International Journal of Cooperative Information Systems

World Scientific
www.worldscientific.com

A PROCEDURE FOR THE ALLOCATION OF TWO-DIMENSIONAL RESOURCES IN A MULTIAGENT SYSTEM

KARTHIK IYER

Department of Computer Science and Engineering
University of South Carolina, Columbia, SC 29208, USA
iyerk0@gmail.com

MICHAEL N. HUHNS

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

This paper presents a constructive solution to the classic problem of land division. It is the first solution that enables the allocation of higher-dimensional resources without degenerating them first into a series of one-dimensional resource allocation problems. We base our allocation procedure on the topology of overlaps among the regions of interest of different agents. Our result is an algorithm suitable for computer implementation, unlike earlier ones that were only existential in nature. It uses the notion of degree of partial overlap to create a sufficiency condition for the existence of a solution, and proposes a procedure to find the overlaps in such a case. The proposed solution is fair, strategy-proof, non-existential, and does not explicitly need the resource to be measurable. The agents do not have to reveal their private utility functions. We extend our earlier result for one-dimensional resource allocation to this two-dimensional one and explain the distinctive issues involved.

Keywords: agent negotiation, resource allocation, cake cutting, land division.

1. Introduction

Multiagent systems have emerged as an important area of research in the field of distributed computing. They are being used for numerous real world applications, such as operating distributed sensor networks, automating auctions, and allocating online as well as physical resources. An important feature of multiagent systems is that the agents behave autonomously. Autonomy means that agents have a high degree of freedom and choice in initiating actions on their own, planning goals for themselves, and taking actions to achieve the goals, all while considering their own self interest. Allocating resources among the entities in a multiagent application is an important requirement. The need for resource allocation may be an end in itself for a particular multiagent system. An example of this is an electronic marketplace, where there might be a mixture of agents and humans trading goods and services. Alternatively, the need for resource allocation may also be a means to an end. Such cases arise when agents consume a common resource in order to achieve a collective goal. In the case of distributed sensor networks[1], the problem might be the allocation of portions of the frequency spectrum to different sensors that are collectively tracking a target. It is possible that the designer of the

multiagent system can specify the details of the resource allocation procedure. But the drawback is that the procedure might not adapt dynamically to changing environmental conditions or agent preferences. Scalability of the design is also a problem, as is the fact that there is a single point of failure.

Multiagent negotiation is one of the important mechanisms that can be used to provide for the allocation of resources. Negotiation[2] is "when two parties strike a deal through argumentation or arbitration for the mutual good of both". Davis and Smith[3] define negotiation as "A discussion in which interested parties exchange information and come to and agreement." 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 protocol by the agents for their negotiation should be immune to information hiding and lying by the 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: the protocol. In addition, protocols can be evaluated on various criteria such as fairness, envy-freeness, equitability, and efficiency[4]. Brams and Taylor have discussed extensively various procedures that can be used to allocate resources. They show that it is generally difficult for any given procedure to fulfill more than two of the above mentioned criteria. These criteria are by no means exhaustive, but may be taken as an initial test of the allocation procedure that is being proposed. For example, other criteria[5] that can used to evaluate protocols are: simplicity, computational complexity, and verifiability.

A protocol is said to be *verifiable* if the allocation of the resource is invariant to the bias of the mediator or agent. The issue of verifiability was encountered when the authors earlier described a procedure for one-dimensional cake cutting[6]. In that paper, a multiagent negotiation protocol was presented that divides a *one-dimensional resource* in a fair and unbiased manner among *n* agents. We provided proof that if the agents followed the protocol, then it is possible to have a fair and unbiased allocation of the resource. In this paper, the authors propose a negotiation protocol and procedure for the allocation of a *two-dimensional resource*. The negotiation protocol in its current form is a one-shot protocol. This is less common than the typical iterative forms of negotiation. At the end of the negotiation, one of the agents volunteers to act as a mediator and executes the procedure. Based on the computation of agent preferences, there is one of two outcomes:

(i) The procedure is able to find a solution and all agents get a fair deal.
(ii) The procedure is unable to find a solution and all agents receive the conflict deal, i.e., no agent receives any part of the resource.

The salient point to be noted here is that if the agent playing the role of the mediator is biased, its attempt to manipulate the results in its favor will be detected. This is because the procedure we put forth does not involve any subjective evaluations. It is an algorithmic method based on the preferences that various agents have indicated. Hence the results of this method are verifiable to any agent who wants to check them. Thus the

role of the mediator need not be performed by an outsider with special characteristics (such as being unbiased). The resource allocation can be handled by the agents themselves. Another point that needs to be mentioned is that we do not make any assumptions about the nature of the utility functions that the agents can use. This is because we do not make any explicit utility comparisons. Hence, utility functions of individual agents can be of any form, as the procedure allocates individual portions based on the topology of the agent preferences. We use the problem domain of cake cutting as a running example to explain our procedure and discuss various issues that need to be addressed for allocating such resources. Before starting the discussion on how two-dimensional resources are to be allocated, we describe the context for such problems.

2. Background

2.1.1. *Cake cutting:*

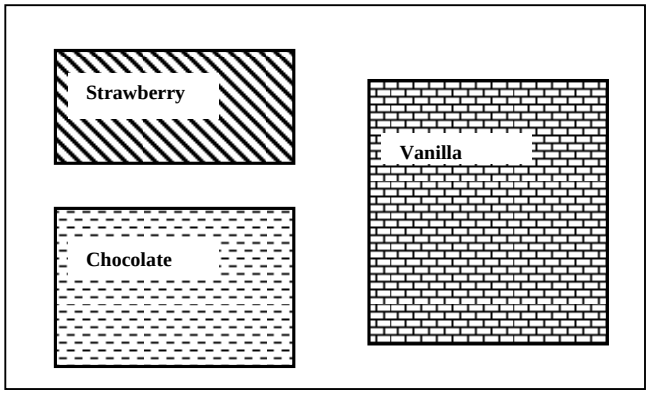

Fig. 1. Various flavors of icing on a cake

Consider a rectangular cake with three flavors of icing to be distributed among three agents as shown in Fig. 1. A typical one-dimensional cake cutting procedure like moving-knife[7] can be applied to allocate portions of the cake to each of the agents in a fair manner. However, if the agents have mutually exclusive and strong preferences for the different icings, the allocation will not be efficient. A simple example of such a case would be when agent 1 values only the chocolate icing, while the rest of the cake is useless to it. Agents 2 and 3 value the vanilla and strawberry icings, respectively, in a similar fashion. Efficiency in this case is meant in the Pareto optimal sense. The agents will be forced to translate their two-dimensional preferences to linear marks on the X-axis (or Y-axis), which is non-trivial. If a protocol is able to take into account agents' preferences along both dimensions, then clearly more efficient solutions can be obtained.

2.1.2. *Land distribution:*

Another problem domain that demands two-dimensional solutions is land division (Fig. 2). Say a tract of land has various assets, such as an oil field, fertile farmland, and some hilly areas, and the land needs to be distributed among three agents. Each agent values the resources differently and in order that they each get the most value, it should be possible to respect their preferences in two dimensions.

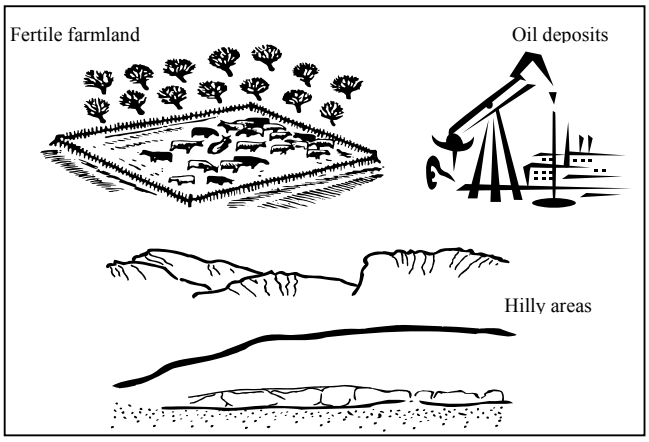

Fig. 2. Land with assets.

2.1.3. *Allocating satellite imaging resources*

Manufacture and launch of satellites is an expensive and risky proposition. It is preferable to share the risk and expense of this operation, while giving up some control on the satellite resources. Say, if the satellite is capable of mapping earth terrain, there could be multiple agencies that require such a service. Suppose the satellite is capable of mapping a patch of the earth's surface of constant width along the earth's latitude but arbitrary longitudinal height. We could have agents [representing these agencies] bid for images of the earth's surface which specify the latitude interval and the time of day in which the image should be taken. This makes the allocation of satellite resource a two dimensional resource allocation problem. The allocation of satellite resources has been previously looked at [8] and is discussed further in the related work section.

2.1.4. *Allocating radio spectrum:*

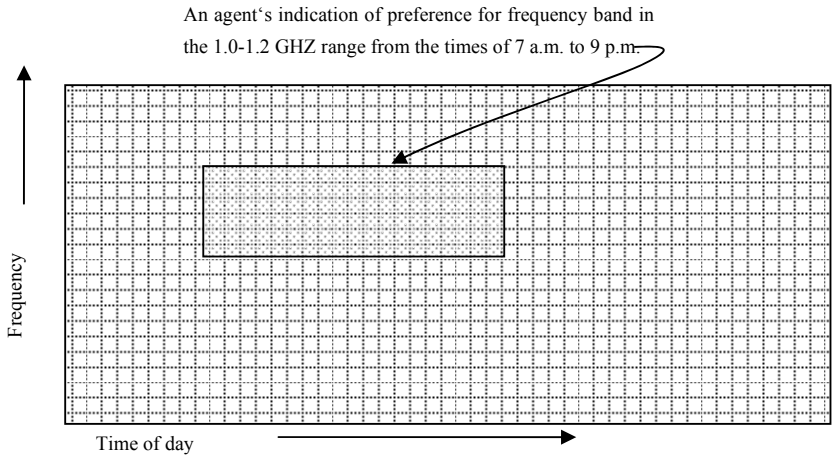

Fig. 3. Allocating frequency spectrum.

One can envision a future auction where the FCC parcels out not only frequencies but also what times of the day the frequencies are available. This will give higher efficiency, as agents will choose the most preferable positions and the remainder can be reused by the FCC for other purposes. Thus the two independent dimensions in this scenario are time and frequency. For example, a particular radio frequency band in the 1.0-1.2 GHz range may be allotted from 7am to 9pm for broadcasting traffic information (Fig. 3), while it may be used for downloading data onto wireless devices during off-peak hours.

The examples mentioned above show that there are many real world problems whose solutions in the two-dimensional domain may prove to be far more worthwhile than using traditional one-dimensional solutions, like the moving knife procedure. Essentially any real world resource, having two independent dimensions and needing to be allocated, will be appropriate for the procedure we describe below.

2.2. *Case for higher efficiency*

Now that the problem space has been illustrated by various examples, a few points about efficiency are worth mentioning. Consider land division once again. So far we have assumed that all portions of the land have positive utility for all agents. Relaxing this assumption, it is possible that portions of land can have negative utility as well. For example, part of the land may be forest that might need to be maintained as per law or be taxed, but accrue no benefit to owners. Assume that the agents view their utility as shown in Fig. 4a.

Agent 1 thinks that the portion allocated by the solid rectangle has positive utility, while the remaining land has negative utility. The same argument can be applied to agent 2 with respect to the dotted rectangle. Now if we use the moving knife technique (with the knife handled by an external unbiased mediator) along one dimension, say the X-axis as per the protocol, each agent is forced to include negative utility areas as well as

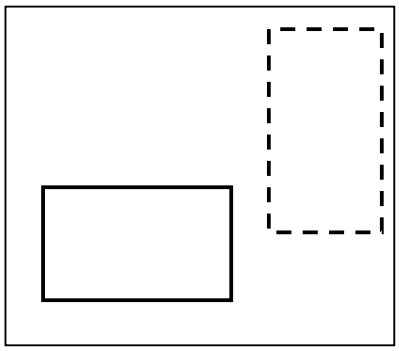

Fig. 4a. Resources with negative utilities.

