizations. The noise can, for example, be Gaussian or non-Gaussian, stationary or non-stationary, etc.

[4] This is the very reason why in the (deterministic) set-membership identification literature priors are added to the problem to make it solvable.

where set_i contains the elements highlighted by a bullet in the table below. For instance: $\text{set}_1 = \{1,2,4,5\}$, $\text{set}_2 = \{1,3,4,6\}$, etc.

	1	2	3	4	5	6	7
set_1	•	•		•	•		
set_2	•		•	•		•	
set_3		•	•		•	•	
set_4	•	•				•	•
set_5	•		•		•		•
set_6		•	•	•			•
set_7				•	•	•	•

To explain things in a more concrete way, we made a simulation: a system with $\theta = 1$ and with noise n_t obtained as a realization of a biased independent Gaussian process with mean 0.5 and variance 1 was simulated and the following input/output strings were obtained:

u_t	-1	1	1	1	-1	1	-1
y_t	-0.45	0.97	1.74	1.80	0.1	1.92	-0.73

The corresponding functions $\text{average}_i(x)$, $i = 1,\ldots,7$, are displayed in Fig. 1.

Now, a simple reasoning suggests that these average lines have a tendency to intersect the x-axis near θ and that, for $x = \theta$, they take on positive or negative value with equal probability. Why is it so? Let us re-write one of these lines, say $\text{average}_1(x)$, as follows:

$$\begin{aligned}\text{average}_1(x) &= \frac{1}{4}\sum_{t\in\{1,2,4,5\}} [y_t \cdot u_t - x] \\ &= \frac{1}{4}\sum_{t\in\{1,2,4,5\}} [\theta u_t^2 + n_t \cdot u_t - x] \\ &= [\text{recall that } u_t^2 = 1] \\ &= [\theta - x] + \frac{1}{4}\sum_{t\in\{1,2,4,5\}} n_t \cdot u_t. \qquad (3)\end{aligned}$$

If $\frac{1}{4}\sum_{t\in\{1,2,4,5\}} n_t \cdot u_t = 0$, the intersection with the x-axis is exactly in $x = \theta$. The vertical "displacement" $\frac{1}{4}\sum_{t\in\{1,2,4,5\}} n_t \cdot u_t$ is a random variable with equal probability to be positive or negative (because u_t has this property), regardless of what n_t is; moreover, due to averaging, the vertical dispersion caused by noise is de-emphasized (it would be more de-emphasized with more data). So, we can expect that θ will be "somewhere" near where the average lines intersect the x-axis.

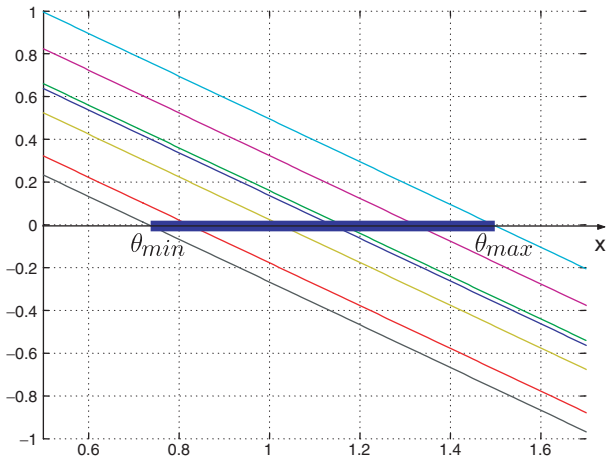

Fig. 1. The $\text{average}_i(x)$ functions together with $[\theta_{min}, \theta_{max}]$.

Following the above reasoning, we recognize that for $x = \theta$ it is unlikely that all the $\text{average}_i(x)$, $i = 1,\ldots,7$, functions have the same sign, and we therefore discard the rightmost and leftmost regions where all functions are less than zero or greater than zero (in Fig. 1 this leads to the interval $[0.73, 1.49]$). The resulting interval is the confidence region for θ.

