Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.18576/amis/120416

Plurality Voting and the Computation of the Average Duration of Frames of Parallel Mutual Exclusion Accesses

Sourav Dutta[1] *and Dimitri Kagaris*[2,*]

[1] School of Theoretical and Applied Science, Ramapo College of New Jersey, Mahwah, NJ 07430, USA
[2] Electrical and Computer Engineering Department, Southern Illinois University, Carbondale, IL 62901, USA

Received: 21 May 2018, Revised: 20 Jun. 2018, Accepted: 25 Jun. 2018
Published online: 1 Jul. 2018

Abstract: In this paper we first show how to compute the probability that a particular candidate c out of m candidates, $1 \leq c \leq m$, wins by plurality voting an election conducted by n voters, where each voter either votes for a single candidate i, $1 \leq i \leq m$, with probability P_i, or abstains with probability $P_0 = 1 - \sum_{i=1}^{m} P_i$. We then show how this result is involved in the computation of the average execution time of a set ("frame") of n simultaneous requests, where each request randomly asks for exclusive access to any of m available non-shareable resources with probability $P_i, 1 \leq i \leq m$, or for non-exclusive access to a common fully shareable resource with probability $P_0 = 1 - \sum_{i=1}^{m} P_i$. We also allow that each resource access has a different duration $D_i, 0 \leq i \leq m$. The formulas that we develop have application in the analysis and evaluation of ensemble classifiers in pattern recognition and classification, and in systems performance evaluation (critical sections in multithreaded programs with barrier synchronization, switch delay in computer networks and interconnection networks).

Keywords: Ensemble classifiers; Plurality voting; Performance evaluation; Mutual exclusion; Critical sections; Multi-threaded programs; Barrier synchronization; Network switching.

1 Introduction

We consider an election with m candidates and n voters. Each voter is allowed to vote only for a single candidate i, $1 \leq i \leq m$, or abstain. Each voter is assumed to vote for each candidate i with a known probability P_i (which is the same among all voters), or abstain with probability $P_0 = 1 - \sum_{i=1}^{m} P_i$. We want to compute the probability that a particular candidate c, $1 \leq c \leq m$, wins the election by plurality (see, e.g., [14], i.e., by receiving the maximum number of votes over all other candidates, without necessarily gaining the absolute majority. This problem finds extensive application in ensemble classification in pattern recognition (see, e.g., [6,11,9,8,7,13,5]) where the "voters" are the classifiers and the "candidates" are the classes that a target object should be classified in, with the idea being that the class that receives the most votes by the classifiers is arguably the "correct" class that the object belongs to. In this paper, we develop a formula to compute the winning probability under the above plurality voting scenario and extend it also to account for ties, i.e., for more than one candidate receiving the same maximum number of votes.

We then show how this formula relates to another problem, that of the computation of the average execution time of frames of parallel mutual exclusion accesses. In the latter problem, we consider a set ("frame") of n simultaneous requests where each request asks randomly for exclusive access to any of m available non-shareable ("critical") resources $R_i, 1 \leq i \leq m$, with probability $P_i, 1 \leq i \leq m$, or for non-exclusive access to a common fully shareable ("non-critical") resource R_0 with probability $P_0 = 1 - \sum_{i=1}^{m} P_i$. Any subset of these requests can be executed in parallel, provided that they all access the non-critical resource R_0 or they all access different critical resources $R_i, 1 \leq i \leq m$. Otherwise, the execution of any requests that access the same resource R_j for some j, $1 \leq j \leq m$, has to be serialized. We assume that each access to resource $i, 1 \leq i \leq m$, takes time D_i, $0 \leq i \leq m$, to be serviced. The execution time ("frame duration") T of the original set of requests is the maximum time that

* Corresponding author e-mail: kagaris@engr.siu.edu

any of the m resources has to take in order to service its own requests. More specifically, if ρ_i denotes the total number of requests in the frame for resource $R_i, 1 \leq i \leq m$, then the total execution time or frame duration T is given by:

$$T = \max_{1 \leq i \leq m}(D_0, \rho_i \cdot D_i).$$

We want to compute the formula for the *average* value of T given that each of the n requests in the frame asks for resource R_i with probability $P_i, 0 \leq i \leq m$. This formula finds applications in systems performance evaluation. For example, in computer architecture (see, e.g., [3,4,10,15]), the set of the n requests corresponds to n "threads" of the same multithreaded program that are executed in parallel by n separate processors or "cores." The m resources correspond to "critical sections" that are protected by mutual exclusion "locks." The shareable resource R_0 corresponds to non-critical code executed by the threads. The "frame" corresponds to threads executed under "barrier synchronization." As another example, in computer networks ((see, e.g., [12]) and interconnection networks (see, e.g., [2]), the frame corresponds to a set of n (possibly empty) packets coming into a switch at the same time to be routed to the m outgoing channels of the switch. Two or more incoming packets may request the same outgoing channel ("critical resource") and in that case they have to be serialized. If a packet is empty, this corresponds to a request for the shareable resource R_0.

In the above frame formulation, we observe that each critical resource in $R_i, 1 \leq i \leq m$, can be considered as a "candidate" and that each of the n requests in the frame can be considered as a "voter," which "votes" for the resource if it wants to access it (with probability P_i, $1 \leq i \leq m$). The shareable resource R_0 corresponds to the abstention option in the voting scenario. Assuming all durations D_i, $0 \leq i \leq m$, are equal to 1, the probability that a candidate (critical resource) or set of candidates win the "election" with j votes, $1 \leq j \leq n$, relates to the probability that the duration of the frame is equal to j (the exact relationship between the two probabilities, as well as the effect of different durations D_i, $0 \leq i \leq m$, is shown in later sections).

