Brief Announcement: Pareto Optimal Solutions to Consensus and Set Consensus

Armando Castañeda[*]
Department of Computer Science, Technion
armando@cs.technion.ac.il

Yannai A. Gonczarowski[†]
Center for the Study of Rationality and Institute of Mathematics, Hebrew University of Jerusalem
yannai@gonch.name

Yoram Moses[‡]
Department of Electrical Engineering, Technion
moses@ee.technion.ac.il

ABSTRACT

A protocol P is *Pareto-optimal* if no protocol Q can decide as fast as P for all adversaries, while allowing at least one process to decide strictly earlier, in at least one instance. Pareto optimal protocols cannot be improved upon. We present the first Pareto-optimal solutions to consensus and k-set consensus for synchronous message-passing with crashes failures. Our k-set consensus protocol strictly dominates all known solutions, and our results expose errors in [1, 7, 8, 12]. Our proofs of Pareto optimality are completely constructive, and are devoid of any topological arguments or reductions.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems-Distributed applications; D.4.5 [**Operating Systems**]: Reliability-Fault-tolerance; D.4.7 [**Operating Systems**]: Organization and Design-Distributed systems

Keywords

Consensus, k-set consensus, optimality, knowledge.

1. INTRODUCTION

The very first consensus protocols were ***worst-case*** optimal [13] (decisions are always taken no later than the known worst-case lower bound), deciding in exactly $\boldsymbol{t}+1$ rounds in all runs [4, 17], where $\boldsymbol{t}$ is an upper bound on the number of failing processes. It was soon realised that these can be strictly improved upon by ***early stopping*** protocols [3], which are also worst-case optimal, but can often decide much faster than the original ones. Following [11], this paper studies protocols that cannot be strictly improved upon, and are thus optimal in a much stronger sense.

An ***adversary*** is a tuple $\alpha = (\vec{v}, \mathsf{F})$, where $\vec{v}$ is a vector of input values from a domain $\mathtt{V}$ and F is a failure pattern. A *context* is a set of adversaries. W.l.o.g., we consider only full-information protocols (fip's). For a protocol P and an adversary $\alpha = (\vec{v}, \mathsf{F})$, we use $P[\alpha]$ to denote the run of P with inputs $\vec{v}$ and failure pattern F. We say that a protocol Q ***dominates*** a protocol P in context γ, denoted by $Q \preceq_\gamma P$ if, for every adversary $\alpha \in \gamma$ and every process i, if i decides in $P[\alpha]$ at time m_i, then i decides in $Q[\alpha]$ at some time $m'_i \leq m_i$. Q ***strictly dominates*** P if $Q \preceq_\gamma P$ and $P \not\preceq_\gamma Q$. Here we consider the synchronous message-passing model with n processes and $t < n$ crash failures.

The early-stopping consensus protocols of [3] strictly dominate the protocols of [17], which always decided at time $\boldsymbol{t}+1$. Nevertheless, these early stopping protocols may not be optimal solutions to consensus. A protocol P is an ***all-case optimal*** solution to a decision task T if P solves T and it dominates every protocol P' that solves T [13]. All-case optimal solutions to the *simultaneous* variant of consensus, in which all decisions are required to occur at the same time were presented in [5]. For the standard *eventual* variant of consensus, in which decisions are not required to occur simultaneously, no all-case optimal solution exists [15]. Consequently, Halpern, Moses and Waarts [12] initiated the study of a notion of optimality that is achievable by eventual consensus protocols:

DEFINITION 1. *A protocol P is a **Pareto-optimal** solution to a decision task T in a context γ if P solves T in γ and no protocol Q solving T in γ strictly dominates P.*

In other words, for all protocols Q that solve T, if there exist an adversary α and process i s.t. i decides in $Q[\alpha]$ strictly

[*]Supported in part at the Technion by an Aly Kaufman Fellowship.

[†]Supported in part by an ISF grant, by the Google Inter-university center for Electronic Markets and Auctions, and by the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013) / ERC grant agreement no. [249159].

[‡]The Israel Polak academic chair at Technion; this work was supported in part by the ISF grant 1520/11.

PODC'13, July 22–24, 2013, Montréal, Québec, Canada.
ACM 978-1-4503-2065-8/13/07.

earlier than in $P[\alpha]$, there must exist some adversary β and process j s.t. j decides in $P[\beta]$ strictly earlier than in $Q[\beta]$.

Halpern, Moses and Waarts logically characterised Pareto optimality, and presented a simple and efficient consensus protocol $P0_{\text{opt}}$ that they claimed was Pareto optimal.

