Chapter 3

Partitioning Search Space

3.1 Partitioning Search Space

In distributed search algorithms, the most common principle is to divide search space equally and distribute the calculation among computational unit equally. During the search, consistent communication among agents regarding the progress of the search is required. However, we consider distributed domains where communication among agent may be lost easily due to various factors. It is a challenge to derive an algorithm which allows for independent search among agents and, more importantly, the search can be carried out and the final result are close to optimality. In the following, we shall discuss the reasoning for dividing the search space and describe how it should be carried out.

3.1.1 Partition Objectives

Distributing workload among agents has been an interesting area of research in multiagent systems. Rahwan et.al [12] proposed an algorithm for computing coalition value among cooperative agents. The algorithm does more than just divides the search space, the number of coalitions and their values, yet to be computed, equally. It also takes into account the computational power of each agent and ensures that each agent will finish the calculation about the same time.

However, in generating coalition structure, dividing the coalitions crisply among agents in order to generate coalition structure seems hard to achieve. Michalaks et.al [9], proposing a distributed algorithm for generating coalition structure, admitted that doing so cannot guarantee that all coalition structures would be generated. Even worst, some kinds of divisions can leave some agents with impossibility to generate even a single coalition structure.

Consider the case which agents are given coalitions whose sizes are all larger than half of n but less than n. These agents simply cannot generate any coalition structure because we cannot put any pair of the coalitions without violating the mutual exclusive condition. Therefore, Michalaks et. al [9] have to store redundant coalitions among agents. In order to generate coalition structures, Michalaks et. al [9] were able to divide the search space among agents lexicographically. In addition, the algorithm requires considerable communication among agents during the search. We argue that this could be a set back for the performance of algorithm because disconnection in communication can lead to the failure of the algorithm.

In this work, the notion of distributing work load among agents is different from other works where the search space is divided by the number of coalition structures. The idea is to distribute the possibility of finding CS^* among all agents equally. This is due to the nature of the search algorithm we are going to use that it is value-oriented, not lexicographic. (Even though both principles apply branch and bound technique to prune the search space, the value-oriented algorithm converges to optimality more quickly.) In our algorithm, the data about all coalitions and their values are kept in all agents. It is also mentioned in Mikalak et. al. [9] that crisply dividing the search space for lexicographic-based algorithm in distributed environment is hard to achieve. There are a lot of redundant coalitions kept among agents.

Here, we setup two objective for designing a distributed algorithm for searching for optimal coalition structures:

- minimize communication among agents, and
- distribute the chance for finding optimal coalition structures among agents equally..

A naive way to distributedly calculate the optimal coalition structure without communication at all is to give every agent the whole task. However, that does not utilize the collective computation power. Since considerable communication can lead to the failure of distributedly generating coalition structures, we opt to minimize the communication. This lead to another dimension of challenges, how could we divide the search space in order reach optimality quickly and reduce the redundant search among agent.

3.1.2 Generate Coalition Structures as Generating Trees

According to Sombattheera et.al. [27], the first coalition, generating coalition structures is like building trees. We denote by CS_ι the ι-th coalition structure, where $1 \leq \iota \leq B$. We denote by $CS^\imath$ the $\imath$-th best coalition structure found so far, where $1 \leq \imath \leq^*$. We denote by S_j^C the j-th coalition of cardinality C. Similarly, we denote by S_j^{CS} the j-th coalition of coalition structure CS. When we generate CS^1, the first coalition structure, the root of the tree, S^{1,CS_1}, is the first coalition of the structure. At the second level, there will be ${}^{n-|S^1,CS_1|}C_{q_1,1\leq q_1<|S^1,CS_1|}$ coalitions. Consequently, there will be

$$^{n-|S^1,CS_1|-|S^2,CS_1|}C_{q_2,1\leq q_2<|S^1,CS_1|+|S^1,CS_1|}$$

$$^{n-|S^1,CS_1|-|S^2,CS_1|....-|s^l,CS_1|}C_{q_l,1\leq q_l<|S^1,CS_1|+|S^1,CS_1|...+|S^{l,CS_1}|}$$

coalitions for the third up to l level respectively.

