

TUHIN KUMAR BISWAS
ANIMESH DUTTA

OPTIMIZING COALITION STRUCTURE IN LARGE SCALE COMBINED SCSGA-MF-TS

Abstract *The key focus of the Simultaneous Coalition Structure Generation and Assignment (SCSGA) problem is to identify a feasible coalition for each task that maximizes the value of the resulting coalition structure. Incorporating multi-dimensional features on agents and tasks, the SCSGA with Multi-dimensional Features (SCSGA-MF) problem aims to minimize the distance between each coalition and its assigned task, while SCSGA-MF prioritizing Task Satisfaction (SCSGA-MF-TS) additionally focuses on maximizing the total number of satisfied tasks. In this work, we study and address the SCSGA-MF-TS problem as a combined combinatorial optimization problem, where the two objectives of SCSGA-MF-TS are combined into a single function. For this problem, we adapt a well known evolutionary algorithm and several heuristics to study their performances on large scale problem instances. Our empirical studies show that the genetic algorithm significantly outperforms other contending methods on the combined SCSGA-MF-TS problem.*

Keywords coalition structure generation, combined combinatorial optimization, task satisfaction, evolutionary algorithm

Citation Computer Science 27(2) 2026: 47–74

Copyright © 2026 Author(s). This is an open access publication, which can be used, distributed and reproduced in any medium according to the Creative Commons CC-BY 4.0 License.

1. Introduction

A significant topic of study in the fields of Multi-Agent Systems (MAS), combinatorial optimization, and Artificial Intelligence (AI) is the Simultaneous Coalition Structure Generation and Assignment (SCSGA) problem. SCSGA is derived from the Coalition Structure Generation (CSG) problem [23], which seeks to create disjoint partitions of the agents in order to maximize the total valuation of the resultant coalition structure. On the other hand, the main goal of SCSGA [22] is to generate a coalition structure where each coalition accomplishes a specific task so as to maximize the total value of the coalition structure. With a search space equal to $(m! \times m^n)$ potential coalition structures, SCSGA explores $m \times 2^n$ coalition values for n agents and m tasks [20]. As the number of agents and tasks increases, the search space grows exponentially, making SCSGA challenging to solve. The SCSGA problem has therefore only been addressed for small-sized inputs in the literature [20–22]. For the SCSGA problem, however, the recently introduced different anytime algorithms [19, 25] have been used on the largest problem sizes, with up to 2000 agents and 50 tasks.

In SCSGA, the value of any coalition assigned to a task is assumed randomly, which does not interpret any specific characteristics of agents and tasks. Hence, to make SCSGA more real-world adapted and human-interpretable, in State-of-the-Art SOTA, two further advancements of SCSGA are established, one is the SCSGA with Multi-dimensional Features (SCSGA-MF) problem [5], and another is the SCSGA-MF prioritizing Task Satisfaction (SCSGA-MF-TS) problem [4]. Both this SCSGA-MF and SCSGA-MF-TS advance the SCSGA problem in terms of incorporating the d -dimensional features on each agent representing agents' different skills and on each task representing different requirements of the tasks. Based on these d -dimensional features, both the SCSGA-MF and SCSGA-MF-TS seek to minimize the total distances between each coalition and its assigned task. In order to better match the requirements of the tasks, this consideration guarantees that the agents with more relevant skills are assigned to the tasks that are closest to them. However, while in SCSGA-MF, the closest agents are often assigned to tasks to prioritize their mutual compatibility, this does not guarantee the satisfaction of the tasks' requirements. Hence, the SCSGA-MF-TS problem further advances the SCSGA-MF problem by considering the optimization of two objectives simultaneously: (1) minimizing the cumulative distance between the features agents present in a coalition and the features of the corresponding assigned task (referred to as solution value), and (2) maximizing the total number of satisfied tasks by their corresponding assigned coalitions. Any task is considered to be satisfied by a coalition if the sums for each feature over all agents in the coalition are at least equal to the corresponding task features. In contrast to SCSGA-MF, the SCSGA-MF-TS utilizes only a single distance metric (Euclidean Distance metric) instead of a set of distance metrics as in SCSGA-MF, to calculate the distance between any two d -dimensional feature vectors.

In this work, following the advancement hierarchy of the SOTA, we focus on addressing the further advancement of the SCSGA-MF-TS problem, as it is the cur-

rent advancement over the SCSGA and SCSGA-MF problems. In SCSGA-MF-TS, two objectives are optimized separately, where in any instance, either of the objectives gets priority in the optimization process. But to make SCSGA-MF-TS more real-world adapted, these two objectives should get equal importance, and the weightage of such importance is likely to be reflected in the problem formulation. One efficient way to achieve this is to represent the SCSGA-MF-TS problem as a combined combinatorial optimization problem, where both objectives of SCSGA-MF-TS are combined with the aim of optimizing a single function that captures the characteristics of both objectives of SCSGA-MF-TS simultaneously. But SCSGA-MF-TS does not have such consideration within it, and it can also be verified by the solution approach of SCSGA-MF-TS. The Compact Agent Neighborhood (CAN) search algorithm [4] optimizes the two objectives of SCSGA-MF-TS in an iterative manner, where in each iteration, any step of the CAN search either focuses on minimizing the solution value or maximizing the number of satisfied tasks by their corresponding assigned coalitions. But these two objectives are not optimized in any single step of the CAN search. It may lead to generating bad solutions, as there may occur any step that selects a set of coalitions that maximizes the number of satisfied tasks, but can not minimize the solution value: maybe the selection of another set of coalitions could produce a much lower solution value. Hence, considering such aspects, the first and foremost contribution of this work is to propose a novel combined combinatorial optimization version of the SCSGA-MF-TS problem. In this work, we propose a combined combinatorial optimization problem, in which we combine the two objectives of SCSGA-MF-TS and provide a unified framework for optimizing those two objectives within a single optimization function. Let us illustrate the significance of such a formulation of SCSGA-MF-TS by visualizing a practical scenario of assigning n employees (agents) to m different projects (tasks). Each employee possesses multiple skills, and each project requires a specific amount of skills in certain domains. Assume each project requires a specific amount of expertise in Java programming, Python programming, and database management. While each employee has these skills to a specific degree. Here, the higher degree of any skill, among employees, represents a higher expertise level the employees have on that skill, and on projects, represents the higher expertise level required on that skill. For such a scenario, the combined SCSGA-MF-TS problem is applicable with the objective of generating feasible coalitions of employees and assigning them to the corresponding project. Here, the applicability of the combined SCSGA-MF-TS provides a unified framework that enables the combining and measuring of the aspects of assigning suitable employees that can best supply the skills required to successfully complete the projects, as well as of satisfying all the requirements of the projects, simultaneously in any instance of the optimization process.

To the best of our knowledge, in the literature, this is the first study on the combined SCSGA-MF-TS problem, so solving such a problem not only specifies a novel significance in SOTA, but also represents a promising research direction in the domain of SCSGA, SCSGA-MF, and SCSGA-MF-TS. Hence, after successfully proposing the

novel combined combinatorial optimization problem against the SOTA in this work, our next contribution is to solve such a problem for large-scale inputs within a reasonable time frame. We discuss that the two objectives of SCSGA-MF-TS are not optimized in any single step of the CAN search [4] algorithm. Also, the CAN search involves a step identifying the neighborhood of agents in different coalitions, which incurs high computation, which limits its ability to search the solution space efficiently within a prespecified computation budget. Over the SOTA, this also motivates us to formulate and study the SCSGA-MF-TS problem as a combined combinatorial optimization problem and to be solved by a more efficient optimization method. In this work, we are motivated by the general success of evolutionary algorithms in addressing different optimization problems, and thus, to address the SCSGA-MF-TS problem as a combined combinatorial optimization problem, we propose adapting one evolutionary algorithm along with several heuristic algorithms. Such heuristics became top-performing methods in SOTA for solving SCSGA-MF-TS. Adapting those for combined SCSGA-MF-TS not only validates the applicability of the heuristics for the combined combinatorial optimization problem but also provides a baseline framework to compare the performance of evolutionary algorithms against an SOTA-like environment.

Specifically, in this work, we adapt the evolutionary method of Genetic Algorithm (GEA), along with the heuristics of Hill Climbing Algorithm (HCA) and Simulated Annealing Algorithm (SAA), which are adapted from the SOTA [4] of the SCSGA-MF-TS problem. To the best of our knowledge, the combined version of SCSGA-MF-TS and the exploration of evolutionary algorithms on such a problem have not been explored previously. Therefore, we study the performances of the GEA, HCA, and SAA on the SCSGA-MF-TS problem treated as a combined combinatorial optimization problem. We also test the performance of these algorithms on the general SCSGA-MF-TS problem as described in [4], in which both objectives of SCSGA-MF-TS are optimized separately. In our empirical studies, we provide comparable setups to evaluate the performance of all the algorithms: for GEA, we generate a set of random candidate solutions that GEA is allowed to improve up to a fixed number of iterations. Both HCA and SAA are set to run up to a fixed time limit, which is equal to the maximum execution time of GEA initialized with a single random candidate solution. The experimental results reveal the superiority in performance of GEA over HCA and SAA for the combined combinatorial optimization problem of SCSGA-MF-TS. Moreover, we represent the capability of the proposed methods to solve the combined SCSAG-MF-TS problem for large-scale inputs within a reasonable time frame.

