Maxima - A Fast Fearless Secure Sharded Block-Chain

Maximilian E. Chamberlin
MA (Oxford) M.Phil (Cambridge)
Maxima Organisation
mec@Maxima.org

Abstract

Recently, high demand and limited scalability have increased the average transaction times and fees in popular cryptocurrencies, yielding an unsatisfactory experience. Here we introduce Maxima a cryptocurrency with a sharded block-chain, one where the network is divided into partitions called shard which maintain their own blocks.

In Maxima, validator committees are sampled for each shard. Validators vote on the availability of blocks to the network, and are shuffled off shards very quickly. Once a committee has determined if a block is accessible to nodes on a shard, nodes on that shard execute the transactions and verify the blocks validity- producing a succint fraud proof if the block is invalid. Cross shard transactions are managed using tools from distributed systems, like locks and yanks.

Maxima is a globally distributed computer that is secure under an honest majority assumption, and puts in place mechanisms to prevent adaptive adversaries from corrupting the network. With 20 shards, Maxima is projected to process over 20,000 transactions per second.

Part I
Introduction

1 Motivation and Outline

In Bitcoin, blocks are added to the block-chain approximately every 10 minutes, at a relatively fixed pace that results in a TPS of 5-7. Simple solutions to expanding the capacity of the block-chain have found natural limits to their efficacy: enlarging blocks runs up against user bandwidth limits; hastening the rate at which blocks are mined increases orphaning.

A more promising solution to the problem of scaling is to run separate chains that process transactions within partitions of the network, called shards. The key challenges are: (1) how to manage the communication between the different shards, a problem which is closely related to ensuring the atomicity of transactions in a distributed system and (2) how to run these many separate block-chains in a way that does not compromise on security, since each shard-chain will only have a fraction of the mining/validation power of the network.

In this paper, we propose mechanisms to ensure the atomicity of transactions and to ensure that the security of separate chains is not compromised. We adopt an honest majority assumption for the network, assuming fewer than a fraction f of 0.25 nodes are Byzantine. Note $f = 0.25$ is an

29th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

arbitrary constant bounded below 1/3 to ensure good constants. We also put in place safe-guards against adaptive adversaries who may bribe network participants.

With this in mind, our protocol can be described below. Validator committees are sampled for each shard. Sampling validators randomly means that with a sufficient number (400) we can ensure an honest majority of 2/3 of per shard almost surely.

Validators then vote on the availability of blocks to the network, and are shuffled off shards very quickly. This fast shuffling is to ensure that an adaptive adversary does not have the time to find and corrupt the validators of each shard. Because of this fast shuffling, the work that a validator can perform must be very minimal. Validators check that the data corresponding to the Merkle root of a block is accessible to a shard, so that the members of a shard can execute those transactions. If there is a dispute about transaction execution, validators also resolve disputes by considering succinct proofs of invalid execution. Cross shard transactions are managed using tools from distributed systems, like yanks, to ensure that transactions can be executed atomically.

2 Inspiration and Original Contributions

We would like to thank: Ethereum and Dfinity for the idea of separating state execution from data availability checking; Ethereum for the idea of using Erasure Coding to prove availability; Zilliqa and Dfinity for the idea of forming committees using random sampling to vote on blocks. Our original contributions are:

- Formalising a concrete mechanism for multi-dimensional self-authenticating erasure codes, without checking more than O(1) Merkle roots. Previous solutions have relied on further data availability proofs or an increased burden on verifiers.
- We are the first to apply erasure codes to prove that a program's intermediate state execution has been committed to the network, which enables the generation of succinct fraud proofs that execution is incorrect. Previous solutions to off-loading state execution, like Trubit, relied on rounds of interactivity to generate proofs of incorrectness.
- We also developed variants of a cross shard locking system which were eventually adapted by Ethereum to a cross-shard yanking system, which we have also adopted.

3 Protocol Structure

The construction of our protocol depends on the following, relatively independent components, which can be subdivided into three themes: voting schemes; data creation and concurrency controls.

