IJITKM 0973-4414 Volume-9, Number-1 Jun-Dec 2015 pp. 32-56 **Impact Factor 2.8** available online at www.csjournalss.com

Serial Dictatorship with Ties

Moon K Chetry, Dipti Deodhare, Sumer Sobti[1], and Kshipra Gurunandan
Center for Artificial Intelligence and Robotics, Bangalore

Abstract: Resource allocation in agent societies, which in current literature, is also called as Multi-Agent Resource Allocation (MARA) problem, is the problem of deciding how to distribute a number of resources amongst a number of agents. It is a growing area of research at the interface of economics and computer science. In this paper, we study the problem of designing a mechanism to allocate a fixed number of objects to agents in a strategy-proof and pareto-efficient way, when each agent has a quota of varying number and indifference of ranking among set of bundles are allowed. Each agents i needs $a_i (\geq 1)$ many objects from the set of objects R, while having preferences over the set of bundles of size a_i.. The motivation for us to study such problem comes from a problem called the site selection problem, which arises often in the military domain. In this paper, we model this site selection problem as a Multi Agent Resource Allocation problem. In our model, though we consider a restricted preference, which is justified by the motivating example that is the site selection problem, our algorithm can also accommodate unrestricted preference. That is preference over all possible bundles of size a_i.. Our main result (Theorem A) ensures that the algorithm developed satisfies both pareto-efficiency and strategy-proofness.

1 Introduction

Resource allocation is the problem of finding a suitable distribution of resources among several entities, called agents here [2]. Allocation of resources to agents is an important issue that has been considered from different perspectives in economics (especially social choice theory) and in computer science (especially artificial intelligence and operations research), and arises in various real-world settings: auctions, electronic spectrum and frequency allocation, airport traffic management, fair and efficient exploitation of earth observation satellites [4], to name a few. In this paper, we model the site selection problem, which we describe in the section 2, as a Multi-Agent Resource Allocation (MARA) problem. A site-selection problem relating to the deployment of military units, which are part of a military formations, within the deployable area selected by the swarm search algorithm. The deployable area, where military units are to be deployed is regarded as an n X n grid consisting of squares each of which have certain terrain attributes (e.g. proximity to road, water source, ease of telecommunications, etc.). Each unit has specific requirements or quotas of set of squares corresponding to the square attributes. Individual units (which could be strategic) report their preferences to higher authority, and the higher authority or the central agent aggregates their preferences and based on their priority allocates the squares.

We consider here the site selection problem as a Multi Agent Resource Allocation problem and study of designing a mechanism to allocate a fixed number of objects to agents in a strategy-proof and pareto-efficient way, when each agent has a quota of varying number and indifference of ranking among set of bundles are allowed. Each agents i needs $a_i (\geq 1)$ many objects from the set of objects R, while having preferences over the set of bundles of size a_i. In our model, we consider a restricted preference (preferences over not all the possible bundles of size a_i), where each agent i reports to the central agent a partition, of the set of object R into subsets of cardinality a_i, except may be the last; and he or she is indifferent in getting any a_i many objects from the same subset.

When only one resource is to be allocated to each agent, *i.e.* $a_i = 1$, and agents are never indifferent between resources, there is a strategy-proof and pareto-efficient mechanism, namely sequential or serial dictatorship (SD). Shapley and Scarf (1974) initiated the study of such problems, which are called matching or house allocation problems. These kinds of problems have two extremes. At one extreme, nobody in the problem is endowed with a resource and there is only a social endowment (Hylland and Zekhauser, 1979), and in the other extreme, everybody

[1] Sumer Sobti was in tenure at the Center for Artificial Intelligence and Robotics, Bangalore, during the formulation and the execution of the problem in the manuscript.

DOI: 10.141079/IJITKM.2015.907

is endowed with a resource. Svensson had shown that the only house allocation mechanism that is strategy-proof, non-bossy, and neutral is the serial dictatorship. In [1], Hatfield had studied the problem of allocating a fixed number of resources to agents, where every agent has a fixed quota (*i.e.* $a_i \geq 1$) that must be filled exactly. He has shown that when each agent needs $Q > 1$ resources, the only strategy-proof, Pareto-optimal, non bossy mechanism are sequential dictatorship. Further, adding the requirement of neutrality, he proved that the only strategy-proof, Pareto-optimal, non bossy and neutral mechanism is serial dictatorship. But his model is confined to cases where indifference in ranking is not allowed.

In [3], Paula Jaramillo and Manjunath (2011), for the first time, have studied this kind of allocation problem where agents do exhibit indifference, but for unit demand; *i.e.* each agent needs exactly one single resource but they may be indifferent among a set of resources. They have shown that without strict preference many results of house allocation problems will no longer hold - a core allocation is no longer guaranteed to exist (Shapley and Scarf, 1974) and the set of competitive allocations no longer coincides with the core (Wako 1991). Thus group strategy-proofness and pareto-efficiency are incompatible (Ehlers 2002). They have shown that there may not even be a competitive allocation that is pareto-efficient. They have further shown that when we drop the assumption that agents always strictly rank the resources, the problem in which either some or all of the resources are not initially owned are equivalent to problems where each resource is initially owned by someone. They have proved that in such a "house allocation with existing tenants" problem, which allows indifference, a generalized version of the top trading cycle mechanism is strategy-proof, pareto-efficient, and individual rational. Moreover, the allocation selected is in the weak core.

In this paper, we have studied the allocation problem where both indifference in ranking and requirement of more than one resources are allowed. This way, we have studied a more general model than [1] and [3].

1.1 Some Basics of Multi-Agent Resource Allocation (MARA)

Consider a set of agents $N = \{1, 2, \ldots, n\}$, indexed by $i \in N$, and a finite set of resources $R = \{a_1, a_2, a_3, \ldots\}$. Each agent i has a preference relation $\succcurlyeq_i$ over the family 2^R of all subsets of R. An allocation $A = (A_1, A_2, \ldots, A_n)$ is a list that assigns a set $A_i \subseteq R$ of goods to each agent. Often we refer to a subset of R as a bundle of resources. An allocation $A = (A_1, A_2, \ldots, A_n)$ is feasible if $A_i \cap A_j = \emptyset$ for all $i \neq j$ and $\bigcup_{i=1}^{n} A_i = R$.

For $A_i \in 2^R$, we denote by $rk_i(A_i)$ the rank of the bundle A_i in the ordering $\succcurlyeq_i$: thus, $rk_i(A_i) = 0$ means that the bundle A_i is ranked at the top in the preference of agent i, etc. Also $rk_i(A) = rk_i(A_i)$ for any allocation $A = (A_1, A_2, \ldots, A_n)$.

Suppose F is the set of all feasible allocations A. The set of Pareto-optimal (or Pareto-efficient) allocations is denoted by *par(F)*; *i.e.* $A \in par(F)$ if A is feasible, and there does not exist a feasible allocation A' that is unanimously (weakly) preferred to A by all agents, with a strict preference for at least one agent. $A = (A_1, A_2, \ldots, A_n)$ is Pareto-efficient if there does not exist an allocation $A' = (A'_1, A'_2, \ldots, A'_n)$ such that there exists an agent $j \in N$ such that $rk_j(A'_j) > rk_j(A_j)$ and $rk_i(A'_i) \geq rk_i(A_i)$ for all $i (\neq j) \in N$.

A mechanism Π is a procedure such that for a given preference profile $(\succcurlyeq_1, \succcurlyeq_2, \ldots, \succcurlyeq_n)$ (also written as $(\succcurlyeq_i, \succcurlyeq_{-i})$) for every agent $i \in N$, $\Pi[(\succcurlyeq_1, \succcurlyeq_2, \ldots, \succcurlyeq_n)]$ or $(\Pi[(\succcurlyeq_i, \succcurlyeq_{-i})])$ gives an allocation. Π is called a Pareto-efficient mechanism if for every preference profile $(\succcurlyeq_i, \succcurlyeq_{-i})$, $\Pi[(\succcurlyeq_i, \succcurlyeq_{-i})]$ is a Pareto-efficient allocation.

Π is called a strategy-proof mechanism if for every preference profile $(\succcurlyeq_i, \succcurlyeq_{-i})$, and any feasible allocation $A = \Pi[(\succcurlyeq_i, \succcurlyeq_{-i})]$, there does not exist any agent i such that whenever i changes his preference $\succcurlyeq_i$ to $\succcurlyeq'_i$, and the rest of the

agents' preferences remain unchanged, the mechanism gives the feasible solution $B = \Pi[(\succcurlyeq'_i, \succcurlyeq_{-i})] \neq A$ such that $B_i \succ_i A_i$, *i.e.* B_i is strictly preferred to A_i by the agent i under the preference relation $\succcurlyeq_i$.

The classic house allocation problem studies the allocation of houses, where there are n agents and n houses. Each agent needs exactly one house and has a strict preference over the houses. Serial Dictatorship mechanism initially assign an ordering of the agents (which is simply a permutation on the set of agents) and based on this ordering agents are allocated their best preferred house . It is easy to see that the Serial Dictatorship mechanism is pareto-efficient and strategy-proof. But if the requirement of each agent is more than one, and agent's are allowed to be indifferent among the objects, then the same may not be true; which is clear from the example below.

Example 1.2

Consider $N = \{1, 2\}$, $R = \{r_1, r_2, r_3, r_4\}$. The ranking for agents 1 and 2 are as follows –

Rank	Agent 1	Agent 2
0	(r_1, r_2), (r_1, r_3), (r_2, r_3)	(r_2, r_3)
1	(r_1, r_4), (r_2, r_4), (r_3, r_4)	(r_1, r_2), (r_1, r_3), (r_2, r_4), (r_3, r_4)
2		(r_1, r_4)

Now, if we break the ties arbitrarily and named the bundle (r_2, r_3) as the top ranked bundle of agent 1, then the serial dictatorship will give the allocation $\{(r_2, r_3), (r_1, r_4)\}$, which is Pareto-dominated by the allocations $\{(r_1, r_2), (r_3, r_4)\}$ and $\{(r_1, r_3), (r_2, r_4)\}$. Therefore the way of breaking ties is crucial in ensuring efficiency. The proposed algorithm precisely does the same.

2 Motivation: Site Selection Problem (Deployment of Military Units)

We have a site selection problem concerning the selection of suitable areas for the deployment of military units, which are part of a military formation, within the deployable area selected by the Swarm Search Algorithm.

Problem definition: The deployable area, where military units are to be deployed is regarded as an $n \times n$ grid consisting of squares, each of which has certain terrain attributes (*e.g.* proximity to road, water source, ease of telecommunications *etc*.).

We have 6 types of military units (MU) and a total of 10 units with one agent representing each unit.

1. Combat Unit A (1)
2. Combat Unit B (1)
3. Combat Unit C (3)
4. Headquarters (1)
5. Support Units (2)
6. Logistics Units (2)

Each unit requires a set of squares whose associated attributes satisfy a set of constraints. Hard constraints need to be exactly satisfied and soft constraints are preferences that need to be optimized as best as possible. This means that if units cannot be assigned squares that are exactly meeting the required attributes, they should be assigned squares with proximity to the desired squares.

Individual Requirements: Individual units, or agents representing the units, have specific preferences over terrain attributes of squares, listed in order of importance. If all the individual preferences cannot be satisfied, the lowest priority ones must be declined first. The agents are arranged in an ordering (a permutation on the set of agents) based on their priorities, and the agent at the top of the priority list has to be allocated his highest preferred set of squares, and so on.

