A Discrete and Bounded Envy-free Cake Cutting Protocol for Any Number of Agents

Haris Aziz and Simon Mackenzie

Data61, CSIRO and UNSW

UNSW
THE UNIVERSITY OF NEW SOUTH WALES

Table of Contents

Introduction

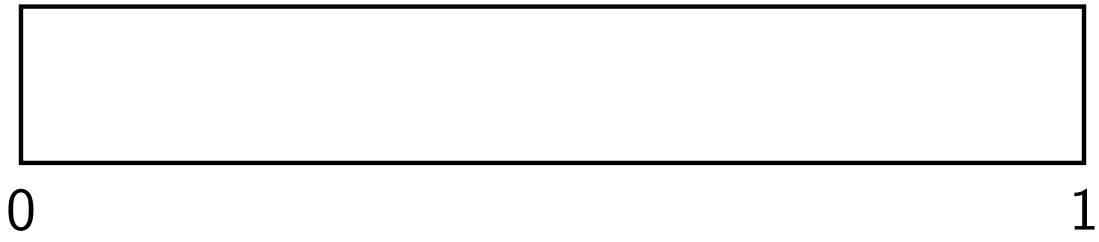
Problem Background

The New Protocol

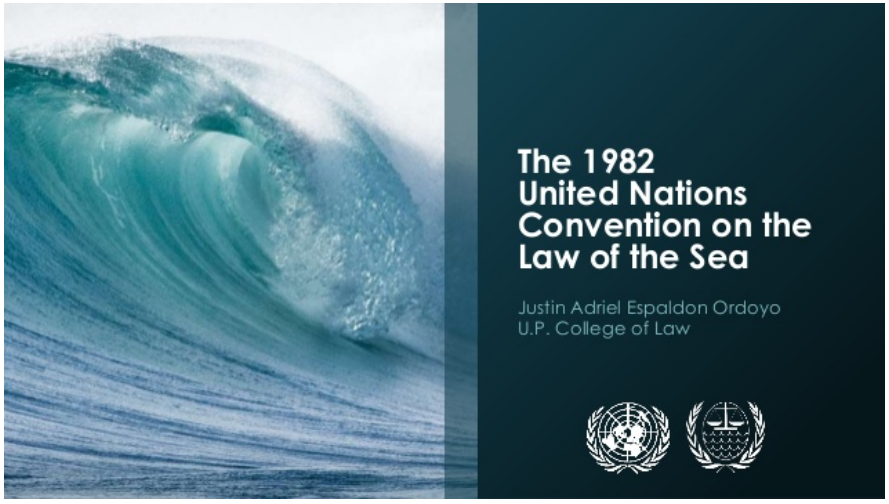
- Simple but Useful Ideas
- Overview
- Core and SubCore Protocol
- Groundwork and New Concepts
- Main Protocol
- Discrepancy Protocol
- GoLeft Protocol

Conclusion

What is cake cutting?

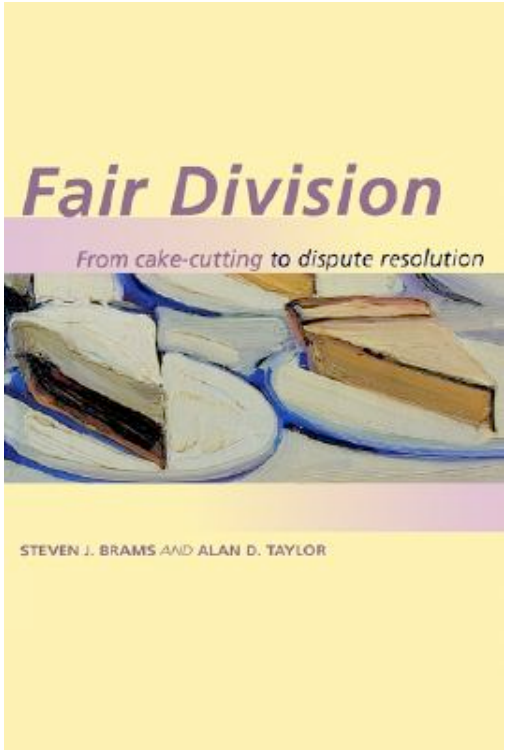


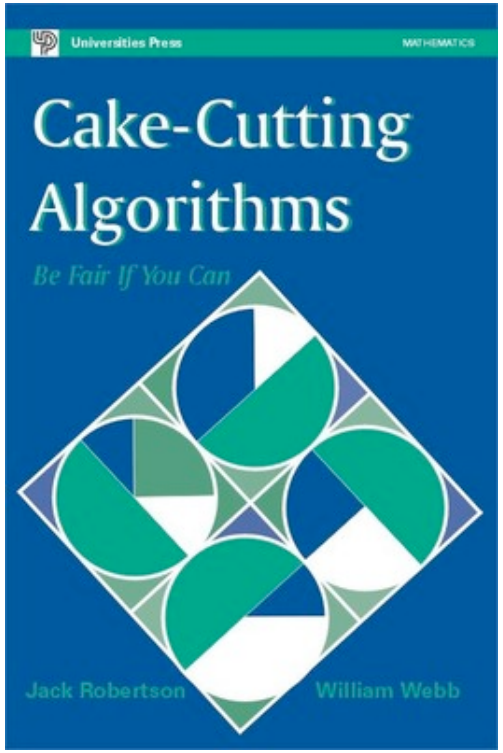
- Cake: a metaphor for a heterogeneous divisible resource (time/land/etc.)
- **Application**: resource allocation and conflict resolution



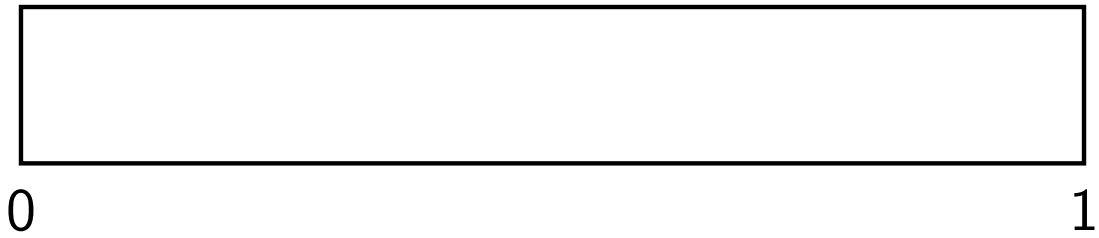
What is cake cutting?

- Cake: a metaphor for a divisible heterogeneous resource
- **Application**: fair scheduling, resource allocation, and conflict resolution
- Studied in maths, economics & computer science





Cake Cutting Model



- A **cake** which is represented by the interval $[0, 1]$.
- **Piece** of cake: union of subintervals of $[0, 1]$.
- Each agent in $N = \{1, \ldots, n\}$ has his own valuation on subsets of $[0, 1]$. The **valuations** are

  (i) **non-negative**: $V_i(X) \geq 0$;
  (ii) **additive**: for all disjoint $X, X' \subseteq [0, 1]$,

  $$V_i(X \cup X') = V_i(X) + V_i(X');$$

  (iii) **divisible** i.e., for every $X \subseteq [0, 1]$ and $0 \leq \lambda \leq 1$, there exists $X' \subset X$ with

  $$V_i(X') = \lambda V_i(X).$$

- X_i: allocation of agent i.

Identifying an Envy-free Allocation

- **Envy-free**: $V_i(X_i) \geq V_i(X_j)$ for all $i, j \in N$.
- Envy-free allocation is guaranteed to exist
- A protocol is *envy-free* if no agent is envious if he follows the protocol (irrespective of whether other agent report truthfully or not).

- **Bounded Protocol**: The number of queries required is bounded by a function in n irrespective of agent valuations.
- **Our Main Results**:
  - First discrete and bounded envy-free protocol for 4 agents. STOC 2016 — requires at most 203 cuts (now 100 cuts)
  - First discrete and bounded envy-free protocol for any number of agents. FOCS 2016.

Table of Contents

Introduction

Problem Background

The New Protocol
- Simple but Useful Ideas
- Overview
- Core and SubCore Protocol
- Groundwork and New Concepts
- Main Protocol
- Discrepancy Protocol
- GoLeft Protocol

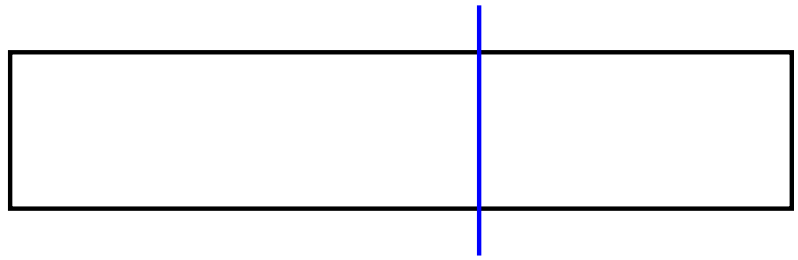
Conclusion

Divide and Choose Protocol

Divide and Choose Protocol

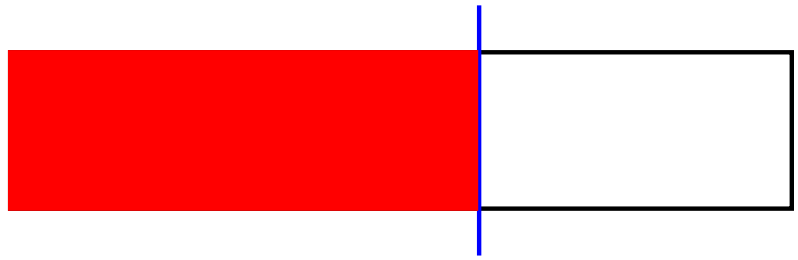
1 2

Divide and Choose Protocol

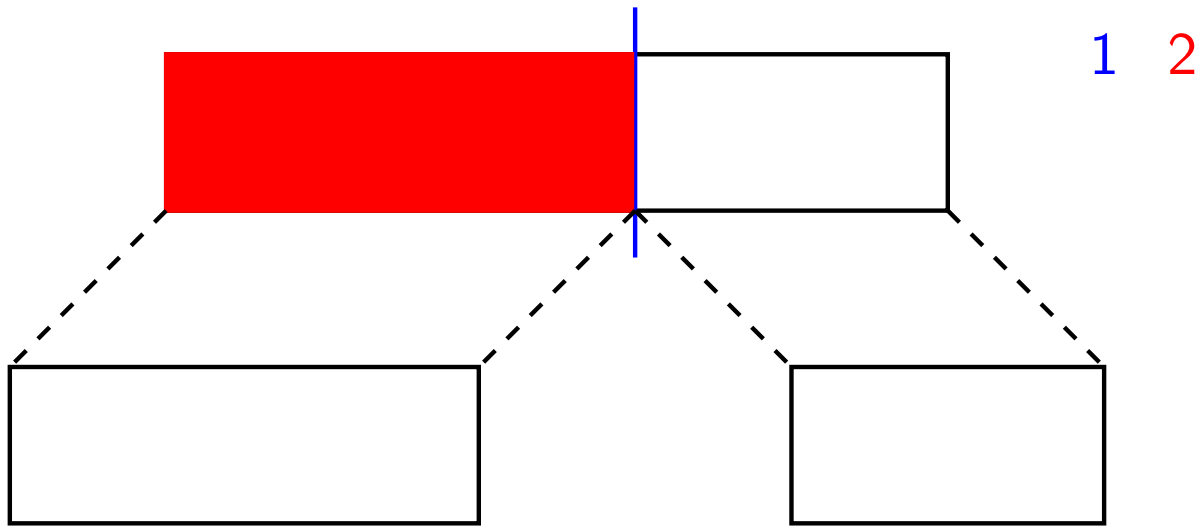


1 2

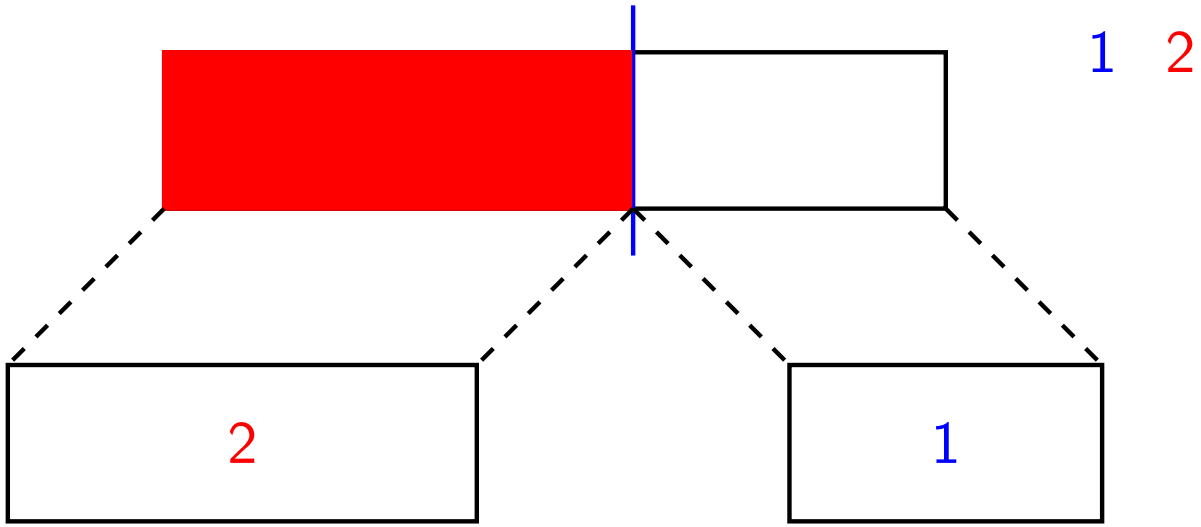
Divide and Choose Protocol



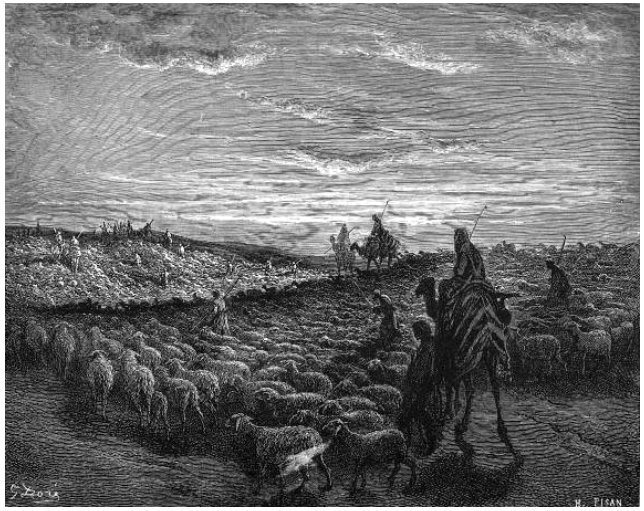
Divide and Choose Protocol



Divide and Choose Protocol



Divide and Choose Protocol



Earliest reference is in the Book of Genesis (Abraham divides the land of Canaan and Lot chooses first).

Cake cutting: Polish origins

Originally studied by Polish mathematicians in the 1940s.

Stefan Banach

Bronisław Knaster

Hugo Steinhaus

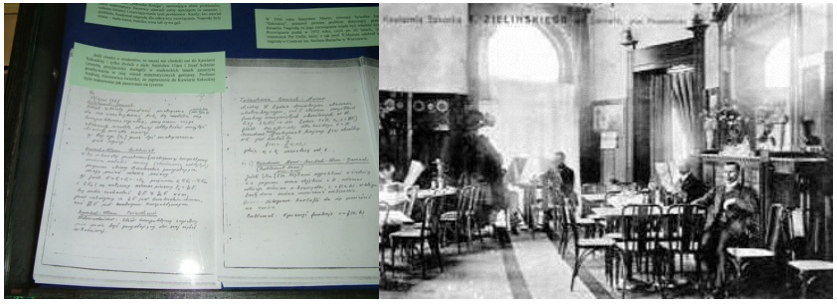
Cake cutting: Polish origins

Stefan Banach

Bronisław Knaster

Hugo Steinhaus

- Mathematicians shared ideas in the *Scottish Cafe* in Lvov.



