# Multiagent Negotiation for Fair and Unbiased Resource Allocation

Karthik Iyer and Michael Huhns

Department of Computer Science and Engineering, University of South Carolina, Columbia SC 29208, USA
{iyerk, huhns}@engr.sc.edu

**Abstract.** This paper proposes a novel solution for the *n* agent cake cutting (resource allocation) problem. We propose a negotiation protocol for dividing a resource among *n* agents and then provide an algorithm for allotting portions of the resource. We prove that this protocol can enable distribution of the resource among *n* agents in a fair manner. The protocol enables agents to choose portions based on their internal utility function, which they do not have to reveal. In addition to being fair, the protocol has desirable features such as being unbiased and verifiable while allocating resources. In the case where the resource is two-dimensional (a circular cake) and uniform, it is shown that each agent can get close to 1/*n* of the whole resource.

## 1 Introduction

Autonomous agents are currently being given responsibilities in numerous applications from the economic, industrial, commercial, social, and entertainment sectors of the world. Autonomy means that the agents have a high degree of freedom and choice in initiating actions on their own, planning goals for themselves, and taking actions to achieve them. In order to achieve their goals agents must have sufficient resources and capabilities. But what if the resources are insufficient for all agents to achieve their goals? This typically is the case where an agent lives in a multiagent system and has to share resources with other agents, coordinate its own tasks to resolve conflicts, and sometimes persuade other agents to do things for it. All this could be done centrally by one designer, who could chalk out every detail of the entire system. The central agency would then be able to resolve conflicts, divide resources, plan schedules, etc. But this is not applicable in open systems where individual agents are designed by different designers and there is no concept of a central agency. Multiagent systems are inherently distributed, heterogeneous, and open. So what could be done in situations like this?

In the real world, people from different backgrounds come together and resolve conflicts or form alliances for mutual benefits. This sort of interaction between people is based on negotiation. Negotiation [1] is "when two parties strike a deal through argumentation or arbitration for the mutual good of both."

Negotiation as per Davis and Smith [2] is "A discussion in which interested parties exchange information and come to an agreement." This work revolves around a resource allocation problem and the main concern of agents participating in the negotiation scheme described is to ensure a fair and unbiased allocation of a resource.

R. Meersman and Z. Tari (Eds.): CoopIS/DOA/ODBASE 2005, LNCS 3760, pp. 453 – 465, 2005.

In open multiagent systems there is generally no global control, no globally consistent knowledge, and no globally shared goals or success criteria. So there exists a real competition among agents, which act to maximize their own utilities. We assume all the utility functions are private to the agents. The negotiation protocol should be immune to information hiding and lying by agents. This has to be ensured as there is no control on the design of agents that interact in open environments and the only check that could be made is by cleverly designing their interaction mechanism. We describe our negotiation protocol that tries to incorporate the desirable characteristics mentioned above. We use the problem domain of cake cutting in order to better visualize and formalize our solution for an *n*-agent resource allocation protocol.

## 2 Background

The cake-cutting problem is a well known example of resource sharing among rational agents. The classical solution for the two-person case, divide and choose, was first proposed by Steinhaus [3]. This solution, where one person divides the cake into two pieces and the other gets first choice of a piece, is both fair and envy free. However, it has been difficult to scale up the solution to *n* agents. One of the solutions [4] for dividing the cake among *n* agents fairly has been to use a moving knife parallel to one of the edges of the cake. The knife cuts when one of the agents yells "Cut!" and the portion traversed by the knife so far is allotted to the agent. This is an elegant and clean *n* agent solution for creating *n* portions fairly in *n-1* cuts. But the moving knife solution has its own set of drawbacks. First, it is not envy-free. Second, it requires the presence of an unbiased mediator who holds the knife and moves it along the cake at a constant rate. Despite this safeguard, it is difficult to verify the cutting of the cake. For example, if one of the agents alleges that the mediator cut the cake an inch shorter than he had expected, it would be difficult to find out who is telling the truth. In a distributed system, synchronization problems may also occur. An agent with a slower connection to the mediating authority will find its bid to cut may reach later than an agent with a faster connection and may consequently lose the bid. Third, the moving knife protocol is also not pareto optimal. A scheme [5] to improve efficiency in the pareto optimal sense has been proposed with the use of two moving knives. However, the solution works only for division of the resource between two agents, and the agent that moves the knives must be able to estimate the utility function of the other agent well.

Other solutions to *n*-person division attempt to create a protocol that does not require assistance from an outsider. One way is to scale up from the two-person solution and iteratively add new agents until all have allocations. For *n=2*, the classic divide and choose is used. When agent 3 is added, agents 1 and 2 each divide their portions into 3 parts. Agent 3 then picks one part from each of the other agents. This continues as each agent is added. The drawback for this protocol is that the earlier agents will be faced with the chore of repeatedly dividing their portion into many pieces. The agent that is added last will get its share by doing the least amount of work.