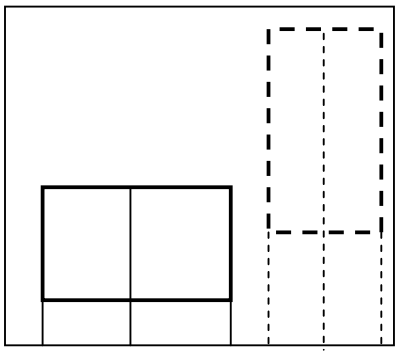

Fig. 4b. Allocating using modified moving knife.

positive ones. Hence, both agents might get low, possibly negative utilities, although their allocations will be fair and envy-free. *Fairness* and *envy-freeness* are criteria used in the cake-cutting domain to judge the effectiveness of allocation procedures. An allocation procedure is called *fair* if it distributes a resource among *n* agents such that every agent values its portion as exactly *1/n* of the total value of the resource. An allocation procedure is called *envy-free* if every agent values its portion at least as much as the portions allocated to other agents. Thus, envy-freeness is a stricter condition than fairness and there are fewer procedures that are envy-free than procedures that are fair. We can modify the moving-knife protocol so that agents can exclude unwanted regions. But the efficiency will still be low, because the agents cannot completely eliminate negative utility regions. Using the modified moving-knife protocol, the agents' divisions will appear as in Fig. 4b. Despite being more efficient than the original moving knife, this modified procedure is still undesirable, because agents are forced to include negative utility areas (below the rectangles) to get their share. Obviously, if the agents were able to apportion in two dimensions, the efficiency of the overall solution could be higher. Clearly there is a need for a two-dimensional allocation procedure.

In the next section we review the literature to see how the two-dimensional cake-cutting problem has been approached in the past. The third section describes our allocation procedure in detail. Finally, we present our conclusions and show what scope exists for future work.

2.3. *Solving the one-dimensional resource allocation problem*

We briefly mention the salient points of the solving the one-dimensional resource allocation based on the topology of overlaps [6]. Consider *n* agents which desire to distribute a linear resource (say slices of rectangular cake) among themselves. 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.

Theorem 2.3.1 If there are *n* agents and each agent makes *n-1* marks, creating *n* portions of a linear cake, then the procedure 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:

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 $n \geq 2$. □

The procedure / proof for allocating a linear resource in the manner described above is fair and unbiased. It does not require the presence of a mediator. This procedure will serve as the starting point for the solution of two-dimensional resource allocation problem discussed in this paper.

3. Related Work

The existing literature on resource allocation can be roughly classified into three categories: resource allocation in multiagent systems, the one-dimensional cake-cutting problem and the two-dimensional land-division problem. We discuss them individually in the following sections.

3.1. *Multiagent resource allocation*

The resource allocation problem has been a domain of interest in the MAS community. Agents acting in a common environment may find the need to share resources due to their limited availability. Examples of resource allocation problems in MAS include, robots on Mars belonging to autonomous agencies (NASA and ESA) which negotiate to share precious equipment time [9], agents which require satellite imaging resources [8, 10], multiagent sensor networks which cooperate to track a common target and coordinate usage of scarce resources such as radio frequency spectrum [11].

The solutions proposed for these problem domains are distinguished by the scope, nature, framework and theory used to solve them. The treatment of the 'Robots on Mars' problem by Sarit Kraus has the following characteristics:

(i) The negotiation protocol is bilateral. Only two agents are assumed to be part of the negotiation.

(ii) Agents are assumed to have utility functions which reflect the constraints of time, resource value, agreement cost, opt out cost etc. This limits the kind of utility functions that can be used by the agents.

(iii) Agents have complete information about each other's utility functions and beliefs.

(iv) The negotiation protocol is iterative (although Sarit Kraus has proven to finish it in two steps for the given set of conditions).

(v) A solution is guaranteed, provided agents adhere to the constraints on beliefs and utility functions that can be used.

The 'Robots on Mars' problem domain put forth by Sarit Karus can be adapted to use the solutions proposed in this paper. Suppose NASA and ESA are negotiating over the use of the common communications line between Earth and Mars. While Kraus defines time of usage as one dimension, we can add frequency bandwidth as another dimension where resource usage can be shared. This transforms the original example into a two dimensional problem making it amenable to our solution approach. The problem can be visualized by the radio frequency example shown in Fig. 3. Our proposed solution to solving two-dimensional resource allocation problems has the following features in contrast to the approach taken in by Sarit Kraus:

(i) The negotiation protocol is multilateral. If ISRO [Indian space research organization], decides to request common resources, our protocol will scale to add new agents.

(ii) The solution is agnostic of the kinds of utility functions that agents will use. Hence agents are not required to adhere to specific types of utility functions in order to participate in the negotiation.

(iii) Agents need not have any information on the other agents' utility functions and beliefs. Thus agents can compete for resources without revealing their true utility functions or beliefs. Privacy is afforded.

(iv) The negotiation protocol is one-shot. Our protocol is more akin to the sealed bid type auctions than the open cry type iterative auctions.

(v) Given the set of agent 'bids' [in the form of rectangles on the two-dimensional resource], there is no guarantee of a solution. Unlike the approach taken by Sarit Kraus, it cannot be guaranteed that a solution exists for every combination of overlap among agent rectangles. See Section 5. for a more detailed analysis.

In summary, it can be stated that while our approach allows for a negotiation that is more scalable and flexible, it cannot offer the guarantee of a solution for all possible scenarios of agent interaction.

In the background section, we mentioned the example of sharing resources on an earth observing satellite. This example is attributable to [8] who discuss the issue of allocating imaging resources in a multiagent system. The authors mention of two main approaches to the allocation problem:

- Decentralized negotiation, "where the agents together agree with game or negotiation rules, and then act freely for their own interest, respecting the common rules.", and
- Centralized arbitration procedures, "where an arbitrator, which is assumed to be fair and to act according to principles that have been accepted by all the agents, decides about the equitable allocation"

Lemaître et al prefer the centralized over the decentralized approach for the following reasons:

(i) Privacy: Agents do not have to reveal their true preferences to each other. They only need to communicate with the central arbitrator.

(ii) Scalablity: Negotiation may be difficult to manage if the number of requests [for the imaging resource] is high.

(iii) Time: If time to conduct the allocation process is short, decentralized negotiation may be too time consuming.

(iv) Efficiency: The efficiency of the allocation may be poor in a decentralized approach.

The authors propose solutions that balance the dual criteria of equity and efficiency. Below, we describe how this contrasts to the approach taken in this paper:

(i) Unlike Lemaitre et al, this paper does not use the utilitarianism framework to model agent preferences. Rather our solution exploits the topology of overlaps to arrive at solution

(ii) Privacy: Our approach does away with the necessity of an 'independent' arbitrator/mediator. This ensures private agent information is not communicated to third parties.

(iii) Scalabity: Like Lemaitre et al., our procedure is centralized. However we believe decentralized procedures and protocols that are well designed tend to scale better than ones that involve communication and computation with / by a single entity.

(iv) Time: A well designed decentralized protocol / procedure may actually perform better than a centralized solution. Our approach requires that agents submit their portions to one mediator - which is a parallel process – will be constant with respect to the number of agents. However, the allocation procedure itself is centralized and time to arrive at a solution is of the order of $O(n^2)$.

(v) Efficiency: Decentralized procedures can be efficient in the pareto optimal sense. However, they may be sub-optimal in the social welfare sense. Lemaitre et al propose solutions that maximize social welfare.

The solutions do not allow for rational agents and free behavior, which could be a drawback in open platforms like the Web, where it is difficult to enforce socially optimal (though not individually rational) solutions.

Combinatorial auctions (CAs) have been extensively studied as a solution to the multiagent resource allocation problem. The problem consists of auctioning multiple

(possibly heterogeneous) items to competing bidders (or agents representing them). The bidders have different preferences over bundles (representing a combination) of goods. There has been a lot of focus on issues relating to:

1. Design of a bidding language that balances a need for expressiveness, succinctness, and computational complexity
2. Design of a winner determination mechanism that achieves a satisfactory trade-off between computational complexity and fitness to a desired social function.

CAs have been built up on the extensive base of auction theory available in economics. It also involves critical inputs from operations research (for optimization over alternative bundles) and computer science. CAs exploit the complementary property of bundles of goods. For example, an agent will have greater utility for a pair of gloves sold as a bundle than each glove sold independently. A traditional single-unit auction does not take into account the inter-dependency of goods in the bundle and is less efficient in extracting the best possible value for the auctioneers. A well known example of a CA is the Vickery-Clarke-Groves (VCG) mechanism[12]. VCG is used as a benchmark to compare alternative proposals for CA mechanisms since it manages to achieve efficiency and truthful bidding. This is achieved by charging each agent the social opportunity cost of its winnings, i.e. each agent compensates for the loss caused to other agents due its addition into the system. Surprisingly, despite such good criteria VCG has rarely been adopted in the real world [12, 13]. The following issues have formed critical stumbling blocks towards adoption of VCG:

- Agents / Bidders do not want their private valuation to be known publicly due to the incentive compatible property of the mechanism.
- The exponential growth of effort related to bid preparation and communication.
- The winner determination problem is NP-complete.
- The mechanism may bring in low revenues to the seller.

One major implicit requirement of CAs is that agents express their preferences over bundles in terms of utility functions. This is not a requirement in our case. In contrast to CAs, we assume the resource to be homogenous and infinitely divisible. We do not account for complementarities between various portions of the resource. On the other hand CA mechanisms do not account for the dimensionality of the resource as has been done in this paper.

3.2. *One-dimensional resource sharing*

The cake-cutting problem is a well known example of resource sharing among rational agents. Most of the literature that exists on the one-dimensional cake-cutting problem has been contributed by mathematicians. The classical solution for the two-person case, divide-and-choose, was first proposed by Steinhaus[14]. This solution, where one person divides a 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 the solution to n agents. One of the solutions[7] 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, because an agent might evaluate all of the pieces—distributed after it was allocated a piece—and decide that one or more were larger than the piece it got. Second, it requires the presence of an unbiased external 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 it 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 arrive later than that of an agent with a faster connection and may consequently lose the bid (the moving knife is similar to the open-cry descending-bid Dutch auction). Third, the moving-knife protocol is also not Pareto optimal. A scheme[15] 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 three 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[16] 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[17] 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 two 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.

Stewart[18] and Huhns and Malhotra[19] discuss how to divide a strip of property along a coastline. However this is a specific solution applicable to only three agents. The result was further extended by the authors to be applicable to *n* agents[6]. The negotiation protocol described by these papers has useful features like: absence of synchronization problems, a mediator is not required and utility functions are not needed for the allocation

procedure. However all of them assume the resource to be one-dimensional. In this paper we extend some of the ideas to allocate a two-dimensional resource among agents.

3.3. *Two-dimensional resource sharing*

The two-dimensional resource allocation problem has been tackled by researchers from diverse fields who have encountered it in various forms. Our literature survey describes researchers using the examples of dividing pizzas, cakes, or land to discuss the existence of fair solutions. Generally, however, the problems tend to have characteristics distinct from the one we are trying to solve. We have not come across any constructive (algorithmic) solutions so far to the generic land-division problem. All the papers have offered existential solutions to qualified versions of the problem.

Hill[20] was one of the first to tackle fairness issues in land division. The problem domain was qualified, because it attempted to allocate portions of land to countries that shared a border with it such that each country received a portion connected with itself. He extends a non-constructive result proposed by Dubins and Spanier[21].

The problem with[21] was that although it could create fair shares for all agents, the pieces of land might not be in the neighborhood of the country to whom the portion is allocated. Hill's solution is to create thin strips of land that connect the isolated portions to the country it is allotted. The strips created in this manner do not intersect, because any pair of points in the set is assumed to be path-connected. There are some drawbacks to this approach. It can happen in reality that the strips that get created may be exceedingly thin so as to render the solution useless. For example, consider two countries that contest the land bordering them. It may be unacceptable for one country to have a single road connected to its allocated portion surrounded by the enemy portions. Besides, this result does not explicitly provide a procedure for enabling such an allocation.