If we use the algorithm to generate CSs lexicographically, starting from $S^{1,CS_1} = \{1\}$, this root will have more branches than any other coalitions. As the algorithm proceeds, $\{2\}$ will become the root at some point. However, the number of branches it can generate will be less than that of $\{1\}$ because $\{1\}$, as the root, has generated all coalition structures it has, including $\{2\}$ as a member of those coalition structures. Therefore, we cannot include $\{1\}$ in coalition structures, whose root is $\{2\}$ again. Otherwise, the number of coalition structures will be redundantly generated, which is wrong. The algorithm [27] prohibits this by generating coalition structures from higher coalitions (ranked in cardinalities) downward. In general, lower coalitions will have less branches that that of higher ones. This is always the case in each cardinality. However, this phenomenon does not exist when we use the algorithm to search. This is due to the fact that coalitions are sorted by their values and, more importantly, the root of each tree can be chosen from any cardinality. This means we may have trees of any size as the algorithm proceeds. In order to analyze the situation more systematically, we can do it according to the eight distributions proposed in [27].

3.1.3 Analyzing Search Space

Data Distribution

In DEC, the small coalitions tend to be the root of the tree first and the tree generated tend to be the large ones, both in terms of width and depth. The chance of finding optimal coalition structures seems to be higher with these trees. There more are chances that small coalitions would mostly comprise optimal coalition structures.

In INC, most coalitions of larger cardinalities tend to have higher agent contribution values. That means value-oriented algorithms would have root coalitions mostly from larger cardinalities and tends to generate small trees