1. Consensus Schemes:
   (a) Voting Schemes - Forkful vs Fork-free
   (b) Validator Sampling and Shuffling
   (c) Marginal Voting Incentives
2. Data Creation
   (a) Block Creation
   (b) Data Availability Checking Through Erasure Codes
   (c) Fraud Proofs
3. Concurrency Controls
   (a) Locking State for Cross Shard Transactions
   (b) Stateless Clients

Part II
Consensus Schemes

4 Voting Schemes - Forkful vs Fork-free

Broadly speaking, there are two kinds of block-chain protocols: those that admit forks and those that don't.

One of the chief benefit of forking consensus protocols like Bitcoin, is that they allow the network to recover from majority attacks. Even if a "dishonest" majority creates an invalid chain, so long as the network recovers its honest majority, the invalid chain may be overtaken by a valid one. we may summarise this by saying: In Bitcoin, votes are continuously validate prior blocks and so no block is truly finalised. It may at any point be reverted. Conversely, in fork-free protocols, such as PBFT, once a vote validating a block occurs it is final.

However, in the context of a sharded block-chain forking consensus protocols need a great deal of extra apparatus due to inter-shard dependencies. If a fork on the block-chain occurs on shard A, and shard B has blocks that depend on the state in shard A, then a reversion of blocks on shard A will need to effect a reversions of blocks on shard B. Tracking all the state dependencies and ensuring that reversions do not revert too much of the global state is still an unsolved problem in the block-chain space. This problem is compounded by the fragmentation of state that comes with sharding. If state is distributed across many shards, then a large proportion of transactions on the block-chain reference state between shards, meaning a large number of dependencies, and reversions on one shard causing reversions on many others . Bearing these thoughts in mind, a sharded block-chain must either:

1. Track inter-shard dependencies created by transactions, and determine an efficient way of reverting those changes, thereby maintaining the resilience of the block-chain to majority attacks.
2. Adopt a consensus protocol that admits no forks at the application layer, thereby losing the flexibility of a block-chain to withstand short-lived 50% attacks on the network.

In Maxima, we adopt the second approach but with the following fail-safe: In the event of a large-scale attack, whether clients would chose the dishonest chain or chose to revert those changes would be determined largely by social consensus: a proof that an attack had been performed by a dishonest majority with encouragement from the Maxima organisation about which block to revert to.

Having made the case for forking chains, it should be noted that in the entire history of Bitcoin only once was a block-chain reverted by more than its finalisation threshold of 6 blocks. This incident occurred due to a mining consortium not checking block validity when mining. However, a social layer of consensus has been used at least twice to change the protocol. This means that using social consensus to resolve short-term attacks does not come with much overhead, if historical precedent is to be assumed.

More concretely, our voting scheme:

5 Achieving Fork-Free Consensus

Fork-free protocols typically use BFT style algorithms. These are systems where a vote for a block consists of a prepare followed by a commit. They have been modified in recent times to create significant improvements.

1. The PBFT modification - have a leader who aggregates votes to improve efficiency
2. Subsample from the entire population so many validators are voting but not all (in Zilliqa, they use X). If say 400 participants are voting and we assume that f<25% are byzantine. We can show that the probability more than 2/3 of those voting are also byzantine is less than the number of atoms in the universe.

3. Ensure the right kind of marginal incentive structures exist. One can dot his by using a system similar to Caspar slashing conditions. Not only does this act as a deterrent, but will also be a key sign of bribary that could lead up to a social consensus step.
4. Selecting the participants randomly and quickly, using DFinity stuyle consensus.

6 Separating Validity checking from State Execution

- Key to security is fast shuffling. This means that the excutors can't do a lot.
- They attest to availability of blocks, fixed if others agree not so if others disagree.

7 Execution disputes and Validity

- Block creaters order and create transactions. The creators are sampled randomly from afixed set.
- They need a way of determining whether a transaction can pay them the gas to run the TX, and also if the ordering of transactions is correct.
- They need to do this without having access to all the state.
- Light clients can store all this information, and transactions need to include access lists of data that they need.
- Zero knowledge proofs that I have the gas needed.
- Q: how can a transaction determine if two things are blocking. Just create a merkle tree that tracks these things.
- We can use a multi-dimensional eraure code to determine quickly if some data is available.
- Key to security is fast shuffling. This means that the excutors can't do a lot. They can't maintain state on each shard.

