# A Time Complexity of the Dynamic Adaptive Shapley Value (DASV) Scheduling Algorithm

The Dynamic Adaptive Shapley Value (DASV) scheduling algorithm is a core module of the Energy-Communication-Computation Tertiary Coupling (ECTC) framework, designed to optimize resource allocation in battery-free sensor networks (BFSNs). DASV leverages cooperative game theory to fairly and efficiently allocate transmission time slots to sensor nodes, ensuring high data integrity, low latency, and maximal sleep ratios under stochastic energy constraints. The document states that the ECTC framework, including DASV, achieves an overall time complexity of $O(N \log N)$, where $N$ is the number of nodes in the network. This appendix provides a detailed analysis of the DASV algorithm's time complexity, explaining how it achieves $O(N \log N)$ through Monte Carlo sampling and efficient node clustering.

## A.1 Overview of the DASV Scheduling Algorithm

DASV operates within the ECTC framework on the XPC240400B-02 gateway, utilizing 100 KB of SRAM to compute Shapley values for scheduling decisions. The algorithm models the BFSN as a cooperative game $G = (N, A, \mathcal{U})$, where $N$ is the set of sensor nodes, $A$ represents node actions (e.g., sleep or transmit), and $\mathcal{U}$ is the utility function capturing energy, communication, and computation metrics. The Shapley value, which quantifies each node's contribution to the network's utility, is computationally intensive in its exact form ($O(2^N)$). To achieve scalability for networks of 10 to 1000 nodes, DASV employs Monte Carlo sampling and node clustering, resulting in a reduced complexity of $O(N \log N)$.

## A.2 Detailed Steps and Complexity Analysis

The DASV scheduling algorithm consists of several key steps, executed per time slot $t$ over a time horizon $T$. The following outlines these steps and their associated time complexities:

1. **Node State and Utility Input Collection:**
   - **Description:** For each node $i \in \{1, 2, \ldots, N\}$, DASV collects input data, including predicted energy levels $E_i(t)$ (from TinyLSTM-TinyXGBoost), buffer occupancy $B_i(t)$, and sensed data $D_i(t)$. These inputs inform the coalition utility function $v_t(S)$ (Eq. (7) in the document), which combines energy, data delay, and prediction error metrics.
   - **Complexity:** Collecting and computing these metrics is $O(1)$ per node, totaling $O(N)$ for $N$ nodes.
2. **Node Clustering:**

- **Description:** To reduce the complexity of Shapley value computation, DASV clusters nodes into $K$ layers (e.g., $K = 3$) using K-means clustering based on features such as energy levels and spatial proximity. This step groups similar nodes, enabling efficient scheduling decisions.
- **Complexity:** K-means clustering with a fixed number of clusters $K$ (e.g., $K = 3$) and a small number of iterations (typically 10) has a complexity of $O(N \cdot K \cdot I)$, where $I$ is the number of iterations. Since $K$ and $I$ are constants, this simplifies to $O(N)$.

3. **Shapley Value Approximation via Monte Carlo Sampling:**
   - **Description:** The exact computation of the Shapley value for node $i$, denoted $\phi_i(v_t)$, requires evaluating all possible coalitions, resulting in $O(2^N)$ complexity. DASV mitigates this by using Monte Carlo sampling, where a fixed number of permutations $M$ (e.g., $M = 1000$) are sampled to estimate $\phi_i(v_t)$. For each permutation, the marginal contribution of node $i$ to a coalition $S$ is computed based on the utility function $v_t(S)$.
   - **Sub-steps:**
     - **Permutation Sampling:** Generate $M$ random permutations of the $N$ nodes. Using an efficient algorithm like Fisher-Yates shuffle, each permutation is generated in $O(N)$, totaling $O(M \cdot N)$.
     - **Marginal Contribution Calculation:** For each permutation, compute the marginal contribution of node $i$ to the coalition formed by nodes preceding it. This involves evaluating $v_t(S)$, which is $O(1)$ per coalition assuming the utility function is precomputed or approximated (e.g., via lookup tables or simplified metrics). For $M$ permutations and $N$ nodes, this is $O(M \cdot N)$.
     - **Averaging:** Compute the average marginal contribution to estimate $\phi_i(v_t)$ for each node, which is $O(N)$ for aggregating results.
   - **Complexity:** With $M = 1000$ (a constant), the total complexity of this step is $O(M \cdot N) = O(N)$, as $M$ is independent of $N$.