Beck[22] proposes a semi-constructive result that improves upon Hill's paper. In order to do this he constructs a unit disc $|z|<1$ that is a homeomorphic map of D, the disputed territory. It is also assumed that individual circles and radial lines in the disc have zero measure. Next, various agents place "bids" in an "auction" by submitting the smallest radius that will enclose a disc, which is valued at $1/n$ for that particular agent. The smallest such radius among the various bids is picked and the disc is awarded to that agent. In order that the portion may be connected to the agent's territory, a small wedge from the unit disc is also allotted. Then successive agents keep trimming the wedge so that the total piece allocated is less than $1/n$ for each of the remaining agents. In order that allotted pieces do not end up having strips that break up other agents' allotments, a complicated procedure of secondary auctions involving rebidding of the same piece along with guidelines for trimming and growing the pieces make it difficult to implement any such scheme. Besides, the solution is still hobbled by the issues affecting Hill's proposition. In fact, there may be additional implementation issues, like getting the appropriate functions for the conformal maps, which make it just as impractical as Hill's procedure. These functions depend on the shape of the land and hence have to be tailor

made for each problem individually. Beck only proves the existence of such functions and does not specify how they can be found.

Webb[23] provides a combinatorial algorithm for the fair-border problem based on Hill's existence results. The algorithm is recursive in nature and works as follows: a region R is bordered on all sides by n countries, $C_1, C_2, \ldots, C_n$. Each country has its own evaluation of the piece of land and draws a region R_i adjacent to itself, such that it is

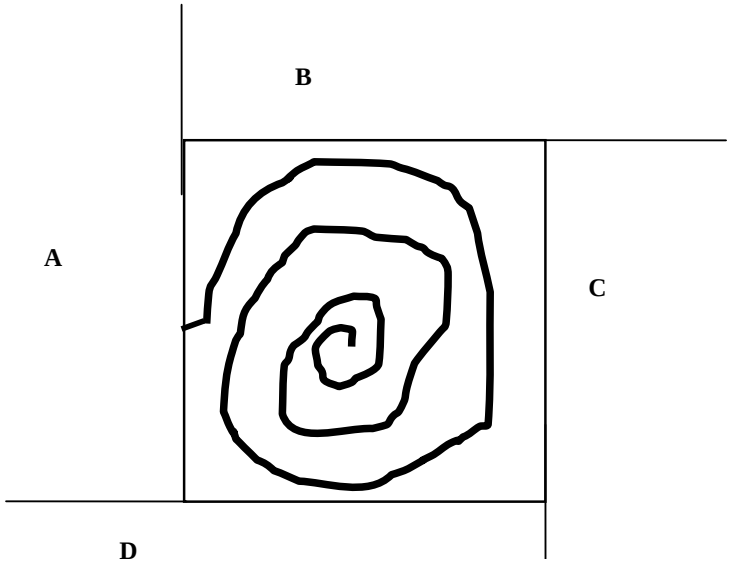

Fig. 5: Agent A's demarcation of its share of the region. The thick line represents the land allocated to A

valued at $1/n$ by its own measure. Then each country in turn (that values R_i greater than $1/n$) trims off a piece of R_i so as to disconnect it from C_i. The country that trims it last gets to keep the modified R_i. This region is attached to the allocated country by a strip of land small enough to be negligible to the others. The remaining land is then redistributed among the remaining agents similarly. The flexibility given by this algorithm in allowing agents to shape the region of interest as they like is also its drawback. If the shape of the land an agent gets is included in the fairness criterion, then earlier agents get a better deal than do later ones. This is because the later agents will have to "draw around" the regions allocated to earlier agents and the effective shape of the land they get might make it worthless for any use. In Fig. 5, consider a square-shaped region that is surrounded on four sides by four countries. Suppose agent A got the first chance to draw its region and it does so in the manner shown in Fig. 5. It can be easily seen how the other agents' allotments are placed at a disadvantage.

Hill's paper provides the basis for another result[24] that proves fair-border solutions exist, even if the utilities are concave. Utilities tend to be concave rather than additive in

the real world. They take into account the fact that the marginal utility of a good decreases as more of it is consumed, due to satiation. Classic cake-cutting algorithms have always assumed that utilities were some sort of probability measure[25]. Consider a measurable space Σ as the resource being allocated. If A and B are two portions of the resource and v is the utility function of the agent, then we have the following property:

$$v(A \bigcup B) = v(A) + v(B) \quad \textit{for all disjoint } A,B \in \sum$$

On the other hand, a concave capacity has the property:

$$v(A \bigcup B) \leq v(A) + v(B) \quad \textit{for all disjoint } A,B \in \sum$$

Concave utility measures can be generated by wrapping a probability measure with a strictly increasing "utility" function u:

$$v(A) = u(\mu(A))$$

The authors were able to prove that the Dubins and Spanier solution holds in the concave domain too. Similarly they also showed that Hill's fair border solution is true for concave utilities. The authors do not, however, mention some benefits of the concave domain. Concave functions are useful because they connect efficiency and fairness. If the valuation function is additive, one of the efficient allocations is to give the whole cake to one agent. Thus efficiency need not induce fairness. If the valuation function is concave, however, it is more efficient to give pieces of the cake to different agents, since the whole cake allocated to one agent is less valuable (in terms of social welfare) than the sum of the values that each agent attributes to the piece allocated to them. Thus an efficient algorithm will also tend to be fairer to all agents. The authors were not able to determine if the same results would hold if the domain was subadditive rather than concave, however. (Concave domains are a subset of subadditive domains.) Finally, the biggest drawback of the paper is that there is no constructive algorithm offered to actually apportion resources.

A novel approach[26] to resource allocation is to model the preference relations over the set of arcs on a circular cake geometrically. The paper considers the resource to be an infinitely divisible, non-homogeneous and atomless one-dimensional continuum whose end-points are topologically identified. It then proceeds to partition the resource into intervals. It is assumed that the recipients are equipped with preferences over intervals that are additive, continuous, and monotonic with respect to interval inclusion. The assumptions about the domain as well as the preference relations are quite similar to the ones in our earlier paper. There are, however, a number of restrictions on the preferences:

- The preferences should be smooth, i.e., they should have continuously differentiable numerical representations. While this is a simple requirement, it is not trivial to have a preference that is "kink-free" due to the circular shape of the cake.
- Preferences should be convex. Again this is a non-trivial condition as convexity depends on the choice of origins.

The paper places emphasis on egalitarian-equivalence solutions rather than envy-free solutions, as the former tend to have more Pareto-efficient solutions than the latter. Egalitarian-equivalence is a distributional requirement that states that there exists a reference consumption that each agent finds indifferent to its own consumption. Egalitarian-equivalence is, however, a weaker condition than envy-freeness. Thus all envy-free solutions are egalitarian-equivalent, while the converse is not true. Since in many domains the set of efficient solutions are not envy free, egalitarian-equivalence is used as a "compromise" fairness requirement. Other limitations in this paper are that the analysis of preference relations over the union of two or more intervals becomes quite complex. Trying to model preference relations over the union of at most two intervals requires four-dimensional space. This paper is limited to the domain of one-dimensional resource allocation and does not offer a constructive algorithm for allocating intervals. One of the salient points is that explicit utility functions are not required. Finally, the paper also discusses strategy-proofness of their solutions. A rule is strategy-proof if no agent ever has an incentive to misrepresent its preferences. Unfortunately, for the case of $n=2$ agents, it turns out that any Pareto optimal solution that is strategy-proof is also dictatorial, i.e., any one agent will always be able to get its most preferred choice of interval irrespective of other agents' choices.

Chambers[27] discusses the various normative properties that land division rules should have based on the principle of utilitarianism. Utilitarianism[2821] is a principle in the theory of ethics that prescribes the quantitative maximization of beneficial consequences for a population. The paper studies the circumstances in which rules can satisfy the division independence property. If the union of the portions allocated to an agent from various subparcels of a parcel is identical to the portion allocated from the initial parcel, then the rule based on which the allocation is made is said to be division independent. This property may be attractive in case a large piece of resource needs to be broken down into smaller pieces in order to make the allocation problem more tractable. Alternatively, if the quantity of a resource keeps increasing over time and the allocation needs to be done over each additional portion incrementally, a rule that is division independent is desirable. The paper goes on to show that any rule that satisfies the above property, along with a few other normative properties like independence of infeasible land and efficiency, is a subrule of the weighted utilitarian rule. This is useful, since utilitarian social welfare functions have been well studied in the literature and can be used to get information about the system. However, the division independence requirement is too constraining and invalidates many other useful properties. For example, division independence is mutually exclusive with the "Strong positive treatment of equals" property. This is discouraging, because the above property is a necessary condition for fair solutions to exist. This is a feature of weighted utilitarian rules in general, and if the rules are required to be scale invariant as well (i.e., the portions allocated by the rule are independent of the scale of the utility function), then only dictatorial rules are possible. Thus, ironically, the social welfare approach ends up empowering one individual at the expense of the rest. The other issues are:

- Division independence is meaningful only in the context of additive functions. If the utilities are concave, then utility of the union of subparcels will be less than the sum of the utilities of the individual subparcels and, hence, there is no notion of division independence.
- The weighted utilitarian rules require the cardinal comparison of utilities to work. This assumption is not tenable in real world situations.
- The definition of utilitarianism is too vague and cannot be fixed by an objective function.

It is thus clear that division independence is too tight of a constraint to be of any significant use in evaluating competing allocation algorithms.

The literature survey shows the multi-faceted nature of the land-division problem. Researchers have focused on various aspects of the problem to get a better handle on how it can be resolved. However, each of their approaches also has limitations, either due to the approach taken (measure-theoretic vs. combinatorial) or due to the nature of the domain. This shows that the problem is complex and newer approaches might help. In the next section, we begin a description of the details of our approach and how it stands up in comparison to ideas in the existing literature.

4. Dividing a two-dimensional resource among *n* agents

In this section, we present the protocol that the agents must follow in order to get a fair share of a two-dimensional resource. We propose two different resource allocation procedures that are applicable if agent preferences fulfill some conditions. If condition A is true, then we map the problem into a one-dimensional resource allocation problem and use results from the one-dimensional resource allocation procedure to allocate portions to the agents. The one-dimensional resource allocation procedure is presented in our earlier paper[5]. Since condition A can be restrictive in how agents mark their preferences, a more flexible condition B is proposed. If condition B is true, then we propose a novel procedure that allocates resources based on the topology of the overlaps of agent preferences. An example showing resource allocation among three competing agents illuminates the salient points of the procedure. Then a proof is presented for this procedure guaranteeing a solution for any *n* number of agents. The assumptions required for this procedure, as well its features, are discussed in greater detail in Sec. 4.

4.1. *Protocol*

The protocol proposed here is a one-shot form of negotiation. At the end of the negotiation either the procedure is able to find a solution and every agent gets a fair share of the resource, or the procedure is unable to find a solution and a *conflict deal* is reached, i.e., no agent gets any part of the resource. We describe the simple protocol below:

If there are *n* agents,

1. Then each agent will create *n* portions of the resource, all of which will be equal by its valuation.

2. The allocations will be in the form of rectangles. Other polygons are not allowed.
3. The portions created by a particular agent should not overlap.

If all agents have followed the protocol, then a solution is guaranteed if one of condition A or B is fulfilled:

Condition A

1. For every agent, the rectangles marked out by a particular agent do not overlap on at least one axis.
2. The above condition is fulfilled by all agents on the same axis.