END OF STEP 2

A rigorous reasoning (given in Appendix A for not breaking the continuity of discourse here) reveals that a fairly strong claim on the so-constructed interval can actually be made:

> **RESULT:** The confidence interval where not all the $\text{average}_i(x)$, $i = 1,\ldots,7$, functions have the same sign has exact probability 75% to contain the true parameter value θ.

Rephrasing the RESULT we can say that, given θ and n, the confidence interval is stochastic because it depends on the random choice of the input signal. The result states that in 75 input choices out of 100, the interval contains θ; moreover, this probability 75% is exact (not a lower bound) and the result holds true for any noise sequence n.

For the reader's convenience, the above discussion is now summarized as follows:

- there is a randomized algorithm that works this way: select an independent random input signal $u_t = \pm 1$ with probability 0.5 each (we thus have 2^7 random input strings, each one with probability $1/2^7$). This random selection represents the random rule in the algorithm. Inject the input sequence into the system, collect the outputs and construct a confidence interval as indicated above.
- For any θ and n (a (θ, n) couple is the equivalent to a row in Table 1 of Example 1), it holds that $\theta \in$ confidence

interval for 75% of the 2^7 choices of the input string (a choice of an input string is the equivalent in Example 1 to selecting a column in Table 1), that is the algorithm is successful with probability $p = 75\%$.

In this example, probability is rather low (75%) and the interval is normally rather large (it was [0.73, 1.49] in the simulation example) because, for the sake of simplicity, we assumed to have 7 data points only. With more data, we would have obtained smaller intervals with higher probability. For example, we run a simulation where 1023 data points were generated with the same characteristics as in the previous simulation example, and we obtained that the interval [0.945, 1.033] had probability 97.5% to contain θ. Thus, set-membership identification with a deterministic requirement of success is impossible without priors on (θ, n), and this severely limits the applicability of the theory. However, moving to a randomized algorithm reveals that the theory can be freed of the stiff deterministic assumptions and still satisfactory methods can be designed provided that one accepts a (small) risk of failure.

In conclusion, we see that

moving from deterministic to randomized algorithms has changed our perspective of "problem solvability," and this example shows that a "hopeless" problem from a deterministic perspective can still be solvable with a (possibly high) probability of success, provided that a randomized approach is adopted.

5. Randomized Algorithms as a Way to Reduce Computational Complexity

Many authors have advocated the use of randomized algorithms as a way to reduce the computational complexity with respect to more standard deterministic approaches. It is our perspective here that this use of randomized algorithms is just a particular case of the general set-up discussed in previous sections.

More precisely, the idea is as follows: suppose a deterministic algorithm requires excessive computations to process all the available information. Then, one can deliberately renounce part of the information and set out to solving a simplified problem with partial information. With partial information, however, deterministic solvability may be impossible to achieve, so that one turns to consider a randomized approach to determine a solution bearing a high probability of success according to the philosophical scheme described in previous sections. The net result is that full guarantee of success is traded for computational tractability.

A simple example better clarifies the aforementioned use of randomized algorithms.

5.1. An Example in Output Variance Evaluation

A nonlinear static system has 10 inputs $u_1, \ldots, u_{10}$ and one output y, and it is described by the relation

$$y = f(u_1, \ldots, u_{10}),$$

where f is some given known function. Suppose that $u_1, \ldots, u_{10}$ are stochastic and independent with uniform density in $[0, 1]$. We want to compute the second-order moment $E[y^2]$ of the output with a maximum approximation error of ϵ, where ϵ is a fixed given quantity.

Since vector $(u_1, \ldots, u_{10})$ has uniform density on $[0, 1]^{10}$, a deterministic algorithm to solve this problem simply consists in evaluating the integral $\int_{[0,1]^{10}} f(x_1, \ldots, x_{10})^2 dx_1, \ldots, dx_{10} = E[y^2]$. Computing such an integral can however be very difficult for an f that is not regular enough.