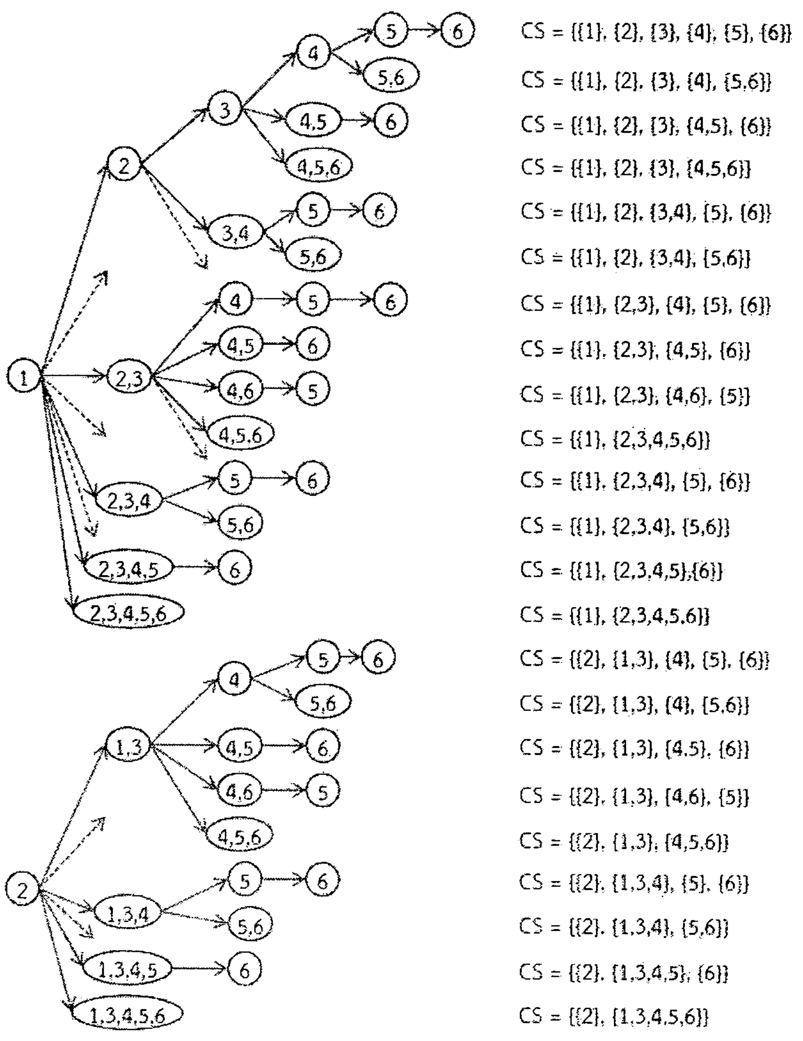

CS = {{1}, {2}, {3}, {4}, {5}, {6}}
CS = {{1}, {2}, {3}, {4}, {5,6}}
CS = {{1}, {2}, {3}, {4,5}, {6}}
CS = {{1}, {2}, {3}, {4,5,6}}
CS = {{1}, {2}, {3,4}, {5}, {6}}
CS = {{1}, {2}, {3,4}, {5,6}}
CS = {{1}, {2,3}, {4}, {5}, {6}}
CS = {{1}, {2,3}, {4,5}, {6}}
CS = {{1}, {2,3}, {4,6}, {5}}
CS = {{1}, {2,3,4,5,6}}
CS = {{1}, {2,3,4}, {5}, {6}}
CS = {{1}, {2,3,4}, {5,6}}
CS = {{1}, {2,3,4,5},{6}}
CS = {{1}, {2,3,4,5,6}}
CS = {{2}, {1,3}, {4}, {5}, {6}}
CS = {{2}, {1,3}, {4}, {5,6}}
CS = {{2}, {1,3}, {4,5}, {6}}
CS = {{2}, {1,3}, {4,6}, {5}}
CS = {{2}, {1,3}, {4,5,6}}
CS = {{2}, {1,3,4}, {5}, {6}}
CS = {{2}, {1,3,4}, {5,6}}
CS = {{2}, {1,3,4,5}, {6}}
CS = {{2}, {1,3,4,5,6}}

Figure 3.1: **Tree-like Coalition Structure Generation** Generating coalition structure is like generating tree. The root of the the tree is the coalition at level 1.

(coalition structures). As shown in the results of AAMAS-08, optimal coalition structures seem to come from these coalition structures. In other words, the solution will be found quickly just by generating trees of the roots sitting higher in the ranking.

In CCD, similar to DEC, small coalitions tend to be the root of the tree first. More interestingly, larger coalitions would more likely comprise optimal coalition structures because their agent contribution values are as good as that of middle-sized coalitions but the larger ones will be chosen first. Therefore, optimal coalition structures tend to be small and could be found quickly.

In CVD where small and large coalitions tend to have lower values than that of the medium ones, medium coalitions tend to have higher agent contribution values and will be used as the root to generate the tree first. These coalitions seem to comprise optimal coalition structures among themselves. More interestingly, these medium-sized coalition constitute more coalition structures than the small and large ones. Therefore, at later stages, small coalitions, which ranked higher than the large ones due to higher agent contribution values, tend to be the root of the trees and comprise coalition structures with large coalitions. With this CVD, higher roots will generate more branches than the lower ones.

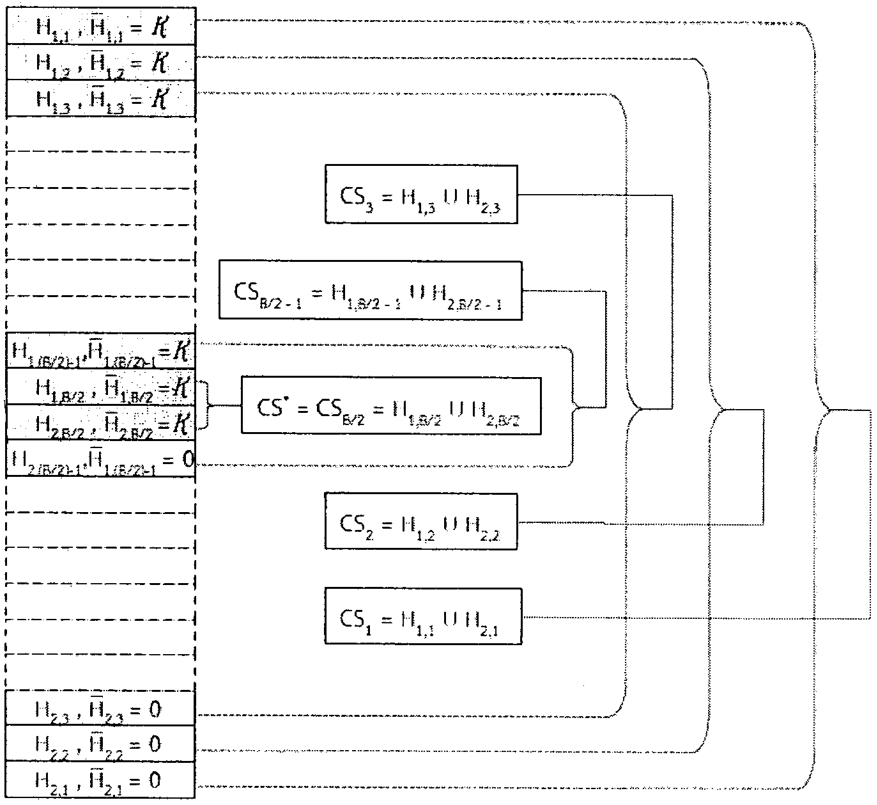
In the case of superadditive environment, it can be shown that the search will be finished very quickly because the grand coalition is the optimal coalition structure and there will be no other coalitions which can propose a chance to improve the solution. It is similar in the case of subadditive environment. Since larger coalitions cannot have higher values than the small ones, there is no chance for other coalitions to comprise optimal coalition structures except the singleton ones.