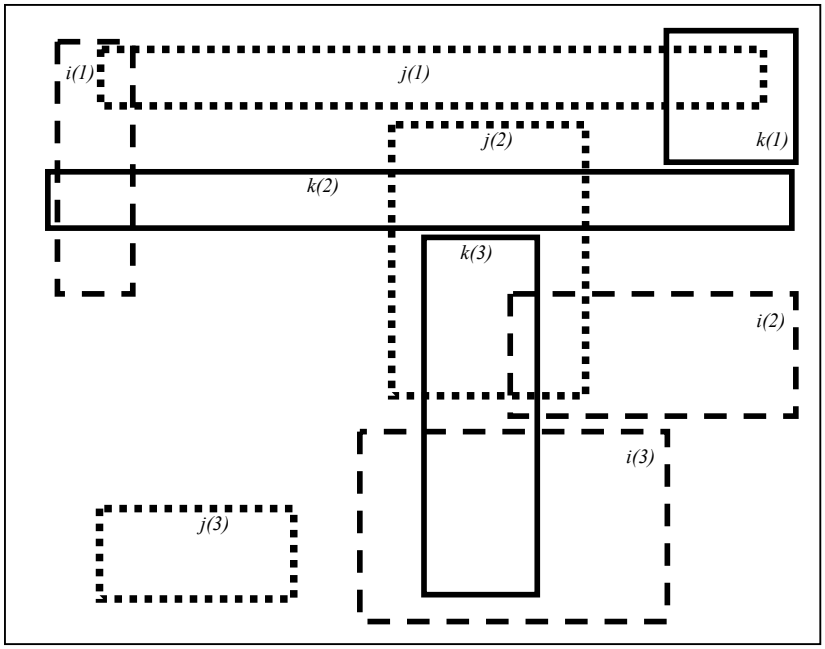

Fig. 6: Agents i, j, and k mark portions that fulfill conditions A1 and A2. The labels of the rectangles denote their respective owners.

We present the procedure for allocating the resource by a motivating example. Consider three agents, labeled *i, j,* and *k,* each marking three rectangles of equal value (by their own evaluation). Without losing generality, we assume that the coordinates do not overlap on the Y-axis (Condition A2). See Fig. 6.

4.2. *Procedure given condition A*

We show how the procedure works for the Y-axis; however, it is applicable on any axis.

1. Project the rectangles onto the Y-axis. These will now look like closed intervals on the Y-axis. Refer to Fig. 7

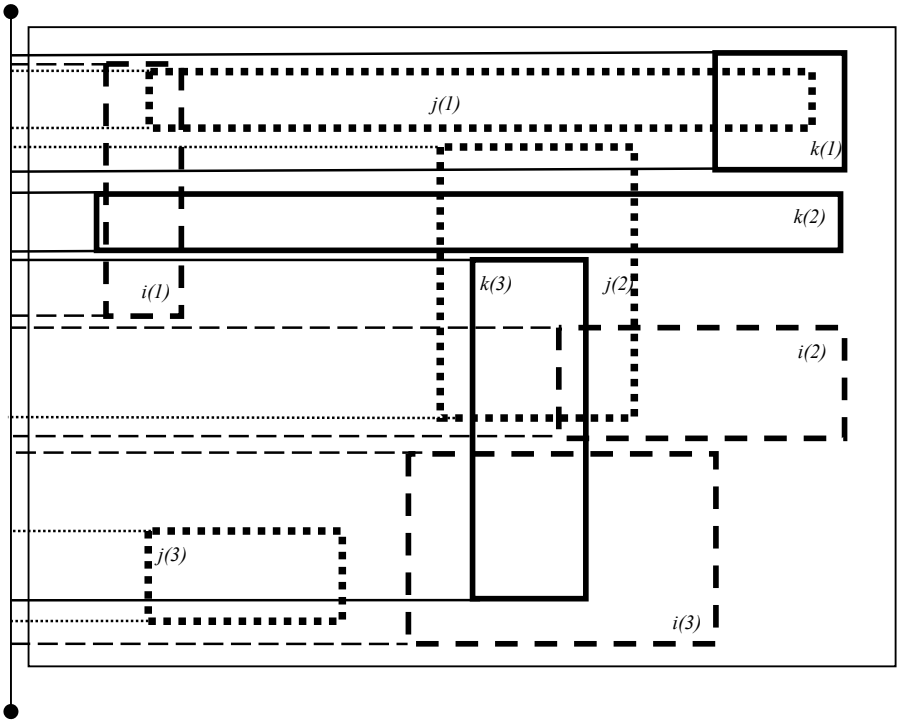

Fig. 7: Projections of the agents' portions onto the Y-axis.

2. Now moving from bottom to top, extend the start point of the interval of each agent to the end point of the previous interval (belonging to the same agent).
3. The first and last mark (when moving from bottom to top) of every agent is extended to the boundary of the resource. Refer to Fig. 8a and Fig. 8b.
4. We have thus transformed this problem into the one-dimensional resource allocation problem. The procedure for allocating resources in a one-dimensional case was presented in our previous paper. This procedure can be applied to obtain a fair allocation for all agents. Proceeding bottom-to-top (an arbitrary choice) and moving along the Y-axis, it is now possible to guarantee each agent will get a rectangle that belongs to itself.

The two-dimensional resource allocation problem has now been converted to a one-dimensional resource allocation problem. It can be solved by the procedure described in section 2.3.

4.3. *Features of procedure for condition A*

The above procedure extends intervals. This is done in order to map the two-dimensional problem into the one-dimensional cake cutting problem. We make use of the property that if the Y-coordinates of the rectangles made by a particular agent do not overlap, then the rectangles do not overlap at all (even if the coordinates on the other axis overlap). By extending the intervals we make sure that the whole Y-axis has been exhaustively used up. Now the problem is equivalent to the one-dimensional case, which has been tackled in our earlier paper. Extending the intervals of the agents does not reduce the value of the

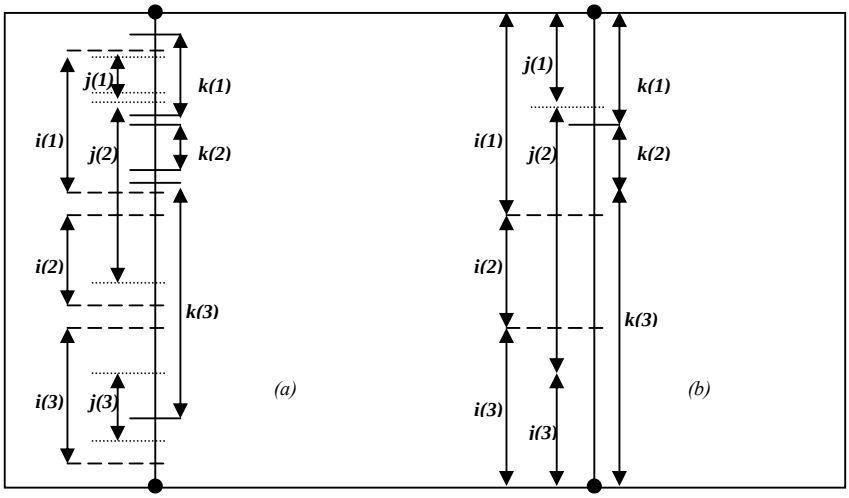

Fig. 8: Transforming the two-dimensional problem into the one-dimensional case. The start mark of i(3), j(3) and k(3) are extended to the lower boundary of the resource. The end mark of i(1), j(1) and k(1) are extended to the upper boundary of resource. The direction of movement is from bottom to top.

portions allocated to them, as this is effectively like padding zero utility regions to the agents' rectangles.

Who executes the procedure? As we mentioned earlier, any one of the participating agents can volunteer to perform the role of a mediator and execute the procedure. If an agent (which was not the mediator) is concerned about the solution being unbiased, it can simply take the record of agent rectangles and run the procedure against them itself. The procedure is deterministic and will deliver the same results every time it is run. This is the main benefit of our allocation procedure as compared to existing ones, such as the moving-knife method. The apportioning of resources can be *verified* by any agent who wishes to do so. Thus the role of mediation does not require external entities with special characteristics, such as impartiality. Due to this, the agents can resolve the allocation issue among themselves, which in turn protects the private information of the participants. A real example that illustrates such a scenario well is a spy satellite whose imaging services need to be allocated to different intelligence agencies. The agencies

competing for the resources need not employ an external arbitrator, thereby mitigating the risk of leaking sensitive information, while simultaneously arriving at a solution that is fair to all. Condition A2 is restrictive. In order to give more flexibility to agents to mark out their rectangles, we can modify the condition and restate it as follows:

Condition B
Each rectangle must have a *degree of partial overlap* at most equal to one.

Condition B allows the agents more flexibility than Condition A. The agents can mark rectangles anywhere. Note that any two intervals belonging to a particular agent may overlap either on the X-axis or the Y-axis, but not both (If this happens, then rectangles belonging to the same agent will overlap, which violates condition 3 of the protocol). Once all the agent preferences (i.e., rectangles) are submitted to the mediator (who could be one of the participating agents), then the mediator checks that condition B is true. If the condition is fulfilled, then the mediator can guarantee that a solution will be found.

4.4. *Degree of partial overlap*

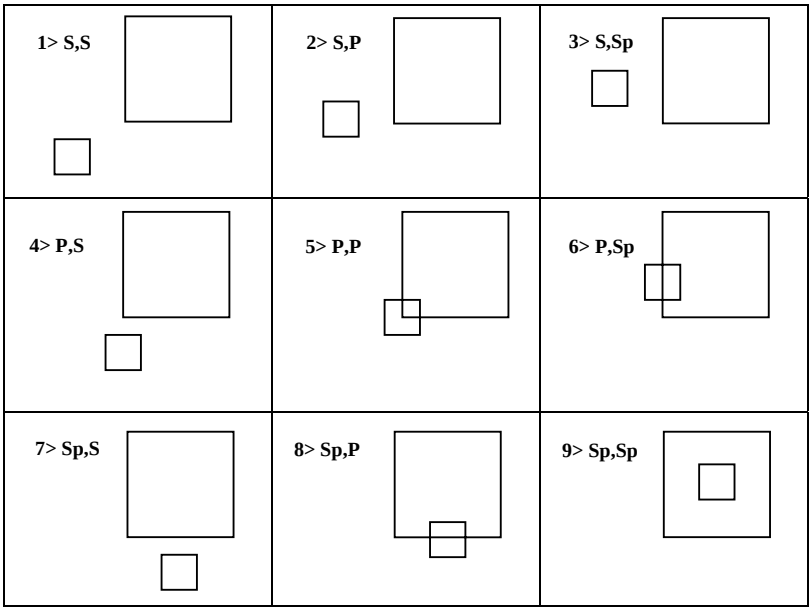

Fig. 9: The different types of overlap between two rectangles.

What is the degree of partial overlap? We explain a few concepts before attempting to define degree of partial overlap. There are a total of 9 possible ways in which overlap between two rectangles can occur. The types of overlap that occur along the X-axis are:

(i) Separate(S): No overlap.
(ii) Partial(P): A partial overlap of the intervals.

(iii) Superset/Subset(Sp): One interval is completely enclosed in another.

The same 3 cases occur along the Y-axis, thus making a total of 3 x 3 = 9 cases. The combinations are (S,S), (S,P), (S,Sp), (P,S), (P,P), (P,Sp), (Sp,S), (Sp,P), (Sp,Sp), as shown in Fig. 9. Cases 5 (P,P), 6 (S,P), and 8 (Sp, P) show the types of partial overlap that can occur. The definitions below elaborate the concept of *degree of partial overlap.*

Definition 1: Any two rectangles that overlap with each other are *neighbors* of each other.

Definition 2: Neighbors that overlap each other only partially are *partial neighbors*.
Case 9 in Fig. 9 is an example of two rectangles that are neighbors of each other but *not* partial neighbors, whereas cases 5, 6, and 8 show examples of partial neighbors.

Definition 3: The *degree of partial overlap* of a rectangle is the largest number of partial neighbors it has, such that all these neighbors belong to the same agent.

Thus by verifying that the degree of partial overlap is not more than one, we ensure that not more than one rectangle of each agent overlaps partially with the rectangle under consideration. If this condition is fulfilled, then the procedure described below is guaranteed to allocate a rectangle to each agent, such that the rectangle was marked by the agent itself.

4.5. *Procedure given condition B*

(i) The rectangles are submitted to a mediator that collects them in a list.