1.1. Scope of the work

According to the above discussion, we summarize that following the advancement hierarchy of the SOTA, the primary significance of this study is to address further advancement of the SCSGA-MF-TS problem for making it a more real-world-adapted

and human-interpretable problem. In SOTA, the SCSGA-MF-TS problem and its solution approach focus on optimizing either of the two objectives in any iteration of the optimization process. Thus, the two objectives of SCSGA-MF-TS do not get equal importance at the same time within any single step of the optimization process. Hence, in this study, such considerations of SOTA motivate us to focus on further advancement of the SCSGA-MF-TS problem by proposing a novel combined combinatorial optimization problem that combines the two objectives of the SCSGA-MF-TS problem. This combined version of SCSGA-MF-TS provides a framework where both objectives will achieve equal importance in the optimization process, irrespective of the solution approach. Thus making the SCSGA-MF-TS a more human-interpretable and real-world-adapted problem. Along with such a first and foremost significance of this study, additionally, the next significance of this study is to solve such a combined combinatorial optimization problem for large-scale inputs within a reasonable time frame. As the exponential growth of the search space also makes the combined SCSGA-MF-TS problem hard to solve, like SCSGA, SCSGA-MF, and SCSGA-MF-TS, when the number of agents and tasks increases. To the best of our knowledge, we believe that this is the first study in the literature that addresses the SCSGA-MF-TS problem as a combined combinatorial optimization, and thus, it is obvious that no explicit methodologies are available to solve such a problem in SOTA. In this study, the exploration of evolutionary and heuristic algorithms not only solves the novel combined SCSGA-MF-TS problem for large-scale inputs but also reveals a promising research direction on the applicability of these algorithms for the advancement of SCSGA, SCSGA-MF, and SCSGA-MF-TS problems for the first time in the literature. Finally, the following is a summary of our primary contributions:

1. We propose and address a novel combined combinatorial optimization problem in which the two objectives of SCSGA-MF-TS are combined. To the best of our knowledge, this is the first study that further advances the current development of SOTA regarding the SCSGA, SCSGA-MF, and SCSGA-MF-TS problem.
2. We propose adapting an evolutionary algorithm and heuristic algorithms and thereby study their performances on the combined SCSGA-MF-TS problem. For the first time in literature, such an exploration of evolutionary algorithms in the domain of SCSGA, SCSGA-MF, and SCSGA-MF-TS provides a promising research direction for solving the combined combinatorial optimization problem.
3. We adapt our proposed heuristic algorithms from the SOTA of the SCSGA-MF-TS problem and evaluate the performance of the proposed evolutionary algorithm against those heuristic algorithms. This offers a comparison depiction of the evolutionary algorithm versus SOTA, as well as the examination of the combined SCSGA-MF-TS problem in an environment that is similar to SOTA.
4. We also study and analyze the performance of the proposed evolutionary and heuristic algorithms on the general SCSGA-MF-TS problem, where the two objectives of SCSGA-MF-TS are optimized separately.
5. Our experimental demonstration reveals the superiority of the evolutionary al-

gorithm over other contending methods when the SCSGA-MF-TS problem is considered as a combined combinatorial optimization problem.

The following is a summary of the remainder of the article. Section 2 presents the formulation of the combined SCSGA-MF-TS problem, and Section 3 discusses the pertinent literature. In Section 4, the proposed algorithms are detailed. Section 5 then discusses the experimental evaluation and analyzes the observed results. Conclusions are finally discussed in Section 6, followed by the relevant references.

2. Problem Formulation

In this work we follow the formulation of the SCSGA-MF-TS problem [4], where, a set of n heterogeneous agents is defined as $\mathcal{A} = \{a_1, \dots, a_n\}$, and a set of m independent tasks as $\mathcal{T} = \{t_1, \dots, t_m\}$. Every task $t_j \in \mathcal{T}$ can be executed independently of the others at the same time. The d -dimensional features that correspond to each agent and task in $\mathcal{A}$ and $\mathcal{T}$, respectively, capture the real values for the various resources or skills of the agents as well as the requirements of the tasks. Formally, the d -dimensional feature vector for each agent $a_i \in \mathcal{A}$ is $\mathbf{R}(a_i) = (r_1(a_i), \dots, r_d(a_i))$, and for each task $t_j \in \mathcal{T}$ is $\mathbf{R}(t_j) = (r_1(t_j), \dots, r_d(t_j))$. Here $r_k(\cdot) \in \mathbb{R}$ is the feature value for the k -th skill of the agents and requirement of the tasks. Therefore, $\mathcal{R}(\mathcal{A}) = \{\mathbf{R}(a_1), \mathbf{R}(a_2), \dots, \mathbf{R}(a_n)\}$ indicates the d -dimensional features for each agent in $\mathcal{A}$, and similarly $\mathcal{R}(\mathcal{T}) = \{\mathbf{R}(t_1), \mathbf{R}(t_2), \dots, \mathbf{R}(t_m)\}$ indicates the d -dimensional features for each task in $\mathcal{T}$.

Let us review the example from section 1 to visualize how these features are distributed effectively in any real-world scenario. The software development company wants to complete two projects, namely t_1 and t_2 . The company has three employees: a_1, a_2, a_3 . Each project requires expertise in three specific skills: r_1 :=Java Programming, r_2 := Python Programming, and r_3 := Database Management. According to these considerations, we present two skill matrices below. For an employee a_i , the feature components $r_1(a_i)$, $r_2(a_i)$, and $r_3(a_i)$ represent the proficiency levels of employee a_i for the skills r_1 , r_2 , and r_3 , with higher values denoting better or, higher proficiencies. Similarly, for each project t_j , the feature components $r_1(t_j)$, $r_2(t_j)$, and $r_3(t_j)$ denote the requirements of a particular amount of skill or resources that are necessary to complete the project successfully.

$$\begin{pmatrix} & r_1 & r_2 & r_3 \\ a_1 & 0.4 & 0.5 & 0.3 \\ a_2 & 1 & 0.8 & 0.2 \\ a_3 & 0.1 & 0.7 & 0.4 \end{pmatrix} \quad \begin{pmatrix} & r_1 & r_2 & r_3 \\ t_1 & 0.8 & 0.7 & 0.1 \\ t_2 & 0.5 & 1 & 0.6 \end{pmatrix}$$

Any coalition $\mathcal{C}_j = \{a_{i_1}, \dots, a_{i_{|\mathcal{C}_j|}}\} \subseteq \mathcal{A}$ is the collection of l agents (where, $l = |\mathcal{C}_j|$) and any coalition structure $\mathcal{CS} = \{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_m\}$, is the partitioning of $\mathcal{A}$ into m disjoint coalitions, where any coalition $\mathcal{C}_j \in \mathcal{CS}$ is assigned to task $t_j \in \mathcal{T}$. Additionally, the following conditions must be satisfied for any coalition structure $\mathcal{CS}$:

$$\forall \mathcal{C}_j, \mathcal{C}_k \in \mathcal{CS} : \mathcal{C}_j, \mathcal{C}_k \neq \emptyset, \mathcal{C}_j \cap \mathcal{C}_k = \emptyset, \bigcup_{j=1}^m \mathcal{C}_j = \mathcal{A}.$$

In this work, we define that for an ordered coalition structure $\mathcal{CS} = \{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_m\}$ and an ordered set of tasks $\{t_1, \dots, t_m\}$, each task t_j is uniquely assigned to a single corresponding coalition $\mathcal{C}_j$. Thus, a coalition $\mathcal{C}_j$ within an ordered m -sized coalition structure $\mathcal{CS}$ is assigned only to task $t_j \in \mathcal{T}$ and hence has only one unique valuation. Likewise, each $\mathcal{CS}$ can be assigned one unique valuation. Generally, any random coalition $\mathcal{C}_j$ is assigned a d -dimensional value as: $v(\mathcal{C}_j) = \{\sum_{i=1}^{|\mathcal{C}_j|} r_1(a_{i_i}), \sum_{i=1}^{|\mathcal{C}_j|} r_2(a_{i_i}), \dots, \sum_{i=1}^{|\mathcal{C}_j|} r_d(a_{i_i})\}$.

We represent the Euclidean distance metric between any two d -dimensional vectors x and y as $d_{\mathcal{M}}(x, y) = \sqrt{(x_1 - y_1)^2 + \dots + (x_d - y_d)^2}$ in our problem setting. As a measure of similarity between the d -dimensional agents and tasks, we efficiently employ such distance metric $d_{\mathcal{M}}(x, y)$. For any coalition $\mathcal{C}_j$ assigned to any task $t_j \in \mathcal{T}$, we define its valuation as: $v(\mathcal{C}_j, t_j) = \sum_{i=1}^{|\mathcal{C}_j|} d_{\mathcal{M}}(\mathbf{R}(a_{i_i}), \mathbf{R}(t_j))$. Likewise, we determine the valuation of any $\mathcal{CS}$ as: $v(\mathcal{CS}) = \sum_{j=1}^m v(\mathcal{C}_j, t_j)$. As the calculation of $v(\mathcal{CS})$ is purely based on a distance metric, hence, minimization of the $v(\mathcal{CS})$ ensures the formation of compact coalitions, where the agents present in a coalition hold the similar characteristics to their corresponding assigned task.

Definition 1. $v(\mathcal{CS})$ represents the **solution value** of any ordered $\mathcal{CS}$, which includes m disjoint coalitions and the **solution value** is therefore defined as the sum of the distances between each coalition and its associated task:

$$v(\mathcal{CS}) = \sum_{j=1}^m v(\mathcal{C}_j, t_j) = \sum_{j=1}^m \sum_{a_i \in \mathcal{C}_j} d_{\mathcal{M}}(\mathbf{R}(a_i), \mathbf{R}(t_j)).$$

It is important to mention that, for any arbitrary $\mathcal{CS}$, the agents in a coalition $\mathcal{C}_j \in \mathcal{CS}$ may not always satisfy all the requirements of the corresponding assigned task $t_j \in \mathcal{T}$. In light of this, we introduce Definition 2, formally defining the scenario in which the requirements of t_j are deemed to be fulfilled by the corresponding assigned coalition $\mathcal{C}_j$ within the coalition structure $\mathcal{CS}$.

Definition 2. Each coalition $\mathcal{C}_j$ in an ordered $\mathcal{CS}$ is specifically assigned to its corresponding task t_j . Each coalition $\mathcal{C}_j$ is said to have satisfied the requirements of each task t_j if and only if the following inequalities hold simultaneously:

$$\left\{ \sum_{a_i \in \mathcal{C}_j} r_1(a_i) \geq r_1(t_j), \dots, \sum_{a_i \in \mathcal{C}_j} r_d(a_i) \geq r_d(t_j) \right\}.$$

The requirements of zero or more tasks can therefore be satisfied by an arbitrarily ordered $\mathcal{CS}$. Therefore, using the Definition 3 stated below, the total number of tasks whose requirements are satisfied by a $\mathcal{CS}$ can formally be counted.