Hence this problem demands the application of a serial dictatorship type of mechanism for the allocation of the squares. But as the requirement of each agent could be more than one square, and there could be ties in their preference list, *i.e.* some agent i may be indifferent between two set of squares of cardinality a_i, so the mechanism and the analysis carried out by Hatfield [1] or Jaramillo and Manjunath [3] are not applicable here. We need a serial dictatorship type of mechanism which considers both the tie in preference and greater than unit demand.

Allocation: Individual units report their preferences to a higher authority. The higher authority, or central agent, aggregates their preferences, and then, based on their priority, allocates the squares.

Desirable Welfare Properties: (Pareto) efficiency and strategy-proofness.

We have studied this problem in details as a case study in section 6. In the section 6, we have taken a toy example of this problem, where the deployable area is an 8 X 8 grid. We consider all the problem constrains and the agents preferences, and use the mechanism we proposed to get a final allocation which is pareto-efficient.

3 Approach: The Model

In section 1.1, we have stated some basics of a general Multi-Agent Resource Allocation (MARA) problem. By a general MARA problem we mean that the preference relation, for any agent i, is over the set of all possible bundles of resources – the preference relations are not restricted. In this section, we model the site selection problem where we restrict the preference relations of the agents. Nevertheless, all the definitions are applicable for both the restricted and the unrestricted case.

- $R = \{r_1, r_2, \ldots, r_m\}$ is the set of resources.
- $N = \{1, 2, \ldots, n\}$ is the set of agents.
- Every agent i needs a_i resources, where $m \geq \sum a_i$.
- Every agent partitions the set of resources R into disjoint subsets based on its preferences.

Agent i: $Y_{i1}, Y_{i2}, \ldots, Y_{ik}$, where $Y_{il} \subseteq R$, $|Y_{il}| \geq a_i$ for each l except $l = k$, and $\sum_{l=1}^{k} Y_{il} = R$, $Y_{ij} \cap Y_{il} = \emptyset$.

- Agent i is indifferent in getting any a_i resources from the same set Y_{il}, $l = 1, 2, \ldots, k$.
- Agents' rankings:

– Any bundle of a_i resources from the set Y_{i1} gives him rank 0. Cardinality of the set of bundles of rank 0 is ${}^{Yi1}C_{ai}$.

– Any bundle of a_i resources of which any $(a_i - 1)$ resources are from the set Y_{i1} and any 1 resource is from the set Y_{i2}, gives him rank 1. Cardinality of the set of bundles of rank 1 is ${}^{Yi1}C_{ai-1} \times {}^{Yi2}C_1$.

– Any bundle of a_i resources of which any $(a_i - 2)$ resources are from the set Y_{i1} and any 2 resources are from the set Y_{i2}, gives him rank 2. Cardinality of the set of bundles of rank 2 is ${}^{Yi1}C_{ai-2} \times {}^{Yi2}C_2$.

⋮

– Any bundle of a_i resources of which any 1 resource is from the set Y_{i1} and any $(a_i - 1)$ resources are from the set Y_{i2}, gives him rank $(a_i - 1)$. Cardinality of the set of bundles of rank $(a_i - 1)$ is ${}^{Yi1}C_1 \times {}^{Yi2}C_{ai-1}$.

– Any bundle of a_i resources from the set Y_{i2} gives him rank a_i. Cardinality of the set of bundles of rank a_i is ${}^{Yi2}C_{ai}$.

⋮

- An agent's bundle ranking is restricted to a subset of 2^R - it is a partial ranking.

- An agent's true preference Y_{i1}, Y_{i2}, …, Y_{ik} is private information known only to agent i.
- Every agent reports its preference (partition of resources) to the central agent.
- Based on the preferences (partition of the resources) of each agent, the central agent ranks the bundles of resources (each of cardinality a_i) for every agent i.
- The central agent orders the agents based on priority, *i.e.* a permutation on the set of agents N.

Let us illustrate the agent ranking with the help of the following example.

Example 3.1

Consider an example with 10 resources and 4 agents.

- R = {a, b, c, d, e, f, g, h, i, j} $\Rightarrow m = 10$.
- N = {1, 2, 3, 4} $\Rightarrow n = 4$.
- The agents require 2, 3, 3, 2 resources respectively; $\sum a_i = 10$ (*i.e.* m).
- The agents partition the set of resources as follows - from left to right in descending order of preference:

	Y_{i1}	Y_{i2}	Y_{i3}	Y_{i4}
Agent 1	a, b, c, j	d, e, f	g, h, i	
Agent 2	c, d, e	a, b, f	g, h, i, j	
Agent 3	e, f, g	a, h, i	b, c, d, j	
Agent 4	g, h, i	a, b, j	c, d, e	f

- Agent 1's rank 0 bundles are {(a,b), (a,c), (a,j), (b,c), (b,j), (c,j)}.
  - Its rank 1 bundles are {(a,d), (a,e), (a,f), …}.
  - Its rank 2 bundles are {(d,e), (d,f), (e,f)},
  - Its rank 3 bundles are {(d,g), (d,h), (d,i), …}
  - Its rank 4 bundles are {(g,h), (g,i), (h,i)}.
- Agent 2's rank 0 bundles are {(c,d,e)}, rank 1 bundles are {(c,d,a), (c,d,b), (c,d,f), …}, … and so on for the subsequent rankings and agents.

4 Algorithm: A Generalized Serial Dictatorship with Ties

$A_{i,j}$ = Set of agent i's j^{th} rank bundles.

- $A_{i,0}$ = Set of agent i's 0^{th} rank bundles.
- $A_{i,1}$ = Set of agent i's 1^{st} rank bundles.
- ⋮

Algorithm:

Let's say the ordering of the agents is the identity permutation, *i.e.* $1 \succ 2 \succ 3 \succ \ldots \succ n$.

Round 1:

Step 0: $A_{1,0}$ = set of agent 1's 0^{th} rank bundles.

Initialize $A_{1,0}^{1,0} \leftarrow A_{1,0}$.

Step 1: Consider $A_{1,0}^{1,0}$

- Check for intersection bundles of the sets $A_{2,j}$ (j = 0, 1, 2, …) with the bundles of the set $A_{1,0}^{1,0}$. *I.e.* compare agent 2's bundles with agent 1's highest ranked bundles.
- Update $A_{2,j}^{1,1} \leftarrow A_{2,j}$ (j = 0, 1, 2, …) by picking up bundles from the set $A_{2,j}$ (j = 0, 1, 2, …) which are disjoint with at least one bundle from the set $A_{1,0}^{1,0}$. Update the ranking of agent 2 to $A_{2,0}^{1,1}$, $A_{2,1}^{1,1}$, $A_{2,2}^{1,1}$, $A_{2,3}^{1,1}$, ... each of which is a set of bundles which are disjoint with at least one bundle of set $A_{1,0}^{1,0}$.
- Update $A_{1,0}^{1,1} \leftarrow A_{1,0}^{1,0}$, where $A_{1,0}^{1,1}$ is the subset of those bundles of $A_{1,0}^{1,0}$ which are disjoint with at least one bundle of $A_{2,0}^{1,1}$.

Step 2: Consider $A_{1,0}^{1,1}$

- Check for intersection bundles of the sets $A_{3,j}$ (j = 0, 1, 2, …) with the bundles of the set $A_{1,0}^{1,1}$. *I.e.* compare agent 3's bundles with agent 1's highest ranked bundles.
- Update $A_{3,j}^{1,1} \leftarrow A_{3,j}$ (j = 0, 1, 2, …) by picking up bundles from the set $A_{3,j}$ (j = 0, 1, 2, …) which are disjoint with at least one bundle from the set $A_{1,0}^{1,1}$. Update the ranking of agent 3 to $A_{3,0}^{1,1}$, $A_{3,1}^{1,1}$, $A_{3,2}^{1,1}$, $A_{3,3}^{1,1}$, ... each of which is a set of bundles which are disjoint with at least one bundle of set $A_{1,0}^{1,1}$.
- Update $A_{1,0}^{1,2} \leftarrow A_{1,0}^{1,1}$, where $A_{1,0}^{1,2}$ is the subset of those bundles of $A_{1,0}^{1,1}$ which are disjoint with at least one bundle of $A_{3,0}^{1,1}$.

⋮

Step (n – 1) : Consider $A_{1,0}^{1,(n-2)}$

- Check for intersection bundles of the sets $A_{n,j}$ (j = 0, 1, 2, …) with the bundles of the set $A_{1,0}^{1,(n-2)}$. *I.e.* compare agent n's bundles with agent 1's highest ranked bundles.
- Update $A_{n,j}^{1,1} \leftarrow A_{n,j}$ (j = 0, 1, 2, …) by picking up bundles from the set $A_{n,j}$ (j = 0, 1, 2, …) which are disjoint with at least one bundle from the set $A_{1,0}^{1,(n-2)}$. Update the ranking of agent n to $A_{n,0}^{1,1}$, $A_{n,1}^{1,1}$, $A_{n,2}^{1,1}$, $A_{n,3}^{1,1}$, ... each of which is a set of bundles which are disjoint with at least one bundle of set $A_{1,0}^{1,(n-2)}$.
- Update $A_{1,0}^{1,(n-1)} \leftarrow A_{1,0}^{1,(n-2)}$, where $A_{1,0}^{1,(n-1)}$ is the subset of those bundles of $A_{1,0}^{1,(n-2)}$ which are disjoint with at least one bundle of $A_{n,0}^{1,1}$.

Step n: $A_{1,0}^{1,n} = A_{1,0}^{1,(n-1)}$

Round 2:

Step 0: Update the rankings of the bundles from $A_{i,0}^{1,1}$, $A_{i,1}^{1,1}$, $A_{i,2}^{1,1}$, $A_{i,3}^{1,1}$, … to $A_{i,0}^{2,1}$, $A_{i,1}^{2,1}$, $A_{i,2}^{2,1}$, $A_{i,3}^{2,1}$, … for every agent i = 2, 3, …, n, where $A_{i,j}^{2,1}$ (i = 2, 3, …, n; j = 0, 1, 2, …) are the new sets of bundles which have empty intersection with at least one bundle of $A_{1,0}^{1,n}$.

Step 1: Consider $A_{2,0}^{2,1}$

- Update $A_{3,j}^{2,2} \leftarrow A_{3,j}^{2,1}$ (j = 0, 1, 2, …) by picking up bundles from the set $A_{3,j}^{2,1}$ (j = 0, 1, 2, …) which are disjoint with at least one bundle from the set $A_{2,0}^{2,1}$. Update the ranking of agent 3 to $A_{3,0}^{2,2}$, $A_{3,1}^{2,2}$, $A_{3,2}^{2,2}$, $A_{3,3}^{2,2}$, ... each of which is a set of bundles which are disjoint with at least one bundle of set $A_{2,0}^{2,1}$.
- Update $A_{2,0}^{2,2} \leftarrow A_{2,0}^{2,1}$, where $A_{2,0}^{2,2}$ is the subset of those bundles of $A_{2,0}^{2,1}$ which are disjoint with at least one bundle of $A_{3,0}^{2,2}$.

