

An Attention Model for the Formation of Collectives in Real-World Domains

Adrià Fenoy,^{1,2} Filippo Bistaffa,¹ Alessandro Farinelli²

¹University of Verona

²IIIA-CSIC

adria.fenoybarcelo@univr.it, filippo.bistaffa@iia.csic.es, alessandro.farinelli@univr.it

Abstract

We consider the problem of forming collectives of agents for real-world applications aligned with *Sustainable Development Goals* (e.g., shared mobility, cooperative learning). We propose a general approach for the formation of collectives based on a novel combination of an *attention model* and an *integer linear program* (ILP). In more detail, we propose an attention encoder-decoder model that transforms a collective formation instance to a *weighted set packing* problem, which is then solved by an ILP. Results on two real-world domains (i.e., ridesharing and team formation for cooperative learning) show that our approach provides solutions that are comparable (in terms of quality) to the ones produced by state-of-the-art approaches specific to each domain. Moreover, our solution outperforms the most recent general approach for forming collectives based on *Monte Carlo tree search*.

1 Introduction

In recent years, more and more scenarios require *Collective Intelligence* solutions enabling novel ways of social production, promoting innovation and encouraging the exchange of ideas (European Commission 2021). Such new forms of collaborative consumption and production rely on a common and fundamental task, i.e., *the formation of collectives*. This task plays a crucial role in a general class of real-world applications aligned with the *UN Sustainable Development Goals*, which enable agents to complete tasks and achieve benefits by means of cooperation.

As an example of this class of domains, in *ridesharing* commuters can form groups and travel together with the objective of reducing transportation costs, mitigate pollutant emissions and alleviate the traffic congestion in urban environments (Bistaffa et al. 2021; Alonso-Mora et al. 2017). Another prominent example can be found in modern educational institutions that aim at implementing cooperative and active learning techniques, which engage students in *teams* to participate in all learning activities in the classrooms. Indeed, as recently shown in (Andrejczuk et al. 2019) collective formation approaches can improve the overall performance of the students by grouping them in teams that maximize the synergies among members.

Due to the inherent complexity and specificity of each application domain, researchers usually tackle the formation of collectives by designing very specific sub-optimal approaches that can solve the associated large-scale optimization problem in a feasible runtime. Unfortunately, one domain-specific approach usually cannot be applied in a different scenario.

In contrast, in this paper, we propose a novel, general approach for the formation of collectives that is based on two fundamental steps. First, we apply deep reinforcement learning techniques to train an attention encoder-decoder model, with the objective of automatically generating a set of *promising collectives* on the basis of the structure of the considered scenario. In the second step, we compile a *weighted set packing* (WSP) instance that, by only taking into account the promising candidates generated in the first step, can be solved by off-the-shelf ILP solvers in a manageable time budget. Thus, our approach does not require to manually specify any domain-specific knowledge, in contrast with the above-mentioned sub-optimal state-of-the-art approaches. Furthermore, by only considering a set of promising candidates rather than the entire set of possible collectives¹, we reduce the complexity of the original problem by several orders of magnitude without sacrificing the quality of the final solution that we produce.

As such, this paper advances the state-of-the-art as follows:

- We propose a general approach for the formation of collectives in real-world domains based on the novel combination of an attention model and WSP.
- We proposed a novel training procedure for our attention model based on *Maximum Entropy Reinforcement Learning*. In contrast to previous approaches which use attention based models for optimization (Kool, van Hoof, and Welling 2019), our solution achieves a wide variety of promising candidates. Such variety is a key feature that allows the ILP solver to group collectives of high value.
- We evaluate our approach in two real-world scenarios (i.e., ridesharing and team formation for cooperative learning) by comparing it with state-of-the-art ap-

¹The number of possible collectives grows exponentially with the number of agents, hence it is not manageable in real-world applications that involve more than a few tens of agents

proaches specific to each domain. Our results show that our approach can produce, in some settings, solutions of comparable quality without requiring any domain-specific knowledge. Moreover, we compare our approach with the most recent general approach for forming collectives based on *Monte Carlo tree search* (Wu and Ramchurn 2020), showing that our solutions clearly outperform (in terms of quality) the ones computed by the counterpart.

