value M_β such that $M_\beta < c_{1,\beta}$, then:

- $M_k = M_k + 1 : k = \beta$
- $M_k = M_{k-1} + 1 : \beta < k \leq s$

This process should be repeated until all the coalitional values in $L_{s,i}$ are calculated. Note that after each agent calculates the values in its share, some values might remain uncalculated. This is because N_s might not be exactly divisible by the number of agents, and in this case, the agents' equal shares will not cover all the required values. In particular, the number of the remaining values would be:

$$N' = N_s - \sum_{j=1}^{n} N_{s,j} = N_s - n * \lfloor N_s/n \rfloor \qquad (3)$$

And the coalitions that need their values to be calculated would be: $c_{N_s - N' + i} : i \in 1, ..., N'$. Note that $N' < n$, and that each agent so far has calculated the same number of values. Therefore, in order to calculate these additional values and keep the distribution as fair as possible, each value should be calculated by a different agent; the agents should agree on a sequence A' which contains N' agents and in which each agent calculates one additional value. This can be done by maintaining a value α, initially set to 1, then for any list L_s, if there are additional values (i.e. if $N' > 0$) then A' would contain N' agents, starting from a_α. Then, each agent in A' calculates one additional value based on its position in A' (if we denote by a'_i the agent located at index i of A', then a'_i should calculate the value of coalition $c_{N_s - N' + i}$). Note that after these values are calculated, the agents need to update α so that for other lists, the next N' agents perform any additional calculations. This way, given any set S, the total number of values calculated by each agent will either be equal, or differ by only one value. Updating α is done as follows:

$$If (\alpha + N' < n)\ then\ \alpha = \alpha + N',\ else\ \alpha = \alpha + N' - n$$

And forming A' such that it contains N' agents, starting from a_α, is done as follows:

$$If (\alpha + N' - 1 < n)\ then\ A' = (a_\alpha, a_{\alpha+1}, ..., a_{\alpha+N'-1})$$
$$else\ A' = (a_\alpha, a_{\alpha+1}, ..., a_n, a_1, ..., a_{\alpha+N'-n})$$

For example if we have 6 agents, then from equation (3) we find that for L_2 we have $N' = 3$. Therefore, A' would be: (a_1, a_2, a_3) and α becomes 4. Then for L_3 we have $N' = 2$. Therefore, A' would be (a_4, a_5) and α becomes 6. Finally for L_4 we have $N' = 3$. Therefore, A' would be (a_6, a_1, a_2) and α becomes 3.

3. For $s = n - 1$, there exists n possible coalitions. Therefore, the calculations are distributed such that each agent calculates one value. This is done by having each agent a_i calculate the value of the coalition in which it is not a member, and every other agent is a member (i.e. $\{1, ..., i-1, i+1, ..., n\}$).

4. For $s = n$, there exists one coalition: $\{1, ..., n\}$. This is similar to the case where $N' = 1$. Therefore, the value of this coalition is calculated by a_α. In our example of 6 agents, this value would be calculated by a_3 and α becomes 4. Note that after all the values are calculated, the value of α remains 4 instead of being initialized to 1. This means that in order to form other coalitions, any additional calculations will start from a_4. By this, the average number of values calculated by each agent becomes equal.

Performance Evaluation

To evaluate the performance of the DCVC algorithm we compare it against the SK algorithm (see Figure 3).

Each agent a_i should perform the following:

- Put in P_i the set of potential coalitions that include up to k agents including a_i.
- While P_i is not empty do:
  - Contact an agent a_j that is a member of a potential coalition in P_i.
  - Commit to the calculation of the values of a subset S_{ij} of the commmon potential coalitions (i.e. a subset of the coalitions in P_i in which a_i and a_j are members).
  - Subtract S_{ij} from P_i. Add S_{ij} to your long-term commitment list.
  - For each agent a_k that has contacted you, subtract from P_i the set S_{ki} of the potential coalitions for which it had committed to calculate values.
  - Calculate the values for the coalitions you have committed to (Sij).
  - Repeat contacting other agents until $P_i = a_i$ (i.e., no more agents to contact).

Figure 3: The SK algorithm.

Specifically, we tested the performance of DCVC and SK for different numbers of agents[6]. The results presented in Table 2 are for the case where coalitions of any size are allowed to form (which means in our terms $S = \{1, ..., n\}$, and in SK's terms: $k = n$). Note that the results for SK were taken as an average of running a number of times; this is because their algorithm gives different results based on the order by which the agents contact each other.

The results show the differences in the performance of both algorithms in terms of:

1. Distribution time: The agents performed significantly faster when using DCVC. This is because in DCVC each agent can start processing its share of coalitions immediately, while in SK, each agent had to start with a list of all the coalitions in which it is a member, and then repeat the process of negotiating with other agents and committing to some coalitions and deleting others, until there were no more agents to contact.

2. Redundant calculations performed: Here by redundant we mean having the value of the same coalition calculated by more than one agent, while it was enough for one agent to calculate it. The table shows that using DCVC results in no redundant calculations (because each agent knows the precise bounding of the calculations it should perform, and these are disjoint). In contrast, SK results in an exponentially large number of redundant calculations; this is because each agent's commitment to a set of coalitions is done with very limited knowledge about the other agents' commitments. For example, agent a_i's knowledge about agent a_j's commitments is restricted to the set S_{ji} that a_j sends to a_i. This means that a_i is not aware of the coalitions to which a_j has committed by contacting other agents. This results in having the agents

[6]The PC on which we ran our simulations had a processor: Pentium(R)4 2.80 GHz, with 1GB of RAM.