- Steinhaus presented the cake cutting problems in 1947, at a meeting of the Econometric Society in Washington, D.C

> *Interesting mathematical problems arise if we are to determine the minimal numbers of "cuts" necessary for fair division. — Hugo Steinhaus*

Timeline: Envy-free Protocols

- **1940s**: Polish School of Maths initiated cake cutting.
- **1960s**: Selfridge-Conway procedure for 3 agents.

- **1995**: Brams and Taylor's general finite but unbounded protocol.

Timeline: Envy-free Protocols

- **1940s**: Polish School of Maths initiated cake cutting.
- **1960s**: Selfridge-Conway procedure for 3 agents.

- **1995**: Brams and Taylor's general finite but unbounded protocol.

- **2008**: Stromquist: no finite *contiguous* envy-free protocol.

Major Open Problem: Bounded Protocol for $n > 3$?

"Question 1: Is there a bounded envy-free protocol for $n = 4$ or $n > 4$?" — *Brams and Taylor [1995]*

"However, no discrete procedure with a bounded number cuts (however large) is known for four players, and such schemes probably don't exist" — *Ian Stewart [2006].*

"one of the most important open problems in the field." — *Saberi and Wang [2009]*

"even for $n = 4$, the development of finite bounded envy-free cake-cutting protocols still appears to be out of reach, and a big challenge for future research." — *Lindner and Rothe [2009]*

"Since the 1940s, the computation of envy-free cake divisions has baffled many great minds across multiple disciplines. Settling this problem once and for all is an important challenge for theoretical computer science." — *Procaccia [2013].*

Partial Allocations

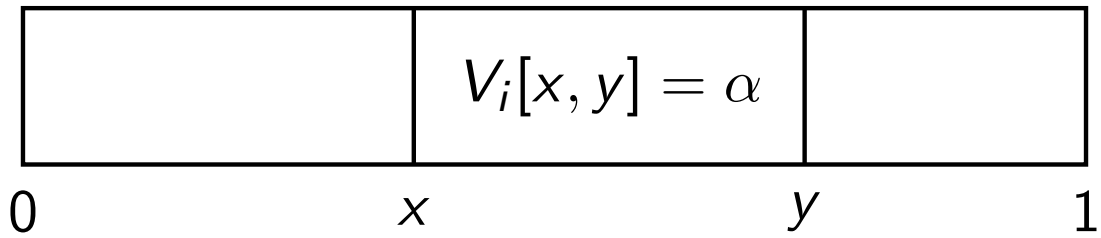
An empty allocation is envy-free but not a very good allocation. What guarantees can we give to the agents?

Even the following problem was open:

> *Does there exists a protocol that may not allocate the whole cake but gives an allocation that is envy-free and each agent get a proportional value.*

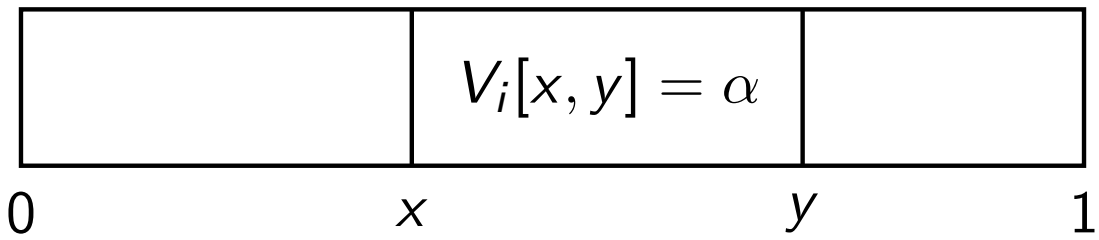
Best known result was: Bounded envy-free protocol that gives at least $1/2^{n-1}$ of value of the cake to each agent [Segal-Halevi, Hassidim, and Aumann, 2015].

Robertson and Webb Model



- Cut query: $\mathbf{Cut}_i(x, \alpha)$ asks agent i to return a point y such that $V_i([x, y]) = \alpha$.
- Evaluate query: $\mathbf{Eval}_i(x, y)$ asks agent i to return a value α such that $V_i([x, y]) = \alpha$.

Robertson and Webb Model



- Cut query: $\textbf{Cut}_i(x, \alpha)$ asks agent i to return a point y such that $V_i([x, y]) = \alpha$.
- Evaluate query: $\textbf{Eval}_i(x, y)$ asks agent i to return a value α such that $V_i([x, y]) = \alpha$.

Bounded number of Robertson & Webb queries implies bounded number of cuts

Outline

Introduction

Problem Background

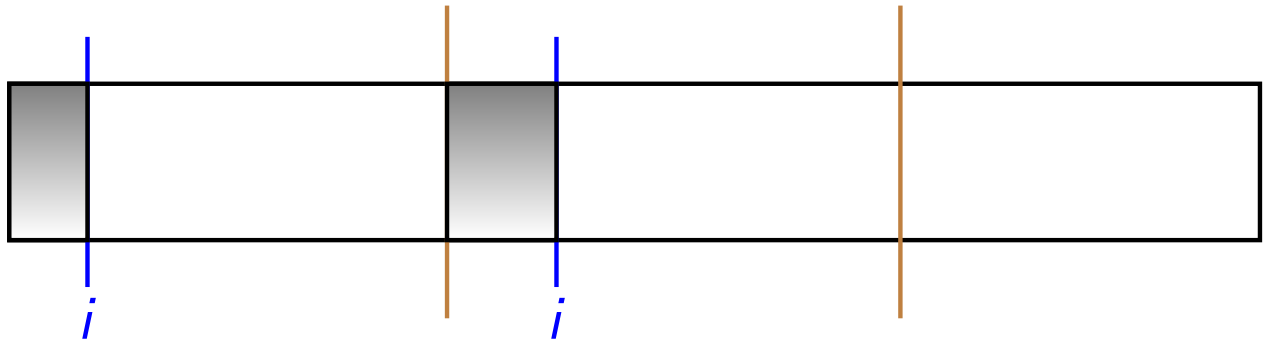
The New Protocol

- Simple but Useful Ideas
- Overview
- Core and SubCore Protocol
- Groundwork and New Concepts
- Main Protocol
- Discrepancy Protocol
- GoLeft Protocol

Conclusion

Useful ideas

- **Trimming**:



Useful ideas

- **Domination:** Given an envy-free partial allocation X of the cake and an unallocated residue R, we say that agent i dominates agent j if i does not become envious of j even if all of R were to be allocated to j.

$$V_i(X_i) \geq V_i(X_j) + V_i(R).$$

Figure: Agent i dominates agent j

Useful ideas

- **Domination:** Given an envy-free partial allocation of the cake and an unallocated residue R, we say that agent i dominates agent j if i does not become envious of j even if all of R were to be allocated to j.

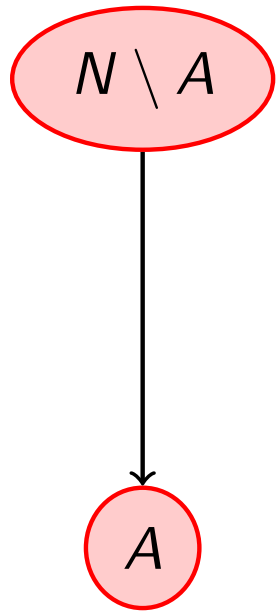
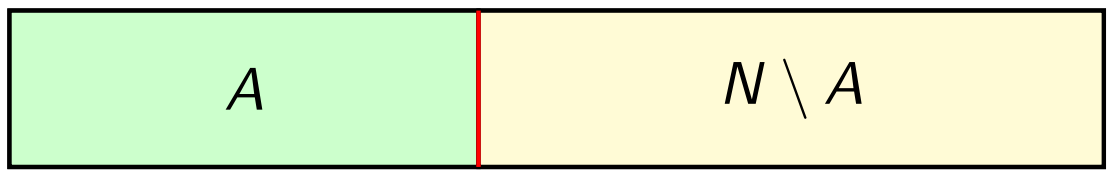


Figure: If agents in $N \setminus A$ dominate agents in A, we can simply allocate the residue among agents in A.

Useful ideas

- **Exploiting Discrepancy + Envy-freeness implies Proportionality** (each agent gets at least $1/n$ value of the cake in an envy-free allocation)



  - Agents in A think that green part has n times more value than yellow part
  - Agents in $N \setminus A$ think yellow part has n times more value than green part

Outline

Introduction

Problem Background

The New Protocol

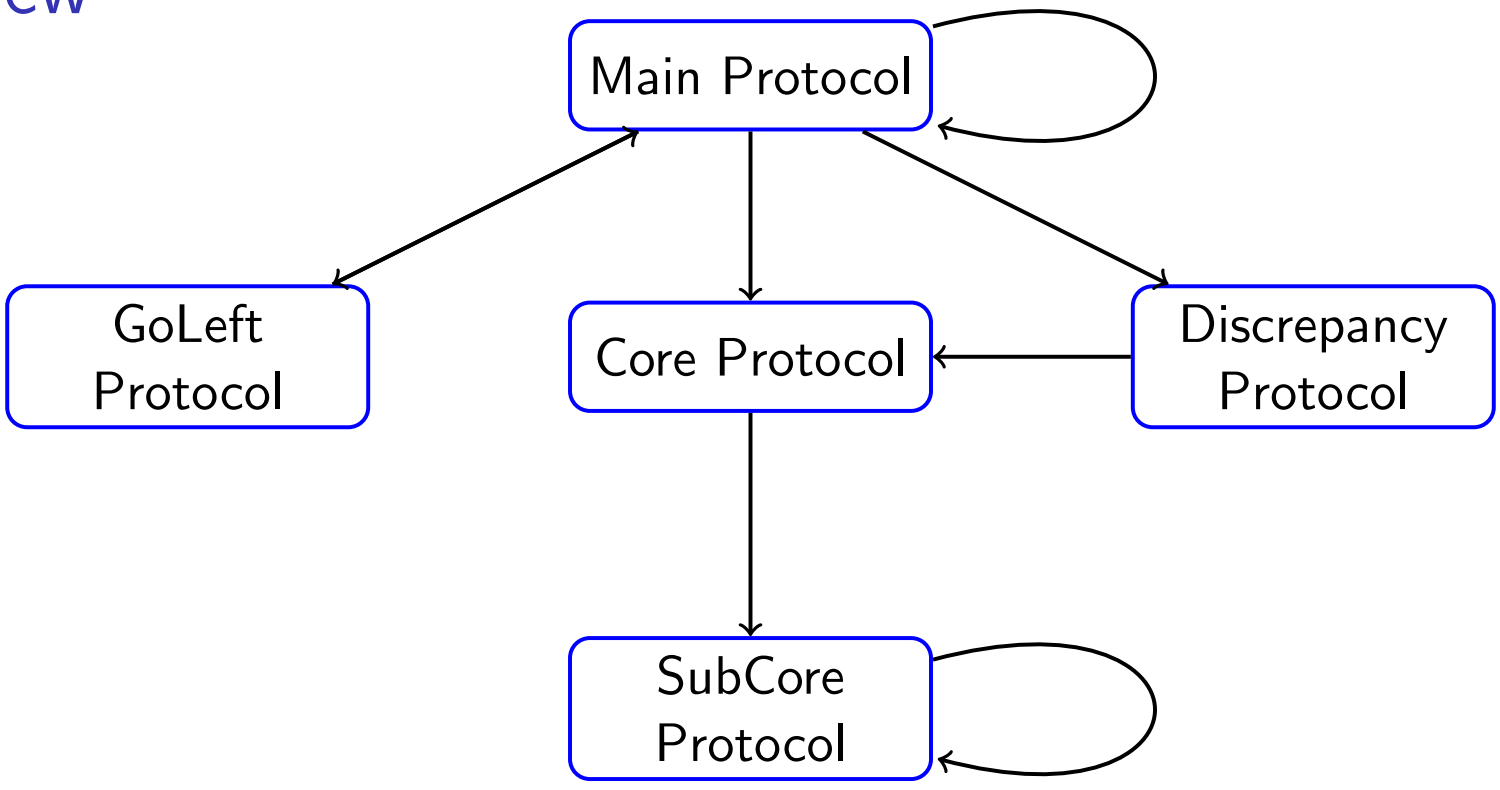
- Simple but Useful Ideas
- Overview
- Core and SubCore Protocol
- Groundwork and New Concepts
- Main Protocol
- Discrepancy Protocol
- GoLeft Protocol

Conclusion

Algorithm in a Nutshell

- Get an envy-free partial allocation
- From the unallocated cake, gradually give agents some cake while maintaining envy-freeness
- Get some agents to dominate all others which reduces the problem to less number of agents.

Overview



- **Main Protocol**: is called to get an envy-free allocation
- **Core Protocol**: workhorse of the overall protocol (makes the residue smaller)
- **Discrepancy Protocol**: exploits huge discrepancy
- **GoLeft Protocol**: allows agents to exchange pieces by additionally attaching insignificant crumbs from the residue.

Overview

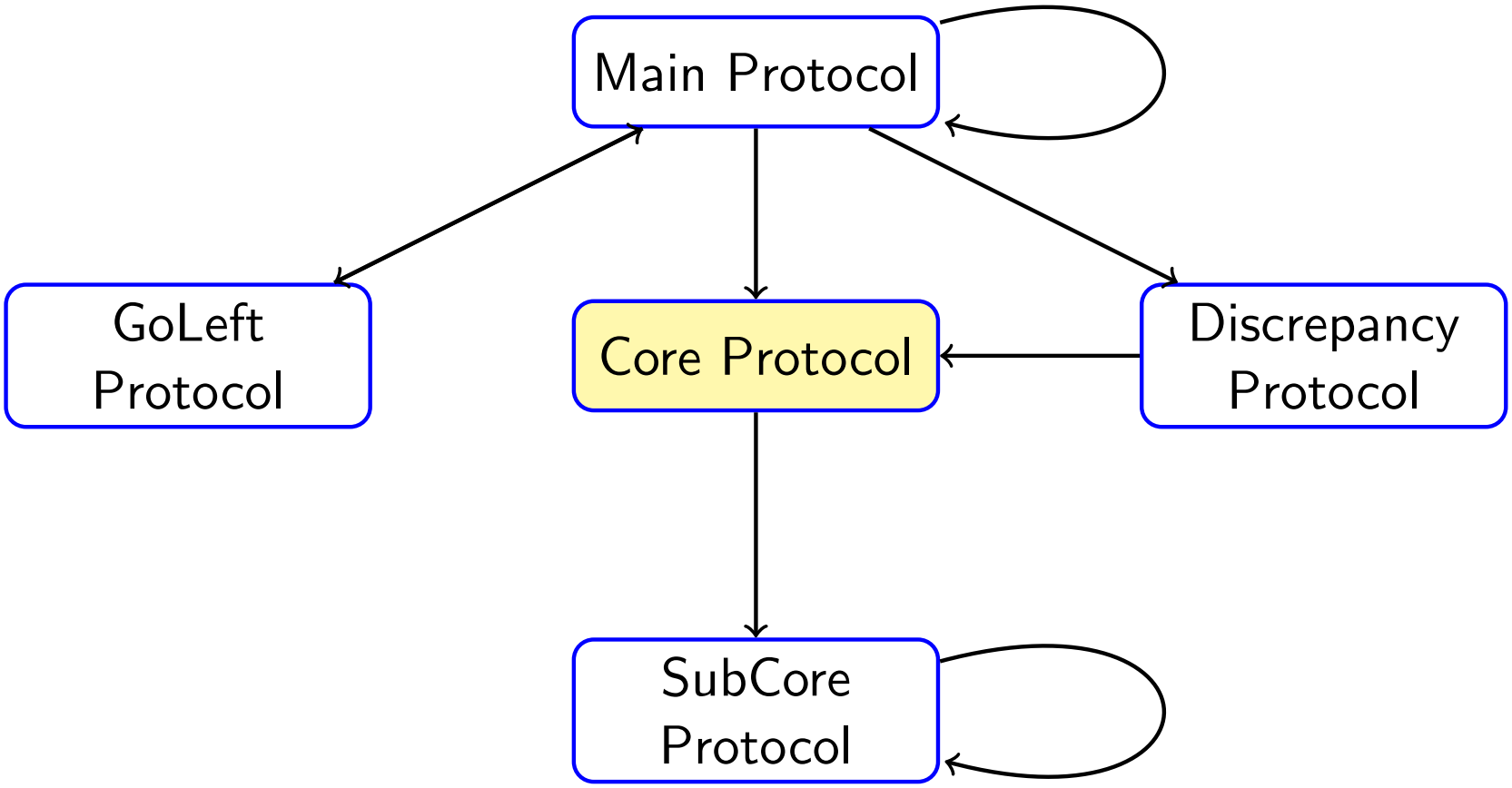


Figure: The envy-free protocol for n agents relies on various protocols. A protocol points to another protocol if it calls the other one. A protocol has a self-loop if it calls itself recursively.

