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

Adrià Fenoy[b,a], Filippo Bistaffa[b], Alessandro Farinelli[a]

[a] *University of Verona*
[b] *IIIA-CSIC*

Abstract

We consider the problem of forming collectives of agents inherent in application domains aligned with *Sustainable Development Goals* 4 and 11 (i.e., team formation and ridesharing, respectively). We propose a general solution approach 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 collective formation problems inherent in the ridesharing and team formation domains show that our approach provides comparable solutions (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*.

Keywords: Attention Models, Reinforcement Learning, Collective Formation, Optimization

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 [1]. Such new forms of collaborative consump-

[*] Corresponding author
Email address: `filippo.bistaffa@iiia.csic.es` (Filippo Bistaffa)

tion and production pose complex, multi-faceted challenges, but they ultimately all rely on a common and fundamental task, i.e., *the formation of collectives.*

In this work, we consider two real-world application domains, i.e., ridesharing and team formation (respectively aligned with *UN Sustainable Development Goals* 11 and 4 [2]), where agents complete tasks and achieve benefits through the formation of collectives to achieve cooperation.

On the one hand, in *ridesharing* commuters can form groups and travel together with the objective of reducing transportation costs, mitigating pollutant emissions, and alleviating traffic congestion in urban environments [3, 4]. 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. As recently shown by Andrejczuk et al. [5], 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. However, 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* based on 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 manually specifying 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 collectives[1], we reduce the complexity of the original problem by several orders of magnitude while producing a high-quality solution.

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 formulation.

- 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 [6], our solution achieves a wide variety of promising candidates. Such variety is a key feature that allows the ILP solver to compute a high-quality solution to the collective formation problem.

- We evaluate our approach on collective formation problems inherent in two real-world domains (i.e., ridesharing and team formation) by comparing it with state-of-the-art approaches specific to each domain. Our results show that our approach can produce 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* (MCTS) [7],[2] showing that our solutions outperform (in terms of quality) the ones computed by the counterpart.

2. Background & Related Work

In this section, we discuss the necessary background and the relevant literature on the formation of collectives. We then elaborate on previous attempts at using machine learning techniques to solve combinatorial optimization problems.

[1] The 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.

[2] By “general approach” here we mean an approach that can be applied across different domains without significant changes, such as MCTS in this case.

2.1. Formation of Collectives of Agents

The problem of forming collectives of agents has been deeply studied from many different perspectives in the scientific literature. Depending on the context and the application domain, collectives of agents are also referred to as *coalitions* [8] or *teams* [5, 9] of agents. Here we adopt the term "collective" to refer to the general concept of a "group" of agents that cooperate to complete tasks or obtain benefits, as we deem it more general and intuitive.

More in particular, in this paper we focus on the optimization problem [10] of computing the best set of non-overlapping collectives (i.e., subsets) of agents belonging to a universal set A, to maximize the total value provided by a domain-specific utility function, e.g., the reduction in terms of cost or CO_2 emissions associated to the arrangement of a shared trip [3] or the improvement thanks to cooperation within a team of students [5].

Formally, we consider a set of n agents $A = \{a_1, a_2, \ldots, a_n\}$ and a utility function $f : \mathcal{F}(A) \to \mathbb{R}$ (also referred to as *characteristic function*) that maps every collective in the feasible set[3] of collectives $\mathcal{F}(A)$ to a real number. We formulate the formation of collectives as the problem of computing the best set $\mathcal{S}^*$ of non-overlapping subsets of A (also referred to as a *coalition structure* [8]) that maximizes the sum of the values associated to each collective $C \in \mathcal{S}^*$, i.e.,

$$\mathcal{S}^* = \underset{\mathcal{S} \in \prod(A)}{\arg\max} \sum_{C \in \mathcal{S}} f(C), \tag{1}$$

where $\prod(A)$ is the set of all partitions of A into non-overlapping feasible subsets.

2.1.1. Complete Approaches

By and large, the formation of collectives requires to solve a *coalition structure generation* (CSG) problem [13, 8] or, equivalently, a *set partitioning* problem [14]. A wealth of complete approaches have been proposed to solve Equation 1 to optimality [13], depending on the properties of the utility function f.

[3] Depending 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., cardinality constraints [11] or graph-based constraints [12]), as explained in Section 2.1.2

Complete CSG algorithms [15, 16] usually make no assumptions on 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 CSG algorithm itself) requires enumerating a number of values that grow exponentially with the number of agents.

For this reason, complete unconstrained CSG algorithms are limited to only 25–30 agents, i.e., a scale that is not sufficient for realistic applications involving hundreds of agents (such as the ones we consider in this paper), hence we will not consider these approaches as benchmarks in our experimental evaluation.

2.1.2. Approaches for Constrained Scenarios

In some application domains, it is possible to exploit specific properties to improve the runtime of the solution algorithm by considering constraints that reduce the number of feasible collectives. Cardinality constraints that limit the maximum size of the collectives to k naturally arise in many realistic scenarios [3, 5, 11]. In addition, a strand of literature pioneered by Myerson [12] has investigated *graph-restricted* scenarios [17, 18, 19, 20] where collectives can be formed only if they induce a connected subgraph of an initial graph defined over the set of agents (e.g., a social network).

Despite considering these constraints can significantly reduce the number of total collectives (up to a polynomial number $\binom{|A|}{k} = O(|A|^k)$ if collectives are restricted to a maximum cardinality of k agents [11]), such a number remains prohibitively large for realistic applications involving hundreds of agents. Indeed, as observed by Bistaffa et al. [3] and Andrejczuk et al. [5], enumerating and computing the utility value for all the collectives of size up to 5 can require *hours*, especially when the computation of the utility function is particularly demanding (e.g., in team formation it requires to solve a small task assignment problem [5]). Along these lines, the authors of [3, 5] concluded that complete algorithms were not a viable solution for real-world scenarios—even constrained ones—, resorting to domain-specific approaches that can compute sub-optimal solutions of good quality in a manageable amount of time (see Section 2.1.5).