Another solution [6] converts the *n* agent division problem into many *n-1* agent problems and then recurses. The recursive calls return when the many two-agent problems are resolved and the answers back up to the top-level call. The drawback is that an agent whose shares remain unallocated till the end has to continuously re-bid for scattered pieces until the iterations end.

A divide-and-conquer procedure [7] instructs the agents to cut the cake into half according to their measure. Then the cuts are ordered and the first *n/2* cuts are allotted the left half of the cake. The rest are allotted the right half. This procedure continues until 2 agents have to cut the cake where the well known divide-and-choose algorithm can be implemented. An obvious drawback of this procedure is that the number of agents needs to be a power of two, which is an unrealistic requirement.

In the next section we present an improved protocol for *n*-agent resource division. First we discuss the case of dividing a linear resource, such as a rectangular piece of cake or property along a coastline. Then we show that we can use this protocol for allocating a two-dimensional resource, such as a circular cake, time slots in a 24 hour cycle, or any resource that does not have a well defined starting point. Finally we describe an improved version of the protocol specifically for a circular resource, which removes some of the drawbacks of the linear case.

## 3 Dividing a Linear Resource Among *n* Agents

### 3.1 Protocol

The problem of dividing a linear resource among *n* agents can be solved by following the protocol below. An example of how a strip of property along a coastline is distributed among three agents is discussed in [8] and [9]. The protocol for the agents is simple: They are required to make *n-1* marks of the property that delimit *n* intervals. To the agent making these marks, each of these intervals should be worth *1/n* of the whole piece and is equally desirable. The procedure guarantees that given such a set of marks, each agent will be guaranteed one of the intervals it has marked. The protocol is fair because each agent will be given one of the intervals it marked.

Refer to figure 1 to see the functioning of the protocol. Given *n* agents who need to divide a resource among themselves, they approach a mediator (which could be one of the bidding agent themselves) to get their portions of the resources allocated. They register with the mediator, informing it of their interest in participating. After the mediator confirms their participation, the agents submit their bids as a set of marks denoting equal-value portions in their estimation. The mediator waits for all the agents to submit their bids. Then it calculates portions to be allocated to each agent and finally informs each agent of the portion allocated to it.

**Theorem 1.** If there are *n* agents and each agent makes *n-1* marks, creating *n* portions of a linear cake, then our protocol guarantees that each agent will be allotted a piece of the cake, such that the piece was one of the *n* portions created by the agent itself.

**Proof:** This proof assumes that the left and right portions of the cake have been allotted to the agents whose marks came first on the left side and right side respectively. We concentrate on allotting pieces of the cake to the agents still remaining after this procedure. Note that after the first 2 agents have received their share, the marks of the remaining agents need not start at the same point. Now we will have *n-2* agents with each agent having at least *n-1* marks creating at least *n-2* pieces of the cake. Without losing generality, we can add 2 to the above numbers and re-state the problem as:

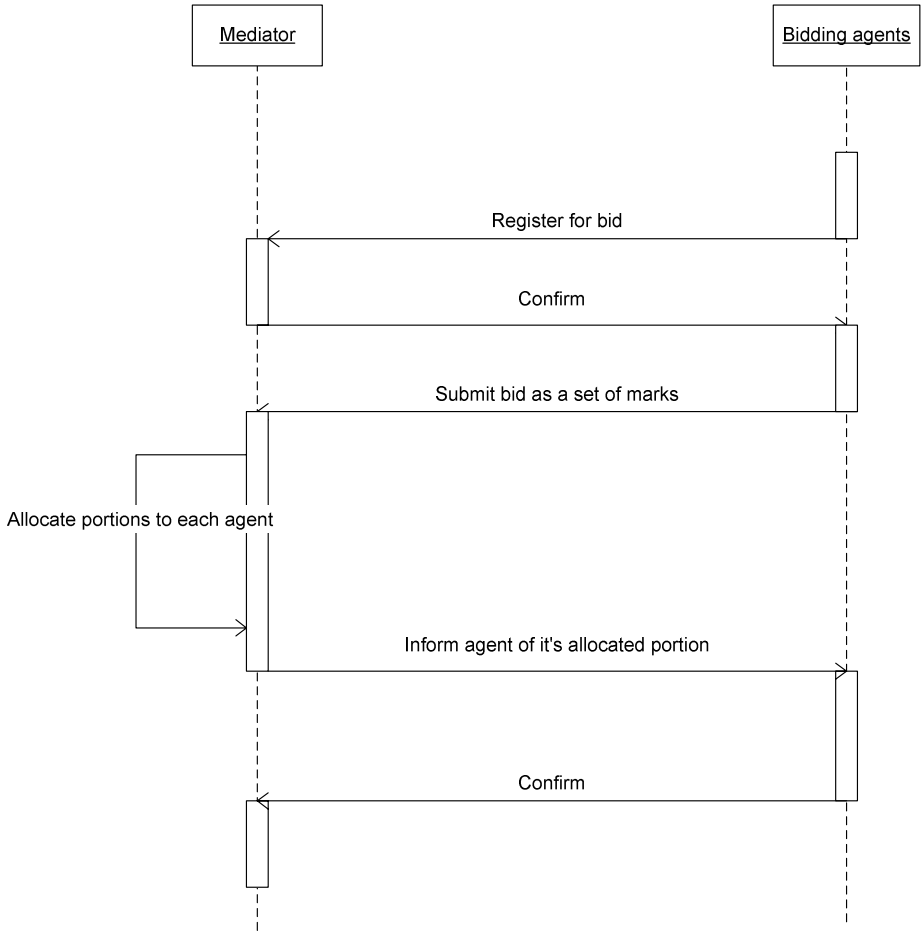


**Fig. 1.** Linear cake cutting protocol

There are $n$ agents with each agent having at least $n+1$ marks creating at least $n$ pieces or intervals of the cake. It is proved that each agent will be guaranteed a piece of the cake such that the piece was marked by the agent himself.

When agents create marks (that represent their cuts), the following possibilities exist for a particular chosen interval.

1. Pure interval, i.e., no other agent's interval intersects this interval
2. Mixed interval
   2.1. The current interval intersects another interval. It does not completely contain any other interval.
   2.2. The current interval contains at least one interval made by at least one agent. In addition, there will definitely be an interval that partially intersects the current interval.

**Base Case *(n=2).*** Each agent will create 2 intervals, each using 3 marks. Find the first mark. Say this mark belongs to agent A. The next mark may or may not be made by A.

1. If the next mark is made by A, this is a pure interval, so allocate the current interval to A. Remove all marks to the left of this interval. Remove all marks made by agent A. We will be left with marks made by agent B to the right of the current interval. Repeat the above procedure. None of B's marks have been deleted yet. Hence the procedure is guaranteed to find an interval to allocate to B, as no other agents are left.
2. If the next mark is made by B, A's interval is mixed. Either case 2.1 or 2.2 will occur.
   2.1. A's interval intersects partially with B. In this case allocate the interval to A. Remove all marks to the left of interval. Remove all of A's marks. Since A's interval intersected with B's interval, the previous step will reduce B's marks to 2. Since B has two marks left demarcating an interval and no other agent is remaining, the procedure is guaranteed to find an interval for B.
   2.2. A contains at least one interval of B. Since there are no more agents, such an interval of B is guaranteed to be a pure interval and is allocated to B. Remove all marks to the left of the interval. Remove all of B's marks. Since A contained B, the previous step will reduce A's marks to 2. Since A has two marks left demarcating an interval and no other agent is remaining, the procedure is guaranteed to find an interval for A.

This proves that A and B can be allocated fair shares, no matter how the marks are arranged.

**For any *n* *(n>2).*** Let us assume that the allocation procedure works for up to $k$ agents. We will show that the procedure works for $k+1$ agents. Each agent will create $k+1$ intervals, each, using $k+2$ marks. Find the first mark. Say this mark belongs to agent $i$. The next mark might or might not be made by $i$.

1. If the next mark is made by $i$, this is a pure interval, allocate current interval to $i$. Remove all marks to the left of this interval. Remove all marks made by agent $i$. None of the marks made by other agents will be removed as this was a pure interval for agent $i$. Hence we will be left with the $k+2$ marks and $k+1$ intervals made by each of the $k$ agents to the right of the current interval. Delete the leftmost mark of each of the $k$ agents. Thus each agent is left with $k+1$ marks and $k$ intervals. This transforms into the allocation procedure for $k$ agents, which we know works. Hence proved.
2. If the next mark is not made by $i$, suppose the mark belongs to agent $j$. Add $i$ to the list of agents whose mark has already been seen. Repeat the allocation procedure with $j$'s mark as the first mark. At some point we will encounter a mark made by one of the agents already in the list. Say the first such agent is $l$. The interval demarcated by $l$ will not contain any other agent's interval. It may or may not partially intersect with other agent's intervals.
   2.1. If $l$'s interval does not intersect with any other interval, then allocate interval to $l$. Remove all marks to the left of this interval. Remove all marks made by agent $l$. The previous step will remove at most one mark of the agents. Other agents will have $k+2$ marks and $k+1$ intervals. Start from the beginning of the

list and as each mark is encountered, check if the mark belongs to an agent already in the agent list. If so, ignore the mark; otherwise add the agent to the list and delete the mark. This step guarantees that we will have $k$ agents, each with $k$ intervals and $k+1$ marks. This transforms into the allocation procedure for k agents, which we know works. Hence proved.