Blocks are created within each shard, and the merkle root is signed on and shared with other shards. The question we now have is this?

(1) How can we ensure that all of the data contained within a shard is available to all members of the shard?

(2) Given that data isavailable, this means anyone can report flaws.

The question is how can we prove that a block is available?

One could require a majority of half the sampled validators perform availability checks and then sign off on the availability of a root.

Question is how can we feed in the right marginal incentives?

- Have rewards for attesting yes, but with penalties if they don't have proof they checked for data availability. This means nodes are incentivised to say yes if they do have the data, but won't declare yes if they don't and know others won't check them. small reward for saying no.
- randomly check availables for a reveal, with sufficinelty high penalty that makes lying unprofitable. Also allow challenges for those who know ing wrong. For those who don't require a random reveal with sufficiently high penalties. => say that a person has 10 blocks to do so before being slashed.
- unavailable|available are losing their rewards

If they are aware many other people do not have access to the data, they may vote yes anyway.

However, we don't want validators to have to actualy execute the state. This would take too long.

1. How can we resolve this issue? Ethereum uses Truebit, which takes the form of an interactive verification game. The main issue with truebit is that it requires multiple rounds of verification and thus can result in low latency in cases where there is a dispute.

2. The approach that we take is to erasure encode the trace of the execution. We use a 3d erasure encoding to ensure that the fraud proofs are of a size cube-root(n).
3. Again using sampling techniques, we can assign validators to each transaction.

Stateless clients

Self-Authenticating Erasure Codes

We want validators to be able to guarantee that honest nodes can access data on the network, or stated more succinctly that **data is available**. A simple way for validators to check that data is available is to download a whole block. So, in the current scheme we would sample a set of validators - say 400, which is enough to guarantee an honest majority of participants. Then all of these shards would attempt to download the data from the network, and finaly take a vote between themselves on whether the data is available.

However, to increase efficiency, we would like to use a scheme where each validator need not download the entire block. If blocks were really large, say GB sized, which would be ideal for scalability, it may be impractical to have validators enter a shard and download multiple blocks to verify availability.

We can avoid this, if we take a sampling approach. However, validators will not sample directly from the block itself, but instead will sample from an erasure code.

Why do we want validators to check that data is available for a block? If the data for a block is available to the network, this means that honest executors on a shard can check to see if there are any faults with it and produce succint fraud proofs, O(1) sized proofs that a block contains invalid data.

What this ultimately means is that executors on a shord can check to see if

We don't actually want the validators to perform the executions themselves because doing so would take too long for large blocks of for transactions that take a long tiem to execute.

- N - the number of verifiers
- P - the number of checks made by each client
- Denot

Collectively N verifiers want to check a block of size M is available. They wish to do so without

8 Construction

- K - The expansion factor of each read solomon code
- Let D_{ij}, F_{ij}, G_{ij} represent the data int he first, second and third squares respectively
- x,y,z are respectively the bilinear accumulator constants for the committed values
- First Square:
  - Let d_{ij} represent chunk i, j of the original data
  - x_{ij} is the bilinear accumulator witness for the tuple (d_{ij}, i, j), that is $e[(d_{ij}, i, j), x_{ij}] = x$
  - D_{ij} represent $(d_{ij}, i, j), x_{ij}$
- Second Square:
  - Let f_{ij} represent the j th evaluation point in the erasure code for row D_{i*}
  - y_{ij} is the bilinear accumulator witness for the tuple (f_{ij}, i, j), that is $e[(f_{ij}, i, j), y_{ij}] = y$
  - F_{ij} represent $(f_{ij}, i, j), y_{ij}$
- Third Square:
  - Let g_{ij} represent the i th evaluation point in the erasure code for column F_{*j}

- z_{ij} is the bilinear accumulator witness for the tuple (g_{ij}, i, j), that is $e[(g_{ij}, i, j), z_{ij}] = z$
- G_{ij} represent $(g_{ij}, i, j), z_{ij}$