Let us consider now forming an estimate of $E[y^2]$ by only evaluating function $f(x_1, \ldots, x_{10})$ at a finite number N of points. In this way, we renounce considering a significant part of the information on f, but we are then dealing with N values only, a much handier information pattern than the whole f function. Using a Monte Carlo scheme, we pick N values $(x_1^{(i)}, \ldots, x_{10}^{(i)})$, $i = 1, \ldots, N$, at random with uniform distribution in $[0, 1]^{10}$, and form an estimate of $E[y^2]$ according to

$$\hat{E}[y^2] = \frac{1}{N} \sum_{i=1}^{N} f(x_1^{(i)}, \ldots, x_{10}^{(i)})^2. \tag{4}$$

Note that (4) is a random variable since it depends on the random extractions $(x_1^{(i)}, \ldots, x_{10}^{(i)})$. Its accuracy is quantified by Hoeffding's inequality, [42]:

HOEFFDING'S INEQUALITY:

Suppose that f is bounded, say for simplicity that its range is in $[0, 1]$. Then,

$$\left| \hat{E}[y^2] - E[y^2] \right| \leq \epsilon. \tag{5}$$

holds with probability $1 - \beta$, provided that

$$N \geq \frac{\ln 2/\beta}{2\epsilon^2}. \tag{6}$$

Note that probability $1 - \beta$ refers to extracting a multisample $(x_1^{(i)}, \ldots, x_{10}^{(i)})$ such that Eq. (5) holds true. Thus, β has to be interpreted as the probability that the randomized algorithm fails to provide an ϵ-accurate answer and the practical use of Eq. (6) is that it suggests the number of random extractions required so that the result is guaranteed at a desired level of probability $1 - \beta$. Since β shows up in Eq. (6) under the sign of logarithm, it can

be made very small without significantly increasing the number N of extractions. For example, with $\epsilon = 0.05$, selecting $\beta = 10^{-20}$ yields $N = 9350$.

To summarize, this very simple example has illustrated the following fact, a fact that is central in all randomized algorithms used to reduce computational complexity:

> **in certain problems, one can conceive a randomized algorithm that is easy to implement and which gives very high probabilistic guarantees of providing a satisfactory answer. This is obtained by renouncing part of the information. In the example at hand, this has consisted in concentrating on only N values of the function f.**

We further notice that, in the example at hand,

> **a deterministic algorithm that provides full guarantee of success by concentrating on N values does not exist. That is, for any deterministic choice of the N values, a function f can be found leading to an error bigger than ϵ.**

Therefore, accepting a small risk of being unsuccessful is strictly necessary.

Remark 2: *Following up what we have said at the beginning of this section, we remark that the frame of work of randomized algorithms in this Section 5 (computational complexity) can be traced back to the general set-up described in previous sections: in previous sections, we have argued that a randomized method can play a role to lift the problem solvability by decreasing the requirement for success when, due to lack of information, any deterministic algorithm fails. Here, the reason for resorting to a randomized method remains the same, and it is lack of information. However, what is different here is the reason for such a lack of information: while it is well possible that information was complete when the problem was assigned, we then deliberately discarded part of the information (to enhance computational tractability) and made it insufficient to deterministically solve the problem.* ∗

5.2. Moving Towards More Significant Problems

Even though the example of computing $E[y^2]$ was very simple, it served the purpose of clarifying the role of randomized algorithms in dealing with computational complexity issues. Within this same logic, impressive results have been obtained by randomized approaches in significant areas of control and systems theory. This is true for instance for robust control problems of convex nature, including the wide class of control problems representable by means of parameter-dependent Linear Matrix Inequalities (LMIs). As is well known, many of these problems are NP-hard in a deterministic set-up, see e.g. [43–47], but one can tackle these problems by resorting to a randomized approach, see e.g. [15]. Likewise, [13] has pioneered the randomized approach for average robust control. The reader is also referred to the very good and comprehensive treatment of the matter provided by reference [37].