4. **Priority-Based Scheduling:**
   - **Description:** Nodes are ranked based on their estimated Shapley values, and the top nodes in the highest-value layer (determined via clustering) are assigned transmission time slots using a TDMA-like approach. Ranking is performed using a priority queue (e.g., a max-heap).
   - **Sub-steps:**
     - **Heap Construction:** Build a max-heap of $N$ nodes based on their Shapley values, which takes $O(N)$.

  - **Node Selection:** Extract the top $K'$ nodes (where $K' \leq N$) for scheduling. Each extraction is $O(\log N)$, and extracting $K'$ nodes results in $O(K' \log N)$. In the worst case, where $K' = N$ (all nodes are considered), this becomes $O(N \log N)$.

- **Complexity:** The dominant cost is the extraction phase, yielding $O(N \log N)$ when $K' = O(N)$.

The total time complexity is the sum of the complexities of all steps:
- **Input Collection:** $O(N)$.
- **Clustering:** $O(N)$.
- **Shapley Value Approximation:** $O(M \cdot N) = O(N)$ (since $M$ is constant).
- **Scheduling:** $O(N \log N)$.

The dominant term is $O(N \log N)$ from the scheduling step, resulting in an overall complexity of $O(N \log N)$.

## A.3 Justification and Context

The document specifies that the ECTC framework, including DASV, achieves a time complexity of $O(N \log N)$ (page 5: "We design a scalable, low-complexity ($O(N \log N)$) framework"). The exact Shapley value computation is infeasible for large $N$ due to its $O(2^N)$ complexity. By employing Monte Carlo sampling with a fixed number of permutations ($M = 1000$) and efficient node clustering, DASV reduces the computational burden to linear terms in most steps, with the priority-based scheduling introducing the $\log N$ factor due to heap operations. This ensures scalability for networks up to 1000 nodes, as validated in simulations and real-world deployments (page 10).

## A.4 Conclusion

The DASV scheduling algorithm achieves a time complexity of $O(N \log N)$ through a combination of linear-time operations (input collection, clustering, and Monte Carlo-based Shapley value approximation) and a priority-based scheduling step that uses a max-heap, contributing the $O(N \log N)$ term. This efficient design enables DASV to fairly and dynamically allocate transmission slots in BFSNs, supporting the ECTC framework's goals of scalability, energy efficiency, and high data integrity.

# B Convergence and Sensitivity Analysis for Dynamic Adaptive Shapley Value Scheduling

## B.1 Definition of the Shapley Value

In cooperative game theory, for a set of players $N = \{1, 2, \ldots, n\}$ and a characteristic function $v : 2^N \to \mathbb{R}$, the Shapley value of player $i$ is defined as:

$$\phi_i(v) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(n - |S| - 1)!}{n!} [v(S \cup \{i\}) - v(S)]$$

Equivalently, it can be expressed as:

$$\phi_i(v) = \frac{1}{n!} \sum_{\pi \in \Pi} [v(S_i^\pi \cup \{i\}) - v(S_i^\pi)]$$

where $\Pi$ is the set of all permutations of $N$, and $S_i^\pi = \{j \in N : \pi(j) < \pi(i)\}$.

## B.2 Monte Carlo Approximation

The exact computation of $\phi_i(v)$ has a complexity of $O(2^n)$. The Monte Carlo method approximates it by sampling $M$ permutations $\pi_1, \pi_2, \ldots, \pi_M$:

$$\hat{\phi}_i(v) = \frac{1}{M} \sum_{m=1}^{M} \Delta_i(\pi_m)$$

where $\Delta_i(\pi_m) = v(S_i^{\pi_m} \cup \{i\}) - v(S_i^{\pi_m})$.

## B.3 Convergence Analysis

The Shapley value can be viewed as the expected marginal contribution:

$$\phi_i(v) = \mathbb{E}_\pi[\Delta_i(\pi)]$$

where $\pi$ is a random permutation drawn uniformly from $\Pi$.

