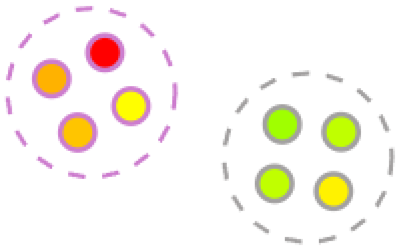


Team Formation in Gamified Environments

Pedro Alexandre Gonçalves Vilela

Thesis to obtain the Master of Science Degree in

Engenharia Informática e de Computadores

Supervisor: Prof. Carlos António Roque Martinho

Examination Committee

November 2022

Acknowledgments

I would like to thank my parents for their friendship, encouragement and caring over all these years, for always being there for me through thick and thin and without whom this project would not be possible. I would also like to thank my grandparents, aunts, uncles and cousins for their understanding and support throughout all these years.

I would also like to acknowledge my dissertation supervisors Prof. Carlos Martinho and Samuel Gomes for their insight, support and sharing of knowledge that has made this thesis possible.

Last but not least, to all my friends and colleagues that helped me grow as a person and were always there for me during the good and bad times in my life. Thank you.

To each and every one of you – Thank you.

Abstract

This work targets the formation of coalitions in a learning environment such that all members gather the most knowledge possible while being engaged in the tasks. To accomplish this objective, various state-of-the-art group formation and agent representation approaches were investigated based on how closely they are related to our problem. The algorithms were implemented in a framework we selected that allows the comparison of the selected algorithms to other, already tested approaches. These previously implemented algorithms are stochastic, therefore less computationally complex, and faster at finding their solution, but with no guarantees to finding the optimal solution. To compare between coalition formation algorithms, we conducted simulations to evaluate their computational efficiency and solution quality. The algorithms are evaluated by the ability and engagement levels of the groups they form, the time they take to find these solutions, and other aspects such as if the solutions always have fixed groups, i.e. groups where the students are the same for every group work, as it can be detrimental to the class dynamic. With the simulations, we reached the conclusion that deterministic approaches can indeed compete and even surpass stochastic ones, for around 20 students. To evaluate the applicability of our algorithms' real-world deployment, we experimented with professors, by having them going through a usability test using an existing group management tool. From the experiments, we gathered some feedback, which we will use to improve the group management tool and the usability of the algorithms.

Keywords

Coalition structure generation; Interaction profile; Exact algorithms; Heuristic algorithms; ODP-IP; Synergistic relations.

Resumo

Este trabalho tem como objetivo a formação de grupos em contexto de aprendizagem tal que todos os membros consigam obter o máximo conhecimento possível enquanto estão empenhados nas tarefas. Para concretizar este objetivo, foram investigados vários algoritmos de formação de grupos e representação de agentes no estado-da-arte, dos quais realçamos os mais relevantes para o nosso problema. Os algoritmos foram implementados numa framework que selecionamos que permite comparar os algoritmos selecionados com outras abordagens já testadas. Estes algoritmos são estocásticos, logo, menos complexos computacionalmente e mais rápidos a encontrar uma solução, mas sem garantia de encontrar a solução ótima. Para comparar as técnicas de formação de grupos, realizámos simulações para os algoritmos. Os algoritmos foram avaliados pelos níveis de habilidade e empenho dos membros dos grupos que formam, o tempo para encontrar uma solução, e outros aspetos como, por exemplo, se as soluções contêm sempre grupos fixos, ou seja, grupos onde os alunos são os mesmos para todos os trabalhos de grupo, aspeto que pode ser prejudicial para a dinâmica da turma. Com as simulações, chegamos à conclusão que as abordagens determinísticas conseguem de facto competir e até mesmo superar as estocásticas, para por volta de 20 estudantes. Para avaliar a aplicabilidade dos nossos algoritmos num contexto mais realístico, fizemos experiências com professores, através de um teste de usabilidade, usando uma aplicação de gestão de grupos já existente. Com as experiências, reunimos feedback que pretendemos usar para melhorar a aplicação de gestão de grupos e a usabilidade dos algoritmos.

Palavras Chave

Geração de grupos; Perfil de interação; Algoritmos exatos; Algoritmos heurísticos; ODP-IP; Relações sinergéticas

Contents

List of Figures

List of Tables

Acronyms

C-Link	Coalition-Link
CFSS	Coalition Formation with Sparse Synergies
CSG	Coalitional Skill Game
DP	Dynamic Programming
DyCE	Dynamic programming for optimal connected Coalition structure Evaluation
GA	Genetic Algorithm
GIMME	Group Interactions Manager for Multiplayer sErious games
GIP	Group Interaction Profile
IP	Integer Partition
KNN	K-Nearest Neighbors
LB	Lower Bound
ODP	Optimal Dynamic Programming
ODP-IP	Optimal Dynamic Programming - Integer Partition
PL	Partition Linkage
PLS	Player Learning State
PRS	Pure Random Search
SEQ	Single Ease Question
SUS	System Usability Scale
TA	Teacher Assistant
UB	Upper Bound
WTSG	Weighted Task Skill Game

1

Introduction

Contents

1.1 Motivation

Recently, there has been a shift from more traditional instructor-centered paradigm, to more contemporary active learning methodologies [6]. Being these active methodologies more student-centered, they apply paradigms such as group based learning in order to improve the educational processes. The formation of groups between students can be seen as a coalition formation problem, given that research usually defines coalitions [5] as groups of agents that agree to coordinate with each other to achieve their goals. Examples of applications of coalition formation include autonomous sensors to improve surveillance [7] and virtual power plants to reduce the uncertainty of their expected outcome [8].

As more active methods promote peer discussions, less apt students can be aided by more apt students and thus are less likely to give up. Not only that, but the protégé effect [9] may also occur whenever more active students learn by explaining content to other students, finding gaps in their own understanding [10]. Besides, different students may have different perspectives on the way they interact with others to accomplish their learning goals, i.e. they may present different interaction profiles. In fact, we will use the term interaction profile recurrently throughout the document, to represent the preference of students for a certain way of interacting. However, more active methodologies may also reveal some problems. In some cases, teachers who attempt cooperative learning encounter some complaints from the students, for example, the brighter students can complain of being held back by the weaker ones [10]. We believe this problem has higher chance of occurring if the groups are formed at random and can happen even if the students choose their group. Given the aforementioned limitations, the goal of this proposal is to find an algorithm that can improve group formation, by joining students deemed as compatible according to their interaction preferences. We believe that this can, in turn, increase the engagement and the ability gained of each student.

1.2 Problem

The problem of dividing students into coalitions can be defined as a coalition structure generation problem. A coalition structure generation problem is defined by the literature as the partitioning of a set of agents into disjoint coalitions to maximize the total reward from the resulting coalitions, and can be divided into three main steps [5]:

- Coalition structure generation: This step involves each agent joining a coalition. The resulting set of disjoint coalitions is called the coalition structure, where each agent within a coalition coordinates their activities, but agents do not coordinate between coalitions;
- Solving the optimization problem of each coalition. In this step, it is found the best way for the members of a coalition to coordinate their actions, maximizing the performance of the coalition;

- Dividing the payoff of each generated coalition among its members.

This work is concerned with the optimization of the first activity of finding the coalition structure that minimizes the agent's incentive to deviate from their coalitions, taking into account their interaction preferences to improve the collective ability of the members of the coalition. To do this, we analysed and implemented a set of algorithms that satisfy this first step. As the supported use case for the algorithms requires the formation of a new coalition structure as an organization for a group project, we will conform our analysis to form coalition structures of around 20 students, the approximate size of a college lab shift.

1.3 Approach

For this work, we investigated various state-of-the-art approaches in group formation and agent representation. The algorithms we researched have not been much applied in school contexts, being deterministic approaches compared to the majority that are stochastic, as mentioned by Cruz et al. [11]. The authors justified the use of such stochastic algorithms due to their applicability to deal with a large number of variables and the possibility to rapidly generate useful (semi)-optimal solutions. We experiment with deterministic approaches to find if they can be viable, despite their low use.

We estimate the quality of a coalition resorting to simulations endowing two traits to each agent: the predicted ability increase, or the predicted knowledge gained by an agent working in said coalition, and the predicted engagement, representing the subjective desire or motivation to continue interacting. The quality of the coalition structure will be equal to the sum of all the coalitions' qualities.

For our proposed solution, we will classify each agent with an interaction profile, according to its preferences, where each agent of the same type has the same contribution to a coalition it is a part of. In addition, we will also consider that agents can have relations of synergy, where they may prefer to work with other agents in function of the compatibility between their interaction preferences. This aspect may be implemented, for instance, by increasing the value of a coalition whenever two agents that are considered to have good interaction compatibilities are joined. Each coalition will then be evaluated based on its members and their relations of synergy, also considering the agents' predicted performance. After evaluating each possible coalition, up to the preferred group size, we will run our proposed group formation algorithm to form the coalition structure with a higher overall performance based on the evaluation made.

As a basis for our implementation, we use Group Interactions Manager for Multiplayer sErious games (GIMME)[1] [12], a group organization model aimed to improve the collective ability of learners working

[1]The GIMME project repository can be found at the following GitHub link: https://github.com/SamGomes/GIMME (as verified by October, 24^{th}, 2022).

in groups. We selected the GIMME framework, because, besides providing a good environment for the implementation of our selected algorithms, it already included a set of stochastic algorithms that we used for comparison, given that the use of stochastic algorithms prevails [11]. To allow professors to more easily use our solution, we integrated it in the GIMME-Web[2] platform, a web program that serves as an interface to allow professors to form groups in a real class environment. We also evaluate this platform in terms of its usability and gather feedback to aid its development.

1.4 Objective

Our objective is to compare the proposed solution with the already implemented algorithms, to reach a conclusion on whether a more complex and deterministic approach can solve the problem and return solutions with a higher ability increase and engagement than the stochastic ones, and also in a useful time frame, to both improve the productivity of the group and the ability of each member in an education/training environment, while taking into account the preferences of the agents when forming the coalitions. More specifically, we will evaluate the algorithms by: (i) the quality of their solutions, meaning the predicted ability increase and engagement of the coalition structure; (ii) the time they need to find their solution, as with the increased complexity, we would not want algorithms that took hours each time a new coalition structure needed to be formed. The relation of quality of solution and time to find solution, will also help us determine if the increased complexity justifies the quality of the solution; (iii) another important aspect to evaluate, as the suggested approaches are deterministic, i.e. the solutions will be comparatively less varied each execution of the algorithms, we will also look at whether or not the algorithms start joining the same people together, which can be seen as detrimental for the classroom dynamic by some teachers [13]. In our preliminary experiments with users, we annotated the professors' opinion to this approach about these aspects.

1.5 Organization of the document

This thesis is organized as follows: Chapter 1 went over the motivation for this work, a brief description of the problem, what we researched to find a solution and our objectives. Chapter 2 details some of the traditional group formation techniques and their applications, the state-of-the-art in group formation algorithms and ways to value a coalition, the model that will serve as the basis for our implementation and how this affects our implementation. In chapter 3 we describe the architecture of the model we will use, the implementation of our solution, some of its limitations, and the updates we made to the interface

[2]The current version of GIMME-Web is available at: http://gimme-web.duckdns.org/ (as verified by September, 14^{th}, 2022). Its repository can be consulted via the following GitHub link: https://github.com/SamGomes/GIMME-Web (as verified by September, 14^{th}, 2022).

of our program. Chapter 4 will explain the tests to validate our solution, how they support the viability of the new options in the model, and the users' experiments conducted to test the usability of the practical web platform. In chapter 5, we conclude the document, presenting and discussing the results from this work as well as some future research directions.

2
Related Work

Contents

In this section, we will start by describing how effective group formation techniques have been implemented, followed by the state-of-the-art in group formation algorithms. Next, we will go over some approaches of how to represent the agents in the group formation context, and then we highlight some of the algorithms to divide these agents into coalitions. We follow this with an overview of GIMME, the model that we uses as the basis of our implementation. We end the Related Work with a discussion of the advantages and the disadvantages of the suggested representations and group formation approaches.

2.1 Traditional group formation

In this section, we start by describing some traditional group organization methods that have been proved to be effective in a variety of instructional settings [10], to insight environments that can easily accommodate the deployment of a group formation approach: Problem Sets, where students complete their assignments in teams, with the objective of only including the names of the people that participated in creating the obtained solution, so that the ones who do not want to do any work but want to pass, are incentivized to also participate; Laboratories and Projects: In some cases, the bulk of the work of group projects and laboratory assignments are done by only some members of the group, so, in addition to evaluating the group, we can have some individual testing to help correct the injustices and to assure everyone learns by working in their part of the assignment; Jigsaw: This method is referred to works that require multiple areas of expertise, for example, a laboratory exercise that might need expertise in experimental design, equipment calibration and operation, data analysis and interpretation in light of theory. After the group is formed, each member is assigned to a part and is given specialized training to complete the assignment. Also, if everyone is tested for every part of the project, it will greatly improve the learning from the assignment; Peer Editing: Instead of the instructor being the one criticizing a lab report or a presentation, it is a pair of groups criticizing each other's work, so that they can improve their solution based in the others' feedback and then submit their work to the instructor. This method also has the advantage of lightening the work of the teacher, who is able to receive work already improved by other students.

Knez et al. [14] experimented with their recommender system, ELARS, by performing online activities in groups of students in computer science courses, forming either homogeneous or heterogeneous groups. Their results concluded that, out of 64 students, 80% of the students thought it was effective doing the activity in groups, with 47% agreeing with the idea of grouping students according to their characteristics and 53% disagreed with fully random groups. Other research [11] mentioned that the majority of group forming algorithms used in instructional contexts are of a probabilistic nature and genetic algorithms. This is "due to their applicability to deal with a large number of variables and the possibility to rapidly generate useful (semi-)optimal solutions." Conversely, in this proposal, we will test some deterministic approaches to see if the group organization techniques highlighted by Rahwan et al. [5] can obtain better results compared to the probabilistic and genetic algorithms already implemented in the GIMME model.