2.1.3. Approaches Focusing on Specific Function Representations

Another strand of literature [21, 22, 23] has focused on alternative utility function representations, which allow one to reduce the computational complexity of the CSG problem by exploiting specific properties of the adopted representation. For example, Ieong and Shoham [21] proposed a concise representation called *marginal contribution nets*, or MC-nets, where the calculation of the utility is based on a collection of *rules.* Tran-Thanh et al. [22] proposed a representation called *coalitional skill vector model*, where there is a set of skills in the system, and each agent has a skill vector (a vector consisting of values that reflect the agents' level in different skills). More recently, Bistaffa et al. [23] focused on the well-known *induced subgraph game* (ISG) representation originally introduced by Deng and Papadimitriou [24] and proposed a CSG algorithm based on graph-clustering that exploits the succinctness of the representation.

By definition, these approaches can only be applied if the utility function f of the collective formation domain satisfies some specific properties (e.g., it can be represented as a combination of *rules* in the case of MC-nets, or it can be represented by a sum of the weights of the graph in case of ISGs). Unfortunately, these specific properties rarely hold in real-world application domains. Indeed, neither of the two considered real-world collective formation domains (i.e., ridesharing and team formation) can be modeled as one of the above-mentioned function representations. In contrast, our work goes into the opposite research direction, i.e., obtaining a general approach for collective formation that does not rely on any specific property. For this reason, we will not compare against these approaches in our experimental evaluation.

2.1.4. Team Formation Approaches

Collective formation has also been widely studied in the context of *team formation*, in which such a problem has been studied from different perspectives. For instance, Gaston and desJardins [25] focused only on local optimization without considering any concept of global optimal solution, proposing a heuristic to modify the graph connecting the agents based on local autonomous reasoning.

Lappas et al. [26] studied the complexity of finding a single group of agents who possess a given set of skills to minimize the communication cost within such a group, and proposed a heuristic algorithm to solve such a problem. Marcolino et al. [27] focused on forming a single group of agents that has the maximum strength in the set of world states, showing that a diverse team can outperform a team formed by uniform members, and proposing optimal voting rules for such a diverse team. Finally, Liemhetcharat and Veloso [28] tackled the task of modeling the values of the utility function based on observations, without considering any partitioning problem on top of it.

Here we focus on the optimization problem of forming disjoint teams with the objective of maximizing the sum of the corresponding utility values. In this context, Andrejczuk et al. [5] proposed a local-search algorithm named SynTeam that heavily relies on the structure of the problem and the considered dataset to form proficient teams that are all assigned the same task (e.g., an English proficiency task, an arts and design task, etc.). This local-search approach was later extended by Georgara et al. [29, 30] to account for multiple tasks. Präntare and Heintz [9], on the other hand, proposed an optimal solution algorithm for the same optimization problem, which involves solving a CSG and a task assignment problem at the same time.

Since we consider the team formation scenario involving one single task, we only consider SynTeam [5] as a competitor among the above-discussed team formation approaches.

2.1.5. Heuristic Approaches

To overcome the scalability limitations discussed in previous sections, 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. Note that, in this case, the domain knowledge that the approaches exploit is not necessarily related to the characteristic function representation (in contrast with the approaches discussed in Section 2.1.3), rather it is related to specific properties of the application domain.

For instance, Bistaffa et al. [3] 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. Previously, Farinelli et al. [31] proposed an approach based on hierarchical clustering that also relies on a greedy heuristic to identify the most promising couple of coalitions that can be merged, until no beneficial merge can be executed. Unfortunately, these approaches cannot be applied in collective formation domains that are not characterized by such a greedy nature, e.g., the team formation domain discussed in [5].

More recently, Wu and Ramchurn [7] proposed a CSG solution algorithm based on MCTS that can be used in any collective formation domain, including ridesharing and team formation. Nonetheless, such a MCTS approach employs a simulation policy based on a greedy heuristic similar to the one proposed by Farinelli et al. [31]. Indeed, the authors report good performance on synthetic datasets that are characterized by a greedy nature.[4]

Along these lines, in our experimental evaluation, we compare against the approaches by Wu and Ramchurn [7] and by Bistaffa et al. [3].

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 few years. According to Bengio et al. [32], machine learning can contribute to the optimization field in twofold ways: i) replace some heavy computations by building fast approximations, and ii) improve the optimization approach by learning domain-specific structure.

Due to the wide diversity among ways of combining machine learning and combinatorial optimization, Bengio et al. [32] classify the different approaches

[4] According to the methodology reported in [7], the value of a coalition C is correlated with its cardinality $|C|$, hence forming bigger coalitions is, on average, more beneficial.

along two axes. Along the first axis, depending on the structure of the overall approach, Bengio et al. [32] identifies two alternatives: i) end-to-end machine learning approaches that are able to directly construct a solution for optimization problems, and ii) mixed approaches that use machine learning as a subroutine of a classical optimization approach, either as a preprocessing step or used alongside the classical approach in an online scheme. On the other hand, the second axis concerns the adopted learning methodology, i.e., supervised, unsupervised, or reinforcement learning.

With respect to this classification, in this paper we propose a mixed approach, where an attention model generates high-valued candidate collectives, which are then encoded as variables in an ILP. Moreover, the attention model is trained by means of reinforcement learning to learn the domain-specific structure of the application domain.

Along these lines, in what follows we first provide an overview of the relevant literature on end-to-end approaches, and then we focus on mixed approaches. Afterward, we discuss approaches trained with different learning methods.

2.2.1. End-to-End Approaches for Optimization

There is a wide variety of end-to-end approaches to optimization, such as Pointer Networks by Vinyals et al. [33], Graph Convolutional Networks by Joshi et al. [34], or Attention mechanisms by Kool et al. [6]. These methods have been applied to various classic problems in Operations Research and Optimization, such as the well-known *traveling salesperson problem* (TSP). Moreover, other end-to-end approaches have been devised for variants of the *vehicle routing problem* (VRP), such as the capacitated VRP [35] or the online capacitated VRP [36]. It is important to note that, while our attention mechanism is inspired[5] by the one by Kool et al. [6] —an end-to-end approach—, our overall approach is

[5] We focus on attention models because recent work [6] shows that they can significantly outperform Pointer Networks on routing problems. Moreover, since inputs to our model are sets of agents (which are permutation invariant), we prefer the use of an attention mechanism, whose output is guaranteed to be invariant to permutations of the input set.

instead a *mixed* one, as mentioned in Section 2.2. More specifically, our attention model does not compute the final solution to the considered problem (i.e., the formation of collectives), but rather a set of candidate collectives that constitute input of the ILP, which then computes the final solution. This combination of Machine Learning and Optimization makes our approach "mixed".