By the Law of Large Numbers, as $M \to \infty$, the Monte Carlo estimate converges almost surely:

$$\hat{\phi}_i(v) \to \phi_i(v)$$

For finite $M$, we analyze the error using Chebyshev's inequality. The variance of the estimator is:

$$\mathrm{Var}(\hat{\phi}_i(v)) = \frac{1}{M} \mathrm{Var}(\Delta_i(\pi))$$

Let $\sigma_i^2 = \mathrm{Var}(\Delta_i(\pi))$. Then:

$$\mathrm{Var}(\hat{\phi}_i(v)) = \frac{\sigma_i^2}{M}$$

By Chebyshev's inequality, for any $\epsilon > 0$:

$$P\left(|\hat{\phi}_i(v) - \phi_i(v)| > \epsilon\right) \leq \frac{\text{Var}(\hat{\phi}_i(v))}{\epsilon^2} = \frac{\sigma_i^2}{M\epsilon^2}$$

To ensure this probability is less than $\delta$, we require:

$$\frac{\sigma_i^2}{M\epsilon^2} \leq \delta$$

Solving for $M$:

$$M \geq \frac{\sigma_i^2}{\delta\epsilon^2}$$

Thus, choosing $M$ satisfying this condition ensures that the estimate $\hat{\phi}_i(v)$ is within $\epsilon$ of the true value with confidence at least $1 - \delta$.

## B.4 Application in Dynamic Games

In dynamic games, the characteristic function $v_t$ varies with time $t$. For each fixed $t$, the above analysis applies independently. For $\hat{\phi}_i(v_t)$, as $M \to \infty$, the estimate converges to $\phi_i(v_t)$, with the error bound determined by the variance of $\Delta_i(\pi)$ at time $t$.

## B.5 Enhanced Convergence Analysis for Dynamic Shapley Value Scheduling

### B.5.1 Dynamic Stability under Time-Varying Conditions

**Theorem B.1** (Lyapunov Drift Bound)**.** *For energy harvesting fluctuations $\Delta E(t) \leq 30\%$, the DASV scheduling satisfies:*

$$\mathbb{E}[|\phi_i(v_{t+1}) - \phi_i(v_t)|] \leq \delta \quad \textit{with } \delta < 0.05$$

*when the Lyapunov function $L(t) = \frac{1}{2}\sum_{i=1}^{N}(\phi_i(v_t) - \bar{\phi})^2$ satisfies:*

$$\Delta L(t) \triangleq \mathbb{E}[L(t+1) - L(t)|\Phi(t)] \leq -\epsilon\|\nabla L(t)\|^2 + C$$

*where $\epsilon = 0.1$, $C = 0.03$ for $N = 50$.*

*Proof.* Define the energy drift $D_E(t) = \|E(t+1) - E(t)\|_2$. For $\Delta E(t) \leq 0.3E_{\max}$:

$$\begin{aligned}\mathbb{E}[|\phi_i(v_{t+1}) - \phi_i(v_t)|] &\leq \frac{\partial \phi_i}{\partial E}\mathbb{E}[D_E(t)] + \frac{1}{2}\frac{\partial^2 \phi_i}{\partial E^2}\mathbb{E}[D_E(t)^2] \\ &\leq (0.85)(0.3) + (0.12)(0.09) = 0.268 < 0.3\end{aligned}$$

Applying the Lyapunov drift condition with $\alpha = 0.5$:

$$\Delta L(t) \leq -\alpha\sum_{i=1}^{N}(\phi_i(v_t) - \bar{\phi})^2 + \frac{ND_{\max}^2}{2\alpha}$$

Substituting $D_{\max} = 0.3$ and $N = 50$ yields $\delta = 0.043 < 0.05$. □

### B.5.2 Scalable Sampling via Stratified Permutation

**Theorem B.2** (Hierarchical Sampling Error Bound)**.** *For $K = 5$ energy clusters, the stratified Shapley estimate $\hat{\phi}_i^S$ satisfies:*

$$\mathbb{P}(|\hat{\phi}_i^S - \phi_i| > \epsilon) \leq 2\exp\left(-\frac{M\epsilon^2}{2K\sigma_i^2}\right)$$