2 Background & Related Work

2.1 Formation of Collectives

The general problem of forming collectives of agents has been deeply studied from many different perspectives in the scientific literature. In this paper we specifically focus on the optimization problem (Cerquides et al. 2014) of computing the set of non-overlapping collectives (i.e., subsets) of agents belonging to a universal set A . Each collective is associated to a value provided by a domain-specific utility function, e.g., the reduction in terms of cost or CO₂ emissions associated to the arrangement of a shared trip (Bistaffa et al. 2021) or the improvement thanks to cooperation within a team of students (Andrejczuk et al. 2019). Thus, our objective is to maximize the sum of the values associated with each formed collective.

By and large, the formation of collectives requires solving a *set partitioning* problem (Lin 1975) or, equivalently, a *coalition structure generation* problem (Michalak et al. 2016). A wealth of approaches have been proposed to tackle the formation of collectives from this perspective (Rahwan et al. 2015). General algorithms (Changder et al. 2020; Michalak et al. 2016) usually make no assumptions on the structure of the utility function, which is treated as a *black-box* oracle. Unfortunately, the mere act of providing the input to the solution algorithm (without even considering the runtime of the algorithm itself) requires enumerating a number of values that grows exponentially with the number of agents and hence becomes quickly not manageable. Notice that in many application domains it is possible to exploit domain specific constraints that limit the number of possible collectives. For example, cardinality constraints that limits the maximum size of the collectives to k naturally arise in ridesharing and team formation. However, even considering such a constraint the number of collectives is $O(|A|^k)$, which can require *hours* to enumerate when the computation of the utility function is complex.

To overcome this limitation, the formation of collectives in real-world domains is usually tackled by means of sub-optimal approaches that trade generality for scalability, i.e., that exploit the specific structure of the considered domain to compute good-quality solutions in a feasible amount of time. For instance, (Bistaffa et al. 2021) proposed a solution algorithm for large-scale ridesharing that, by heavily relying on the greedy nature of the domain, is capable of computing solutions of very good quality for hundreds of agents within one minute. Unfortunately, this approach cannot be applied in collective formation domains that are not characterized by such a greedy nature, e.g., the team forma-

tion domain discussed in (Andrejczuk et al. 2019). Here, the authors proposed a local-search algorithm that, once again, heavily relies on the structure of the problem and the considered dataset.

Against this background, in this paper we propose an approach for the formation of collectives that does not require to manually specify any domain-specific knowledge but aims at learning such structure by applying deep reinforcement learning techniques.

2.2 Machine Learning for Optimization

The use of machine learning techniques to solve combinatorial optimization problems is a recent yet very active topic that has received a lot of attention during the last years. According to (Bengio, Lodi, and Prouvost 2020), machine learning can contribute to the optimization field in a twofold way: i) replace some heavy computations by building fast approximations and ii) improve the optimization approach by learning domain-specific structure.

In the first and most common case, machine learning is employed to train a model so as to “imitate” the behavior of the original algorithm in terms of solution quality, but being much faster. Relevant examples are the work of (Kool, van Hoof, and Welling 2019), which proposes a machine learning approach based on an attention model to solve several variations of the *vehicle routing problem* (VRP) or the work of (Nair et al. 2021), which proposes a deep learning model to solve *mixed-integer linear programs* (MILPs).

The second direction mentioned by (Bengio, Lodi, and Prouvost 2020) is more aligned with our work. Indeed, we consider a scenario where the algorithmic decisions (in our case, deciding what are the best collectives to form) rely on highly specific hard coded knowledge that is difficult and costly to acquire. Thus, our goal is to employ machine learning to automatically exploit the structure of the domain and learn the best performing behavior (i.e., a policy) to guide the formation of collectives.

In the following section we discuss our proposed approach that aims at achieving this objective.