In what follows, we discuss other relevant works in the category of mixed approaches.

2.2.2. Mixed approaches for Optimization

Mixed approaches aim at combining machine learning with classic optimization procedures. In this line of research, a strand of literature focuses on using machine learning as a preprocessing step before a classical optimization approach. In particular, the approach proposed by Ding et al. [37] solves 8 different classical problems, including TSP and VRP, by means of Graph Convolutional Neural Networks which predict the value of binary variables in a *mixed-integer linear program* (MILP) formulation of these problems. The solution is then found by employing a branch and bound approach which uses the values to guide the search. In the same way, Li et al. [38] tackle benchmark satisfiability and problems related to social networks by means of Graph Convolutional Neural Networks to select the nodes in a graph that are likely to appear in an optimal solution and then solve the problem for the reduced graph. Similar to neural network-based approaches, *support vector machine*s (SVMs) also can play the role of a heuristic approach as a preprocessing step for optimization problems. Examples of this are the work of Xavier et al. [39], which uses k-Nearest Neighbors in addition to SVMs to significantly reduce the problem size for Security-Constrained Unit Commitment problem in power systems and electricity markets. Another example is the work of Sun et al. [40], which uses SVMs to find a reduction of the graph for the maximum weight clique problem. Many of the approaches that use machine learning as a preprocessing step focus on reducing the optimization problem (i.e., making the problem smaller and computationally tractable). Despite problem reduction with machine learning does not ensure any optimality guarantee, these approaches can usually provide high-quality solutions thanks to the "backbone" structure of the optimization problems they tackle (according to the terminology adopted by [41]), i.e., optimal solution and high-quality solutions are likely to share a specific structure. Our work aims at following a similar problem reduction strategy by lowering the number of variables in an ILP formulation for the collective formation problem, but in contrast with the above-mentioned works (which build the *entire* problem instance and then reduces it), we directly generate the reduced ILP by generating collectives which are likely to be in the optimal solution.

Another family of mixed approaches is the one that aims at integrating machine learning within a classical approach for optimization. For example, Hottung and Tierney [42] use an attention model to reconstruct solutions inside a Large Neighborhood Search setting for the Capacitated VRP. Another example is the work of Hottung and Tierney [42], in which neural networks are used as a heuristic to guide search inside a tree search approach for the *container pre-marshaling problem.* In this respect, our approach is similar to the one proposed by Bistaffa et al. [3], which uses a domain-specific heuristic as a preprocessing step for the formation of candidate collectives and then computes the final solution employing an ILP. However, instead of using an ad-hoc heuristic designed for a specific domain, we introduce an attention model that learns the domain-specific structure of a problem. Therefore, by not relying on any domain-specific component, our approach can be applied to structurally different collective formation problems.

2.2.3. Learning Methods

Depending on the adopted learning methodology, an approach can be classified as supervised, unsupervised, or based on reinforcement learning. Several approaches adopt supervised learning strategies to imitate the outputs of an already existing solution algorithm, hence, by following Bengio et al. [32]'s terminology, "replacing it by its ML approximation". For example, the above-discussed work by Li et al. [38] makes use of supervised learning to train Graph

Convolutional Neural Networks for Maximal Independent Set, Minimum Vertex Cover, Maximal Clique, and SAT. Khalil et al. [43] propose an approach to learn from a precomputed dataset a ranking function which is then used as a branching heuristic for MILP. Gasse et al. [44] and Nair et al. [45] also aim at constructing a branching heuristic for solving MILP, but in their case, they directly learn to imitate the decisions made by an expert employing a cross-entropy loss. The majority of these approaches aim at selecting an option among several alternatives (e.g., variable selection in branch-and-bound approaches), some approaches use machine learning to decide if an action (which is usually very demanding from a computational point of view) needs to be performed or not. This is the case of the work by Kruber et al. [46] where they use a supervised learning approach to decide whether a Dantzig-Wolfe decomposition should be applied to a MIP instance to solve it faster.

Unsupervised learning tries to capture patterns appearing in unlabeled data. Since its purpose does not pursue finding fast approximations and it is not obvious in which way it could improve the solution quality of current approaches, its use when combining machine learning with optimization is not very common. To the best of our knowledge, the only approach that applies an unsupervised machine learning technique to optimization is the work by Karalias and Loukas [47], where a probabilistic loss is proposed to train a Graph Neural Network for the Graph partitioning and maximum clique problems.

Finally, *reinforcement learning* (RL) aims at learning a policy (i.e., a system that determines a course of action) that optimizes the sum of future expected rewards observed from the actions performed by the policy. The policy is usually modeled inside a Markov Decision Process and interacts with the environment by executing an action and receiving an observation and a reward signal. The use of RL is appealing in the context of mixing machine learning with optimization because classical algorithms used for this purpose usually involve sequential decisions, which can be modeled as a Markov Decision Process. Moreover, in contrast with supervised learning, RL does not require other approaches to learn from, since it learns purely from experience. Several approaches aim at replacing supervised learning with RL in the context of combinatorial optimization. For example, the work by Bello et al. [48] and the aforementioned work by Nazari et al. [35] which proposes RL as an alternative way to train Pointer Networks, previously trained with supervised learning [33].

Related to our work, the approach by Kool et al. [6] employs the REINFORCE algorithm, a RL approach introduced by Williams [49], to train an attention-based model for the Traveling Salesperson Problem and the VRP. Along these lines, we adopt RL instead of other learning approaches, since we aim at achieving a general approach that is not limited by the quality of the examples used in a supervised environment. Because of the importance of the REINFORCE algorithm, which constitutes the training framework of our attention-based model, we devote the following section to discussing this algorithm in more detail.

2.3. The REINFORCE Algorithm

The REINFORCE algorithm proposed by Williams [49] constitutes one of the pillars of RL and has inspired many modern RL approaches. The general idea of the algorithm is to update the parameters of a model utilizing gradient methods in the direction that reinforces actions with higher rewards.