An example of the assumed setup for the resource requests in a frame is shown in Fig. 1. In this example we have a frame of 9 requests (Fig. 1(a)), that ask access to any of 3 available resources A, B, C. The absence of request for any of A, B, C is indicated by ε. Assuming $D_A = D_B = D_C = D_\varepsilon = 1$, the execution time T is $T = 3$ (Fig. 1(b)) due to the serialization on A. Assuming $D_A = 1, D_B = 2, D_C = 3, D_\varepsilon = 1$, the execution time T is $T = 4$ (Fig. 1(c)) due to the serialization on B (the number of requests for B is less than the number of requests for A, but $D_B > D_A$). Assuming $D_A = 1, D_B = 2, D_C = 3, D_\varepsilon = 5$, the execution time T will be $T = 5$ (Fig. 1(b)) due to the domination of D_ε.

The rest of the paper is organized as follows: We first develop the formula (Section 2) for computing the probability for a given candidate (or subset of candidates in case of a tie) to win the plurality voting. Then we show (Section 3) how this formula relates to the computation of the average frame execution time T in the case where all durations $D_i, 0 \leq i \leq m$ are equal to 1 (probabilities $P_i, 1 \leq i \leq m$ are arbitrary, with $P_0 = 1 - \sum_{i=1}^{m} P_i$). Then we extend the latter formula (Section 4) to account for arbitrary critical durations $D_i, 1 \leq i \leq m$, provided that the non-critical duration D_0 is equal to 1. Then we show (Section 5) the modification needed to account for arbitrary (critical or non-critical) durations $D_i, 0 \leq i \leq m$. An experimental evaluation of the formula in terms of accuracy and time complexity is shown in Section 6 and we conclude in Section 7.

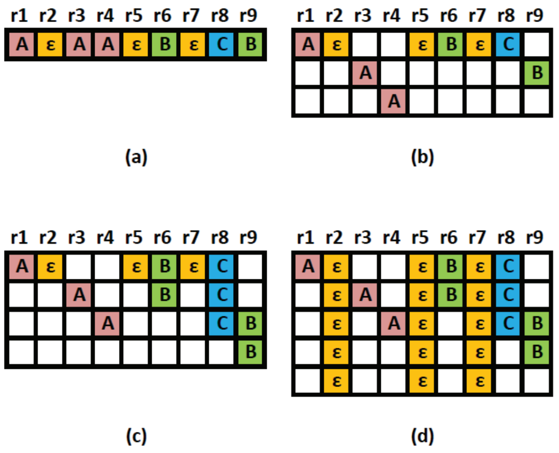

Fig. 1: Execution of a frame with critical resources A, B and C: (a) Request frame; (b) Execution assuming $D_A = D_B = D_C = D_\varepsilon = 1$; (c) Execution assuming $D_A = 1, D_B = 2, D_C = 3, D_\varepsilon = 1$; (d) Execution assuming $D_A = 1, D_B = 2, D_C = 3, D_\varepsilon = 5$.

2 Computation of the winning probability for plurality voting

In this section we deal with the problem of computing the probability that a specific candidate c, $1 \leq c \leq m$, wins the election by plurality voting as described in the Introduction. The winner candidate c has to receive a number of votes j such that every other candidate $i \neq c$, $1 \leq i \leq m$, receives a number of votes strictly less than j. In computing this probability, we actually consider first that ties among winners are allowed, i.e., we compute the probability that a given subset s of candidates from the set $C = \{1, 2, \ldots, m\}$ tie as winners of the election (in an $|s|$-way tie), with all other candidates receiving fewer votes than any candidate in s. Then the probability that the specific candidate c, $1 \leq c \leq m$, is the sole winner of the election is simply going to be computed from the latter formula by setting $s = \{c\}$.

Consider a (nonempty) subset $s \subseteq C$ of candidates that end up as winners in an $|s|$-way tie by receiving the same maximum number j of votes each. Since there are n voters and each voter can vote for at most one candidate, the product $|s| \cdot j$ cannot be greater than n. Given a subset $s \subseteq C$, the probability that j voters vote for the ith candidate in s is $(P_{s_i})^j$ where P_{s_i} is the probability of voting for the ith candidate in s. The first candidate in subset s can receive j votes in $\binom{n}{j}$ ways, the second candidate in s can receive j votes from the remaining $n-j$ voters in $\binom{n-j}{j}$ ways and the ith candidate in s can receive j votes from the remaining $n-(i-1)j$ voters in $\binom{n-(i-1)j}{j}$ ways. Hence, the probability $P_D(s,j)$ that a set of candidates s tie as winners in an $|s|$-way tie with each candidate in s having received j votes and any other candidate in $C-s$ having received less than j votes can be written as:

$$\begin{aligned} P_D(s,j) &= (\prod_{i=1}^{|s|}\binom{n-(i-1)j}{j}\{P_{s_i}\}^j)\cdot P_N(\bar{s},j) = \\ &(\frac{n!}{j!(n-j)!}\frac{(n-j)!}{j!(n-2\cdot j)!}\cdots\frac{(n-(|s|-1)j)!}{j!(n-|s|\cdot j)!} \\ &\cdot\prod_{i=1}^{|s|}\{P_{s_i}\}^j)\cdot P_N(\bar{s},j) \\ &= \frac{n!}{(j!)^{|s|}(n-|s|\cdot j)!}\prod_{i=1}^{|s|}\{P_{s_i}\}^j\cdot P_N(\bar{s},j) \end{aligned} \quad (1)$$

where $P_N(\bar{s},j)$ is the probability of having at most $j-1$ votes for any candidate taken from the set $\bar{s} = C-s$, i.e., the probability that at most $j-1$ of any remaining voters vote for the same candidate from set $\bar{s}$, or abstain.