Outline

Introduction

Problem Background

The New Protocol

- Simple but Useful Ideas
- Overview
- Core and SubCore Protocol
- Groundwork and New Concepts
- Main Protocol
- Discrepancy Protocol
- GoLeft Protocol

Conclusion

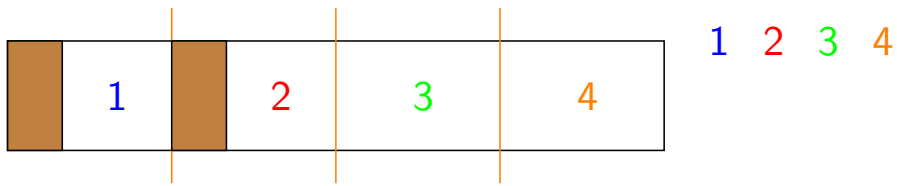
Core Protocol

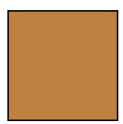
Input: Specified cutter (say agent n) and agent set N and unallocated cake R

Output: An envy-free partial allocation for agents in N.

1: Ask agent n to cut the cake into n equally preferred pieces.
2: Run SubCore Protocol on the n pieces with agents set $N \setminus \{n\}$.
3: Given n one of the unallocated untrimmed pieces.
4: **return** Partial Envy-free Cake for agents in N as well as the unallocated cake.

- Each agent gets a part of exactly one of the pieces and no other pieces.
- Cutter and one more agent gets a full piece.
- From the cutter's perspective at least $2/n$ value of the cake is allocated.





SubCore Protocol

Input: Cake cut into n pieces to be allocated among set $N' \subset N$ with $n' = |N'|$

Output: An envy-free partial allocation for agents in N'.

1: Take permutation $1, 2, 3, \ldots, n'$ as the order of agents in N'
2: **for** $m = 1$ to n' **do**
3: we ask the m-th agent his most preferred piece among the n pieces.
4: **if** his most preferred piece is an unallocated piece **then**
5: we are already done as we can give the m-th agent that piece and go to next iteration of for loop
6: **else** Need to do more work
7: **return** Partial Envy-free Cake for agents in N' as well as the unallocated cake.

SubCore Protocol: More Work including Recursion

1: The first m agents are contesting for the same $m-1$ tentatively allocated 'contested pieces' Then each agent in $[m]$ is asked to trim pieces among the $m-1$ tentatively allocated pieces that are of higher value than his most preferred piece outside the $m-1$ allocated pieces so that value of the former is same as the value of the latter.

2: Set W to be the set of agents who trimmed most (had the rightmost trim) in some piece.

3: **while** $|W| < m-1$ **do**

4: Ignore the previous trims of agents in W from now on and forget the previous allocation.

5: Run SubCore Protocol on all of the n pieces with W as the target set of agents and for each piece, the left side of the right-most trim by an agent in $[m] \setminus W$ is ignored.

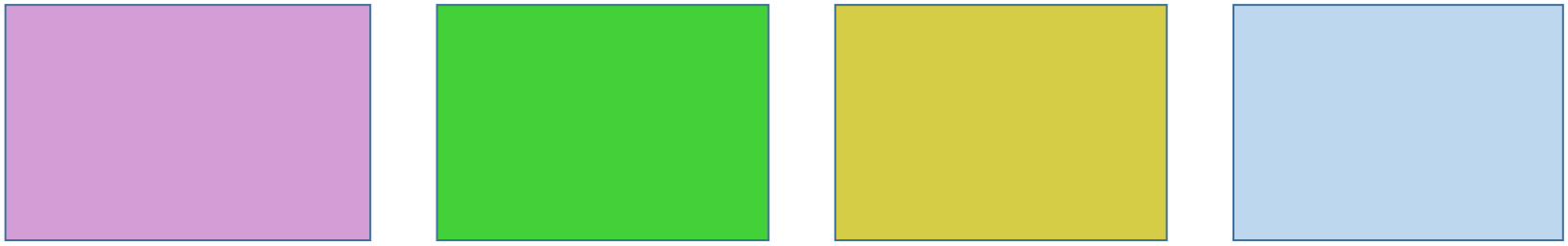
6: The result of SubCore is an allocation that gives a (partial) contested piece to each of the agents in W. Take any unallocated contested piece a for which the current left margin (beyond which the piece is ignored) is by agent $i \in [m] \setminus W$.

$$W \longleftarrow W \cup \{i\}.$$

End While

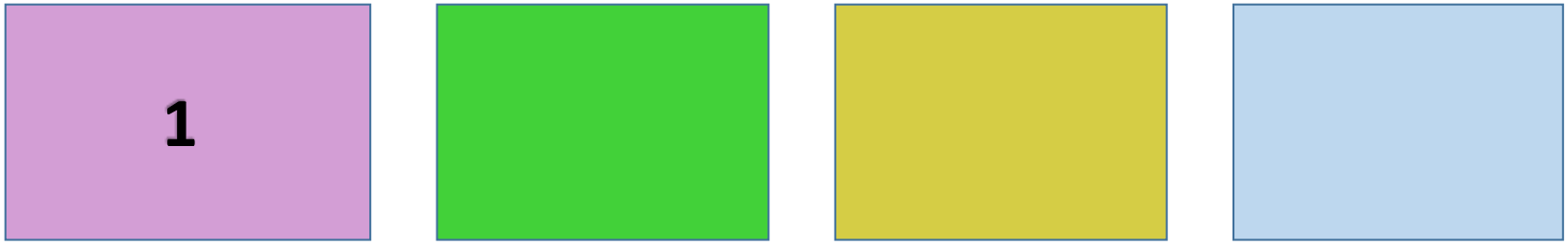
7: Each agent in W can get the right hand side of the piece that he trimmed most. RunSubCore with W and the contested pieces while ignoring the LHS of trim of only agent in $[m] \setminus W$. The only agent remaining in $[m] \setminus W$ can be given the most preferred uncontested piece.

Core & SubCore: Example



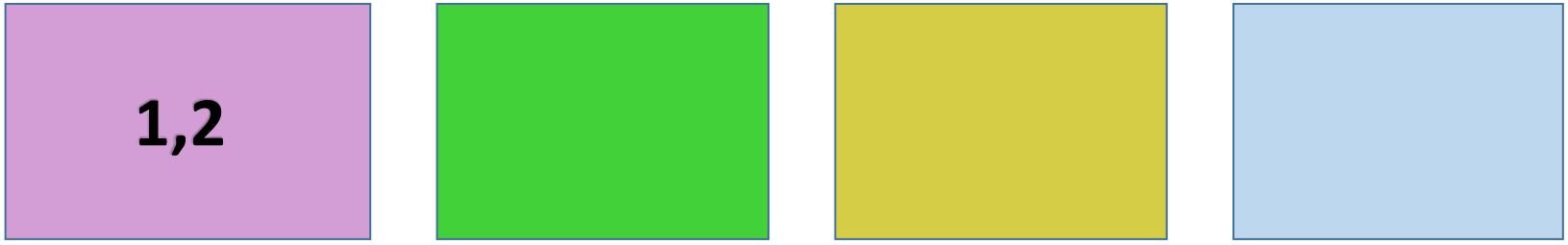
Agent 4 (cutter) divides the cake into 4 equally preferred pieces.
Next, SubCore is called to allocate the pieces among $1, 2, 3$.

Core & SubCore: Example



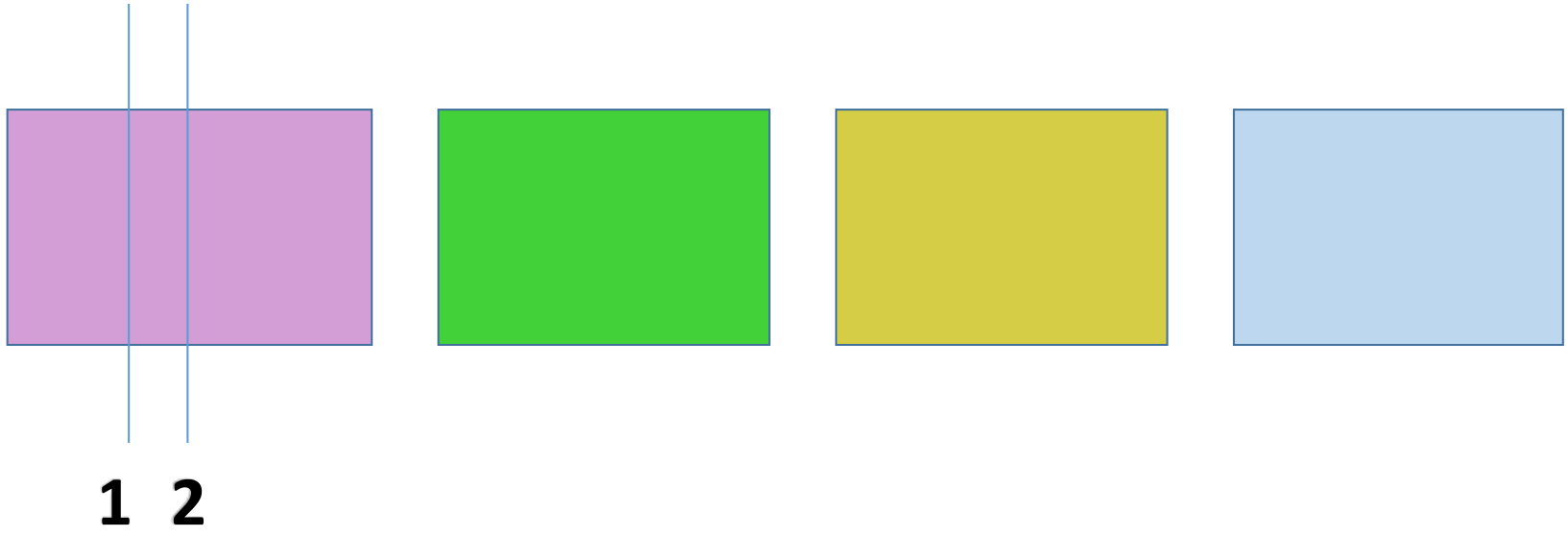
Agent 1 takes his most preferred piece (tentatively).

Core & SubCore: Example



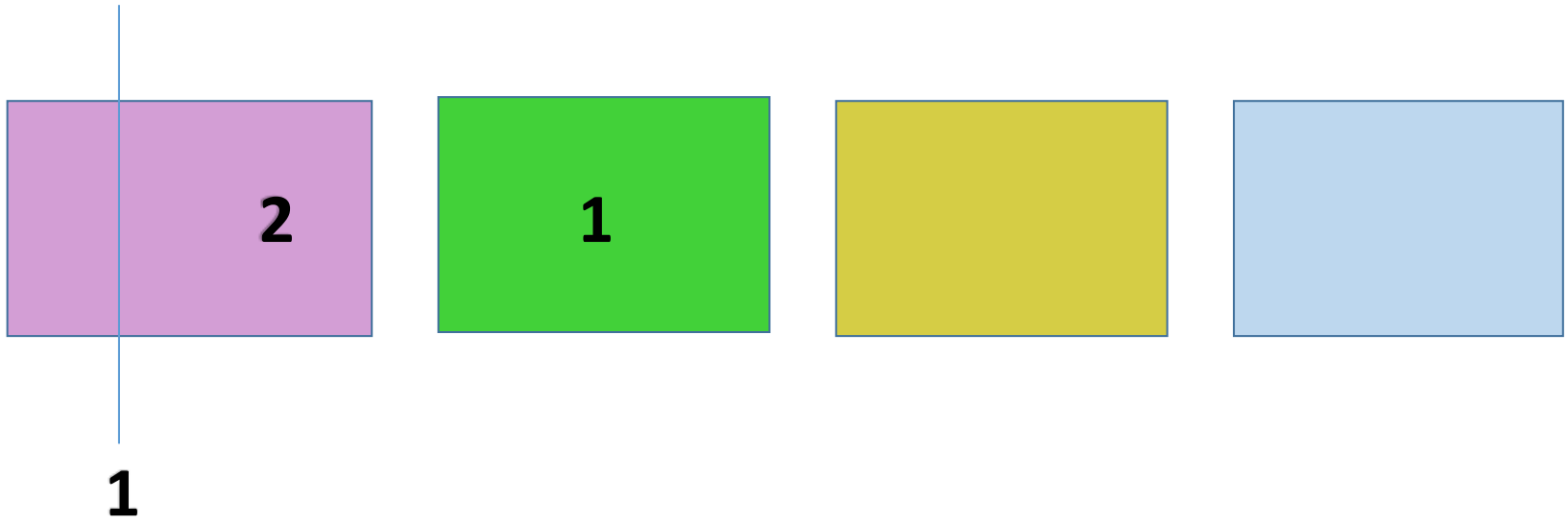
But agent 2 most prefers the same piece.

Core & SubCore: Example



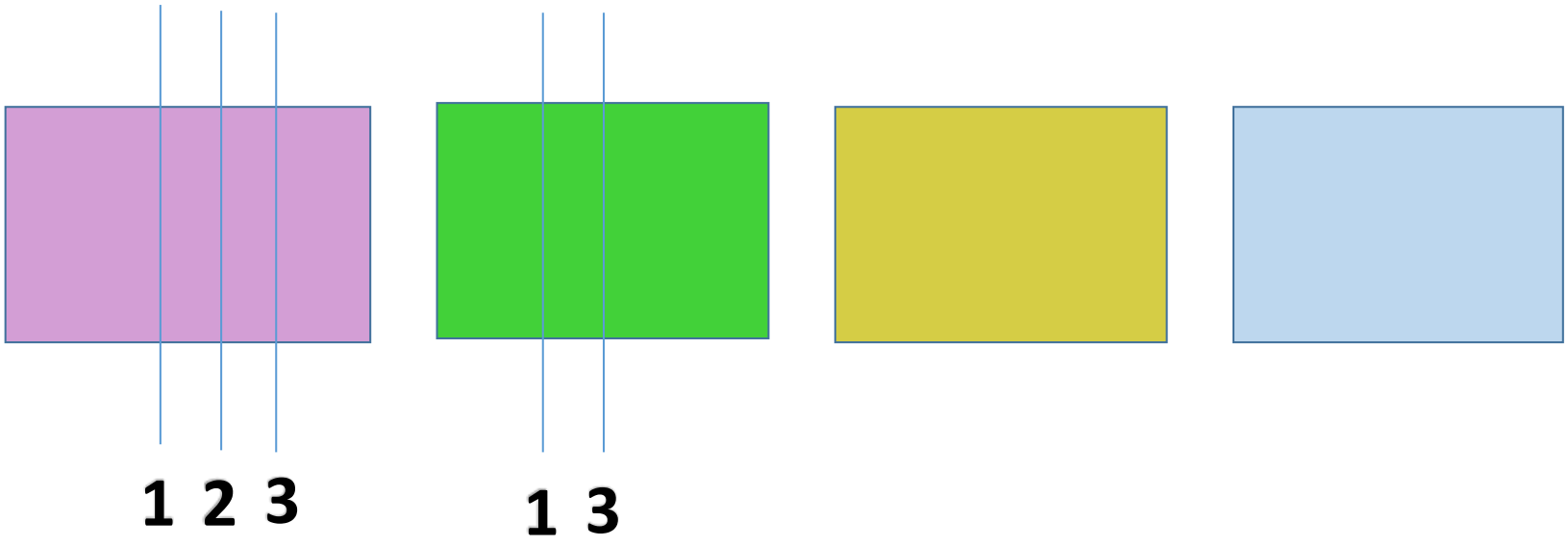
Both 1 and 2 trim the most preferred piece to the value of next most preferred piece.