Step 2: Consider $A_{2,0}^{2,2}$

- Update $A_{4,j}^{2,2} \leftarrow A_{4,j}^{2,1}$ (j = 0, 1, 2, …) by picking up bundles from the set $A_{4,j}^{2,1}$ (j = 0, 1, 2, …) which are disjoint with at least one bundle from the set $A_{2,0}^{2,2}$. Update the ranking of agent 4 to $A_{4,0}^{2,2}$, $A_{4,1}^{2,2}$, $A_{4,2}^{2,2}$, $A_{4,3}^{2,2}$, ... each of which is a set of bundles which are disjoint with at least one bundle of set $A_{2,0}^{2,2}$.
- Update $A_{2,0}^{2,3} \leftarrow A_{2,0}^{2,2}$, where $A_{2,0}^{2,3}$ is the subset of those bundles of $A_{2,0}^{2,2}$ which are disjoint with at least one bundle of $A_{4,0}^{2,2}$.

⋮

Step (n – 1) : Consider $A_{2,0}^{2,(n-2)}$

- Update $A_{n,j}^{2,2} \leftarrow A_{n,j}^{2,1}$ (j = 0, 1, 2, …) by picking up bundles from the set $A_{n,j}^{2,1}$ (j = 0, 1, 2, …) which are disjoint with at least one bundle from the set $A_{2,0}^{2,(n-2)}$. Update the ranking of agent n to $A_{n,0}^{2,2}$, $A_{n,1}^{2,2}$, $A_{n,2}^{2,2}$, $A_{n,3}^{2,2}$, ... each of which is a set of bundles which are disjoint with at least one bundle of set $A_{2,0}^{2,(n-2)}$.
- Update $A_{2,0}^{2,(n-1)} \leftarrow A_{2,0}^{2,(n-2)}$, where $A_{2,0}^{2,(n-1)}$ is the subset of those bundles of $A_{2,0}^{2,(n-2)}$ which are disjoint with at least one bundle of $A_{n,0}^{2,2}$.

Step n: $A_{2,0}^{2,n} = A_{2,0}^{2,(n-1)}$

⋮

Round (n – 1):

Step 0: Update the rankings of the bundles from $A_{i,0}^{(n-2),(n-2)}$, $A_{i,1}^{(n-2),(n-2)}$, $A_{i,2}^{(n-2),(n-2)}$, $A_{i,3}^{(n-2),(n-2)}$, … to $A_{i,0}^{(n-1),(n-2)}$, $A_{i,1}^{(n-1),(n-2)}$, $A_{i,2}^{(n-1),(n-2)}$, $A_{i,3}^{(n-1),(n-2)}$, … for every agent $i = n$-1, n, where $A_{i,j}^{(n-2),(n-2)}$ ($i = n$-1, n; j = 0, 1, 2, …) are the new sets of bundles which have empty intersection with at least one bundle of $A_{(n-2),0}^{(n-2),n}$.

Step 1: Consider $A_{(n-1),0}^{(n-1),(n-2)}$

- Update $A_{n,j}^{(n-1),(n-1)} \leftarrow A_{n,j}^{(n-1),(n-2)}$ (j = 0, 1, 2, …) by picking up bundles from the set $A_{n,j}^{(n-1),(n-2)}$ (j = 0, 1, 2, …) which are disjoint with at least one bundle from the set $A_{(n-1),0}^{(n-1),(n-2)}$. Update the ranking of agent n to $A_{n,0}^{(n-1),(n-1)}A_{n,1}^{(n-1),(n-1)}$, $A_{n,2}^{(n-1),(n-1)}$, $A_{n,3}^{(n-1),(n-1)}$, ... each of which is a set of bundles which are disjoint with at least one bundle of set $A_{(n-1),0}^{(n-1),(n-2)}$.
- Update $A_{(n-1),0}^{(n-1),(n-1)} \leftarrow A_{(n-1),0}^{(n-1),(n-2)}$, where $A_{(n-1),0}^{(n-1),(n-1)}$is the subset of those bundles of $A_{(n-1),0}^{(n-1),(n-2)}$ which are disjoint with at least one bundle of $A_{n,0}^{(n-1),(n-1)}$.

Step 2: $A_{(n-1),0}^{(n-1),(n-1)} = A_{(n-1),0}^{(n-1),n}$

Round n:

Update the rankings of the bundles from $A_{n,0}^{(n-1),(n-1)}$, $A_{n,1}^{(n-1),(n-1)}$, $A_{n,2}^{(n-1),(n-1)}$, $A_{n,3}^{(n-1),(n-1)}$, … to $A_{n,0}^{n,(n-1)}$, $A_{n,1}^{n,(n-1)}$, $A_{n,2}^{n,(n-1)}$, $A_{n,3}^{n,(n-1)}$, …, where $A_{n,j}^{n,(n-1)}$ (j = 0, 1, 2, …) are the new sets of bundles of agent n which have empty intersection with at least one bundle of $A_{(n-1),0}^{(n-1),n}$. $A_{n,0}^{n,n} = A_{n,0}^{n,(n-1)}$.

The solution will be the feasible allocation from the set

$$\{(a_1, a_2, \ldots, a_n)/a_i \in A_{i,0}^{i,n} \text{ for all } i = 1, 2, 3, \ldots, n\}$$

Let us consider the example 3.1, and run the algorithm on it.

Example 4.1

Let the ordering of the agents in the example 3.1 be the identity permutation.

Round 1:

Step 0: $A_{1,0} = A_{1,0}^{1,0} = \{(a,b), (a,c), (a,j), (b,c), (b,j), (c,j)\}$

Step 1: Consider $A_{1,0}^{1,0} = \{(a,b), (a,c), (a,j), (b,c), (b,j), (c,j)\}$

- $A_{2,0}^{1,1} = \{(c,d,e)\}$

$A_{2,1}^{1,1} = \{(c,d,a), (c,d,b), (c,d,f), \ldots\}$

$\vdots$

$A_{2,6}^{1,1} = \{(g,h,i), (g,h,j), (g,i,j), (h,i,j)\}$

- $A_{1,0}^{1,1} = \{(a,b), (a,j), (b,j)\}$

Step 2: Consider $A_{1,0}^{1,1} = \{(a,b), (a,j), (b,j)\}$

- $A_{3,0}^{1,1} = \{(e,f,g)\}$

$A_{3,1}^{1,1} = \{(e,f,a), (e,f,h), (e,f,i), \ldots\}$

$\vdots$

$A_{3,6}^{1,1} = \{(b,c,d), (c,d,j)\}$

- $A_{1,0}^{1,2} = \{(a,b), (a,j), (b,j)\}$

Step 3 : Consider $A_{1,0}^{1,2} = \{(a,b), (a,j), (b,j)\}$

- $A_{4,0}^{1,1} = \{(g,h), (g,i), (h,i)\}$

$A_{4,1}^{1,1} = \{(g,a), (g,b), (g,j), \ldots\}$

$\vdots$

$A_{4,4}^{1,1} = \{(c,f), (d,f), (e,f)\}$

- $A_{1,0}^{1,3} = \{(a,b), (a,j), (b,j)\}$

Step 4: $A_{1,0}^{1,4} = A_{1,0}^{1,3} = \{(a,b), (a,j), (b,j)\}$

Round 2:

Step 0: Consider $A_{1,0}^{1,4} = \{(a,b), (a,j), (b,j)\}$

$A_{2,0}^{2,1} = \{(c,d,e)\}$, $A_{2,1}^{2,1} = \ldots$

$A_{3,0}^{2,1} = \{(e,f,g)\}$, $A_{3,1}^{2,1} = \ldots$

$A_{4,0}^{2,1} = \{(g,h), (g,i), (h,i)\}$, $A_{4,1}^{2,1} = \ldots$

Step 1: Consider $A_{2,0}^{2,1} = \{(c,d,e)\}$

- $A_{3,0}^{2,1} = \{(f,g,a), (f,g,h), (f,g,i)\}$

$A_{3,1}^{2,1} = \{(a,f,h), (a,f,i), (f,h,i), \ldots\}$

$\vdots$

$A_{3,3}^{2,1} = \{(b,h,i), (h,i,j)\}$

- $A_{2,0}^{2,2} = \{(c,d,e)\}$

Step 2: Consider $A_{2,0}^{2,2} = \{(c,d,e)\}$

- $A_{4,0}^{2,1} = \{(g,h), (g,i), (h,i)\}$

$A_{4,1}^{2,1} = \{(a,g), (b,g), (g,j), \ldots\}$

- $A_{2,0}^{2,3} = \{(c,d,e)\}$

Step 3 : $A_{2,0}^{2,4} = A_{2,0}^{2,3} = \{(c,d,e)\}$

Round 3:

Step 0: Consider $A_{2,0}^{2,4} = \{(c,d,e)\}$

$A_{3,0}^{3,2} = \{(f,g,a), (f,g,h), (f,g,i)\}$, $A_{3,1}^{3,2} = \ldots$
$A_{4,0}^{3,2} = \{(g,h), (g,i), (h,i)\}$, $A_{4,1}^{3,2} = \ldots$

Step 1: Consider $A_{3,0}^{3,2} = \{(f,g,a), (f,g,h), (f,g,i)\}$

- $A_{4,0}^{3,3} = \{(h,i)\}$

$A_{4,1}^{3,3} = \{(a,h), (b,h), (j,h), \ldots\}$

- $A_{3,0}^{3,3} = \{(f,g,a)\}$

Step 3 : $A_{3,0}^{3,4} = A_{3,0}^{3,3} = \{(f,g,a)\}$

Round 4:

Step 0: Consider $A_{3,0}^{3,4} = \{(f,g,a)\}$

$A_{4,0}^{4,3} = \{(h,i)\}$, $A_{4,1}^{4,3} = \ldots$

Step 1: $A_{4,0}^{4,4} = A_{3,0}^{4,3} = \{(h,i)\}$

Allocation:

We have

i. $A_{1,0}^{1,4} = \{(a,b), (a,j), (b,j)\}$
ii. $A_{2,0}^{2,4} = \{(c,d,e)\}$
iii. $A_{3,0}^{3,4} = \{(f,g,a)\}$
iv. $A_{4,0}^{4,4} = \{(h,i)\}$

The only feasible solution is: **{(b,j), (c,d,e), (f,g,a), (h,i)}**

The allocation is ***Pareto-optimal***.

5 Results

In this section, we will show some properties of the $A_{i,0}^{i,n}$ sets. We will also prove an important lemma which will subsequently be used in proving the Pareto-efficiency of the mechanism.

Properties:

1) $A_{1,0}^{1,n} = A_{1,0}^{1,(n-1)} \subseteq A_{1,0}^{1,(n-2)} \subseteq \ldots \subseteq A_{1,0}^{1,2} \subseteq A_{1,0}^{1,1} \subseteq A_{1,0}^{1,0} = A_{1,0}$
2) $A_{2,0}^{2,n} = A_{2,0}^{2,(n-1)} \subseteq A_{2,0}^{2,(n-2)} \subseteq \ldots \subseteq A_{2,0}^{2,2} \subseteq A_{2,0}^{2,1} \subseteq A_{2,0}^{1,1}$
3) $A_{3,0}^{3,n} = A_{3,0}^{3,(n-1)} \subseteq A_{3,0}^{3,(n-2)} \subseteq \ldots \subseteq A_{3,0}^{3,2} \subseteq A_{3,0}^{2,2}$
4) $A_{4,0}^{4,n} = A_{4,0}^{4,(n-1)} \subseteq A_{4,0}^{4,(n-2)} \subseteq \ldots \subseteq A_{4,0}^{4,3} \subseteq A_{4,0}^{3,3}$
⋮

n) $A_{n,0}^{n,n} = A_{n,0}^{n,(n-1)} \subseteq A_{n,0}^{(n-1),(n-1)}$

n+1) $r(a_i)$ is the same for all $a_i \in A_{i,0}^{i,n}$.