At a general level, a RL setup is characterized by a sequence of states, actions, and rewards, i.e., $(s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_H, a_H, r_H)$. An agent in this setup is modeled by a parameterized policy $\pi_{\boldsymbol{\theta}}$ and it decides on the actions that are performed according to the observed states. Moreover, in the RL setup, the agent receives a reward as feedback from its actions, which reinforces actions leading to a desired behavior. The reward is defined according to the optimization goal. In practice, when considering RL for combinatorial optimization, the reward is assigned to be the utility function of the optimization domain. However, this reward is usually not defined for intermediate states. In such situations, an expected return G_t is used instead, which can be computed from the probabilities determined by the policy at each state or by estimating it through a simulated trajectory referred to as a rollout. In this work, we employ the rollout technique to estimate the expected return of intermediate states corresponding to incomplete coalitions. To update the policy parameters $\boldsymbol{\theta}$, Williams [49] proposes to use gradient methods, which update the parameters in the direction that maximizes the expected return with

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha G_t \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}, \tag{2}$$

where α is the size of the learning step. In this work, we adopt a technique from actor-critic approaches [50], which introduces a baseline to reduce the variance in the expected return. This technique consists of employing the increment of the expected return with respect to a baseline b_t, which will be defined in the following sections. Then, the update rule becomes

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha (G_t - b_t) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}. \tag{3}$$

Previous work employs REINFORCE to train a policy to produce solutions to combinatorial optimization problems [48, 6], such as the TSP and the VRP. For such applications, the characteristic function of the problem is used as a reward to guide the models towards generating solutions of continuously improving quality until they converge to close-to-optimal solutions. Here we employ the REINFORCE algorithm to train a policy to generate collectives of high quality. Thus, the quality of a collective, determined by the utility function of a specific collective formation domain, is used as the reward.

Very recently, the use of entropy within REINFORCE has been independently proposed by the researchers of the team *Ratel* during the *AI4TSP* competition [51]. More specifically, *Ratel*'s approach employs entropy regularization "to enhance exploration" [51, Section 4.2.2] within RL for the solution of TSP. Similarly, in our collective formation approach, entropy is used to increase the *diversity* of the generated candidates (via enhanced exploration), as we discuss in Section 3.1.4. Notice that, in contrast to *Ratel*'s approach, we do not use machine learning to directly provide a solution to the considered optimization problem, but to provide a set of diverse candidate solutions that are then processed by a classical optimization approach (i.e., an ILP).

After having discussed the RL setup, which will be part of our approach, we now proceed to present our solution approach for optimization problems involving the formation of collectives.

3. Our Solution Approach

As seen in previous sections, collective formation approaches that aim at considering the entire set of possible collectives cannot be applied to realistic application domains since such a set is prohibitively large to enumerate. In this respect, it is crucial to avoid the generation of such an impractically large problem instance in the first place since it would be impossible to handle for any solution algorithm. Indeed, our approach follows this rationale and works by breaking the problem into two parts. First, by means of an attention model, we generate candidate collectives, one at a time, to construct a set of candidate collectives. Second, we encode these collectives into an ILP, which computes the final solution.

To accommodate the discussion of our approach, we first reformulate the optimization problem in Equation 1 as an ILP:

$$\begin{aligned} \text{maximize} \quad & \sum_{C \in \mathcal{F}(A)} f(C) \cdot x_C, \\ \text{subject to} \quad & \sum_{C \in \mathcal{F}(A)} b_{i,C} \cdot x_C \leq 1, \quad \forall a_i \in A, \end{aligned} \tag{4}$$

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

Notice that the ILP formulation in Equation 4 only considers the constraint that collectives must be non-overlapping. Depending on the considered application domain, it is also possible to impose that the formed collectives cover the entire set A by enforcing

$$\sum_{C \in \mathcal{F}(A)} b_{i,C} \cdot x_C = 1, \quad \forall a_i \in A. \tag{5}$$

The problem in Equation 4 can be easily recognized as a WSP, one of the original 21 Karp's NP-complete problems [52]. Because of the computational complexity of such problems, the ILP can only be solved for small instances (i.e., A with less than a couple of tens of agents) by employing 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 $f(C)$ requires a significant computational effort, such as the synergistic value proposed by Andrejczuk et al. [5].

On the other hand, 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.,

$$\begin{aligned} \text{maximize} \quad & \sum_{C \in \mathcal{R}(A)} f(C) \cdot x_C, \\ \text{subject to} \quad & \sum_{C \in \mathcal{R}(A)} b_{i,C} \cdot x_C \leq 1, \quad \forall a_i \in A. \end{aligned} \tag{6}$$

Figure 1 illustrates the two-step approach we propose for the formation of collectives. In the first step, the attention model produces collectives, one by one, to build a set $\mathcal{R}(A)$ of candidate collectives. In the second step, Equation 6 is solved with the set $\mathcal{R}(A)$ as input to solve the collective formation problem. The attention model will be discussed in the following sections.

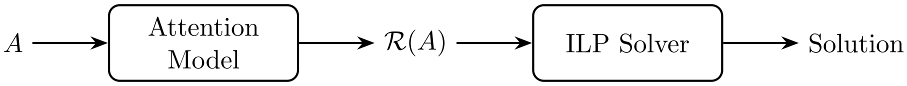


Figure 1: Proposed approach for the formation of collectives. The attention model generates a reduced set of collectives from which an ILP solver computes the solution.

Algorithm 1 provides the pseudocode of our general approach for the formation of collectives. Following a standard practice [3], we assume that our entire approach is provided with a time budget t and we distribute such a time budget

Algorithm 1 Pseudocode of our approach for the formation of collectives

```
Input: set of agents $A$, overall time budget $t \in \mathbb{R}^+$, generation portion $k \in [0,1]$
Output: the computed set of collectives
1: $\mathcal{R}(A) \leftarrow \emptyset$ {Initialize empty set of candidates}
2: repeat
3:     $A' \leftarrow A$ {Local copy of input set of agents}
4:     while $|A'| > 0$ do
5:         $S \leftarrow$ subset of $A'$ generated by means of our attention model
6:         $\mathcal{R}(A) \leftarrow \mathcal{R}(A) \cup S$ {Add $S$ to the set of candidates}
7:         $A' \leftarrow A' \setminus S$ {Remove $S$ from the local set of agents}
8:     end while
9: until the time budget $t \cdot k$ expires
10: Formulate the model in Equation 6 given $\mathcal{R}(A)$ computed in Lines 2–9
11: Solve such a model with an ILP solver given a time budget of $(1-k) \cdot t$
```

between the two phases illustrated in Figure 1. More specifically, we devote a time budget of $k \cdot t$ (with $k \in [0,1]$) to the first phase, in which we generate the reduced set $\mathcal{R}(A)$ by repeatedly generating a collective S with our attention model and removing it from the initial set A until such a set has been entirely consumed. If the generation time budget has not been exhausted, we repeat the same procedure with another copy of the initial set. The remaining part $(1-k) \cdot t$ is devoted to the solution of the just-computed reduced ILP model in Equation 6, by providing such a time budget to the off-the-shelf ILP solver. In our experiments in Section 4 we compute the value of the parameter k via IRACE [53], a widely used software for tuning algorithmic parameters.