3 Our Solution Approach

Given a pool of n agents $A = \{a_1, a_2, \dots, a_n\}$, we tackle the formation of collectives as the problem of computing the best set S of non-overlapping subsets of A which optimizes the sum of the values associated to each subset $S \in \mathcal{S}$ by a utility function $f : \mathcal{F}(A) \rightarrow \mathbb{R}$. Such utility function maps every collective in the feasible set² of collectives $\mathcal{F}(A)$ to a real number. This problem can be naturally formulated as an ILP:

$$\begin{aligned} & \text{maximize} && \sum_{S \in \mathcal{F}(A)} f(S) \cdot x_S, \\ & \text{subject to} && \sum_{S \in \mathcal{F}(A)} b_{i,S} \cdot x_S \leq 1, \quad \forall a_i \in A, \end{aligned} \tag{1}$$

²Depending on the considered domain, such a set of feasible collectives can be the entire set of subsets of A or, for example, the set of all collectives that satisfy a given constraint (e.g., a cardinality constraint).

where x_S is a binary decision variable that encodes whether collective S is in the set \mathcal{S} and $b_{i,S}$ is a binary value that encodes whether agent $a_i \in A$ belongs to the collective S .

The problem in (1) can be easily recognized as a WSP and, for small-scale instances (i.e., A with less than a couple of tens of agents), directly solved as an ILP by means of off-the-shelf solvers. On the other hand, in real-world scenarios the *generation* of such an ILP (let alone its solution) can require hours of computation, due to the necessity of enumerating all feasible collectives. This complexity is further increased in scenarios where determining each value $f(S)$ requires a significant computational effort (Andrejczuk et al. 2019).

On the other hand, usually, by exploiting the inherent structure of the domain, an expert might propose a reduced set of *promising* collectives $\mathcal{R}(A)$, from which a sub-optimal solution of high quality can be obtained. Following this approach, in this paper we propose an attention-based model that learns this structure to generate an *ILP of manageable size*, i.e.,

$$\begin{aligned} & \text{maximize} && \sum_{S \in \mathcal{R}(A)} f(S) \cdot x_S, \\ & \text{subject to} && \sum_{S \in \mathcal{R}(A)} b_{i,S} \cdot x_S \leq 1, \quad \forall a_i \in A. \end{aligned} \quad (2)$$

In Figure 1 we provide an overall scheme that illustrates our approach for the formation of collectives.

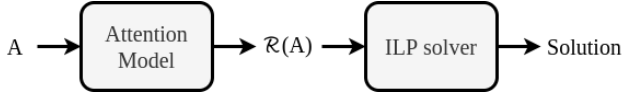


Figure 1: Proposed approach for the formation of collectives. The attention model generates a reduced set of collectives from which an ILP solver obtains the solution.

Following a standard practice (Bistaffa et al. 2021), we assume that the entire approach depicted in Figure 1 is provided with a time budget t in which a solution has to be computed. Notice that, since our approach comprises two phases, we need to distribute such a time budget for each of the two steps, hence we assign a runtime of $k \cdot t$ to the first phase (i.e., generation of the WSP instance via the attention model) and the remaining part $(1 - k) \cdot t$ to the solution of such instance by the ILP solver.

3.1 Attention Model

Our attention model can be considered as a decision-making process, where collectives are built incrementally by selecting elements from the set of agents A and adding them to the collective. Our model receives the set A as a list of d_x dimensional vectors, where d_x is the number of features, and a binary encoding of a collective $S = \{b_{1,S}, b_{2,S}, \dots, b_{n,S}\}$, where $b_{i,S}$ are binary values determining whether the respective agents a_i are in the collective or not. Therefore, the state of the problem at each step is represented by the tuple $s = (A, S)$

Given a state s , we design an attention-based encoder-decoder model based on the one proposed by (Vaswani et al. 2017) which defines a stochastic policy $\pi_\theta(s)$ parameterized by θ , determining the probability for each element in the pool of agents A to be included in the collective S . The encoder produces an embedding for each element in the pool. Then, as illustrated in Figure 2, the decoder receives the embedding and the collective in order to compute the probabilities.

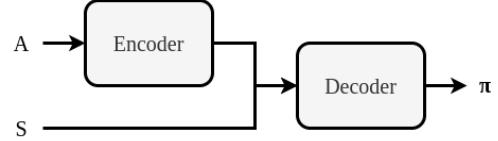


Figure 2: The encoder-decoder approach computes the probability π_θ for each agent in A to be added to the collective S .