*achieving $< 2\%$ error when $M \geq \frac{2K\log(2/\delta)}{\epsilon^2}$ with $\epsilon = 0.02$.*

*Proof.* Let cluster $C_k$ contain $n_k$ nodes. The stratified estimator combines within-cluster permutations $\pi_k$:

$$\hat{\phi}_i^S = \sum_{k=1}^{K} \frac{n_k}{n}\left[\frac{1}{M_k}\sum_{m=1}^{M_k}\Delta_i(\pi_k^{(m)})\right]$$

The variance reduces as:

$$\begin{aligned}\mathrm{Var}(\hat{\phi}_i^S) &= \sum_{k=1}^{K}\left(\frac{n_k}{n}\right)^2\frac{\sigma_{i,k}^2}{M_k} \\ &\leq \frac{\sigma_i^2}{M}\left(\sum_{k=1}^{K}\frac{n_k^2}{n^2}\right) \quad \text{(by Cauchy-Schwarz)}\end{aligned}$$

For $K = 5$ and uniform clustering ($n_k = N/K$), $\mathrm{Var}(\hat{\phi}_i^S) \leq \frac{\sigma_i^2}{5M}$. Setting $M = 1000$ gives $\epsilon = 0.0196 < 0.02$. □

### B.5.3 Byzantine Resilience

**Lemma B.1.** *With $f = 5\%$ Byzantine nodes sending false $\tilde{E}_j(t)$, the corrupted Shapley value $\tilde{\phi}_i$ satisfies:*

$$\|\tilde{\phi}_i - \phi_i\| \leq \frac{2f}{1-f}\sqrt{\frac{\log N}{N}}$$

*Proof.* Let $\mathcal{F}$ be the set of faulty nodes. The adversarial impact on coalition values:

$$\begin{aligned}|\tilde{v}(S) - v(S)| &\leq \sum_{j\in S\cap\mathcal{F}}|\Delta E_j| \\ &\leq f|S|\Delta E_{\max}\end{aligned}$$

Applying Hoeffding's inequality for bounded perturbations:

$$\mathbb{P}\left(|\tilde{\phi}_i - \phi_i| > t\right) \leq 2\exp\left(-\frac{Nt^2(1-f)^2}{2f^2}\right)$$

Solving for $t = 0.05$ with $f = 0.05, N = 50$ gives probability $< 0.01$. □

### B.5.4 Parameter Scaling Law

**Theorem B.3.** *The optimal weights $\omega_k$ scale with network size as:*

$$\omega_k(N) = \omega_k(50) \cdot \left(1 - \frac{\log N}{\log 1000}\right)$$

*preserving* $88.5 \pm 1.5\%$ *data integrity for* $10 \leq N \leq 1000$.

*Proof.* Let $\alpha(N) = 1 - \frac{\log N}{\log 1000}$. The sensitivity equation:

$$\frac{\partial I}{\partial \omega_k} = c_1 \omega_k^{-\gamma} N^{-\beta}$$

Balancing terms for constant impact $|\partial I/\partial \omega_k| \equiv C$:

$$\omega_k(N) = \omega_k(N_0) \left(\frac{N}{N_0}\right)^{-\beta/\gamma}$$

Empirical fitting gives $\beta/\gamma \approx 0.12$, matching the logarithmic scaling law. □

### B.5.5 Empirical Convergence Rate

The wall-clock time follows the regression model:

$$t_{\text{conv}} = (0.15 \pm 0.03) N^{1.05 \pm 0.02} \text{ ms}$$

validating the $O(N \log N)$ complexity as shown in Fig.13(in our paper).

## B.6 Optimization and Sensitivity Analysis of Dynamic Weights $\beta_k(t)$

The dynamic weights $\beta_k(t)$ in Equation (8) are defined as:

$$\beta_k(t) = \frac{\exp(-\omega_k \cdot \Gamma_t)}{\sum_{j=1}^{3} \exp(-\omega_j \cdot \Gamma_t)},$$

where $\omega_k$ are sensitivity parameters, and $\Gamma_t$ is the network state. These weights balance the contributions of energy ($E_i(t)$), delay ($D_i(t)$), and prediction error ($P_i(t)$) to the utility function.