6. An Additional Observation on the Use of Probabilistic Results

In this section, we want to highlight a further aspect that has contributed to the success of randomized algorithms, the fact that the performance and probability of success of randomized algorithms can be evaluated with extraordinarily powerful analytical methods.

We start by observing that in the output variance example of Section 5.1 we could as well have thought of pursuing a different route than randomized methods: if designing a deterministic algorithm using N values of f that works for all f functions is not possible, one can still go for a deterministic algorithm that works for most f functions, that is the algorithm is worst-case successful over a large subset of cases.

While the above is a logic alternative, it however comes with an attendant difficulty: finding the subset where the algorithm is successful is generally a formidable task. For instance, in the problem of computing $E[y^2]$ by referring to a deterministic grid of $(x_1^{(i)}, \ldots, x_{10}^{(i)})$ values, one can easily draw the conclusion that the method works well for f functions that are regular enough, but determining the real set of functions for which this method works (which is in actual effects way much larger than regular functions) can be extremely difficult.

A similar mechanism was noted by M. Gevers in a different context of system identification as early as in 1991. In [48], he wrote, *"... there are situations where it is known a priori that some noise is always smaller than some finite bound, but in most cases disturbances or measurement errors may occasionally be larger than they are on average. ... A reasonable approach would then be to replace the hard bounds on model error in the robustness criteria by confidence interval bounds corresponding to a sufficiently high probability."* Thus, while it is clear that the system identification procedure works deterministically well when the noise stays small, still this condition is stiff and deterministically studying all the conditions where a given procedure actually works is difficult. A better solution is to move to a stochastic approach.

Turning to randomization, we see that using a randomized approach gives us a formidable opportunity:

> **we can use amazingly powerful results from probability theory to obtain suitable quantitative assessments of success.**

This is, for example, the case with the use of Hoeffding's inequality in the previous example, leading to the conclusion that, for any f, the probability of being unsuccessful (i.e. $\epsilon > 0.05$) with $N = 9350$ extractions is below 10^{-20}! We then have to decide whether we prefer a probability of success of $1 - 10^{-20}$ with no restrictions on the f function, or a sure success under the hypothesis that the function is regular enough. While it is impossible to provide a universal answer, we tend to agree with R.V. Hogg, [49], that *"... we know in practice that most models will seldom fit exactly the real situations. Thus, for the sake of application, it seems ridiculous to try to get the last ounce of mathematical efficiency out of some assumed situation. A more realistic approach would be to seek statistical procedures good for a broad class of possible underlying models, but which are not necessarily best for any of them."*

Generalizing this observation and using again our Table 1 to visualize things, we can say that estimating the probabilistic proportion of whites in the rows (for which we can use probabilistic tools) is in general a more accessible task than determining the portion of whites in the columns. If it is true that this advantage is merely of practical nature, nevertheless it has a great importance to obtain quantitative assessments of success and, thereby, to make randomized algorithms gain wide-spread acceptance among users.

7. Drawing the Conclusions

This write-up has presented some observations on randomized methods. I hope the reflections here contained, though partial and limited, will stimulate further thinking and help gain a deeper understanding of what the potentials and limitations of randomization are.

One important observation is that a randomized algorithm is simply an algorithm where some choices are made at random, that is where fate is called on to choose for us. Since fate is not an expert of anything, the advantage of using a randomized algorithm cannot be found in the goodness of fate's choices, it more simply stays in our perception of problem solvability: resorting to a random choice permits to average the probability of failure of many deterministic choices and a black in a table like Table 1 is spread through the row resulting into a grey. In a worst-case perspective, a grey may be more acceptable than having a few blacks among many whites and so randomization makes the problem look more solvable.