Definition 3. In an ordered $\mathcal{CS}$, if task t_j is satisfied by coalition $\mathcal{C}_j$ as per Definition 2, then we define $\text{count}(\mathcal{C}_j, t_j) = 1$. The **Task satisfaction Count** $TC(\mathcal{CS}, \mathcal{T})$ is then defined as: $TC(\mathcal{CS}, \mathcal{T}) = \sum_{j=1}^m \text{count}(\mathcal{C}_j, t_j)$, which represents the total number of tasks satisfied by $\mathcal{CS}$.

Using all of these factors, we present the following SCSGA-MF prioritizing Task Satisfaction (SCSGA-MF-TS) problem, which is to find a coalition structure $\mathcal{CS}$ that maximizes the count of tasks satisfied, given by $TC(\mathcal{CS}, \mathcal{T})$, while simultaneously minimizing the solution value $v(\mathcal{CS})$.

Problem: SCSGA-MF prioritizing Task Satisfaction (SCSGA-MF-TS)

Input: A tuple $\{\mathcal{A}, \mathcal{T}, \mathcal{R}(\mathcal{A}), \mathcal{R}(\mathcal{T}), v\}$; where $\mathcal{A}$ is the set of n agents $\{a_1, a_2, \dots, a_n\}$, $\mathcal{T}$ is the set of m tasks $\{t_1, t_2, \dots, t_m\}$, $\mathcal{R}(\mathcal{A}) = \{\mathbf{R}(a_1), \dots, \mathbf{R}(a_n)\}$ represents the feature vectors of the agents, $\mathcal{R}(\mathcal{T}) = \{\mathbf{R}(t_1), \dots, \mathbf{R}(t_m)\}$ represents the feature vectors of the tasks, $v(\mathcal{C}_j, t_j)$ denotes the value of a coalition $\mathcal{C}_j$ assigned to task t_j , and $v(\mathcal{CS})$ denotes the value of a $\mathcal{CS}$.

Output: An ordered coalition structure, $\mathcal{CS} = \{\mathcal{C}_1, \dots, \mathcal{C}_m\}$, that optimizes the following:

$$\text{minimizing } v(\mathcal{CS}) \text{ \& \textit{maximizing} } TC(\mathcal{CS}, \mathcal{T}), \quad (1)$$

where each coalition $\mathcal{C}_j$ is assigned to task t_j and $v(\mathcal{CS}) = (\sum_{j=1}^m v(\mathcal{C}_j, t_j))$.

In this work, we study the SCSGA-MF-TS problem as a combined combinatorial optimization problem, where the two objectives of Equation 1 are combined together, and thus we further redefine Equation 1 that minimizes the following:

$$(w_1 \times (v(\mathcal{CS})) - w_2 \times TC(\mathcal{CS}, \mathcal{T})), \quad (2)$$

where w_1 and w_2 is considered as a weight value that normalized both the $v(\mathcal{CS})$ and $TC(\mathcal{CS}, \mathcal{T})$ respectively.

3. Related Works

The SCSGA-MF and SCSGA-MF-TS problems are the two distinct variations of the SCSGA problem. The SCSGA problem aims to generate m disjoint coalitions among n agents, for m tasks, such that the total valuation of the coalition structure is maximized. Extending the SCSGA with d -dimensional features for each agent and task, SCSGA-MF focuses on minimizing the distance between each coalition and its assigned task. Over SCSGA-MF, the SCSGA-MF-TS problem further prioritizes the measure of task satisfaction between each coalition and its assigned task. In order to give a thorough overview, we first discuss the fundamental history of the SCSGA problem, and later we mention its extensions in the domain of MAS.

The SCSGA problem has been studied extensively over the last few years, which includes the development of a diverse range of effective algorithms, such as exact, approximation, anytime, hybrid, and heuristic algorithms. With this, researchers

also focused on solving SCSGA in both small scale and large scale problem instances. In [22], the SCSGA problem was first introduced, devising a dynamic programming method that makes use of preexisting subset memorization. Later on, a hybrid variant of such a dynamic programming algorithm was presented in [21], and [20] proposed an anytime algorithm with an application in a real-world combinatorial game. While all of these methods solve the SCSGA problem for small scale problem instances with up to 30 agents and 20 tasks, a handful of studies have been established that can solve the SCSGA for large scale problem instances in an acceptable time frame. Solving SCSGA problems for large scale problem instances with up to 2000 agents and 50 tasks was the sole emphasis of the SOTA. Regarding this, in order to attain near-optimal solutions on a large scale problem, [19] devised a number of anytime heuristic algorithms. More recently, [25] presented a unique permutation-based index search technique to effectively solve the SCSGA problem for large scale problem instances.

Being the successive variant of SCSGA, the SCSGA-MF problem was introduced in [5], which focuses on the minimization of the distance between each coalition and its corresponding assigned task, ensuring the assignment of the most compatible agents into each task. SCSGA-MF incorporates d -dimensional features on each agent and task, and the value of a coalition is defined as the average of each feature of the agents present in that coalition. Further, the value of a coalition when it is assigned to any task is defined by the distance between the coalition's own value and the feature value of that task. To calculate such distance between a coalition and its assigned task, p number of fixed distance metrics were used and the sum of such p distances was considered as the final distance.

In SOTA, after the SCSGA-MF, the exploration of SCSGA-MF-TS was first proposed in [4]. Being an additional variant of SCSGA, the SCSGA-MF-TS problem aims to optimize two objectives: (i) minimizing the distance between each coalition and its assigned task, and (ii) maximizing the total number of satisfied tasks. In contrast to the SCSGA-MF problem, which solely focuses on reducing the distances between each coalition and its designated task, the SCSGA-MF-TS problem also seeks to maximize the number of satisfied tasks. Also, SCSGA-MF utilizes a variety of distance metrics and their sum to determine the distance between each coalition and its assigned task, but SCSGA-MF-TS only takes the Euclidean distance metric into account for such purpose.

In this work, our proposed combined SCSGA-MF-TS problem follows the formulation of the original SCSGA-MF-TS problem as introduced in [4] and additionally combines the two objectives of SCSGA-MF-TS together, making it a combined combinatorial optimization problem. SCSGA-MF-TS attempts to minimize an overall function by combining the solution value and task satisfaction counts together. While task satisfaction counts indicate the total number of tasks satisfied by their respective coalitions, the solution value in this case refers to the cumulative distance between each coalition and its corresponding assigned task. The characteristics of the two goals of the general SCSGA-MF-TS problem are contained in the overall function of

the combined SCSGA-MF-TS problem. For the SCSGA-MF-TS problem, the CAN search method proposed in [4] tackles the optimization of the solution value and task satisfaction counts independently, whereas our proposed combined SCSGA-MF-TS seeks to optimize these two goals at the same time. We propose and utilize the well-known evolutionary algorithm technique for efficient optimization for such coupled optimization.

The architecture of SCSGA-MF-TS's fundamental hierarchy is comparable to the CSG problem and some MAS advancements. Additionally, other studies pertaining to CSG and MAS have investigated the presence of multi-dimensional features within agents. Those research works motivate us to further formulate the combined SCSGA-MF-TS over SCSGA [22], SCSGA-MF [5], and the general SCSGA-MF-TS problem [4]. It is pertinent to discuss those SOTA works here. Numerous works of literature and a thorough analysis of different algorithms in the context of the CSG problem can be found in [23]. The behavior of MAS with the task allocation problem was described in [6], which presents a detailed analysis of MAS, particularly in relation to task allocation challenges. For the CSG problem, the multi-dimensional vectors on agents are introduced in [16] for a compact representation of coalitional games. Moreover, the skill vectors for coalitional games were introduced in [3] and applied to solve the CSG problem in [2]. Later research extended this with an efficient representation of the search space in [26]. While all these works only focus on incorporating multi-dimensional vectors for the CSG problem, [24] provided a detailed study of task allocation methods applied to the CSG problem with different skill vectors in both homogeneous and heterogeneous environments. In order to provide a compact representation of coalitional games for the CSG problem, [16] introduced the multi-dimensional vectors on agents. Additionally, in [3], the skill vectors for coalitional games were presented, and applied to solve the CSG problem in [2]. Subsequent studies expanded on this by providing an effective search space representation in [26]. While all these works only focus on incorporating multi-dimensional vectors for the CSG problem, [24] provided a detailed study of task allocation methods applied to the CSG problem with different skill vectors in the context of both homogeneous and heterogeneous environments.

Additionally, there are several explorations of CSG, MAS, and their real-world utilization in the literature. A metaheuristic algorithm, such as ant colony optimization, was utilized to solve the vehicle routing problem applied to a real-world problem of transport logistics [18]. In the domain of multi-robot task allocation, [29] introduced a framework for task allocation with energetic as well as spatio-temporal constraints. Solving inverse problems through the application of evolutionary multi-agent computing was introduced in [27]. Moreover, different agent-based frameworks applying various real-world scenarios have been proposed in literature, such as a hierarchical approach on bag of tasks in clouds [10], energy redistribution applying agent-based hybrid algorithm [11], group membership management [17], etc. Moreover, there exist certain developments of MAS applying clustering techniques in the literature, where

the coalitions of agents are considered as the clusters of agents. Few of such developments can be found in [8, 9, 28] for generating feasible coalition, [15] for managing the UAV (unmanned aerial vehicles) and in [1, 7, 14] for coordinating a set of robots assumed to be assigned for different task.

4. Proposed Algorithms

In this section, we propose three algorithms to solve the proposed combined SCSGA-MF-TS problem, with a focus on developing efficient methods to explore exponential search spaces for large scale problem instances within a reasonable time frame, which is a crucial aspect of the SCSGA-MF-TS problem. Among the three methods, the first is the evolutionary optimization approach of Genetic Algorithm, and the other two heuristic approaches are the Hill-Climbing and the Simulated Annealing algorithm for the combined SCSGA-MF-TS problem. Genetic algorithm is a well-known evolutionary optimization algorithm, for which we propose a new encoding of the solutions for SCSGA-MF-TS that will encourage it to effectively explore the search space. The two heuristics we explore are among the top-performing methods in SCSGA [19] and SCSGA-MF-TS problems [4]. Next, to solve large scale combined SCSGA-MF-TS problems, we discuss the genetic algorithm in Section 4.1, followed by the hill-climbing algorithm in Section 4.2, and the simulated annealing algorithm in Section 4.3.