	Time (in seconds)		Redundancy		Communication (in bytes)		Memory (in bytes)		Difference	
	DCVC	SK	DCVC	SK	DCVC	SK	DCVC	SK	DCVC	SK
16 agents	< 0.01	2.27	0	513452	0	735408	2	65536	1	8424
17 agents	< 0.01	6.11	0	1208715	0	2350481	3	196608	1	12886
18 agents	< 0.01	13.76	0	2583828	0	4974743	3	393216	1	26071
19 agents	< 0.01	32.29	0	5506420	0	10538129	3	786432	1	52890
20 agents	< 0.01	72.27	0	11659720	0	22152227	3	1572864	1	103484
21 agents	< 0.01	159.89	0	24605666	0	46512635	3	3145728	1	208931
22 agents	< 0.01	372.58	0	52170535	0	97957698	3	6291456	1	454812
23 agents	< 0.01	881.64	0	108933551	0	204911555	3	12582912	1	880428
24 agents	0.01	2280.43	0	210504067	0	429009502	3	25165824	1	2191528
25 agents	0.02	5298.52	0	477826101	0	1188779705	4	67108864	1	3043149

Table 2. Simulation results.

commit to coalitions without knowing that other agents have already committed to them.

3. Communication between the agents: Communication is usually necessary in order for each agent to know its share of the calculations to perform. SK requires sending an exponentially large number of bytes between the agents; this is mainly because if an agent a_i commits to a set S_{ij} of coalitions, then a_j would have to subtract this set from its list, and in order to do so, a_i would have to send S_{ij} to a_j. In contrast, DCVC requires no communications between the agents because each agent knows its share of calculations by using the provided equations, and not by negotiating with other agents.

4. Memory requirements: Any coalition of n agents can be saved in memory using n bits, where each bit indicates whether an agent is a member of the coalition. However, since the minimum unit of memory that can be allocated is one byte, we can say that the memory required per coalition is $\lceil n/8 \rceil$ bytes. Given this, Table 2 shows the number of bytes required per agent to save the necessary coalitions. As can be seen, the memory requirements grow exponentially for SK. This is because their algorithm cannot be applied without having each agent start with a list of all the possible coalitions in which it is a member. However, when using DCVC, each agent only needs to maitain in memory one coalition at a time. This makes DCVC particularly suitable for domains where very little memory space is available for the agents (e.g. agents located on mobile devices). In our case, for example, one kilobyte of memory per agent would be enough for up to 8192 agents, while each agent would have required more than $5.2 * 10^{2459}$ Gigabytes if it used SK.

5. Equality of agents' shares: Table 2 shows the difference between the agent that had the biggest share of the calculations and the one that had the smallest. DCVC has a maximum difference of 1 (because of the way it maintains and updates α). However with SK, the difference grows exponentially with the number of agents. This is because the agents' shares were arbitrarily determined based on the order in which they contacted each other. Thus, some agents were contacted by more agents than others, and so removed more coalitions from their list, and ended up with smaller shares. On the other hand, some agents contacted more agents than others, and thus committed to more coalitions, and ended up with larger shares.

Conclusions and Future Work

In this paper, we developed a novel algorithm for distributing the coalitional value calculations among cooperative agents. We then benchmarked the performance of our algorithm against the only available one in the literature. This comparison showed that our algorithm is significantly faster, requires significantly less memory space, and requires much less communication. These improvements stem from the fact that our algorithm performs no redundant calculations and distributes the calculations equally among the agents. Thus, DCVC can be seen to represent a significant advance in the state of the art.

For future work, we will concentrate on developing the enforcement mechanism so that DCVC can be applied in environments where the agents are selfish. In such cases, the agents might not necessarily perform all the calculations they are assigned or they might lie about the results they found in order to improve the outcome for themselves. The enforcement mechanism should motivate the agents to calculate the values they are assigned and to truthfully reveal the results they find.

References

Conway, J. H., and Guy, R. K. 1996. *The Book of Numbers*. New york, USA: pub-COPERNICUS.

Dang, V. D., and Jennings, N. R. 2004. Generating coalition structures with finite bound from the optimal guarantees. In *AAMAS*, 564–571.

Norman, T. J.; Preece, A.; Chalmers, S.; Jennings, N. R.; Luck, M.; Dang, V. D.; Nguyen, T. D.; Deora, V.; Shao, J.; Gray, A.; and Fiddian, A. 2004. Agent-based formation of virtual organisations. *International Journal of Knowledge Based Systems* 17(2–4):103–111.

Sandholm, T.; Larson, K.; Andersson, M.; Shehory, O.; and Tohmé, F. 1999. Coalition structure generation with worst case guarantees. *Artificial Intelligence* 111(1–2):209–238.

Shehory, O., and Kraus, S. 1998. Methods for task allocation via agent coalition formation. *Artificial Intelligence* 101(1-2):165–200.

Tsvetovat, M.; Sycara, K. P.; Chen, Y.; and Ying, J. 2000. Customer coalitions in the electronic marketplace. In *AAAI/IAAI*, 1133–1134.