Core & SubCore: Example



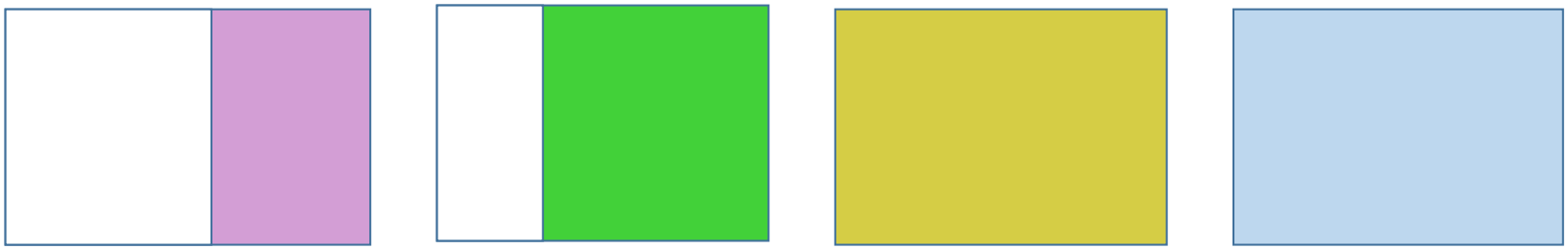
Agent 2 gets the contested piece because he trimmed more and 1 gets second most preferred piece.
Next we want to handle agent 3.

Core & SubCore: Example



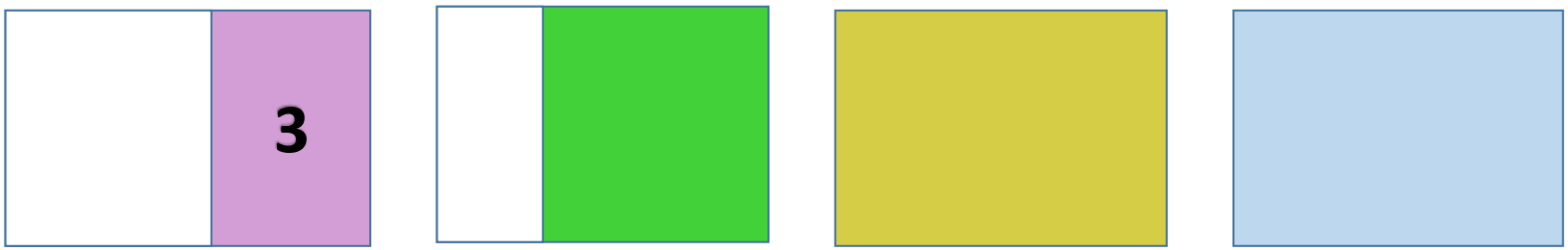
Agent 3 win in both pieces. $W = \{3\}$. We want to increase W until $|W| = 2$. Next, we ignore the left side of the right most non-winner's trim in each contested pieces.

Core & SubCore: Example



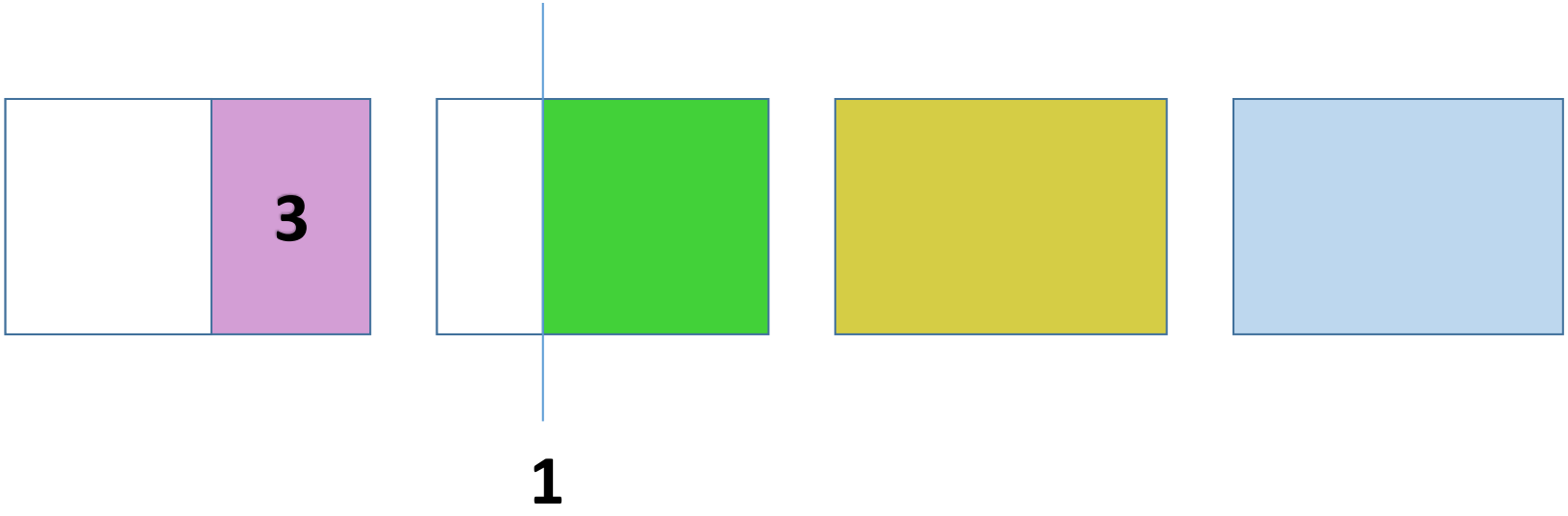
Next we call SubCore with $W = \{3\}$.

Core & SubCore: Example



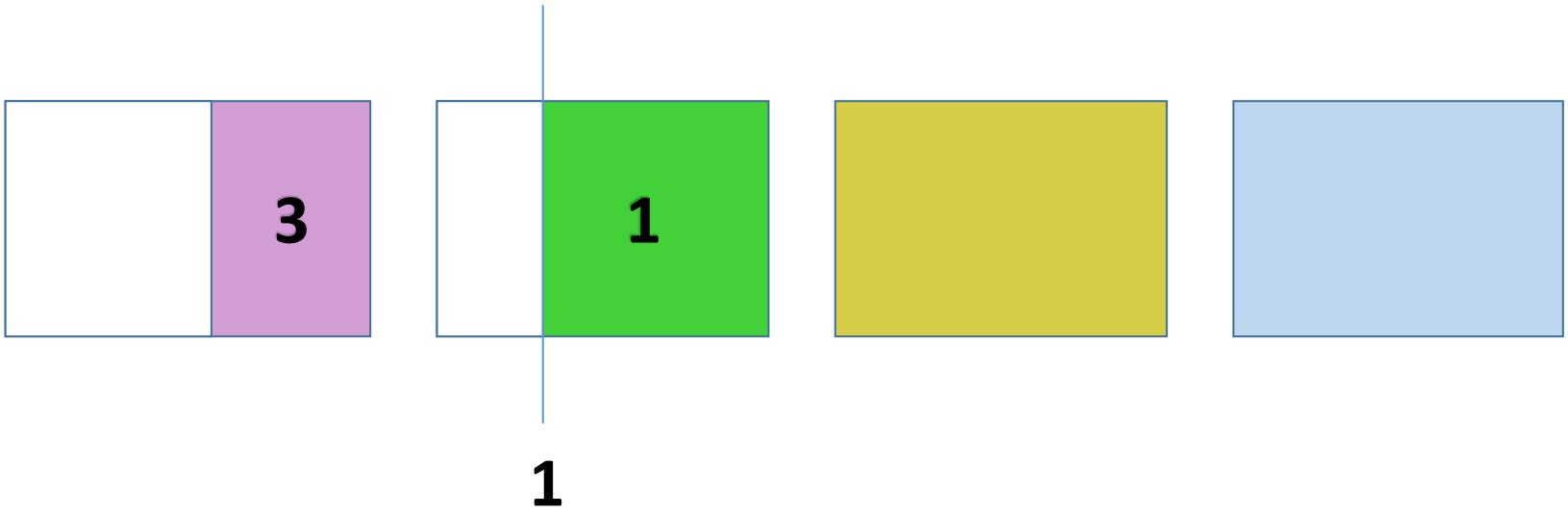
The recursive call of the SubCore results in 3 getting the pink piece. There exists a contested piece whose current left margin coincides with the one of the non-winner's trim.

Core & SubCore: Example



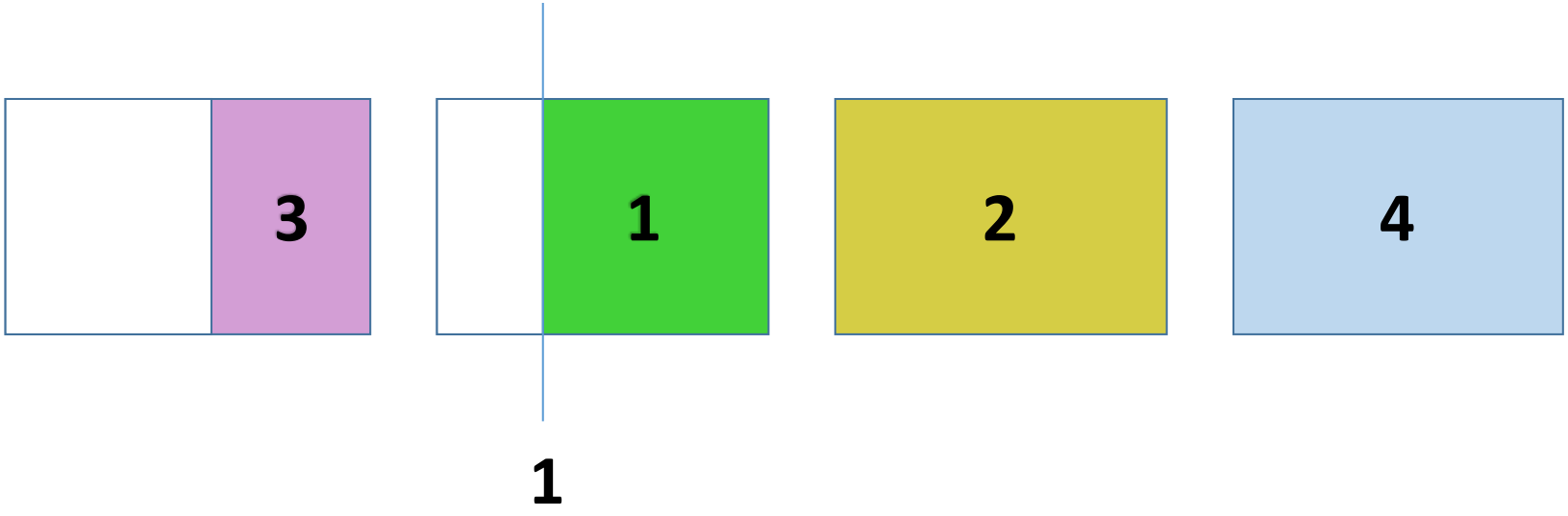
That non-winner is agent 1.

Core & SubCore: Example



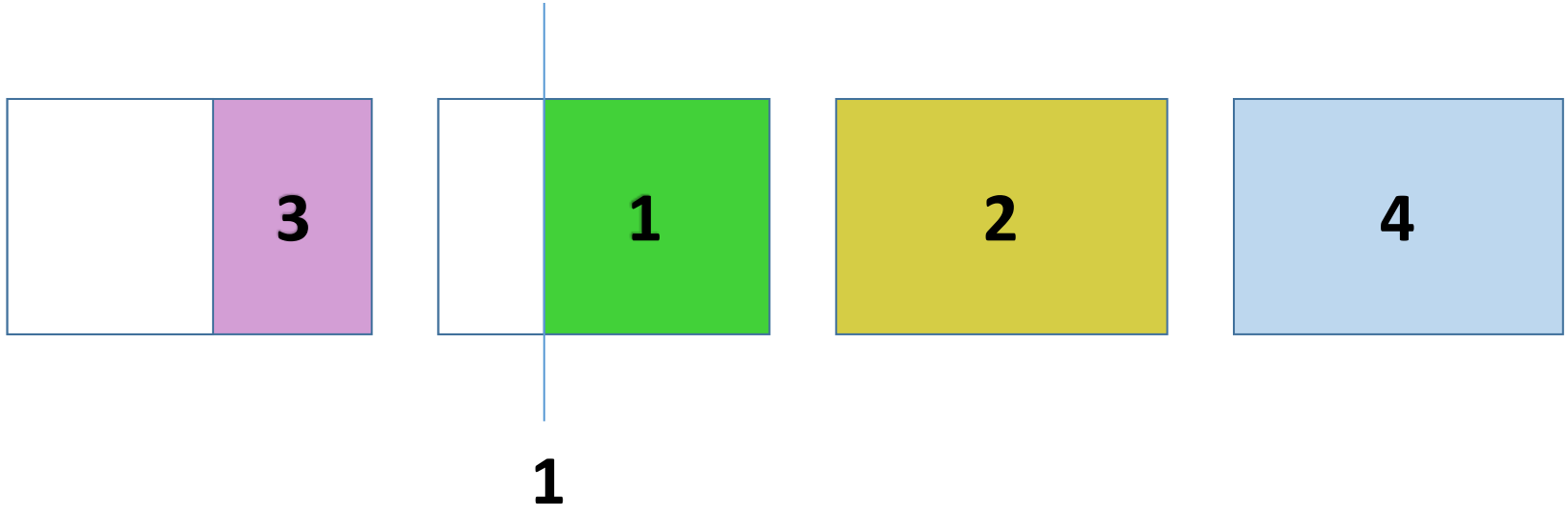
Agent 1 gets the green piece. At this point the set of contested piece (pink and green) are allocated whereas the uncontested piece have not been trimmed. We ask agent 2 to get his most preferred uncontested piece.

Core & SubCore: Example



Agent 2 picks his most preferred unallocated piece.

Core & SubCore: Example



Then agent 4 the cutter picks the last remaining piece.

A Proportional and Envy-free Protocol!

Consider the following protocol that may not allocate the whole cake:

> *Run Core Protocol on the cake with an agent who has not acted as cutter. If there is some cake left, repeat.*

The protocol is envy-free and proportional wrt the whole cake! It takes $n^3(n^2)^n$ queries.

Argument for proportionality:

- When we call the Core Protocol n times each time with a different cutter, each agent gets at least $1/n$ of the value of the cake allocated in that call.
- Moreover when an agent himself is the cutter, he gets at least $1/n$ of the value of the remaining cake.

Concepts: Core Snapshots

Definition (Snapshots)

When we run the Core Protocol we end up with a partial envy-free allocation of the cake and a residue. We call the partial allocation a *snapshot*.

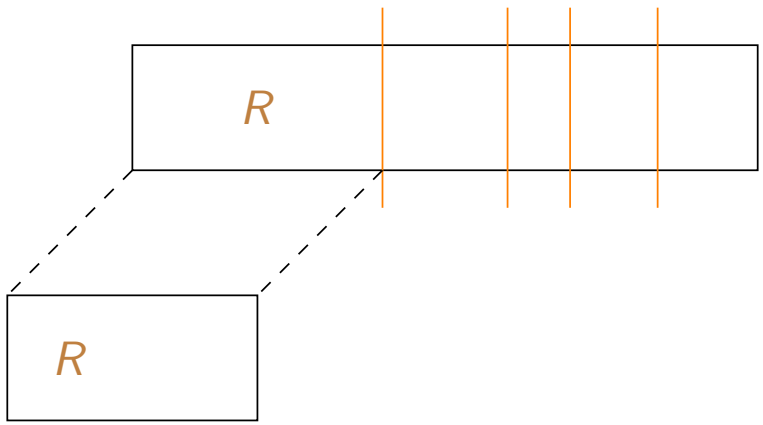


Figure: Core Snapshot.

Core & SubCore: Recap

- We have an envy-free Core protocol that takes $(n^2)^{n+1}$ queries and allocates at least $2/n$ of the cake from the cutter's perspective.
- Repeating it gives no guarantee that the whole cake will be allocated even in finite time.
- Need to do more work.