We omit the proofs for its obviousness.

Lemma 1:

Let $a \in (a_1, a_2, \ldots, a_n) \in (A_{1,0}^{1,n}, A_{2,0}^{2,n}, \ldots, A_{n,0}^{n,n})$. Then there does not exist any feasible allocation $(b_1, b_2, \ldots, b_n)$ such that there exists $i \in N$ and

1. $\text{rank}(b_i) > \text{rank}(a_i)$
2. $\text{rank}(b_j) = \text{rank}(a_j)$ $\forall\, j = 1, 2, \ldots, i - 1.$
3. $\text{rank}(b_j) \geq \text{rank}(a_j)$ $\forall\, j = i + 1, \ldots, n.$

Proof:

If $i = 1$, then $r(b_1) > r(a_1)$, and $r(b_i) \geq r(a_i)\ \forall i \geq 2$. But $a_1 \in A_{1,0}^{1,n}$, which is a subset of $A_{1,0}^{1,0} = A_{1,0}$ *i.e.* the set of highest ranked bundles of agent 1, therefore $r(a_i) = 0$. So $r(b_1) \not\succ r(a_1)$. Thus it is clear that $i \neq 1$.

So let $i = 2$. Then

i. $r(b_2) > r(a_2)$
ii. $r(b_1) = r(a_1)$
iii. $r(b_i) \geq r(a_i),\ \forall\, i \geq 3.$

Now, $r(b_1) = r(a_1)$ and $a_1 \in A_{1,0}^{1,0}$ ($\because A_{1,0}^{1,n} \subseteq A_{1,0}^{1,0}$ and $a_1 \in A_{1,0}^{1,n}$), therefore

$$b_1 \in A_{1,0}^{1,0} \tag{1}$$

Again, $A_{2,0}^{1,1}$ is the set of highest ranked bundles of agent 2 such that for any bundle of $A_{2,0}^{1,1}$ there exists a disjoint bundle in $A_{1,0}^{1,0}$. Now, *w.l.o.g.*, we can safely assume (for the sake of simplification of the analysis) that b_2 is the highest ranked bundle of agent 2 such that $r(b_2) > r(a_2)$ with $b_2 \neq b_1$ and $r(b_1) = r(a_1)$. So $b_2 \in A_{2,0}^{1,1}$ (from (1)). Now, from property (n+1), $r(p)$ is the same for all $p \in A_{2,0}^{1,1}$. Since $b_2 \in A_{2,0}^{1,1}$ and $r(b_2) > r(a_2)$: $a_2 \notin A_{2,0}^{1,1}$. From property (2) we have $A_{2,0}^{2,n} \subseteq A_{2,0}^{1,1}$, which implies that $a_2 \notin A_{2,0}^{2,n}$. This is a contradiction, therefore $r(b_2) \not\succ r(a_2)$, and $i \neq 2$.

So let $i = 3$. Then

i. $r(b_3) > r(a_3)$
ii. $r(b_1) = r(a_1),\ r(b_2) = r(a_2)$
iii. $r(b_i) \geq r(a_i),\ \forall\, i \geq 4.$

$A_{2,0}^{1,1}$ is the set of highest ranked bundles of agent 2 such that for any bundle of $A_{2,0}^{1,1}$ there exists a disjoint bundle in $A_{1,0}^{1,0}$. Now, since $b_2 \neq b_1 \in A_{1,0}^{1,0}$, $r(b_2) = r(a_2)$. Now $a_2 \in A_{2,0}^{1,1}$ ($\because a_2 \in A_{2,0}^{2,n} \subseteq A_{2,0}^{1,1}$ from property (2)), so b_2 is also the highest ranked bundle of agent 2 which is disjoint with $b_1 \in A_{1,0}^{1,1}$. Therefore

$$b_2 \in A_{2,0}^{1,1} \tag{2}$$

Now, *w.l.o.g.*, as in case for $i = 2$, we can safely assume that b_3 is the highest ranked bundle of agent 3 such that $r(b_3) > r(a_3)$ with $b_3 \neq b_1, b_3 \neq b_2, b_2 \neq b_1$, and $r(b_1) = r(a_1)$, $r(b_2) \geq r(a_2)$. Now, arguing as in the case for $i = 2$, we can show that $a_3 \notin A_{3,0}^{3,n}$. Now, going through steps 1 to n of round 1, we will show that $b_1 \in A_{1,0}^{1,n}$.

Step 1:

- $b_2 \in A_{2,0}^{1,1}$ (from (2))
- $b_1 \in A_{1,0}^{1,1}$ (since $b_1 \neq b_2$, and $b_1 \in A_{1,0}^{1,0}$)

Step 2:

- $b_3 \in A_{3,0}^{1,1}$ (since $b_3 \neq b_1$, and $b_1 \in A_{1,0}^{1,1}$)
- $b_1 \in A_{1,0}^{1,2}$ (since $b_1 \neq b_3$, and $b_1 \in A_{1,0}^{1,1}$)

Step 3:

- $b_4 \in A_{4,0}^{1,1}$ (since $b_4 \neq b_1$, and $b_1 \in A_{1,0}^{1,2}$; assume *w.l.o.g.*, b_4 is the highest ranked bundle of agent 4 *s.t.* $b_4 \neq p$ and $p \in A_{1,0}^{1,2}$)
- $b_1 \in A_{1,0}^{1,3}$ (since $b_1 \neq b_4$, and $b_1 \in A_{1,0}^{1,2}$)

⋮

Step n - 1:

- $b_n \in A_{n,0}^{1,1}$ (since $b_n \neq b_1$, and $b_1 \in A_{1,0}^{1,(n-2)}$; assume *w.l.o.g.*, b_n is the highest ranked bundle of agent n *s.t.* $b_n \neq q$ and $q \in A_{1,0}^{1,(n-2)}$)
- $b_1 \in A_{1,0}^{1,(n-1)}$ (since $b_1 \neq b_n$, and $b_1 \in A_{1,0}^{1,(n-2)}$)

Step n:

- $b_1 \in A_{1,0}^{1,(n-1)} = A_{1,0}^{1,n}$

Thus we have shown that $b_1 \in A_{1,0}^{1,n}$. Now $b_2 \in A_{2,0}^{1,1}$ (from step 1) and $b_2 \neq b_1$ with $b_1 \in A_{1,0}^{1,n}$. Therefore

$$b_2 \in A_{2,0}^{2,1} \tag{3}$$

Also, $b_3 \in A_{3,0}^{1,1}$ (from step 2) and $b_3 \neq b_1$ with $b_1 \in A_{1,0}^{1,n}$. Therefore

$$b_3 \in A_{3,0}^{2,1} \tag{4}$$

Now, since $b_3 \neq b_2$, from (3) and (4), we get $b_3 \in A_{3,0}^{2,2}$. As $r(b_3) > r(a_3)$ and $b_3 \in A_{3,0}^{2,2}$, we get $a_3 \notin A_{3,0}^{2,2}$ ($r(p)$ is the same for all $p \in A_{3,0}^{2,2}$). Now, since $A_{3,0}^{3,n} \subseteq A_{3,0}^{2,2}$ (from property (3)), we get $a_3 \notin A_{3,0}^{2,2}$, which is a contradiction. Therefore $r(b_3) \not> r(a_3)$, and $i \neq 3$.

Going ahead with the same argument, we can prove that there does not exist any $i \in N$ such that

1. $r(b_i) > r(a_i)$

2. $r(b_j) = r(a_j)$ $\forall\, j = 1, 2, \ldots, i - 1.$
3. $r(b_j) \geq r(a_j)$ $\forall\, j = i + 1, \ldots, n.$

Therefore, for any $a \in (a_1, a_2, \ldots, a_n) \in (A_{1,0}^{1,n}, A_{2,0}^{2,n}, \ldots, A_{n,0}^{n,n})$, there does not exist any feasible allocation $(b_1, b_2, \ldots, b_n)$ which satisfies the above.

Theorem A: The proposed mechanism is Pareto-efficient and strategy-proof.

Proof:

Pareto-efficiency:

Suppose allocation $a \in (a_1, a_2, \ldots, a_n)$ is the solution given by the mechanism, where $a_i \in A_{i,0}^{i,n}$ $\forall i \in N$.

Suppose a is not a pareto-efficient allocation. Then there exists a feasible allocation $b = (b_1, b_2, \ldots, b_n)$ which Pareto-dominates the allocation a. Then $r(b_i) \geq r(a_i)$ $\forall i \in N$ and there exists j such that $r(b_j) > r(a_j)$.

Now we can see that $j \neq 1$. We know that $a_1 \in A_{1,0}^{1,n}$, and $A_{1,0}^{1,n} \subseteq A_{1,0}$, where $A_{1,0}$ is the set of highest ranked bundles of agent 1, therefore a_1 is the highest ranked bundle of agent 1. So $r(b_1) \not\succ r(a_1)$. Therefore $j \neq 1$.

Now, as b dominates a, there exists $i \in N$ such that i is the first agent such that

1. $r(b_i) > r(a_i)$
2. $r(b_j) = r(a_j)$ $\forall\, j = 1, 2, \ldots, i - 1.$
3. $r(b_j) \geq r(a_j)$ $\forall\, j = i + 1, \ldots, n.$

But by the lemma, there does not exist any such allocation $b = (b_1, b_2, \ldots, b_n)$ such that there exists $i \in N$ and

1. $r(b_i) > r(a_i)$
2. $r(b_j) = r(a_j)$ $\forall\, j = 1, 2, \ldots, i - 1.$
3. $r(b_j) \geq r(a_j)$ $\forall\, j = i + 1, \ldots, n.$

Thus there does not exist any feasible allocation $b = (b_1, b_2, \ldots, b_n)$ which Pareto-dominates the allocation $a \in (a_1, a_2, \ldots, a_n)$ given by the mechanism as a solution.
Thus $a \in (a_1, a_2, \ldots, a_n)$ is a Pareto-optimal allocation, and since a is arbitrary, the mechanism is Pareto-efficient.

Strategy-proofness:

Agent 1 does not get anything better by misreporting his preferences, as he is guaranteed one of his most preferred bundles, *i.e.* a bundle from the set $A_{1,0}$. Hence there is no incentive for agent 1 to lie.

Further, agent 1's preference does not depend upon agent 2's preference. The mechanism ensures the best ranked bundle (that does not overlap with at least one highest ranked bundle of agent 1) that could be possible for agent 2, therefore agent 2 will also not be benefited by misreporting his preference.

The same argument holds for all the agents. Thus the mechanism is strategy-proof.

6. Implementation and Case Study

6.1. Implementation

Java and JADE have been used to develop a multi-agent platform and interface through which agent systems and allocations can be loaded and updated.

Java was chosen as the implementation language as it is object-oriented, type-safe and user-friendly. The framework JADE (**J**ava **A**gent **DE**velopment framework) is fully developed in Java. JADE simplifies the implementation of multi-agent systems through a FIPA compliant middle-ware and a set of graphical tools that supports the debugging and deployment phases. Fig. 6.1.1 shows a snapshot of the JADE graphical tool.