To compute probability $P_N(\bar{s},j)$, we proceed as follows: the total number of remaining voters is $V = n-|s|\cdot j$. The maximum number of voters that can vote for a candidate in set $\bar{s}$ (non-winning candidate) is $min(V, j-1)$. Let i_k, $k = 1,2,...|\bar{s}|$, be the number of votes that the kth candidate in $\bar{s}$ receives. Since the kth non-winning candidate can be voted by at most $min(V, j-1)$ voters, i_k can be at most $min(V, j-1)$. The probability of the kth non-winning candidate receiving i_k votes is $\{P_{\bar{s}_k}\}^{i_k}$. The number of voters that are left to vote for the kth non-winning candidate in $\bar{s}$ is $V-\sum_{r=1}^{k-1} i_r$. So, i_k voters can vote for the k-th non-winning candidate in $\binom{V-\sum_{r=1}^{k-1} i_r}{i_k}$ ways and each way has a probability of $\{P_{\bar{s}_k}\}^{i_k}$ to occur. Therefore, the the probability of i_k voters voting for the kth non-winning candidate is:

$$\sum_{i_k=0}^{min(V,j-1)}\binom{V-\sum_{r=1}^{k-1} i_r}{i_k}\cdot\{P_{\bar{s}_k}\}^{i_k}.$$

To account for abstention, the probability that $V-\sum_{k=1}^{|\bar{s}|} i_k$ voters abstain is $P_0^{V-\sum_{k=1}^{|\bar{s}|} i_k}$. Therefore, probability $P_N(\bar{s},j)$ can be written as follows:

$$\begin{aligned} P_N(\bar{s},j) = &\sum_{i_1=0}^{min(V,j-1)}\binom{V}{i_1}\{P_{\bar{s}_1}\}^{i_1}\sum_{i_2=0}^{min(V,j-1)}\binom{V-i_1}{i_2}\{P_{\bar{s}_2}\}^{i_2} \\ &\cdots\sum_{i_{|\bar{s}|}=0}^{min(V,j-1)}\binom{V-\sum_{k=1}^{|\bar{s}|-1} i_k}{i_{|\bar{s}|}}\{P_{\bar{s}_{|\bar{s}|}}\}^{i_{|\bar{s}|}} \\ &\cdot\{P_0\}^{V-\sum_{k=1}^{|\bar{s}|} i_k} \end{aligned}$$

We can rewrite the above expression as:

$$\begin{aligned} P_N(\bar{s},j) = &\sum_{\substack{0\le i_k\le min(V,j-1)\\ i_1+i_2+...+i_{|\bar{s}|}\le V}}((\binom{V}{i_1}\binom{V-i_1}{i_2} \\ &\cdots\binom{V-\sum_{k=1}^{|\bar{s}|-1} i_k}{i_{|\bar{s}|}}) \\ &\cdot\prod_{k=1}^{|\bar{s}|}\{P_{\bar{s}_k}\}^{i_k}\cdot\{P_0\}^{V-\sum_{k=1}^{|\bar{s}|} i_k}) \end{aligned} \quad (2)$$

The summation above is taken over all integer partitions (see, e.g., [1]) of the number V into $|\bar{s}|$ parts where no part is larger than $min(V, j-1)$. After simplification we have:

$$\begin{aligned} P_N(\bar{s},j) = &\sum_{\substack{0\le i_k\le< min(V,j-1)\\ i_1+i_2+...+i_{|\bar{s}|}\le V}}(\frac{V!}{\prod_{k=1}^{|\bar{s}|} i_k!(V-\sum_{k=1}^{|\bar{s}|} i_k)!} \\ &\cdot\prod_{k=1}^{|\bar{s}|}\{P_k\}^{i_k}\cdot\{P_0\}^{V-\sum_{k=1}^{|\bar{s}|} i_k}) \end{aligned} \quad (3)$$

Using function $f(X,K) = \frac{X^K}{K!}$, Equation (3) can be written as follows:

$$P_N(\bar{s},j) = \sum_{\substack{0\le i_k\le< min(V,j-1)\\ i_1+i_2+...+i_{|\bar{s}|}\le V}}(V!f(P_0, V-\sum_{k=1}^{|\bar{s}|} i_k)\prod_{k=1}^{|\bar{s}|} f(P_{s_i}, i_k)) \quad (4)$$

The probability then $P_{\text{TW}}(s)$ that a given nonempty subset s of candidates from the set $C = \{1,2,...,m\}$ tie as winners of the election, with all other candidates receiving fewer votes than any candidate in s is given by

$$P_{\text{TW}}(s) = \sum_{j=1}^{\lfloor\frac{n}{|s|}\rfloor} P_D(s,j) \quad (5)$$

Then the probability $P_{\rm W}(c)$ that a specific candidate $c \in C$ is the sole winner of the election is computed from the latter formula by setting $s = \{c\}$, i.e.,

$$P_{\rm W}(c) = P_{\rm TW}(\{c\}) = \sum_{j=1}^{n} P_D(\{c\}, j) \qquad (6)$$

2.1 Computing probability $P_{\rm W}(c)$ by a recursive procedure

The complexity of the formula for probability $P_{\rm W}(c)$ as computed by Eqs. (6), (1), and (4) is determined by the number of subsets of C and the number of integer partitions of each number V (we exclude from this complexity account the actual cost of the multiplications needed for finding factorials, exponentiations, etc). A generally faster way to compute the *same* value for $P_{\rm W}(c)$ is by means of a recursive procedure DOM($s,\ d,\ v,\ t$), which is described below.