We present Pareto-optimal protocols for consensus and k-set consensus. A new knowledge-based analysis [6,10] allows a simpler and more intuitive approach to Pareto optimality than that used in [12]. Our contributions are:

1. A Pareto-optimal consensus protocol, which strictly dominates the $P0_{\text{opt}}$ protocol from [12], proving that $P0_{\text{opt}}$ is, in fact, *not* Pareto optimal.
2. A Pareto-optimal protocol for k-set consensus, which strictly dominates all published solutions for k-set consensus in the synchronous model [2, 7–9, 16].
3. For a run with $\boldsymbol{f}$ failures, our protocols decide in at most $\boldsymbol{f}+1$ and $\lfloor \frac{\boldsymbol{f}}{k} \rfloor + 1$ rounds, respectively, contradicting lower bound proofs in [1,8] and possibly [7], and answering an open problem from [8]. This emphasises the subtlety of topology-based lower bounds [8] and of reduction-based ones [1, 7]. Notably, our proofs of Pareto optimality are completely constructive, devoid of any topological arguments or reductions.

2. PARETO-OPTIMAL CONSENSUS AND SET CONSENSUS

A ***node*** is a pair $\langle i, m\rangle$ referring to i's state at time m. $\langle j, \ell\rangle$ is ***seen*** by $\langle i, m\rangle$ (in a given run r) if there exists a message chain from j at time ℓ to i at time m. $\langle j, \ell\rangle$ is ***hidden*** from $\langle i, m\rangle$ (in r) if (a) i does not know that j has failed before time ℓ, and (b) $\langle j, \ell\rangle$ is not seen by $\langle i, m\rangle$. A ***hidden path*** w.r.t. $\langle i, m\rangle$ in run r is a sequence of processes $j_0, \ldots, j_{m-1}, j_m$ s.t. $\langle j_\ell, \ell\rangle$ is hidden from $\langle i, m\rangle$, for all ℓ.

Our construction of Pareto optimal protocols is assisted and guided by a knowledge-based analysis, in the spirit of [6, 10]. We consider the truth of facts at *points* (r, m)—time m in run r, with respect to a set of runs R (which we call a ***system***). The systems we are interested in have the form $R_P = R(P, \gamma)$ where P is a protocol and γ is the t-resilient synchronous message-passing model with inputs in $\mathtt{V} = \{0, 1\}$. We write $(R, r, m) \models A$ to state that fact A holds, or is satisfied, at (r, m) in the system R. We write $K_i A$ to denote that ***process*** i ***knows*** A, and define: $(R, r, m) \models K_i A$ iff $(R, r', m) \models A$ for all $r' \in R$ s.t. i has the same local state at (r, m) and (r', m).

A Pareto-optimal consensus protocol.

The definition of consensus implies that $\exists v$ (the fact "***some process started with*** v") is a precondition for deciding v. Thus, the Knowledge of Preconditions Theorem [14] implies:

LEMMA 1. *$K_i\exists v$ is a precondition for i deciding on v, for every value v in any consensus protocol.*

While $K_i\exists v$ is a necessary condition for deciding v, if $K_i\exists 0$ is used as a sufficient condition for decide_0 then $K_i\exists 1$ cannot be sufficient for decide_1, since this may prevent agreement: Everyone would decide on their own value at time 0. The following is a consensus protocol in which decisions on 0 are performed as soon as possible:

Protocol P_0 (for an undecided process i at time m):

if $K_i\exists 0$	**then** decide_0
if $m = \boldsymbol{t}+1$ and $\neg K_i\exists 0$	**then** decide_1

The following lemma provides a key step to designing a Pareto-optimal consensus protocol that dominates P_0:

LEMMA 2. *If $Q \preceq P_0$ solves consensus, then every active process i decides 0 in Q when $K_i\exists 0$ first holds.*

In consensus, a precondition for deciding 1 in run r is that no correct process ***ever*** decides 0. By Lemma 2, in any consensus protocol that dominates P_0 processes decide 0 as soon as they know $\exists 0$. It follows that a precondition for deciding 1 in such a protocol is that no correct process will *ever* know $\exists 0$ (denoted by $\mathsf{never\text{-}known}(\exists 0)$). Indeed, by the Knowledge of Preconditions Theorem [14], a process deciding 1 must know this fact. This is equivalent to knowing that no active process *currently knows* $\exists 0$.