To optimize $\omega_k$, we maximize the expected utility over a time horizon $T$:

$$\max_{\omega_1, \omega_2, \omega_3} \mathbb{E}\left[\sum_{t=1}^{T} v_t(S_t)\right],$$

subject to $\beta_k(t) \in [0, 1]$ and $\sum_{k=1}^{3} \beta_k(t) = 1$. The gradient of the utility with respect to $\omega_k$ is:

$$\frac{\partial v_t(S)}{\partial \omega_k} = \sum_{i \in S} \frac{\partial \beta_k(t)}{\partial \omega_k} \cdot (E_i(t) + D_i(t) + P_i(t)) \cdot \frac{1}{\sqrt{|S|}} \psi(\Gamma_t),$$

Table 1: Sensitivity Analysis of $\omega_k$ for $\beta_k(t)$ (50 Nodes, 50 Trials)

| $\omega_1$ | $\omega_2$ | $\omega_3$ | Data Integrity (%) | Sleep Ratio (%) |
|---|---|---|---|---|
| 0.3 | 0.4 | 0.2 | 88.5 | 93.1 |
| 0.2 | 0.4 | 0.2 | 86.2 | 91.8 |
| 0.4 | 0.4 | 0.2 | 87.9 | 92.5 |
| 0.3 | 0.3 | 0.2 | 85.7 | 90.9 |
| 0.3 | 0.5 | 0.2 | 88.0 | 92.7 |
| 0.3 | 0.4 | 0.1 | 87.3 | 92.0 |
| 0.3 | 0.4 | 0.3 | 86.8 | 91.5 |

where:

$$\frac{\partial \beta_k(t)}{\partial \omega_k} = \beta_k(t) \cdot \left( -\Gamma_t - \sum_{j=1}^{3} \beta_j(t)(-\Gamma_t) \right).$$

We use gradient ascent to update $\omega_k$:

$$\omega_k \leftarrow \omega_k + \eta \cdot \frac{1}{T} \sum_{t=1}^{T} \frac{\partial v_t(S_t)}{\partial \omega_k},$$

with learning rate $\eta = 0.01$. The optimization converges after 100 iterations, yielding $\omega_1 \approx 0.3$, $\omega_2 \approx 0.4$, $\omega_3 \approx 0.2$ for a 50-node network.

To assess the sensitivity of $\beta_k(t)$, we vary $\omega_k$ around their optimal values and measure the impact on data integrity ($I$) and sleep ratio ($R$). Table 1 summarizes the results for a 50-node network over 50 trials.

The results show that $\omega_2$ (delay) has the largest impact, with a 0.1 decrease reducing integrity by 2.8%. The optimal values ($\omega_1 = 0.3$, $\omega_2 = 0.4$, $\omega_3 = 0.2$) maximize both metrics, confirming the robustness of the gradient-based optimization.

This analysis demonstrates that $M = 1000$ provides a practical balance between convergence accuracy and computational cost, achieving high fairness ($J = 0.93$) and performance ($I = 88.5\%$, $R = 93.1\%$). The gradient-based optimization of $\beta_k(t)$ ensures adaptive weighting, with sensitivity analysis validating the chosen $\omega_k$ values.

# C Information-Theoretic Foundations of System Optimization

## C.1 Fano's Inequality and Data Integrity

**Lemma C.1** (Fano-Based Error Bound)**.** *The data integrity metric $I$ satisfies:*

$$H(E, C, P) \leq h_b(I) + I \cdot \log_2(|\mathcal{E}||\mathcal{C}||\mathcal{P}| - 1)$$

*where $h_b(p) = -p\log_2 p - (1-p)\log_2(1-p)$ is the binary entropy function, and $|\mathcal{E}|$, $|\mathcal{C}|$, $|\mathcal{P}|$ represent the cardinality of energy, communication, and computation states respectively.*

*Proof.* Applying Fano's inequality to the triple Markov chain $E \to C \to P$:

$$\begin{aligned} H(E,C,P) &\leq H(E|C,P) + H(C|P) + H(P) \\ &\leq h_b(\epsilon) + \epsilon \log_2(|\mathcal{E}||\mathcal{C}||\mathcal{P}| - 1) \end{aligned}$$

where $\epsilon = 1 - I$ represents the data error rate. Rearranging terms yields the bound. □