2.2 State-of-the-art group formation and agent representation

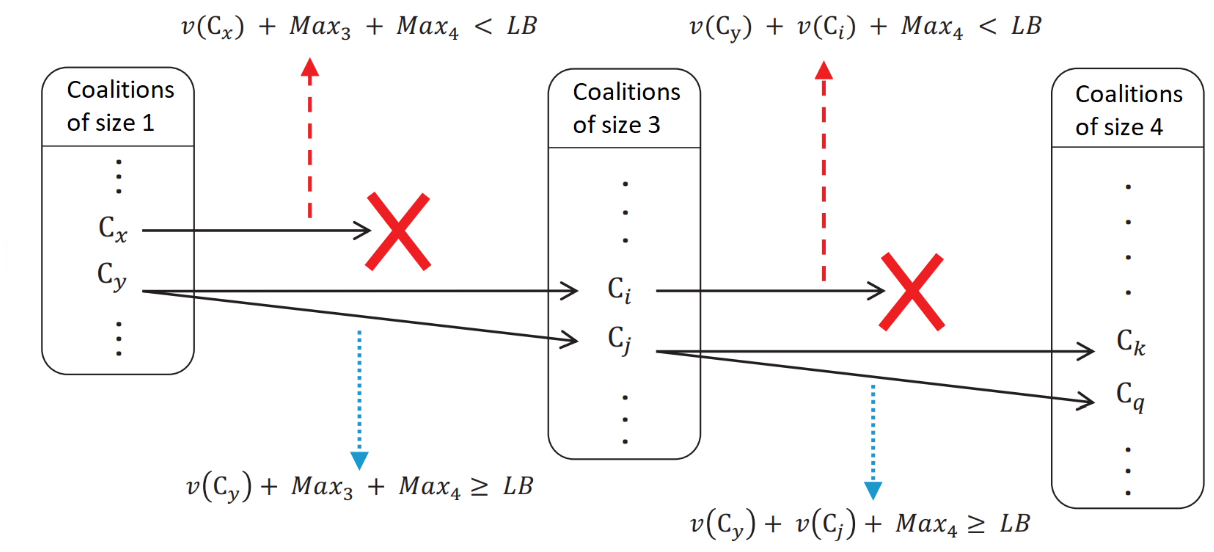
Next, we present some approaches defined in the literature as the state-of-the-art of group formation algorithms and agent representations. Rahwan et al. [5] mentioned various algorithms and representations for the problem of finding coalition structures that maximize social welfare. Given that our goal is to explore various approaches, hoping to discover which ones can better suit our group organization needs, we started by reviewing some coalition structure generation algorithms highlighted by recent research [5] that would be appropriate for the situation we are considering.

One algorithm that was proposed in the literature is the Inclusion-Exclusion algorithm created by Björklund [15], which is an exact dynamic programming algorithm for set partitioning. The algorithm is exact because it always returns the optimal solution, if it exists, and is dynamic because it sub-divides a complicated problem into simpler ones, and, recursively, solves the sub-problems. This algorithm is the theoretic state-of-art in terms of the worst-case scenario, running in O(2^n). However, because the algorithm encodes the information in extremely large numbers, with hundreds or even thousands of digits each, and requires multiplying those numbers, the runtime of this algorithm grows at a rate of O(6^n) rather than the theoretic O(2^n), as indicated by the tests performed by Michalak et al. [1]. Particularly, for a set of 15 agents, the algorithm required more than five months to terminate, as of the date of this document. We expect, in our case, to simulate the execution of our approach in a school or college class, which typically has more than 15 students, thus rendering such a computationally expensive method inadequate for our scenario.

Another approach mentioned is Integer Programming, which expresses the optimization of a linear function restricted to a set of linear constraints with the variables also restricted to integers. However, this approach has been shown to be inefficient, e.g., the industrial-strength solver created by IBM, ILOG CPLEX[1], quickly runs out of memory with around 20 agents, but tends to be very efficient in solving other combinatorial optimization problems. Nonetheless, Integer Programming was shown to be significantly slower than other algorithms, such as the Integer Partition (IP) [16] and Dynamic programming for optimal connected Coalition structure Evaluation (DyCE) [3] algorithms.

The IP algorithm [16, 17] represents its search space with disjoint subspaces based on the sizes of each partition. For example, for a set of 5 agents, one possible partition would be $\{$1,2,2$\}$, which means

[1]A link to IBM's ILOG CPLEX can be found at the following link: https://www.ibm.com/products/ilog-cplex-optimization-studio (as verified by December, 18^{th}, 2021).



$v(C_x) + Max_3 + Max_4 < LB$
$v(C_y) + v(C_i) + Max_4 < LB$
Coalitions of size 1
Coalitions of size 3
Coalitions of size 4
C_x
C_y
C_i
C_j
C_k
C_q
$v(C_y) + Max_3 + Max_4 \geq LB$
$v(C_y) + v(C_j) + Max_4 \geq LB$

Figure 2.1: An illustration of IP's branch-and-bound technique when searching a subset of 8 agents, for the partition {1,3,4}. Here the algorithm recognises that the coalition structure containing C_x or C_y, C_i cannot be optimal, and does not proceed deeper into the search tree. $v(C)$ is the value of the coalition C given by the characteristic function. Max_3 and Max_4 represent the max value of any coalition of size 3 and 4 respectively. Source: [1].

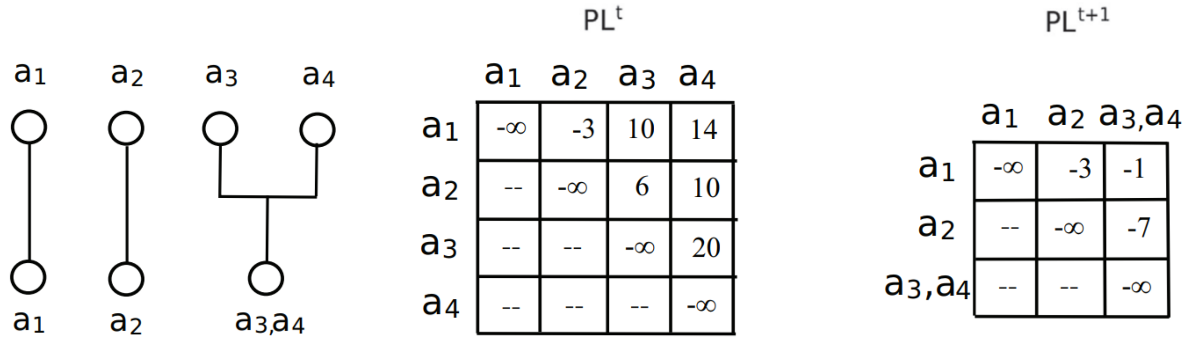
that one coalition has one member, and the other two have two members each. With this representation, it is possible to compute Lower Bounds (LBs) and Upper Bounds (UBs) on the quality of the best solution, based on the average and on the maximum value, given by the characteristic function of every coalition structure, respectively. In the context of our proposal, the characteristic function refers to the function that maps each coalition to its value. Using LB, it is possible to eliminate some of the subspaces that are not promising to contain the best solution, i.e. the subspaces with lower value than the LB. As for the UB, it can be used to bound the quality of the best solution found so far, (quality of solution = UB ÷ value of solution). The algorithm constructs a tree for every subspace, where every node represents a subset and every path a partition. These trees are traversed in a depth-first manner, applying a branch-and-bound technique, where the solutions are represented in a tree and the branches that have no potential of becoming an optimal solution are eliminated, using the previously calculated LB and UB for the coalition structures' values. Figure 2.1 shows how IP searches a subspace of 8 agents, for the partition {1,3,4}. Additionally, this algorithm has the characteristic of being anytime, meaning that its solution improves gradually overtime, allowing it to return a valid solution at any moment during its execution.

Another algorithm is the Dynamic Programming (DP) algorithm, proposed by Yeh [18]. This algorithm is based on the dynamic-programming principle, which is appropriate when the problem can be broken down into smaller problems. The way this algorithm solves the smallest problems first is by, given a set of agents, finding an optimal partition for every coalition of size 2 and storing them in memory, and then use these results to compute the optimal partition of the coalitions of size 3 and so on until it finds an optimal partition for the original set. Later, Rahwan et al. [19] observed, through the graphical representation of the search space created by Sandholm et al. [20], where every node represents a coalition structure, that some of DP's operations were not needed to ensure an optimal solution and used this idea to develop the Optimal Dynamic Programming (ODP) algorithm, that performs only one third of DP's operations, without losing the guarantee of finding an optimal solution.

Using the previous approaches, a new algorithm was developed by Rahwan et al. [21], creating the Optimal Dynamic Programming - Integer Partition (ODP-IP). This algorithm uses an adapted version of both algorithms and executes them concurrently, using the information provided by ODP to speed up IP's search. This combination has the advantages of both of its parts, being $O(3^n)$ and exact like ODP and returning solutions anytime like IP, outperforming both ODP and IP, although it is unlikely to solve group formation problems with 40 agents or more [5]. This is the fastest anytime exact algorithm to date, thus we believe in our case it may be able to find a solution in a short amount of time, given that our problem implies groups with a considerably lower number of agents than the algorithm's experimentally defined upper bound.

Another class of approaches to solve the coalition structure generation problem are the heuristics algorithms. These algorithms do not guarantee that an optimal solution is ever found, and they also cannot provide any guarantees on the quality of their solution. However, they are more suitable to larger problems that exact approaches cannot handle. From this class of algorithms, we highlight the Coalition-Link (C-Link) algorithm, proposed by Farinelli et al. [2]. This algorithm aims to partition a set of agents into groups, based on the concept of similarity. The similarity would be estimated with a suitability function $sf(C_i, C_j)$ to determine at each step how convenient it is to join the two coalitions C_i and C_j. The authors suggest that this gain can be calculated as the value of the junction of two coalitions after subtracting the value of the two coalitions before joining. To store this information, the algorithm updates in each iteration the Partition Linkage matrix PL$^{(t)}$, with the values of $sf(C_i, C_j)$ in the entry (i, j). The approach starts from a partition where every coalition is formed by a single agent and initializes the PL matrix with the suitability values of each pair of agents. In every iteration after that, the algorithm computes the most suitable pair of coalitions and, if merging a pair of coalitions has a positive gain, they are removed from future iterations and their union is added instead. If the best suitability is negative or if the grand coalition was formed, the algorithm stops. Figure 2.2 shows an example of a matrix update step of the C-Link algorithm. With this method we may be able to achieve good-enough solutions, using much less time and memory compared with a more complex approach, as it was empirically shown to have good results, requiring $O(n^3)$ operations in the worst case.

Yet another way of approaching the coalition structure formation problem is by putting constraints on the possible coalitions. The first algorithm we present of this category is the DyCE, proposed by Voice et al. [3]. The particularity of DyCE is that it resorts to a graph $G = (I, E)$, with I being the set of



	a_1	a_2	a_3	a_4
a_1	$-\infty$	-3	10	14
a_2	--	$-\infty$	6	10
a_3	--	--	$-\infty$	20
a_4	--	--	--	$-\infty$

	a_1	a_2	a_3,a_4
a_1	$-\infty$	-3	-1
a_2	--	$-\infty$	-7
a_3,a_4	--	--	$-\infty$

Figure 2.2: This figure depicts an example of a matrix update step of the C-Link algorithm, where the optimal coalition structure is $\{\{a_1\},\{a_2\},\{a_3,a_4\}\}$, and shows an illustration of the update on the dendrogam on the left and on the PL matrix. Source: [2]

agents and E the set of edges between agents, and the algorithm only considers the feasible coalitions, which are the ones represented in this graph. An example of this graph can be seen in Figure 2.3. The algorithm proceeds by calculating the w value for each feasible coalition. The authors informally define the w function as, for any coalition C, the maximum total value of any sub-coalition of the coalition C. The algorithm starts by initializing every coalition w value as $-\infty$, which, as mentioned by the authors, requires a large enough memory block to store all of these values. Then it evaluates each feasible coalition and stores their values given by the characteristic function. After this initialization step, the algorithm goes over every coalition C in order of their size and, if the coalition C is feasible, it computes $w(C)$ and updates the stored value in memory. To calculate $w(C)$ for a coalition $C \neq I$, DyCE cycles through every feasible sub-coalition $C' \subset C$ that is smaller than $n - |C|$, (with n being the total number of agents), and $|C| \div 2$, and then stores in $w(C)$ the maximum between $w(C \setminus C') + w(C')$ and the value given to C by the characteristic function. As an example considering the graph from Figure 2.3, for the coalition {1,4,5}, its w value would be $max(v(\{1,4,5\}), w\{1\} + w(\{4,5\}), w(\{5\}) + w\{1,4\})$, with the v function representing the characteristic function. The last step consists on the computation of $w(I)$, working similarly to the previous step. To get the optimal coalition structure from $w(I)$ the algorithm recursively replaces a coalition C with C' and $C \setminus C'$ such that $w(C) = w(C \setminus C') + w(C')$, stopping when no more subdivisions are possible.

An alternative algorithm was proposed by Bistaffa et al. [4], the Coalition Formation with Sparse Synergies (CFSS) algorithm, an anytime algorithm based on edge contraction, i.e. removing an edge from a graph and merging the two nodes that were joined by that edge. In our case, each node would initially represent a singleton coalition, and whenever two nodes were merged, we would join the two coalitions represented by those nodes, as seen in Figure 2.4(A). Furthermore, the contraction of an edge can be interpreted as the transition from one coalition structure to another. As an example, contracting the edge $(\{a_1\},\{a_3\})$ in Figure 2.4(B) corresponds to transitioning from $\{\{a_1\},\{a_2\},\{a_3\},\{a_4\},\{a_5\}\}$ to

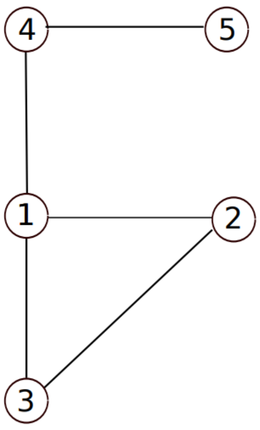


Figure 2.3: Example of a synergy graph with 5 agents used by the DyCE algorithm. Source: [3]

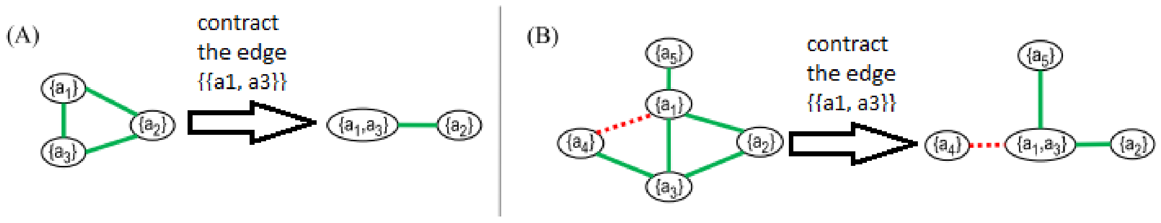


Figure 2.4: Example of edge contraction performed by the CFSS algorithm. Sources: [4, 5]

$\{\{a_1, a_3\},\{a_2\},\{a_4\},\{a_5\}\}$. Based on this observation, the algorithm repeats the process of contracting edges until it visits all coalition structures, marking the previously contracted edges to avoid contracting them in the future, ensuring that each coalition structure is visited no more than once. In Figure 2.4, the red dashed lines are these previously visited edges, while the green lines are the edges whose contraction is still possible. Similarly to the IP algorithm, a branch-and-bound technique is used to speed up the search, to determine if a coalition structure is worth searching, by calculating an upper bound for said coalition structure and, if this upper bound is not greater than the best solution found so far, the coalition structure is skipped. Also similarly to IP, the CFSS algorithm can also return anytime solutions if the problem it is trying to solve is too complex. The experiments conducted by the authors show that the CFSS algorithm can solve problems of 60 nodes in a matter of seconds, but is unlikely to scale beyond tens of agents.