Given a subset $s \subseteq C$ of candidates, a number of received votes d, another subset $v \subseteq C$ with $v \cap s = \varnothing$, and a number of voters t, procedure DOM($s,\ d,\ v,\ t$) computes recursively the probability that each candidate in s receives exactly $d \le t/|s|$ votes and that each candidate in v receives at most $d-1$ votes. A pseudocode of procedure DOM($s,\ d,\ v,\ t$) is given below:

```
**Procedure** DOM($s,\ d,\ v,\ t$)
L1: IF (($s == \varnothing$) OR ($t < 0$) OR ($d \cdot |s| > t$))
        THEN return(0);
L2: Initialize $p = 0$ ;
L3: Set $a = \frac{t!}{(t-|\sigma|\cdot d)!} \prod_{i=1}^{|\sigma|} f(P_{\sigma_i}, d)$;
L4: Initialize $b = (P_0)^{t-|s|\cdot d}$ ;
L5: FOR each value of $\hat{d}$ from $d-1$ down to 1 DO {
L6:   FOR each non-empty subset $\sigma$ of $v$ DO {
L7:     Update $b = b + {\rm DOM}(\sigma,\ \hat{d},\ v-\sigma,\ t-|s|\cdot\hat{d})$ ;
L8:   }
L9: }
L10: Update $p = p + a \cdot b$ ;
L11: Return($p$);
**End Procedure**
```

Line L3 of DOM() computes the first part of Eq. (1) and lines L4-L10 compute recursively (based on every non-empty subset σ of v) the part corresponding to $P_N(\bar{s}, j)$ and the total probability $P_{\rm TW}(s)$ (Eq. (5)).

The initial call to procedure DOM() and the actual computation of $P_{\rm TW}(s)$ is done by a driver procedure TIEWIN($s,\ n,\ C$) where n is the original number of voters, and C is the original set of the candidates. The pseudocode of procedure TIEWIN() is given below:

```
**Procedure** TIEWIN($s,\ n,\ C$)
L1: Initialize $p = 0$;
L2: FOR each value of $j$ from 1 to $n$ DO {
L3:   Update $p = p + {\rm DOM}(s,\ j,\ C-s,\ n)$ ;
L4: }
L5: Return($p$);
**End Procedure**
```

Procedure TIEWIN() computes exactly the same value of probability $P_{\rm TW}(s)$ as the formula in Eq. (5). Its complexity is the product of all numbers of subsets that are encountered during the recursive calls. Actual complexity counts and comparison with the complexity of Eq. (5) are reported in the experimental results section.

3 Computation of average duration T of frames of parallel mutual exclusion accesses

In this section we deal with the problem of computing the average duration T of frames of parallel mutual exclusion accesses, as described in the Introduction. We show how the result computed in the previous section is involved in the computation of T. We start the computation of the average execution time T by assuming that all durations $D_i, 1 \le i \le m$ (whether critical or non-critical) are the same (assumed to be 1).

We define the *dominance degree* j of a request frame to be the maximum number of requests for the same critical resource in the frame. A frame with dominance degree j takes exactly j time units to execute. A critical resource that has been requested j times in the frame is referred to as a *dominant* critical resource. In general, there may be more than one dominant critical resources, i.e., two or more critical resources may have the same number j of occurrences in the frame. Therefore, in general, we consider that a subset $s \subseteq C$ of critical resources are dominant in the frame.

Each critical resource in C can be considered as a "candidate" and that each of the requests in the frame can be considered as a "voter," which "votes" for the resource if it wants to access it. The critical resources that are dominant in the frame correspond to the subset s of candidates that tie as winners of the "election." The dominant degree of the frame corresponds to the number of votes that each candidate in s has received. Therefore the probability $P_{\rm DOM}(j)$ that a frame has dominance degree $j > 0$ and therefore experiences a duration of j time units is equal to the probability given by procedure DOM($s, j, C-s, n$) after this probability has been summed over all nonempty subsets s of C. Namely,

$$P_{\rm DOM}(j) = \sum_{s \in C, s \neq \varnothing} {\rm DOM}(s,\ j,\ C-s,\ n) \qquad (7)$$

The average frame duration T of a request frame is then given by:

$$T = \sum_{j=1}^{n} P_{\rm DOM}(j) \cdot j + (P_0)^n \qquad (8)$$

The last additive term is due to the fact that if no request in the frame asks for a critical resource (equivalently, the frame has dominance degree $j = 0$), the duration of the frame is still 1 time unit.

A procedure TD() for the computation of T following equations (8) and (7) and using procedure DOM() as a subroutine is given below:

```
**Procedure** TD $(n, C)$
L1: Initialize $T = (P_0)^n$ ;
L2: FOR each value of $j$ from 1 to $n$ DO {
L3:   FOR each non-empty subset $\sigma$ of $C$ DO {
L4:     Update $T = T + j \cdot \mathrm{DOM}(\sigma,\ j,\ C - \sigma,\ n)$ ;
L5:   }
L6: }
L7: Return($T$);
**End Procedure**
```

4 Computation of average T under arbitrary critical durations when non-critical duration is 1.

In this section, we extend the computation of the average for the case where the durations of the service requests by the resources are not equal to the unit time, but can take arbitrary values $D_i, 1 \le i \le m$. We show the modifications needed in procedures DOM() and TD(). We start by assuming that the duration of the non-critical access is equal to 1 (which is the smallest possible duration), i.e., $D_0 = 1$, so that the non-critical access never dominates the duration time by “hiding” the contribution of the critical resource durations. Later in Section 4, we eliminate that restriction too.