(ii) The X-coordinates of the intervals are read into the Xlist. The procedure sorts the X-coordinates of the intervals in the order of their occurrence from left to right. The Xlist stores them in this sorted order. Similarly the procedure sorts the Y-coordinates of the intervals in the order of their occurrence from bottom to top. The sorted Y-coordinates are stored in the Ylist. Note that we will determine which rectangle needs to be allotted to an agent based on the orderings of the intervals. The actual values of the X-coordinate (or Y-coordinate) of the intervals is not used for computation. Since there is no cardinal comparisons between agent rectangles (i.e., "areas" of rectangles are not computed), we obviate the need for the resource to be measurable.

(iii) We determine the relation of the X intervals to each other. The following possibilities exist between any two intervals:
   (a) The intervals do not overlap (S).
   (b) The intervals partially overlap (P).
   (c) One interval is completely contained in another (Sp).

   Each rectangle has a set of X relations (a relation being one of S, P, Sp or Sb) with its neighbors after we parse through the Xlist. The procedure processes the Ylist in the same way creating a set of Y relations for each rectangle and its neighbors.

(iv) We create a scoring matrix. Refer to Fig. 10. Note that the matrix has 16 values, although we discussed nine scenarios earlier. This is because topology-wise, if interval A is the superset (Sp) of interval B, then it is the same as saying interval B is the subset (Sb) of interval A. However, the scoring matrix belongs to a particular rectangle and is from the "point of view" of a particular rectangle. Thus a given rectangle being a subset of a larger rectangle is a distinct case compared to the rectangle being a superset of a smaller rectangle. Note that the sufficient condition for any two rectangles not to overlap is that their respective intervals do not overlap in at least one of the axes. Since such rectangles are not neighbors of each other, they are not assigned a score.

	S	**P**	**Sp**	**Sb**
S	--	--	--	--
P	--	**0**	**0**	**0**
Sp	--	**0**	**-1**	**0**
Sb	--	**0**	**0**	**1**

Fig. 10: The scoring matrix for a rectangle. The types of interval overlaps are Separate(S), Partial(P), Superset(Sp), Subset(Sb).

(v) Based on the types of overlap each rectangle has with its neighbors, we can create a directed graph that represents these connections. The directed graph is created based on rules described below. For any two rectangles:
   (a) Each rectangle is represented as a node.
   (b) If there is no overlap between the two rectangles, there is no edge between the corresponding nodes.
   (c) If there is a partial overlap between two rectangles, then each node will have a directed edge connecting the other node with an edge weight of 0.
   (d) If rectangle A is a subset of rectangle B, the edge going from Node B to Node A will have a weight of 1, whereas the edge going from Node A to Node B will have a weight of -1.

(vi) Once we have the topology of the connections in graph form, the procedure will arbitrarily start at some rectangle(or Node) and allot the particular rectangle based on the following conditions:
   (a) If there is no outgoing edge with edge weight of 1, then allocate this particular Node.
   (b) If there is such an edge, then travel along this edge and move to the Node connected to it. Now let this be the new Node under consideration. Repeat the procedure using this node as the starting point.

Once a particular Node (or the rectangle corresponding to it) has been allocated, then we remove the following Nodes from the graph, for future consideration:

(c) The node just allocated.
(d) All the neighbors of the current node
(e) All the nodes belonging to the agent that was the owner of the current node

The procedure is more clearly spelled out by the following pseudocode:

```
 //create a new empty graph of nodes representing the
 rectangles
 graph = createNewGraph()
 // populate the Xlist and Ylist
while (agentList.hasMoreAgents())
 currentAgent=agent.getNextAgent()
 while (currentAgent.hasMoreRectangles())
 //process each rectangle
   rectangle=currentAgent.getNextRectangle()
   node = createNewNode()
   node.setRectangle(rectangle)
   graph.addNode(node)
   rectangleList.add(rectangle)
   xInterval=rectangle.getXInterval()
   xIntervalList.add(xInterval)
   addCoords(xList, xInterval)
 // Do the same for y intervals here
   …
   …
   …
 end while hasMoreRectangles
end while hasMoreAgents

//sorting can be done by using a suitable sorting algorithm

sort(xList)
sort(yList)

//Create relationship matrix for X and Y coordinates

xRelations=formRelations(xIntervalList)
yRelations=formRelations(yIntervalList)
```

```
// implement scoring matrix rules here [Fig.10]
rectangleArray=rectangleList.listToArray()
arraySize = rectangleArray.getSize()
for i = 1 to arraySize do
 for j = 1 to arraySize do
 if xRelations[i][j]=='P' and yRelations[i][j]=='P' then
 // Create connections between nodes in the graph
 node1 = graph.getNodeForRectangle(i)
 node2 = graph.getNodeForRectangle(j)
 edge1 = node1.getEdgeWith(node2)
 edge1.setEdgeWeight(0)
 edge2 = node2.getEdgeWith(node1)
 edge2.setEdgeWeight(0)
 // Set up the remaining scoring matrix rules here
 …
 …
 …


// allocate the rectangles using hill-climbing algorithm
while (graph.hasMoreNodes())
 currentNode=graph.getNewNode()
 while (currentNode.hasMoreEdges())
   currentEdge=currentNode.getNewEdge()
       if (currentEdge.getEdgeWeight()==1)
          currentNode=currentEdge.getOtherNode(currentNode)
       end if
 end while hasMoreEdges
 currentNode.setAllocated(true)
 graph.remove(currentNode)
 graph.remove(currentNode.getNeighbors())
 graph.remove(currentNode.getOwner().getOwnedRectangles())
end while hasMoreNodes


//Various procedures used in pseudo-code

addCoords(coordsList, Interval)
 start=Interval.getIntervalStart()
 end=Interval.getIntervalEnd()
 coordslist.add(start)
 coordslist.add(end)
```

```
STOP

formRelations(intervalList)
intervalArray=intervalList.listToArray()
arraySize=IntervalArray.getSize()
for i = 1 to arraySize do
 interval1=intervalArray[i]
 start1=interval1.getIntervalStart()
 end1=interval1.getIntervalEnd()

 for j = 1 to arraySize do
   interval2=intervalArray[j]
   start2=interval2.getIntervalStart()
   end2=interval2.getIntervalEnd()

   if interval1.getOwner() == interval2.getOwner()
       relations[i][j]='--'
   else
       if start1<start2 then
          if end1<start2 then
             relations[i][j]='S'
          else if end1<end2
             relations[i][j]='P'
             else
             relations[i][j]='Sp'

    //test for other start, end conditions as shown in Fig.9
       …
       …
       …

 end for j
end for i
RETURN with relations
```

Note that based on the rules of graph creation mentioned above, it is impossible that two nodes belonging to the same agent will ever share a common edge. This is because edges are created only between nodes whose rectangles overlap and our protocol ensures that no two rectangles belonging to the same agent overlap.

Another point is that the mediator who runs the procedure will be required to keep a list of all the Nodes traversed during procedure execution. This list will be public information accessible to all participating agents. In case an agent (which was not the

mediator) needs to verify that the allocation was fair, all that it needs to do is to make sure that the path traversed by the mediator during process execution was a valid one by creating the graph of agent rectangles and executing the procedure. Note that the allocation may change if the mediator chose a different Node as a starting point. However, as long as the list of Nodes traveled is made public, the procedure is verifiable. This procedure will allocate to each agent a rectangle that was marked by the agent itself and, hence, is a fair division procedure. We clarify the allocation procedure mentioned above with the following example.

4.6. *An example allocation problem*

We take again the running example of three agents, *i, j,* and *k,* to whom a resource has to be allocated. The agents are expected to follow the protocol as laid out above. Each agent therefore marks out three rectangles on the resource, each of which it considers to be of

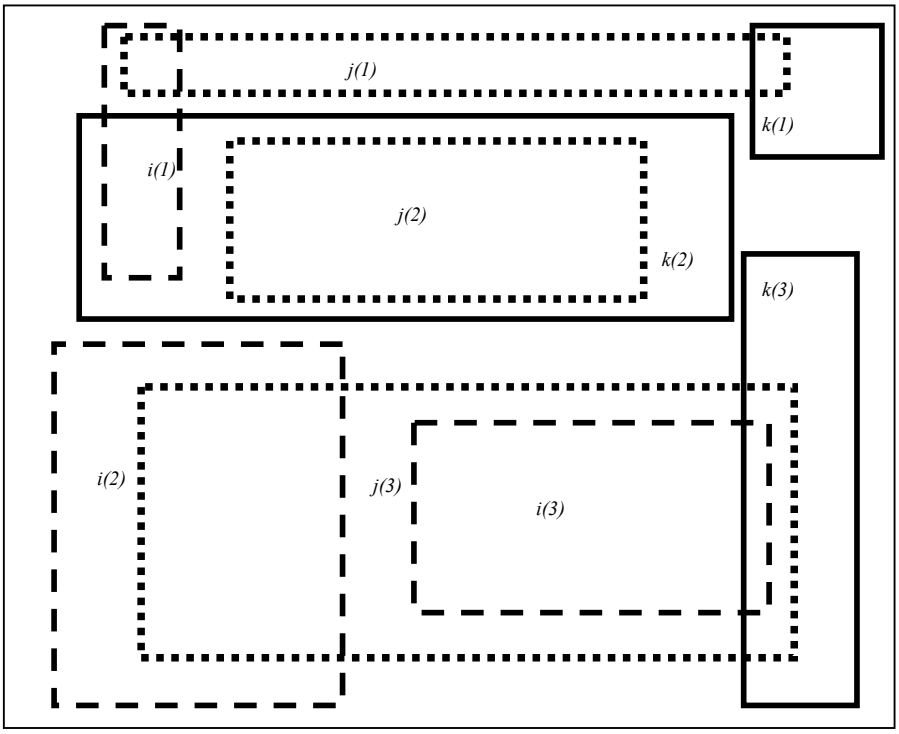

Fig. 11: Agents i, j, and k mark portions that fulfill condition B. The labels of the rectangles denote their respective owners.

value one-third of the total value of the resource. Based on how the rectangles are marked, we compute that the degree of partial overlap is at most one. Hence, condsition

B is fulfilled. Therefore the procedure presented above should be able to find a suitable allocation. Fig. 11 shows how the agents might have marked out their rectangles.

As described in the procedure, we first create a sorted Xlist and Ylist (step 2) based on the X and Y coordinates of the rectangles. Then we determine how each interval overlaps with the others (step 3). Based on the scoring matrix (step 4), we create a graph (step 5) representing the overlaps of the rectangles with each other. Refer to Fig. 12 for the corresponding graph. Next, we choose any node arbitrarily (step 6), say *i(1).* Observe that there are no outgoing edges from *i(1)* that have an edge weight of 1. So we can allot *i(1)* to agent *i.* Remove the following nodes from the graph:

- *i(1):* This has been allocated and hence should be removed from the graph.
- *j(1)* and *k(2):* These are neighbors of *i(1)* and in the process of allocating node (rectangle) *i(1),* nodes (rectangles) *j(1)* and *k(2)* are destroyed, hence they should be removed from the graph

- *i(2)* and *i(3):* Agent *i* has already been allocated *i(1)* and hence all of agent *i*'s nodes (rectangles) should be removed from the graph.

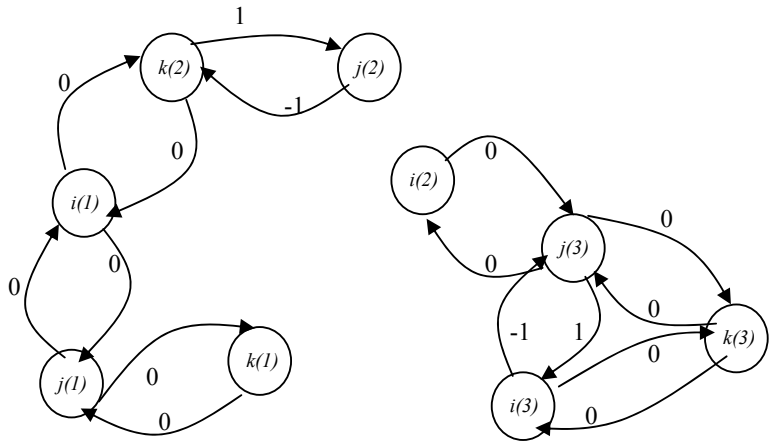

Fig. 12: The directed graph based on the topology of overlapping rectangles.