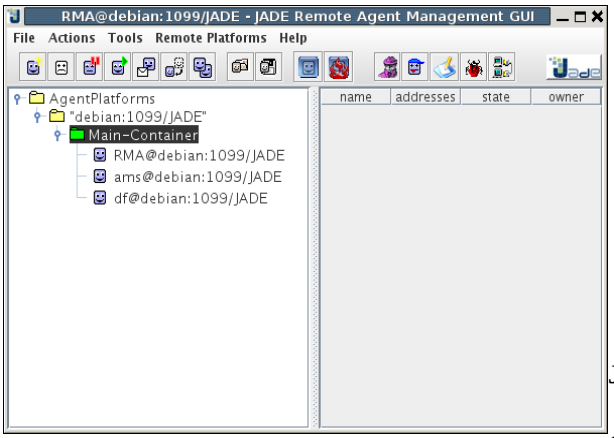

Fig. 6.1.1

JADE Key Concepts

1) Behaviors: Behaviors implement the tasks of an agent. They are logical activity units that can be composed to achieve complex execution patterns which can run concurrently.
2) Communication/messages: Messages are exchanged between agents as Agent Communication Language (ACL).
3) Interaction Protocols: Patterns of message exchange between agents based on communicative acts are specified by FIPA.

Features and Advantages of JADE

1) JADE allows intelligence, information, initiative, resources and control to be fully distributed.
2) The environment can evolve dynamically with agents that can appear in the system according to the needs and requirements of the application environment.
3) Agents can dynamically discover and communicate with other agents.
4) JADE hides the complexity of distributed architecture composed of autonomous entities. Thus the focus of software development can be logic of the application.
5) JADE is a well-documented open-source project.

Components of an Agent System

1. Resource: In this problem, we consider discrete, indivisible, non-shareable, static and single-unit objects. Thus resources have been implemented as uniquely identifiable String class objects.
2. Bundle: Bundles are implemented as unordered collections of resources, from which resources can be removed or added.
3. Preference Ranking: Preference ranking is implemented as an ordered list of bundles of resources, from most-preferred to least-preferred.
4. Agents: Agents need to fulfill certain requirements -

- They have preferences over bundles, and are autonomous entities requesting and receiving resources.
- They respond to communication initiated by the central agent.

- They are uniquely identifiable and discoverable.

The agents have been implemented as objects of a subclass of the JADE Agent class.

5. Allocation: An allocation has been implemented as an ordered list of bundles, with each bundle corresponding to an agent in the system. This implementation allows determination of feasibility as well as testing for various measures of social welfare.

Allocation Algorithm

The allocation algorithms have been built as interaction protocol based communicative acts and interleaving of behaviors.

The interaction protocol allows the central agent (initiator) to elicit the preference rankings of a set of agents (responders), and then make an allocation based on preference rankings.

The JADE Remote Agent Management GUI (Fig. 6.1.2) is used to start a new agent, and the parameters for the central agent are entered, *i.e.* the list of resources available in the agent system.

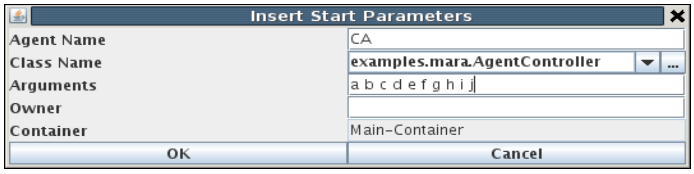

Fig. 6.1.2

All the other agents in the system are now started (Fig. 6.1.3) and their start parameters, i.e. number of resources required and the preference ranking, are entered.

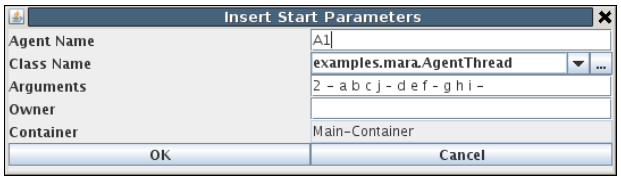

Fig. 6.1.3

Another GUI (Fig. 6.1.4) is activated, with certain functionalities. Using this, one can update the list of agents, trigger an allocation algorithm and see the list of free and held resources.

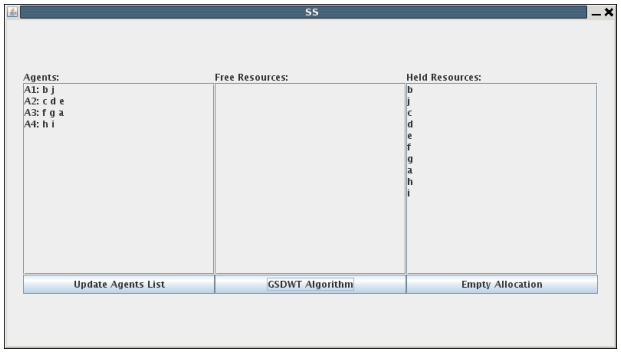

Fig. 6.1.4

User action, such as clicking a button, causes behaviors to be added to the agents and run concurrently. The behaviours involved when an allocation algorithm is triggered are:

- Ranking Inform Behavior – This cyclic (responder) behavior is part of each non-central agent from start-up. The behavior responds to requests from the central agent for its preference ranking of bundles.
- Central Allocator Behavior – This behavior is added to the central agent to initiate communication. It requests the bundle rankings from the agents, receives the responses, runs the allocation procedure and informs the agents of their allocated bundles. If the response from an agent is a failure message, then the bundle ranking for that agent is taken to be a single bundle, *i.e.* the empty bundle.
- Update Resources Behavior – This cyclic (responder) behavior is part of each non-central agent from start-up. It receives the allocated bundle from the central agent and updates the resources of the agent.

Allocation Procedure

The Central Allocator Behavior added to the central agent computes allocations by running the Generalised Serial Dictatorship with Ties allocation procedure.

Pseudocode of the GSDwT mechanism

Input**:** Resources, agents, agent preferences, agent requirement.
Output**:** List of pareto-optimal allocations.
Functions called: set_rankings(), remove_unobtainable_resources(), refine_lists(), get_allocations().

/* Terminology:
- agent_preferences – partition of set of resources for each agent.
- agent requirement a_i – number of resources required by agent i.

*/

```
1.    START GSDwT
2.    READ agent_preferences (input).
3.    Set agent_rankings ← SET_RANKINGS().
4.    FOR each agent i
5.    Set agent_rankings ←          REMOVE_UNOBTAINABLE_RESOURCES(i).
6.    Set agent_rankings ← REFINE_LISTS(i).
7.    Set agent_rankings[i] ← top-ranked-bundles of  agent_rankings[i].
8.    END FOR.
9.    Set solution_set = GET_ALLOCATIONS().
10.   STOP.
```

Function**:** set_rankings().
Called by: GSDwT.
Functions called: none.

```
1.    FUNCTION SET_RANKINGS()
2.    FOR each agent i
3.    Initialise list i_rankings = {}.
4.    FOR each bundle b in  agent_preferences[i]
5.          Create all possible combinations of size ai, using resources from bundles b and b+1.
6.    Rank each combination, and add to i_rankings.
7.    Sort i_rankings by rank of the combinations.
8.       END FOR
9.       Add i_rankings to agent_rankings.
10.   END FOR
11.   RETURN agent_rankings.
```

Function**:** remove_unobtainable_resources().

Called by**:** GSDwT.
Functions called**:** none.

1. FUNCTION REMOVE_UNOBTAINABLE_RESOURCES(i)
2. FOR each agent j = i to n
3. Update agent_rankings[j] such that its bundles are disjoint with at least one bundle of the top-ranked-bundles of agent_rankings[i].
4. END FOR.
5. RETURN agent_rankings.

Function**:** refine_lists().
Called by**:** GSDwT.
Functions called**:** none.

1. FUNCTION REFINE_LISTS(i)
2. FOR each agent j = i+1 to n
3. Update agent_rankings[j] such that its bundles are disjoint with at least one bundle of the top-ranked-bundles of agent_rankings[i].
4. Update top-ranked-bundles of agent_rankings[i] such that its bundles are disjoint with at least one bundle of the top-ranked-bundles of agent_rankings[j].
5. END FOR.
6. RETURN agent_rankings.

Function**:** get_allocations().
Called by**:** GSDwT.
Functions called**:** none.

1. FUNCTION GET_ALLOCATIONS()
2. SET solution_set ← full set of feasible allocations such that each agent is assigned a bundle from its top-ranked-bundles.
3. RETURN solution_set.

6.2 Case Study: Site Selection Problem

Problem Statement: A site-selection problem relating to the *selection of suitable areas for the deployment of military units which are part of a military formation*, within the deployable area selected by the Swarm Search Algorithm.

Problem Constraints: The solution of the problem is subject to 3 constraints -
1) Global logic of the military formation.
2) Interactive requirements of the military units.
3) Individual unit preferences for deployment.
Global logic is a *hard constraint* while the latter are *soft constraints*.

Problem Definition: The deployable area is an 8x8 grid consisting of squares which have certain terrain attributes. Each square in the grid represents 1 sq. km. The Swarm Search algorithm is used to select a deployable area off a digital map, using certain basic deployability attributes given to it. A deployable area has no built up area (BUA), no large water bodies and no railway lines. All squares in the grid are deployable. However, every square in the grid

does not satisfy all the individual preferences of units, which are more detailed than the basic attributes indicated to the Swarm Search algorithm. The logic given to the Swarm Search algorithm is a hard constraint – the units cannot deploy outside the area selected by the algorithm.

Resources: Every square has (or not) certain terrain attributes -

1) Water source
2) Telecommunications line
3) Electricity line
4) Road passing through it
5) Proximity to BUA

When squares do not have desirable terrain attributes, their desirability, in comparison to other squares, is a function of their proximity to squares with the relevant attributes.

Agents: We assume six types of military units in the military formation. The number in the brackets denotes how many units of each types are present in the systems. We have a total of 10 units with one agent representing each unit.

1) Combat Unit A (1)
2) Combat Unit B (1)
3) Combat Unit C (3)
4) Headquarters (1)
5) Support Units (2)
6) Logistics Units (2)

Each unit has specific requirements corresponding to the square attributes. If units cannot be assigned squares with the required attributes, they should be assigned squares with proximity to the desired squares.

Agent Preferences: For the purpose of this case study, the military units require specific terrain attributes, some more important than others. The requirements are listed in order of priority. If all individual preferences cannot be satisfied, the *lowest listed ones must be declined first*.

1) Combat Unit A
a) Requires four squares.
b) Not within 2 km of BUA.
c) Within 2 km of a road.
d) Within 2 km of a water source.
2) Combat Unit B
a) Requires two squares.
b) Not within 2 km of BUA.
c) Within 2 km of a road.
d) Within 2 km of a water source.
3) Combat Unit C
a) Requires one square.
b) Within 1 km of a water source.
c) Within 1 km of a road.
4) Headquarters
a) Requires two squares.
b) Within 1 km of a telecom line.
c) Within 1 km of a road.
d) Within 1 km of an electricity source.
5) Support units
a) Requires two squares.
b) Within 1 km of a telecom line.
c) Within 1 km of a road.
d) Within 1 km of an electricity source.
6) Logistics

a) Requires two squares.
b) Within 1 km of a road.
c) Within 1 km of an electricity source.
d) Within 1 km of a telecom line.
e) Within 1 km of a water source.