Encoder Our encoder is similar to the one in the architecture discussed by (Vaswani et al. 2017). In contrast with the original model, we omit positional encoding, since the order of the elements in the pool of agents is not relevant for the formation of collectives. Nevertheless, we use an input feed-forward layer to encode elements in the pool of agents A from its d_x dimensional feature representation to a d_h dimensional embedding before main attention blocks. To get the encoded representation of the pool of agents \mathbf{h}_A , the input embeddings are updated using N attention blocks depicted in Figure 3, each one consisting of two sub-layers: a multi-head self-attention and a feed-forward layer. Each sub-layer adds a residual connection (He et al. 2016) and performs layer normalization (Ba, Kiros, and Hinton 2016) on its outputs, i.e., $\text{LayerNorm}(x + \text{sub-layer}(x))$. To facilitate residual connections, all sub-layers in the encoder use the same dimensionality d_h .

Decoder In order to compute the probabilities $\pi_\theta(s)$, the decoder performs attention between the encoded pool $\mathbf{h}_A = \{\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_n\}$ and an encoding of the collective \mathbf{h}_S . To obtain this encoding, the following reduction is applied on the encoded pool:

$$\mathbf{h}_S = \frac{\sum_i b_{i,S} \cdot \mathbf{h}_i}{\sum_i b_{i,S}}. \quad (3)$$

At the initial state the collective is empty, which means that $\sum_i b_{i,S} = 0$, in that case we use a d_h dimensional learnable parameter \mathbf{v} as a placeholder, $\mathbf{h}_S = \mathbf{v}$.

Finally, to obtain the probabilities $\pi_\theta(s)$, the decoder performs two last attention steps. The first one computes attention between \mathbf{h}_A and \mathbf{h}_S to obtain a combined encoding \mathbf{h}' . The second step computes the compatibility u_i between \mathbf{h}_S and \mathbf{h}' ,

$$u_i = \frac{(\mathbf{h}_S W^q)(\mathbf{h}' W^k)^T}{\sqrt{d_h}}, \quad (4)$$

where W^q and W^k are two learnable linear transformations and the output is scaled with a factor of $\frac{1}{\sqrt{d_h}}$. The compati-

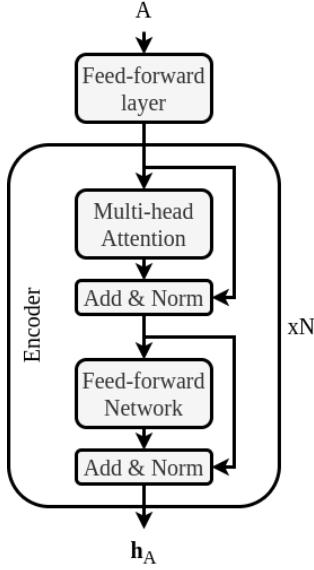


Figure 3: Encoder architecture.

bility is normalized by applying a softmax in order to obtain the probability of each agent being added to a collective S :

$$\pi_{\theta,i}(s) = \frac{e^{\gamma \tanh u_i}}{\sum_j e^{\gamma \tanh u_j}}, \quad (5)$$

where the function $\gamma \tanh u_i$ is applied onto the compatibilities so as to control the exploration of the model by adjusting parameter γ .

Maximum Entropy Policy Gradient In the previous section we defined the attention model for computing the probabilities $\pi_{\theta}(s)$ for the formation of collectives modeled as a decision making process. In this section, we elaborate on how we optimize the parameters θ for this task.

In the context of such a discussion, it is important to recall our ultimate goal: forming a set of collectives $\mathcal{R}(A)$ from which, by means of an ILP solver, it can be obtained a solution of good quality for a particular instance of a collective formation problem. For our approach to be effective, we have to guarantee (i) that $\mathcal{R}(A)$ contains collectives associated to high utility values by the function f , but also that (ii) such set contains a sufficient number of diverse collectives. Such a diversity is fundamental because, due to the presence of the non-overlapping constraint in (1), the optimal solution is likely to contain, not only collectives with the highest possible value, but also collectives of lower value. Henceforth, providing a sufficient number of alternatives to the ILP solver is crucial to achieve a final solution of good quality. Along these lines, we define the loss:

$$\mathcal{L}(\theta|s) = \mathbb{E}_{\pi_{\theta}(S|s)} [f(S)] + \tau \mathcal{H}(\pi_{\theta}(s)), \quad (6)$$

where $\mathcal{H}(\pi_{\theta}(s))$ is the entropy of the model at state s and τ is a temperature parameter. Optimizing the first term in (6) produces a policy which builds collectives of high utility. In addition, we consider a second entropy term to the loss, whose objective is to foster diversity.