In NMD and UNI, we cannot really analyze the generation of coalition structures because we do not have any useful information about the relation between the size and value of coalitions.

Given the above information, it is necessary that the search space partition has to propose a fair distribution of chance of finding the optimal coalition structure equally among agents.

3.1.4 Worst Case Scenario

Given a non-singleton coalition structure, CS, where $|CS| > 1$ (one with at least two coalitions), we can split it into two halves, $H_1 = \bigcup_{S_i \in CS} S_i$, where $i < |CS|$ and $|H_1| \geq \frac{|CS|}{2}$, and $H_2 = CS \setminus H_1$, where $|H_1| - |H_2|$ is minimal. Let $V(H_1) = \sum_{S_i \in CS} V(S_i)$ be the value of H_1 and $V(H_2) = \sum_{S_j \in CS} V(S_j)$



$H_{1,1}, \bar{H}_{1,1} = \bar{K}$
$H_{1,2}, \bar{H}_{1,2} = \bar{K}$
$H_{1,3}, \bar{H}_{1,3} = \bar{K}$
$CS_3 = H_{1,3} \cup H_{2,3}$
$CS_{B/2-1} = H_{1,B/2-1} \cup H_{2,B/2-1}$
$H_{1,(B/2)-1}, \bar{H}_{1,(B/2)-1} = \bar{K}$
$H_{1,B/2}, \bar{H}_{1,B/2} = \bar{K}$
$CS^* = CS_{B/2} = H_{1,B/2} \cup H_{2,B/2}$
$H_{2,B/2}, \bar{H}_{2,B/2} = \bar{K}$
$H_{2,(B/2)-1}, \bar{H}_{1,(B/2)-1} = 0$
$CS_2 = H_{1,2} \cup H_{2,2}$
$CS_1 = H_{1,1} \cup H_{2,1}$
$H_{2,3}, \bar{H}_{2,3} = 0$
$H_{2,2}, \bar{H}_{2,2} = 0$
$H_{2,1}, \bar{H}_{2,1} = 0$

Figure 3.2: **Bi-partitioned Search SpaceIn** order to visualize the worst-case scenario, each coalition structure can be divided into two halves and be sorted by their values. The worst-case scenario is when the optimal pair lie in the middle of the rank.

be the value of H_2. We have $|\bar{H}_1| = \frac{V(H_1)}{|H_1|}$ and $|\bar{H}_2| = \frac{V(H_2)}{|H_2|}$ the average agent contribution of H_1 and H_2 respectively.

If we rank all of these two halves of all coalition structures by average agent contribution value and their sizes in ascending order, the worst case scenario for applying the value-oriented search algorithm is when all of H_1s have the same agent contribution value $\bar{H}_1 > 0$ and their respective H_2s have the same agent contribution value $\bar{H}_2 = 0$, except H_2 of the last H_1 with the same agent contribution value as its H_1's. In this scenario, the algorithm will go through all H_1 before the last one because their $\bar{CS}$ will always less than $\bar{H}_1$. This is due to the fact that the the size of CS is larger than the size of H_1 whereas the value of CS remains unchanged.