We still have the following nodes left to be allocated: *k(1), j(2), j(3)* and *k(3).* We select one of the nodes arbitrarily, say *j(3)* and apply the allocation procedure to this node (rectangle). Note that there is no outgoing edge in *j(3)* with edge weight 1 (*i(3)* has already been eliminated) . So we allocate this node to *j*. The following nodes are removed from the graph in this process: *j(3), k(3)* and *j(2).* Now only one node left is *k(1),* which is then allocated to agent *k.* This completes the allocation procedure. If we had chosen a different node as the starting point, it is quite possible that a different set of rectangles

might have been allocated to the agents. However, no matter which node we select a feasible allocation can be attained. The mediator will keep track of the nodes visited and display the information publicly in case any agents want to verify if the allocation was fair. Is there a guarantee that there exists a feasible allocation no matter how the agents lay out their rectangles? Yes, if condition B is true. In Sec. 3.7 we provide proof that the procedure is guaranteed to come up with a fair allocation provided condition B is fulfilled.

4.7. *Proof given condition B*

Theorem 4.7.1: If there are *n* agents, and each agent makes *n* rectangles, creating *n* portions of a rectangular cake, and if the rectangles are so marked that the degree of partial overlap is not greater than 1, then our protocol guarantees that each agent will be allotted a piece, such that the piece was one of the *n* portions created by the agent itself.

Proof.

Base Case *(n=2).* Each agent makes two rectangles. Arbitrarily start with one rectangle. Say this rectangle belongs to agent *i*. Due to condition A1, it is clear that none of the neighbors will belong to agent *i*. The following cases arise with respect to this rectangle, which we label as *i(1)*:

1. *i(1)* has no neighbors. In this case we allocate *i(1)* and remove all other rectangles made by *i*. None of agent *j's* rectangles are destroyed in this process. The procedure is guaranteed to allocate a rectangle to *j*, as no other agents are left.
2. *i(1)* has neighbors. Note that *i(1)* cannot have more than one partial neighbor, because at most one of *j's* rectangles is allowed to intersect partially with *i(1)* (by condition B). The following cases arise:
   - 2.1. *i(1)* contains at least one of *j*'s rectangles completely. In this case, *j*'s rectangle is a subset and the procedure allocates this rectangle to *j*. This will destroy exactly one of *i*'s rectangles. Remove all rectangles made by *j*. This leaves one of *i*'s rectangles intact, which can now be allocated to *i*.
   - 2.2. *i(1)* does not contain any of *j*'s rectangles completely. In this case there is exactly one partial neighbor of *i(1)*. The procedure allocates *i(1)*. In this process, one of *j*'s rectangles is destroyed. Remove all rectangles made by *i*. One of *j*'s rectangles remains, which is now allocated to *j*.
   - 2.3. *i(1)* is contained in one of *j*'s rectangles. This is the same as case 2.1, (by swapping the labels on the rectangles) where it has already been shown that both agents are guaranteed an allocation of one of their rectangles.

This proves that *i* and *j* can be allocated their fair shares no matter how the rectangles are arranged.

For any n (n>2). Let us assume that the allocation procedure works for up to *k* agents. We next show that the procedure works for *k+1* agents. Arbitrarily choose a rectangle. Say this rectangle belongs to agent *i*. The following cases arise with respect to this rectangle, which we label as *i(j)* (read as the j^{th} rectangle of the i^{th} agent):

1. *i(j)* has no neighbors. In this case we allocate *i(j)*. Remove all other rectangles made by agent *i*. No other agents' rectangles are destroyed in this process. Thus we have *k* agents, each with *k+1* rectangles. We arbitrarily remove one rectangle of each of the remaining agents. Thus we have *k* agents, each with *k* rectangles that we know can be allocated. Hence proved.
2. *i(j)* has neighbors. Note that *i(j)* cannot have more than one partial neighbor for each of the other agents (by condition B). The following cases arise:
   2.1. *i(j)* completely contains one (or more) of the rectangles belonging to one (or more) of the other agents. The procedure will make the interior rectangle the currently considered rectangle and reapply all steps beginning with case 1. In any case, one of the rectangles belonging to, say, agent *h* will be allotted. Allocating this rectangle to *h* will destroy at most one of other agents' rectangles who may be its partial neighbors (by condition B). All agents whose rectangles formed the superset of this rectangle will lose exactly one rectangle. Next the procedure will remove the other rectangles belonging to *h*. Thus each of the other agents (including *i*) would have lost at most one rectangle. Other agents will have *k+1* rectangles left. Arbitrarily remove one of the rectangles for each of the agents which have *k+1* rectangles. Thus we have k agents, each with k rectangles that we know can be allocated. Hence this is proved.
   2.2. *i(j)* does not contain any other agents' rectangles completely. Thus *i(j)* can have at most one partial neighbor of each of the other agents. The procedure allocates *i(j)*. In this process, at most one rectangle of each of the other agents is destroyed. Remove all other rectangles of *i*. Arbitrarily remove one rectangle of each of the agents that has *k+1* rectangles. Thus we have *k* agents, each with *k* rectangles that we know can be allocated. Hence this is proved.
   2.3. *i(j)* is contained in some other agents' (say one of them is agent *h*) rectangles. This is the same as case 2.1 (by swapping labels on the rectangles) where it has already been shown that all agents are guaranteed allocation of one of their rectangles.

This proves that the allocation procedure works for *n* agents for any *n≥2*, if the degree of partial overlap is not more than one. □

In the next section we discuss the various assumptions we have made in proposing this procedure. We also discuss what parts of the solution space are covered by this procedure and the properties of such solutions.

4.8. *Features of procedure for condition B*

The procedure proposed in this paper is novel in its approach and requires further clarification about the assumptions made and a discussion of the various aspects of the algorithm. Let us begin with a discussion of the various issues involved with the proposed allocation procedure when condition B is true (Section 3.3).

The motivation for the procedure is the hill-climbing algorithm. The procedure tries to find the "highest point" among various nodes. If another node is on the "same level," then moving to it does not improve our situation; hence we do not move to nodes on the

same level. We also do not move to nodes on a "lower level." If a node is on a "higher level" than the node we are currently occupying, then we have found the highest point relative to the node we started with and we allot this node (or the rectangle corresponding to this node).

The above procedure places less restriction on the way agents mark rectangles. However, the mediator will have to verify that the degree of partial overlap is not more than one, so it can guarantee a solution. Will the procedure work if the degree of overlap is greater than one? It is certainly possible that the procedure may find a solution, but the theorem cannot guarantee a solution anymore and it becomes a matter of trial and error to find the solution. If the degree of overlap is more than one, the procedure may fail for the following reasons:

(i) A solution exists, but the procedure did not find them. A solution might have been found had we chosen a different starting node. In such a situation, we can try each node as the first node and run the procedure to find out if an allocation exists. However, the running times may increase by an order of magnitude compared to the normal case.

(ii) A solution does not exist. This is due to the high degree of overlap among the rectangles. In this case, the procedure will not find the solution even if we run it repeatedly, with a different node as the starting node each time.

The analysis shows that the degree of overlap property is a *sufficient* condition for the existence of a solution, but not a *necessary* one. In the next section, we discuss various features of the conditions A and B and the quality of the allocation results that they generate.

5. Discussion

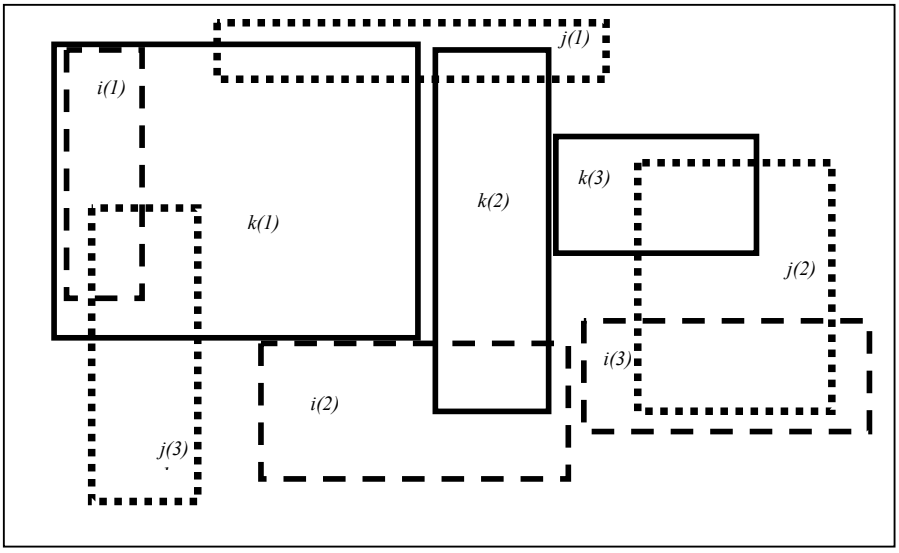

Fig. 13. Conditions A and B are true

What does the solution space look like with respect to condition A and condition B? In general, conditions A and B can be true or false independent of each other. The examples below show all the possible scenarios that can occur and give us a picture of how the sets are related to each other. We continue with the running example of this paper and denote the agents as *i, j,* and *k.*

(i) Conditions A and B are true (Fig. 13)

- A: The X coordinates of the rectangles of each agent do not overlap each other.
- B: The degree of partial overlap of each rectangle is at most 1

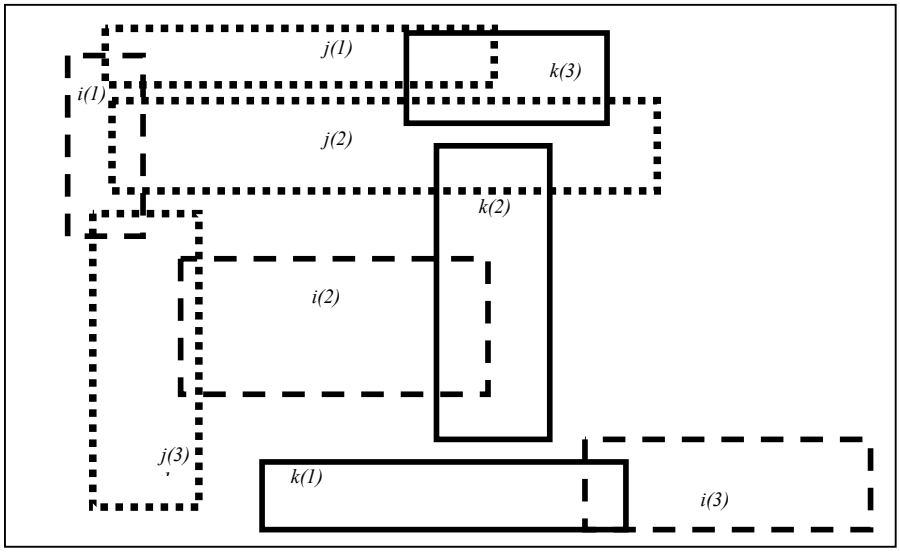

Fig. 14: Condition A is true and condition B is false.