2.2. If $l$'s interval partially intersects with some other interval, then allocate the interval to $l$. _Remove all marks to the left of this interval. Remove all marks made by agent $l$. The previous step will remove at most one mark of the agents. Other agents will have $k+2$ marks and $k+1$ intervals. Start from the beginning of the list and as each mark is encountered, check if the mark belongs to an agent already in the agent list. If so, ignore the mark; otherwise, add the agent to the list and delete the mark. This step guarantees that we will have $k$ agents each with $k$ intervals and $k+1$ marks. This transforms into the allocation procedure for $k$ agents, which we know works. Hence proved.

This proves that the allocation procedure works for $n$ agents, for any $n \geq 2$.

### 3.2 Features

If, rather, the resource is circular in shape (e.g., time slots in a 24 hour cycle or a circular cake), can we use the above procedure to get a solution? Henceforth we use the example of a circular cake to explain the division of a circular resource. A quick way to apply the linear protocol to a circular cake is the following.

The protocol begins with a mediator making a mark on the cake denoting it as a starting point. Now take a strip of length equal to the circumference of the cake. Wrap the strip around the cake such that the start and end points of the strip meet at the point marked on the cake. This will show agents the relative locations of their favored pieces. Lay out the strip on a flat surface and let $n$ agents make $n-1$ marks on it demarcating $n$ intervals.

Next, use the allocation procedure for the linear problem to allocate intervals to agents. Take the strip that now shows the intervals allocated to the agents and wrap it around the cake with the start and end of the strip meeting at the point marked on the cake. Make radial portions of the cake according to the intervals marked on paper and allot them to the agents based on their markings. This protocol divides a circular cake among $n$ agents in a fair manner. It has the following features:

1) The protocol is fair because each agent gets one of the pieces it demarcated for itself.
2) It might not be envy-free, because the initial mark made by the mediator might break up a region of interest for some agents into start and end parts. Such agents will end up at a disadvantage over the others and might envy that the allocation procedure tilted in favor of other agents.
3) The linearization makes it difficult for the agents to visualize which portion of the cake they really wanted to have. The portions of the circular cake have to be converted to linear lengths on the strip, which can be non-trivial.

Attempting to apply the linear protocol on a circular cake will end up creating a bias, because it requires a starting point to be declared, which might favor one of the agents. Is there an *unbiased* way to allocate portions of a circular cake? That is the topic of the next section.

# 4 A Fair and Unbiased Protocol to Partition a Circular Resource Among *n* Agents

## 4.1 Protocol

The protocol that needs to be followed by the agents is fairly simple. Given a circular cake, an agent can make radial marks at any point. If there are $n$ agents then every agent needs to make $n+1$ radial marks on the cake creating $n+1$ pieces of the cake. The allocation procedure will allot one of these pieces to the agent. Thus there will be a total of $(n+1)^2$ marks of the cake. Once agents submit their bid on how the cake will be cut, each agent will be allotted one piece of the cake demarcated by its own marks that will have a worth of $1/(n+1)$ in its opinion. If every agent follows the protocol, then fair allotments are guaranteed for every agent.

Figure 2 shows the interaction diagram for the protocol. As will be shown later, due to inefficiency in the allocation process, there will be leftovers which may not be negligible in the valuation of the agents. In such a case the agents may ask to bid for the leftovers using the same bidding protocol.

**Theorem 2.** If there are $n$ agents and each agent makes $n+1$ radial marks, creating $n+1$ portions of the circular cake, then our protocol guarantees that each agent will be allotted a piece of the cake, such that the piece was one of the $n+1$ pieces created by the agent itself.

**Proof:** This proof assumes that the marks are strictly in increasing order, viz. no two points are in exactly the same position. We define two terms: A *k-circle* and a *k-sector.*

**k-circle.** A k-circle consists of a circle that has $k+1$ marks made by each of the $k$ agents demarcating $k+1$ intervals.

**k-sector.** A k-sector is formed when one or more intervals have been removed from a circle. A k-sector contains at least $k+1$ marks demarcating at least $k$ portions for each of the $k$ agents. By this definition, a k-sector is also a (k-1)-sector, which is also a (k-2)-sector, etc. A k-sector may or may not be a (k+1)-sector however.

When agents mark their intervals, the following possibilities exist for a particular chosen interval.

1. Pure interval, i.e., no other agent's interval intersects this interval
2. Mixed interval
   - 2.1. The current interval partially intersects another interval. It does not completely contain any other interval.
   - 2.2. The current interval contains at least one interval made by at least one agent. In addition, there will definitely be an interval that partially intersects the current interval.