4.1. Genetic Algorithm for the combined SCSGA-MF-TS (GEA)

We devise the Genetic Algorithm for the combined SCSGA-MF-TS (GEA) problem, in which each member of the population is considered as a coalition structure $\mathcal{CS}$, which in turn is considered as a solution of the combined SCSGA-MF-TS problem. GEA calculates the fitness of solutions using Equation 2, and applies the evolutionary strategies of Genetic Algorithm (GA) on them. The evolutionary strategies involve, first, the selection of a set of p random solutions to form a population whose members will undergo evolution. Second, the evolutionary operations of selection, crossover, and mutation are applied to the selected population to generate offspring. Finally, GEA forms a new population containing the offspring along with some of the best solutions from the previous population. This entire procedure is repeated over several iterations until a stopping criterion is reached. When forming a population, we must set some specific constraints to ensure the solutions generated by GEA are valid ones. The constraints are: (i) we set each solution to have at least one agent assigned to each task, and (ii) each agent should be assigned only once to any task. We set a high penalty value for any invalid solution that does not follow these constraints so as to encourage the omission of that solution by GEA. The initial population is also generated, maintaining these constraints. The corresponding pseudo code for GEA is provided in Algorithm 1.

Algorithm 1: Genetic Algorithm (GEA)

Input: Set of n agents $\mathcal{A}$, set of m tasks $\mathcal{T}$, $\mathcal{R}(\mathcal{A})$, $\mathcal{R}(\mathcal{T})$.

Output: Coalition Structure $\mathcal{CS} = \{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_m\}$.

- 1 Generate *init_pop* : p -sized initial population.
 - 2 **while** *stopping condition is not reached* **do**
 - 3 Evaluate all individuals of *init_pop* using Equation 2.
 - 4 Apply selection, crossover, and mutation to generate offspring.
 - 5 Apply a high penalty on the fitness value for any invalid offspring.
 - 6 Update the population and retain the best individuals from the previous generation.
 - 7 **end**
 - 8 Obtain the best solution from the final population.
-

4.2. Permutative Hill-Climbing Algorithm for the combined SCSGA-MF-TS (HCA)

In order to minimize Equation 2, HCA iteratively improves an initial random coalition structure $\mathcal{CS}$, which is considered as a solution of the combined SCSGA-MF-TS problem. HCA generates a random permutation of n agents on set $\mathcal{A}$ for such iterative improvements. In each iteration, HCA sequentially chooses an agent from the random permutation and switches it from its original coalition into all other $m - 1$ coalitions present in the initial $\mathcal{CS}$. This entire iterative process is repeated several times, either until a predetermined user-defined computing budget is achieved or as long as the improvement on the value of Equation 2 is achieved. We want to prevent pathological cases, where the switching of any agent $a_i \in \mathcal{C}_j$ (where $\mathcal{C}_j$ is assigned to any task t_j) may produce an empty coalition, $\mathcal{C}_j = \emptyset$, i.e. when there would be no agent assigned to task t_j . Therefore, we set a constraint, where agent a_i will not be switched away from coalition $\mathcal{C}_j$ if it is the only member of $\mathcal{C}_j$. The corresponding algorithm of HCA is outlined in Algorithm 2.

4.3. Simulated Annealing Algorithm for the combined SCSGA-MF-TS (SAA)

SAA is considered a more computationally efficient and randomized version of the hill-climbing method. Similar to HCA, SAA begins with an initial random coalition structure $\mathcal{CS}$ as a solution to the SCSGA-MF-TS problem and works to improve it iteratively by minimizing the value of Equation 2. To check if it minimizes the value of Equation 2, SAA chooses any random agent in each iteration and moves it from its initial coalition into one of the $m - 1$ alternative coalitions. This iterative process is repeated until a user-specified computing budget is achieved. To prevent from pathological cases, where the switching of any agent $a_i \in \mathcal{C}_j$ (where $\mathcal{C}_j$ is assigned to any task t_j) may produce an empty coalition, $\mathcal{C}_j = \emptyset$, i.e. when there would be no

Algorithm 2: Permutative Hill-Climbing Algorithm (HCA)**Input:** Set of n agents $\mathcal{A}$, set of m tasks $\mathcal{T}$, $\mathcal{R}(\mathcal{A})$, $\mathcal{R}(\mathcal{T})$ **Output:** Coalition Structure $\mathcal{CS} = \{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_m\}$.

```

1 Randomly generate a  $\mathcal{CS}$ .
2 Return the random  $\mathcal{CS}$  from the previous step as an initial solution  $\mathcal{CS}$ .
3 Initialize a random permutation of  $\mathcal{A}$ .
4 while improvement in the value of Equation 2 is obtained, or the computation
   budget has not been exceeded do
5   for each  $a_i$  in the random permutation do
6     if  $a_i \in \mathcal{C}_j$  is not the only member of  $\mathcal{C}_j \in \mathcal{CS}$  then
7       for each  $t_j$ :  $j \leftarrow 1$  to  $m$  do
8          $\mid$  assign  $a_i$  in  $t_j$  if it decreases the value of Equation 2.
9       end
10    end
11  end
12 end

```

agent assigned to task t_j . In that case, we set a constraint, where agent a_i will not be moved away from coalition $\mathcal{C}_j$ to another coalition, if it is the only member of $\mathcal{C}_j$. The pseudo-code of SAA is outlined in Algorithm 3

Algorithm 3: Simulated Annealing Algorithm (SAA)**Input:** Set of n agents $\mathcal{A}$, set of m tasks $\mathcal{T}$, $\mathcal{R}(\mathcal{A})$, $\mathcal{R}(\mathcal{T})$ **Output:** Coalition Structure $\mathcal{CS} = \{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_m\}$.

```

1 Randomly generate a  $\mathcal{CS}$ .
2 Return the random  $\mathcal{CS}$  from the previous step as an initial solution  $\mathcal{CS}$ .
3 while the computation budget has not been exceeded do
4   Select a random  $a_i$ .
5   if  $a_i \in \mathcal{C}_j$  is not the only member of  $\mathcal{C}_j \in \mathcal{CS}$  then
6     Select a random  $t_j$ .
7     Assign  $a_i$  in  $t_j$ , if it decreases the value of Equation 2.
8   end
9 end

```

Computational complexity of the proposed algorithms: we first note that for any $\mathcal{CS}$ to compute the value of $v(\mathcal{CS})$ any algorithm computes the distance between each agent a_i present in a coalition $\mathcal{C}_j \in \mathcal{CS}$ and it's corresponding assigned task $t_j \in \mathcal{T}$ in $O(d)$ time. Hence, for n agents, a total of $O(nd)$ time is required to compute the value of $v(\mathcal{CS})$. Likewise, to compute $TC(\mathcal{CS}, \mathcal{T})$, first we need to calculate $v(\mathcal{C}_j)$ for each coalition $\mathcal{C}_j \in \mathcal{CS}$. The formulation to compute $v(\mathcal{C}_j)$ is defined

in Section 2, as the sum of each corresponding feature of each agent present in $\mathcal{C}_j$. To compute such $v(\mathcal{C}_j)$ across all the $\mathcal{C}_j$ present in $\mathcal{CS}$, $O(nd)$ time is required, as $\mathcal{CS}$ contains all the n agents and each agent has d -dimensional feature. Next, to compute $TC(\mathcal{CS}, \mathcal{T})$ we need to compare the $v(\mathcal{C}_j)$ with the d -dimensional feature vector of the corresponding task t_j ($\mathbf{R}(t_j)$) and this can be done in $O(d)$ time. Hence, for m coalitions on m tasks, it requires $O(md)$ time. Thus, to compute $TC(\mathcal{CS}, \mathcal{T})$ for any $\mathcal{CS}$ total $O(nd) + O(md)$ time is required. Finally, any algorithm calculates the value of $v(\mathcal{CS})$ and $TC(\mathcal{CS}, \mathcal{T})$ for it's generated $\mathcal{CS}$ in $O(nd) + O(nd) + O(md) \equiv O(nd + md)$ time.

The GEA explores p solutions in each iteration and as we have n agents so each solution size is equal to n . So, GEA operates on an n -sized solution with total p solutions in each iteration and completes the computation in $O(pn)$ time. Additionally, according to the complexity of calculating $v(\mathcal{CS})$ and $TC(\mathcal{CS}, \mathcal{T})$ for any $\mathcal{CS}$, GEA calculates the fitness value of each solution in $O(nd + md)$ time. Hence, in each iteration, GEA completes its execution in $O(p \times (nd + md)) = O(npd + mpd)$ time.

The HCA algorithm moves each agent a_i to all other $(m - 1)$ coalitions, and computes $v(\mathcal{CS})$ and $TC(\mathcal{CS}, \mathcal{T})$ for each of these possible assignments. These operations can be done in $O(m \times (nd + md))$ time, where $O(nd + md)$ time is required to compute $v(\mathcal{CS})$ and $TC(\mathcal{CS}, \mathcal{T})$. Hence, for n agents HCA completes its execution for one iteration in $O(n \times (m \times (nd + md))) = O(nm \times (nd + md)) = O(n^2md + nm^2d)$. **In this same manner, SAA** switches any agent a_i into any one task only and computes $v(\mathcal{CS})$ and $TC(\mathcal{CS}, \mathcal{T})$ only once in $O(nd + md)$ time. Hence, SAA incurs $O(nd + md)$ time to complete its execution for one iteration.

5. Experimental Evaluation

In this section, we conduct suitable experiments to empirically compare the performance of the proposed algorithms, to determine which one empirically excels over the others. We aim to investigate three distinct research questions through various experiments. Our primary research question for such an investigation is to determine which algorithm can find coalition structures with higher task satisfaction counts while minimizing the solution value. The second question we intend to investigate is which specific algorithm can optimize both the task satisfaction counts and solution values within a single optimization function, when SCSGA-MF-TS is considered as a combined combinatorial optimization problem. We also investigate whether the proposed algorithms can produce a good solution for large-scale problem instances in an acceptable amount of time. To measure the performance of any proposed algorithm, we choose the task satisfaction count and solution value as the performance evaluation metrics. Also, we report the average execution time of each algorithm on large scale problem instances.