3.2 Partitioning Search space

3.2.1 Distributing Chance Equally

In general, the value-oriented algorithm such as [27] has some advantages and disadvantages. When used for generating coalition structures with input, coalitions, sorted lexicographically, the algorithm seems to waste time for locating the next candidate in each cardinality. However, when used for searching for optimal coalition structure, one of the obvious advantages is that it converges to optimality relatively quickly. This can be achieved due to the monotonicity brought in by sorting input, coalitions, by value in descending order. Unfortunately, the algorithm appears to terminate late due to the expense of searching on spaces which do not lead to a better solution. Another disadvantage of the algorithm is that locating the next candidate for each cardinality can be expensive if we want to generate all coalition structures because the algorithm may have to go through invalid coalitions (because their members are part of the coalitions already used in the building coalition structures). Furthermore, as a root coalition of the building coalition structure, the coalitions which sit higher in a cardinality will have more branches and leaves, or produces more coalition structures, than those sitting lower in the same cardinality.

In the context of distributing search space equally among agents, this is a controversial issue because coalitions sitting higher appears to have more search space as discuss above. However, since the coalitions are sorted by value and with the application of branch and bound that the algorithm will proceed in any new branch when there is a chance for improving the solution, there are chances where the branches of higher coalitions can be pruned massively. To be exact, we do not know in advance what data distribution (of the input, with regards to the size of coalitions and their respective values), will be. In stead of distributing the search space physically equally, the work propose a different notion of search by distributing the chance of finding optimal coalition structure equally.

3.2.2 Number of Coalition Structures of a Coalition

In general, given a coalition S, the number of coalition structures to which this coalition belongs is

$$B(S) = B^{n-|S|}.$$

Using the value-oriented algorithm, this fact holds when it becomes the root before being a node in a graph of other roots. On the other hand, the number of coalition structures it generates as a root becomes smaller after it becomes a node of other roots. When using the value-oriented algorithm to generate coalition structures, starting from smaller nodes as roots, the sizes of the trees are larger than those generated from large coalition roots because $n - |S|$ tends to be smaller. In order to distributing chance for reaching optimality among agents equally, we simply have to create more search space portions from higher roots than that from lower roots.

On the other hand, when using value-oriented algorithm to search for optimal coalition structures, we cannot always guarantee that such behavior will hold, simply due to the fact that coalitions in each cardinality are sorted by their values. In distribution patterns such as RND or CNV, it is impossible to predict anything. Given this simple fact, we have to ensure that the distributing of search space portion has to yield the equality in chance of reaching optimality equally among agents, i.e. with minimal communication and redundant search space.

3.2.3 Nested Search Space

In order to achieve the aforementioned goals, we propose a novel principle for partitioning search space. In stead of partition the search space lexicographically, we rather partition it by values of coalitions. As discussed above that our algorithm will follow the value-oriented approach which works similarly to generating graph, partitioning search space can be considered as partitioning graph. Since we know that the number of nodes at a certain level of a given root depends on its value, i.e. if the value is high then the graph is more likely to be larger, we propose to partition the search space at a proper level, which would ensure that agents will have chances to reach optimality equally. In other words, every agent will be close to optimality eventhough it does not exist in the search space portion given to the agent. Figure 3.3 illustrates this principle. Since a given level of the graph of a root can be explored by multiple agents, we call it nested search space. The next question is how large the search portion should be?

3.2.4 Size of Search Space Portion

In order to achieve this goal, there are two key factors that we have to consider:

- How large is each search portion?, and
- Where does the nested space ends?