***Inter se* Priorities:** There are bound to be conflicts in interest over squares with desirable terrain attributes. To resolve such disputes, we consider the *inter se* priority of the units, i.e. which unit is more important in comparison to the other. When global or space constraints limit or deny individual satisfaction, priority setting minimizes the global cost.

1) Combat Unit A, Combat Unit B, and Combat Unit C.
2) Headquarters
3) Support units.
4) Logistics units

Interactive Logic: Units also have preferences with regard to other agents. This differs from individual preferences in that individual preferences deals with requirements of individual units without reference to the other agents.

1) Headquarters and Support units must be collocated with no *inter se* separation. There should be no geographical gap between the squares assigned to these agents.
2) Logistics units should be collocated with an *inter se* separation of 1 square. The units should not be situated too far from or right next to each other.

Global Logic: The global logic will be implemented by a separate planning agent, which represents no unit. It implements the global perspective of the military formation, which is not visible to the individual units. The planning agent devises a formation layout schema within which the units must deploy. Fig. 6.2.1 illustrates one such formation layout schema.

The grid has three tiers of deployment – the forward, middle and rear tiers. The middle tier has a core region.

1) The Combat Unit A, Combat Unit B, and Combat Unit C must be deployed in the forward tier.
2) The Headquarters and the Support units must be deployed in the middle tier, and preferably in the core region. The Headquarters has higher priority for the core region.
3) Logistics must be deployed in the rear tier.

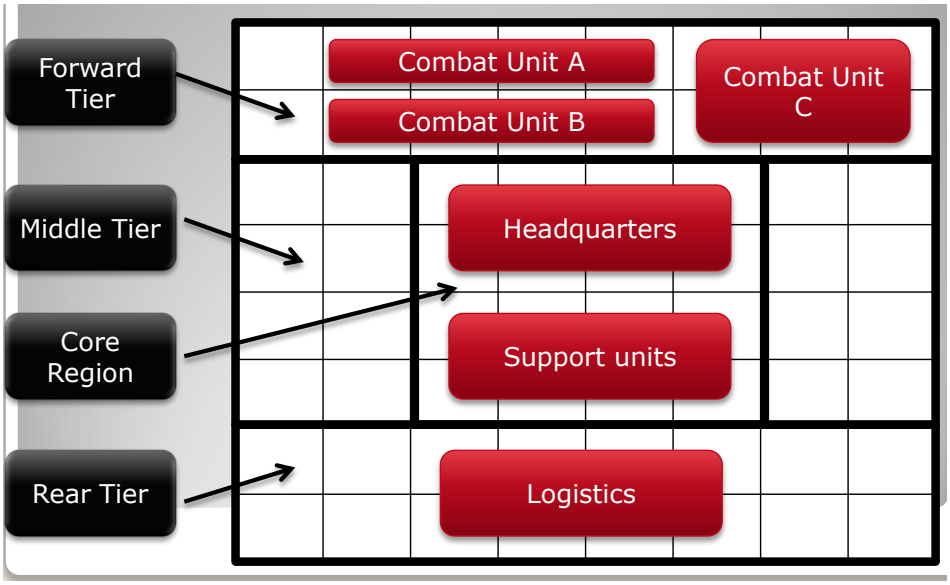

Fig. 6.2.1

Social Welfare: Satisfaction of preferences is based on game theoretic approach. In certain situations it is important to satisfy *most* of the units' preferences, in others only their most critical preferences, and in other situations, only the preferences of the highest priority units must be satisfied.

1) Utilitarian – This ensures that maximum preferences of individual units are satisfied.
2) Egalitarian – This ensures that the most critical preferences will be satisfied.
3) Elitist – This ensures that all or most preferences of higher priority units are met, only critical preferences of lower priority units need to be met.

Solution: Procedure

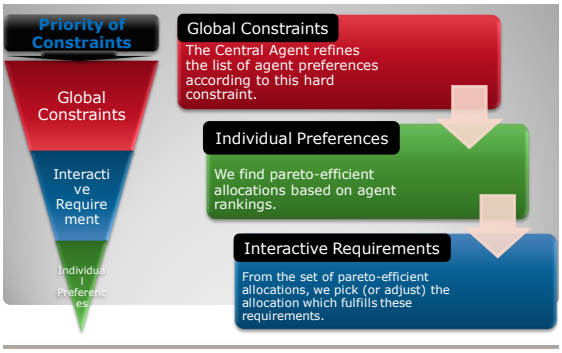

Fig 6.2.2

The procedure to assign squares to the units, taking into consideration all the constraints and the desired welfare characteristics of the final allocation, is in three steps or phases, as illustrated in Fig. 6.2.2.

Phase I:

Agents:
Each agent makes a list of
1) All squares with all the n desirable attributes.
2) All squares with the top n-1 desirable attributes.
3) :
:
In this way, the agent partitions the set of squares in the 8x8 (or n x n) grid.

The partitions may not exhaust the entire set of available resources. In this case, it is assumed that the unranked squares form the lowest preferred partition of squares with no desirable attributes. This preference ranking is communicated to the Central agent (or planning agent).

Central Agent:
The Central Agent collects the rankings from each agent, and refines this list according to the global constraints.

Step 1: From each agent's preference rankings, the central agent removes any squares that do not belong to the region / tier that the agent is allowed to deploy in.
Each partition should contain *at least* as many squares as the unit requires.

Step 2: Making a list of the unranked squares in the tier, the Central Agent adds this list (as the lowest ranked partition) to the bottom of the list obtained in step 1.
In the case of units to be deployed in the middle tier, which contains the core region -
a) The Central Agent first adds the list of unranked squares in the *core region*.

b) Next, the list of unranked squares in the rest of the middle tier is now added as a separate partition.

This phase takes care of the global constraints of the military formation. Being a hard constraint, it needs to be dealt with first.

Phase II:

Central Agent:
The Central Agent ranks the agents in order of priority. The Generalised Serial Dictatorship with Ties algorithm is now used to find all feasible (pareto-optimal) allocations. The algorithm may return one or more allocations.

This phase takes care of the individual preferences of the military units, and returns a list of pareto-optimal allocations. Thus a second constraint is dealt with.

Phase III:

Central Agent:
The Central Agent now considers the allocations, all of which are equally preferable to the agents.

The interactive logic is now considered. If any of the allocations returned by the SDT algorithm fulfils this constraint, then it is the final allocation.

If none of the feasible allocations conform to the interactive logic, one or more of the units need to be relocated. Depending on the situation mentioned under 'Social Welfare', the Central Agent may

a) Relocate both agents to squares which fulfil the interactive constraint for utilitarian and egalitarian allocations. (Perhaps negotiation may be used by the agents.)
b) Relocate the agent with lower priority for elitist allocations.

This is the final allocation.
This phase deals with the interactive logic constraint.

6.3 Toy Example

Consider an 8×8 grid with the attributes as shown in Fig 6.3.1.

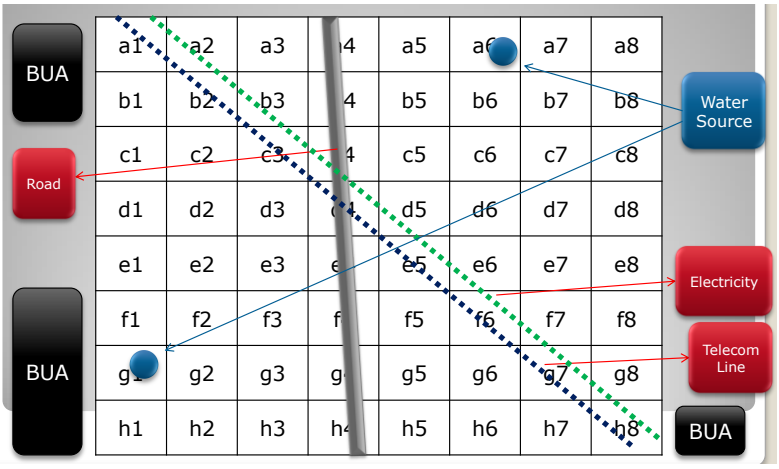

Fig. 6.3.1

Phase I

Each agent ranks the squares according to its individual preferences –

1. Combat Unit A
1.1. Squares fulfilling all 4 requirements: a4, a5, a6, b4, b5, b6, c4, c5, c6.
1.2. Squares fulfilling only top 3 requirements: a3, b3, c3, d3, d4, d5, d6, e3, e4, e5, e6, f3, f4, f5, f6, g3, g4, g5, g6, h3, h4, h5, h6.
1.3. Squares fulfilling only top 2 requirements: d1, d2, a7, a8, b7, b8, c7, c8, d7, d8, e7, e8.
1.4. Squares fulfilling only the top requirement: rest of the squares.
2. Combat Unit B
2.1. Squares fulfilling all 4 requirements: a4, a5, a6, b4, b5, b6, c4, c5, c6.
2.2. Squares fulfilling only top 3 requirements: a3, b3, c3, d3, d4, d5, d6, e3, e4, e5, e6, f3, f4, f5, f6, g3, g4, g5, g6, h3, h4, h5, h6.
2.3. Squares fulfilling only top 2 requirements: d1, d2, a7, a8, b7, b8, c7, c8, d7, d8, e7, e8.
2.4. Squares fulfilling only the top requirement: rest of the squares.
3. Combat Unit C
3.1. Squares fulfilling all 3 requirements: none.
3.2. Squares fulfilling only top 2 requirements: a5, a6, a7, a8, b5, b6, b7, f1, f2, g1, g2, h1, h2.
3.3. Squares fulfilling only the top requirements: rest of the squares.
4. Headquarters
4.1. Squares fulfilling all 4 requirements: a3, b3, b4, c3, c4, c5, d3, d4, d5, e3, e4, e5, f4, f5, g5.
4.2. Squares fulfilling only top 3 requirements: none.
4.3. Squares fulfilling only top 2 requirements: a1, a2, b1, b2, c1, c2, d2, d6, e6, e7, f6, f7, f8, g6, g7, g8, h6, h7, h8.
4.4. Squares fulfilling only the top requirement: rest of the squares.
5. Support units
5.1. Squares fulfilling all 4 requirements: a3, b3, b4, c3, c4, c5, d3, d4, d5, e3, e4, e5, f4, f5, g5.
5.2. Squares fulfilling only top 3 requirements: none.
5.3. Squares fulfilling only top 2 requirements: a1, a2, b1, b2, c1, c2, d2, d6, e6, e7, f6, f7, f8, g6, g7, g8, h6, h7, h8.
5.4. Squares fulfilling only the top requirement: rest of the squares.
6. Logistics
6.1. Squares fulfilling all 5 requirements: none.
6.2. Squares fulfilling only top 4 requirements: a3, b3, b4, c3, c4, c5, d3, d4, d5, e3, e4, e5, f4, f5, g5.
6.3. Squares fulfilling only top 3 requirements: none.
6.4. Squares fulfilling only top 2 requirements: a4, a5, b5, f3, g3, g4, h3, h4, h5.
6.5. Squares fulfilling only the top requirement: rest of the squares.

This is communicated to the Central Agent.

Next, the Central Agent refines and adjusts the lists to conform to global constraints-

1. Combat Unit A
1.1. a4, a5, a6, b4, b5, b6.
1.2. a3, b3.
1.3. a7, a8, b7, b8.
1.4. Squares remaining in front tier: a1, a2, b1, b2.