Outline

Introduction

Problem Background

The New Protocol

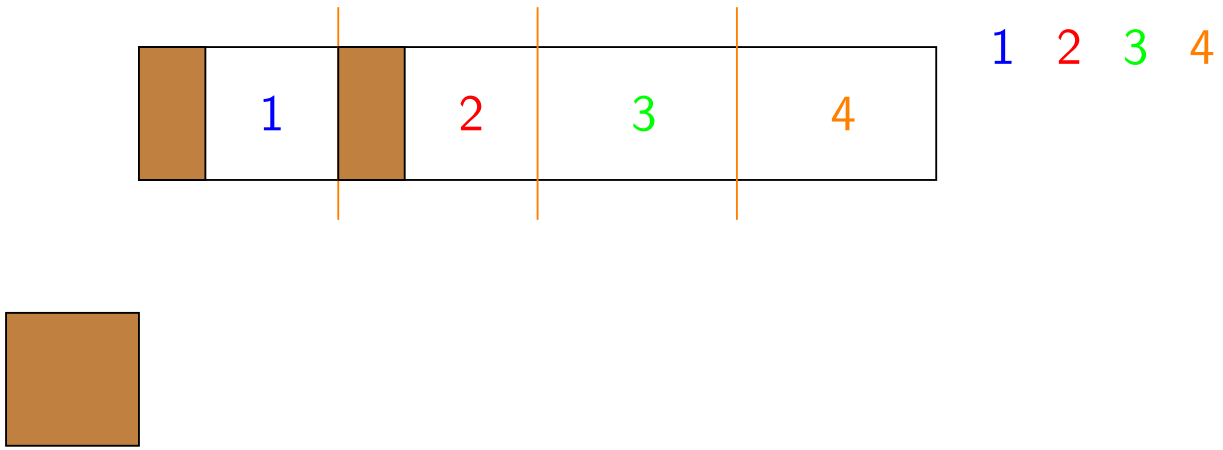
- Simple but Useful Ideas
- Overview
- Core and SubCore Protocol
- Groundwork and New Concepts
- Main Protocol
- Discrepancy Protocol
- GoLeft Protocol

Conclusion

Concepts: Significant Value

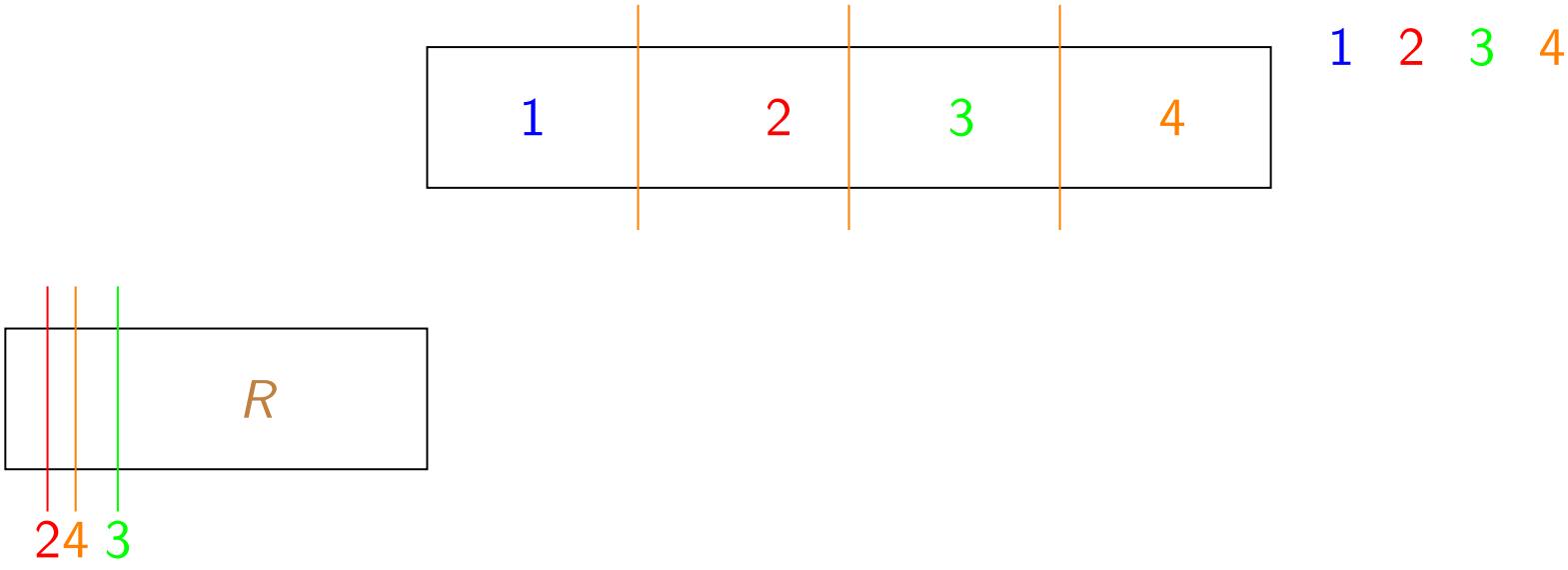
Definition (Significant value)

An agent i thinks a value is *significant* if the value is $\geq V_i(R)(\frac{n-2}{n})^B$ where R is the unallocated residue. A piece is *significant* for an agent if it has significant value.

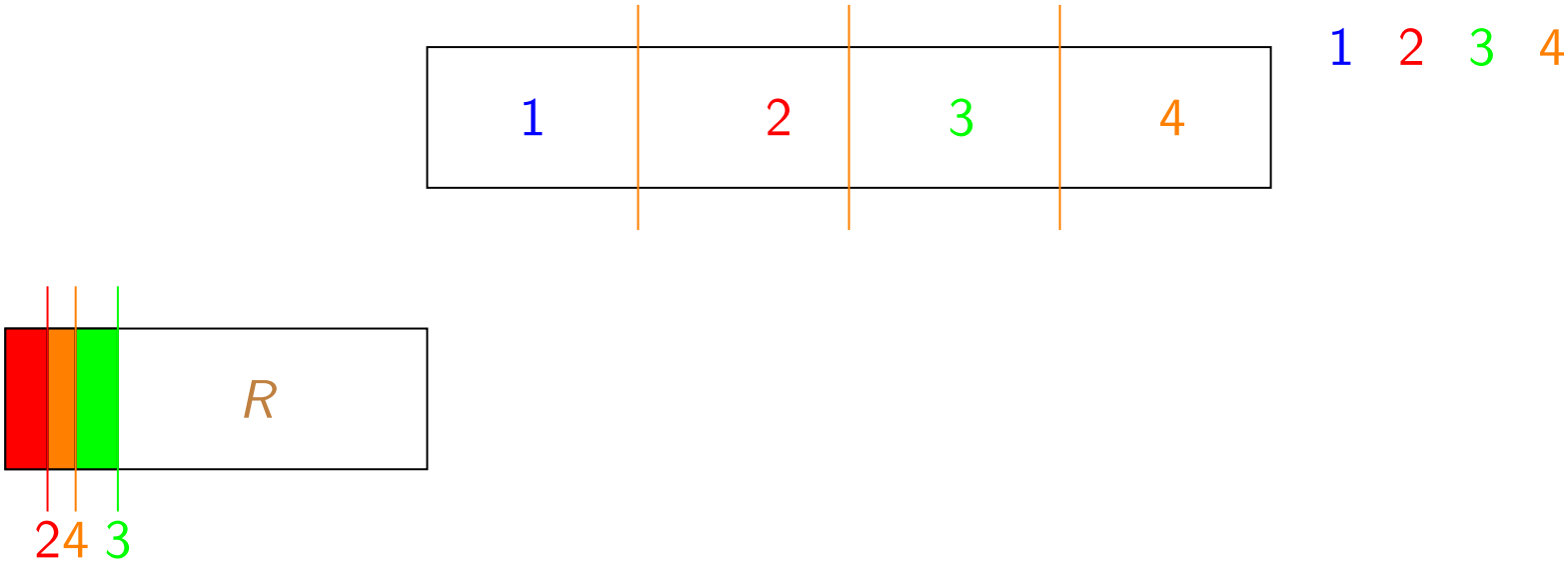


- If an agent i has significant bonus over another agent j, then we can make the residue smaller than the bonus in bounded steps which means that i dominates j.
- A cutter has a significant bonus over at least one other agent.

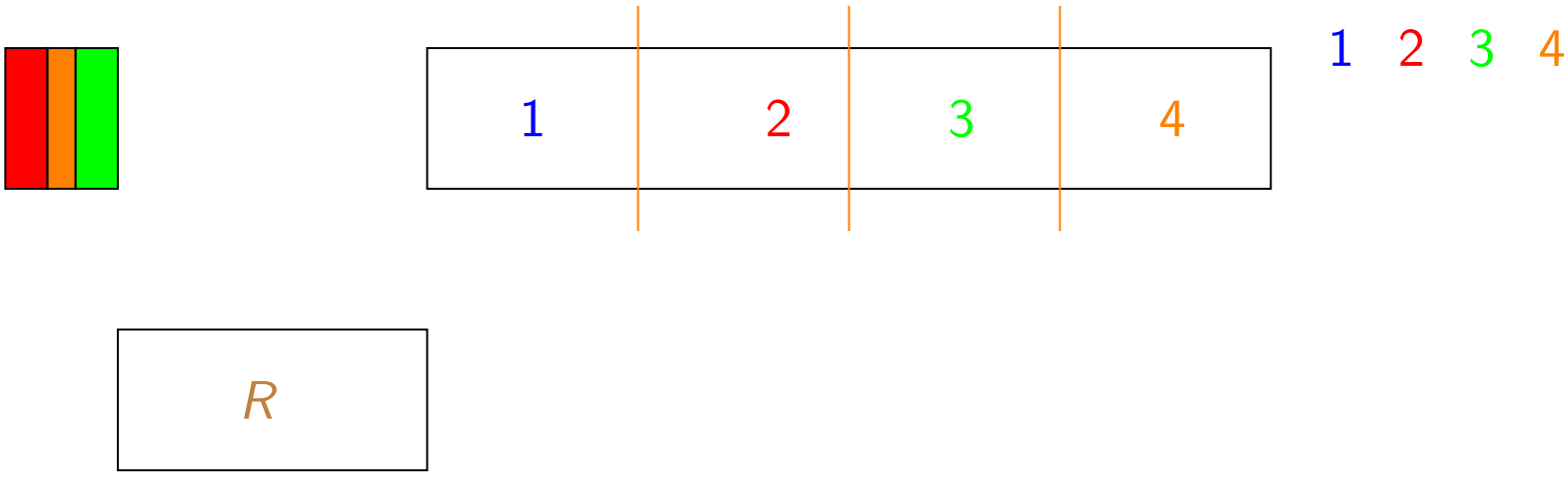
Concepts: Extraction



Concepts: Extraction

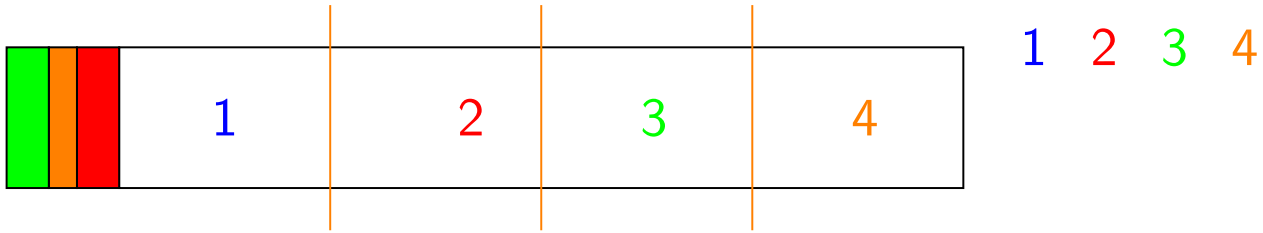


Concepts: Extraction



- we only extract if the pieces are insignificant for every agent (we want to have most of the residue unused for further operations)

Concepts: Extraction and Attachment



- The extracted pieces are attached one by one in the GoLeft protocol. As we attach another piece, we are ‘going left’.

Concepts: Isomorphic Snapshots

Definition (Isomorphic snapshots/pieces of cake)

We call a set of snapshots **isomorphic** to each other if for each piece c_i in the snapshot allocated to agent i, the agents who extracted cake from the residue and associated to c_i are the same and did so in the same order.

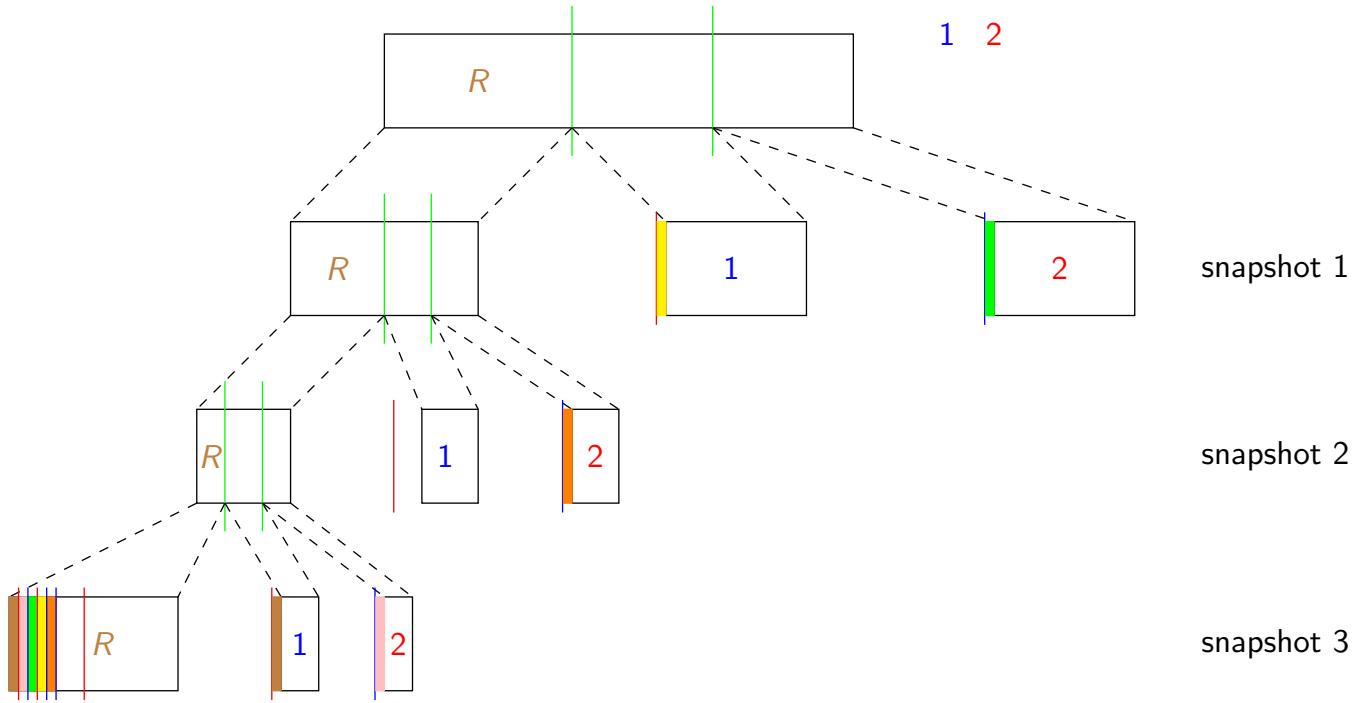


Figure: Snapshots with extracted pieces. Snapshot 1 and 3 are isomorphic. In the second snapshot a piece was not extracted because agent 2's bonus value was significant.

Concepts: Bounds

$$C \ll C' \ll B \ll B'.$$

- $C'n$ is the number of snapshots generated and labelled by the main protocol
- When we run GoLeft we look at a subset of Cn isomorphic snapshots from the $C'n$ total snapshots.
- The value B' corresponds to the total number of queries required to run the whole protocol.
- B is used to define what we call significant pieces or values.

$C = n^{n^n}$; $C' = n^{n^{n^n}}$; $B = n^{n^{n^{n^n}}}$; $B' = n^{n^{n^{n^{n^n}}}}$.