The primary changes needed in procedures DOM() and TD() are the following: (a) the dominance degree j is not anymore an integer in the range of 1 to n but is now a multiple of some value $D_i, 1 \le i \le m$ (we keep the term “degree” for simplicity, although this is now weighted by the duration values); and (b) the recursion is not done simply on subsets of C (the set of the indices j, $1 \le j \le m$, of the critical resources) but on subsets of C that contain resources whose durations divide the target dominance degree that is considered each time.

The modified driver procedure TD(), called now TDU(), is given below:

```
**Procedure** TDU $(n, C)$
L1: Initialize $T = (P_0)^n$ ;
L2: Initialize set $U$ to be the set of all distinct
          critical duration values $D_i$, $1 \le i \le m$ ;
L3: Initialize set $V$ to $\varnothing$ ;
L4: FOR each value of $j$ from 1 to $n$ DO
L5:   FOR each duration $d \in U$ DO {
L6:     Set $D = j \cdot d$ ;
L7:     IF $D \in V$ THEN continue; ELSE $V = V \cup \{D\}$ ;
L8:     Construct set $s$ such that
L9:           $s = \{k : k \in C$ and $D_k$ divides $D\}$ ;
L10:    FOR each non-empty subset $\sigma$ of $s$ DO {
L11:      Update $T = T + D \cdot \mathrm{DOMU}(\sigma,\ D,\ C - \sigma,\ n)$ ;
L12:    }
L13: }
L14: }
L15: Return($T$);
**End Procedure**
```

The modified recursive procedure DOM(), called now DOMU(), is given below:

```
**Procedure** DOMU $(s,\ d,\ v,\ t)$
L1: IF (($s == \varnothing$) OR ($t < 0$)) THEN return(0);
L2: Compute the sum $q$ of the quotients $d/D_j$,
          for all $j \in s$ ;
L3: IF ($q > t$) THEN return(0);
L4: Initialize $p = 0$ ;
L5: Set $a$ = PD($d, \sigma, t$);
L6: Initialize $b = (P_0)^{t-q}$ ;
L7: FOR each value of $\hat{D}$ from $d - 1$ down to 1 DO {
L8:   Construct set $r$ such that
L9:         $r = \{j : j \in v$ and $D_j$ divides $\hat{D}\}$ ;
L10: FOR each non-empty subset $\sigma$ of $r$ DO {
L11:   Update $b = b + \mathrm{DOMU}(\sigma,\ \hat{D},\ v - \sigma,\ t - q)$ ;
L12: }
L13: }
L14: Update $p = p + a \cdot b$ ;
L15: Return($p$);
**End Procedure**
```

5 Computation of average T under arbitrary critical or non-critical durations

In this section, we allow the duration D_0 of the non-critical access to be arbitrary too. We note that the recursive procedure DOMU() cannot now be called for every value of D as in line L5 of TDU() because the contribution to the average duration T of DOMU() in Line L11 of TDU() is not necessarily D but it may be D_0 instead, if $D_0 > D$ and somewhere in the recursion of DOMU() at least one non-critical access is involved in the computation. For this reason we proceed as follows:

We parameterize procedure DOMU() with the probability of the non-critical access P_ε and create a new procedure DOMG($s, d, v, t, P_\varepsilon$) that technically differs from DOMU() only in the extra parameter P_ε. We have then procedure DOMG() called by a new driver procedure TDG(). For all target durations D (line L6 of TDG()) that are smaller than D_0 we call procedure DOMG($s, d, v, t, P_\varepsilon$) by setting d to D, s to the current subset σ, v to $C - \sigma$, t to n and P_ε to 0 (line L12 of TDG()) so that DOMG() computes the probability of attaining the dominance degree D without any non-critical accesses (they are forced out of the computation by $P_\varepsilon = 0$). The contribution to the average T for the case where the currently targeted duration D is less than D_0 and non-critical accesses are absent is found then by multiplying D with the probability computed by DOMG() (line L13 of TDG()). The contribution to the average T for the case where the currently targeted duration D is less than D_0 and non-critical accesses are present is found by multiplying D_0 with the probability computed by DOMG() with parameter P_ε set to P_0 minus the probability computed by DOMG() with parameter P_ε set to 0 (line L15 of TDU()). Otherwise, if $D \geq D_0$, the contribution is computed as in TDU(), namely by calling DOMG() with parameter P_ε set to P_0 (line L17 of TDG()). Procedures TDG() and DOMG() are given below:

```
**Procedure** TDG $(n, C)$
L1: Initialize $T = (P_0)^n \cdot D_0$ ;
L2: Initialize set $U$ to be the set of all distinct
        critical duration values $D_i$, $1 \leq i \leq m$ ;
L3: Initialize set $V$ to $\varnothing$ ;
L4: FOR each value of $j$ from 1 to $n$ DO
L5:   FOR each duration $d \in U$ DO {
L6:     Set $D = j \cdot d$ ;
L7:     IF $D \in V$ THEN continue; ELSE $V = V \cup \{D\}$ ;
L8:     Construct set $s$ such that
L9:         $s = \{k : k \in C$ and $D_k$ divides $D\}$ ;
L10:    FOR each non-empty subset $\sigma$ of $s$ DO {
L11:      IF $D < D_0$ THEN
L12:        Set $y = \mathrm{DOMG}(\sigma,\ D,\ C - \sigma,\ n,\ 0)$ ;
L13:        Update $T = T + D \cdot y$ ;
L14:        Set $z = \mathrm{DOMG}(\sigma,\ D,\ C - \sigma,\ n,\ P_0)$ ;
L15:        Update $T = T + D_0 \cdot (z - y)$ ;
L16:      ELSE
L17:        Update $T = T + D \cdot \mathrm{DOMG}(\sigma, D, C - \sigma, n, P_0)$;
L18:    }
L19: }
L20: }
L21: Return($T$);
**End Procedure**
```

```
**Procedure** DOMG $(s,\ d,\ v,\ t,\ P_\varepsilon)$
L1: IF (($s == \varnothing$) OR ($t < 0$)) THEN return(0);
L2: Compute the sum $q$ of the quotients $d/D_j$,
          for all $j \in s$ ;
L3: IF ($q > t$) THEN return(0);
L4: Initialize $p = 0$ ;
L5: Set $a$ = PD($d, \sigma, t$);
L6: Initialize $b = (P_\varepsilon)^{t-q}$ ;
L7: FOR each value of $\hat{D}$ from $d - 1$ down to 1 DO {
L8:   Construct set $r$ such that
L9:       $r = \{j : j \in v$ and $D_j$ divides $\hat{D}\}$ ;
L10: FOR each non-empty subset $\sigma$ of $r$ DO {
L11:   Update $b = b + \mathrm{DOMG}(\sigma,\ \hat{D},\ v - \sigma,\ t - q,\ P_\varepsilon)$ ;
L12: }
L13: }
L14: Update $p = p + a \cdot b$ ;
L15: Return($p$);
**End Procedure**
```