- An example is given below with K =2/3:

$(d_{00}, 0, 0), x_{00}$	$(d_{01}, 0, 1), x_{01}$	$(f_{00}, 0, 0), y_{00}$	$(f_{01}, 0, 1), y_{01}$	$(f_{02}, 0, 2), y_{02}$
$(d_{10}, 1, 0), x_{10}$	$(d_{11}, 1, 1), x_{11}$	$(f_{10}, 1, 0), y_{10}$	$(f_{11}, 1, 1), y_{11}$	$(f_{12}, 1, 2), y_{12}$
		$(g_{00}, 0, 0), z_{00}$	$(g_{01}, 0, 1), z_{01}$	$(g_{02}, 0, 2), z_{02}$
		$(g_{10}, 1, 0), z_{10}$	$(g_{11}, 1, 1), z_{11}$	$(g_{12}, 1, 2), z_{12}$
		$(g_{20}, 2, 0), z_{20}$	$(g_{21}, 2, 1), z_{21}$	$(g_{22}, 2, 2), z_{22}$

8.1 Enforcing Authentication

If an element in D or F does not satisfy it's authentication constraint, i.e $e[(f_{ij}, i, j), y_{ij}] \neq y$ and element is available then this can be proved in $O(\sqrt{m})$. Say that an element in the second square does not satisfy its constraint, this can be proven from $\sqrt{(m)}$ elements in the corresponding column.

8.2 Proof of Uniqueness

Given all available elements are authenticated. If it is possible to construct more than one element in the $ijth$ index of any of D, F, G, then an O(1) proof is given by presenting something like:

$(d_{00}, 0, 0), x_{00}$, $(d_{00}^{'}, 0, 0), x_{00}^{'}$, where $d_{00} \neq d_{00}^{'}$

This means all values must be unique or a succinct fraud proof can be given.

8.3 Proof of Liveness

Given points 8.1, we can prove liveness.

Suppose all available elements of D, F satisfy the authentication constraints then:

If the data for a row D_{i*} is unavailable then:

#available elements in $F_{i*} < \sqrt{m}$ or equivalently:

#unavailable elements in $F_{i*} > (k-1)\sqrt{m}$. This implies:

#unavailable or **unauthenticated** elements in $G > (k-1)^2 m$ or equivalently:

#available and **authenticated** elements in $G < k^2 m - (k-1)^2 m = (2k-1)m$

Therefore a liveness check needs to ensure that more than $(2k-1)m$ elements of G are available and **authenticated** with a high likelihood. If this be the case, we can reconstruct some dataset D.

Collaborative Generation

$(d_{00}, 0, 0), w_{00}), w_{w_{00}}$	$(d_{01}, 0, 1), x_{01}$	$(d_{02}, 0, 2), x_{02}$	$(f_{00}, 0, 0), y_{00}$	$(f_{01}, 0, 1), y_{01}$	$(f_{02}, 0, 2), y_{02}$
$(d_{10}, 1, 0), x_{10}$	$(d_{11}, 1, 1), x_{11}$	$(d_{12}, 1, 2), x_{12}$	$(f_{10}, 1, 0), y_{10}$	$(f_{11}, 1, 1), y_{11}$	$(f_{12}, 1, 2), y_{12}$
			$(g_{00}, 0, 0), z_{00}$	$(g_{01}, 0, 1), y_{01}$	$(g_{02}, 0, 2), y_{02}$
			$(g_{10}, 1, 0), y_{10}$	$(g_{11}, 1, 1), y_{11}$	$(g_{12}, 1, 2), y_{12}$
			$(g_{20}, 2, 0), y_{20}$	$(g_{21}, 2, 1), y_{21}$	$(g_{22}, 2, 2), y_{22}$

Doable each person computes row witnesses,

Someone then computes a column witness,

Someone then computes the row witnesses.

Fault attribution is easy.

Slashing is fairly easy, just slash main guy unless he gives name of wrong doer.

8.4 Other

A note - instead of bilinear accumulator, one can create certificates using merkle branches, and this would be quantum resistant but decrease the concrete efficiency. The security assumption of this scheme relies on the discrete log Assumption.