In the case of a k-circle, the following events may occur:

1. If the interval allotted was pure, then the other agent's intervals were outside this interval, i.e., the portions demarcated by other agents included the current interval and some extra portions ahead and behind the current interval. Since the current interval was allocated, the corresponding portions of the other agents are destroyed because they can no longer include the currently allocated portion.

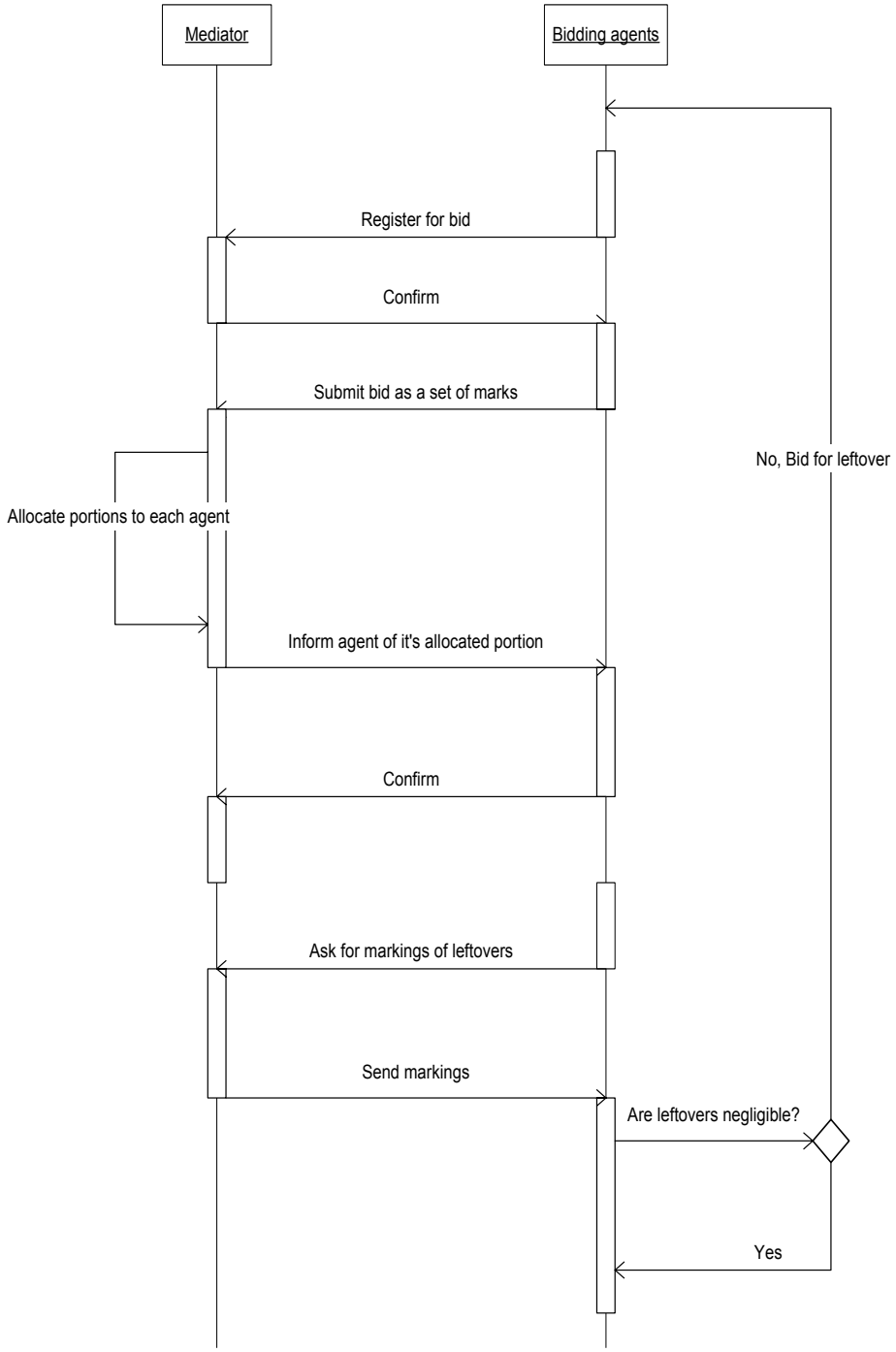


**Fig. 2.** Circular cake cutting protocol

2. If the interval allotted was mixed, then there was some overlap with other agents' portions. When this interval is allocated, then it destroys the two intervals of the other agents and they can no longer be allocated to the respective agents.

The above cases show that for a k-circle, allotting an interval will destroy at least one and at most 2 intervals of other agents.

In the case of a k-sector, the following events may occur.