Similar to (Kool, van Hoof, and Welling 2019) we optimize our model by gradient descent with the well-known REINFORCE algorithm (Williams 1992). However, in contrast to such work, we introduce an additional entropy term:

$$\nabla_{\theta} \mathcal{L} = \mathbb{E}_{\pi_{\theta}(S|s)} [(f(S) - b(s)) \nabla_{\theta} \log \pi_{\theta}(S|s)] + \tau \nabla_{\theta} \mathcal{H}(\pi_{\theta}(s)), \quad (7)$$

where $b(s)$ is a baseline to reduce the variance of the gradient. Popular choices for the baseline are using an exponential moving average or training a critic to estimate value function given an state s . While the first one does not provide a baseline for a particular state s , the second one produces a complex training setup with two networks to optimize simultaneously. Therefore, we opted to compute the value with a rollout baseline, which estimates the value by performing rollout from a given state with the best policy obtained so far.

Algorithm 1: REINFORCE with Rollout Baseline

Input: number of epochs E , number of iterations per epoch I , batch size B , significance α

Output: trained model parameters θ

- 1: Init θ
 - 2: **for** $epoch = 1, \dots, E$ **do**
 - 3: **for** $iter = 1, \dots, I$ **do**
 - 4: $s_i \leftarrow \text{randomState}() \quad \forall i \in \{1, \dots, B\}$
 - 5: $S_i \leftarrow \text{rollout}(s_i, \theta) \quad \forall i \in \{1, \dots, B\}$
 - 6: $S_i^{BL} \leftarrow \text{rollout}(s_i, \theta_{BL}) \quad \forall i \in \{1, \dots, B\}$
 - 7: $\nabla \mathcal{L} \leftarrow \sum_{i=1}^B (f(S_i) - f(S_i^{BL})) \nabla_{\theta} \log \pi_{\theta}(S_i|s_i)$
 - 8: $\nabla \mathcal{L}_{\mathcal{H}} \leftarrow \tau \sum_{i=1}^B \nabla_{\theta} \mathcal{H}(\pi_{\theta}(s_i))$
 - 9: $\theta \leftarrow \text{Adam}(\theta, \nabla \mathcal{L} + \nabla \mathcal{L}_{\mathcal{H}})$
 - 10: **end for**
 - 11: **if** $\text{OneSidedPairedTTest}(\pi_{\theta}, \pi_{\theta_{BL}}) \leq \alpha$ **then**
 - 12: $\theta_{BL} \leftarrow \theta$
 - 13: **end if**
 - 14: **end for**
 - 15: **return** θ
-

After the loss is computed, the model parameters θ are updated using an Adam optimizer (Kingma and Ba 2014). At the end of each epoch, the model and the baseline are evaluated by performing a complete rollout over several examples. Then, the models are compared by means of a paired T-test. In the case that the model outperforms the baseline, the last one is updated with the model parameters. The full training procedure is detailed in Algorithm 1.

4 Experimental Evaluation

The main objective of our experimental evaluation is to assess the performance of our general collective formation approach in two structurally different real-world scenarios. On the one hand, we consider the ridesharing scenario discussed in (Bistaffa et al. 2021), where, as the authors show, an algorithm characterized by a strongly greedy component can produce solutions close to the optimal for hundreds of agents within one minute. On the other hand, we consider the team formation scenario discussed in (Andrejczuk et al. 2019), in

which greedy approaches cannot be used due to the presence of domain-specific constraints.