3.1. Attention Model

Our attention model implements a decision-making process where collectives are built incrementally by selecting elements from the set of agents A, one at every decision step, and adding them to the collective C. During such a decision-making process, these two fundamental pieces of information (i.e., A and C) are internally maintained as the *state* of the model.

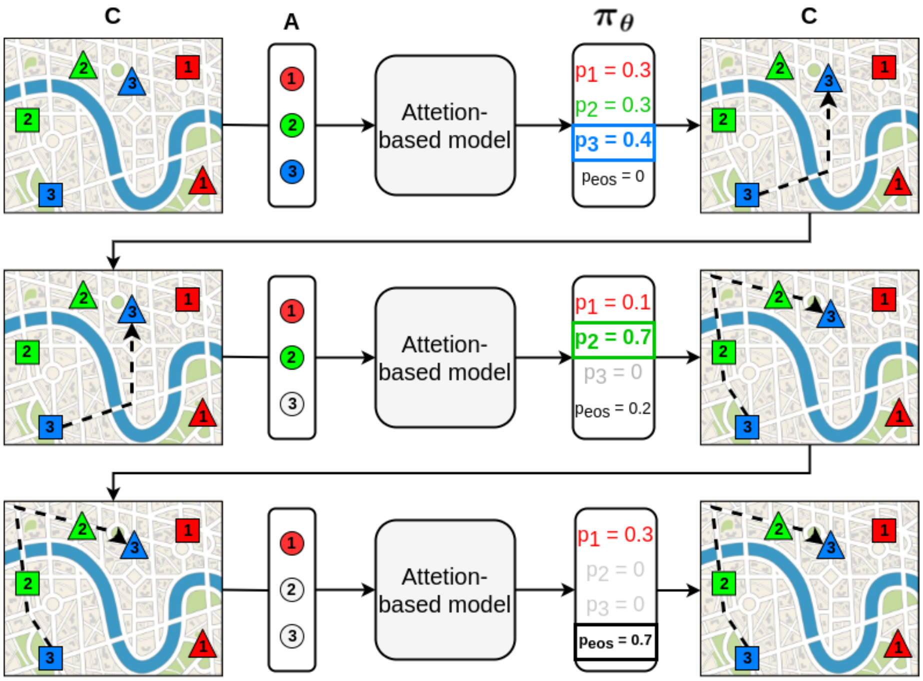


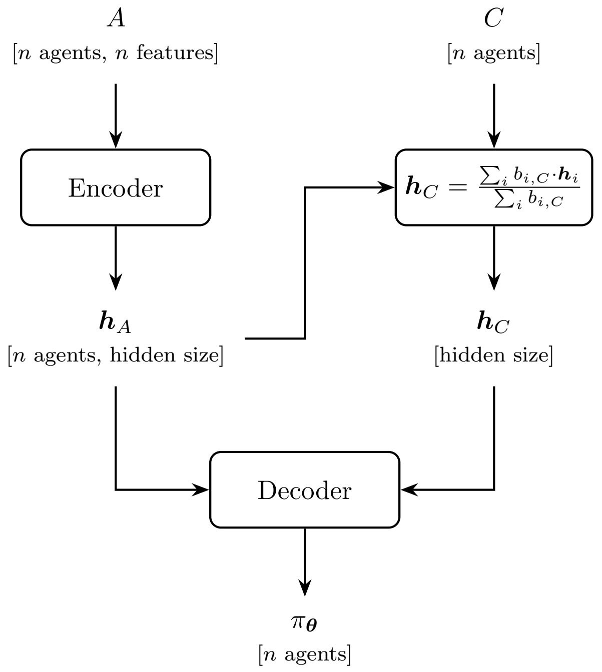
Figure 2: An illustrative example of the process carried out by the attention model to form a collective in a ridesharing environment. In this example concerning the ridesharing domain, three agents (1: red, 2: green, and 3: blue) are the input of the model, together with the current collective, which is initialized empty. The collective is represented on a map to show the spatial relations between the origin (squares) and destination (triangles) locations. The model outputs a vector of probabilities, one for each agent plus one for the "end-of-sequence" (*eos*) token, which stops the formation of the collective. During the selection process, agents that have already been selected are masked out (indicated in gray in this example). Best viewed in colors.

Figure 2 illustrates an example of the process of forming a collective by means of the attention-based model in a ridesharing application domain. Initially, an empty collective and the set A of agents are provided as input to the model, which outputs a vector of probabilities, one for each agent plus one for the "end-of-sequence" (*eos*) token (indicating that no agent is selected and terminating the process of building the current collective). The probabilities, indicating the best agent to be selected, are used to sample one agent and add it to the collective. The collective is then updated, and a new iteration starts. A mask that forces the probability of selected agents to zero is used to avoid one agent being selected twice. The process finishes when the *eos* token is sampled.

More formally, the internal state s of the model is represented by the tuple $s = (A, C)$, where A is the set of agents and C is the collective currently being built. With a small abuse of notation,[6] our model receives the set of agents A as a list of d_a dimensional feature vectors, where d_a is the number of features. Following standard practice [54], our model assumes that an element (in our case an agent) can be represented as a vector of features (e.g., origin and destination locations in the ridesharing scenario, or students' personality traits and competence levels in the team formation one). The model also receives a binary encoding of a collective $C = \{b_{1,C}, b_{2,C}, \dots, b_{n,C}\}$, where $b_{i,C}$ are binary values determining whether the respective agents a_i are in the collective or not.

Given a state s, we design an attention-based encoder-decoder model based on the one proposed by [6], which defines a stochastic policy $\pi_{\boldsymbol{\theta}}(s)$ parameterized by a set of learnable parameters $\boldsymbol{\theta}$ representing the weights and biases of a neural network) that determine the probability for each element in the pool of agents A to be included in the collective C. Our encoder produces an embedding, i.e., a continuous representation of the input, for each element in the pool. Then, as illustrated in Figure 3, the decoder receives the embedding and the collective to compute the probabilities.