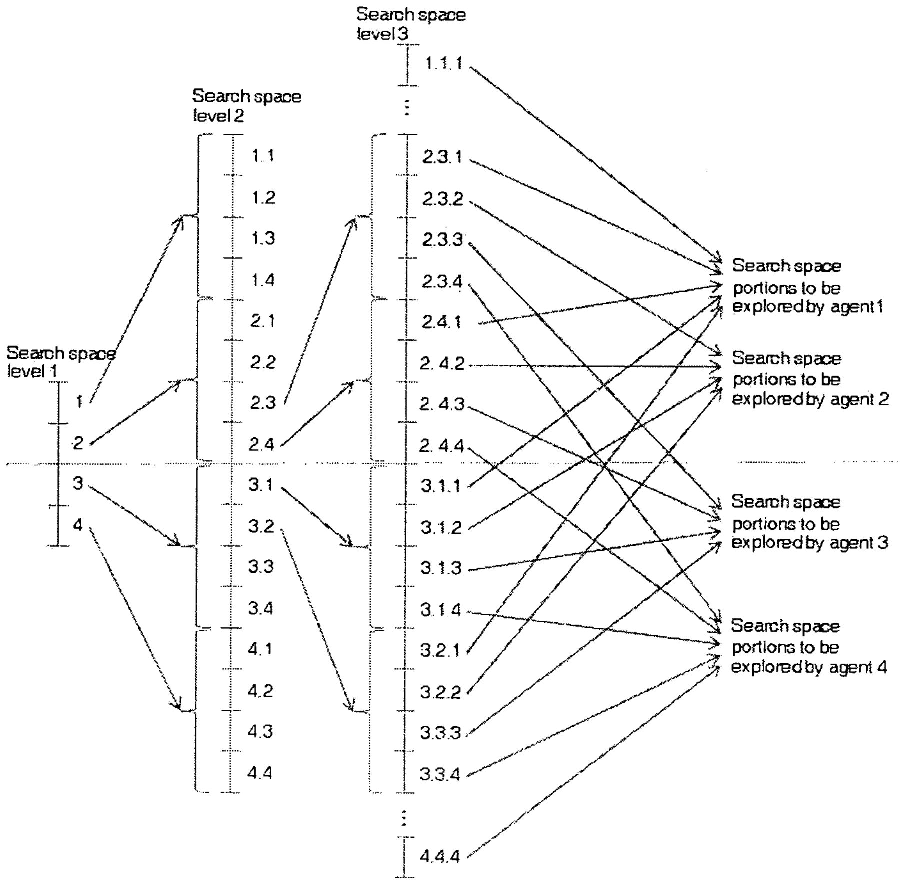


Figure 3.3: **Nested Partition Search Space**Down to a certain level of the graph of a given node, that part of the graph can be explored by multiple agents.

Note that the idea of distributing search space among agents in this work is meant to be a trade off between the closeness to the optimality and the size of each portion.

There are two concerns:

- Due to inconsistent communication network, each agent has to be able to carry out the search independently and finish the search as close as possible to optimality.
- The easiest way to achieve this is to allocate to each agent the whole search space. However, doing this is too naive and does not exploit the aggregated computational power of all the agents.

The challenge here is to balance these two at and appropriate point. Furthermore, it is important that each agent does not go into the illusive trap of the search when the solution has been found by some other agents but it is not aware of and keeps wasting time searching.

According to the idea presented in the above section, available entitled coalitions in each level has to be divided for each agent. Therefore, there will be multiple agents searching in their assigned portions at each level of a given root. Division of search space is done based on the available root coalitions.

Let the number of entitled coalitions be W. Let $K \in \mathbb{Z}^+$ be a positive number. Let $1 \leq k \leq K$ be another positive number. Let the distance between power of k,

$$Diff(k) = W - k^k$$

be the difference between k power of k and W, where $k^k < W$. The reference level l_k^* is the level in which the partition of search space takes place where

$$k^* = arg_{min} Diff(k).$$

Every time the root is changed, we will calculate for this k^*.

3.2.5 Locating Head and Tails of Search Space Portion

Coalitions with $\bar{S} > \bar{CS}^1$ are called entitled coalitions. Cardinalities which have entitled coalitions are called entitled cardinalities. Of course, the maximum number of entitled cardinalities, C^E is n. We refer to each of these entitled cardinality as the e-th cardinality, denoted by C^e where $1 \leq e \leq n$. In each C^e, there is S^f, which is the first coalition of C^e and there is S^l, which is the last entitled coalition of that C^e.