For the coalition structure generation algorithms to work, they need to represent a population of agents with a diverse set of characteristics. To satisfy this requirement, we present some agent representations that can alleviate the computations needed to determine the values of each coalition. These representations work by restricting the coalitions that can be formed or by making generalizations of the agents, and how they can be applied in group organization.

The first representation, proposed by Bachrach et al. [22], referred to as a Coalitional Skill Game (CSG), defines every agent with a set of skills and every task with a set of required skills to complete it. A CSG has the objective of creating teams with a diverse skill set, so that they are able to respond to a more varied set of tasks. In this representation, the set of skills of a coalition is composed by the union of the sets of its members and a coalition is valid if it has the necessary skills to solve the tasks they are required to solve. So, the value of each coalition is based on the number of tasks its members can achieve. The authors also defined another type of CSG, the Weighted Task Skill Game (WTSG), where each task has a weight associated and the value of each coalition is now the sum of the weights of the tasks it can perform. This allows every coalitional game to be converted to a WTSG, although it can generate a very large CSG representation, with the number of tasks exponential to the number of agents.

Another representation, proposed by Aziz et al. [23], endows the agents with a set of types, in the sense that if two agents, when added to any coalition, increase its value by the same amount, they are of the same type and also referred to as strategically equivalent. The authors prove that, if we are given a valid type-partition, i.e. the partition of an agent population into sets according to the type of the agents, the number of types is bounded by a constant, and if the characteristic function is easy to compute, then an optimal coalition structure can be computed in polynomial time.

Other approaches described by research to characterize agents in coalition structure generation problems, are synergy-based representations [24], whose objective is to represent the links between agents that, for example, already know each other or like to work together. Originally proposed by Conitzer et al. [25], this representation considers that each agent has a utility when they are joined with other agents, and, if two agents with a synergistic link are joined, they give a bonus gain to the coalition they are a part of. Conversely, in the case that the agents of a coalition do not have a synergistic link, the algorithm assumes that a coalition formed by such agents has no added value and that the agents will partition themselves in the best possible way. This representation was then applied by Otha et al. [24] in an algorithm that would add an agent to a coalition at random, and then, with some probability, repeatedly add a new random agent until an agent was not added or the coalition had all agents. With this approach, they were able to solve, in a few milliseconds, group formation problems with around a hundred agents, and a few links between agents.

2.3 Group Interactions Manager for Multiplayer sErious games

We will now describe Group Interactions Manager for Multiplayer sErious games (GIMME) [12], a model developed for managing collective learning, and that will serve as a basis to our implementations and exploratory study. This model aims to improve the collective ability of learners integrated in an interactive serious environment, e.g. a group project, and operates iteratively, repeating three steps:

1. **Initialize/Update players' states**, where the player and group states are created or updated from the results of previous iterations. These players' states are characterized by two metrics: ability, or the previously acquired knowledge, and engagement, the desire or motivation to continue interacting. In the first iteration, the player states can be initialized, e.g., from personality questionnaires;

2. **Organize the players into groups**. Next, the players are organized into groups, and a Group Interaction Profile (GIP) is defined for each group, according to each members' preferences. A GIP represents the types of interactions that need to be promoted to improve collective ability;

3. **Promote the group interactions in each group**: finally, appropriate techniques are selected to promote the styles of interaction conveyed in the GIP of each group.

We will now go over steps 1 and 2 in more detail. The GIMME model characterizes a Player Learning State (PLS) by two metrics: ability – the player's previously acquired knowledge – and engagement – the subjective desire or motivation to continue interacting. A PLS is represented by the tuple $<$ ability, engagement $>$. Ability can be estimated based in the progression of a player in a task. Engagement can be estimated, for instance, by the ratio of time spent attending the task and completing it [26], which means that a person that spends most of the time failing the task or a player that completes the task very fast or does not even try to complete it, is classified as having low engagement. In any other case the player is classified as engaged.

To profile the players' interactions, GIMME defines an interaction style as the space formed by two dimensions: Challenge – the intention of the agent of providing/embracing an easier or more challenging route for task completion, between "Facilitate" and "Complicate" – and Focus – the agent's predisposition to interact while paying attention to themselves or others. As an example, if a player's preference relates Challenge with Focus, and is reflected in the range [-3, 3] in both dimensions by the profile $<$2.74;0.20$>$, this means the player is classified as a Mutual Facilitator. As each agent is endowed with one of these nine styles of interaction, they can be used as the types referred in section 2.2, when representing agents with types.

We can discretize this space into 9 different areas, as seen in Figure 2.5:

- Self Facilitator: the focus is on facilitating own task progression;

- Mutual Facilitator: the consideration of mutual help as a way to learn;

- Others' Facilitator: the consideration of a fully altruistic task completion as a way to learn, even while disregarding own harmful consequences;

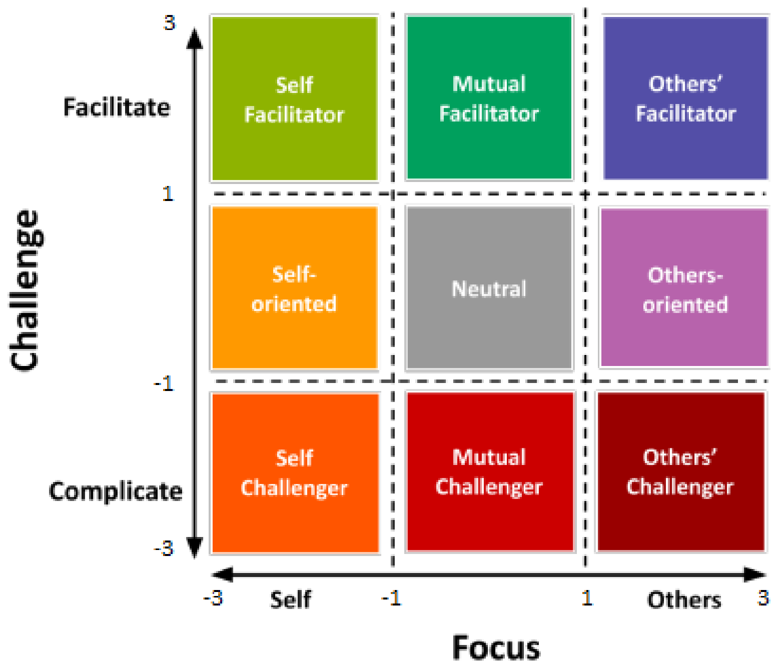


Figure 2.5: The space formed by the interaction dimensions: Focus and Challenge

- Self-oriented: interaction consisting on a balanced urge for self development;
- Neutral: unpolarized interaction with no predominant challenge or focus. It can be considered as a transitive state, before the agent assumes a stronger interaction style in the future;
- Others-oriented: the focus is on others' task completion, but does not influence others positively nor by making the task more challenging;
- Self Challenger: the completion of complicated self-oriented tasks as a way to learn;
- Mutual Challenger: the consideration of mutual competition as a way to learn;
- Others' Challenger: complicating others' tasks in the hope of valuing own efforts.

As for the group organization itself, several approaches have already been explored in preliminary tests performed in the scope of the GIMME model [12]. The most simple was inspired by the Pure Random Search (PRS) algorithm. The PRS-inspired approach generates several group organization samples, evaluates their quality, and selects the best one. The problem with this approach is that it is based exclusively on exploration, disregarding any information obtained from previous iterations. Besides, in order for it to be more effective, its implementation also relied on the strong assumption that the optimal groups' interaction corresponded to the average of the group members' interaction preferences.

Due to the limitations of the PRS algorithm, an evolutionary approach, in the form of a genetic algorithm, was then considered and tested. In this implementation, each genetic individual is endowed with a group configuration, represented by an array of player id sets, and an array of GIPs, so that each interaction profile is associated with one group of the configuration. Three main genetic operators were considered: (i) the selection operator, where the algorithm ranks the individuals based on their fitness, calculated as the sum of the predicted ability increases and engagement values of all agents since the last iteration; (ii) the crossover operator, that works on two phases: in the first one, the groups are crossed and in the second phase, the GIPs of the groups are crossed; (iii) and the mutation operator, that also has two phases. First the groups are mutated and then the GIPs. The mutation of a group configuration consists on swapping two groups' members, and the mutation of a GIP in the increase or decrease of its value by a low random margin.

To evaluate the quality of a GIP in respect to a player, the K-Nearest Neighbors (KNN) was used for Regression, computing the predicted performance based on the average of the past performances' GIPs experienced. Both the PRS and genetic approaches suffer from the low accuracy problem of the KNN estimator, because of the lack of stored data, taking several iterations before being able to make reliable predictions. This problem can be mitigated with a bootstrapping phase to populate the estimator. The bootstrap, that considers initial players' preferences, inferred, for example, from personality questionnaires, that then simulates an execution of the algorithm through several iterations using these initial preferences. By generating an initial population of estimated player performances for different GIPs, the early solutions can have an increased accuracy, regardless of the optimization process.

2.4 Discussion

Having presented the algorithms and representations, we are now going to mention the advantages and disadvantages of each.

Starting with the representations, seeing as this is to be applied to a class of around 20 people, there might not be a varied skill set among the agents, as everyone should have all the necessary skills to take on the tasks. So it still appears to be better to apply representations that consider the preferences of each agent. By classifying each agent with an interaction profile based on their preferences, that are more varied than the agents' skill sets, we can easily distinguish each agent by their type and estimate what their contribution might be when in a coalition, using the nine interaction profiles described in section 2.4 as the set of possible types the agents can have. Besides, we can also consider synergistic relations between interactions, i.e. that some interactions are compatible with other, specific interactions. A synergy-based representation can benefit from a graph representation, allowing us to trim infeasible coalitions whose members are not linked. Furthermore, it could allow the professor to have a better control over the coalition formation process.

Regarding group formation, we already presented some deterministic and stochastic algortithms that can be applied. In fact, the present work serves as an extension to the GIMME model, that adds the deterministic ODP-IP and C-Link algorithmic alternatives to the already implemented PRS and genetic stochastic group formation procedures. Starting with the PRS algorithm, it is a simple first approach that relies exclusively on exploration. In preliminary experiments, the algorithms also made some strong assumptions such as the preferences of a group being the averages of the preferences of their members. It also did not take into account possible synergies between agents, problem we will try to resolve with the new representation. Still, a genetic algorithm was developed next, targeted to fix these issues with exploitation and lack of assumptions. These two approaches have a common problem, being the number of iterations needed in order to make well-founded, reliable predictions, due to the impreciseness of the KNN estimator. This problem may also occur with the deterministic approaches, as these also depend on the characteristic function to classify the coalitions. To solve this problem, a bootstrapping phase can be used, adapting the one already implemented for the new approaches, by considering initial players' preferences, inferred, for example, from personality questionnaires and simulating a GIMME execution through several iterations using these initial preferences, generating an initial population, alleviating the initial low accuracy.

Each of the deterministic algorithms also has its own disadvantages. The ODP-IP algorithm, being an exact algorithm, guaranteeing to find an optimal solution, is much more computationally complex than the other algorithms, so it can take a long time to find solutions for a higher number of agents, problem that the C-Link algorithm avoids by sacrificing the quality of the solution, but since the number of agents is quite low in the proposed use case, we consider the ODP-IP algorithm a better approach, as it will find the optimal coalition structure, according to the characteristic function. We also presented the DyCE and CFSS algorithms. The former is a dynamic programming algorithm like ODP, that, by default, restricts the feasible coalitions by considering only the coalitions that contain agents with synergies, and the latter uses the concept of edge contraction and branch-and-bound techniques. Both algorithms should be faster than ODP-IP without putting restrictions over the agents, as the experiments made by Voice et al. [3] show that DyCE is faster than ODP and, in turn, Bistaffa et al. [4] showed that CFSS is faster than DyCE, although this requires that the synergy links between the agents to form the synergistic graph are not hard to identify. Additionally, Bistaffa et al. [4] noted some advantages CFSS has over DyCE. Firstly, as CFSS partitions the search space non-redundantly, it allows parallelisation, while DyCE does not due to the exponential memory requirements in the number of agents, limiting its scalability. The CFSS also has the advantage of allowing anytime solutions, similar to ODP-IP. However, because of this similarity, it might be redundant to test another algorithm that would consider every possible coalition, giving priority to the ones that contain synergies, therefore, we find more interesting to test DyCE that only considers the feasible coalitions. Either way, if we make it so the previous algorithms are able to define feasible or infeasible coalitions, the need for the DyCE approach, whose only advantage over ODP-IP is an easy representation of the feasible coalitions, ceases to exist. Besides comparing the speed of each approach to solve the coalition structure formation problem, our objective is also to determine if the algorithms can solve our problem in an adequate time frame, i.e. in a couple of minutes, and if the resulting coalition structures have high quality, i.e. result in coalitions allowing the proliferation of high ability increase and engagement.

Concluding this section, the most relevant researched algorithms are: ODP-IP and C-Link algorithms. We will consider a representation where agents are classified in one of nine interaction profiles based on their preferences, and having the agents connected by synergies, that can either be used to represent how much two profiles want to be paired up or to represent the feasible coalitions, by choosing players that either have to be joint in the same coalition, or separated into different coalitions.

3

Solution Implementation

Contents

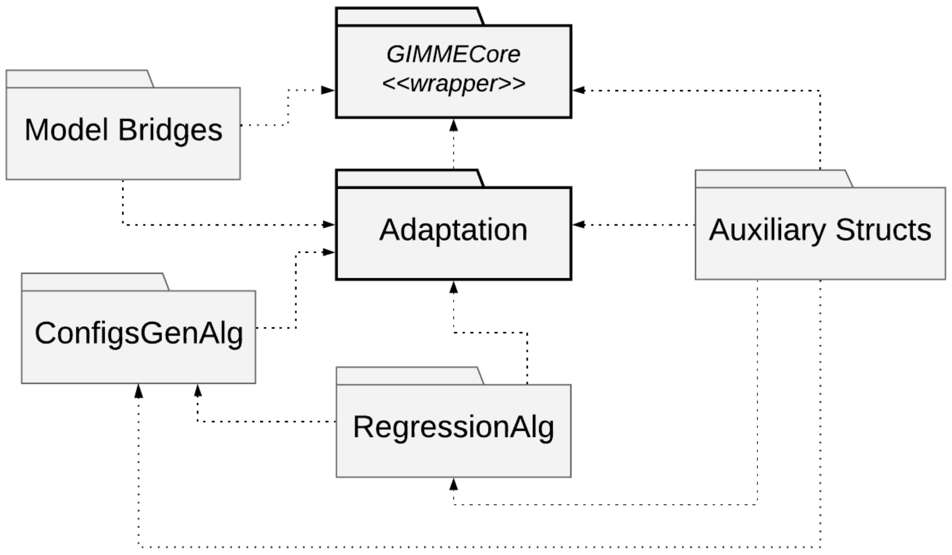


Figure 3.1: UML of the GIMME model

In this section, we will detail the architecture of the GIMME model, how we implemented the coalition structure generation algorithms, the main adjustments we made to the original implementation of the new algorithms, the previous algorithms and interfaces, some limitations found during implementation and at the end, we describe the web platform serving as an interface for deploying our algorithms.