1. If the interval allotted was pure then at most 1 interval of other agents is destroyed.
2. If the interval allotted was mixed then it means other agents interval started after the current interval. Allocating the current interval would destroy at most one interval of the other agents.

The above cases show that in case of a k-sector, allotting an interval will destroy at most one interval of the other agents. We will now show an inductive proof for the allocation procedure.

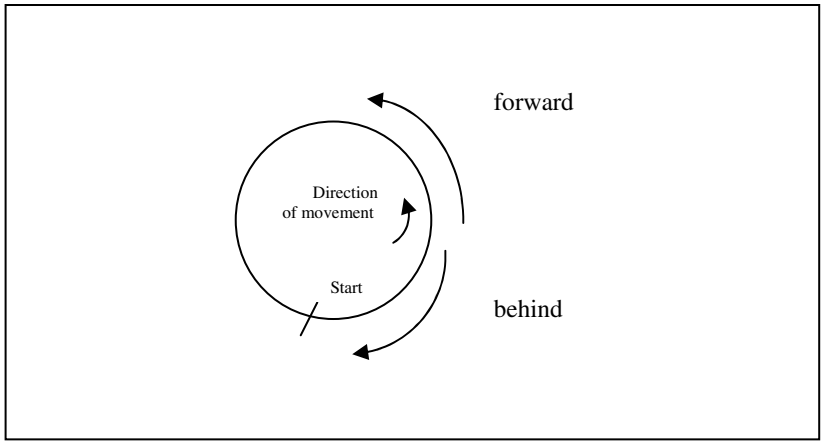


**Fig. 3.** Conventions used in the proof

**Base Case *(n=2)*.** For 2 agents we will have a 2-circle with each agent making 3 marks and demarcating 3 intervals respectively. Let the agents be A and B. Arbitrarily mark a position on the circumference as a "start" mark. Without losing generality, we choose to move in the counterclockwise direction. In this proof, "moving back" means moving clockwise till one has reached the start mark. Similarly "moving forward" means moving counterclockwise till one has reached the start mark. Find the first mark and assume this mark belongs to agent A. The next mark may or may not be made by A.

1. If the next mark is made by A, this is a pure interval, so allocate this interval to A. Remove all marks to the back of this interval. Remove all marks made by agent A. Thus we will be left with a 1-sector having marks from agent B only. Since none of B's marks have been erased, the 1-sector will have 3 marks demarcating 2 intervals. Allot any of these intervals to B.

2. If the next mark is made by B, A's interval is mixed. Either case 2.1 or 2.2 will occur.
   2.1. A's interval intersects partially with B. In this case allocate the interval to A. Remove all marks behind the interval. Remove all of A's marks. We will be left with a 1-sector. Since A's interval intersected with B's interval, the previous step will reduce B's marks to 2. Thus the 1-sector will have 2 marks demarcating 1 interval. Allot this interval to B.
   2.2. A contains at least one interval of B. Since there are no more agents, such an interval of B is guaranteed to be a pure interval and is allocated to B. Remove all marks behind the interval. Remove all of B's marks. We will be left with a 1-sector. Since A contained B, the 1-sector will have 2 marks demarcating 1 interval. Allot this interval to A.

This proves that A and B can be allocated fair shares, no matter how the marks are arranged. Now let us turn our attention to the general case of *n* agents.