Table 1: Timing comparison (sec) of Eq. (5) and TIEWIN()

n	m	Eq. 8	TD
8	4	0	0
8	6	0	0
8	8	0.016	0.008
16	4	0.001	0
16	6	0.032	0.016
16	8	1.764	0.188
24	4	0.004	0.004
24	6	0.296	0.092
24	8	37.436	1.984
32	4	0.016	0.008
32	6	1.584	0.404
32	8	342.16	12.66

6 Experimental results

All procedures are implemented in C. The time requirements of the procedures are computed on a Dell Precision T7500 workstation with dual Intel Xeon Quad core E5620 2.40GHz processors with 6GB RAM. Table 1 shows the comparison of the time requirement of the computation of the winning voting probability by Eq. (5) vs. its computation by procedure TIEWIN(). As can be seen, procedure TIEWIN() is significantly faster for larger number of candidates m and voters n.

The rest of the results concern the frame execution time. We demonstrate by simulation the correctness of the values computed by procedure TD() for the unit durations case and procedure TDG() for the unequal durations case.

For the unit durations case, we consider different values for the number n of requests in the frame, $n = 2, 4, 8, 16$ and different values for the number m of critical resources, $m = 1, 2, 3, 4$. For each value of n, we generated randomly 1000 frames for each n, where in each request in the frame, either non-critical code is executed with probability P_ε ($P_\varepsilon = 0.5, 0.6, 0.7, 0.8, 0.9$) or one of the m critical resources is accessed with probability $P_{C_i} = (1 - P_\varepsilon)/m$, $1 \leq i \leq m$. Then we compute the average T over 1000 frames.

Table 2 shows the average simulated value of the completion time (assuming a left-to-right priority policy for resolution of conflicts for the same critical resource) (column labeled by "Sim.") and the average value of T obtained by TD (column labeled by TD). As can be seen, the values computed by TD are corroborated by the simulation.

For the unequal durations case, we assume a value of $m = 2$ critical resources with probabilities $P_0 = 0.4$, $P_1 = 0.4$, $P_2 = 0.2$ for a first scenario and $P_0 = 0.4$, $P_1 = 0.1$, $P_2 = 0.2$ for a second scenario. For each of these cases, we vary the durations to $[D_0 = 1,\ D_1 = 4,\ D_2 = 2]$, $[D_0 = 1,\ D_1 = 2,\ D_2 = 4]$, $[D_0 = 3,\ D_1 = 4,\ D_2 = 2]$ and $[D_0 = 3,\ D_1 = 2,\ D_2 = 4]$. The results (for 1000 random frames) are shown in Table 3. As can be seen again, the values computed by TDG are corroborated by the simulation.

Finally, Table 4 gives the time that procedure TDG() takes for unequal durations when D_0, D_1, D_2 take values from the set $\{[4, 2, 1], [400, 4, 1], [4, 400, 3], [4, 400, 100]\}$, while all remaining durations $D_i, 3 \leq i \leq m$, are set to