(ii) Condition A is true, but condition B is false (Fig. 14)

- A: The Y coordinates of the rectangles of each agent do not overlap each other.
- B: The degree of partial overlap of *i(1)* is 3 (more than 1)

(iii) Condition A is false, but condition B is true (Fig. 15)

- A: For agent *j, j(1)* and *j(2)* overlap on the X-axis, but *j(2)* and *j(3)* overlap on the Y-axis
- B: The degree of partial overlap of each rectangle is at most 1

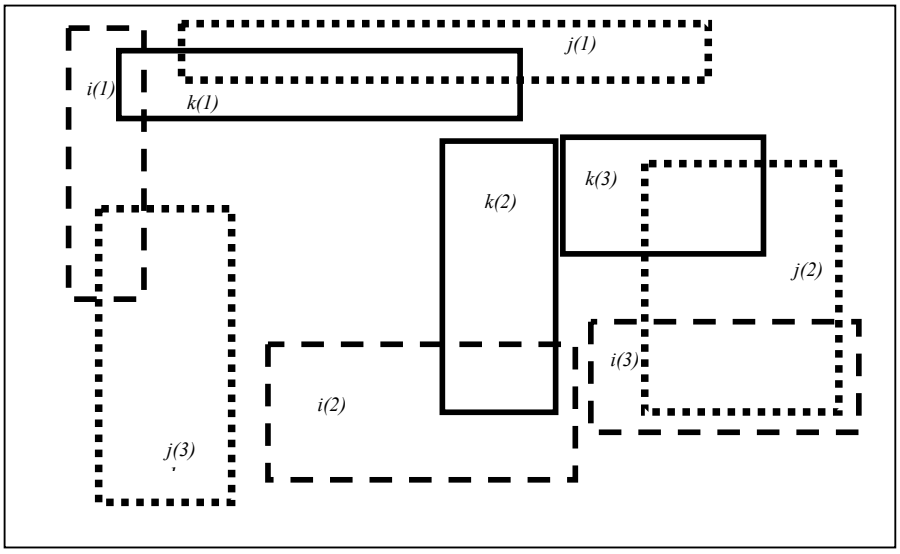

Fig. 15: Condition A is false and condition B is true.

From the examples shown, we can conclude that the solution spaces for conditions:

- A and B are not subsets of each other.
- A and B are not mutually exclusive.

Thus while condition B can be taken as more flexible, it is not necessarily a weaker condition. We can look upon condition B as a way to expand the space of solutions available to us. The solution space looks as shown in Fig. 16.

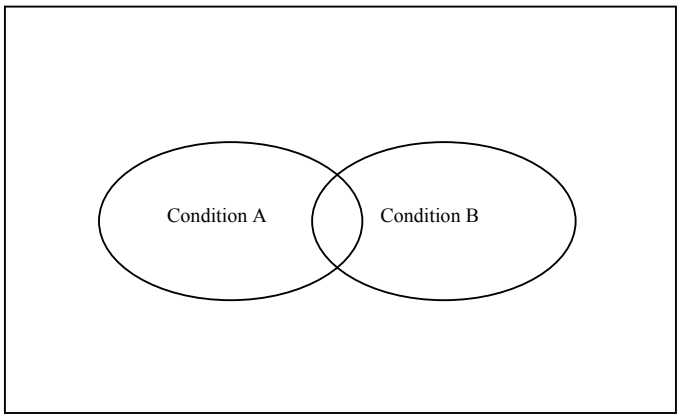

Fig. 16. The solution space for conditions A and B

Either of these conditions tests for the presence of a solution, but their absence cannot be taken as a guarantee that a solution does not exist. Is it possible for any algorithm to guarantee an allocation if both conditions A and B are false, even though the agents follow the protocol? We show that there exists no algorithm that can guarantee a solution, if just the protocol is followed. Specifically we focus on step 3 of the protocol, which requires that the agents' own portions do not overlap.

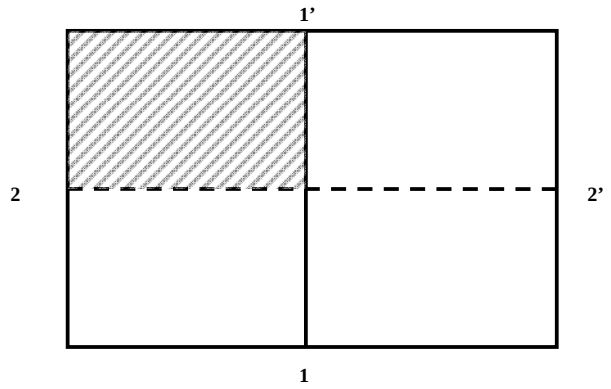

Fig. 17: Agent 1 divides the land into two vertical strips, while agent 2 divides them into two horizontal strips.

Theorem 5.1: If all agents follow only the protocol, then there exists no algorithm that can guarantee an allotment of rectangles, such that each agent will get a rectangle marked by itself.

Proof: We prove the above statement by showing a contradictory example. Consider two agents who want a plot of land divided between them. The agents create rectangles as shown in Fig. 17. In such a case, no matter what interval is allocated to agent 1, it will conflict with an interval of agent 2 and vice versa.

□

What the theorem tries to formalize is the idea that in addition to the protocol, some condition is required that will guarantee a feasible allocation exists. We have put forth two such conditions in this paper. It is quite possible that one can come up with other conditions that may further extend the solution space. The ideal case would be if one could find the perfect condition X that, if fulfilled, will include all the feasible allocations, i.e., it would be the largest superset of conditions A and B that covers the entire solution space. This is quite unlike the one-dimensional case, where just following the protocol guaranteed that the solution could be found. We discover that if the resource is two-dimensional it is impossible to guarantee a solution for every possible permutation

of agent rectangles. Ensuring that the agents follow the protocol alone is not a sufficient condition for a solution to exist. As future work, we are looking into the possibility of discovering conditions that may be able to exhaustively cover the solution space.

In the two-agent case shown above, there exists a procedure that tends to fairness, if we allow multiple iterations of the procedure to run. In the example shown, suppose the allocation procedure allocates the left column to agent 1 and the top row to agent 2 based on their markings, then we get the shaded region as the disputed area. Now the agents will again be required to repeat the procedure of markings, but this time only with the smaller disputed area. After a finite number of iterations the procedure stops, once agents can feel satisfied that they have a piece at least as large as the other agent's piece within the limit of tolerance. If the resource being divided is not recombinable (like land), successive applications of the procedure may create allotted portions that are not contiguous with previously allotted portions. Thus there may be isolated portions of land belonging to agent 1 surrounded by land belonging to agent 2. Hence the iterative solution is applicable only in qualified situations.

Next, let us look at how the procedure stands up to the various criteria mentioned in the literature survey.

- **Fairness:** The proposed protocol is fair. As mentioned earlier in the paper, an agent will find the allocation *fair,* if it feels it got exactly *1/n* of the value of the entire resource. Since as per the protocol the agent divides the resource into *n* rectangles of equal value and the procedure (given condition A and/or B is true) will allot one such rectangle to that agent, the agent gets $1/n^{th}$ (by its own valuation) of the entire resource. Thus our procedure allocates resources in a fair manner.
- **Envy-freeness:** Envy-freeness means that every agent thinks that the portion allocated to it is greater or equal in value than the portions other agents received. Thus no agent envies another agent's portion. It is strongly desirable for procedures to be envy-free, but it is a tougher criterion to fulfill than just fairness. There exist very few procedures that can guarantee envy-free division. Our procedure, while fair, is not envy-free. It is quite possible that though an agent may consider the value of the portion it received to be *1/n* of the total (and hence fair), it may be envious of another agent whose portion it thinks is of greater value than its own.
- **Efficiency:** Efficiency is also a criterion for judging procedures. It provides a measure of how much of the resource is wasted in the process of allocation. We use the term "efficient" in the Pareto optimal sense, i.e., is it possible that there exists an alternative allocation in which at least one agent is better off while the others are no worse than the current allocation? The answer in the case of our procedure is yes, i.e., our procedure is inefficient at allocating resources. This is because all the portions allocated to agents do not exhaustively use up the resource. The "leftovers" can be redistributed among the agents by running the protocol again and letting interested agents submit their preferences. This can be done repeatedly until finally every agent thinks the remnants are negligible in value. Such an iterative version of the protocol will certainly help improve the efficiency of resource distribution. However, the effectiveness of such an approach depends on the type of resource being distributed. If the resource is *infinitely divisible* and *recombinable* (like a cake)

then such an iterative procedure is appealing. But some resources, like scheduling time on a supercomputer to users or land division might only be infinitely divisible (though not recombinable) and in such cases the repeated divisions of the leftovers will make the agents portions so small as to be basically worthless. Our protocol and procedure is, however, more efficient than the traditional one-dimensional cake-cutting, especially when agents value some portions of the resource negatively. This is because we allow the agents to cut in two dimensions, thus completely eliminating negative utility regions from being included in the allocation. While there is a clear need to improve the efficiency of our single-shot protocol and procedure, it needs to be matched against the concurrent increase in the complexity of the protocol as well as the procedure.

- **Complexity:** What is the space and time complexity of this procedure? Let us calculate the space complexity first. Each rectangle is fixed by four points, with each point having two values (one each for the X and Y coordinates). Thus each rectangle is described by eight values. Each agent marks n rectangles, creating a total of n^2 rectangles (by n agents). Thus the space complexity of the procedure is $8n^2$**.** Now we compute the time complexity. First the list of X and Y coordinates needs to be sorted. An efficient sorting algorithm, such as insertion sort, will take $nlog\ n$ time, which translates to $2n^2log\ n$ time (replacing n by n^2) for each axis. Next X and Y relations need to be created. Since condition B is true, each rectangle can have at most $n-1$ partial neighbors. If one rectangle completely subsumes the other (i.e., they are neighbors but not partial neighbors), then the inner rectangle can have at most $n-2$ partial neighbors. Thus the rectangles will have an average of $n-1$ neighbors in the worst case, and a total of $n(n-1)= n^2-n$ relations are possible on each axis. Analogously, the graph created out of these relations will have vertices equal to n^2 and the number of edges equal to n^2-n (in the worst case). The running time of the procedure to traverse these nodes is therefore $O(n^2)$.
- **Strategy-proof:** A protocol is said to be *strategy-proof,* if each agent declaring its true evaluation is a dominant strategy. A typical example of a strategy-proof mechanism is the Vickery auction, where agents submit sealed bids to the auctioneer and the agent quoting the highest price wins, but is only required to pay the amount of the second highest bid. Our protocol is strategy-proof. The dominant strategy for any agent while drawing rectangles is to make sure that it values each of the rectangles it draws as exactly $1/n$. If an agent tries to draw a rectangle that is smaller in value than any other agent's rectangle in order to guarantee itself a specific piece of the resource, it may end up getting the resource, but the value of that resource will be smaller than $1/n$, thus the agent ends up getting less than its fair share. On the other hand, if an agent tries to get more than $1/n$ by drawing a rectangle as large as possible, then it is quite likely that it will subset some other agents' rectangle, which will end up getting allocated to another agent. For clarification, consider the following extreme example: Agent i is greedy and wants as much of the resource as possible. So it draws one large rectangle that covers the entire resource, while other agents follow the dominant strategy and draw n rectangles of equal value. What will agent i receive? Agent i receives nothing, because the procedure that allots rectangles takes as input one large rectangle (which agent i had drawn) and $n-1$ rectangles of zero value and the procedure ends up allocating one as these zero-valued rectangles

as agent *i*'s allocation. Thus agent *i* gets zero utility by being greedy. Our protocol is therefore not open to manipulation by greedy agents and any attempts to misreport preferences will backfire. There is no incentive for the agents to lie and hence the protocol is strategy-proof.

- **Measurability:** This is an important notion for mathematicians and economists. The basic assumption is that any resource being allocated must be "measurable", i.e., there must exist a function that can assign a number (like "length," "area," or "volume") to subsets of a given set (like land, for example). This notion paves the way for a cardinal comparison of various subsets for enabling an allocation. However our procedure does not need cardinal comparisons to make an allocation. We allocate portions based on the topology of overlaps. If the one rectangle is completely contained in another then the smaller rectangle is allocated. If two rectangles only partially overlap each other, then the one that was encountered first (by the procedure) is allocated first. We do away with the notion of comparison of agent rectangles based on their "areas." While agents may need the set to be measurable in order to "measure" out equal portions of the resource, such a restriction is not imposed by the protocol itself. Agents may use alternate means to create equal portions, like creating *n* rectangles arbitrarily and adjusting their sizes iteratively until it doesn't prefer any one over the other. This is a way to *ordinally* create *n* rectangles of equal value without using the notion of a "measure." Unlike earlier work in land division, we do not explicitly require that the resource to be allocated be measurable.
- **Constructive (Non-existential):** The procedure proposed here is algorithmic. Most of the early literature dealing with two-dimensional resources (primarily in the form of land division) is dominated by economists and mathematicians. Their results are primarily existential in nature viz. they analyze various features of the problem (like types of utility functions, fairness and efficiency of the possible allocations, etc.) and show whether or not allocations "exist" with given properties. They do not however propose ways to find such solutions. Thus it is still unknown that even if feasible solutions exist whether the problem of finding an allocation procedure is tractable. Because of a lack of constructive solutions to allocation problems, it is not possible to realize them in the real world where agents negotiate with one another and a mediator is able to compute a feasible allocation in a reasonable amount of time. For multiagent system designers, in addition to the knowledge that feasible allocations exist, the following points need to be considered:
  (a) Is there a procedure to find these allocations? (Yes)
  (b) What is the communication cost of the procedure? ($O(n^2)$)
  (c) What is the computational complexity of the procedure? ($O(n^2)$)
  (d) Can the agent preferences be kept private? (Yes)
  (e) Can the procedure be manipulated by agents' lies? (No)
  (f) What is the efficiency of the allocation procedure? (Inefficient)
  (g) Are the procedure and the protocol iterative? (No)
  (h) Are the agents restricted in using certain classes of utility functions? (No)

  We answer some of the questions in this paper. Ours is the first result that can be expressed as a computer algorithm. Not only does it test for the existence of a solution, it is able to find one if it exists (given condition A and/or B is true).