Importantly, in the process of assessing the level of grey we can resort to probabilistic results whose power and effectiveness is at times truly surprising. This adds to our confidence in the use of randomized methods.

Acknowledgement

This work is supported by MIUR (Ministero dell'Istruzione, dell'Universita' e della Ricerca) under the project *New methods for Identification and Adaptive Control of Industrial Systems.* The author gratefully acknowledges insightful discussions with Prof. M. Gevers on various topics contained in this manuscript. The example in Section 4 is due to the author's work with Prof. E. Weyer to whom I am most grateful for many years of fruitful collaboration and friendship. I would finally like to thank Dr. S. Garatti for reading this manuscript and for providing many thoughtful comments.

References

1. Metropolis N, Ulam SM. The Monte Carlo method. *J. Am. Stat. Assoc.*, 1949; 44: 335–341.
2. Metropolis N, Rosenbluth A, Rosenbluth M, Teller A, Teller E. Equation of state calculations by fast computing machines. *J. Chem. Phys.*, 1953; 21: 1087–1092.
3. Hastings WK. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 1970; 57: 97–109.
4. Jerrum M, Sinclair A. The Markov Chain Monte Carlo method: An approach to approximate counting and integration. In DS Hochbaum (Ed.) *Approximation algorithms for NP-hard problems*, PWS Publishing, Boston, pp. 109–127, 1996.
5. Gilks WR, Richardson S, Spiegelhalter DJ. *Markov Chain Monte Carlo in practice*, Chapman and Hall, London, 1996.
6. Gentle JE. *Random number generation and Monte Carlo methods*, Springer-Verlag, New York, 1998.
7. Robert CP, Casella G. *Monte Carlo statistical methods*, Springer-Verlag, New York, 1999.
8. Vapnik VN, Chervonenkis AY. Uniform convergence of the frequencies of occurrence of events to their probabilities. Soviet Math. *Doklady*, 1968; 9: 915–918.
9. Vapnik VN, Chervonenkis AY. On the uniform convergence of relative frequencies to their probabilities. *Theory Probab. Appl.*, 1971; 16: 264–280.
10. Blumer A, Ehrenfeucht A, Haussler D, Warmuth M. Learnability and the Vapnik-Chervonenkis dimension. *J. ACM*, 1989; 36: 929–965.
11. Vidyasagar M. *A theory of learning and generalization*, Springer, New York, 1997.
12. Vapnik VN. *Statistical learning theory*, Wiley, New York, 1998.
13. Vidyasagar M. Randomized algorithms for robust controller synthesis using statistical learning theory. *Automatica*, 2001; 37: 1515–1528.
14. Calafiore G, Campi MC. Uncertain convex programs: Randomized solutions and confidence levels. *Math. Program.*, 2005; 102(1): 25–46.
15. Calafiore G, Campi MC. The scenario approach to robust control design. *IEEE Trans. Autom. Control*, 2006; 51: 742–753.
16. Campi MC, Garatti S. The exact feasibility of randomized solutions of uncertain convex programs. *SIAM J. Optim.*, 2008; 19(3): 1211–1230.