Table 2: Average T for unit durations

$n=2$ requests								
	$m=1$		$m=2$		$m=3$		$m=4$	
P_ε	Sim.	TD	Sim.	TD	Sim.	TD	Sim.	TD
0.5	1.2525	1.2500	1.1274	1.1250	1.0843	1.0833	1.0640	1.0625
0.6	1.1626	1.1600	1.0821	1.0800	1.0547	1.0533	1.0407	1.0400
0.7	1.0919	1.0900	1.0462	1.0450	1.0308	1.0300	1.0224	1.0225
0.8	1.0409	1.0400	1.0203	1.0200	1.0130	1.0133	1.0099	1.0100
0.9	1.0103	1.0100	1.0052	1.0050	1.0035	1.0033	1.0026	1.0025
$n=4$ requests								
	$m=1$		$m=2$		$m=3$		$m=4$	
P_ε	Sim.	TD	Sim.	TD	Sim.	TD	Sim.	TD
0.5	2.0675	2.0625	1.6131	1.6094	1.4339	1.4329	1.3372	1.3359
0.6	1.7348	1.7296	1.4133	1.4096	1.2890	1.2868	1.2230	1.2208
0.7	1.4433	1.4401	1.2416	1.2410	1.1683	1.1665	1.1275	1.1272
0.8	1.2118	1.20.96	1.1129	1.1116	1.0765	1.0761	1.0582	1.0578
0.9	1.0556	1.0561	1.0291	10290	1.0197	1.0195	1.0149	1.0147
$n=8$ requests								
	$m=1$		$m=2$		$m=3$		$m=4$	
P_ε	Sim.	TD	Sim.	TD	Sim.	TD	Sim.	TD
0.5	4.0071	4.0039	2.7939	2.7894	2.3388	2.3346	2.0923	2.0863
0.6	3.2226	3.2168	2.3180	2.3134	1.9754	1.9712	1.7872	1.7824
0.7	2.4595	2.4576	1.8542	1.8520	1.6230	1.6201	1.4937	1.4921
0.8	1.7671	1.7678	1.4375	1.4366	1.3111	1.3103	1.2423	1.2418
0.9	1.2304	1.2305	1.1252	1.1248	1.0870	1.0861	1.0664	1.0657
$n=16$ requests								
	$m=1$		$m=2$		$m=3$		$m=4$	
P_ε	Sim.	TD	simul	TD	Sim.	TD	Sim.	TD
0.5	8.0020	8.0000	5.1223	5.1196	4.0796	4.0754	3.5226	3.5181
0.6	6.4039	6.403	4.2014	4.1976	3.3943	3.3918	2.9612	2.9591
0.7	4.7996	4.8033	3.2602	3.2611	2.6873	2.6896	2.3779	2.3800
0.8	3.2284	3.2281	2.3180	2.3183	1.9752	1.9749	1.7884	1.7864
0.9	1.7828	1.7853	1.4485	1.4486	1.3220	1.3205	1.2503	1.2508

$D_i = 1$. It can be seen that the time depends also now on the magnitude of the durations.

7 Conclusion

We have shown how to compute the probability that a particular candidate c out of m candidates, $1 \le c \le m$, wins by plurality voting an election conducted by n voters, where each voter either votes for a single candidate i, $1 \le i \le m$, with probability P_i, or abstains with probability $P_0 = 1 - \sum_{i=1}^{m} P_i$. We have then showed how this result is involved in the computation of the average execution time of a frame of n simultaneous requests, where each request randomly asks for exclusive access to any of m available non-shareable resources with probability $P_i, 1 \le i \le m$, or for non-exclusive access to a common fully shareable resource with probability $P_0 = 1 - \sum_{i=1}^{m} P_i$. Each resource access is allowed to have a different duration D_i, $0 \le i \le m$. Applications of the formulas developed include analysis of ensemble classifiers and systems performance evaluation (critical sections in multithreaded programs with barrier synchronization, switch delay in computer networks and interconnection networks).

References

[1] G. E. Andrews and K. Eriksson, Integer Partitions, Cambridge University Press (2004).

[2] W. J. Dally and B. P. Towles, Principles and Practices of Interconnection Networks, Morgan-Kaufmann (2004).

[3] J. L. Hennessy and D. A. Patterson, Computer Architecture: A Quantitative Approach, 5th Edition, Morgan-Kaufmann (2011).

[4] S. Eyerman and L. Eeckhout, Modeling Critical Sections in Amdahl's Law and its Implications for Multicore Design, Proceedings of the International Symposium on Computer Architecture, pp. 362-370 (2010).

[5] S. C. Kwasny, B. L. Kalman, A. M. Engebretson and W. Wu, Real-Time Identification of Language from Raw Speech Waveforms, Proceedings of the International Workshop on Applications of Neural Networks to Telecommunications, pp. 161-167 (1993).

[6] F. Leon, S. A. Floria and C. Badica, Evaluating the Effect of Voting Methods on Ensemble-Based Classification,