Enforcing a Gap

For any $f(n) > 1$, if an agent thinks that a piece is of value of more than $1/f(n)$ of the significance threshold but less than $f(n)$ times the significance threshold, we continue calling Core protocol until the piece's value is $f(n)$ times the significance threshold.

Outline

Introduction

Problem Background

The New Protocol

- Simple but Useful Ideas
- Overview
- Core and SubCore Protocol
- Groundwork and New Concepts
- Main Protocol
- Discrepancy Protocol
- GoLeft Protocol

Conclusion

Overview

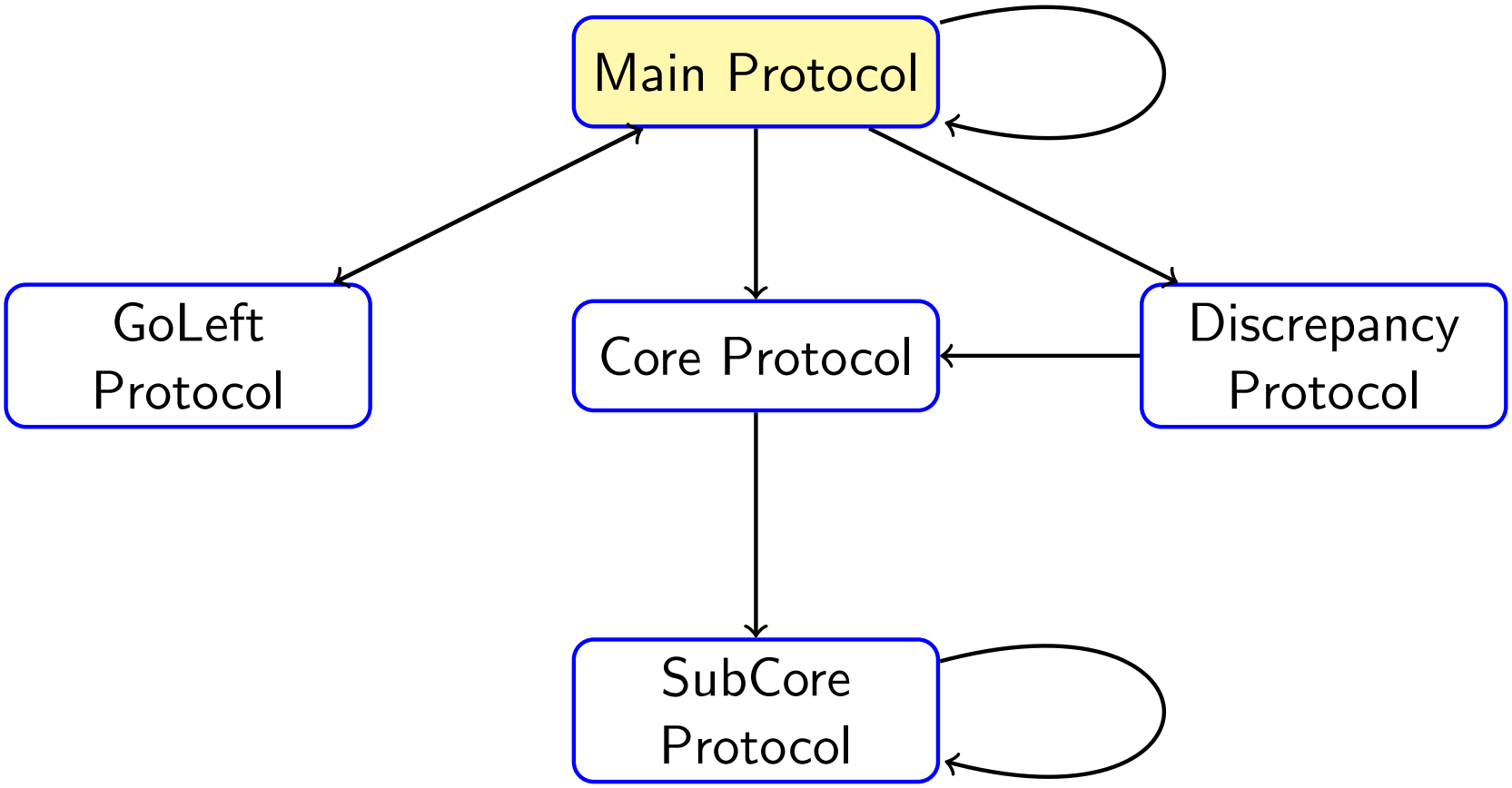


Figure: The envy-free protocol for n agents relies on various protocols. A protocol points to another protocol if it calls the other one. A protocol has a self-loop if it calls itself recursively.

Main Protocol

Input: Cake R and a set of agents N.
Output: An envy-free allocation

1: **BASE CASE**: If $|N| \leq 4$, use a known bounded EF protocol
2: **CORE**: Generate core snapshots.
3: **EXTRACTION**: Start extracting pieces from the residue corresponding to the core snapshot pieces
4: **DISCREPANCY**: call Discrepancy if there is some piece to be extracted over which there is discrepancy. If there is real discrepancy, then some agents D go for the discrepant piece, others D' go for the residue! (Call Main Protocol).
5: **GOLEFT**: Call GoLeft
6: Call Core Protocol $2B$ times to change significant bonuses into dominance
7: Call Main Protocol(R, $A \subset N$), that is the Main Protocol on a subset of agents $A \subset N$ output by the GoLeft Protocol and with R as the input cake
8: **return** allocation of the cake to the agents

Overview

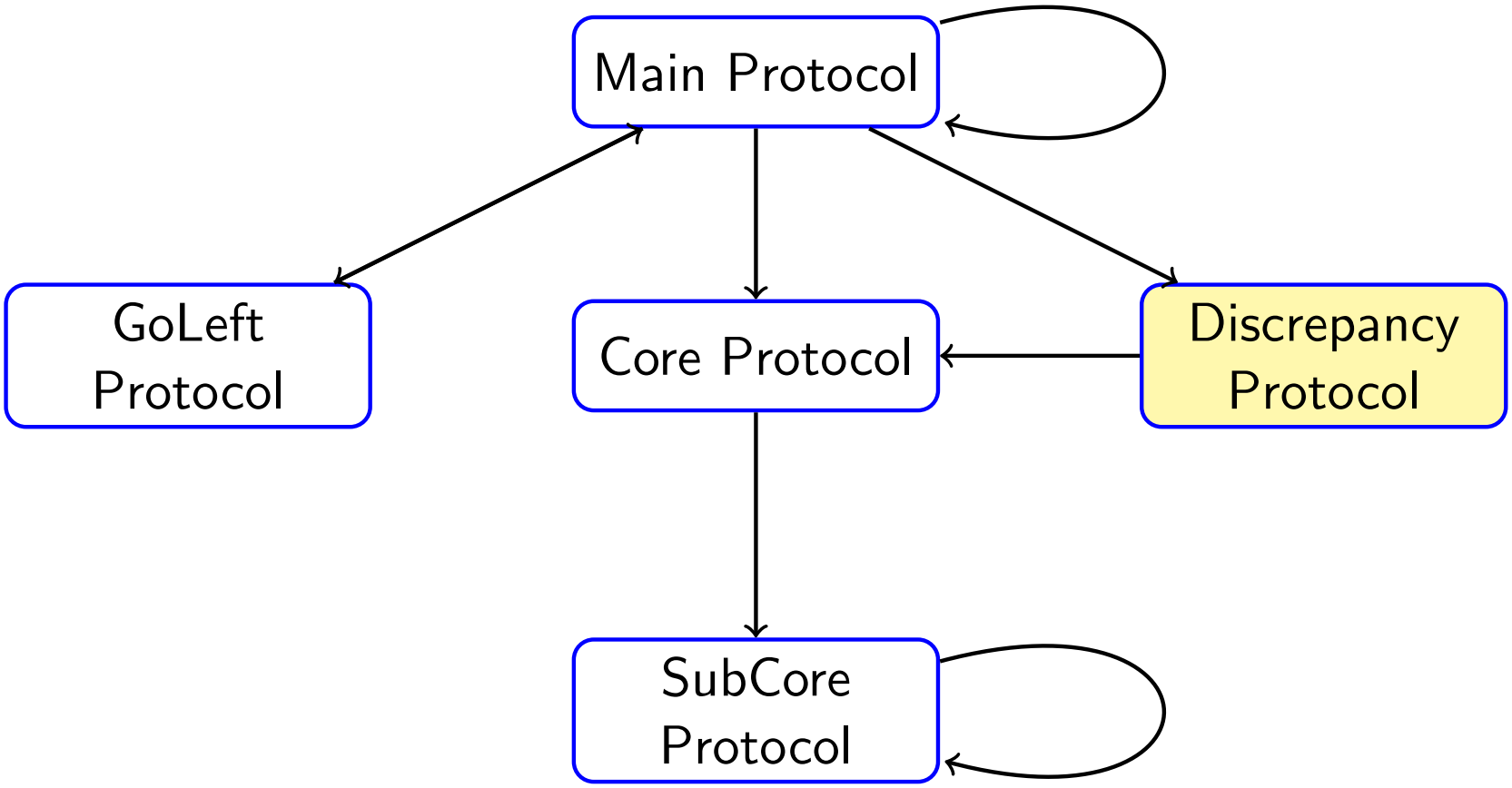


Figure: The envy-free protocol for n agents relies on various protocols. A protocol points to another protocol if it calls the other one. A protocol has a self-loop if it calls itself recursively.

Outline

Introduction

Problem Background

The New Protocol

- Simple but Useful Ideas
- Overview
- Core and SubCore Protocol
- Groundwork and New Concepts
- Main Protocol
- Discrepancy Protocol
- GoLeft Protocol

Conclusion

Discrepancy Protocol

Input: Residue R, *discrepant* piece e_{jkl}, set of extracted pieces of cake, agent set N.

Output: Possibly modified residue R, Boolean value called DISCREPANCY, set of agents D, set of agents D'

1: Remove the discrepant piece e_{jkl} from R; Reinsert all the *extracted* pieces back into the residue, relabel the aggregate piece R

2: While for some agent i, it is the case that $(\frac{n-2}{n})^B \frac{V_i(R)}{n} \leq V_i(e_{jkl}) \leq V_i(R)n$, run Core Protocol($R$,$i$,$N$) B times.

3: If all agents now consider e_{jkl} significant, then D' to N and DISCREPANCY to 0.

4: If agents disagree, then set D to $\{i \in N : V_i(e_{jkl}) \geq V_i(R)n\}$ and set D' to $\{i \in N : V_i(e_{jkl}) \leq \frac{V_i(R)}{n}\}$; RETURN R, DISCREPANCY=1, D and D'.

Either the discrepant piece becomes unanimously significant or we get a separation of agents. Both cases are helpful!

Main Protocol

Input: Cake R and a set of agents N.
Output: An envy-free allocation

1: **BASE CASE**: If $|N| \leq 4$, use a known bounded EF protocol
2: **CORE**: Generate core snapshots.
3: **EXTRACTION**: Start extracting pieces from the residue corresponding to the core snapshot pieces
4: **DISCREPANCY**: call Discrepancy if there is some piece to be extracted over which there is discrepancy. If there is real discrepancy, then some agents D go for the discrepant piece, others D' go for the residue! (Call Main Protocol).
5: **GOLEFT**: Call GoLeft
6: Call Core Protocol $2B$ times to change significant bonuses into dominance
7: Call Main Protocol(R, $A \subset N$), that is the Main Protocol on a subset of agents $A \subset N$ output by the GoLeft Protocol and with R as the input cake
8: **return** allocation of the cake to the agents

Overview

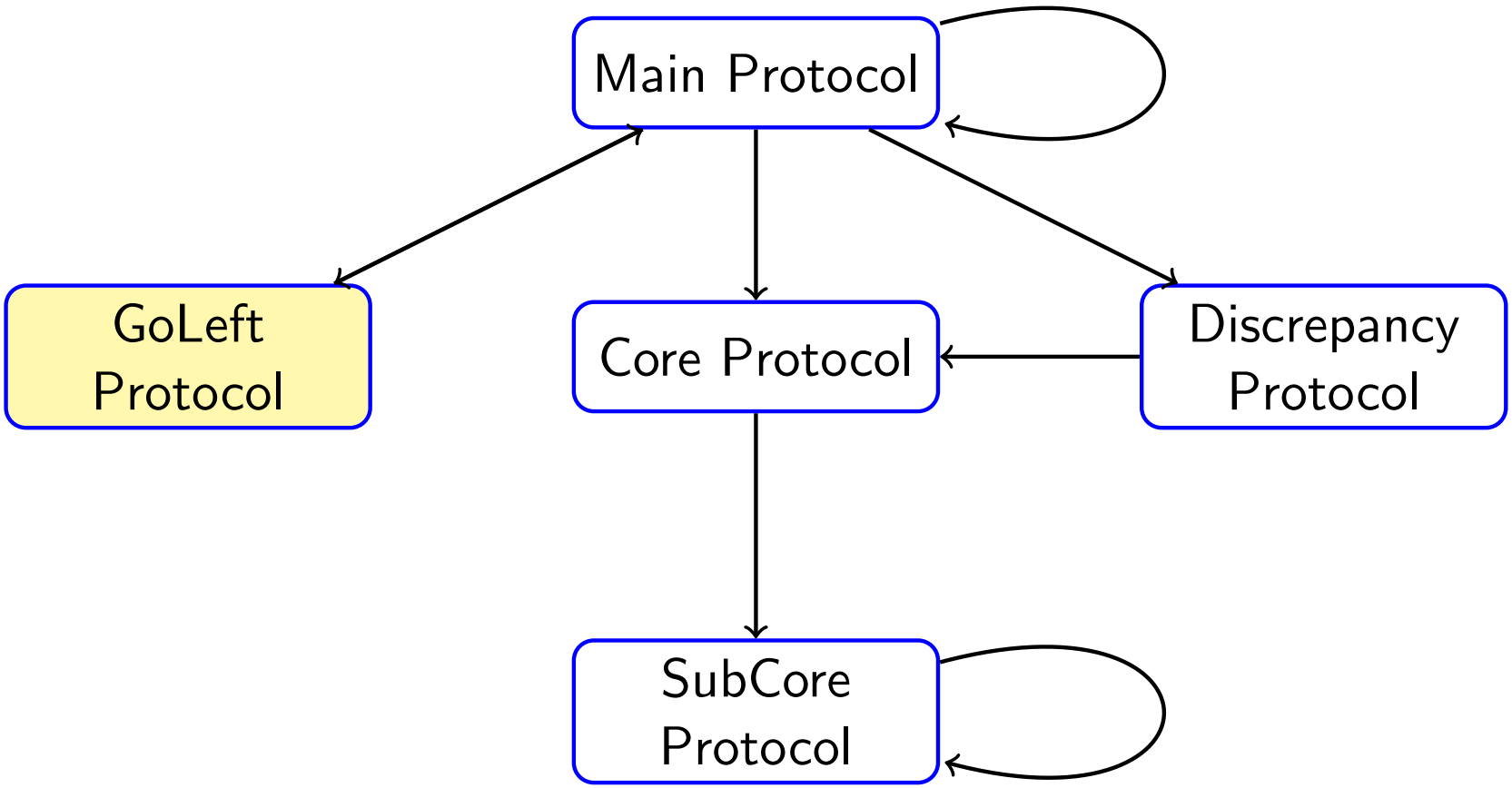


Figure: The envy-free protocol for n agents relies on various protocols. A protocol points to another protocol if it calls the other one. A protocol has a self-loop if it calls itself recursively.