4.1 Application Domains

The ridesharing problem discussed in (Bistaffa et al. 2021) takes place in a map of zones $\mathcal{Z} = \{z_1, z_2, \dots, z_m\}$. An instance of this problem involves a pool of agents $A = \{a_1, a_2, \dots, a_n\}$, where each agent wants to travel from an origin to a destination, formally $a_i \in \mathcal{Z} \times \mathcal{Z}$. In the ridesharing domain we consider collectives with cardinality $1 \leq |S| \leq 5$ to reflect the usual capacity of cars. Each collective has an associated value assigned by a utility function $f(S)$ which represents the quality of service (e.g., the delay experienced by the users) and environmental benefits (e.g., the reduction of pollutant emissions or traffic) for the agents inside the collective.

The team formation problem discussed in (Andrejczuk et al. 2019) consists of a set of students $A = \{a_1, a_2, \dots, a_n\}$, which are assigned a task that has to be solved cooperatively in a team. In all our experiments, we consider the “English” task. Each student is represented by a tuple $(g, \mathbf{p}, \mathbf{l})$, where g is a binary value indicating the gender, \mathbf{p} is a vector with four personality traits, evaluated in the range $[-1, 1]$, and \mathbf{l} is a vector with seven competence levels in the range $[0, 1]$. Each task involves covering different competences that have to be covered by a student in the team.

Notice that, since the goal of team formation is to obtain a balanced set of teams so as to foster cooperation and inclusiveness, the authors of (Andrejczuk et al. 2019) originally defined the team formation problem as the maximization of a *Nash product*, which is then transformed into a linear optimization problem by considering the sum of the logarithms of the utility values of the teams. Here we adopt the same transformation, i.e., we consider the linearized formalization of team formation.

4.2 Baselines

To evaluate the performance of our approach in each of the above-mentioned domains, we employ the state-of-the-art approaches proposed in (Bistaffa et al. 2021) and (Andrejczuk et al. 2019), which we denote as PG² and SynTeam, respectively. For both approaches we use the parameters specified by the authors.

We remark that, as already mentioned in Section 2.1, these approaches already achieve close-to-optimal performance in their respective domains, hence our goal here is *not* to claim an improvement over these domain-specific solutions. We also remark that neither PG² nor SynTeam can be used outside of the domain in which they were originally designed. Thus, our goal is to show that our general approach can provide a performance comparable to these approaches without being restricted to any specific application domain.

Additionally, we compare our approach to the *Monte Carlo tree search* (MCTS) algorithm presented in (Wu and Ramchurn 2020), which, to the best of our knowledge, is the most recent general approach for the formation of collectives. Notice that such approach uses a greedy rollout policy based on selecting collectives with the best value increment

at each step. As already mentioned above, such a greedy policy can not be directly used in the team formation domain, which prevents the approach from (Wu and Ramchurn 2020) from finding any feasible solution in this case.

For this reason, we decided to consider a second version of MCTS that employs an heuristic which prevents choosing actions which might lead to an unfeasible collective during rollout. For the sake of completeness, we also consider a standard MCTS that employs a random rollout, i.e., that selects actions from a uniform distribution. These three MCTS approaches are referred to as G-MCTS (greedy), A-MCTS (i.e., adapted) and R-MCTS (i.e., random), respectively.

4.3 Methodology

We test the above-mentioned algorithms using real-world datasets obtained by the authors of the articles of the two considered case studies. Specifically, we consider 50 problem instances for ridesharing and 20 problem instances for team formation. For each instance, we run each algorithm using 50 different seeds (i.e., $0, \dots, 49$) and we compute the ratio between the average of the obtained solution values and the value of the optimal solution, i.e., the one obtained by solving (1) to optimality. In the experiments we scale to sizes for which we cannot compute the optimal, thus we used the state-of-the-art approach as a reference for these sizes. We then report the average over all instances of such optimality ratios. We do not report standard deviations since in all cases is < 0.02 .

Training & Evaluation The runtime and the hardware employed for training and evaluation are the following:

- Training times may vary depending on the size of the training instances, but in general it takes between 12 and 24 hours on standard hardware (NVIDIA RTX 2080 Ti GPU).
- For evaluation we consider a total time budget of 60 seconds. We employ IRACE (López-Ibáñez et al. 2016), a widely used software for tuning algorithmic parameters. Specifically we use it to determine the portion of the total time budget devoted to each part of our algorithm, as discussed in Section 3. The optimal portion of time budget devoted to the generation of promising candidates is 50 seconds. We observed that our model is capable of generating tens of thousands of collectives during this time budget on a Tesla Volta V100 PCIe GPU.