In each C^e, let w^e be the number of entitled coalitions. The number of all entitled coalitions is

$$W = \sum w^{eE}_{e=1}$$

Each of these entitled coalition will be the starting point of the searching coalition structure, S^1_{CS}. These starting points will be divided among agents. The original size of the search space portion for each agent is

$$\omega = \lfloor \frac{W}{n} \rfloor$$

Given this original distribution, there will be

$$\eta = W - (\omega \cdot n)$$

remaining coalitions. Note that this number may be zero but will always be less than n, $0 \leq \eta < n$. These coalitions will distributed equally to the first η agents. Therefore, the size of the search space for agent i is

$$\Omega^i = \begin{cases} \omega + 1 & \text{if } i \leq \eta \\ \omega & \text{if } i > \eta \end{cases}$$

The next thing we need to do is to locate the exact locations for the first and last coalitions in each search space portion. Since the number of entitled coalitions in each entitled cardinality varies, locating the first and last coalitions become slightly complex. Essentially, we need to find out, for each agent a_i, the exact cardinality, C_ϵ, and exact position in that cardinality of the first, $S_{i,\alpha}$, and last, $S_{i,\beta}$, coalitions of the search space of a_i.

In order to achieve this, the algorithm for locating $S_{i,\alpha}$ and $S_{i,\beta}$ needs extra information, just have to advance through all the entitled cardinalities one by one, while accumulating the number of entitled coalitions it has gone pass, and verify if the

Let the portion offset

$$\phi = \sum_{i=1}^{n} \Omega_{i-1}$$

be the distance (number of coalitions) from the starting point to the head of the portion.

The head of the portion

$$\rho_h = \phi + 1$$

is the position next to ϕ.

The tail of the portion

$$\rho_t = \phi + \Omega_i$$

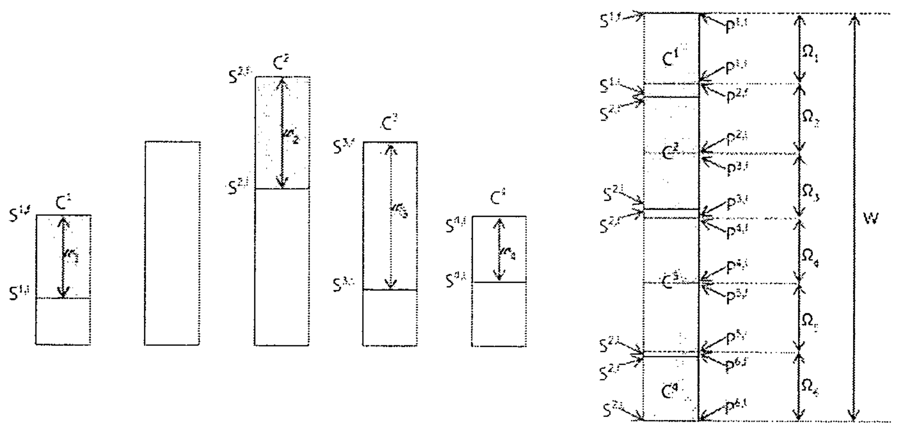


Figure 3.4: **Search Space** This Figure illustrates the heads and tails of search space portion.

is the position which is $\phi + \Omega_i$ away from the offset.

The cardinality offset

$$\theta = \sum_{i=1}^{E} |C_{i-1}|$$

is the distance from the first entitled coalition of the first entitled cardinality to the first entitled coalition of this entitled cardinality.

The relative cardinality head

$$\Theta_{h,i} = \theta_i + 1$$

is the first coalition of this entitled cardinality. Its position is located next to the cardinality offset.

The relative cardinality tail

$$\Theta_{t,i} = \theta_i + |C_i|$$

is the last coalition of this entitled cardinality. Its position is located at the position which is $|C_i|$ away from the offset.

In order to find where the beginning and ending coalitions of each portion, we refer to them as at which entitled cardinalities and which entitled of these coalitions. In order to fin the cardinality in which ρ_h falls, the following property must hold:

$$\Theta_{h,i} \leq \rho_h \leq \Theta_{l,i}.$$

Similarly, in order to find the cardinality in which ρ falls, the following property must hold:

$$\Theta_{l,i} \leq \rho_t \leq \Theta_{l,i}.$$

The head of search space portion of agent $1 \leq i \leq n$ is

$$P_h^i = S_\alpha^{C_\epsilon}$$

the ε-th coalition of the ϵ-th entitled cardinality.

The tail of search space portion of agent $1 \leq i \leq n$ is

$$P_t^i = S_\beta^{C_\varepsilon}$$

the ψ-th coalition of the φ-th entitled cardinality.

Each agent simply follows these principles to calculate its search space portions. Once achieve that, these portion can be inseasrted into a queue which will be further explored by the algorithm presented in the next chapter.