To the best of our knowledge, this is the first time the evolutionary algorithm GEA is being explored for the SCSGA-MF-TS problem. However, HCA and SAA are adapted from the SOTA of the SCSGA [19] and SCSGA-MF-TS [4] problems. Thus,

from our empirical studies, we will be able to determine how well the SCSGA-MF-TS SOTA approaches of HCA and SAA perform when the SCSGA-MF-TS problem is viewed as a combined combinatorial optimization problem utilizing Equation 2. The studies also provide us with suitable baseline performances against which the performance of GEA can be compared.

5.1. Experimental Setup

The empirical studies are carried out on datasets generated using standard benchmark procedures. The parameter settings for the competing approaches and the metrics to assess their performances are presented next.

5.1.1. Dataset Description

For all the experiments, the performances of the contending methods are evaluated on the data generated from the benchmark data distribution of Uniform Probability Distribution (UPD). Such benchmark distribution have been used previously to sample the feature values for agents and tasks in SCSGA-MF problem [5] and SCSGA-MF-TS problem [4] as well as to generate the valuation of any coalition assigned to any task in SCSGA problem [19]. Hence, considering the similar setup of the SCSGA-MF-TS problem [4], in this work, all the agents' and tasks' features are drawn using UPD, additionally specifying different levels of difficulty. Because when the tasks and agents have a similar range of feature values, the agents have a better likelihood of satisfying any task. However, it should get harder to satisfy every task, though, when the task feature values are far higher than the agent features. So, in this work, when sampling from the UPD, we additionally employ different variances on the dataset to represent these challenges.

Following such manner, we generate each dataset, with agents a_i , $i = 1, \dots, n$, and tasks t_j , $j = 1, \dots, m$, where each feature component $r_h(a_i)$ and $r_h(t_j)$, for $h = 1, \dots, d$, is drawn in the following manner:

- **UPD:** $r_h(a_i) \sim \mathcal{U}(l_1, l_2)$, where, $l_1 = 0$ & $l_2 = 1$.
- **UPD:** $r_h(t_j) \sim \mathcal{U}(l_1, l_2)$, where, $l_1 = 0$ & $l_2 = g_j \times \lceil \frac{n}{m} \rceil$

In this work, we utilize three potential data variances, denoted by setting $g_j := \{0.5, 1, 1.5\}$, to reflect three levels of difficult problem settings. We anticipate that problem settings with lower values of g_j will be easier, whereas those with greater values of g_j will be more difficult. This approach is justified by the need to verify the proposed algorithms' performance in various agent-task distributions. Additionally, we create five datasets for every data variance setting of the UPD, for a total of fifteen datasets, in order to produce reliable estimates of the performance.

5.1.2. Experiment Protocol

Using each generated dataset, we conduct several experiments on each proposed algorithm and derive the coalition structure $\mathcal{CS}$ from each competing method. Finally, the resultant $\mathcal{CS}$ is assessed using its solution value $v(\mathcal{CS})$ and task satisfaction count

$TC(\mathcal{CS}, \mathcal{T})$. On these two metrics, by calculating the average rank of each proposed method and performing a paired hypothesis test (using the Wilcoxon signed-rank test), we are able to summarize the different experimental results. For the pairwise hypothesis test, the alternate hypothesis $\mathcal{H}_1$ proposes a significant difference between any two methods, while the null hypothesis $\mathcal{H}_0$ believes that the outcomes of any two methods are chosen from the same distribution. Subjected to the Wilcoxon signed-rank test, we consider a 95% confidence interval, which corresponds to a p -value of 0.05 for every method pair.

We note that the prior researches were carried out on smaller problem sizes, such as $n = 1000$ agents and $m = 50$ tasks [19] or $n = 2000$ agents and $m = 20$ tasks [25], for the choices of n and m . Also, $n = 6000$ agents and $m = 120$ tasks were used to investigate the SCSGA-MF-TS problem [4]. In this work, we create the problem instances with the number of agents ranging from $n=1000$ to 10000 and the number of tasks ranging from $m=20$ to 200 in order to guarantee that the empirical research is carried out on large-scale problem instances. Additionally, we set the value of $d = 10$ (number of features) on each agent and task across all the experiments.

In this work, one of our research contributions is to study the SCSGA-MF-TS problem as a combined combinatorial optimization problem, utilizing Equation 2. To conduct the experiments, for Equation 2 the value of $TC(\mathcal{CS}, \mathcal{T})$ and $v(\mathcal{CS})$ can be obtained by the final coalition structure $\mathcal{CS}$ generated by the each proposed method. Now, we estimate the value of w_1 and w_2 by employing the normalization method, formally: for any $\mathcal{CS}$, $w_1 = \frac{v(\mathcal{CS}) - \min_v(\mathcal{CS})}{\max_v(\mathcal{CS}) - \min_v(\mathcal{CS})}$ and $w_2 = \frac{TC(\mathcal{CS}, \mathcal{T}) - \min_TC(\mathcal{CS}, \mathcal{T})}{\max_TC(\mathcal{CS}, \mathcal{T}) - \min_TC(\mathcal{CS}, \mathcal{T})}$, where, $\min_v(\mathcal{CS})$ and $\max_v(\mathcal{CS})$ denotes minimum and maximum value of $v(\mathcal{CS})$ obtained over all observations respectively. Likewise, $\min_TC(\mathcal{CS}, \mathcal{T})$ and $\max_TC(\mathcal{CS}, \mathcal{T})$ denotes the minimum and maximum value of $TC(\mathcal{CS}, \mathcal{T})$ obtained over all observations respectively. Now, as we calculating the distance to obtain $v(\mathcal{CS})$ hence, we set $\min_v(\mathcal{CS}) = 0$ and also for $TC(\mathcal{CS}, \mathcal{T})$, we set $\min_TC(\mathcal{CS}, \mathcal{T}) = 0$ and $\max_TC(\mathcal{CS}, \mathcal{T}) = m = \text{total number of tasks}$. Additionally, we construct a $n \times m$ distance matrix between each agent and task, where any $(i, j)^{th}$ value denotes the euclidean distance between the feature values of agent $a_i \in \mathcal{A}$ and $t_j \in \mathcal{T}$, formally: $d_{\mathcal{M}}(\mathbf{R}(a_i), \mathbf{R}(t_j))$. From Definition 1, we assume that the maximum value of $v(\mathcal{CS})$ for any coalition structure will be, the sum of top n largest values of that $n \times m$ distance matrix and we denote this value as n_lar . Finally, to calculate w_1 , we set the value of $\max_v(\mathcal{CS})$ is equal to n_lar .

For GEA, we set the simulated binary crossover operator with a crossover probability of 0.9, and the polynomial mutation operator with a mutation probability of 0.3. A random selection is done to randomly select solutions that will undergo the evolutionary operations. The stopping condition for GEA is set to 500 iterations, and the initial population is randomly sampled with a size of $p=100$. The computation budget for HCA and SAA is determined by measuring the maximum execution time of GEA across all generated datasets.

An AMD Ryzen 9 7900X CPU with 128 GB of RAM was used for all the experi-

ments. <https://github.com/Tuhin-kr/Combined-SCSGA-MF-TS> contains the implementations of all algorithms in the Python language.

5.2. Evaluation of Contending Methods on SCSGA-MF-TS Problem

In this section, we analyze the performance of the proposed algorithms when the solution values and the task satisfaction counts are measured separately according to the general SCSGA-MF-TS problem [4]. For this experiment, we measure the solution values and task satisfaction counts of any solution (an ordered $\mathcal{CS}$) before combining them using Equation 2. Here, we perform three sets of experiments varying the number of agents and tasks.

First, Table 1 represents the experimental results obtained by GEA, HCA, and SAA having 1000 agents and 20 tasks. From Table 1 we observe that in all cases, GEA outperforms HCA and SAA in terms of the task satisfaction counts. From the average ranks, we note that GEA achieves the lowest average rank, becoming the best method over HCA and SAA in terms of task satisfaction counts, whereas SAA is the second best and HCA is the third best method. Specifically, in terms of task satisfaction counts GEA satisfies $(\frac{19}{20} \times 100) = 95\%$ tasks for the lowest data variance. In such a case, similarly, HCA is able to satisfy a maximum of 70% tasks, and SAA is able to satisfy at most 10% tasks for the lowest data variance. For the second highest data variance, i.e., $g_j = 1$, the GEA is able to satisfy at most 25% tasks, and the HCA and SAA satisfy a maximum of 5% and 15% tasks, respectively. For the last case, when the data variance is 1.5, GEA satisfies a maximum of 15% tasks, while HCA fails to satisfy any tasks, and SAA satisfies at most 5% tasks. This observation reveals that GEA performs better than HCA and SAA for both the easy and the challenging cases. When the data variance is low, the performance of SAA is poorer than HCA. In fact, HCA fails to satisfy any task in most cases when the data variance is 1 and 1.5. It is to be noted that for the easy problem setting, i.e., when the data variance is 0.5, GEA is able to satisfy most of the tasks for every problem instance. When the data variance is increased, the performance of GEA decreases from 95% to 25% for $g_j = 1$, and to 15% for $g_j = 1.5$. We also observe that in terms of solution value, SAA generates lower solution values in every case, irrespective of data variances over GEA and HCA. This can also be visualized from the average ranks, where SAA achieves the lowest average rank in terms of solution values. By comparison, GEA is the second best and HCA is the worst, according to the average ranks. Additionally, according to the hypothesis test in Table 1, we observe that the performance of GEA is significantly different from HCA and SAA in terms of both task satisfaction counts and solution values.