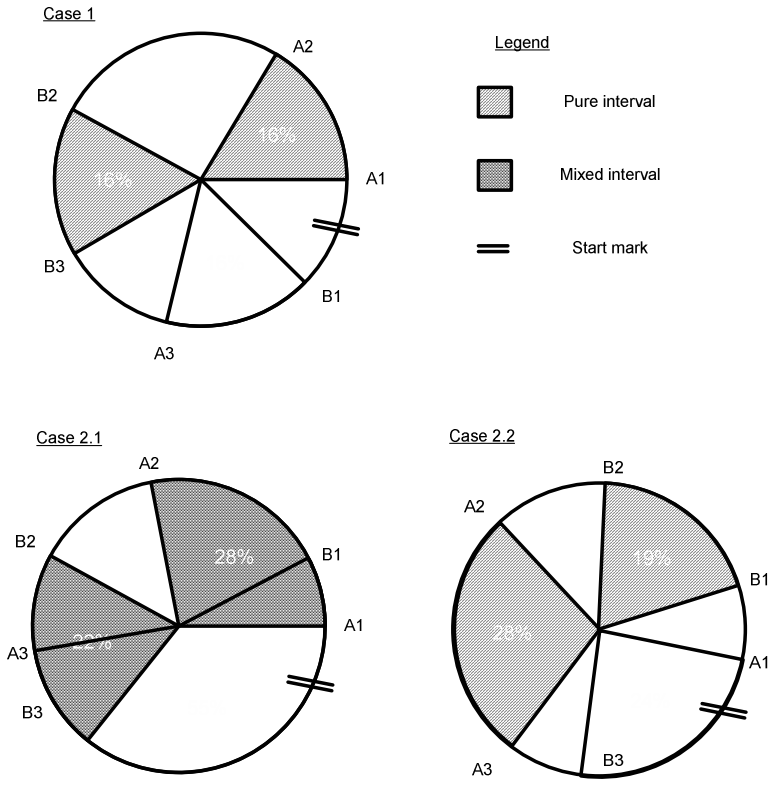


**Fig. 4.** Allocations for *n=2* for a circular resource

**For any *n (n>2)*.** Let us assume that the allocation procedure works for up to *k* agents. In order for the allocation procedure to work, one of the *k* agents would have

been allotted an interval from a k-circle. After this a (k-1)-sector would have formed, which was used to allot intervals for the rest of the *k-1* agents. Hence we can conclude that the allocation procedure works for up to (k-1)-sector. We will show that the procedure works for *k+1* agents.

For *k+1* agents we create a (k+1)-circle which will have *k+2* marks demarcating *k+2* portions for each of the agents. Find the first mark. Say this mark belongs to agent *i*. The next mark may or may not be made by *i*.

1. If next mark is made by *i*, this is a pure interval, allocate current interval to *i*. Remove all marks behind this interval. Remove all marks made by agent *i*. None of the marks made by other agents will be removed as this was a pure interval for agent *i*. We will be left with a k-sector that has *k+2* marks demarcating *k+1* intervals. Read in the next mark. Say it belongs to agent *j*.
   1.1. If the mark after this belongs to *j* then this is a pure interval which can be allocated to *j*. Remove all marks made by *j*. We will be left with a (k-1)-sector that has *k+2* portions demarcating *k+1* intervals for each of the *k-1* agents. Since *k-1* sector problem has been solved, Hence proved.
   1.2. If the mark belongs to some other agent *l*, then add agent *j* to list of agents whose mark has already been seen and repeat allocation procedure for the k-sector with agent *l*'s mark as the first mark. Once an interval has been allocated to an agent, remove all marks of the agent. Remove all marks behind this interval. Other agents may lose at most one mark. Thus we now have a (k-1)-sector problem where *k-1* agents have at least *k+1* marks and demarcating *k* intervals. Hence proved.
2. If the next mark is not made by *i,* suppose the mark belongs to agent *j*. Add *i* to the list of agents whose mark has already been seen. Repeat the allocation procedure with *j*'s mark as the first mark. At some point we will encounter a mark made by one of the agents already in the list. Say the first such agent is *l*. The interval demarcated by *l* will not contain any other agent's interval. It may or may not partially intersect with other agent's intervals.
   2.1. If *l*'s interval does not intersect with any other interval, then allocate the interval to *l*. Remove all marks made by agent *l*. We will be left with a k-sector that has *k+2* marks demarcating *k+1* intervals for each of the *k* agents. This k-sector is the same as the one discussed in case 1.1 whose solution has been described. Hence proved.
   2.2. If *l*'s interval partially intersects some other interval, then allocate the interval to *l*. Remove all marks made by agent *l*. The previous step will remove at most one mark of an agent. Other agents will have *k+2* marks and *k+1* intervals. We will be left with a k-sector having at least *k+1* marks demarcating *k* intervals. Read in the next mark. Say it belongs to agent *j*.
      2.2.1. If the mark after this belongs to *j* then this is a pure interval, which can be allocated to *j*. Remove all marks made by *j*. We will be left with a (k-1)-sector having at least *k+1* marks demarcating *k* intervals for each of the *k-1* agents. Since *k-1* sector problem has been solved, hence proved.
      2.2.2. If the mark belongs to some other agent *l*, then add agent *j* to the list of agents whose mark has already been seen and repeat allocation procedure for the k-sector with agent *l*'s mark as the first mark. Once an interval has been allocated to an agent, remove all marks of the agent. Remove all marks behind this interval. Other agents may lose at most

one mark. Thus we now have a (k-1)-sector problem where *k-1* agents have at least *k* marks and demarcating *k-1* intervals, hence proved.

This proves that the allocation procedure works for *n* agents, for any $n \geq 2$. Next we discuss the important features of this protocol.