17. Stengel RF, Ray LR. Stochastic robustness of linear time-invariant control systems. *IEEE Trans. Autom. Control*, 1991; 36: 82–87.
18. Ray LR, Stengel RF. A Monte Carlo approach to the analysis of control system robustness. *Automatica*, 1993; 29: 229–236.
19. Khargonekar PP, Tikku A. Randomized algorithms for robust control analysis have polynomial time complexity. In *Proceedings of the Conference on Decision and Control*, pp. 3470–3475, 1996.
20. Tempo R, Bai EW, Dabbene F. Probabilistic robustness analysis: Explicit bounds for the minimum number of samples. *Syst. Control Lett.*, 1997; 30: 237–242.
21. Barmish BR, Lagoa CM. The uniform distribution: A rigorous justification for its use in robustness analysis. *Math. Control, Signals, Syst.*, 1997; 10: 203–222.
22. Bai EW, Tempo R, Fu M. Worst case properties of the uniform distribution and randomized algorithms for robustness analysis. *Math. Control, Signals, Syst.*, 1998; 11: 183–196.
23. Calafiore G, Dabbene F, Tempo R. Randomized algorithms for probabilistic robustness with real and complex structured uncertainty. *IEEE Trans. Autom. Control*, 2000; 45: 2218–2235.
24. Oishi Y, Kimura H. Randomized algorithms to solve parameter-dependent linear matrix inequalities and their computational complexity. In *Proceedings of the Conference on Decision and Control, Orlando, FL*, pp. 2025–2030, 2001.
25. Calafiore G, Polyak BT. Stochastic algorithms for exact and approximate feasibility of robust LMIs. *IEEE Trans. Autom. Control*, 2001; 46: 1755–1759.
26. Fujisaki Y, Dabbene F, Tempo R. Probabilistic robust design of LPV control systems. *Automatica*, 2001; 39: 1323–1337.
27. Polyak BT, Tempo R. Probabilistic robust design with linear quadratic regulators. *Syst. Control Lett.*, 2001; 43: 343–353.
28. Kanev S, De Schutter B, Verhaegen M. The ellipsoid algorithm for probabilistic robust controller design. In *Proceedings of the Conference on Decision and Control*, Las Vegas, NV, pp. 2248–2253, 2002.
29. Ishii H, Basar T, Tempo R. Randomized algorithms for quadratic stability of quantized sampled-data systems. *Automatica*, 2004; 40: 839–846.
30. Doucet A, Logothetis A, Krishnamurthy V. Stochastic sampling algorithms for state estimation in jump Markov linear systems. *IEEE Trans. Autom. Control*, 2000; 45: 188–202.
31. Koltchinskii V, Abdallah CT, Ariola M, Dorato P, Panchenko D. Improved sample complexity estimates for statistical learning control of uncertain systems. *IEEE Trans. Autom. Control*, 2000; 46: 2383–2387.
32. Oishi Y. Probabilistic design of a robust state-feedback controller based on parameter-dependent Lyapunov functions. In *Proceedings of the Conference on Decision and Control*, Maui, HI, pp. 1920–1925, 2003.
33. Oishi Y, Kimura H. Computational complexity of randomized algorithms for solving parameter-dependent linear matrix inequalities. *Automatica*, 2003; 39: 2149–2156.
34. Campi MC, Prandini M. Randomized algorithms for the synthesis of cautious adaptive controllers. *Syst. Control Lett.*, 2003; 49: 21–36.
35. Smith RL. Efficient Monte-Carlo procedures for generating points uniformly distributed over bounded regions. *Oper. Res.*, 1984; 32: 1296–1308.
36. Calafiore G, Dabbene F. A probabilistic framework for problems with real structured uncertainty in systems and control. *Automatica*, 2002; 38: 1265–1276.
37. Tempo R, Calafiore G, Dabbene F. *Randomized algorithms for analysis and control of uncertain systems*, Springer-Verlag, New York, 2005.
38. Yao A. Probabilistic computations: toward a unified measure of complexity. In *Proceedings of the 18th Symposium on Foundations of Computer Science—FOCS*, Providence, RI, pp. 222–227, 1977.
39. Campi MC, Weyer E. Guaranteed non-asymptotic confidence regions in system identification. *Automatica*, 2005; 41: 1751–1764.
40. Campi MC, Weyer E. Identification with finitely many data points: The LSCR approach. In *Proceedings of Symposium on System Identification—SYSID*, semi-plenary presentation, Newcastle, Australia, pp. 46–64, 2006.
41. Goldenshluger A, Polyak B. Estimation of regression parameters under arbitrary noise, *Math. Methods Stat.*, 1993; 2: 18–29.
42. Hoeffding W. Probability inequalities for sums of bounded random variables. *J. Am. Stat. Assoc.*, 1963; 58: 13–30.
43. Nemirovski A. Several NP-hard problems arising in robust stability analysis. *J. Matrix Anal. Appl.*, 1993; 6: 99–105.
44. Poljak S, Rohn J. Checking robust nonsingularity is NP-hard. *Math. Control, Signals, Syst.*, 1993; 6: 1–9.
45. Becker G, Packard A. Robust performance of linear parametrically varying systems using parametrically-dependent linear feedback. *Syst. Control Lett.*, 1994; 23: 205–215.
46. Braatz RP, Young PM, Doyle JC, Morari M. Computational complexity of μ calculation. *IEEE Trans. Autom. Control*, 1994; 39: 1000–1002.
47. Blondel VD, Tsitsiklis JN. A survey of computational complexity results in systems and control. *Automatica*, 2000; 36: 1249–1274.
48. Gevers M. Connecting identification and robust control: a new challenge. In *Proceedings of IFAC Symposium on Identification and System Parameter Estimation*, plenary presentation, Budapest, Hungary, pp. 1–10, 1995.
49. Hogg RV. Adaptive robust procedures: A partial review and some suggestions for future applications and theory. *J. Am. Stat. Assoc.*, 1974; 69: 909–923.