Next, Table 2 shows the experimental results obtained by GEA, HCA, and SAA over problem instances containing 5000 agents and 100 tasks. From Table 2, we observe that GEA outperforms HCA in terms of task satisfaction counts for every case. GEA outperforms SAA only when the data variance is low, i.e., $g_j = 0.5$. For data variance, $g_j = 1$ and $g_j = 1.5$, SAA outperforms GEA and HCA in terms of task

Table 1

The task satisfaction counts $TC(\mathcal{CS}, \mathcal{T})$ and solution values $v(\mathcal{CS})$ observed for coalition structures generated by GEA, HCA, and SAA for 1000 agents and 20 tasks

Datasets	Task Satisfaction Counts			Solution Values		
	GEA	HCA	SAA	GEA	HCA	SAA
$r_h(a_i) \sim \mathcal{U}(0, 1),$ $r_h(t_j) \sim \mathcal{U}(0, 0.5[\frac{n}{m}])$	18	8	2	4.22E+04	4.39E+04	2.77E+04
	19	13	1	4.39E+04	4.42E+04	3.38E+04
	19	10	1	4.45E+04	4.60E+04	3.31E+04
	18	11	1	4.17E+04	4.40E+04	3.29E+04
	19	14	1	4.28E+04	4.36E+04	3.46E+04
$r_h(a_i) \sim \mathcal{U}(0, 1),$ $r_h(t_j) \sim \mathcal{U}(0, [\frac{n}{m}])$	4	0	1	7.20E+04	8.69E+04	5.34E+04
	5	0	1	7.98E+04	9.06E+04	4.68E+04
	4	0	3	7.92E+04	8.67E+04	7.00E+04
	3	1	1	7.35E+04	8.75E+04	6.11E+04
	4	0	1	8.05E+04	8.80E+04	6.61E+04
$r_h(a_i) \sim \mathcal{U}(0, 1),$ $r_h(t_j) \sim \mathcal{U}(0, 1.5[\frac{n}{m}])$	2	0	1	1.18E+05	1.33E+05	1.06E+05
	3	0	1	1.20E+05	1.29E+05	9.86E+04
	3	0	1	1.23E+05	1.38E+05	9.64E+04
	1	0	1	1.17E+05	1.35E+05	9.97E+04
	3	0	1	1.18E+05	1.26E+05	9.59E+04
Avg. Ranks	1.0	2.6	2.27	2.0	3.0	1.0
GEA	Hyp. test	—	$\mathcal{H}_1$	—	$\mathcal{H}_1$	$\mathcal{H}_1$
	p-value	—	0.00064	—	0.00065	6.10E-05

satisfaction counts. However, the average ranks reveal that both the GEA and SAA achieve the same rank, as both methods achieve similar performances considering overall observations on the task satisfaction counts. Specifically, for the lowest data variance, i.e., when $g_j = 0.5$, GEA satisfies 90% tasks, while HCA is able to satisfy a maximum of 67% tasks, and SAA is able to satisfy at most 33% tasks. For the second highest data variance, i.e., $g_j = 1$, the GEA is able to satisfy at most 14% tasks, but SAA satisfies a maximum of 22% tasks, and for this case, HCA fails to satisfy any of the tasks. For the last case, when the data variance is $g_j = 1.5$, GEA satisfies a maximum of 4% tasks, and SAA satisfies at most 15% tasks, whereas HCA again fails to satisfy any tasks. This observation reveals that for lower data variance, GEA outperforms both HCA and SAA, and has comparable performance to SAA in the easy and the hard cases.

As in Table 1, similar conclusions can also be drawn from the observations in Table 2. In the easy problem setting, i.e., when the data variance is 0.5, GEA is able to satisfy most of the tasks in every case. But when the data variance is increased, the performance of GEA decreases from 90% to 14% (for $g_j = 1$), and finally to 4% (for $g_j = 1.5$). The performance trade-off across the data variances is lower for SAA, as SAA satisfies 31% tasks on the lower data variance, and this performance decreases from 31% to 22% (for $g_j = 1$) to 15% (for $g_j = 1.5$). After satisfying 67% tasks for

Table 2

The task satisfaction counts $TC(\mathcal{CS}, \mathcal{T})$ and solution value $v(\mathcal{CS})$ observed for coalition structures generated by GEA, HCA, and SAA for 5000 agents and 100 tasks.

Datasets	Task Satisfaction Counts			Solution Values		
	GEA	HCA	SAA	GEA	HCA	SAA
$r_h(a_i) \sim \mathcal{U}(0, 1),$ $r_h(t_j) \sim \mathcal{U}(0, 0.5 \lceil \frac{n}{m} \rceil)$	89	58	31	2.14E+05	2.17E+05	1.73E+05
	90	67	33	2.19E+05	2.21E+05	1.76E+05
	89	64	32	2.17E+05	2.21E+05	1.73E+05
	86	61	30	2.20E+05	2.23E+05	1.80E+05
	90	65	31	2.17E+05	2.20E+05	1.83E+05
$r_h(a_i) \sim \mathcal{U}(0, 1),$ $r_h(t_j) \sim \mathcal{U}(0, 1.5 \lceil \frac{n}{m} \rceil)$	11	0	20	4.19E+05	4.44E+05	3.61E+05
	9	0	21	4.32E+05	4.48E+05	3.62E+05
	11	0	19	4.29E+05	4.51E+05	3.57E+05
	9	0	20	4.23E+05	4.40E+05	3.56E+05
	14	0	22	4.25E+05	4.41E+05	3.52E+05
$r_h(a_i) \sim \mathcal{U}(0, 1),$ $r_h(t_j) \sim \mathcal{U}(0, \lceil \frac{n}{m} \rceil)$	3	0	15	6.23E+05	6.61E+05	5.18E+05
	4	0	13	6.44E+05	6.74E+05	5.62E+05
	2	0	14	6.34E+05	6.68E+05	5.39E+05
	3	0	14	6.46E+05	6.77E+05	5.58E+05
	2	0	15	6.24E+05	6.68E+05	5.52E+05
Avg. Ranks	1.67	2.67	1.67	2.0	3.0	1.0
GEA	Hyp. test	—	$\mathcal{H}_1$	—	$\mathcal{H}_1$	$\mathcal{H}_1$
	p-value	—	0.00064	—	0.00064	0.00065

the lower data variance, the performance of HCA drastically decreased to 0% for the rest of the cases. We further observe that for $g_j = 0.5$, the performance of GEA is approximately 34% better than SAA, as for this case GEA and SAA satisfies 90% and 31% of the tasks respectively, hence the trade-off is $\frac{31}{90} \times 100 \approx 34\%$. Likewise, we also observe that the performance of SAA is 63% and 26% better than GEA when $g_j = 1$ and $g_j = 1.5$, respectively. However, in terms of solution value, SAA generates lower solution values compared to GEA and HCA in every case, irrespective of the data variances. Hence, SAA achieves the lowest average rank, and is thus the top performing method in terms of solution values, whereas GEA is the second best and HCA is the third best method. Moreover, the hypothesis tests in Table 2 reveal that, the performance of GEA is significantly different over HCA and SAA in terms of the solution values, whereas in terms of task satisfaction counts the performance of GEA and SAA are comparable and the performance of GEA is significantly different over HCA.

Thirdly, in Table 3 we show the experimental results obtained by GEA, HCA, and SAA having 10000 agents and 200 tasks. From Table 3, for low data variance ($g_j = 0.5$), we observe that GEA clearly outperforms HCA and SAA in terms of obtaining higher task satisfaction counts. For the other two cases of $g_j = 1$ and $g_j = 1.5$, in terms of task satisfaction counts, SAA outperforms GEA and HCA. Specific observation reveals that, for the lowest data variance, i.e., when $g_j = 0.5$, GEA is able to satisfy at most 90.50% tasks, while HCA is able to satisfy at most

70% tasks, and in this case, SAA satisfies 44.50% tasks. For the second case, when $g_j = 1$, GEA satisfies 8% tasks, while SAA satisfies at most 19% tasks, and here HCA fails to satisfy any task except for one case, where it satisfies only 0.5% of the tasks. For the third case, when $g_j = 1.5$, GEA is able to satisfy a maximum of 2% of tasks, and SAA is able to satisfy at most 5.50% of tasks. In this case, again, HCA fails to satisfy any of the tasks. The observations of Table 3 reveal that GEA is able to satisfy the maximum number of tasks over HCA and SAA only when the data variance is lowest. Also, for the lowest data variance, HCA performs better than SAA in terms of task satisfaction counts. But, when the data variance is increased to 1 and 1.5, then SAA is able to satisfy the maximum number of tasks. HCA was unable to satisfy any task in most cases for $g_j = 1$ and $g_j = 1.5$. The average ranks reveal that both the GEA and SAA achieve the same rank, as both of these methods achieve similar performances overall for the task satisfaction counts. Specifically, we observe a better performance of SAA over GEA in the last two cases, but we can conclude that for $g_j = 0.5$ the performance of GEA is approximately 50% better than SAA, as for this case GEA and SAA satisfy 90.50% and 44.50% tasks respectively, hence the trade-off is $\frac{44.50}{90.50} \times 100 \approx 50\%$. Considering this logical aspect, we observe that the performance of SAA is 42% and 36% better than GEA when $g_j = 1$ and $g_j = 1.5$, respectively. From Table 3 we also observe that in terms of solution values, SAA generates lower solution values in every case, irrespective of the data variances over GEA and HCA. Hence, SAA achieves the lowest average rank, becoming the best performing method in terms of solution values, whereas GEA is the second best and HCA is the third best method. An identical nature of hypothesis test is observed in Table 3 in comparison with Table 2. Hence, in Table 3, the hypothesis tests summarize that the performance of GEA is significantly different over HCA and SAA in terms of the solution values, whereas in terms of task satisfaction counts, the performance of GEA and SAA are comparable, and the performance of GEA is significantly different over HCA.

A note on overall observations:

From Tables 1, 2, and 3, our observations draw the conclusion that when the number of agents and tasks are lower (within $n = 1000$ and $m = 20$) GEA achieves better performance in terms of task satisfaction counts over HCA and SAA. When the number of agents and tasks is increased (for $n/m = 5000/100$ and $n/m = 10000/200$), GEA achieves better performances in terms of task satisfaction counts when the data variance is the lowest, i.e., $g_j = 0.5$. On the other hand, SAA outperforms GEA in terms of task satisfaction counts for cases of higher data variances (when $g_j = 1$ and $g_j = 1.5$), and for $n/m = 5000/100$, and $n/m = 10000/200$. HCA performs better than SAA in terms of task satisfaction counts for the case of the lowest data variance $g_j = 0.5$ only, irrespective of the number of agents and tasks. On the other hand, in terms of solution values, SAA always outperforms GEA and HCA for all cases of Tables 1, 2, and 3.

From all such observations, we recommend utilizing GEA for the SCSGA-MF-TS problem, as in terms of task satisfaction counts GEA outperforms all the contending

Table 3

The task satisfaction counts $TC(\mathcal{CS}, \mathcal{T})$ and solution value $v(\mathcal{CS})$ observed for coalition structures generated by GEA, HCA, and SAA for 10000 agents and 200 tasks.