Our attention model is implemented in PyTorch. We employ CPLEX 20.1.0.0 as an ILP solver.

Hyperparameters We initialize the model parameters with $Uniform(-1/\sqrt{d}, 1/\sqrt{d})$, where d is the input size. For the attention mechanism we use 8 heads and $d = 256$, whereas for the feed-forward layers we use $d = 512$. The encoder is composed of 3 attention blocks. We train the models during 100 epochs consisting of 400 batches with 256 instances each. For evaluation we use 100 batches with the same number of instances. For optimization of the model parameters we use a learning rate of 10^{-4} and a significance of $\alpha = 0.05$ for the one-sided paired T-test. For the purposes of

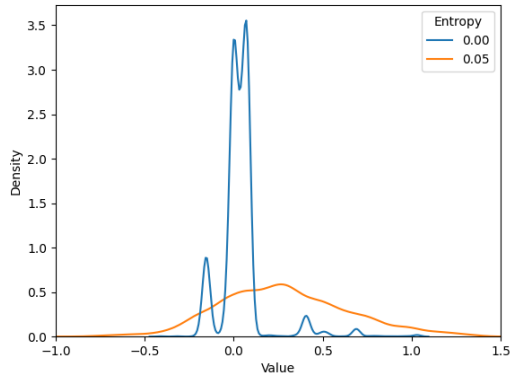


Figure 4: Probability density of the values of collectives generated by our attention model employing $\tau = 0.00$ and $\tau = 0.05$ for an example ridesharing problem instance.

this work, we did not perform an exhaustive parameter optimization. We recommend doing further parameter search in order to obtain more refined models.

4.4 Results

One of the most important findings we report from our experimental evaluation is that a higher entropy term produces more variety in the set of candidate collectives. Indeed, Figure 4 shows the distribution of values of the collectives generated by our model for one example ridesharing instance. It is clear that the model with the higher entropy term results in a higher diversity, while also producing collectives characterized by slightly higher utility values. During training, we also noticed that increasing $\tau > 0.05$ was effecting negatively to the convergence of the models. Therefore, we trained all models with $\tau = 0.05$, since it produced better results for our models. All our results we report refer to this value of τ . We now proceed to discuss the results of our comparison between our approach and the baselines discussed in Section 4.2 in the two considered collective formation domains.

Ridesharing Table 1 reports the results of our experiments on the ridesharing domain. Our results show that the optimality ratio obtained by our approach is comparable to the one obtained by the best performing MCTS approach (i.e., the greedy one), and clearly superior to the other MCTS approaches (including R-MCTS, the only true general approach in our comparison), who cannot compute a solution of acceptable quality. Our approach is comparable with the state-of-the-art PG^2 for $n = 50$ and 100 , but PG^2 still outperforms our model for $n = 200$. This result is not surprising, since PG^2 is the state-of-the-art specifically designed for this problem domain.

Team Formation Table 2 reports the results of our experiments on the team formation domain. The optimality ratio obtained in that case is significantly better than the one obtained by other MCTS approaches. Moreover, by compar-

n	AM (ours)	G-MCTS	A-MCTS	R-MCTS	PG^2
50	0.89	0.92	0.09	0.08	0.98
100	0.76	0.88	0.04	0.04	0.98
200*	0.65	< 0.01	0.01	0.02	1.00

Table 1: Optimality ratio of the attention model and baselines for a time budget of 60s for Ridesharing. *For $n = 200$ we report the ratio with respect to the solution computed by PG^2 since computing the optimal solution is not possible.

n	AM (ours)	G-MCTS	A-MCTS	R-MCTS	ST
50	0.97	< 0.01	0.84	< 0.01	0.99
60	0.95	< 0.01	0.78	< 0.01	0.99
100*	0.92	< 0.01	< 0.01	< 0.01	1.00

Table 2: Optimality ratio of the attention model and baselines for a time budget of 60s for Team Formation. *For $n = 100$ we report the ratio with respect to the solution computed by SynTeam, since computing the optimal is not possible.

ing our approach to SynTeam we can see that the gap between our approach and the domain-specific state-of-the-art approach for team formation is smaller than for ridesharing, specially for the smaller problem instances. Notice that our approach clearly outperforms all MCTS approaches on team formation, even the one we specifically adapted for this domain (i.e., A-MCTS).