Outline

Introduction

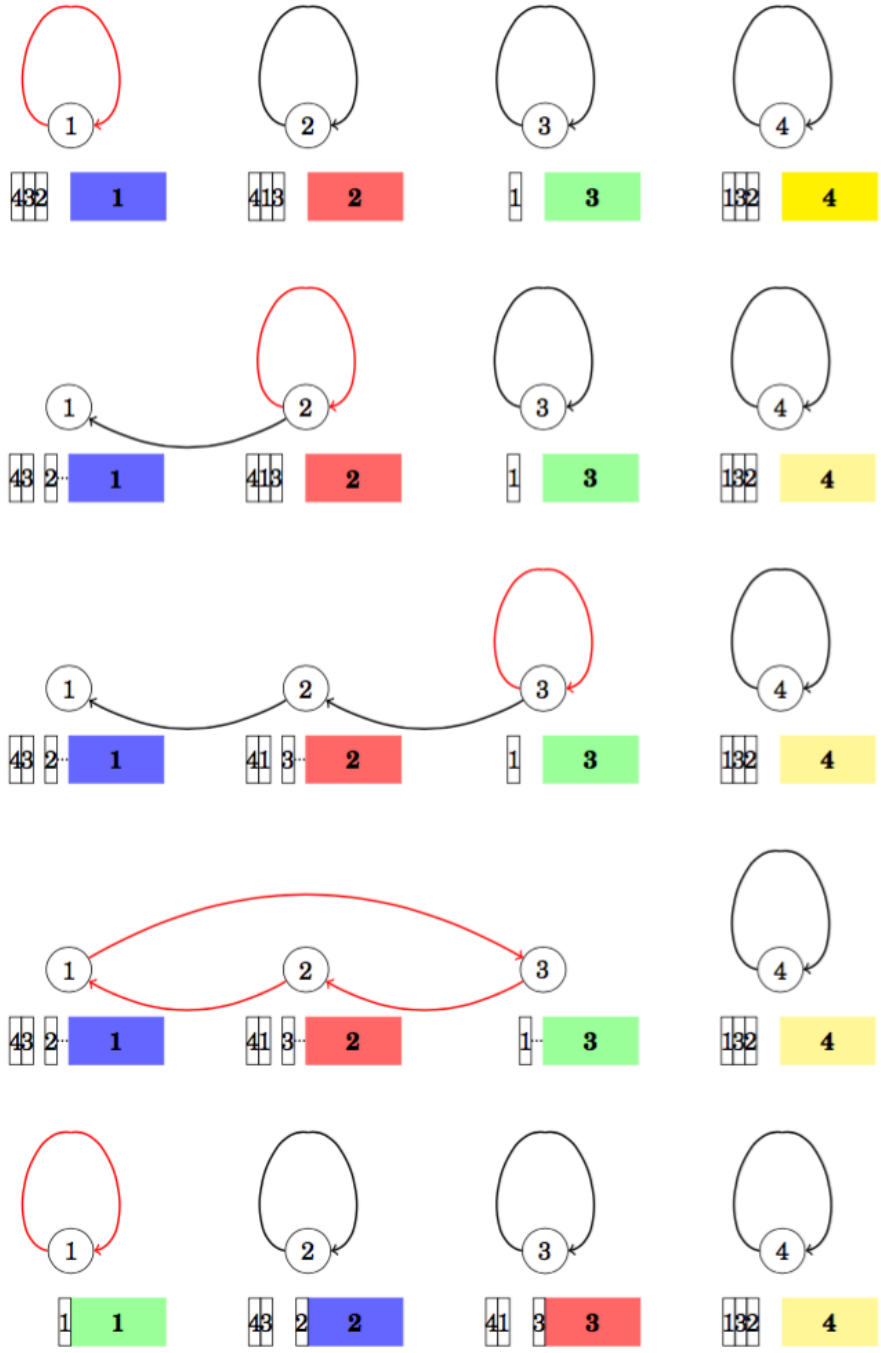
Problem Background

The New Protocol

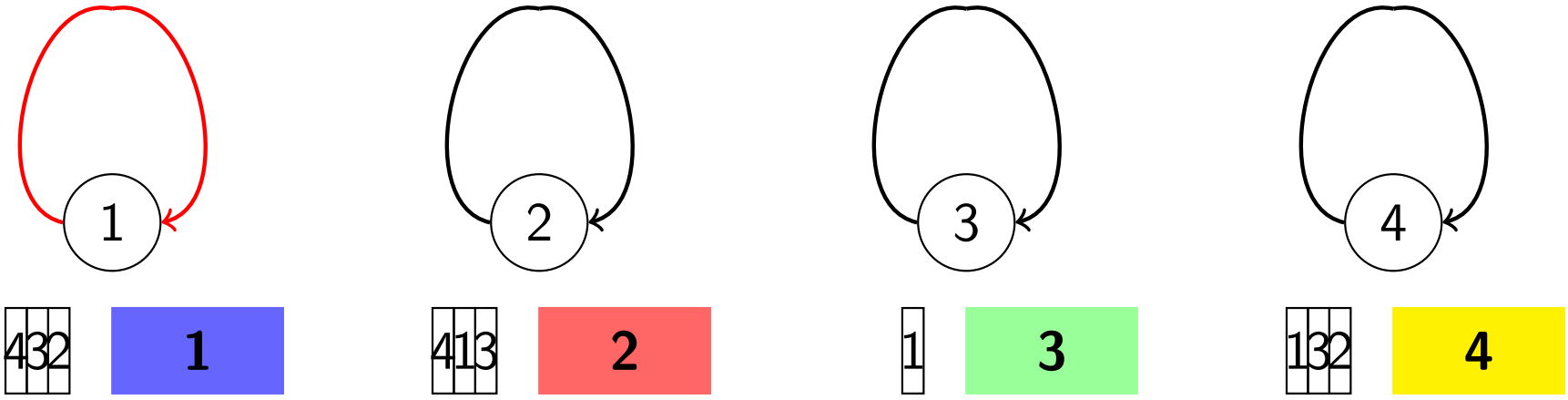
- Simple but Useful Ideas
- Overview
- Core and SubCore Protocol
- Groundwork and New Concepts
- Main Protocol
- Discrepancy Protocol
- GoLeft Protocol

Conclusion

Exchanges in the GoLeft protocol

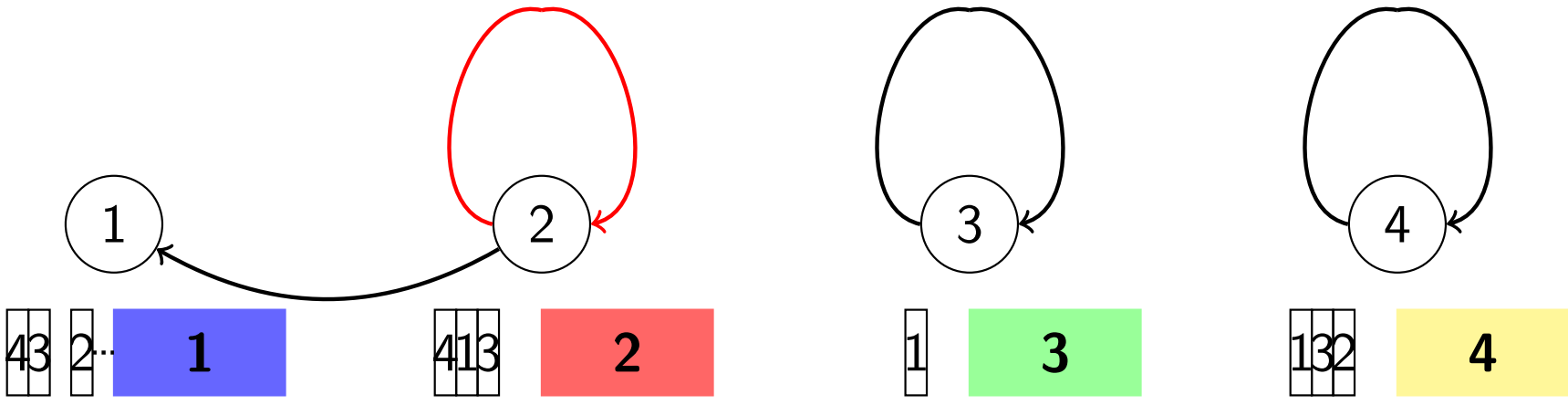


Exchanges in the GoLeft Protocol



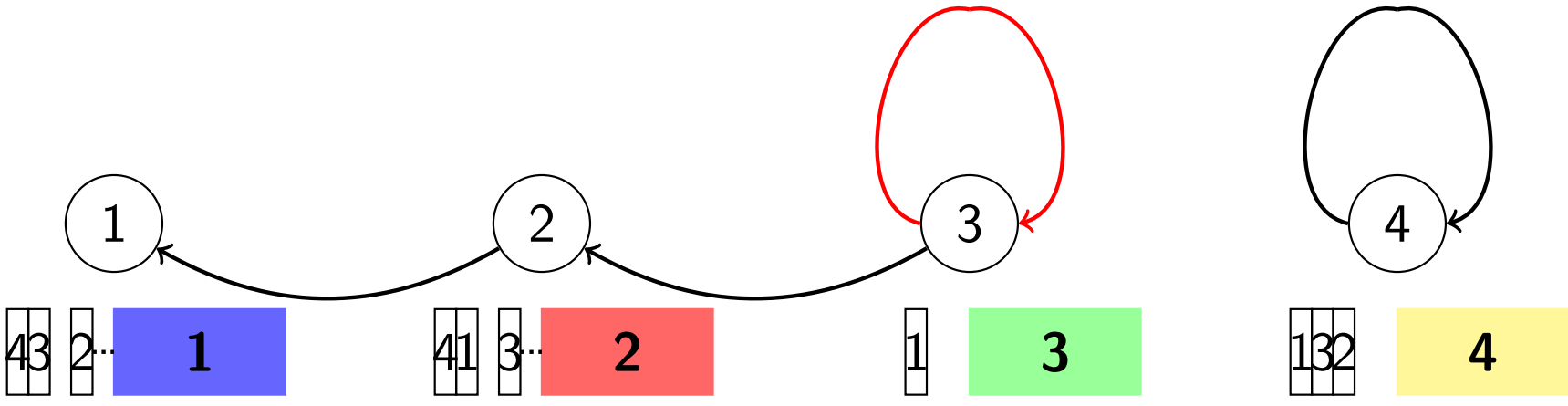
In the start each agent (node) points to itself. A cycle is guaranteed in which some agent has piece for which some extracted piece has not been attached. We want to attach the extracted piece to the piece.

Exchanges in the GoLeft protocol



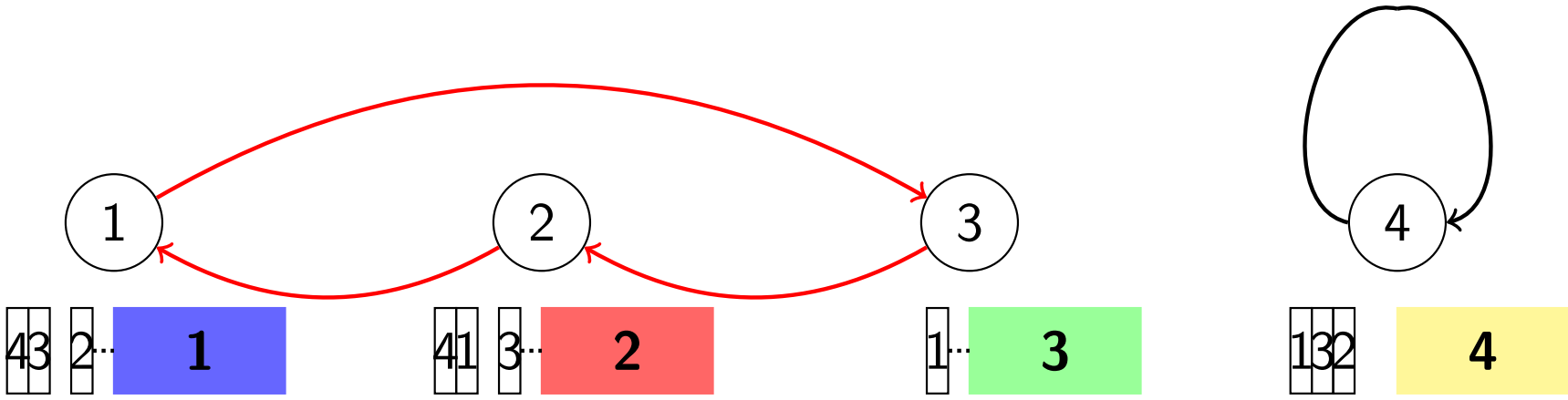
Extracted piece is attached to the blue piece. Agent 2 points to agent 1 because the piece extracted by agent 2 has been attached to 1's held (blue) piece. .

Exchanges in the GoLeft protocol



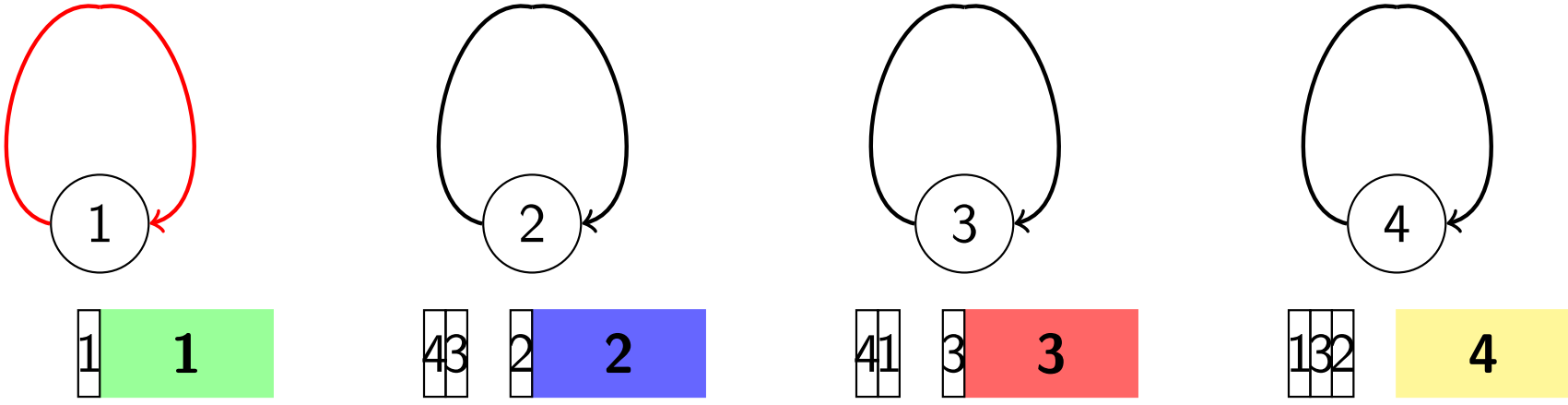
Agent 3 points to agent 2 because the piece extracted by agent 3 has been attached to 2's held piece.

Exchanges in the GoLeft protocol



Agent 1 points to agent 3 because the piece extracted by agent 1 has been attached to 3's held piece.
Agents $1, 2, 3$ exchange their currently held piece and are allocated cake up to their extracted piece.

Exchanges in the GoLeft protocol



Agent 1 holds a piece up till his extraction but neither agent 2 or 4 extracted pieces for the piece that agent 1 holds. This means that agents 2 and 4 have a significant advantage over agent 1. Initially the (green) piece was held by 3 and still is in discarded isomorphic snapshots. This implies significant advantage of 2 and 4 over 3. Therefore agent 2 and 4 can be made to dominate 1 and 3.