LEMMA 3. *The following are equivalent at time m:*
(i) $K_i(\mathsf{never\text{-}known}(\exists 0))$, *and*
(ii) $\neg K_i\exists 0$ **&** *there is no hidden path w.r.t. $\langle i, m\rangle$.*

I.e., as long as there is a hidden path w.r.t. $\langle i, m\rangle$, process i considers it possible that some process currently knows $\exists 0$. Once such a path is excluded, the process can safely decide 1. This leads to a Pareto-optimal (fip) protocol in which decisions on 0 occur as soon as possible, and on 1 as soon as a process knows that 0 will never be decided on:

Protocol OPT_0 (for an undecided process i at time m):

if $K_i\exists 0$	**then** decide_0
elseif no hidden path w.r.t. $\langle i, m\rangle$ exists	**then** decide_1

THEOREM 1. OPT_0 *is a Pareto optimal consensus protocol; in every execution, all processes decide in* OPT_0 *by time $\boldsymbol{f}+1$ at the latest, where $\boldsymbol{f}$ is the number of processes that actually fail in the execution.*

Both OPT_0 and the protocol $P0_{\text{opt}}$ from [12] decide 0 when $\exists 0$ is known, but they differ in the rule for deciding 1. In $P0_{\text{opt}}$ a process decides 1 following a round in which it has not discovered a new failure. This condition implies the nonexistence of a hidden path, but is strictly weaker than it. E.g., in a run in which all initial nodes are seen at $\langle i, 2\rangle$ but process i has seen one failure in each of the first two rounds, i decides in OPT_0 but does not decide in $P0_{\text{opt}}$.

COROLLARY 1. *Protocol $P0_{\text{opt}}$ [12] is not Pareto optimal.*

A Pareto-optimal k-set consensus protocol.

OPT_0 can readily be extended to cover the case in which $\mathtt{V} = \{0, \ldots, d\}$ for $d > 1$. The rule for 0 is unchanged, and if no hidden path exists a process can decide on the minimal value it has seen. Thus, a process decides v when it knows $\exists v$ and that correct processes will never see a smaller value. We call this protocol $\text{OPT}_{\min}$.

For k-set consensus the input domain is $\mathtt{V} = \{0, .., d\}$, $d \geq k$, and it is required that the correct processes decide on at most k distinct values (thus 1-set consensus is consensus).

We present a k-set consensus protocol $\text{OPT}_{\text{min-}k}$ that generalizes OPT_{min}, in which every process decides on a ***low*** value (i.e. a value in $\{0, \ldots, k-1\}$) as soon as possible, and decides on a ***high*** (i.e. non-low) value w as soon as it knows that no k values smaller than w will be decided on. In every run, let $V\langle i, m\rangle$ denote the set of all values v s.t. $K_i \exists v$ holds at m. Process i is called ***low*** at time m if $V\langle i, m\rangle$ contains a low value, otherwise it is ***high***. We call $v \in \mathtt{V}$ ***minimal*** in r if it is a minimal value of some set $V\langle i, m\rangle$ in r. Finally, the ***hidden capacity*** $\mathsf{HC}(i, m)$ of $\langle i, m\rangle$ (in r) is the number c of pairwise node-disjoint hidden paths w.r.t. $\langle i, m\rangle$.

Our Pareto-optimal k-set consensus protocol is the fip with the following single decision rule:

Protocol $\text{OPT}_{\text{min-}k}$ (for an undecided process i at time m):

if $\langle i, m\rangle$ is low or $\mathsf{HC}(i, m) < k$ **then** $\mathsf{decide}_{\min V\langle i,m\rangle}$

Hidden capacity plays an analogous role to hidden paths. We note that it is possible both to implement fip's for crash failures and to compute $\mathsf{HC}(i, m)$ efficiently. Our correctness proof for $\text{OPT}_{\text{min-}k}$ is based on a generalization of Lemma 3:

LEMMA 4. *In the crash model, if $\langle i, m\rangle$ is a high node with minimal value v, then K_i(fewer than k values smaller than v will ever be minimal values) is equivalent to $\mathsf{HC}(i, m) < k$.*

To show that $\text{OPT}_{\text{min-}k}$ is Pareto optimal, one additionally needs an analogue of Lemma 2. Unfortunately, while in every protocol dominating $\text{OPT}_{\text{min-}k}$ every process must decide when it becomes low, it is no longer true, due to the relaxed k-set agreement condition, that every such process must decide on its minimal value. Nonetheless, we show that under certain conditions, a low process knowing exactly one low value must decide on it. Establishing this analogue of Lemma 2 is the main technical challenge in our proof. Notably, this proof is constructive, and does not employ topological arguments, reductions or simulations. Fortunately, this analogue of Lemma 2, despite the added conditions it requires, allows us to prove the following, showing that no k-set consensus protocol strictly dominates $\text{OPT}_{\text{min-}k}$:

COROLLARY 2. *Let P be a k-set consensus protocol, in which an undecided low process decides immediately. Then no high process with hidden capacity $\geq k$ can decide in P.*

Using Lemma 4 and Corollary 2, we can prove:

THEOREM 2.