A. Proof of the Result in Section 4

To simplify the notation, also introduce $\text{average}_8(x) = 0$, the function identically equal to zero.

The true parameter value θ falls outside the confidence interval when – corresponding to $x = \theta$ – all functions $\text{average}_i(x)$, $i = 1, \ldots, 7$, are bigger than $\text{average}_8(x) = 0$, or all functions are smaller than $\text{average}_8(x)$. We claim that each one of these two events has probability $1/8$ to happen, so that the probability that θ falls in the confidence interval is $1 - 2 \cdot 1/8 = 0.75$, as claimed in the RESULT.

We next concentrate on computing the probability of the event where "all functions $\text{average}_i(\theta)$, $i = 1, \ldots, 7$, are bigger than $\text{average}_8(\theta) = 0$." The probability of the other event is derived similarly.

Using (3), the event under consideration is determined by conditions

$$\begin{cases} u_1n_1 + u_2n_2 + u_4n_4 + u_5n_5 > 0 \\ u_1n_1 + u_3n_3 + u_4n_4 + u_6n_6 > 0 \\ u_2n_2 + u_3n_3 + u_5n_5 + u_6n_6 > 0 \\ u_1n_1 + u_2n_2 + u_6n_6 + u_7n_7 > 0 \\ u_1n_1 + u_3n_3 + u_5n_5 + u_7n_7 > 0 \\ u_2n_2 + u_3n_3 + u_4n_4 + u_7n_7 > 0 \\ u_4n_4 + u_5n_5 + u_6n_6 + u_7n_7 > 0. \end{cases} \tag{7}$$

To compute the probability that all these 7 inequalities are simultaneously true let us ask the following question: what should we write if instead of comparing $\text{average}_i(\theta)$, $i = 1, \ldots, 7$, with $\text{average}_8(\theta)$ we would compare $\text{average}_i(\theta)$, $i = 2, \ldots, 8$, with $\text{average}_1(\theta)$? The conditions would in this case be:

$$\begin{cases} u_1n_1 + u_3n_3 + u_4n_4 + u_6n_6 \\ \quad > u_1n_1 + u_2n_2 + u_4n_4 + u_5n_5 \\ u_2n_2 + u_3n_3 + u_5n_5 + u_6n_6 \\ \quad > u_1n_1 + u_2n_2 + u_4n_4 + u_5n_5 \\ u_1n_1 + u_2n_2 + u_6n_6 + u_7n_7 \\ \quad > u_1n_1 + u_2n_2 + u_4n_4 + u_5n_5 \\ u_1n_1 + u_3n_3 + u_5n_5 + u_7n_7 \\ \quad > u_1n_1 + u_2n_2 + u_4n_4 + u_5n_5 \\ u_2n_2 + u_3n_3 + u_4n_4 + u_7n_7 \\ \quad > u_1n_1 + u_2n_2 + u_4n_4 + u_5n_5 \\ u_4n_4 + u_5n_5 + u_6n_6 + u_7n_7 \\ \quad > u_1n_1 + u_2n_2 + u_4n_4 + u_5n_5 \\ 0 > u_1n_1 + u_2n_2 + u_4n_4 + u_5n_5, \end{cases}$$

or, moving everything to the left-hand-side,

$$\begin{cases} -u_2n_2 + u_3n_3 - u_5n_5 + u_6n_6 > 0 \\ -u_1n_1 + u_3n_3 - u_4n_4 + u_6n_6 > 0 \\ -u_4n_4 - u_5n_5 + u_6n_6 + u_7n_7 > 0 \\ -u_2n_2 + u_3n_3 - u_4n_4 + u_7n_7 > 0 \\ -u_1n_1 + u_3n_3 - u_5n_5 + u_7n_7 > 0 \\ -u_1n_1 - u_2n_2 + u_6n_6 + u_7n_7 > 0 \\ -u_1n_1 - u_2n_2 - u_4n_4 - u_5n_5 > 0. \end{cases}$$

If we now define $\tilde{u}_1 = -u_1$, $\tilde{u}_2 = -u_2$, $\tilde{u}_4 = -u_4$, $\tilde{u}_5 = -u_5$, the latter condition re-writes as

$$\begin{cases} \tilde{u}_2n_2 + u_3n_3 + \tilde{u}_5n_5 + u_6n_6 > 0 \\ \tilde{u}_1n_1 + u_3n_3 + \tilde{u}_4n_4 + u_6n_6 > 0 \\ \tilde{u}_4n_4 + \tilde{u}_5n_5 + u_6n_6 + u_7n_7 > 0 \\ \tilde{u}_2n_2 + u_3n_3 + \tilde{u}_4n_4 + u_7n_7 > 0 \\ \tilde{u}_1n_1 + u_3n_3 + \tilde{u}_5n_5 + u_7n_7 > 0 \\ \tilde{u}_1n_1 + \tilde{u}_2n_2 + u_6n_6 + u_7n_7 > 0 \\ \tilde{u}_1n_1 + \tilde{u}_2n_2 + \tilde{u}_4n_4 + \tilde{u}_5n_5 > 0. \end{cases} \tag{8}$$

Except for the "˜" showing up here and there, this latter set of inequalities is the same as the original set of inequalities in Eq. (7) (the order in which the inequalities are listed is different but the inequalities altogether are the same). Moreover, since the u_t variables are symmetrically distributed, the change of sign implied by the '˜' is immaterial as far as the probability of satisfaction of the inequalities in Eq. (8) is concerned, so that we conclude that (7) and Eq. (8) are satisfied with the same probability.

Now, instead of comparing $\text{average}_i(\theta)$, $i = 2, \ldots, 8$, with $\text{average}_1(\theta)$, we could have compared $\text{average}_i(\theta)$, $i = 1, 3, 4, 5, 6, 7, 8$, with $\text{average}_2(\theta)$, or $\text{average}_i(\theta)$, $i = 1, 2, 4, 5, 6, 7, 8$, with $\text{average}_3(\theta)$ or ... arriving all the time to a similar conclusion that the probability does not change. Since these 8 events ($\text{average}_1(\theta)$ is the smallest, $\text{average}_2(\theta)$ is the smallest, etc.) are disjoint and cover all possibilities and all of them have the same probability, we finally draw the conclusion that each and every event has exactly probability $1/8$ to happen. It remains therefore proven that the initial condition (7) is satisfied with probability $1/8$ and this concludes the proof.