3.1 Architecture

We will now give a more detailed description of how the GIMME group management meta-model is implemented and how the new algorithms were integrated. First, from a technical perspective, the GIMME model is implemented mostly in Python and is currently divided into the following modules: Adaptation, ConfigsGenAlg, RegressionAlg, Auxiliary Structs and Model Bridges (see the UML presented in Figure 3.1).

Adaptation encapsulates all functionalities of GIMME and is responsible for initializing and executing adaptation iterations. ConfigsGenAlg is an abstract base class for the coalition structure generation algorithms. It is in this module where we find the implementations of all the coalition structure generation algorithms. For the implementation of the new algorithms we had to implement a sub-module of ConfigsGenAlg, the GIMME Solver module, constituted by a C++ implementation of the ODP-IP and C-Link algorithms, that the Python side calls to generate a coalition structure. The other most relevant module for our work is the RegressionAlg module. Here we can find the implementation of the algorithms that serve as characteristic functions, namely the previously mentioned KNN and the newly added tabular algorithm, which considers the synergies between interaction profiles. Model Bridges allows for transparent model connections whenever data needs to be fetched, decoupling the algorithms and the storage structures. These bridges are mainly used to store the information of the agents and the created tasks.

3.2 ODP-IP

This algorithm was integrated in the GIMME model in two parts: the Python[1] side, responsible for computing the values for each possible feasible coalition, according to the characteristic function, their respective profiles and to translate the optimal coalition structure to the correct format from the answer given by the C++ module side, containing the implementation of the ODP-IP algorithm. We decided to implement it this way, because the programming language used for the GIMME platform was Python, but the ODP-IP algorithm is so complex that, when implemented in Python, it took too much time to complete even with only 16 players and so we let most of the computations be handled by the faster C++ module. We used the original implementation of ODP-IP made by the creators of the ODP-IP algorithm as a basis for our implementation[2], implemented in Java.

The C++ module receives as input: the number of players, an array with the values for every possible coalition, given by the characteristic function, where every position of the array corresponds to the value of the coalition represented by that position in binary (for example, if the value of the 75^{th} position of the array is 5.0, it means that the coalition 1001011, 75 in binary, consisting of a_1, a_2, a_4 and a_7, has the value given by the characteristic function of 5.0), the minimum and the maximum number of players per group, and two lists with constraints on how the agents can be distributed: one list contains the agents that need to be together in a group and, the other, the agents that cannot be in the same group.

The ODP-IP algorithm consists of two algorithms: ODP and IP that, in this case, are operating iteratively, starting by initializing various structures and variables, followed by an iteration of the ODP algorithm, that goes over 2/3 of the search space, finishing with IP that goes over the remaining subspaces, repeating these last two steps if an acceptable solution was not found.

The original implementation, already described in section 2.2, supported the use of feasible coalitions during the IP part of the algorithm, but due to, in our case, the ODP side being the one used the most, because we are forming groups with a low number of agents, i.e. the range of values of which ODP is responsible, we needed to extend this functionality to also be considered during the ODP's calculations.

[1] https://www.python.org/ (as verified by September, 14^{th}, 2022).
[2] https://github.com/trahwan/ODP-IP (as verified by October, 18^{th}, 2022).

To achieve this, we changed how the feasible coalitions are considered and also added a phase to compute this feasible coalitions. In the original implementation, they were either true or false, whether they were feasible or not. The feasible coalitions computation phase allowed us to add other features, such as: choose the minimum and maximum number of players per group and the possibility to create any number of constraints between the agents, where the user can select any set of agents and whether they need to be in the same group or separated. This phase goes as follows: (i) initialise an array of booleans with 2^n positions, where every position says if the coalition that corresponds to the value of that position in binary is feasible or not; (ii) go through every position of the array and mark as true all coalitions with a valid number of players per group, and also give a boost to their value; (iii) go through every position of the array again and verify, for every feasible coalition, if they satisfy every constraint defined by the user. If they do not, set them to false and change their value to a low positive number, to make the algorithm prioritize the other coalitions, but to choose one of these if the constraints set make it impossible to satisfy every constraint, for example, if a_1 and a_2, and a_2 and a_3, need to be together, but a_1 and a_3 have to be separated, there is no coalition structure that can satisfy all three constraints. If this happens, the algorithm would still return a solution with all three together as it results in the least number of unfeasible coalitions, i.e. a higher overall value for the coalition structure. In the case that there are constraints that require agents to be together, every coalition with all those agents, will receive an additional boost to their value, to prioritize coalitions that satisfy these constraints over the others.

3.3 C-Link

The C-Link approach is deterministic and greedy that, at each time step, will merge the two coalitions that result in a higher value. Like the ODP-IP algorithm, this approach has a Python side that computes every coalition value, according to the characteristic function, up to the max number of players per coalition, and is also responsible for constructing the optimal coalition in the correct format from the C++ module side. This module contains the C-Link algorithm following Farinelli et al.'s pseudocode [2], that we adapted according to the constraints of our problem.

The algorithm uses a linkage function to determine the value of merging two coalitions together. As suggested by the authors, this value is computed as $v(C_1 \cup C_2) - v(C_1) - v(C_2)$, where C_1 and C_2 are two coalitions and $v(C)$ the value of the coalition C according to the characteristic function. This approach starts by initializing the coalition structure with singleton coalitions, i.e. one agent per coalition and the Partition Linkage (PL) matrix which stores the value lf(C_i, C_j) in the entry (i, j). In the next step we compute the best linkage value and the coalitions that achieve it. We will use these coalitions in the main loop that involves: (i) removing the two coalitions we found in the previous step and adding the merged coalition to the coalition structure, (ii) deleting the rows and columns of the removed coalitions and adding one row and one column for the new merged coalition, and (iii) computing the new two best coalitions to merge.

The main difference between the original C-Link and our version is the condition to stop the loop. Originally, when there is no possible merge with a positive linkage value or the size of the coalition structure is 1, i.e. a coalition with all agents, the algorithm would stop. However, as in our case we have to respect the minimum and maximum number of players per group, we allow merges with negative linkage value, considering always the best of such merges and as long as the merge respects the maximum number of players per group, and we now stop if in any iteration we cannot find any merge that would respect the number of players per group limitations, for example, if the maximum number of players per group was 4, but we could only get coalitions with 5 or more players. In addition, we stop if the number of coalitions in the coalition structure is equal to the maximum number of groups. After exiting the loop we redistribute the players from coalitions smaller than the minimum size. These two steps of our version of C-Link will guarantee that the final coalition structure will have coalitions with sizes between the minimum and the maximum plus one number of players per group, in case someone remains when every other coalition already has maximum size.

3.4 Coalitions’ value estimation phase

For the ODP-IP and C-Link algorithms to compute their solutions, they require the value for every coalition they will consider. To do this, we execute a function before the algorithms, to estimate the value for each coalition, according to the characteristic function. This estimation phase starts from the total number of coalitions minus one down to zero, one by one, and does the following: (i) it converts the current number to binary and checks if it corresponds to a group with a valid group size and, if it does, (ii) compute the interaction profile of the group as the average of the interaction profiles of its members. (iii) Finally, it predicts, for every valid coalition, its value according to the characteristic function. This estimation phase, in contrast to the algorithms is implemented in Python.

3.5 Synergy-based characteristic function

Our original idea for a new characteristic function was based on the concept that every person would have some connections to some other people in the class and they would prefer to work with said people. However, finding these connections might be to difficult to do in every single class and also, two people knowing each other does not necessarily mean they will work well together.

Because of this, we developed a new characteristic function in addition to KNN, that we refer to tabular regression characteristic function. This method consists of using a table whose values correspond to the synergy between two interaction profiles, as shown in Table 3.1. We consider an interaction profile to have two dimensions, challenge and focus. For this characteristic function, we consider the range of each dimension of the interaction profile. If the individual is closer to complicate in the challenge dimension, then we assign it 0 in the first position, and 1 if closer to facilitate. Likewise for the focus dimension, if the individual is closer to self, we assign 0 at the second position, and if closer to others, we assign 1. So, if an interaction profile is “00”, it means, the individual is categorized as “Self-Challenger” (refer to Figure 2.5). It is to note that, with this representation, we cannot represent neutral interaction profiles. Due to our assumption that in our simulation an exact neutral value of 0.5 will be rare, it was assumed that an exact value of 0.5 would be considered equal to 1 in that dimension.

With this, we have a more general way to define synergies that will not depend on the individual and can be estimated either during the execution of the algorithms, or with questionnaires. To determine the value of a certain coalition, we compute the average value of each pair of players, according to the table. There is also another part of this characteristic function, that uses another table, with the same format, but this time, it is used between each player of a coalition and each of the possible tasks, to find the best task to assign to that coalition. In this case, the task chosen will be the one with an higher sum of values for each member of the group.

Table 3.1: Example of tabular characteristic function, where every interaction profile wants to be in the same group as similar interaction profiles.

Interaction Profile	00	01	10	11
00	1	0	0	0
01	0	1	0	0
10	0	0	1	0
11	0	0	0	1

This method will serve as an alternative to the KNN characteristic function, as it is less complex and allows for the user to freely choose how each profile interacts with one another.

3.6 Algorithm limitations

In this section, we will mention some of the limitations of this solution. The first is the memory required by the algorithms. Both algorithms always need to keep the quality of every possible coalition in memory, their average ability and engagement and each of their profiles, computed as the average of the preferences of each player in that coalition. These arrays have all size 2^n, as each position of the array in binary corresponds to one coalition. Additionally, all the computations need to be done for all coalitions that can possibly be selected, which, in our case, means that they need to respect the minimum and maximum size per group. As for the profiles and average characteristics of each coalition, the characteristic function requires these values in order to get the coalitions’ values, therefore we cannot only compute the profiles and characteristics of the coalitions of the coalition structure.

For ODP-IP the only change in terms of memory made was the addition of the feasible coalition array, an array of booleans, that for each position, says if the corresponding coalition in binary is feasible. This array has size 2^n and so, as mentioned by Michalak et al. [1], ODP-IP is O(2^n) in terms of memory.

The C-Link algorithm is much less complex than ODP-IP, requiring only the PL matrix to be stored in memory, which has n^2 entries. Unfortunately, due to estimation phase in Python, our implementation of the C-Link algorithm is also O(2^n), losing some of its advantageous points over ODP-IP. Additionally, because of C-Link's performance in our simulations, we decided to not implement restrictions at this stage, as it would slow it down even more, making it a less reasonable solution than ODP-IP.

3.7 GIMME-Web updates

In this section, we describe the updates performed to GIMME-Web, a web program that serves as an interface to allow professors to form groups in a real class environment. A screenshot of its home page is displayed in Figure 3.2. In terms of backend, GIMME-Web was implemented using the Django[3] web framework, however, we had to change the database system, from SQLite[4], the default service when creating a Django project, to MySQL[5]. This change was required, due to the time it took for the website to respond to any access to the database. Additionally, while working on the databases, we noticed that some code in the GIMME side was accessing the database during execution to get one value at a time instead of getting every value all at once, which also slowed down the execution of the program. The rest of the changes in the backend were just to integrate the new approaches and add the functionality for the tutorial section of the platform.

In the platform, users can register as Professors or Students, with each role having a different dashboard. We will start with the professor role, which was the side we changed the most, whose main objective is to manage the class. This dashboard has the following functionalities:

Manage adaptation iterations: In this section shown in Figure 3.3, the professor can change parameters such as the group size, by adjusting the slider to select the minimum and maximum number of students per group; and configure the bootstrap phase. The Additional Adaptation Parameters tab is where the professor can select between different group formation and regression approaches and configure some of their parameters.

Configure students and tasks: The professor can use the tabs in Figure 3.4 to select, from all registered students or tasks, which of them may be selected when forming the groups. If necessary, the professor can also change the students' learning state in the Student Setup tab, as well as create new tasks or edit existing ones in the Task Setup tab. Additionally, in the students setup tab, the professor

[3]https://www.djangoproject.com/ (as verified by September, 14^{th}, 2022).
[4]https://www.sqlite.org/index.html (as verified by September, 14^{th}, 2022).
[5]https://www.mysql.com/ (as verified by September, 14^{th}, 2022).

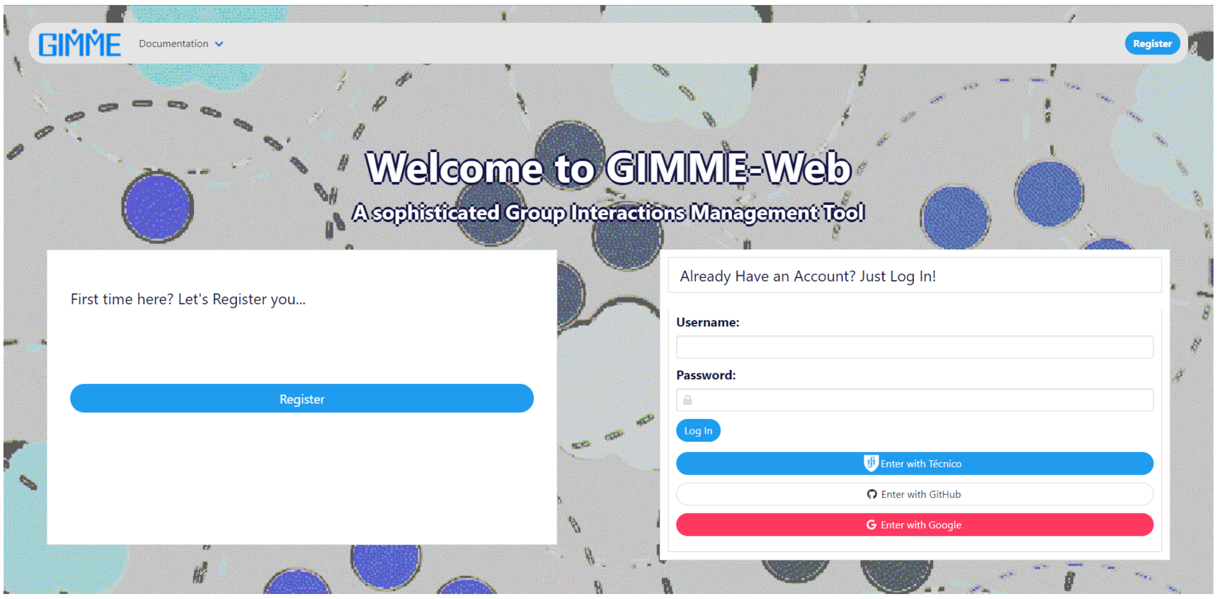


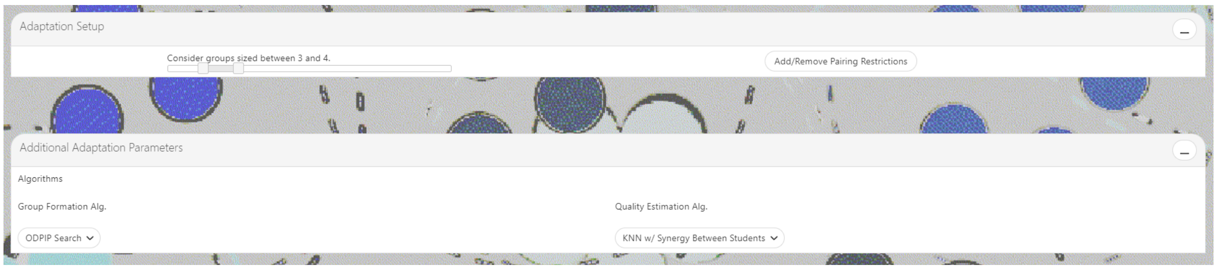
Figure 3.2: A screenshot of GIMME-Web's home page.

can define restrictions to the groups, i.e. whether students need to be separated or in the same group. The page to create the restrictions can be seen in Figure 3.5.