Table 3: Average T for different durations for $m = 2$

	$[P_0 = 0.4,\ P_1 = 0.4,\ P_2 = 0.2]$							
	$[D_0 = 1,\ D_1 = 4,\ D_2 = 2]$		$[D_0 = 1,\ D_1 = 2,\ D_2 = 4]$		$[D_0 = 3,\ D_1 = 4,\ D_2 = 2]$		$[D_0 = 3,\ D_1 = 2,\ D_2 = 4]$	
n	Sim.	TDG	Sim.	TDG	Sim.	TDG	Sim.	TDG
2	73.88	3.84	2.98	3.04	4.26	4.32	3.60	3.68
4	6.76	6.80	5.01	4.97	6.90	6.90	5.20	5.12
8	12.99	13.02	9.07	8.87	12.91	13.02	8.86	8.88
16	25.72	25.67	16.26	16.29	25.66	25.67	16.17	16.29
32	51.09	51.21	30.30	30.54	51.23	51.21	30.59	30.54
64	102.32	102.40	57.80	58.19	30.59	30.54	58.14	58.19
	$[P_0 = 0.7,\ P_1 = 0.1,\ P_2 = 0.2]$							
	$[D_0 = 1,\ D_1 = 4,\ D_2 = 2]$		$[D_0 = 1,\ D_1 = 2,\ D_2 = 4]$		$[D_0 = 3,\ D_1 = 4,\ D_2 = 2]$		$[D_0 = 3,\ D_1 = 2,\ D_2 = 4]$	
n	Sim.	TDG	Sim.	TDG	Sim.	TDG	Sim.	TDG
2	2.03	2.01	2.34	2.41	3.26	3.27	3.46	3.53
4	3.04	3.01	3.97	3.85	3.77	3.77	4.51	4.47
8	4.87	4.98	7.15	6.83	5.17	5.22	6.94	7.01
16	8.98	8.85	12.86	13.03	8.78	8.87	12.81	13.04
32	16.38	16.28	25.72	25.67	16.46	16.28	25.97	25.67
64	30.23	30.53	50.82	51.21	30.32	30.53	51.14	51.21
	$[P_0 = 0.4,\ P_1 = 0.4,\ P_2 = 0.2]$							
	$[D_0 = 10,\ D_1 = 4,\ D_2 = 2]$		$[D_0 = 10,\ D_1 = 2,\ D_2 = 4]$		$[D_0 = 30,\ D_1 = 4,\ D_2 = 2]$		$[D_0 = 30,\ D_1 = 2,\ D_2 = 4]$	
n	Sim.	TDG	Sim.	TDG	Sim.	TDG	Sim.	TDG
2	8.49	8.48	8.08	8.00	20.94	21.28	21.52	20.80
4	10.23	10.33	9.75	9.73	27.22	27.53	26.92	27.11
8	13.96	13.93	10.90	10.80	29.90	29.86	29.72	29.73
16	25.74	25.69	16.31	16.35	31.47	31.47	30.02	30.02
32	51.09	51.21	30.30	30.54	51.33	51.30	32.54	32.57
64	102.32	102.40	57.80	58.19	102.73	102.40	58.14	58.19
	$[P_0 = 0.7,\ P_1 = 0.1,\ P_2 = 0.2]$							
	$[D_0 = 10,\ D_1 = 4,\ D_2 = 2]$		$[D_0 = 10,\ D_1 = 2,\ D_2 = 4]$		$[D_0 = 30,\ D_1 = 4,\ D_2 = 2]$		$[D_0 = 30,\ D_1 = 2,\ D_2 = 4]$	
n	Sim.	TDG	Sim.	TDG	Sim.	TDG	Sim.	TDG
2	9.52	9.50	9.63	9.62	27.74	27.70	28.44	27.82
4	9.97	9.99	10.07	10.05	29.80	29.82	29.88	29.85
8	10.10	10.10	10.80	10.68	30.00	30.00	30.00	30.00
16	11.05	10.99	14.07	14.18	30.00	30.00	30.02	30.02
32	16.52	16.42	25.78	25.73	30.08	30.04	31.92	31.93
64	30.23	30.53	50.82	51.21	32.85	32.96	51.33	51.39

Table 4: Timing (sec) of TDG()

n	m	D_0=4, D_1=2, D_2=1	D_0=400 D_1=4, D_2=1	D_0=4, D_1=400, D_2=3	D_0=4, D_1=400, D_2=100
8	4	0	0	0.008	0.004
8	6	0.004	0.008	0.068	0.072
8	8	0.028	0.052	0.392	0.484
16	4	0	0	0.024	0.028
16	6	0.032	0.068	0.264	0.344
16	8	0.484	0.94	1.704	2.436
24	4	0.004	0.008	0.048	0.064
24	6	0.172	0.344	0.636	0.98
24	8	4.4	8.62	6.96	9.5
32	4	0.016	0.032	0.088	0.136
32	6	0.7	1.4	1.536	2.408
32	8	25.28	49.596	29.156	35.052

Proceedings of the IEEE International Conference on Innovations in Intelligent Systems and Applications, pp. 1-6 (2017).

[7] G. Levitin, Evaluating Correct Classification Probability for Weighted Voting Classifiers with Plurality Voting, European Journal of Operational Research, Vol. 143, No. 3, pp. 596-607 (2002).

[8] X. Lin, S. Yacoub, J. Burns and S. Simske, Performance Analysis of Pattern Classifier Combination by Plurality Voting, Pattern Recognition Letters, Vol. 24, No. 12, pp. 1959-1969 (2003).

[9] X. Mu, P. Watta and M. H. Hassoun, Analysis of a Plurality Voting-Based Combination of Classifiers, Proceedings of the IEEE International Joint Conference on Neural Networks, pp. 304-309 (2008).

[10] S. Narayanan , B. N. Swamy and A. Seznec, An Empirical High-Level Performance Model for Future Many-Cores, Proceedings of the ACM International Conference on Computing Frontiers, pp.1-8 (2015).

[11] S. Sahai and A. Ekbal, Combining Multiple Classifiers Using Vote-Based Classifier Ensemble Technique for Named Entity Recognition, Data and Knowledge Engineering, Vol. 85, pp. 15-39 (2013).

[12] A. S. Tanenbaum and D. J. Wetherall, Computer Networks, 5th Edition, Pearson (2011).

[13] M. Van Erp, L. Vuurpijl and L. Schomaker, An overview and Comparison of Voting Methods for Pattern Recognition, Proceedings of the International Workshop on Frontiers in Handwriting Recognition, pp. 195-200 (2002).

[14] W. D. Wallis, The Mathematics of Elections and Voting, 4th Edition, Springer (2014).

[15] L. Yavits , A. Morad, and R. Ginosar, “The Effect of communication and synchronization on Amdahl’s law in Multicore Systems, Parallel Computing, Vol. 40, No. 1, pp.1-16 (2014).

Sourav Dutta is currently an Assistant Professor of Computer Science at Ramapo College of New Jersey, USA. He received the Ph.D. degree in Electrical and Computer Engineering from Southern Illinois University, Carbondale, USA, in 2017 and a Master’s degree in Electrical Engineering from the National Institute of Technology, Kurukshetra, India. His research interests include performance estimation and optimization of parallel programs.

Dimitri Kagaris holds a Ph.D. and M.S. degree in Computer Science from Dartmouth College, Hanover, New Hampshire, USA, and a Diploma degree in Computer Engineering and Informatics from the University of Patras, Greece. He is currently a professor in the Electrical and Computer Engineering Department, Southern Illinois University at Carbondale, USA. His current research interests include digital design automation, computer architecture, and formal verification.