- **Nature of agent utility functions:** Our literature survey shows a number of restrictions placed on agent utility functions, none of which is applicable in our case. The agents may draw rectangles based on their internal utility functions or completely do away with the use of utility functions. As stated in the discussion earlier about measurability, agents can also draw rectangles using ordinal rather than cardinal comparisons. Briefly, we mention the typical restrictions on agent utility functions historically mentioned in the literature, which are *not* applicable to our case:
  (a) Non-atomic: This requirement specifies that the agent utility functions smoothly decrease to zero as the amount of the resource tends to zero, i.e., there should be no "atoms" in the resource, where portions smaller than the "atom" are of zero value to the agent.
  (b) Additive: This is a common requirement for utility functions and helps simplify much analysis. If A and B are two disjoint subsets, then simple additivity states that:

$$v(A \bigcup B) = v(A) + v(B) \quad \textit{for all disjoint A, B} \in \sum$$

  (c) Concave: Concave domains are a subset of subadditive domains and the utility functions have the property:

$$v(A \bigcup B) \leq v(A) + v(B) \textit{ for all disjoint A,B} \in \sum$$

  They are less commonly modeled than simple additivity, because it is difficult to prove results in this domain. However they more realistically reflect the utility of obtaining additional resources in the real world.
  (d) Continuous: Utility functions are assumed to be continuous. This means that for every amount x of the resource the agent has a unique value for that amount. This simplifies analysis, but might not be necessarily true in the real world. In addition, utility functions are said to be smooth if there are no "kinks" in the function value anywhere, i.e., the function is differentiable at all points of the curve.

6. Future Work

We have presented a constructive solution to the allocation of two-dimensional resources. The protocol and procedure we present is an important first step that clarifies questions about whether the solutions for agent negotiations can be implemented. There are, however, several ways in which this work can be extended to increase relevance for real world allocation problems. We mention the most pertinent issues that are useful for future work:

- **Increasing the space of feasible allocations:** The protocol in its current form is a single-shot protocol. Either the agents' rectangles are laid out in a manner where feasible allocations exist or no agent gets any part of the resource (a conflict deal is reached). There is a need for creating an iterative version of the protocol whereby agents can send proposals to one another by continuously changing the shape of their

rectangles and finding a solution that is feasible (and preferably optimal) to all the agents. It is quite likely that there will be a concomitant increase in the complexity of the protocol, communications, and the procedure. Thus while the space of solutions will certainly increase, mechanism designers will have to balance this against increased cost of computation both for the agents as well as the mediator.

- **Search for condition X:** As mentioned in Sec. 4, a more generic condition than the two we have proposed will serve to increase the space of feasible allocations. Conditions A and B serve as a benchmark against which future conditions can be compared. Conditions A and B are *sufficient* but not *necessary* to guarantee the existence of a solution. There can exist other conditions that behave similarly. The ideal condition X would be one that is both sufficient and necessary. In such a case condition X would exhaustively cover the solution space and would be the (largest) superset of both solutions covered by conditions A and B.
- ***n*-dimensional resources:** In this paper we consider the case where a resource is two-dimensional. We extended our earlier result of one-dimensional resource allocation to two dimensions. But there were significant qualitative differences between the problems. We presented a proof that if condition B is true then a feasible allocation exists and the procedure would be able to find the allocation. One can fashion proofs in an analogous manner for three (and higher) dimensional resources. But the challenge is to prove that the proof for condition B holds for arbitrary *n*-dimensional resources. Is it possible to create a (meta) proof to prove a feasible allocation exists — if condition B is true — for an *n*-dimensional resource with arbitrary *n*?
- **Efficiency:** The current protocol, while being more efficient than one dimensional resource allocation schemes (like moving knife), is inefficient because there will always be leftovers from the allocation. As mentioned earlier, efficiency can be improved by creating a simple iterative version of the single-shot protocol. But that has limitations (refer Sec. 4, discussion on efficiency). Any work done to improve the efficiency of allocation will greatly increase the appeal of this procedure but one is likely to increase the complexity of the protocol as well the procedure in the process.
- **Allowing more general shapes:** The protocol as it is presented currently allows agents to mark regions of interest in the shape of rectangles only. This no doubt simplifies the allocation procedure, but it is quite restrictive. If agents were allowed to mark out regions of interest as more general polygons then it will improve the efficiency of the system and increase the utility received by agents. It may also increase the space of feasible allocations. This is a challenging problem. If a procedure were found that is able to allocate general polygons, then it might be possible to extrapolate those results to the case where agents can draw amoeba shaped regions and the procedure can find a satisfactory solution.
- **Allowing other topologies:** We assume the resource to be a two-dimensional flat plane. Would it be possible to extrapolate the procedure to other shapes like the surfaces of cylinders, spheres or torii? Our procedures are not directly applicable to such shapes because of their different topology. But one can easily visualize resources having such topologies; for example: dividing underwater mineral resources may require the agents to take into account that the earth has the topology of a sphere rather than approximate it as a flat plane.

- **Creating a decentralized version of the procedure:** The procedure in its current form is centralized. Since the running time is $O(n^2)$,it is quite appealing to create a distributed form of this procedure, so that the running time of the procedure is reduced. While creating a distributed version of the procedure is a non-trivial task in itself, the following issues need to be considered as well:
  (a) Is a different set of agents needed to execute the distributed procedure? Or can the participating agents themselves execute the procedure?
  (b) How can the information about agent preferences be kept private?

7. Conclusion

This paper has presented a constructive solution to the classic land division problem. This is the first result that is in the form of an algorithm suitable for computer implementation, unlike earlier ones that are only existential in nature. It uses the notion of degree of partial overlap to create a sufficiency condition for the existence of a solution, and proposes a procedure to find one in such a case. The proposed solution is fair, strategy-proof, constructive, and does not explicitly need the resource to be measurable. We discovered that unlike the case for one-dimensional resource allocation, there may not exist a feasible allocation for every permutation of agent rectangles in the two dimensional case. This procedure should be taken as the first step in hyperdimensional resource allocation. While our allocation procedure is unique in the sense that it does not convert the two-dimensional resource allocation problem into a one-dimensional problem (in case of procedure given condition B) before executing, there is nevertheless considerable scope for tweaking many parts of the procedure. Creating a distributed version of the procedure will reduce the workload on any one agent (which assumed the role of the mediator) and reduce running times. It will be interesting to see the existence and properties of procedures for higher-dimensional resources and resources with topologies different from a plane.

References

1. Soh, L.-K.; Tsatsoulis, C., A Real-Time Negotiation Model and A Multi-Agent Sensor Network Implementation. *Autonomous Agents and Multi-Agent Systems* **Nov 2005,** 11, (3), 215-271.
2. Rosenschein, J. S.; Zlotkin, G., *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*. MIT Press: London, England, 1994.
3. Davis, R.; Smith, R. G., Negotiation Distributed as a Metaphor for Problem Solving. *Artificial Intelligence* **January 1983,** 20, (1), 63-109.
4. Brams, S. J.; Taylor, A. D., *Fair Division: From Cake-Cutting to Dispute Resolution*. Cambridge University Press: New York, 1996.
5. Iyer, K.; Huhns, M. N. *Negotiation Criteria for Multiagent Resource Allocation.*; Department of Computer Science and Engineering, University of South Carolina: Feb 2007.

6. Iyer, K.; Huhns, M. In *Multiagent Negotiation for Fair and Unbiased Resource Allocation*, OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2005, Oct 2005; Meersman, R.; Tari, Z., Eds. Springer: Oct 2005; pp 453-465.
7. Austin, A. K., Sharing a cake. *Mathematical Gazette* **Oct 1982,** 66, 212-215.
8. Lemaître, M.; Verfaillie, G.; Fargier, H.; Lang, J.; Bataille, N.; Lachiver, J.-M. In *Equitable Allocation of Earth Observing Satellites Resources* Proc. of 5th ONERA-DLR Aerospace Symposium (ODAS'03), Toulouse, France, 2003; Toulouse, France, 2003.
9. Kraus, S., *Strategic Negotiation in Multiagent Environments*. MIT Press: 2001.
10. Chevaleyre, Y.; Dunne, P. E.; Endriss, U.; Lang, J.; Lemaitre, M.; Maudet, N.; Padget, J.; Phelps, S.; Rodriguez-Aguilar, J. A.; Sousa, P., Issues in Multiagent Resource Allocation. *Informatica* **2006,** 30, (1), 3-31.
11. Lesser, V.; Ortiz, C. L.; Tambe, M., *Distributed Sensor Networks: A Multiagent Perspective*. 2 ed.; Springer: 2003.
12. Cramton, P. C.; Shoham, Y.; Steinberg, R., *Combinatorial Auctions*. MIT Press: 2006.
13. Rothkopf, M. H., Thirteen Reasons Why the Vickrey-Clarke-Groves Process Is Not Practical. *Operations research* **March–April 2007,** 55, (2), 191-197.
14. Steinhaus, H., The problem of fair division. *Econometrica* **1948,** 16, 101-104.
15. Sen, S.; Biswas, A. In *More than Envy-Free*, ICMAS'00, 2000; AAAI Press: 2000; p 0433.
16. Tasnádi, A., A new proportional procedure for the n-person cake-cutting problem. *Economics Bulletin* **2003,** 4, (33), 1-3.
17. Robertson, J.; Webb, W., *Cake Cutting Algorithms*. A K Peters Ltd: Nattick, MA, 1998.
18. Stewart, I., Your Half 's Bigger Than My Half!. *Scientific American* December 1998, pp 112-114.
19. Huhns, M. N.; Malhotra, A. K., Negotiating for Goods and Services. *IEEE Internet Computing* **July 1999,** 3, (4), 97-99.
20. Hill, T. P., Sharing a cake. *Mathematical Gazette* **Oct 1982,** 66, 212-215.
21. Dubins, L. E.; Spanier, E. H., How to cut a cake fairly. *American Mathematical Monthly* **Jan 1961,** 68, 1-17.
22. Beck, A., Constructing a Fair Border. *American Mathematical Monthly* **Feb 1987,** 94, 157-162.
23. Webb, W. A., A Combinatorial Algorithm to Establish a Fair Border. *European Journal of Combinatorics* **May 1990,** 11, 301-304.
24. Maccheroni, F.; Marinacci, M., How to cut a pizza fairly: Fair division with decreasing marginal evaluations. *Social Choice and Welfare* **2003,** 20, (3), 457-465.
25. Bartle, R. G., *The Elements of Integration and Lebesgue Measure*. Wiley-Interscience: New York, 1995.
26. Thomson, W. *Children crying at birthday parties. Why? Fairness and incentives for cake division problems*; 526; Oct 2005.
27. Chambers, C. P., Allocation rules for land division. *Journal of Economic Theory* **2005,** 121, 236-258.
28. Mill, J. S.; Sher, G., *Utilitarianism*. Hackett Publishing Company: 2002.