8.5 Strategy

We now need only find an efficient strategy for proving that 90% of block data is available.

Data Availability, State Execution and fast shuffling

The first question we should ask of the architecture is the nature of forks. Should a sharded-block-chain be fork-free?

Dfinity overview:

- Random number generation to select block maker and notarisation group
- Notarization proves that a block has been published at some time, based on a group vote.
- Notarisation is more secure than a single vote. => Finality after two rounds.
- **Kind of taking a hybrid approach pBft and block based consensus.**
- Good: **notarization in Dfinity is not primarily a validity guarantee but rather a timestamp plus a proof of publication'**
- **Good: almost fork-free; Good: still uses probabilistic consensus; Prevents long-range stake attacks**

Neo:

- Delegated Proof of Stake.
- Finality after a single round. Why based on voting.

Zilliqa:

- Sampling into shards.
- Honest majority assumotio + PBFT, much more robust.
- **Good: Uses random sampling so robust vopting groups;**
- **Bad: no slashable conditions.**

Ethereum:

- Splitting problems of availability from validity?
- Availability can be decided quickly.
- Validity decided by a kind of Trubit systle mechanism, which means validity can then be checked quickly.

Our Approach:

- Solve availability:
  - Block-chain based .. check the last 25 nodes for availability by downloading
  - dfinity based.. similar, perhaps more robust.
  - Proof based: Present a proof of data availability, i.e erasure coding a block and getting signatures from half of validators
  - Validators need only sign a hash, so the amount of work is constant* number of shards.
  - **Putting accountability into dfinity, all one needs is a numnber that gets revealed after the event.**

  - **Q: How do we put in place marginal rewards on voting**
  - **A1:** You sample a set of nodes to vote for availability, say 400; each round secret number per user, they need to compute a number based on this with their vote. ==> they have checked...
  - **a1**: Lying is just contradicti9ng the majority... study largest block reversions in bitcoin...
  - **want key shares to be revealed dfinity style.**
  - **A2: deposit into the chains to validate them, withdraw out when you can.**
- Solve validity
- truebit style execution
  - main disadvantage is the interactivity.
  - one can imagine erasure codign execution steps to find efficient fraud proofs
  - **Instead of a trubeit style verification game, one can just introduce hash checkpoints in compuitation. This reduces interactivity to 2 rounds, while keeping the amount of work done linear in the program size and**
- Ensure that consensus is fork-free, fast and robust

There is a new style file for papers submitted in 2016!

NIPS requires electronic submissions. The electronic submission site is

`https://cmt.research.microsoft.com/NIPS2016/`

Please read carefully the instructions below and follow them faithfully.

8.6 Style

Papers to be submitted to NIPS 2016 must be prepared according to the instructions presented here. Papers may only be up to eight pages long, including figures. Since 2009 an additional ninth page *containing only acknowledgments and/or cited references* is allowed. Papers that exceed nine pages will not be reviewed, or in any other way considered for presentation at the conference.

The margins in 2016 are the same as since 2007, which allow for $\sim 15\%$ more words in the paper compared to earlier years.

Authors are required to use the NIPS LaTeX style files obtainable at the NIPS website as indicated below. Please make sure you use the current files and not previous versions. Tweaking the style files may be grounds for rejection.

8.7 Retrieval of style files

The style files for NIPS and other conference information are available on the World Wide Web at

`http://www.nips.cc/`

The file `nips_2016.pdf` contains these instructions and illustrates the various formatting requirements your NIPS paper must satisfy.

The only supported style file for NIPS 2016 is `nips_2016.sty`, rewritten for LaTeX 2_ε. **Previous style files for LaTeX 2.09, Microsoft Word, and RTF are no longer supported!**

The new LaTeX style file contains two optional arguments: `final`, which creates a camera-ready copy, and `nonatbib`, which will not load the `natbib` package for you in case of package clash.

At submission time, please omit the `final` option. This will anonymize your submission and add line numbers to aid review. Please do *not* refer to these line numbers in your paper as they will be removed during generation of camera-ready copies.