Going Left

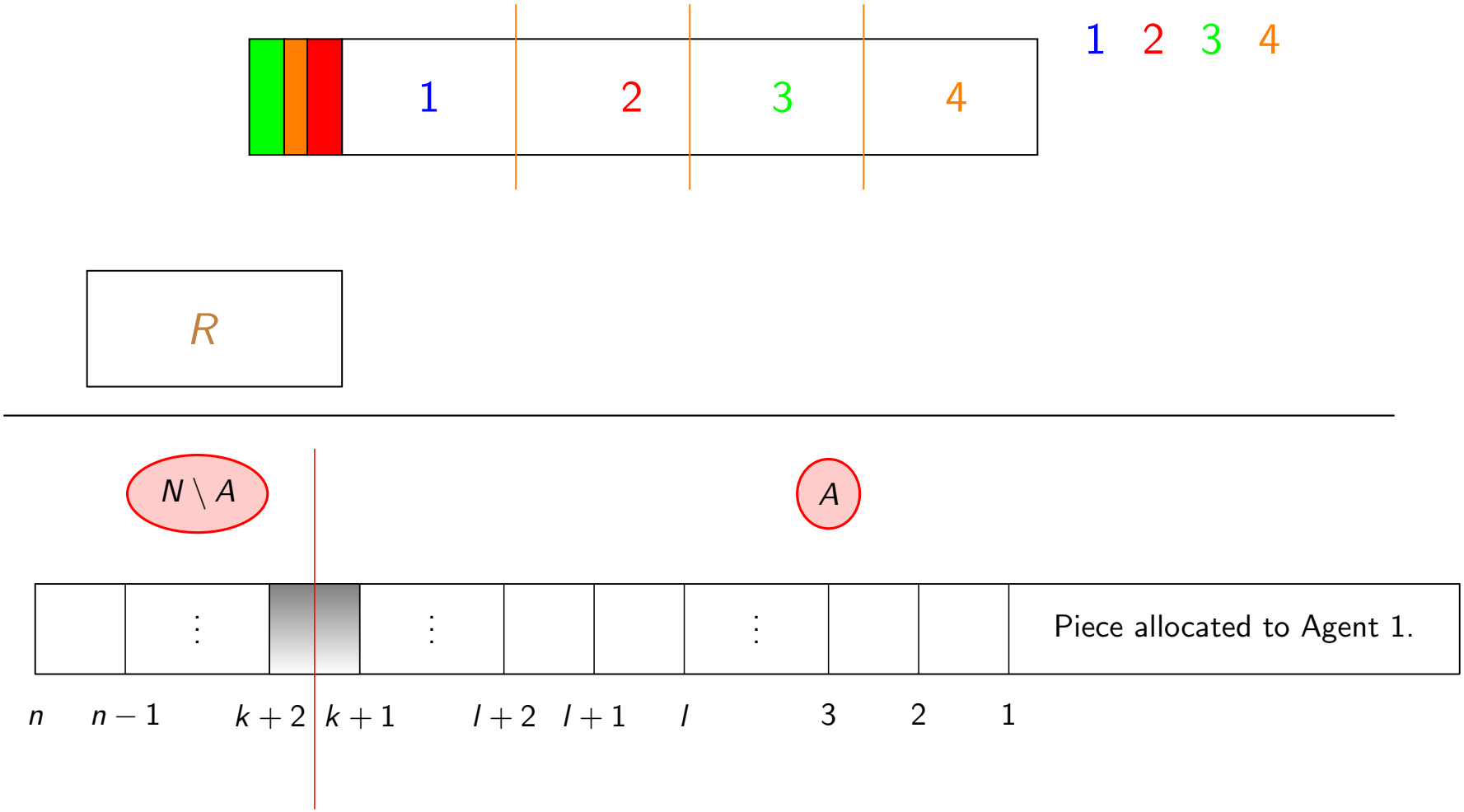


Figure: Illustration of the GoLeft protocol on a particular piece of cake that is originally allocated to agent 1. Agents $k+2$ to n will not go left and are the prospective dominators because they find the shaded space between the trims of $k+2$ and $k+1$ significant. Agents 2 to $k+1$ are the agents that go left.

GoLeft Protocol

Input: Set of C' snapshots, corresponding extracted pieces, and residue R.
Output: A set of agents $A \subset N$ such agents in $N \setminus A$ dominate agents in A

1: Select C' isomorphic snapshots (set S); Build the permutation graph.
   - T set of nodes with pieces for which $n-1$ extracted pieces have not been attached.
   - T' set of nodes with pieces for which $n-1$ extracted pieces have been attached.
2: **while** there is a node in T **do**
3: Find a cycle that includes a node that is from T (such a cycle always exists).
4: **PERMUTATION**: along the cycle. Update Permutation graph.
5: **if** there is a piece corresponding that is not from T' but has no more associated pieces to be attached **then**
6: {**SEPARATION**} found a separation! return the dominated set of agents A
7: **ATTACHMENT** : attach in a subset of the snapshots the set of extracted pieces $\{e_{k(l+1)}\}$ to the set of pieces $\{c_k\}$, thus making $\{c_k\}$ desirable to the agent who extracted $\{e_{k(l+1)}\}$. $S \leftarrow S \setminus S' \cup S''$. Update Permutation graph to reflect attachment. If the piece has had all $n-1$ extracted pieces attached, add the corresponding node to T' and make every node point to it.

GoLeft Protocol: Attachment

When we attach a piece for agent $l+1$ to come there. Other agents may be envious now.

- **Agents $l+1$ to n.** They had some bonuses in snapshots over agent l's allocation. They 'reserve' enough snapshots S' with the best bonuses by 'discarding those snapshots.'
- **Agents 1 to l.** Ask each of these agents to choose high enough fraction of the snapshots in which they value the $\{e_{k(l+1)}\}$ pieces. Bunch these pieces together and call Main protocol to divide in envy-free way. By proportionality they get enough value that they are okay with $l+1$ getting other $\{e_{k(l+1)}\}$ pieces. The corresponding set snapshopts S'' are 'discarded'.

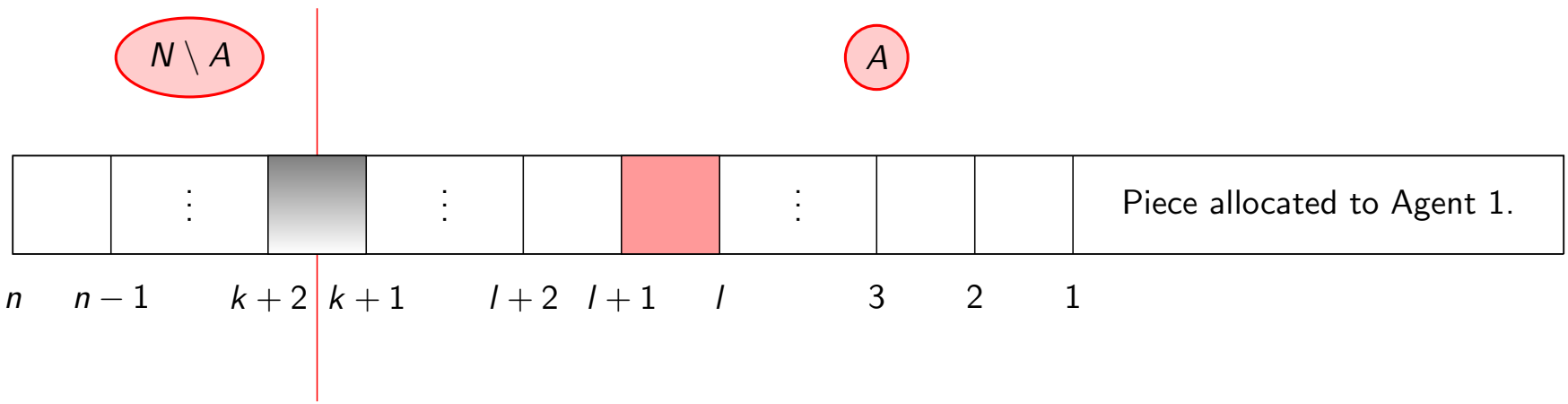


Figure: Attachment of red piece extracted by $\ell+1$.

Going Left: Attachment involves discarding snapshots

Figure: Illustration of how the set of snapshots S becomes smaller as GoLeft is run. Every time GoLeft attaches a new extracted piece to an allocated piece, snapshots are discarded (S' and S'') and a new set S which is a subset of the old one is formed.

GoLeft Protocol

Input: Set of C' snapshots, corresponding extracted pieces, and residue R.
Output: A set of agents $A \subset N$ such agents in $N \setminus A$ dominate agents in A

1: Select C' isomorphic snapshots (set S); Build the permutation graph.
   - T set of nodes with pieces for which $n-1$ extracted pieces have not been attached.
   - T' set of nodes with pieces for which $n-1$ extracted pieces have been attached.
2: **while** there is a node in T **do**
3: Find a cycle that includes a node that is from T (such a cycle always exists).
4: **PERMUTATION**: along the cycle. Update Permutation graph.
5: **if** there is a piece corresponding that is not from T' but has no more associated pieces to be attached **then**
6: {**SEPARATION**} found a separation! return the dominated set of agents A
7: **ATTACHMENT** : attach in a subset of the snapshots the set of extracted pieces $\{e_{k(l+1)}\}$ to the set of pieces $\{c_k\}$, thus making $\{c_k\}$ desirable to the agent who extracted $\{e_{k(l+1)}\}$. $S \leftarrow S \setminus S' \cup S''$. Update Permutation graph to reflect attachment. If the piece has had all $n-1$ extracted pieces attached, add the corresponding node to T' and make every node point to it.

Overview

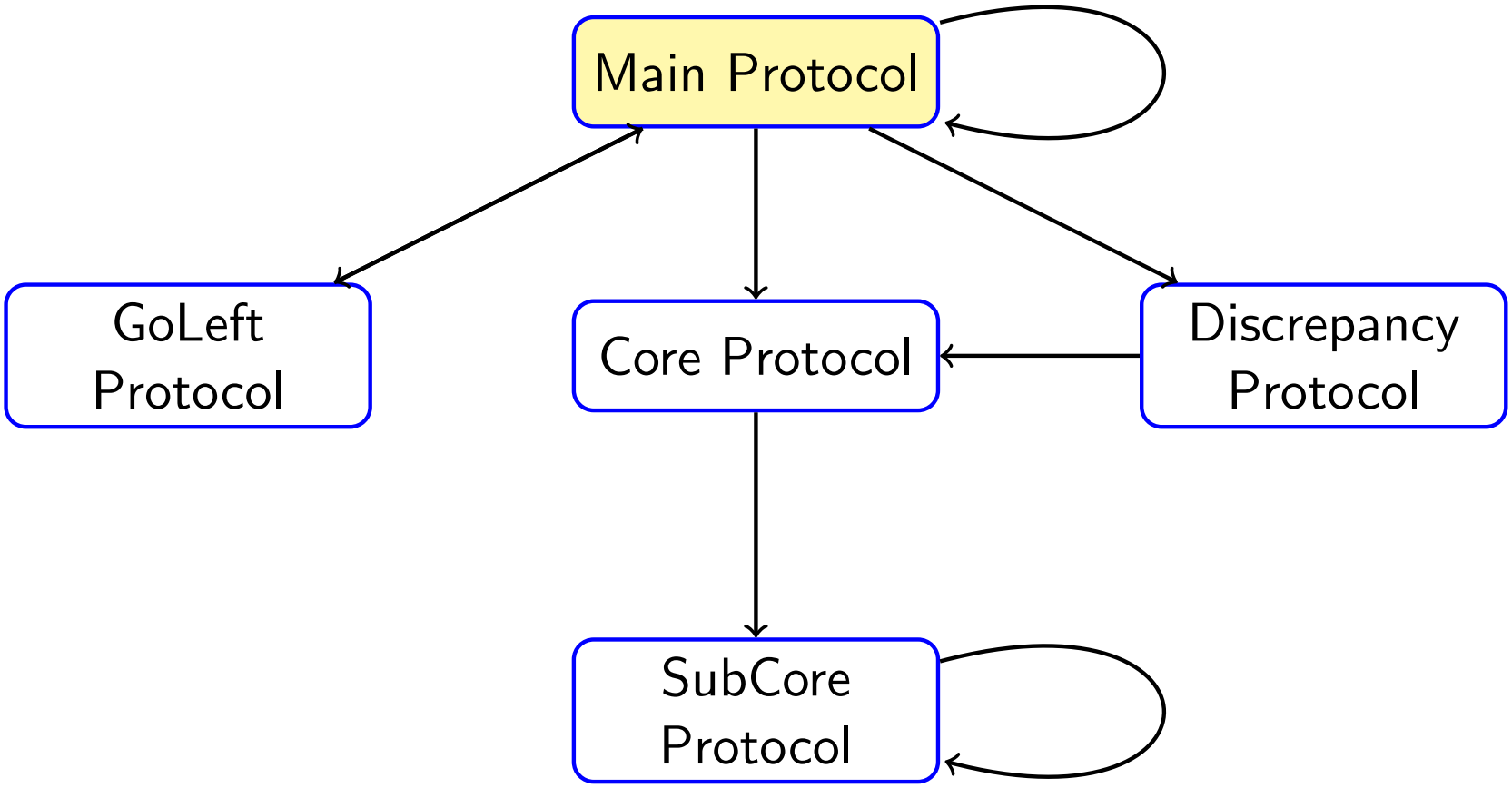


Figure: The envy-free protocol for n agents relies on various protocols. A protocol points to another protocol if it calls the other one. A protocol has a self-loop if it calls itself recursively.

Main Protocol

Input: Cake R and a set of agents N.
Output: An envy-free allocation

1: **BASE CASE**: If $|N| \leq 4$, use a known bounded EF protocol
2: **CORE**: Generate core snapshots.
3: **EXTRACTION**: Start extracting pieces from the residue corresponding to the core snapshot pieces
4: **DISCREPANCY**: call Discrepancy if there is some piece to be extracted over which there is discrepancy. If there is real discrepancy, then some agents D go for the discrepant piece, others D' go for the residue! (Call Main Protocol).
5: **GOLEFT**: Call GoLeft
6: Call Core Protocol $2B$ times to change significant bonuses into dominance
7: Call Main Protocol(R, $A \subset N$), that is the Main Protocol on a subset of agents $A \subset N$ output by the GoLeft Protocol and with R as the input cake
8: **return** allocation of the cake to the agents

Algorithm in a Nutshell

- Get an envy-free partial allocation
- From the unallocated cake, gradually give agents some cake while maintaining envy-freeness
- Get some agents to dominate all others which reduces the problem to less number of agents.

Conclusions

- Presented the first bounded envy-free cake cutting- algorithm
- Gap between lower bound ($\Omega(n^2)$) and upper bound ($n^{n^{n^{n^n}}}$)
- In at most $n^3(n^2)^n$ queries an envy-free and proportional partial allocation can be found!
- In at most $n^3(n^2)^n$ queries an envy-free partial allocation can be found in which each agent gets a connected piece of value $1/(3n)$ of the whole cake.

Conclusions

- Presented the first bounded envy-free cake cutting- algorithm
- Gap between lower bound ($\Omega(n^2)$) and upper bound ($n^{n^{n^{n^n}}}$)
- In at most $n^3(n^2)^n$ queries an envy-free and proportional partial allocation can be found!
- In at most $n^3(n^2)^n$ queries an envy-free partial allocation can be found in which each agent gets a connected piece of value $1/(3n)$ of the whole cake.

Nice surveys

- A. D. Procaccia. Cake cutting: Not just child's play. Communications of the ACM. 2013.
- A. D. Procaccia. Cake Cutting Algorithms. Handbook of Computational Social Choice. Cambridge University Press.
- C. Lindner and J. Rothe. Cake-cutting: Fair division of divisible goods. In J. Rothe, editor, Economics and Computation: An Introduction to Algorithmic Game Theory, Computational Social Choice, and Fair Division, chapter 7. Springer-Verlag, 2015.

Nice surveys

- A. D. Procaccia. Cake cutting: Not just child's play. Communications of the ACM. 2013.
- A. D. Procaccia. Cake Cutting Algorithms. Handbook of Computational Social Choice. Cambridge University Press.
- C. Lindner and J. Rothe. Cake-cutting: Fair division of divisible goods. In J. Rothe, editor, Economics and Computation: An Introduction to Algorithmic Game Theory, Computational Social Choice, and Fair Division, chapter 7. Springer-Verlag, 2015.

References I

S. J. Brams and A. D. Taylor. An envy-free cake division protocol. *The American Mathematical Monthly*, 102(1):9–18, 1995.

C. Lindner and J. Rothe. Degrees of guaranteed envy-freeness in finite bounded cake-cutting protocols. In *Proceedings of the 5th International Workshop on Internet and Network Economics (WINE)*, volume 5929 of *Lecture Notes in Computer Science (LNCS)*, pages 149–159. Springer-Verlag, 2009.

A. D. Procaccia. Cake cutting: Not just child's play. *Communications of the ACM*, 56(7):78–87, 2013.

A. Saberi and Y. Wang. Cutting a cake for five people. In *Proceedings of the 5th International Conference on Algorithmic Aspects in Information and Management (AAIM)*, Lecture Notes in Computer Science (LNCS), pages 292–300. Springer-Verlag, 2009.

References II

E. Segal-Halevi, A. Hassidim, and Y. Aumann. Waste makes haste: Bounded time protocols for envy-free cake cutting with free disposal. In *Proceedings of the 14th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 901–908. IFAAMAS, 2015.