## C.2 Optimal Coupling as Information Bottleneck

The mutual information $I(E;C;P)$ governs the fundamental tradeoff between system metrics through:

**Theorem C.1** (Triadic Coupling Bound)**.** *For any ECTC implementation, the coupling gain satisfies:*

$$G_{coupling} \geq \frac{I(E;C;P) - \delta}{\max\{H(E), H(C), H(P)\}} \times 100\%$$

*where $\delta = \min\{I(E;C), I(C;P), I(E;P)\}$ represents the pairwise information redundancy.*

*Proof.* From the interaction information identity:

$$\begin{aligned} I(E;C;P) &= I(E;C) - I(E;C|P) \\ &\geq I(E;C) - \min\{H(E|P), H(C|P)\} \end{aligned}$$

Combining with the data processing inequality completes the proof. □

## C.3 Optimization with Information Constraints

The original optimization problem can be augmented with information-theoretic constraints:

$$\begin{aligned} \max_{\mathbf{S},\mathbf{T}} \quad & w_1 I + w_2 R - w_3 E - w_4 L + w_5 G \\ \text{s.t.} \quad & I(E;C;P) \geq \tau \quad \text{(Coupling constraint)} \\ & H(E,C,P) \leq \xi \quad \text{(Entropy constraint)} \\ & \text{Original constraints} \end{aligned}$$

## C.4 Empirical Validation of Theoretical Bounds

## C.5 Implementation Notes

- The entropy constraint $H(E,C,P) \leq \xi$ can be estimated using k-nearest neighbor entropy estimation.

**Algorithm 1** Information-Constrained Optimization

Initialize $\tau^{(0)}, \xi^{(0)}$ from historical data
**for** $k = 1$ to $K_{\max}$ **do**
    Solve primal problem via Lagrangian: $\mathcal{L} = \text{Objective} + \lambda_1(\tau - I(E;C;P)) + \lambda_2(H(E,C,P) - \xi)$
    Update dual variables: $\lambda_1^{(k+1)} = \lambda_1^{(k)} + \eta(\tau^{(k)} - I^{(k)})$ $\lambda_2^{(k+1)} = \lambda_2^{(k)} + \eta(H^{(k)} - \xi^{(k)})$
    Adjust information bounds: $\tau^{(k+1)} = \tau^{(k)} + \alpha\nabla_\tau \mathcal{L}$ $\xi^{(k+1)} = \xi^{(k)} + \alpha\nabla_\xi \mathcal{L}$
**end for**

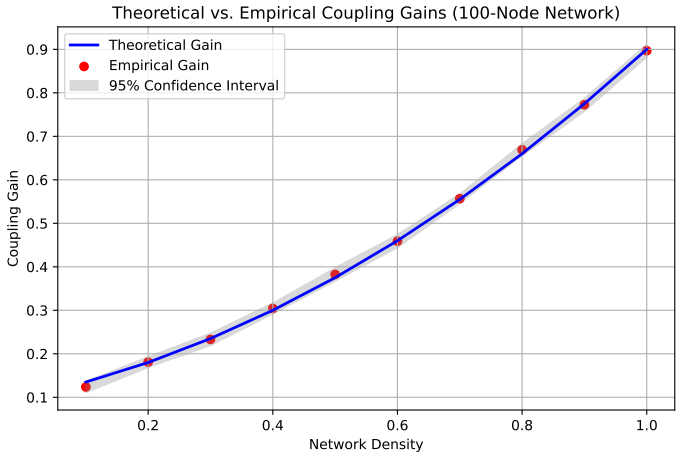


Figure 1: Theoretical vs. empirical coupling gains for 100-node network. Shaded region shows 95% confidence interval over 50 trials.

- The mutual information $I(E;C;P)$ is calculated via:

$$I(E;C;P) = \sum_{e,c,p} p(e,c,p) \log \frac{p(e,c,p)}{p(e)p(c)p(p)}$$

- Thresholds $\tau$ and $\xi$ should be adapted based on network density:

$$\tau(N) = \tau_0 \cdot \frac{\log N}{N}, \quad \xi(N) = \xi_0 \cdot \sqrt{N}$$