[6] Thus far, we considered A and C to be sets of agents. Now we redefine them respectively as a list of vectors and a list of binary variables.



A
[n agents, n features]
C
[n agents]
Encoder
$\boldsymbol{h}_C = \frac{\sum_i b_{i,C} \cdot \boldsymbol{h}_i}{\sum_i b_{i,C}}$
$\boldsymbol{h}_A$
[n agents, hidden size]
$\boldsymbol{h}_C$
[hidden size]
Decoder
$\pi_{\boldsymbol{\theta}}$
[n agents]

Figure 3: General scheme of the encoder-decoder approach that computes the probability $\pi_{\boldsymbol{\theta}}$ for each agent in A to be added to the collective C. The sizes for input, output, and intermediate hidden states are specified for each element. The specific details of the probability computation are provided in Section 3.1.1.

3.1.1. Multi-head Attention

The main block of our encoder-decoder model is based on the attention mechanism by Vaswani et al. [55]. Within our approach, attention works as a mapping where the inputs are, following the standard nomenclature [55, 6], the queries Q and the keys K represented by a set of d_q and d_k dimensional vectors respectively. The outputs are the attention weights a_{ij}, which reflect the normalized compatibility of the query $\boldsymbol{q}_i$ with the key $\boldsymbol{k}_i$. The first step to obtain the attention weights is to compute the compatibilities

$$u_{ij} = \frac{(W^q \boldsymbol{q}_i)^T (W^k \boldsymbol{k}_j)}{\sqrt{d_q}}, \tag{7}$$

where W^q and W^k are two learnable linear transformations, and the output is scaled with a factor of $\frac{1}{\sqrt{d_q}}$. The attention weights are then obtained by normalizing the compatibilities with a softmax:

$$a_{ij} = \frac{e^{u_{ij}}}{\sum_{j'} e^{u_{ij'}}}. \tag{8}$$

These attention weights already capture the desired information (compatibility between queries and keys) and can already be used for a given purpose, as we do in the last step of the decoder, where we interpret these attention weights as probabilities for elements in A to be included in C; that is, the higher the compatibilities, the higher the probabilities will be. Nevertheless, the main use of the attention weights is to update a set of values V represented by a set of d_v dimensional vectors by computing a linear combination of the values weighted by the attention weights

$$\boldsymbol{v}'_i = \sum_j a_{ij}(W^v \boldsymbol{v}_j), \tag{9}$$

where W^v is again a learnable linear transformation. In practice, according to the multi-head attention approach, it is beneficial to compute attention in parallel M times with different parameters W^q, W^k, and W^v, and combine the outputs at the end, i.e.,

$$\boldsymbol{v}'_i = \sum_{m=1}^{M} W^o_m \boldsymbol{v}'_{im}, \tag{10}$$

where W^o is again a learnable linear transformation.

In our approach, we use the attention mechanism in the encoder to incorporate information from the set of agents A into the representation of an agent a_i by using agents in A as queries, keys, and values. Thus, the attention mechanism behaves as a message-passing approach between consecutive neural layers in the attention-based model. This mechanism is known as self-attention since the elements in A are updated according to elements in the same set [55]. Further, in the decoder, we update the representation of a collective C with information about the agents A, i.e., we employ C as query and value, while A behaves as keys, following the standard nomenclature adopted for attention models [55, 6].

3.1.2. Encoder

Our encoder is inspired by the one proposed by Vaswani et al. [55], but 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. Instead, we use an input feed-forward layer to encode elements in the pool of agents A from its d_a dimensional feature representation to a d_h dimensional embedding before the main attention blocks.

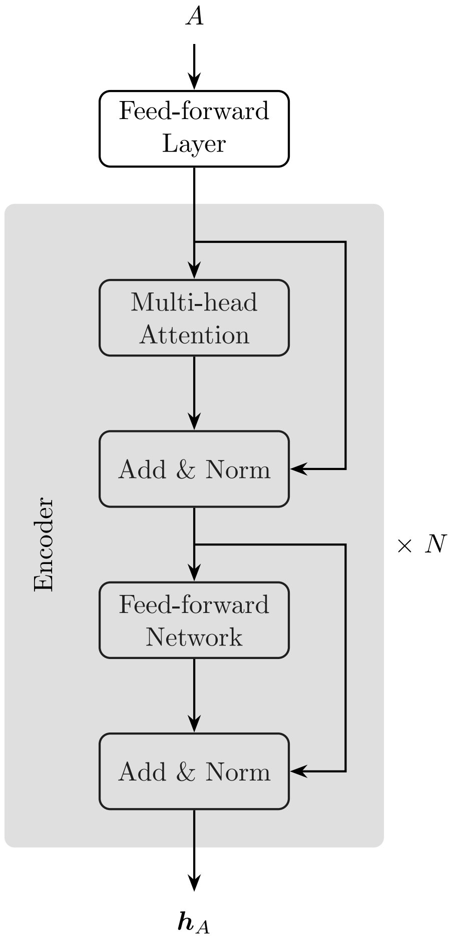


Figure 4: General scheme of our encoder architecture. Similar to the one proposed by Vaswani et al. [55], it combines several steps of multi-head attention, feedforward layers, layer normalization, and residual connections. The encoding step is repeated N times.

To get the encoded representation of the pool of agents $\boldsymbol{h}_A$, the input embeddings are updated using N attention blocks depicted in Figure 4, each one consisting of two sub-layers: a multi-head self-attention and a feed-forward layer. Each sub-layer adds a residual connection [56] and performs layer normalization [57] 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.

3.1.3. Decoder

In order to compute the probabilities $\pi_{\boldsymbol{\theta}}(s)$, the decoder performs attention between the encoded pool $\boldsymbol{h}_A = \{\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots, \boldsymbol{h}_n\}$ and an encoding of the collective $\boldsymbol{h}_C$. This encoding is needed since we aim to obtain a single scalar value for each element in the set of agents, which will represent the probability. This can be better understood by looking at Equation 7, where the dot product outputs a scalar value, which is then used as input in Equation 12 to compute the probabilities. To obtain this encoding, the following reduction is applied to the encoded pool:

$$\boldsymbol{h}_C = \frac{\sum_i b_{i,C} \cdot \boldsymbol{h}_i}{\sum_i b_{i,C}}. \tag{11}$$

At the initial state, the collective is empty, which means that $\sum_i b_{i,C} = 0$. In that case, we use a d_h dimensional zero vector as a placeholder, $\boldsymbol{h}_C = \boldsymbol{0}$.