Not enough squares in 2nd ranking, so combining 2. and 3., we get the final ranking:

1.1. a4, a5, a6, b4, b5, b6.
1.2. a3, b3, a7, a8, b7, b8.
1.3. a1, a2, b1, b2.

2. Combat Unit B

2.1. a4, a5, a6, b4, b5, b6.

2.2. a3, b3.

2.3. a7, a8, b7, b8.

2.4. Squares remaining in front tier: a1, a2, b1, b2.

3. Combat Unit C

3.1. a5, a6, a7, a8, b5, b6, b7.

3.2. Squares remaining in front tier: a1, a2, a3, a4, a8, b1, b2, b3, b4, b8.

4. Headquarters

4.1. c3, c4, c5, d3, d4, d5, e3, e4, e5, f4, f5.

4.2. c1, c2, d2, d6, e6, e7, f6, f7, f8.

4.3. Squares remaining in core region: c6, f3.

4.4. Squares remaining in middle tier (non-core): c7, c8, d1, d7, d8, e1, e2, e8, f1, f2.

5. Support units

5.1. c3, c4, c5, d3, d4, d5, e3, e4, e5, f4, f5.

5.2. c1, c2, d2, d6, e6, e7, f6, f7, f8.

5.3. Squares remaining in core region: c6, f3.

5.4. Squares remaining in middle tier (non-core): c7, c8, d1, d7, d8, e1, e2, e8, f1, f2.

6. Logistics

6.1. g5.

6.2. g3, g4, h3, h4, h5.

6.3. Squares remaining in rear tier: g1, g2, g6, g7, g8, h1, h2, h6, h7, h8.

Not enough squares in 1st ranking, so combining 1. and 2., so we get the final ranking:

6.1. g5, g3, g4, h3, h4, h5.

6.2. g1, g2, g6, g7, g8, h1, h2, h6, h7, h8.

Phase II:

Use the Generalised Serial Dictatorship with Ties Algorithm. The bundle rankings derived from agent preferences are -

1. Combat Unit A

1.1. *rank 0 bundles*: [a4, a5, a6, b4] [a4, a5, a6, b5] [a4, a5, a6, b6] [a4, a5, b4, b5] [a4, a5, b4, b6] [a4, a5, b5, b6] [a4, a6, b4, b5] [a4, a6, b4, b6] [a4, a6, b5, b6] [a4, b4, b5, b6] [a5, a6, b4, b5] [a5, a6, b4, b6] [a5, a6, b5, b6] [a5, b4, b5, b6] [a6, b4, b5, b6]

2. Combat Unit B

2.1. *rank 0 bundles*: [a4, a5] [a4, a6] [a4, b4] [a4, b5] [a4, b6] [a5, a6] [a5, b4] [a5, b5] [a5, b6] [a6, b4] [a6, b5] [a6, b6] [b4, b5] [b4, b6] [b5, b6]

2.2. *rank 1 bundles*: [a4, a3] [a4, b3] [a5, a3] [a5, b3] [a6, a3] [a6, b3] [b4, a3] [b4, b3] [b5, a3] [b5, b3] [b6, a3] [b6, b3]

2.3. *rank 2 bundles*: [[a3, b3]

2.4. *rank 3 bundles*: [a3, a7] [a3, a8] [a3, b7] [a3, b8] [b3, a7] [b3, a8] [b3, b7] [b3, b8]

2.5. *rank 4 bundles*: [a7, a8] [a7, b7] [a7, b8] [a8, b7] [a8, b8] [b7, b8]

2.6. *rank 5 bundles*: [a7, a1] [a7, a2] [a7, b1] [a7, b2] [a8, a1] [a8, a2] [a8, b1] [a8, b2] [b7, a1] [b7, a2] [b7, b1] [b7, b2] [b8, a1] [b8, a2] [b8, b1] [b8, b2]

2.7. *rank 6 bundles*: [a1, a2] [a1, b1] [a1, b2] [a2, b1] [a2, b2] [b1, b2]

3. Combat Unit C

3.1. *rank 0 bundles*: [a5] [a6] [a7] [a8] [b5] [b6] [b7]

3.2. *rank 1 bundles*: [a1] [a2] [a3] [a4] [b1] [b2] [b3] [b4] [b8]

4. Headquarters

4.1. *rank 0 bundles*: [c3, c4] [c3, c5] [c3, d3] [c3, d4] [c3, d5] [c3, e3] [c3, e4] [c3, e5] [c3, f4] [c3, f5] [c4, c5] [c4, d3] [c4, d4] [c4, d5] [c4, e3] [c4, e4] [c4, e5] [c4, f4] [c4, f5] [c5, d3] [c5, d4] [c5, d5] [c5, e3] [c5, e4]

[c5, e5] [c5, f4] [c5, f5] [d3, d4] [d3, d5] [d3, e3] [d3, e4] [d3, e5] [d3, f4] [d3, f5] [d4, d5] [d4, e3] [d4, e4] [d4, e5] [d4, f4] [d4, f5] [d5, e3] [d5, e4] [d5, e5] [d5, f4] [d5, f5] [e3, e4] [e3, e5] [e3, f4] [e3, f5] [e4, e5] [e4, f4] [e4, f5] [e5, f4] [e5, f5] [f4, f5]

5. Support Units

5.1. *rank 0 bundles*: [c3, c4] [c3, c5] [c3, d3] [c3, d4] [c3, d5] [c3, e3] [c3, e4] [c3, e5] [c3, f4] [c3, f5] [c4, c5] [c4, d3] [c4, d4] [c4, d5] [c4, e3] [c4, e4] [c4, e5] [c4, f4] [c4, f5] [c5, d3] [c5, d4] [c5, d5] [c5, e3] [c5, e4] [c5, e5] [c5, f4] [c5, f5] [d3, d4] [d3, d5] [d3, e3] [d3, e4] [d3, e5] [d3, f4] [d3, f5] [d4, d5] [d4, e3] [d4, e4] [d4, e5] [d4, f4] [d4, f5] [d5, e3] [d5, e4] [d5, e5] [d5, f4] [d5, f5] [e3, e4] [e3, e5] [e3, f4] [e3, f5] [e4, e5] [e4, f4] [e4, f5] [e5, f4] [e5, f5] [f4, f5]

5.2. *rank 1 bundles*: [c3, c1] [c3, c2] [c3, d2] [c3, d6] [c3, e6] [c3, e7] [c3, f6] [c3, f7] [c3, f8] [c4, c1] [c4, c2] [c4, d2] [c4, d6] [c4, e6] [c4, e7] [c4, f6] [c4, f7] [c4, f8] [c5, c1] [c5, c2] [c5, d2] [c5, d6] [c5, e6] [c5, e7] [c5, f6] [c5, f7] [c5, f8] [d3, c1] [d3, c2] [d3, d2] [d3, d6] [d3, e6] [d3, e7] [d3, f6] [d3, f7] [d3, f8] [d4, c1] [d4, c2] [d4, d2] [d4, d6] [d4, e6] [d4, e7] [d4, f6] [d4, f7] [d4, f8] [d5, c1] [d5, c2] [d5, d2] [d5, d6] [d5, e6] [d5, e7] [d5, f6] [d5, f7] [d5, f8] [e3, c1] [e3, c2] [e3, d2] [e3, d6] [e3, e6] [e3, e7] [e3, f6] [e3, f7] [e3, f8] [e4, c1] [e4, c2] [e4, d2] [e4, d6] [e4, e6] [e4, e7] [e4, f6] [e4, f7] [e4, f8] [e5, c1] [e5, c2] [e5, d2] [e5, d6] [e5, e6] [e5, e7] [e5, f6] [e5, f7] [e5, f8] [f4, c1] [f4, c2] [f4, d2] [f4, d6] [f4, e6] [f4, e7] [f4, f6] [f4, f7] [f4, f8] [f5, c1] [f5, c2] [f5, d2] [f5, d6] [f5, e6] [f5, e7] [f5, f6] [f5, f7] [f5, f8]

5.3. *rank 2 bundles*: [c1, c2] [c1, d2] [c1, d6] [c1, e6] [c1, e7] [c1, f6] [c1, f7] [c1, f8] [c2, d2] [c2, d6] [c2, e6] [c2, e7] [c2, f6] [c2, f7] [c2, f8] [d2, d6] [d2, e6] [d2, e7] [d2, f6] [d2, f7] [d2, f8] [d6, e6] [d6, e7] [d6, f6] [d6, f7] [d6, f8] [e6, e7] [e6, f6] [e6, f7] [e6, f8] [e7, f6] [e7, f7] [e7, f8] [f6, f7] [f6, f8] [f7, f8]

5.4. *rank 3 bundles*: [c1, c6] [c1, f3] [c2, c6] [c2, f3] [d2, c6] [d2, f3] [d6, c6] [d6, f3] [e6, c6] [e6, f3] [e7, c6] [e7, f3] [f6, c6] [f6, f3] [f7, c6] [f7, f3] [f8, c6] [f8, f3]

5.5. *rank 4 bundles*: [c6, f3]

5.6. *rank 5 bundles*: [c6, c7] [c6, c8] [c6, d1] [c6, d7] [c6, d8] [c6, e1] [c6, e2] [c6, e8] [c6, f1] [c6, f2] [f3, c7] [f3, c8] [f3, d1] [f3, d7] [f3, d8] [f3, e1] [f3, e2] [f3, e8] [f3, f1] [f3, f2]

5.7. *rank 6 bundles*: [c7, c8] [c7, d1] [c7, d7] [c7, d8] [c7, e1] [c7, e2] [c7, e8] [c7, f1] [c7, f2] [c8, d1] [c8, d7] [c8, d8] [c8, e1] [c8, e2] [c8, e8] [c8, f1] [c8, f2] [d1, d7] [d1, d8] [d1, e1] [d1, e2] [d1, e8] [d1, f1] [d1, f2] [d7, d8] [d7, e1] [d7, e2] [d7, e8] [d7, f1] [d7, f2] [d8, e1] [d8, e2] [d8, e8] [d8, f1] [d8, f2] [e1, e2] [e1, e8] [e1, f1] [e1, f2] [e2, e8] [e2, f1] [e2, f2] [e8, f1] [e8, f2] [f1, f2]

6. Logistics

6.1. *rank 0 bundles*: [g5, g3] [g5, g4] [g5, h3] [g5, h4] [g5, h5] [g3, g4] [g3, h3] [g3, h4] [g3, h5] [g4, h3] [g4, h4] [g4, h5] [h3, h4] [h3, h5] [h4, h5]

6.2. *rank 1 bundles*: [g5, g1] [g5, g2] [g5, g6] [g5, g7] [g5, g8] [g5, h1] [g5, h2] [g5, h6] [g5, h7] [g5, h8] [g3, g1] [g3, g2] [g3, g6] [g3, g7] [g3, g8] [g3, h1] [g3, h2] [g3, h6] [g3, h7] [g3, h8] [g4, g1] [g4, g2] [g4, g6] [g4, g7] [g4, g8] [g4, h1] [g4, h2] [g4, h6] [g4, h7] [g4, h8] [h3, g1] [h3, g2] [h3, g6] [h3, g7] [h3, g8] [h3, h1] [h3, h2] [h3, h6] [h3, h7] [h3, h8] [h4, g1] [h4, g2] [h4, g6] [h4, g7] [h4, g8] [h4, h1] [h4, h2] [h4, h6] [h4, h7] [h4, h8] [h5, g1] [h5, g2] [h5, g6] [h5, g7] [h5, g8] [h5, h1] [h5, h2] [h5, h6] [h5, h7] [h5, h8]

6.3. *rank 2 bundles*: [g1, g2] [g1, g6] [g1, g7] [g1, g8] [g1, h1] [g1, h2] [g1, h6] [g1, h7] [g1, h8] [g2, g6] [g2, g7] [g2, g8] [g2, h1] [g2, h2] [g2, h6] [g2, h7] [g2, h8] [g6, g7] [g6, g8] [g6, h1] [g6, h2] [g6, h6] [g6, h7] [g6, h8] [g7, g8] [g7, h1] [g7, h2] [g7, h6] [g7, h7] [g7, h8] [g8, h1] [g8, h2] [g8, h6] [g8, h7] [g8, h8] [h1, h2] [h1, h6] [h1, h7] [h1, h8] [h2, h6] [h2, h7] [h2, h8] [h6, h7] [h6, h8] [h7, h8]

This is a partial ranking as it does not rank every possible combination of squares.