Monitor the state of the class: In the first half of this tab (Figure 3.6, the professor can start a new iteration, observe the proposed groups and their characteristics, such as their preferences and assigned tasks, can verify individual metrics for each student, such as their estimated ability, engagement, preferences and grade, swap students between groups and change how the color scale is displaying, either a relative or an absolute range. The scale only changes the minimum and maximum value of the scale. The relative scale has the lowest grade as minimum and the highest as maximum, and the absolute scale has 0% as minimum and 100% as maximum. In either case, closer to red means lower ability level and closer to green higher ability level. In the second half displayed in Figure 3.7, there is a scatter plot of the interaction profiles attributed to the groups, as well as a bar chart representing the average ability and engagement of all students currently selected for the current iteration.

Tutorial: In the Tutorial tab, shown in Figure 3.8, the professor can simulate their participation in a course, during 6 weeks, where in each week, they are given a set of tasks to do in that week, going over the main functionalities of the professor dashboard. This section was originally created to test the interface with users and was adapted to aid inexperienced users to navigate the interface.

The students can monitor their current learning state, obtain descriptions and files of the tasks assigned to them, and upload the results of those tasks. Besides this, the students also have access to the current group they are on, the tasks available to complete and the student's past preferences and learning state as perceived by the system (see Figure 3.9).


Adaptation Setup

Consider groups sized between 3 and 4.

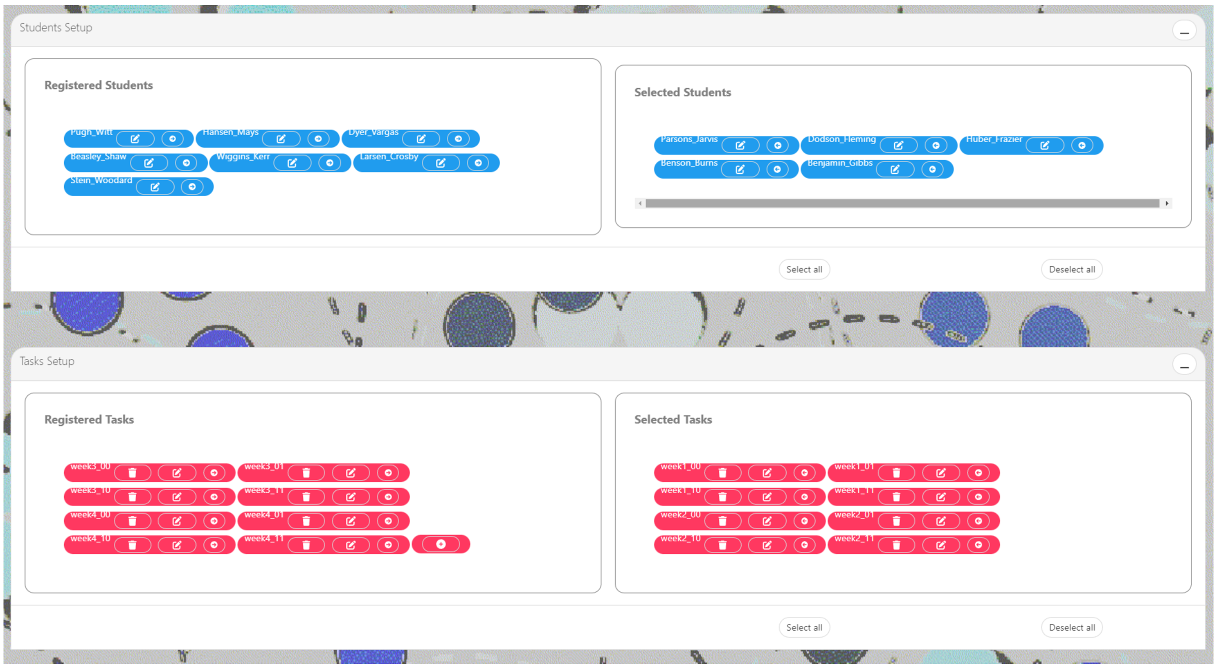
Additional Adaptation Parameters

Algorithms

Group Formation Alg.: ODPIP Search

Quality Estimation Alg.: KNN w/ Synergy Between Students

Figure 3.3: The Adaptation Setup and Additional Adaptation Parameters tabs of the Professor dashboard.


Students Setup

Registered Students

Pugh_Witt, Hansen_Mays, Dyer_Vargas, Beasley_Shaw, Wiggins_Kerr, Larsen_Crosby, Stein_Woodard

Selected Students

Parsons_Jarvis, Dodson_Fleming, Huber_Frazier, Benson_Burns, Benjamin_Gibbs

Tasks Setup

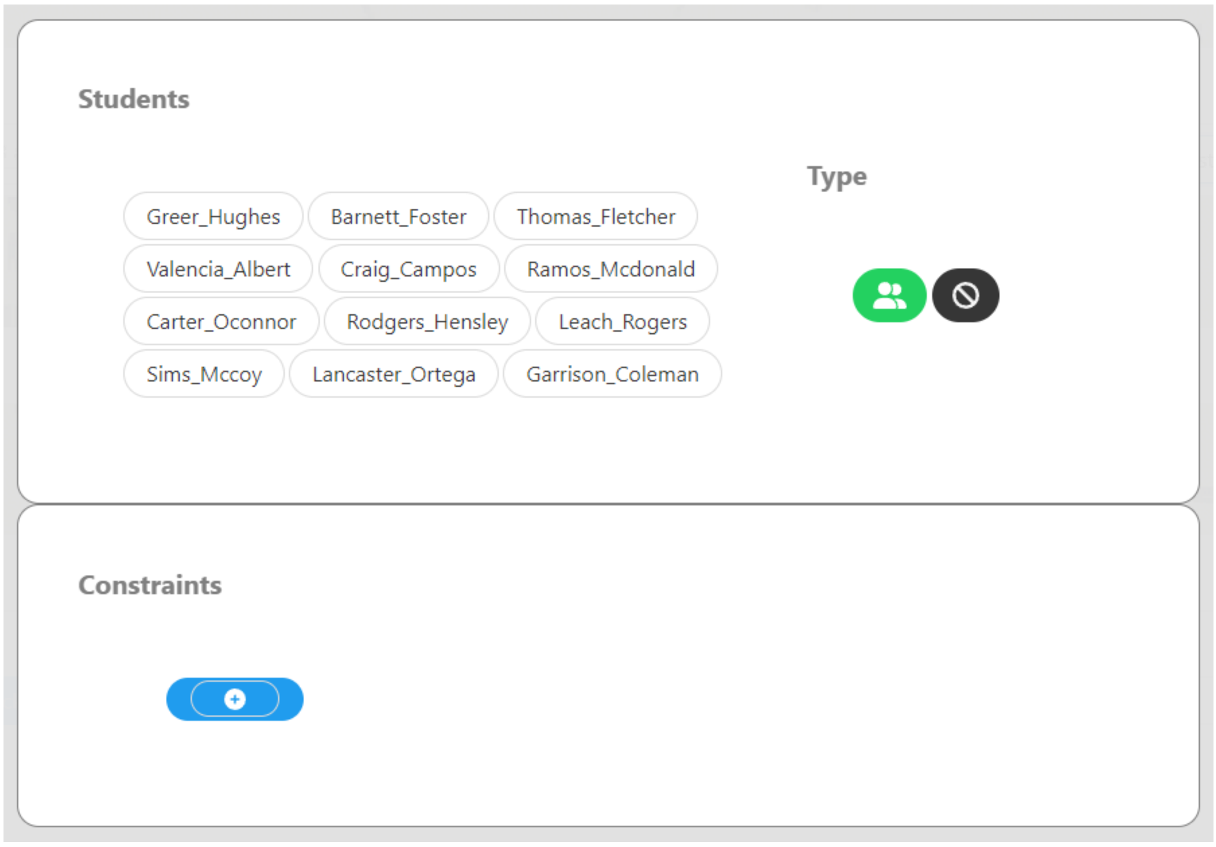
Registered Tasks

week3_00, week3_01, week3_10, week3_11, week4_00, week4_01, week4_10, week4_11

Selected Tasks

week1_00, week1_01, week1_10, week1_11, week2_00, week2_01, week2_10, week2_11

Figure 3.4: The Students Setup and Tasks Setup tabs of the Professor dashboard.



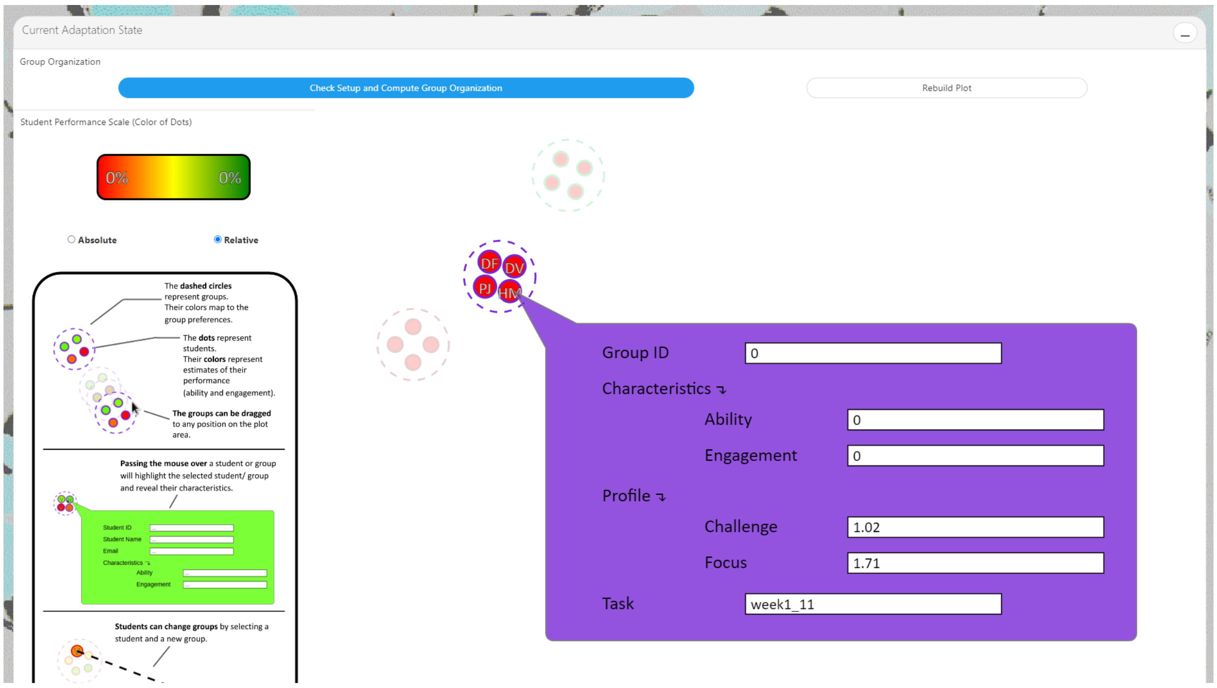
Students

Greer_Hughes
Barnett_Foster
Thomas_Fletcher
Valencia_Albert
Craig_Campos
Ramos_Mcdonald
Carter_Oconnor
Rodgers_Hensley
Leach_Rogers
Sims_Mccoy
Lancaster_Ortega
Garrison_Coleman

Type

Constraints

Figure 3.5: Create Restriction page of the Professor Dashboard.


Current Adaptation State
Group Organization
Check Setup and Compute Group Organization
Rebuild Plot
Student Performance Scale (Color of Dots)
0%
0%
DF
DV
PJ
HM

[ ] Absolute [✓ Relative]

The **dashed circles** represent groups. Their colors map to the group preferences.

The **dots** represent students. Their **colors** represent estimates of their performance (ability and engagement).

The groups can be dragged to any position on the plot area.

Passing the mouse over a student or group will highlight the selected student/ group and reveal their characteristics.

Student ID
Student Name
Email
Characteristics ↴
Ability
Engagement

Students can change groups by selecting a student and a new group.

Group ID: 0
Characteristics ↴
Ability: 0
Engagement: 0
Profile ↴
Challenge: 1.02
Focus: 1.71
Task: week1_11

Figure 3.6: The first half of the Class Learning State tab of the Professor dashboard.