Finally, to obtain the probabilities $\pi_{\boldsymbol{\theta}}(s)$, the decoder performs two last attention steps, which update $\boldsymbol{h}_C$ employing $\boldsymbol{h}_A$ and $\boldsymbol{h}_C$ as keys and queries. After the second attention step, the compatibilities u_i[7] are normalized by applying a softmax to obtain the probability of each agent being added to a collective C, i.e.,

$$\pi_{\boldsymbol{\theta}}(i \mid s) = \frac{e^{\gamma \tanh u_i}}{\sum_j e^{\gamma \tanh u_j}}, \tag{12}$$

where the function $\gamma \tanh u_i$ is applied to the compatibilities as in [48] to control the exploration of the model by shrinking the compatibilities to the range $[-\gamma, \gamma]$.

[7] Notice how the compatibilities here do not have a j component as in Equation 7, because the encoder of the collective $\boldsymbol{h}_C$, consisting of a single vector, is employed as a single query.

To ensure that an agent appears only once in a collective, a mask is applied to the agents that have been already selected before computing the softmax, i.e., $u_i = -\infty$ for such agents. To stop the decoding process, a vector of carefully selected values[8] is prepended to the set of agents A to be identified as a stop token by the model. When the stop token is selected, the decoding process is terminated.

The full decoding process is shown in Figure 3, which also reports the interaction between the encoder, decoder, and the encoding of the collective, together with the change in the dimensionality that each process involves.

3.1.4. Maximum Entropy Policy Gradient

In the previous section, we defined the attention model that computes the probabilities $\pi_{\boldsymbol{\theta}}(s)$ for the formation of collectives modeled as a decision-making process. In this section, we elaborate on how we optimize the parameters $\boldsymbol{\theta}$ 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 a good-quality solution to the collective formation problem can be obtained by means of an ILP formulation. For our approach to be effective, we have to guarantee that (i) $\mathcal{R}(A)$ contains collectives associated with high utility values by the function f, but also that (ii) such a set contains a sufficient number of diverse collectives. Such diversity is fundamental because, due to the presence of the non-overlapping constraint in Equation 4, 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 achieving a final solution of good quality. To reflect the importance of these two complementary aspects, i.e., having a sufficient number of high-utility collectives but also a

[8] The values selected for the vector should not be among the ones possible for regular agents.

sufficient diversity among them, we define the following two-term loss function:

$$\mathcal{L}(\theta|s) = \underbrace{\mathbb{E}_{\pi_{\boldsymbol{\theta}}(C|s)}\left[f(C)\right]}_{\text{Quality}} + \underbrace{\tau \cdot \mathcal{H}(\pi_{\boldsymbol{\theta}}(s))}_{\text{Diversity}}, \tag{13}$$

where $\mathcal{H}(\pi_{\boldsymbol{\theta}}(s))$ is the entropy of the model at state s and τ weights the contribution of the entropy to the loss function. In Section 4.5 we empirically evaluate the impact of such an entropy term controlled by τ on the performance of our approach. Optimizing the first term in Equation 13 produces a policy that builds collectives of high utility. In addition, we consider a second entropy term to the loss, whose objective is to foster diversity.

Similar to [6], we optimize our model by gradient descent with the well-known REINFORCE algorithm [49]. However, in contrast to such work, the gradient considers the two terms in Equation 13. Thus, we optimize:

$$\nabla_{\boldsymbol{\theta}}\mathcal{L} = \mathbb{E}_{\pi_{\boldsymbol{\theta}}(C|s)}\left[\left(f(C) - b(s)\right)\nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(C|s)\right] + \tau \nabla_{\boldsymbol{\theta}}\mathcal{H}(\pi_{\boldsymbol{\theta}}(s)), \tag{14}$$

where $b(s)$ is a baseline to reduce the variance of the gradient. Popular choices for the baseline are using an exponential moving average [49] or training a critic to estimate value function given a state s [58]. 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 a rollout from a given state with the best policy obtained so far. We also considered more advanced RL techniques such as *proximal policy optimization* (PPO) [59] or the *policy optimization with multiple optima* (POMO) baseline proposed by Kwon et al. [60], which nonetheless did not produce any improvement in our case. Therefore, following Kool et al. [6] we opted for a simpler REINFORCE approach with a rollout baseline for the sake of simplicity.

After the loss is computed, the model parameters $\boldsymbol{\theta}$ are updated using an Adam optimizer [61]. 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 employing a paired t-test. In the case that the model

Algorithm 2 REINFORCE with Rollout Baseline

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

Output: trained model parameters $\boldsymbol{\theta}$

```
Init $\boldsymbol{\theta}$ and $\boldsymbol{\theta}_{BL}$
for $epoch = 1, \dots, E$ do
    for $iter = 1, \dots, I$ do
        $s_i \leftarrow$ randomState()    $\forall i \in \{1, \dots, B\}$
        $C_i \leftarrow$ rollout$(s_i, \boldsymbol{\theta})$    $\forall i \in \{1, \dots, B\}$
        $C_i^{BL} \leftarrow$ rollout$(s_i, \boldsymbol{\theta}_{BL})$    $\forall i \in \{1, \dots, B\}$
        $\nabla\mathcal{L} \leftarrow \sum_{i=1}^{B} \left(f(C_i) - f(C_i^{BL})\right) \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(C_i|s_i)$
        $\nabla\mathcal{L}_{\mathcal{H}} \leftarrow \tau \sum_{i=1}^{B} \nabla_{\boldsymbol{\theta}} \mathcal{H}(\pi_{\boldsymbol{\theta}}(s_i))$
        $\boldsymbol{\theta} \leftarrow$ Adam$(\boldsymbol{\theta}, \nabla\mathcal{L} + \nabla\mathcal{L}_{\mathcal{H}})$
    end for
    if OneSidedPairedTTest$(\pi_{\boldsymbol{\theta}}, \pi_{\boldsymbol{\theta}_{BL}}) \leq \alpha$ then
        $\boldsymbol{\theta}_{BL} \leftarrow \boldsymbol{\theta}$
    end if
end for
return $\boldsymbol{\theta}$
```

outperforms the baseline, the last one is updated with the model parameters. The full training procedure is detailed in Algorithm 2.