The file `nips_2016.tex` may be used as a "shell" for writing your paper. All you have to do is replace the author, title, abstract, and text of the paper with your own.

The formatting instructions contained in these style files are summarized in Sections 9, 10, and 11 below.

9 General formatting instructions

The text must be confined within a rectangle 5.5 inches (33 picas) wide and 9 inches (54 picas) long. The left margin is 1.5 inch (9 picas). Use 10 point type with a vertical spacing (leading) of 11 points. Times New Roman is the preferred typeface throughout, and will be selected for you by default. Paragraphs are separated by $1/2$ line space (5.5 points), with no indentation.

The paper title should be 17 point, initial caps/lower case, bold, centered between two horizontal rules. The top rule should be 4 points thick and the bottom rule should be 1 point thick. Allow $1/4$ inch space above and below the title to rules. All pages should start at 1 inch (6 picas) from the top of the page.

For the final version, authors' names are set in boldface, and each name is centered above the corresponding address. The lead author's name is to be listed first (left-most), and the co-authors' names (if different address) are set to follow. If there is only one co-author, list both author and co-author side by side.

Please pay special attention to the instructions in Section 11 regarding figures, tables, acknowledgments, and references.

10 Headings: first level

All headings should be lower case (except for first word and proper nouns), flush left, and bold.

First-level headings should be in 12-point type.

10.1 Headings: second level

Second-level headings should be in 10-point type.

10.1.1 Headings: third level

Third-level headings should be in 10-point type.

Paragraphs There is also a `\paragraph` command available, which sets the heading in bold, flush left, and inline with the text, with the heading followed by 1 em of space.

11 Citations, figures, tables, references

These instructions apply to everyone.

11.1 Citations within the text

The `natbib` package will be loaded for you by default. Citations may be author/year or numeric, as long as you maintain internal consistency. As to the format of the references themselves, any style is acceptable as long as it is used consistently.

The documentation for `natbib` may be found at

`http://mirrors.ctan.org/macros/latex/contrib/natbib/natnotes.pdf`

Of note is the command `\citet`, which produces citations appropriate for use in inline text. For example,

```
\citet{hasselmo} investigated\dots
```

produces

Hasselmo, et al. (1995) investigated…

If you wish to load the `natbib` package with options, you may add the following before loading the `nips_2016` package:

```
\PassOptionsToPackage{options}{natbib}
```

If `natbib` clashes with another package you load, you can add the optional argument `nonatbib` when loading the style file:

```
\usepackage[nonatbib]{nips_2016}
```

As submission is double blind, refer to your own published work in the third person. That is, use "In the previous work of Jones et al. [4]," not "In our previous work [4]." If you cite your other papers that are not widely available (e.g., a journal paper under review), use anonymous author names in the citation, e.g., an author of the form "A. Anonymous."

11.2 Footnotes

Footnotes should be used sparingly. If you do require a footnote, indicate footnotes with a number[1] in the text. Place the footnotes at the bottom of the page on which they appear. Precede the footnote with a horizontal rule of 2 inches (12 picas).

Note that footnotes are properly typeset *after* punctuation marks.[2]

11.3 Figures

All artwork must be neat, clean, and legible. Lines should be dark enough for purposes of reproduction. The figure number and caption always appear after the figure. Place one line space before the figure caption and one line space after the figure. The figure caption should be lower case (except for first word and proper nouns); figures are numbered consecutively.

You may use color figures. However, it is best for the figure captions and the paper body to be legible if the paper is printed in either black/white or in color.

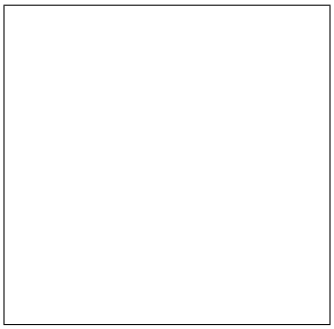

Figure 1: Sample figure caption.

11.4 Tables

All tables must be centered, neat, clean and legible. The table number and title always appear before the table. See Table 1.