(i) $\text{OPT}_{\text{min-}k}$ *is a Pareto optimal k-set consensus protocol.*

(ii) In every execution, all processes decide in $\text{OPT}_{\text{min-}k}$ *by time $\lfloor \frac{f}{k} \rfloor + 1$ at the latest, where $\boldsymbol{f}$ is the number of processes that actually fail in the execution.*

Discussion. Interestingly, all known k-set consensus protocols in the synchronous crash model [2,7,9,16] are ***strictly dominated*** by $\text{OPT}_{\text{min-}k}$. Moreover, as pointed out by an anonymous referee, its properties contradict the published lower bounds in [1,8] and possibly [7] (whose model is slightly nonstandard). Although $\text{OPT}_{\text{min-}k}$ decides in $\lfloor \frac{f}{k} \rfloor + 1$ rounds, since $\boldsymbol{f}$ is not known in advance it would be able to stop only in $\min\{\lfloor \frac{t}{k} \rfloor + 1, \lfloor \frac{f}{k} \rfloor + 2\}$ rounds. In the case of consensus, this is perfectly consistent with [3], who mention in passing that decision by time $\boldsymbol{f} + 1$ is possible. However, [1, 7, 8] claim to prove explicitly that no k-set consensus protocol can always decide by time $\lfloor \frac{f}{k} \rfloor + 1$ (also contradicting [3] and OPT_0 even when $k = 1$). In fact, [8] pose as an open question whether decision is *ever* possible before time $\lfloor \frac{f}{k} \rfloor + 2$. Both of our Pareto-optimal protocols OPT_0 and $\text{OPT}_{\text{min-}k}$ contradict these stated lower bounds, and provide a negative answer to this open problem. Moreover, they are not only optimal in a worst-case sense; they are truly unbeatable in the sense that no protocol can strictly improve upon them. These are the first such protocols.

3. REFERENCES

[1] D. Alistarh, S. Gilbert, R. Guerraoui, and C. Travers. Of choices, failures and asynchrony: The many faces of set agreement. *Algorithmica*, 62(1-2):595–629, 2012.

[2] S. Chaudhuri, M. Herlihy, N. A. Lynch, and M. R. Tuttle. Tight bounds for k-set agreement. *J. ACM*, 47(5):912–943, 2000.

[3] D. Dolev, R. Reischuk, and H. R. Strong. Early stopping in Byzantine agreement. *J. of the ACM*, 34(7):720–741, 1990.

[4] D. Dolev and H. R. Strong. Requirements for agreement in a distributed system. In H. J. Schneider, editor, *Distributed Data Bases*, pages 115–129. North-Holland, Amsterdam, 1982.

[5] C. Dwork and Y. Moses. Knowledge and common knowledge in a Byzantine environment: crash failures. *Information and Computation*, 88(2):156–186, 1990.

[6] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, 2003.

[7] E. Gafni, R. Guerraoui, and B. Pochon. The complexity of early deciding set agreement. *SIAM J. Comput.*, 40(1):63–78, 2011.

[8] R. Guerraoui, M. Herlihy, and B. Pochon. A topological treatment of early-deciding set-agreement. *Theor. Comput. Sci.*, 410(6-7):570–580, 2009.

[9] R. Guerraoui and B. Pochon. The complexity of early deciding set agreement: How can topology help? *Electr. Notes Theor. Comput. Sci.*, 230:71–78, 2009.

[10] J. Y. Halpern and Y. Moses. Knowledge and common knowledge in a distributed environment. *J. of the ACM*, 37(3):549–587, 1990.

[11] J. Y. Halpern, Y. Moses, and O. Waarts. A characterization of eventual Byzantine agreement. In *Proc. 9th ACM Symp. on Principles of Distributed Computing*, pages 333–346, 1990.

[12] J. Y. Halpern, Y. Moses, and O. Waarts. A characterization of eventual byzantine agreement. *SIAM J. Comput.*, 31(3):838–865, 2001.

[13] M. Herlihy, Y. Moses, and M. R. Tuttle. Transforming worst-case optimal solutions for simultaneous tasks into all-case optimal solutions. In *PODC*, pages 231–238, 2011.

[14] Y. Moses. *Knowledge and Distributed Coordination*. Morgan Claypool. in preparation.

[15] Y. Moses and M. R. Tuttle. Programming simultaneous actions using common knowledge. *Algorithmica*, 3:121–169, 1988.

[16] P. Raipin Parvédy, M. Raynal, and C. Travers. Early-stopping k-set agreement in synchronous systems prone to any number of process crashes. In *PaCT*, pages 49–58, 2005.

[17] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *J. of the ACM*, 27(2):228–234, 1980.