Datasets	Task Satisfaction Counts			Solution Values		
	GEA	HCA	SAA	GEA	HCA	SAA
$r_h(a_i) \sim \mathcal{U}(0, 1),$ $r_h(t_j) \sim \mathcal{U}(0, 0.5 \lceil \frac{n}{m} \rceil)$	169	123	89	4.35E+05	4.38E+05	3.98E+05
	181	126	85	4.29E+05	4.31E+05	3.81E+05
	171	125	88	4.40E+05	4.43E+05	4.01E+05
	179	140	85	4.33E+05	4.37E+05	3.90E+05
	174	126	84	4.31E+05	4.37E+05	3.93E+05
$r_h(a_i) \sim \mathcal{U}(0, 1),$ $r_h(t_j) \sim \mathcal{U}(0, \lceil \frac{n}{m} \rceil)$	16	1	38	8.69E+05	8.92E+05	8.00E+05
	12	0	36	8.57E+05	8.86E+05	7.94E+05
	13	0	36	8.64E+05	8.87E+05	8.01E+05
	14	0	38	8.54E+05	8.87E+05	8.04E+05
	15	0	37	8.61E+05	8.79E+05	7.99E+05
$r_h(a_i) \sim \mathcal{U}(0, 1),$ $r_h(t_j) \sim \mathcal{U}(0, 1.5 \lceil \frac{n}{m} \rceil)$	4	0	7	1.29E+06	1.35E+06	1.23E+06
	2	0	8	1.31E+06	1.36E+06	1.25E+06
	3	0	11	1.29E+06	1.34E+06	1.22E+06
	3	0	6	1.27E+06	1.33E+06	1.21E+06
	3	0	11	1.31E+06	1.36E+06	1.24E+06
Avg. Ranks	1.67	2.67	1.67	2.0	3.0	1.0
GEA	Hyp. test	—	$\mathcal{H}_1$	—	$\mathcal{H}_1$	$\mathcal{H}_1$
	p-value	—	0.00064	—	0.00064	0.00064

methods for the lower range of number of agents and tasks. While for the higher number of agents and tasks, GEA again outperforms all the methods for the lowest data variance. Even for the higher number of agents and tasks, the overall performance of GEA is comparable to that of SAA, where GEA and SAA are achieving the same rank. Hence, in terms of task satisfaction counts a diverse nature of the performance of SAA is observed, as in a few cases SAA outperforms GEA, while for the lower number of agents and tasks, as well as for the lowest data variance on higher number of agents and tasks, SAA is not able to provide better performance than GEA. Also, the cases where SAA outperforms GEA are not significantly marked when the overall performance is measured utilizing average ranks, as overall performance, GEA achieves the identical performance as SAA. Thus, a constant nature in the performance of GEA is observed in terms of task satisfaction counts: GEA either outperforms SAA or achieves the same performance as of SAA while considering the overall observations. Moreover, in terms of task satisfaction counts, GEA outperforms HCA in all the cases, irrespective of the number of agents and tasks, as well as data variances. Hence, considering all such aspects, we recommend utilizing GEA for the SCSGA-MF-TS problem, as in SCSGA-MF-TS, we are prioritizing the importance of task satisfaction counts over solution values. While the next Section 5.3 provides the final recommendation for the combined SCSGA-MF-TS problem, where the performance of any algorithm is measured considering both task satisfaction counts and solution values simultaneously by combining them together

It is to be noted that, GEA, HCA, and SAA all are set to run upto a fixed amount of time (let say t seconds), and within this t seconds GEA explores 100 solutions per iteration and a total of 50000 solutions, as we set population size to be equal to 100 with termination condition of 500 iterations. While HCA explores m solutions and SAA explores only one solution per iteration.

A note on execution time of each proposed method: we measure the maximum execution time of GEA for every data set. We observe that GEA generates the final results within at most 19 seconds for 1000 agents/20 tasks, 138 seconds for 5000 agents/100 tasks, and 310 seconds for 10000 agents/200 tasks. Hence, these time limits were set for HCA and SAA as their computation budget. These settings validate the performance of HCA and SAA over GEA on an identical time frame, where we can easily compare the solution quality of each algorithm after they have explored the search space for an equal amount of time.

5.3. Evaluation of Contending Methods on SCSGA-MF-TS as a Combined Combinatorial Optimization Problem

After the analysis of the performance of each proposed method on the general SCSGA-MF-TS problem, we further test the performances of the contending methods on the SCSGA-MF-TS problem following Equation 2 as discussed in Section 2. This experiment reveals the analysis of contending methods when they are applied to a combined combinatorial optimization problem. Hence, for this experiment, we define that any resulting coalition structure $\mathcal{CS}$ generated by any of the proposed methods is evaluated using Equation 2. Here, the **objective value** (ov) of any $\mathcal{CS}$ is defined as follows:

$$ov(\mathcal{CS}) = (w_1 \times (v(\mathcal{CS})) - w_2 \times TC(\mathcal{CS}, \mathcal{T}))$$

In Figure 1, we observe the values of $ov(\mathcal{CS})$ obtained by each of the proposed methods across all data sets. From Figure 1 (a)-(c), we observe that for each case, the GEA generates a lower objective value than HCA and SAA. The performance of HCA and SAA is similar in most cases. Hence, considering these aspects, we observe that GEA outperforms SAA and HCA across all the cases, generating the lowest objective value. Therefore, we finally recommend utilizing GEA when the SCSGA-MF-TS problem is considered as a combined combinatorial optimization problem.

5.4. Ablation Study

The experimental evaluation discussed in Section 5.2 and 5.3 validates the superiority and applicability of the GEA algorithm over other adapted heuristics to solve general SCSGA-MF-TS as well as combined SCSGA-MF-TS problems. However, in this section, we seek to further analyze the performance of the other well-known alternatives of the GEA algorithm on the combined combinatorial optimization problem. Besides the genetic algorithm, there exist different types of evolutionary algorithms that may outperform for such a combined combinatorial optimization problem. One

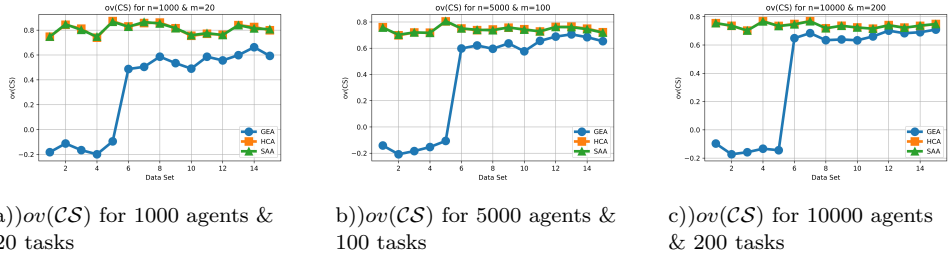


Figure 1. Comparison of objective values for different contending methods.

of such well-known and widely utilized algorithms is the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) algorithm. The CMA-ES was first proposed in [13] and further optimized by [12] in terms of computational complexity. CMA-ES is an efficient evolutionary algorithm for optimizing different challenging non-linear and non-convex problems in the continuous domain and thus becomes an effective tool for solving the proposed combined combinatorial optimization problem as discussed in Section 2.

As in this study, we solely focus on addressing a combined combinatorial optimization problem; hence, we seek to adapt the CMA-ES algorithm to be applicable for our proposed combined combinatorial optimization problem, subjected to the optimization of the Equation 2. For our proposed combined combinatorial optimization problem, the working principle of CMA-ES is almost identical to that of the GEA algorithm described in Algorithm 1. CMA-ES first starts with a randomly selected initial population of candidate solutions and iteratively aims to find the best solution by maintaining a covariance matrix that controls the exploration and exploitation of the search space. Next, the solutions are evaluated using the Equation 2 and top-performing parents are selected for the next generation. It is to be noted that, like step 5 of Algorithm 1, we also apply a high penalty on the fitness value for any invalid solution to omit it from the search space. Then CMA-ES employs a recombination step followed by the mutation strategy to combine the parents for new offspring. Unlike the genetic algorithm, CMA-ES additionally adapts the covariance matrix to learn the optimal step size and direction for the next generation. Thus, the generated new offspring replaces the old population, and the process repeats until the termination condition or the maximum number of iterations is reached.

To maintain a similar structure to that of GEA, we set the population size to 100 and the termination condition to 500 iterations, for CMA-ES also. Hence, like GEA, CMA-ES also explores 100 solutions per iteration and a total of 50000 solutions. While we already described in Section 5.2 that HCA and SAA are set to run up to a fixed amount of time equal to the maximum execution time of GEA for every data set, exploring m solutions and one solution per iteration by HCA and SAA, respectively. We also note that, like GEA, CMA-ES also incurs $(n_{pd} + m_{pd})$

computational complexity to complete its execution in each iteration. Additionally, CMA-ES incurs (n^2p) complexity for multiplication of covariance matrix factorization on sampling the offspring and (n^2) complexity for updating the covariance matrix per iteration. Finally, in each iteration, CMA-ES completes its execution in $O(npd + mpd) + O(n^2p) + O(n^2) = O(npd + mpd + n^2p)$ time.

Now, in Figure 2, we represent the comparative study on the performance of our proposed algorithms against the CMA-ES algorithm in terms of the objective value of the final CS generated by each method.

From Figure 2(a), we observe that the performance of GEA is mostly identical to CMA-ES in most cases, while in a few cases either GEA outperforms CMA-ES or CMA-ES outperforms GEA. Next from Figure 2(b), we observe that GEA consistently outperforms CMA-ES for the lowest variance of data, and for the two highest variances of data, the performance of GEA is mostly identical to CMA-ES in most cases. While such same pattern of the performances is also observed in Figure 2(c).