Place one line space before the table title, one line space after the table title, and one line space after the table. The table title must be lower case (except for first word and proper nouns); tables are numbered consecutively.

[1] Sample of the first footnote.

[2] As in this example.

Table 1: Sample table title

	Part	
Name	Description	Size (μm)
Dendrite	Input terminal	$\sim$100
Axon	Output terminal	$\sim$10
Soma	Cell body	up to 10^6

Note that publication-quality tables *do not contain vertical rules.* We strongly suggest the use of the `booktabs` package, which allows for typesetting high-quality, professional tables:

`https://www.ctan.org/pkg/booktabs`

This package was used to typeset Table 1.

12 Final instructions

Do not change any aspects of the formatting parameters in the style files. In particular, do not modify the width or length of the rectangle the text should fit into, and do not change font sizes (except perhaps in the **References** section; see below). Please note that pages should be numbered.

13 Preparing PDF files

Please prepare submission files with paper size "US Letter," and not, for example, "A4."

Fonts were the main cause of problems in the past years. Your PDF file must only contain Type 1 or Embedded TrueType fonts. Here are a few instructions to achieve this.

- You should directly generate PDF files using `pdflatex`.
- You can check which fonts a PDF files uses. In Acrobat Reader, select the menu Files>Document Properties>Fonts and select Show All Fonts. You can also use the program `pdffonts` which comes with `xpdf` and is available out-of-the-box on most Linux machines.
- The IEEE has recommendations for generating PDF files whose fonts are also acceptable for NIPS. Please see `http://www.emfield.org/icuwb2010/downloads/IEEE-PDF-SpecV32.pdf`

  The `\bbold` package almost always uses bitmap fonts. You should use the equivalent AMS Fonts:

  ```
  \usepackage{amsfonts}
  ```

  followed by, e.g., `\mathbb{R}`, `\mathbb{N}`, or `\mathbb{C}` for $\mathbb{R}$, $\mathbb{N}$ or $\mathbb{C}$. You can also use the following workaround for reals, natural and complex:

  ```
  \newcommand{\RR}{I\!\!R} %real numbers
  \newcommand{\Nat}{I\!\!N} %natural numbers
  \newcommand{\CC}{I\!\!\!\!C} %complex numbers
  ```

  Note that `amsfonts` is automatically loaded by the `amssymb` package.

If your file contains type 3 fonts or non embedded TrueType fonts, we will ask you to fix it.

13.1 Margins in LaTeX

Most of the margin problems come from figures positioned by hand using `special` or other commands. We suggest using the command `includegraphics` from the `graphicx` package. Always specify the figure width as a multiple of the line width as in the example below:

```
\usepackage[pdftex]{graphicx} ...
\includegraphics[width=0.8\linewidth]{myfile.pdf}
```

See Section 4.4 in the graphics bundle documentation (`http://mirrors.ctan.org/macros/latex/required/graphics/grfguide.pdf`)

A number of width problems arise when LaTeX cannot properly hyphenate a line. Please give LaTeX hyphenation hints using the `\-` command when necessary.

Acknowledgments

Use unnumbered third level headings for the acknowledgments. All acknowledgments go at the end of the paper. Do not include acknowledgments in the anonymized submission, only in the final paper.

References

References follow the acknowledgments. Use unnumbered first-level heading for the references. Any choice of citation style is acceptable as long as you are consistent. It is permissible to reduce the font size to `small` (9 point) when listing the references. **Remember that you can use a ninth page as long as it contains *only* cited references.**

[1] Alexander, J.A. & Mozer, M.C. (1995) Template-based algorithms for connectionist rule extraction. In G. Tesauro, D.S. Touretzky and T.K. Leen (eds.), *Advances in Neural Information Processing Systems 7*, pp. 609–616. Cambridge, MA: MIT Press.

[2] Bower, J.M. & Beeman, D. (1995) *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System.* New York: TELOS/Springer–Verlag.

[3] Hasselmo, M.E., Schnell, E. & Barkai, E. (1995) Dynamics of learning and recall at excitatory recurrent synapses and cholinergic modulation in rat hippocampal region CA3. *Journal of Neuroscience* **15**(7):5249-5262.