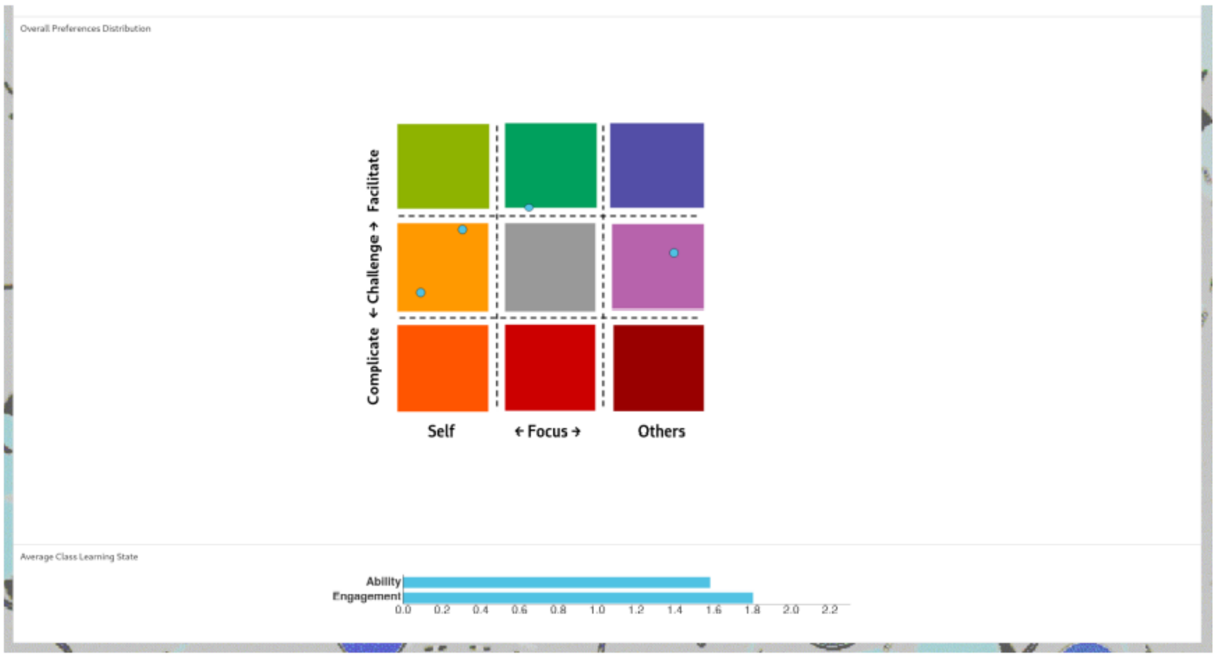
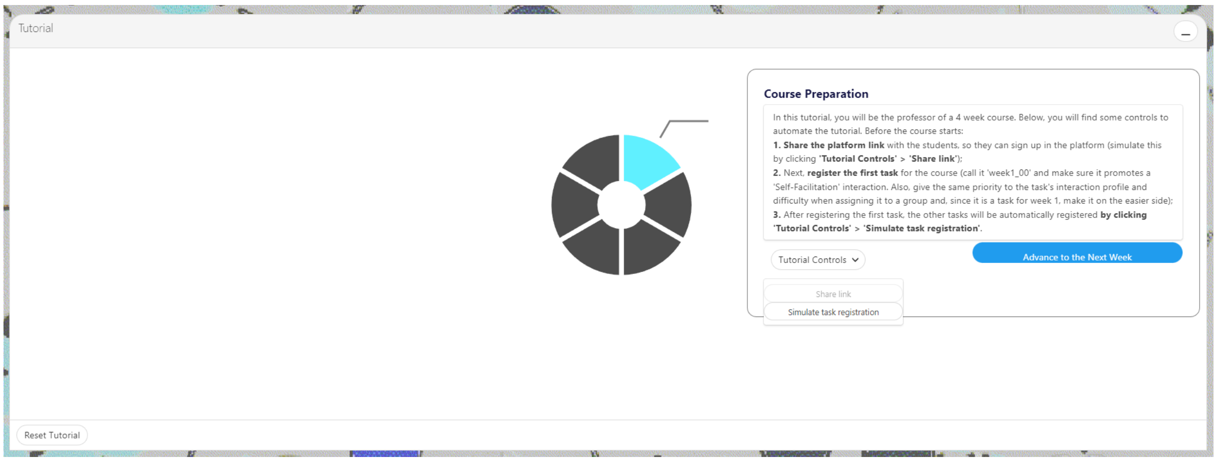

Overall Preferences Distribution
Complicate ← Challenge → Facilitate
Self
← Focus →
Others
Average Class Learning State
Ability
Engagement
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1.8
2.0
2.2

Figure 3.7: The second half of the class Learning State tab of the Professor dashboard.



Course Preparation

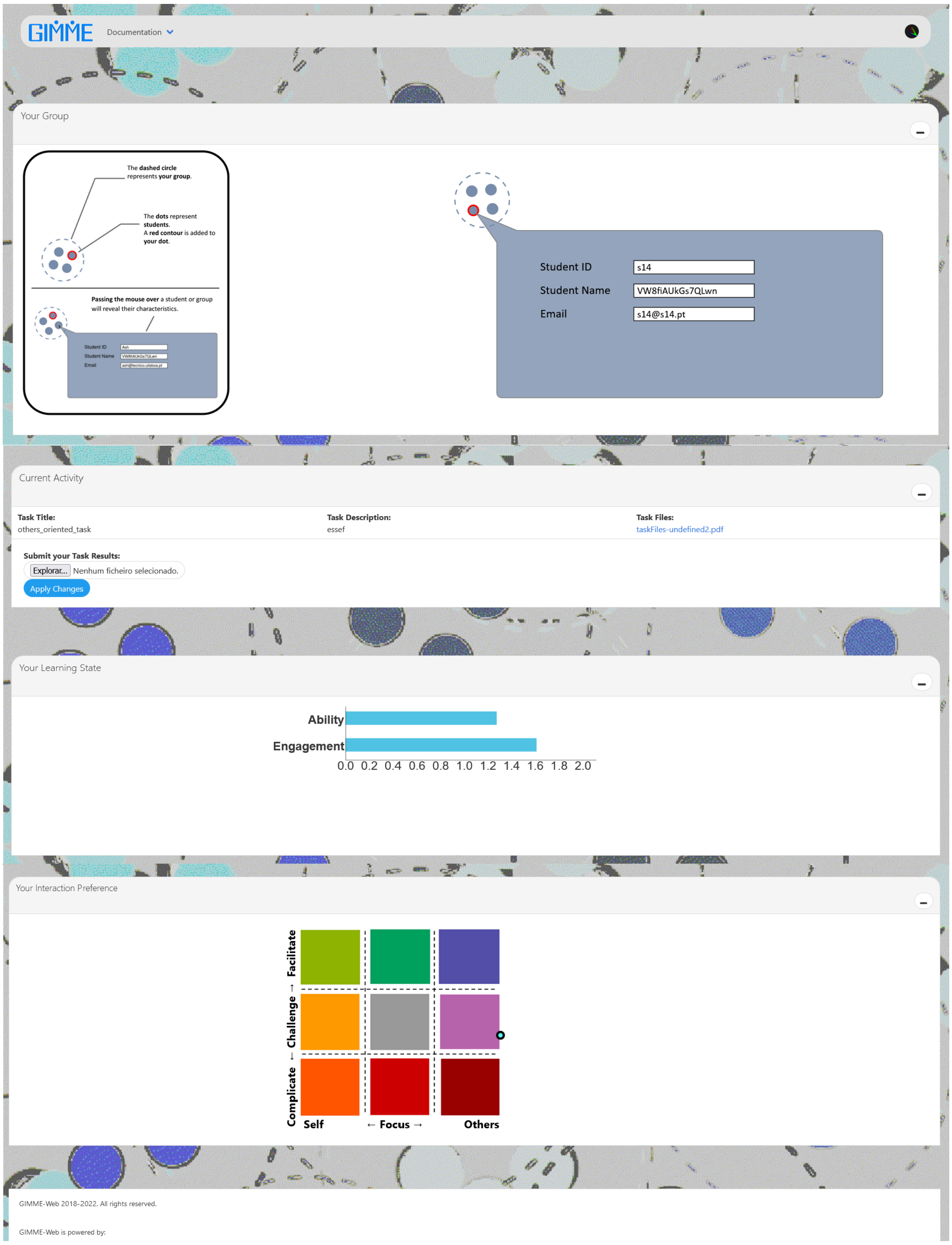
In this tutorial, you will be the professor of a 4 week course. Below, you will find some controls to automate the tutorial. Before the course starts:

1. Share the platform link with the students, so they can sign up in the platform (simulate this by clicking **'Tutorial Controls'** > **'Share link'**);

2. Next, **register the first task** for the course (call it 'week1_00' and make sure it promotes a 'Self-Facilitation' interaction. Also, give the same priority to the task's interaction profile and difficulty when assigning it to a group and, since it is a task for week 1, make it on the easier side);

3. After registering the first task, the other tasks will be automatically registered **by clicking 'Tutorial Controls'** > **'Simulate task registration'**.

Figure 3.8: The Simulations tab, serving as a tutorial for the Professor dashboard. The pie chart in the middle represents the progress in the tutorial. The stronger blue slice represents the current week, in this case week one, while the light blue slice represents the previous weeks. Hovering over the previous weeks' slices, shows the corresponding message for that week.



The **dashed circle** represents **your group**.

The **dots** represent **students**.
A **red contour** is added to **your dot**.

Passing the mouse over a student or group will reveal their characteristics.

Student ID	Ash
Student Name	VW8fiAUkGs7QLwn
Email	ash@tecnico.ulisboa.pt

Student ID	s14
Student Name	VW8fiAUkGs7QLwn
Email	s14@s14.pt

Task Title:	**Task Description:**	**Task Files:**
others_oriented_task	essef	taskFiles-undefined2.pdf

Submit your Task Results:
Explorar... Nenhum ficheiro selecionado.
Apply Changes

GIMME-Web 2018-2022. All rights reserved.

GIMME-Web is powered by:

Figure 3.9: The Student dashboard of GIMME-Web.

4 Evaluation

Contents

In this section, we will go over the tests we made to evaluate the new approaches and their use when integrated with the GIMME-Web program, what we expected to happen and what we actually observed.

4.1 Algorithms validation

After implementing the described approaches, we conducted some experiments, comparing both the previous and the new algorithms in terms of computational efficiency and quality of the solution. For both cases we run a simulation, consisting of: (i) define the number of runs and iterations, create the agents and tasks, each with a random interaction profile, and initialize the approach to be tested; (ii) execute the algorithm with the desired configuration; (iii) after each iteration, simulate how each agent performed according to the coalition structure returned by the algorithm. These experiments where executed on a computer using an AMD Ryzen 5 3600 with 3.6 GHz of clock speed and 16GB of RAM.

4.1.1 Computational efficiency tests

In terms of computational efficiency, we executed one run of 20 iterations with 4, 8, 12, 16, 20 and 24 players for each algorithm (Random, Pure Random Search, Genetic Algorithm (GA), ODP-IP with the KNN regression algorithm, ODP-IP with the tabular regression algorithm, C-Link with the KNN regression algorithm and C-Link with the tabular regression algorithm), with 4 players per group.

What was expected was: ODP-IP is the most complex approach, as it is deterministic and exact, with complexity of O(3^n) and so, it should be the algorithm that takes the longest and should also have worst scaling capabilities for a higher number of agents. As for C-Link, its purpose is to be much faster to find a solution, as it is a greedy approach with a much lower complexity of O(n^3), being in theory even faster than the previous best approach, the Genetic Algorithm.

As we can see from the graph in Table 4.1, for 20 players, ODP-IP is indeed the approach that takes the longest, as expected. However, the C-Link algorithm, although much less complex, for the same number of agents, ends up taking times closer to ODP-IP instead of being even faster than the GA. This repeats again for 24 players, but now with an even bigger increase in execution time. In the case of ODP-IP, this is mostly due to its complexity, as even in the github page with the original implementation of the algorithm in Java, they do not recommend going over 24 agents, as the algorithm would take too long to finish. With our implementation, for 24 players, the ODP-IP implementation of the algorithm in C++ takes $\sim$185 seconds, or $\sim$63% of the total execution time. As for the rest of the execution time, it is spent in the estimation phase, implemented in Python. The same ratio between algorithm and estimation phase, does not apply when looking at the C-Link algorithm. An execution of the C-Link algorithm with 24 agents, takes $\sim$74 seconds, with the C++ portion running for only $\sim$0.42 seconds, i.e. $\sim$0.56% of the total execution time. As we can see, the C-Link algorithm is working as intended, but is being held back

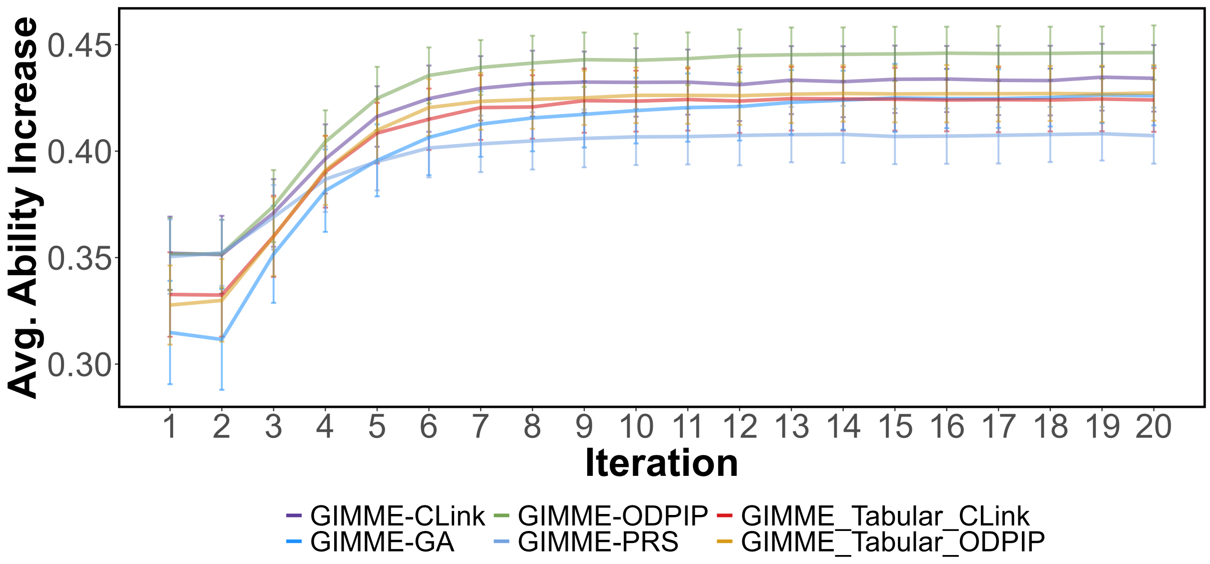


Figure 4.1: Results of executing 120 runs of 20 iterations for each algorithm.

by the estimation phase, making it so this implementation of the C-Link algorithm, which is supposed to be even faster than GA, in practice, many times slower than this approach.

Table 4.1: Time spent in seconds for each implemented algorithm, when run with 4, 8, 12, 16, 20, 24 players and forming groups of four agents.

	4(s)	8(s)	12(s)	16(s)	20(s)	24(s)
Random	5.01E-05	5E-05	0.0001	0.0002	0.0002	0.00025
PRS	0.0635	0.1224	0.1806	0.2405	0.3032	0.3717
GA	-	1.2413	1.7463	2.2762	2.7918	3.4218
ODP-IP	0.0860	0.1547	0.2963	0.7954	6.4920	289.11
Tabular ODP-IP	0.0925	0.1490	0.2355	0.5750	5.7021	290.63
C-Link	0.0573	0.1264	0.2854	0.7756	4.9758	74.390
Tabular C-Link	0.0539	0.1116	0.1932	0.5007	4.3812	71.379

4.1.2 Quality assurance tests

We also performed some tests to evaluate the quality of the solutions of the new algorithms. These tests consisted on executing 120 runs with 20 iterations for every algorithm, and estimating the average ability increase of each player for every iteration. The results of this test were plotted in Figure 4.1.