From such observations, we conclude that when the number of agents and tasks is less ($n = 1000, m = 20$), in that scenario, CMA-ES outperforms GEA in very few cases. Otherwise, for the rest of the cases (i.e., for fewer as well as larger numbers of agent-task), the GEA either outperforms CMA-ES, or performs identically in nature with that of CMA-ES. Also, from the discussion of the computational complexity, we observe a higher computational complexity of CMA-ES than that of GEA. Here, over GEA, CMA-ES incurs additional n^2p complexity, which limits the ability of CMA-ES to solve our proposed combined combinatorial optimization problem for large-scale inputs within a reasonable time frame compared to GEA. It is to be noted that, in Section 1, we already discussed that one of our key objectives of this study is to solve the proposed problem for large-scale inputs within a reasonable time frame. Because the combined SCSGA-MF-TS problem also incurs exponential growth of search space with the increment of the number of agents and tasks, which makes this problem hard to solve, similar to SCSGA, SCSGA-MF, and SCSGA-MF-TS.

Hence, from such a conclusion, comparing the GEA and CMA-ES in terms of objective value and computational complexity, we recommend utilizing the GEA for solving the proposed combined combinatorial optimization problem on large-scale input within a reasonable time frame. However, CMA-ES always outperforms HCA and SAA in every case, like GEA. Hence, adapting CMA-ES for our proposed problem can be an efficient choice as an alternative to HCA and SAA.

6. Conclusion

The general SCSGA-MF-TS problem considers optimizing two objectives: maximization of task satisfaction counts and minimization of the solution value, in an iterative manner. While in each iteration, either of the objectives is optimized, but not both simultaneously. Hence, in this work, we study the SCSGA-MF-TS problem as a combined combinatorial optimization problem, where we combine the two objectives of the general SCSGA-MF-TS problem. This provides a unified framework for optimizing

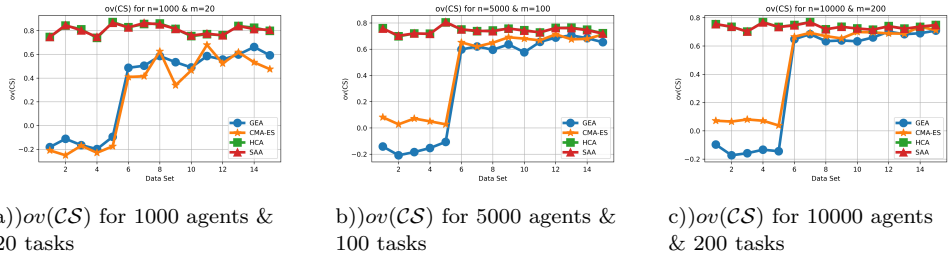


Figure 2. Comparison of objective values for CMA-ES, GEA, HCA, and SAA algorithms.

both objectives simultaneously, thus making the SCSGA-MF-TS a more real-world-adapted, human-interpretable problem. Following the general success of evolutionary algorithms in optimization problems, we propose adapting the genetic algorithm for solving such a combined combinatorial optimization problem. We additionally propose hill climbing and simulated annealing algorithms, which are adapted from the SOTA of the SCSGA-MF-TS problem. To the best of our knowledge, for the first time in the literature, in this study, we explore the applicability of such algorithms in a combined combinatorial optimization problem. We study the performance of all three proposed algorithms on the general SCSGA-MF-TS problem, as well as on the SCSGA-MF-TS problem considered as a combined combinatorial optimization problem. We experimentally provide the recommendation of utilizing the genetic algorithm for the general SCSGA-MF-TS problem as well as for the combined combinatorial optimization version of the SCSGA-MF-TS problem. All the proposed algorithms are capable of handling large-scale problem instances within a reasonable time frame, to explore the exponential search space of SCSGA-MF-TS. Even if this work provides a broad analysis of the combined SCSGA-MF-TS problem, future research could explore identifying mitigation techniques for unsatisfied tasks, especially those that can be incorporated into each proposed algorithm. Additionally, developing an extensive study on the applicability of different evolutionary algorithms that may outperform the genetic algorithm on combined combinatorial optimization problems could offer a promising direction for future research.

References

- [1] Alitappeh R.J., Jeddisaravi K.: Multi-robot exploration in task allocation problem, *Applied Intelligence*, vol. 52, pp. 2189–2211, 2022.
- [2] Bachrach Y., Meir R., Jung K., Kohli P.: Coalitional Structure Generation in Skill Games. In: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 24, pp. 703–708, 2010.
- [3] Bachrach Y., Rosenschein J.S.: Coalitional Skill Games. In: *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, pp. 1023–1030, 2008.

- [4] Biswas T.K., Gupta A., Changder N., Das S., Taguelmimt R., Aknine S., Dutta A.: Compact agent neighborhood search for the SCSGA-MF-TS: SCSGA with multi-dimensional features prioritizing task satisfaction, *Information Sciences*, vol. 706, p. 122021, 2025.
- [5] Biswas T.K., Gupta A., Changder N., Taguelmimt R., Aknine S., Chattopadhyay S., Dutta A.: Coalition Formation for Task Allocation Using Multiple Distance Metrics (Student Abstract), *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 23443–23444, 2024.
- [6] Dorri A., Kanhere S.S., Jurdak R.: Multi-Agent Systems: A Survey, *IEEE Access*, vol. 6, pp. 28573–28593, 2018.
- [7] Elango M., Nachiappan S., Tiwari M.K.: Balancing task allocation in multi-robot systems using K-means clustering and auction based mechanisms, *Expert Systems with Applications*, vol. 38, pp. 6486–6491, 2011.
- [8] Farinelli A., Bicego M., Bistaffa F., Ramchurn S.D.: A hierarchical clustering approach to large-scale near-optimal coalition formation with quality guarantees, *Engineering Applications of Artificial Intelligence*, vol. 59, pp. 170–185, 2017.
- [9] Farinelli A., Bicego M., Ramchurn S., Zucchelli M.: C-Link: A hierarchical clustering approach to large-scale near-optimal coalition formation. In: *IJCAI*, 2013.
- [10] Funika W., Mazurek K., Kruczkowski W.: Agent-based hierarchical approach for executing bag-of-tasks in clouds, *Computer Science*, vol. 15(1), p. 35, 2014.
- [11] Godzik M.: Energy redistribution in autonomous hybridization of agent-based computing, *Computer Science*, vol. 22(3), 2021.
- [12] Hansen N., Müller S.D., Koumoutsakos P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES), *Evolutionary computation*, vol. 11(1), pp. 1–18, 2003.
- [13] Hansen N., Ostermeier A.: Completely derandomized self-adaptation in evolution strategies, *Evolutionary computation*, vol. 9(2), pp. 159–195, 2001.
- [14] Janati F., Abdollahi F., Ghidary S.S., Jannatifar M., Baltes J., Sadeghnejad S.: Multi-robot Task Allocation Using Clustering Method. In: *Robot Intelligence Technology and Applications 4*, pp. 233–247, 2017.
- [15] Ma Y., Bai S., Xu Z., Yang J.: The Formation Method of Task Coalition for Multi-UAV Cooperation. In: *IEEE 16th International Conference on Control & Automation (ICCA)*, pp. 713–718, 2020.
- [16] Ohta N., Iwasaki A., Yokoo M., Maruono K., Conitzer V., Sandholm T.: A Compact Representation Scheme for Coalitional Games in Open Anonymous Environments. In: *AAAI*, vol. 6, pp. 697–702, 2006.
- [17] Opiola L., Jarosz K., Dutka L., Slota R.G., Kitowski J.: Group Membership Management Framework for Decentralized Collaborative Systems, *Computer Science*, vol. 23(4), 2022.
- [18] Ostrowski K., Starzec M., Starzec G.: The Ant Colony Optimization Algorithm Applied in Transport Logistics, *Computer Science*, vol. 25(3), 2024.

- [19] Prantare F., Appelgren H., Heintz F.: Anytime Heuristic and Monte Carlo Methods for Large-Scale Simultaneous Coalition Structure Generation and Assignment, *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 11317–11324, 2021.
- [20] Prantare F., Heintz F.: An anytime algorithm for optimal simultaneous coalition structure generation and assignment, *Autonomous Agents and Multi-Agent Systems*, vol. 34, pp. 1–31, 2020.
- [21] Prantare F., Heintz F.: Hybrid Dynamic Programming for Simultaneous Coalition Structure Generation and Assignment. In: *International Conference on Principles and Practice of Multi-Agent Systems*, pp. 19–33, 2020.
- [22] Prantare F., Ragnemalm I., Heintz F.: An Algorithm for Simultaneous Coalition Structure Generation and Task Assignment. In: *20th International Conference on the Principles and Practice of Multi-Agent Systems*, pp. 514–522, 2017.
- [23] Rahwan T., Michalak T.P., Wooldridge M., Jennings N.R.: Coalition structure generation: A survey, *Artificial Intelligence*, vol. 229, pp. 139–174, 2015.
- [24] Service T., Adams J.: Coalition formation for task allocation: Theory and algorithms, *Autonomous Agents and Multi-Agent Systems*, vol. 22, pp. 225–248, 2011.
- [25] Taguelmimt R., Aknine S., Boukreda D., Changder N.: Anytime Index-Based Search Method for Large-Scale Simultaneous Coalition Structure Generation and Assignment. In: *European Conference on Artificial Intelligence*, pp. 2282–2289, 2023.
- [26] Tran-Thanh L., Nguyen T., Rahwan T., Rogers A., Jennings N.R.: An Efficient Vector-Based Representation for Coalitional Games. In: *IJCAI*, pp. 383–389, 2013.
- [27] Wrobel K., Torba P., Paszynski M., Byrski A.: Evolutionary Multi-Agent Computing in Inverse Problems, *Computer Science*, vol. 14(3), p. 367, 2013.
- [28] Wu Y., Wu D., Yang L., Shi X., Ao L., Fu Q.: Matching-Coalition Based Cluster Formation for D2D Multicast Content Sharing, *IEEE Access*, vol. 7, pp. 73913–73928, 2019.
- [29] Zitouni F., Harous S., Maamri R.: Towards a Distributed Solution to the Multi-Robot Task Allocation Problem with Energetic and Spatiotemporal Constraints, *Computer Science*, vol. 21, pp. 3–24, 2020.

Affiliations

Tuhin Kumar Biswas

Department of Computer Science and Engineering, National Institute of Technology
Durgapur, Durgapur-713209, India, tuhin11biswas@gmail.com

Animesh Dutta

adutta.cse@nitdgp.ac.in

Received: 16.06.2025

Revised: 26.12.2025

Accepted: 14.09.2026

Early bird