### 4.2 Features

The protocol above has the following features:

1) The protocol is fair because each agent gets one of the pieces he demarcated for himself and that piece is worth at least $1/(n+1)$ of the whole cake.
2) It is unbiased because there is no mediator bias as seen in the linear version of the algorithm. Each agent can have its own start mark, rather than the mediator choosing the start mark.
3) Since each agent makes $n+1$ marks, the space complexity is $O(n^2)$. The protocol may make $n$ comparisons in the first iteration, $n-1$ comparisons in the next, and so on in the worst case scenario. Thus the time complexity is $O(n(n+1)/2)$.
4) One feature of this algorithm is that agents need not be ordered. Most other protocols assume that agents agree to order themselves in a certain way, but such an issue can be contentious in itself. For example, in the successive pairs algorithm agents might prefer to come later rather than earlier to avoid the labor of redividing their portions into ever smaller pieces in each iteration.

The algorithm is inefficient, because for *n* agents, each agent is expected to create $n+1$ intervals. When one of the intervals is allotted to the agent, it is actually getting $1/(n+1)$ of the whole cake. This inefficiency can however be reduced by joining together the wasted pieces and forming a sector. All the agents can then make $n+1$ marks demarcating *n* intervals in this sector. The sector algorithm can then be applied to allocate portions to each. This can be repeated until the wasted portion is negligible in every agent's opinion. The algorithm is therefore applicable only when resources are infinitely divisible into combinable portions.

Thus by reapplying the sector algorithm a finite number of times, this allocation procedure can get arbitrarily close to the $1/n^{th}$ allocation for each agent. This protocol assumes that the problem domain is infinitely divisible and combinable. In order to give an idea of the various types of problem domains that exist, refer to the following table.

**Table 1.** Various problem domain types

| Problem domain properties | Infinitely divisible | Combinable |
|---|---|---|
| A single row of seats in a theatre | -- | -- |
| Time scheduling | X | -- |
| Circular cake | X | X |

## 5 Conclusion and Discussion

We have presented a new protocol for allocating linear and circular resources, extending the classic cake cutting problem. This algorithm is applicable for *n* agents in general. It has been modified specially for the case of a circular resource. The proof

shows that an allotment chosen by the agent itself is guaranteed for each of the $n$ agents. In addition to this, the protocol is fair and unbiased, features that are highly desirable but generally difficult to achieve. However, the first allotment of $n$ pieces is inefficient, because in each agent's opinion only $1/(n+1)^{th}$ portion of the resource is received. To improve efficiency, we reapply the allotment procedure to the wasted portions of the resource. Agents can run the procedure a fixed number of times or until they all agree that the wasted portion is negligible and further division is unnecessary. The current procedure neither demands nor exposes the utility functions of individual agents. Although a mediator may be used to allot the various portions of the resource, the solution of the mediator can be verified, unlike the case of the moving knife solution. Hence disputes can be resolved easily. Also to be noted is that the mediator need not have any features, such as being unbiased. In fact the mediator can be one of the bidding agents. This is possible because the allocation procedure is completely algorithmic and does not depend on the subjectivity of the mediator. If any agent thinks the allocation was unfair, it can re-run the procedure to confirm the validity of the allocation. Unlike many other protocols, such as successive pairs or divide-and-conquer, our protocol does not require an implicit ordering of agents. This avoids any disputes as to which will be the first one to divide the resource. However, the space and time complexity of this procedure is relatively poor. Hence users will have to study the cost versus quality trade-offs to determine which protocol they find most suitable for their resource allocation needs. Among the issues that need to be looked into further are:

1. Improving the space- time complexity of the allocation procedure
2. Since the procedure requires a centralized mediator to run the allocation procedure, a distributed version of the allocation procedure could ease the load of the computation on the mediator.

## References

[1] Rosenchein, J.S., and Zlotkin, G. 1994. Rules of Encounter. London, England.: MIT Press.
[2] Davis, R. and Smith, R. January 1983. Negotiation as a Metaphor for Distributed Problem Solving. Artificial Intelligence 20(1): 63-109
[3] Steinhaus, H. 1948. The problem of fair division. Econometrica 16:101-104
[4] Brams, S. J., and Taylor, A. D. 1996. Fair Division: From cake-cutting to dispute resolution. Cambridge,UK.: Cambridge University Press.
[5] Biswas, A. and Sen, S. More than envy-free. In the Working Papers of the AAAI-99 Workshop on Negotiation: Settling Conflicts and Identifying Opportunities, 44-49. Menlo Park, CA: AAAI Press.
[6] Tasnadi, A. November 2003. A new proportional procedure for the n-person cake-cutting problem. Economics Bulletin 4:1-3.
[7] Robertson, J., and Webb, W. 1998. Cake-Cutting Algorithms: Be Fair if You Can. Nattick, MA: A.K.Peters.
[8] Huhns, M.N. and Malhotra, A.K. July 1999. Negotiating for Goods and Services. IEEE Internet Computing, 3(4): 97-99.
[9] Stewart, I. December 1998. Mathematical Recreations: Your Half 's Bigger Than My Half!. Scientific American 112-114.