From this we can clearly see that the approach that achieves the best results is the ODP-IP algorithm, followed by the C-Link algorithm, both using KNN to estimate the profiles of each player. Going into more detail, in the first iterations, the GA algorithm has the worst results, as the estimator it uses to predict the players' state lacks stored data to make reliable and well-founded assertions. The algorithms ODP-IP, C-Link and PRS have similar results at the start, because neither of the approaches has any data to make informed decisions, so they are all in the same situation and any group they choose will have around the same results. They still get better results than GA algorithm, due to assuming that the profile of a coalition is the average of the profiles of its members, adding some information to the early iterations. In the case of ODP-IP and C-Link, as they are deterministic, if no restrictions are applied, they will always give the same answer, so the variation of their results only depended on the profiles generated for the players and how these players react to the coalition they are in. They then both grow at a faster rate than PRS, because they consider more coalitions. For ODP-IP, as it is complete, it considers every coalition that can possibly be formed, and C-Link builds the coalition structure in a greedy manner, choosing at each step the best coalitions to merge, while PRS just generates random coalition structures and chooses the best one, so, in the best case, the answer will be the same as ODP-IP, but in practice, on average, it cannot guarantee as good as an approximation as C-Link.

Looking at the different versions of ODP-IP and C-Link, we can see that, with the tabular characteristic function, the algorithms have worse results than with the KNN characteristic function. The table used for the characteristic function is the same as in Table 3.1, where an agent will prefer to be paired with other agents with the similar interaction preferences. One of the reasons for these results comes from the simplicity of the tabular approach, as the value of a coalition is simply the average of the values of the links between each player of the coalition, where the value of each link is the position in the table that corresponds to the estimated profile of both players in link, while KNN considers how each player reacted to the profiles in previous iterations. This, however, does not the mean that in a real context, these characteristic functions will obtain the same results as in the simulations.

Another important aspect to consider, is the variety of the solutions. With the new approaches being deterministic, we incur the risk of the groups formed being very repetitive, which can be seen as detrimental for the classroom dynamic by some teachers [13]. Looking at the quality of the solutions graph 4.1, we can see that, at around iteration 7/8, for the simulations used for the experiment, the average ability increase stabilizes. At this point, the algorithms have a good enough estimate of the players' interaction profiles and, in the case of the deterministic approaches, as their estimates will not change much from these iterations onward, they return almost always the same coalition structures. Even so, this happens at such a late step that the repetitiveness of the coalitions might not be too relevant, i.e. at the final weeks of the semester, and, before this point, the deterministic approaches give always better results than the other approaches. In the case the professor chooses to maintain the groups, he/she can create restrictions to keep certain students together.

4.2 Usability tests

We also conducted some usability tests, in order to test the usability of the GIMME-Web platform where the algorithms were integrated, which we will describe in more detail in this section.

4.2.1 Method

The objective of this experiment was to evaluate the usability with professors, to also get their opinions about the current implemented functionalities, if they were useful and, if not, what would be needed to make them useful. We believe that these tests, notably the observations we performed, are useful to guide future improvements for (the user interface of) the platform. Our usability tests were conducted with a small number of participants, including both professors and Teacher Assistants (TAs) from different departments, with two observers. We decided to test both professors and TA, because: the professors, having more experience, are more capable at giving feedback about features that might be helpful to manage a class of students; as for TAs, these are mostly assigned to the laboratory classes, where most of the group work occurs, so we were looking for feedback on the applicability of the platform to manage the work done in such classes.

The usability tests started with a brief description of the context the tool was to be used in and of a few concepts that would appear during the experiment, that we felt necessary for the user to know when using the platform. These concepts were the interaction profile (and the focus and challenge dimensions), and the student state, i.e., ability and engagement. These definitions are also present as tooltips in the platform. Following this introduction, we asked the users to read the first page of the questionnaire, that gives instructions on how to answer the questions, and then start the experiment. The experiment proceeded as follows: the user would be asked to follow the tutorial section of the GIMME-Web, as seen previously in Figure 3.8. This tutorial presented a scenario in which the user is a professor of a 6 week course and is asked to do the tasks expected to be done during a semester. To aid the user during the experiment, and also to make the test less time consuming and repetitive, we provided some controls to simulate these tasks, for example, when the user first evaluates one student, we gave the option to simulate the evaluation for the rest of the class automatically. One of the weeks of the simulation is also a "free" week, where the user can change any parameter and test the platform as they wish.

Alongside the tutorial, we asked the user to answer some questions in the questionnaire related to the tasks they had done in each week. We based the post-task questions on Single Ease Question (SEQ) [27] questionnaire. This consists on a single Likert Scale question, with a scale from 1 to 7, on how easy the task was to complete. After the experiment, we also asked the users to answer a post-study questionnaire about their satisfaction with the platform in general. For this purpose we used the System Usability Scale (SUS) [28], constituted by Likert Scale questions, with a scale from 1 to 5, whose final score gives us the overall satisfaction of the platform of the user. This score is computed as (X + Y) * 2.5, where X is the sum of the points for all odd-numbered questions minus 5 and Y is 25 minus the sum of the points for all even-numbered questions. Finally, we have a final open question to allow the users to freely write any additional comment the user might have had. This questionnaire can be consulted in Appendix A. Besides the questionnaire we observed the user interact with the platform, registering unintended actions and criticism they pointed during the experiments.

4.2.2 Results

4.2.2.A Questionnaire

From the questionnaire, we gathered information about the tasks and of the platform as a whole. Starting with the feedback of the tasks, we were trying to understand how difficult they were to complete and reached the conclusion that, for almost all the users, most of the tasks felt straightforward to complete. There were some exceptions however. Even so, the self-reported results from SEQ also demonstrated that there were three tasks that felt slightly difficult (to complete) for at least one of the participants. We will now describe each of these tasks and our reasoning for this difficulty felt by the users:

Select Tasks With how the simulation is structured, there are 4 tasks per week that user needs to select, i.e., add it to the group of tasks that will be assigned to the groups when forming them, while also having to remove the tasks from the previous week. There are buttons that select or remove all tasks at once, but, in this case, the select all button selects all tasks from every week, so the user is forced to select that week's tasks one by one, making more cumbersome to complete.

Create group restrictions This task requires the user to create two restrictions, one separating and one joining two students. This is by far the most complex task in terms of number of clicks. It also requires a not so obvious at first glance series of inputs. Looking at Figure 3.5 as an example, the user must select the students that will be a part of the restriction, have the type of restriction selected and then click on the plus sign in the "Constraints" section.

Change method to create groups This was one of the optional tasks, suggested to experiment with during the free week. By "method to create groups", we mean the group formation and quality estimation algorithms previously mentioned in this thesis. This functionality was incorporated in the Additional Adaptation Parameters, as seen in Figure 3.3. Because these parameters are too advanced for the average user, it is kept hidden and thus more difficult to find. Some of the difficulty may also come from not understanding what each algorithm does or even what is the benefit of changing the approach.

The second part of the questionnaire was about the platform as a whole. Overall, the users found the system easy to use, the functionalities well integrated and the platform was not unnecessarily complex. In the table 4.2, we can see the results for the following questions of the questionnaire:

Question 2 I found the system unnecessarily complex

Question 3 I thought the system was easy to use

Question 5 I found the various functions in this system were well integrated

Table 4.2: Frequency of each answer for questions 2, 3 and 5 of the SUS questionnaire. First row represent options of the scale (complete questionnaire found in Appendix A)

	1	2	3	4	5
Question 2	1	3	1	0	1
Question 3	0	1	2	1	2
Question 5	1	1	0	3	1

As for the scores obtained with the SUS questionnaire, we performed 6 tests with different users and got from lowest to highest, in a scale of 0 to 100: 40.0, 45.0, 67.5, 80.0, 92.5 and 92.5, corresponding to an average of 69.6, i.e., an acceptable score but still requiring some work. We note the lower scores we got, meaning that, in some cases, our platform can still be hard to use to some users.

4.2.2.B Observations

During the experiment, we recorded some of the actions the users did while completing the tasks and also the criticism they made. This feedback will later be used, to improve the platform, adding functionalities and adjusting the currently implemented ones.

One thing to note that is exclusive to the experiment is the fact that the tutorial section of the dashboard is located at the top of the page. This is where the instructions for the experiment are located and so, the user would often need to scroll up to see the next step and would also make it so they wouldn't be able to see the instructions while executing the tasks. The extra time and actions just to check what was asked of the users, is not relevant to the usability of the platform, and thus not taken into consideration. The tests also occurred in a controlled environment, and the users could take the time they needed and could also ask for any clarification if stuck in some step.

We will now go over our observations. One of the most common issues we noticed was the user not noticing the select all button in the student tab. This was due to the position of the button, being in the opposite side of where the students to select were, and also the color of the button that, as seen in Figure 3.4 is the same color as the background.

Another common issue observed, was related with the functionality to manually evaluate a student. To accomplish this, we need to click the "Manually evaluate student" button, located in the students setup, also in Figure 3.4. What actually happened was, when asked to evaluate a student, the users went first to the group sections and looked over there. After failing to find the button they returned to the student setup tab and, even then, they took some time to find the button. With the users comments, we reached the conclusion that the button was not where they expected and was also not clear that that was the button, as only when hovering over it can the tooltip be seen. Some users also pointed out how awkward it is to select each week tasks. If we want to select more than one task per week, but not all of them, we have to select them one by one, a time consuming process.

Another common observation was in relation to the interaction profiles of the formed groups. These are indicated in the groups by the color of the traced circle around the students, as seen in Figure 3.6. Additionally, the groups appear as points in the “Overall Performance Distribution” tab, as seen in Figure 3.7, and the colors of the circles correspond to the squares they land on in this graph. The problem observed was that the users did not relate the points to the groups or, if they did, which group corresponded to each point. It is important to note that, when hovering over a group, it would tell its position in graph, but due to the inexperience of the users, this was not enough to easily identify the groups in the graph. In regards to the colors of the circles that represent the students, these change depending on their performance and on the currently selected scale, either absolute or relative, as mentioned in section 3.6. However, the relative scale was not easily understood by most users, with one user pointing out that, with the default selection being the relative scale, which has both extremes equal to 0%, it caused even more confusion. Other users also commented that the relative scale might make it seem that some students might be having some difficulties with the material, when they can have over 80% ability, but be the lowest grade overall, therefore, appearing as red.

Still in relation to the groups, some users questioned the position where these appeared. The groups position in the section is completely random, and, so, they can appear over other groups or even close to the edge, which causes the pop-up with the group’s information to not be visible. This problem could be solved by placing each of the formed groups in a specific position every time, or making it so the boxes appear only where they can be completely seen.

In regards to the previously mentioned restriction creation functionality, besides the “creating a restriction” procedure already described, after creating a restriction, we noticed that most people went back to double check if the restriction was successfully created, as there was no indication that it was. This was also aided by the fact that, in contrast with other pages, this was a pop-up that didn’t have a confirmation button.

The feedback given by the users also included some suggestions of the changes we could implement to make GIMME-Web more user friendly. Some of the more common suggestions can be found in Appendix B

4.3 Discussion

In this section we will go over the results of the tests, what we expected to occur versus what actually happened, and the conclusions we can take from the experiments.

Firstly, looking at the performance tests, the ODP-IP algorithm is the algorithm with the worst scaling, taking the longer than the other algorithms when the number of students increases. This is indeed what we expected and can be explained by the complexity of the algorithm, being O(3^n), i.e. exponential in the number of agents. Conversely, the C-Link approach did not behave as expected. The main advantage of this approach was how fast it can find a good enough solution in a short amount of time. In practice this was not what happened, due to the estimation phase that both deterministic approaches need to execute in order to have the values for the coalitions. In the case of C-Link, this part of the algorithm takes over 99% of the execution time and so, with the current implementation, makes the C-Link approach almost obsolete. During the development phase of these algorithms, we raised the possibility of implementing the GIMME framework in C++, to help solve the performance problems of the more complex approaches. With this change, the C-Link algorithm would be able to fulfil its role of handling cases with a higher number of students.

Going over the quality of solution tests, we can see that the deterministic approaches achieve higher or equal average ability increase than the other algorithms. This is due to the deterministic approaches considering more coalitions than the other algorithms, and in the case of the ODP-IP, every possible coalition, so these results are expected. In the case of the tabular characteristic function, it still achieves less favorable results than the KNN characteristic function. This could have happened, because the table used for the simulations might not have been the most optimal one, or the fact that this approach does not consider previous approaches when assigning a value to a coalition. Nevertheless, one advantage of the tabular function, is that it requires less memory accesses, so it is a bit faster when the algorithms are executed through the GIMME-Web platform.

In conclusion, based on how the algorithms are currently implemented, we can consider two scenarios:

20 students or less : With this number of students, ODP-IP does not take a long time to compute a solution and, as it always achieves results with the most average ability increase, it should be the best choice.

More than 20 students : At this stage, ODP-IP and C-Link start to take much longer due to their poor scaling, the estimation phase specifically for C-Link, so they cannot practically be used. So the best choice for the algorithms should be the PRS algorithm for the first couple of iterations due to the initial solutions having higher quality, followed by GA when it starts to outperform PRS.

If C-Link had the expected performance, it would make it a valid option for a higher number of students. Unfortunately it is not case, so for now, the PRS and GA are the better options for cases with more students. What can be done instead is, when using the platform, we can change the algorithms implicitly depending on the situation as, from the results of the user tests, the “Additional Adaptation Parameters” section is not very user friendly and, so, most of the users will not use this functionality efficiently.

Overall, from the usability tests, we expected to get feedback to improve the professor dashboard of the GIMME-Web platform, as it was being improved upon alongside the integration of the new algorithms. With the feedback obtained, we were able to identify some issues in the platform, including problems with the interface itself, i.e., some functionalities are not to obvious to use, and some functionalities that are still missing.

5 Conclusion

Contents

In this section, we will go over our findings and if we were able to accomplish our goal of discovering an algorithm that can help us in the group formation of students while also considering their interaction preferences.