We obtain the highest-ranked bundles for each agent which are disjoint with at least one of highest-ranked bundles of every other agent. The solution will be a feasible allocation consisting of a combination of the following bundles, one for each agent:

1. Combat Unit A– [[a4, a5, a6, b4], [a4, a5, a6, b5], [a4, a5, a6, b6], [a4, a5, b4, b5], [a4, a5, b4, b6], [a4, a5, b5, b6], [a4, a6, b4, b5], [a4, a6, b4, b6], [a4, a6, b5, b6], [a4, b4, b5, b6], [a5, a6, b4, b5], [a5, a6, b4, b6], [a5, a6, b5, b6], [a5, b4, b5, b6], [a6, b4, b5, b6]]
2. Combat Unit B– [[a4, a5], [a4, a6], [a4, b4], [a4, b5], [a4, b6], [a5, a6], [a5, b4], [a5, b5], [a5, b6], [a6, b4], [a6, b5], [a6, b6], [b4, b5], [b4, b6], [b5, b6]]
3. Comabt Unit C 1 – [[a5], [a6], [a7], [a8], [b5], [b6], [b7]]
4. Combat Unit C 2 – [[a5], [a6], [a7], [a8], [b5], [b6], [b7]]
5. Combat Unit C 3 – [[a5], [a6], [a7], [a8], [b5], [b6], [b7]]

6. Headquarters – [[c3, c4], [c3, c5], [c3, d3], [c3, d4], [c3, d5], [c3, e3], [c3, e4], [c3, e5], [c3, f4], [c3, f5], [c4, c5], [c4, d3], [c4, d4], [c4, d5], [c4, e3], [c4, e4], [c4, e5], [c4, f4], [c4, f5], [c5, d3], [c5, d4], [c5, d5], [c5, e3], [c5, e4], [c5, e5], [c5, f4], [c5, f5], [d3, d4], [d3, d5], [d3, e3], [d3, e4], [d3, e5], [d3, f4], [d3, f5], [d4, d5], [d4, e3], [d4, e4], [d4, e5], [d4, f4], [d4, f5], [d5, e3], [d5, e4], [d5, e5], [d5, f4], [d5, f5], [e3, e4], [e3, e5], [e3, f4], [e3, f5], [e4, e5], [e4, f4], [e4, f5], [e5, f4], [e5, f5], [f4, f5]]
7. Support Unit 1 – [[c3, c4], [c3, c5], [c3, d3], [c3, d4], [c3, d5], [c3, e3], [c3, e4], [c3, e5], [c3, f4], [c3, f5], [c4, c5], [c4, d3], [c4, d4], [c4, d5], [c4, e3], [c4, e4], [c4, e5], [c4, f4], [c4, f5], [c5, d3], [c5, d4], [c5, d5], [c5, e3], [c5, e4], [c5, e5], [c5, f4], [c5, f5], [d3, d4], [d3, d5], [d3, e3], [d3, e4], [d3, e5], [d3, f4], [d3, f5], [d4, d5], [d4, e3], [d4, e4], [d4, e5], [d4, f4], [d4, f5], [d5, e3], [d5, e4], [d5, e5], [d5, f4], [d5, f5], [e3, e4], [e3, e5], [e3, f4], [e3, f5], [e4, e5], [e4, f4], [e4, f5], [e5, f4], [e5, f5], [f4, f5]]
8. Support Unit 2 – [[c3, c4], [c3, c5], [c3, d3], [c3, d4], [c3, d5], [c3, e3], [c3, e4], [c3, e5], [c3, f4], [c3, f5], [c4, c5], [c4, d3], [c4, d4], [c4, d5], [c4, e3], [c4, e4], [c4, e5], [c4, f4], [c4, f5], [c5, d3], [c5, d4], [c5, d5], [c5, e3], [c5, e4], [c5, e5], [c5, f4], [c5, f5], [d3, d4], [d3, d5], [d3, e3], [d3, e4], [d3, e5], [d3, f4], [d3, f5], [d4, d5], [d4, e3], [d4, e4], [d4, e5], [d4, f4], [d4, f5], [d5, e3], [d5, e4], [d5, e5], [d5, f4], [d5, f5], [e3, e4], [e3, e5], [e3, f4], [e3, f5], [e4, e5], [e4, f4], [e4, f5], [e5, f4], [e5, f5], [f4, f5]]
9. Logistics Unit 1 – [[g5, g3], [g5, g4], [g5, h3], [g5, h4], [g5, h5], [g3, g4], [g3, h3], [g3, h4], [g3, h5], [g4, h3], [g4, h4], [g4, h5], [h3, h4], [h3, h5], [h4, h5]]
10. Logistics Unit 2 – [[g5, g3], [g5, g4], [g5, h3], [g5, h4], [g5, h5], [g3, g4], [g3, h3], [g3, h4], [g3, h5], [g4, h3], [g4, h4], [g4, h5], [h3, h4], [h3, h5], [h4, h5]]

Phase III:

From the list of feasible allocations, the Central Agent picks an allocation that conforms to interactive logic -

1) Combat Unit A– a4, a5, b4, b5.
2) Combat Unit B– a3, b3.
3) Combat Unit C 1 – a7.
4) Combat Unit C 2 – b6.
5) Combat Unit C 3 – b7.
6) Headquarters – d4, d5.
7) Support Unit 1 – d3, e3.
8) Support Unit 2 – e4, e5.
9) Logistics Unit 1 – g3, h3.
10) Logistics Unit 2 – g5, h5.

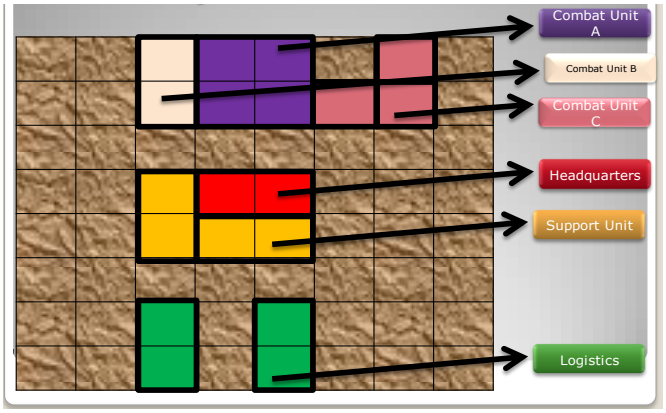

Fig. 6.3.2

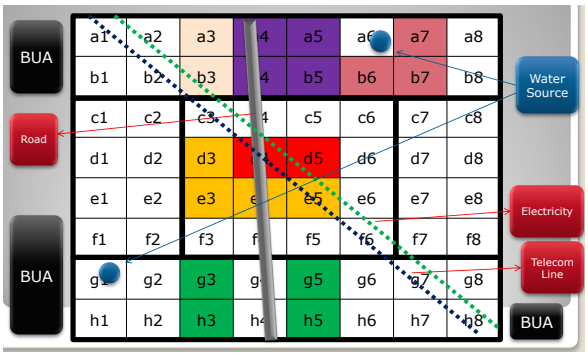

Fig. 6.3.3

Fig. 6.3.2 and Fig. 6.3.3 illustrate the final allocation. Each agent has received its most preferred bundle. Using this 3 phases procedure, we can get the best allocation which conforms to the constraints of the problem. Since the global logic is dealt with first, there is no cost incurred in relocating agents if their desired squares are not in their deployable region; also since we want to fulfill as many individual preferences of the military units as possible, we deal with their requirements next. Use of the GSDwT algorithm ensures that higher priority agents get their most preferred squares, and at the same time ensures a pareto-optimal allocation. By the nature of the algorithm, the allocation is elitist.

Conclusion

In this paper, we have studied the allocation problem where both indifference in ranking and requirement of more than one objects is allowed. The motivation to study this problem comes from a problem called the site selection problem, arising often in the military domain. We have modeled this problem as a Multi Agent Resource Allocation problem, and studied in details as a case study with the help of a toy example. We have proposed a novel mechanism, which we call Generalized Serial Dictatorship with Ties (GSDwT), where both indifference in ranking and requirement of more than one objects is allowed. We have, further, proved that GSDwT is both pareto-efficient and strategy-proof.

Acknowledgment

The authors would like to thank Director, CAIR, for the support provided in carrying this research work. The first author would also like to thank and express his gratitude to Prof. Y. Narahari, CSA Dept. IISC, for his encouragement and constant support.

References

[1] John Willium Hatfield, *Strategy-proof, efficient, and non bossy quota allocations*, Social Choice and Welfare, Vol. 33, 505-515, 2009.

[2] Y. Chevaleyre, P. E. Dunne, U. Endriss , J. Lang, M. Lemaitre, N. Maudet, *Issues in Multi-Agent Resource Allocation*, Informatica, Vol. 30(1), 3-31, 2006.

[3] Paula Jaramillo, Vickram Manjunath, *The Difference Indifference Makes in Strategy-Proof Allocation of Objects*, Working paper, 2011.

[4] M. Lemaitre, G. Verfallie, H. fargier, J. lang, N. Bataille and J. M. Lachiver, *Equitable allocation of earth observing satellites resources*, In Proc. 5th ONERADLR Aerospace Symposium (ODAS-2003).

[5] Ehlers Lars, *Coalitional Strategy-proof House Allocation*, Journal of Economic Theory, Vol. 105(2), 298-317, 2002.

[6] Hylland Aanund and Richard Zeckhauser, *The Efficient allocation of Individuals to Positions*, Journal of Political Economy, Vol. 87(2), 293-314, 1979.

[7] Shapley Lloyd and Herbert Scarf, *On cores and indivisibility* , Journal of Mathematical Economics, Vol. 1, 23-37, 1974.

[8] Wako Jun, " *Some properties of weak domination in an exchange market with indivisible goods*", Economic Studies Quarterly, Vol. 42, 303-314, 1991.

[9] Moon K. Chetry and Dipti Deodhare, " *Multi Agent Resource Allocation - Serial Dictatorship with Ties*", Proceedings of IEEE International Conference on Computational Intelligence and Computing Research, (IEEE-ICICCIC 2012)}, pp 149-156, 18-20 Dec, 2012, Coimbatore, India.