5 Conclusions

In this work we proposed a general approach for the formation of collectives in real-world domains based on the novel combination of an attention model and an ILP. We show that our approach is superior to previous general approaches for the formation of collectives, despite domain-specific approaches still show superior performance in some settings.

Closing the gap with respect to domain-specific approaches is an important direction for future research. We believe that investigating alternatives to introduce diversity in the generation of collectives is a promising line of research, since we accredit entropy for a big part of the success of the attention model as an heuristic for the formation of collectives. Moreover, generalizing to larger problem instances is something our model does not excel at. Addressing these problems in future work is an important step to release the potential of our method, which is already competitive with respect to other general approaches for the generation of collectives.

Overall, we believe that this work is a first important step to foster the use of machine learning approaches for the formation of collectives. The good results achieved in the two structurally different domains that we used as benchmarks show that our attention model can indeed learn the inherent structure of the domain and exploit this to generate solutions of high quality.

References

- Alonso-Mora, J.; Samaranayake, S.; Wallar, A.; Frazzoli, E.; and Rus, D. 2017. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proceedings of the National Academy of Sciences*, 114(3): 462–467.
- Andrejczuk, E.; Bistaffa, F.; Blum, C.; Rodríguez-Aguilar, J. A.; and Sierra, C. 2019. Synergistic team composition: A computational approach to foster diversity in teams. *Knowledge-Based Systems*, 182: 104799.
- Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.
- Bengio, Y.; Lodi, A.; and Prouvost, A. 2020. Machine learning for combinatorial optimization: a methodological tour d’horizon. *European Journal of Operational Research*.
- Bistaffa, F.; Blum, C.; Cerquides, J.; Farinelli, A.; and Rodríguez-Aguilar, J. A. 2021. A computational approach to quantify the benefits of ridesharing for policy makers and travellers. *IEEE Transactions on Intelligent Transportation Systems*, 22(1): 119–130.
- Cerquides, J.; Farinelli, A.; Meseguer, P.; and Ramchurn, S. D. 2014. A Tutorial on Optimization for Multi-Agent Systems. *The Computer Journal*, 57(6): 799–824.
- Changder, N.; Aknine, S.; Ramchurn, S.; and Dutta, A. 2020. ODSS: Efficient hybridization for optimal coalition structure generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 7079–7086.
- European Commission. 2021. Collective Awareness Platforms for Sustainability and Social Innovation. <https://ec.europa.eu/digital-single-market/en/collective-awareness>.
- He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.
- Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Kool, W.; van Hoof, H.; and Welling, M. 2019. Attention, Learn to Solve Routing Problems! In *International Conference on Learning Representations*.
- Lin, C.-H. 1975. *Corporate tax structures and a special class of set partitioning problems*. Ph.D. thesis, Case Western Reserve University Cleveland, OH, USA.
- López-Ibáñez, M.; Dubois-Lacoste, J.; Cáceres, L. P.; Birattari, M.; and Stützle, T. 2016. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3: 43–58.
- Michalak, T.; Rahwan, T.; Elkind, E.; Wooldridge, M.; and Jennings, N. R. 2016. A hybrid exact algorithm for complete set partitioning. *Artificial Intelligence*, 230: 14–50.
- Nair, V.; Bartunov, S.; Gimeno, F.; von Glehn, I.; Lichocki, P.; Lobov, I.; O’Donoghue, B.; Sonnerat, N.; Tjandraatmadja, C.; Wang, P.; et al. 2021. Solving mixed integer programs using neural networks. *arXiv preprint arXiv:2012.13349*.
- Rahwan, T.; Michalak, T. P.; Wooldridge, M.; and Jennings, N. R. 2015. Coalition structure generation: A survey. *Artificial Intelligence*, 229: 139–174.
- Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.
- Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3): 229–256.
- Wu, F.; and Ramchurn, S. D. 2020. Monte-Carlo tree search for scalable coalition formation. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence and the 17th Pacific Rim International Conference on Artificial Intelligence. International Joint Conferences on Artificial Intelligence*.