5.1 Conclusions

Our main objective was to find algorithms that could help us form groups of students, while considering the different interaction profiles these students might have. After researching and comparing different approaches, we implemented two grouping algorithms and one quality estimation algorithm to the GIMME framework, a model developed for managing collective learning, and that served as a basis to our implementations and exploratory study. Later, it was integrated in the GIMME-Web platform, an interface through which, professors use the algorithms to form groups. From the simulations conducted on these algorithms, we reached very promising results both in terms of time and quality, with ODP-IP, for small-medium sized classes, i.e., 20 students, took an acceptable time to complete while also providing the best results of all the algorithms on average. The specifics of ODP-IP that made these results possible is it being complete, i.e. it searches the entire subspace of possible coalitions, and it is always able to find the best coalition for a given characteristic function. This is especially relevant in school contexts, where deterministic algorithms have not been used much in opposition to stochastic approaches, as mentioned by Cruz et al. [11]. Nevertheless with these results we hope to show that deterministic approaches can still be a viable way to compute groups in a school environment.

5.2 System limitations and future work

During the implementation of the algorithms we noticed some limitations of our system. Python, at least for the amount of computations required for more complex approaches, such as the new algorithms, is too slow to allow an acceptable computation time. We first tried to alleviate the problem by implementing the new approaches in C++, and using them as external modules in Python, delegating a great portion of the computations to a much faster language. This, however, did not solve the problem, as one computation-heavy part of the algorithms, where we determine the value of each coalition, was still implemented in Python, with no option to extract it to C++ due to how the rest of the project was implemented. For this thesis, unfortunately, there was not enough time to change all of the GIMME framework to a faster language and, so, one of the approaches ended up not having its expected results. In the future, we hope to implement the majority of the platform to C++, where our new deterministic approaches can work more efficiently. With this change, it might be also possible to have C-Link handle restrictions, like the rest of the approaches.

In regards to the GIMME-Web platform, with the feedback garnered from the usability tests, we noticed various of this system's limitations and we hope to change them accordingly, to have a more user-friendly platform.

Bibliography

[1] T. Michalak, T. Rahwan, E. Elkind, M. Wooldridge, and N. R. Jennings, “A hybrid exact algorithm for complete set partitioning,” *Artificial Intelligence*, vol. 230, pp. 14–50, 2016.

[2] A. Farinelli, M. Bicego, F. Bistaffa, and S. D. Ramchurn, “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.

[3] T. Voice, S. D. Ramchurn, and N. R. Jennings, “On coalition formation with sparse synergies,” in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. Citeseer, 2012, pp. 223–230.

[4] F. Bistaffa, A. Farinelli, J. Cerquides, J. A. Rodríguez-Aguilar, and S. Ramchurn, “Anytime coalition structure generation on synergy graphs,” 2014.

[5] T. Rahwan, T. P. Michalak, M. Wooldridge, and N. R. Jennings, “Coalition structure generation: A survey,” *Artificial Intelligence*, vol. 229, pp. 139–174, 2015.

[6] J. H. Tomkin, S. O. Beilstein, J. W. Morphew, and G. L. Herman, “Evidence that communities of practice are associated with active learning in large stem lectures,” *International Journal of STEM Education*, vol. 6, no. 1, pp. 1–15, 2019.

[7] Z. Han and H. V. Poor, “Coalition games with cooperative transmission: a cure for the curse of boundary nodes in selfish packet-forwarding wireless networks,” *IEEE transactions on communications*, vol. 57, no. 1, pp. 203–213, 2009.

[8] E. Y. Bitar, E. Baeyens, P. P. Khargonekar, K. Poolla, and P. Varaiya, “Optimal sharing of quantity risk for a coalition of wind power producers facing nodal prices,” in *2012 American Control Conference (ACC)*. IEEE, 2012, pp. 4438–4445.

[9] “The protégé effect: How you can learn by teaching others.” [Online]. Available: https://effectiviology.com/protege-effect-learn-by-teaching/

[10] R. M. Felder and R. Brent, "Cooperative learning," *Active learning: Models from the analytical sciences*, vol. 970, pp. 34–53, 2007.

[11] W. M. Cruz and S. Isotani, "Group formation algorithms in collaborative learning contexts: A systematic mapping of the literature," in *CYTED-RITOS International Workshop on Groupware*. Springer, 2014, pp. 199–214.

[12] S. Gomes, J. Dias, and C. Martinho, "Gimme: Group interactions manager for multiplayer serious games," in *2019 IEEE Conference on Games (CoG)*. IEEE, 2019, pp. 1–8.

[13] J. Budden, "Grouping students," 2008. [Online]. Available: https://www.teachingenglish.org.uk/article/grouping-students

[14] T. Knez, M. H. Dlab, and N. Hoic-Bozic, "Implementation of group formation algorithms in the elars recommender system." *International Journal of Emerging Technologies in Learning*, vol. 12, no. 11, 2017.

[15] A. Björklund, T. Husfeldt, and M. Koivisto, "Set partitioning via inclusion-exclusion," *SIAM Journal on Computing*, vol. 39, no. 2, pp. 546–563, 2009.

[16] T. Rahwan, S. D. Ramchurn, V. D. Dang, A. Giovannucci, and N. R. Jennings, "Anytime optimal coalition structure generation," in *AAAI*, vol. 7, 2007, pp. 1184–1190.

[17] T. Rahwan, S. D. Ramchurn, N. R. Jennings, and A. Giovannucci, "An anytime algorithm for optimal coalition structure generation," *Journal of artificial intelligence research*, vol. 34, pp. 521–567, 2009.

[18] D. Y. Yeh, "A dynamic programming approach to the complete set partitioning problem," *BIT Numerical Mathematics*, vol. 26, no. 4, pp. 467–474, 1986.

[19] T. Rahwan and N. R. Jennings, "An improved dynamic programming algorithm for coalition structure generation," in *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3*, 2008, pp. 1417–1420.

[20] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohmé, "Coalition structure generation with worst case guarantees," *Artificial intelligence*, vol. 111, no. 1-2, pp. 209–238, 1999.

[21] T. Rahwan, T. P. Michalak, E. Elkind, M. Wooldridge, and N. R. Jennings, "An exact algorithm for coalition structure generation and complete set partitioning." Citeseer, 2013.

[22] Y. Bachrach, R. Meir, K. Jung, and P. Kohli, "Coalitional structure generation in skill games," in *Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[23] H. Aziz and B. De Keijzer, “Complexity of coalition structure generation,” *arXiv preprint arXiv:1101.1007*, 2011.

[24] N. Ohta, V. Conitzer, R. Ichimura, Y. Sakurai, A. Iwasaki, and M. Yokoo, “Coalition structure generation utilizing compact characteristic function representations,” in *International Conference on Principles and Practice of Constraint Programming*. Springer, 2009, pp. 623–638.

[25] V. Conitzer and T. Sandholm, “Complexity of constructing solutions in the core based on synergies among coalitions,” *Artificial Intelligence*, vol. 170, no. 6-7, pp. 607–619, 2006.

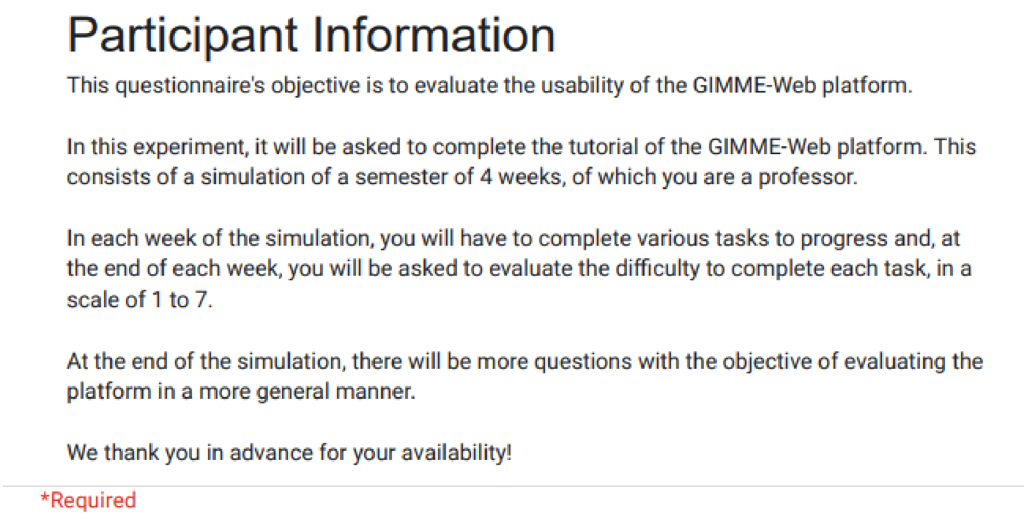
[26] S. Järvelä, M. Veermans, and P. Leinonen, “Investigating student engagement in computer-supported inquiry: A process-oriented analysis,” *Social Psychology of Education*, vol. 11, no. 3, pp. 299–322, 2008.

[27] J. Sauro, “If you could only ask one question, use this one,” 2010. [Online]. Available: https://www.measuringu.com/single-question

[28] J. Brooke *et al.*, “Sus-a quick and dirty usability scale,” *Usability evaluation in industry*, vol. 189, no. 194, pp. 4–7, 1996.

A

Questionnaire for user tests



Participant Information

This questionnaire's objective is to evaluate the usability of the GIMME-Web platform.

In this experiment, it will be asked to complete the tutorial of the GIMME-Web platform. This consists of a simulation of a semester of 4 weeks, of which you are a professor.

In each week of the simulation, you will have to complete various tasks to progress and, at the end of each week, you will be asked to evaluate the difficulty to complete each task, in a scale of 1 to 7.

At the end of the simulation, there will be more questions with the objective of evaluating the platform in a more general manner.

We thank you in advance for your availability!

*Required

Voluntary Participation and Data Safety

Before we start, make sure you understand and consent the following terms:

1 - You have read and understood the meaning of this study. You have the opportunity to ask questions, if necessary, and collect the respective answers;
2 - You understand that participation in this study is voluntary and that you can withdraw at any time without giving any explanation. If this happens, you will not be penalized and the data relating to your experiment will be removed and destroyed;
3 - You authorize the recording of data during the session. The data will be archived in private clouds (Drive) and private physical memories, and will be destroyed in accordance with the General Data Protection Regulation;
4 - You authorize the collection and processing of your anonymized data within the scope of this project for the purposes of analysis, investigation and dissemination of results in scientific publications or conferences in the project area;
5 - You authorize your participation in this study and accept its conditions.

The lead researcher responsible for this study is Prof. Dr. Carlos Martinho:

carlos.martinho@tecnico.ulisboa.pt

Intelligent Agents and Synthetic Characters Group - INESC-ID
Instituto Superior Técnico, Campus Taguspark
Av. Prof. Doutor Cavaco Silva, Room 2N7.21
2744-016 Porto Salvo, Portugal

If you consent to participate, feel free to proceed to the next step! We appreciate your help!

1. Age *

2. Gender *

Mark only one oval.

- ◯ Male
- ◯ Female
- ◯ Prefer not to say
- ◯ Other: ______________________

Course Preparation

Please answer, in a scale of 1 to 7, how easy it was to complete the following tasks.

3. Register in the platform *

Mark only one oval.

	1	2	3	4	5	6	7	
Very Difficult	◯	◯	◯	◯	◯	◯	◯	Very Easy

4. Create a task *

Mark only one oval.

	1	2	3	4	5	6	7	
Very Difficult	◯	◯	◯	◯	◯	◯	◯	Very Easy

Week 1

Please answer, in a scale of 1 to 7, how easy it was to complete the following tasks.

5. Select students *

Mark only one oval.

	1	2	3	4	5	6	7	
Very Difficult	◯	◯	◯	◯	◯	◯	◯	Very Easy

6. Select Tasks *

Mark only one oval.

	1	2	3	4	5	6	7	
Very Difficult	◯	◯	◯	◯	◯	◯	◯	Very Easy

7. Create the groups *

Mark only one oval.

	1	2	3	4	5	6	7	
Very Difficult	◯	◯	◯	◯	◯	◯	◯	Very Easy

8. Evaluate a student *

Mark only one oval.

	1	2	3	4	5	6	7	
Very Difficult	◯	◯	◯	◯	◯	◯	◯	Very Easy

Week 2

Please answer, in a scale of 1 to 7, how easy it was to complete the following tasks.

9. Remove a student *

Mark only one oval.

	1	2	3	4	5	6	7	
Very Difficult	◯	◯	◯	◯	◯	◯	◯	Very Easy

Week 3

Please answer, in a scale of 1 to 7, how easy it was to complete the following tasks.

10. Create group restrictions *

Mark only one oval.

	1	2	3	4	5	6	7	
Very Difficult	◯	◯	◯	◯	◯	◯	◯	Very Easy

Week 4

Please answer, in a scale of 1 to 7, how easy it was to complete the following tasks.
As this is a free week, answer only about the tasks you have completed.

11. Adjust group sizes

Mark only one oval.

	1	2	3	4	5	6	7	
Very Difficult	◯	◯	◯	◯	◯	◯	◯	Very Easy

12. Change method to create groups

Mark only one oval.

	1	2	3	4	5	6	7	
Very Difficult	◯	◯	◯	◯	◯	◯	◯	Very Easy

Usability of GIMME-Web

The following questions' objective is to evaluate the usability of the platform in a more general manner.
While answering, please record your immediate response to each item.

Thank you!

13. I think that I would like to use this system frequently *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

14. I found the system unnecessarily complex *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

15. I thought the system was easy to use *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

16. I think that I would need the support of a technical person to be able to use this system *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

17. I found the various functions in this system were well integrated *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

18. I thought there was too much inconsistency in this system *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	◯	◯	◯	◯	◯	Strongly Agree

19. I would imagine that most people would learn to use this system very quickly *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

20. I found the system very cumbersome to use *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

21. I felt very confident using the system *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

22. I needed to learn a lot of things before I could get going with this system *

Mark only one oval.

	1	2	3	4	5	
Strongly Disagree	○	○	○	○	○	Strongly Agree

Additional Comments

23. If you have any comment that you like to make about the platform, please write it in the text box below.

B Suggestions given by the users for GIMME-Web

In this section, we will list some of the most common suggestions given by the users from the user tests:

"Select all" button One suggestion we got was to change the color of the "select all" button to blue, to also be consistent with other buttons found throughout the platform. This change will also be applied to most other buttons.

Manually evaluate students The most common suggestion we got was to move the button to evaluate the students to the groups section.

Selecting tasks each week The most common suggestion given to help select the tasks each week was to have a filter to only show the tasks of a specific week, so the users could use the "select all" button to instantly select all the needed tasks.

Showing students performance To more clearly show how the students are performing during the semester, it was suggested to use as default the absolute scale, instead of the relative scale.

Identification of the groups and their interaction profiles To more easily know the interaction profile of each group, it was suggested we identified the points in the “Overall Preferences Distribution” graph, for example, with the group id, and to make it so we could look at the groups and graph at the same time, currently not possible with the page layout. One user also pointed out that, for colorblind people, the current information displayed would be really difficult to understand, and the identification of the points would also help in such cases.

Creating restrictions The confirmation button mentioned in the Observations section, section 4.2.2.B was, in fact, the most common suggestion to indicate the restriction was successfully created.