3.1.5. Soft Baseline Update

Another popular update for the baseline network parameters adopted by several RL approaches [62, 63], consists of updating the parameters after each iteration by computing a moving average in the direction of the main network parameters, i.e.,

$$\boldsymbol{\theta}_{BL} \leftarrow \lambda \cdot \boldsymbol{\theta} + (1 - \lambda) \cdot \boldsymbol{\theta}_{BL}, \tag{15}$$

where λ is the size of the step towards the network parameters. This update rule achieves a more uniform change of the baseline parameters when compared to the t-test, which updates the baseline only when it is outperformed by the model by directly copying its weights. The soft update produces a more robust training setup, usually leading to a better overall performance of the models. We have observed that this improvement was significant in the ridesharing domain.

This update is introduced at the end of the inner for loop in Algorithm 2, and the t-test performed in line 11 is removed, as its purpose is already covered by the soft update. We compare the performance of the soft baseline update with the t-test update in the experimental analysis, which is discussed hereafter.

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 [3], where, as the authors show, an algorithm strongly characterized by a greedy nature 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 [5], in which greedy approaches cannot be used due to the presence of domain-specific constraints. We discuss both these domains in more detail in the following section.

4.1. Application Domains

The ridesharing scenario discussed in [3] takes place in a map of zones $\mathcal{Z} = \{z_1, z_2, \ldots, z_m\}$. An instance of this problem involves a pool of agents $A = \{a_1, a_2, \ldots, a_n\}$, where each agent wants to travel from an origin to a destination, formally $a_i \in \mathcal{Z} \times \mathcal{Z}$. In this domain, we consider collectives with cardinality $1 \leq |C| \leq 5$ to reflect the usual capacity of cars. Each collective has an associated value assigned by a utility function $f(C)$:

$$f(C) = \sigma \cdot E(C) + (1 - \sigma) \cdot Q(C), \tag{16}$$

where $E(C)$ quantifies the environmental benefits of forming the collective S, $Q(C)$ quantifies the quality of service incurred by the members of C, and $\sigma \in$

$[0, 1]$ controls the importance of each of the above-mentioned components. In our experiments, we use $\sigma = 0.5$ to obtain equal importance between environmental benefits and quality of service. We refer the reader to [3] for more details about each term in the utility function.

We also remark that, in this paper, we only focus on the one-shot optimization problem of forming collectives inherent in the ridesharing domain discussed in [3]. Other aspects of the dynamic ridesharing scenario originally discussed by the authors have been left out for the sake of conciseness since they are orthogonal to the problem tackled here.[9]

The team formation problem discussed in [5] consists of a set of students $A = \{a_1, a_2, \ldots, a_n\}$, which are assigned to a task that has to be solved cooperatively in a team. Andrejczuk et al. [5] study four different tasks: body rhythm, entrepreneur, art design, and English. In all our experiments, we consider the "English" task. Each student is represented by a tuple $(g, \boldsymbol{p}, \boldsymbol{l})$, where g is a binary value indicating the gender, $\boldsymbol{p}$ is a personality vector with four dimensions: sensing-intuition, thinking-feeling, extroversion-introversion, perception-judgment, each one evaluated in the range $[-1, 1]$. $\boldsymbol{l}$ is a vector measuring seven competence levels in the range $[0, 1]$ including linguistic, logic mathematics, visual-spatial, bodily-kinesthetic, musical, intrapersonal and interpersonal. For each task, different competencies need to be covered by at least one student in the team. We consider an utility function $f(C)$ proposed by Andrejczuk et al. [5], which assigns a value to each team:

$$f(C) = \psi \cdot u_{prof}(C, task) + (1 - \psi) \cdot u_{con}(C), \tag{17}$$

where $u_{prof}(C, task)$ measures the proficiency of a team C on a given $task$, $u_{con}(C)$ measures the congeniality of a team, and $\psi \in [0, 1]$ controls the rela-

[9] As discussed in Section 4.2, we compare our attention-based approach with the PG^2-based approach by Bistaffa et al. [3], as they both solve the problem of the formation of collectives. Since such approaches are interchangeable, one could apply the same methodology detailed in [3, Section III-A] to build a dynamic ridesharing solution relying on our attention-based approach. Such an exercise has been left out since it is not in the scope of the current work.

tive importance between this two terms. For more details on how each term is computed, we refer the reader to [5]. Notice that, since the goal of the team formation scenario proposed in Andrejczuk et al. [5] is to obtain a balanced set of teams to foster cooperation and inclusiveness, the authors originally defined the corresponding collective 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 linearized formalization.

For both collective formation domains, our experimental evaluation considers the same real-world dataset employed by the authors of the original works. On the one hand, for ridesharing, we consider the "TLC Trip Record Data" provided by New York City Taxi and Limousine Commission.[10] On the other hand, for team formation we consider the data collected by Andrejczuk et al. [5], corresponding to 210 students, each one identified by gender information, the personality profile, and the competence levels regarding seven competencies. We report results for different problem sizes, i.e., for sets of agents of different sizes. Each result reports the corresponding number n of agents.

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 [3] and [5], which we denote as PG^2 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 PG^2 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 performance comparable to these approaches without being restricted to any

[10] Available at `https://www.nyc.gov/site/tlc/about/tlc-trip-record-data.page`.

specific application domain.

We do not report results for state-of-the-art complete CSG approaches discussed in Section 2.1.1 since they cannot handle the number of agents we consider in our experiments.

Additionally, we compare our approach to the MCTS algorithm presented in [7], which, to the best of our knowledge, is the most recent general approach for the formation of collectives. According to [7], such an 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 in [7] from finding any feasible solution in this case.

For this reason, we decided to consider a second version of MCTS that employs a heuristic preventing the choice of actions leading to a potentially unfeasible collective during rollout. This heuristic uses symmetries in the search space to avoid branches of the tree which lead to a permutation of a collective already explored. 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. Empirical Methodology

The empirical analysis is composed of two parts: training and evaluation. Training concerns the optimization of the model parameters towards learning the formation of collectives for each application domain employing the RL techniques described in the previous sections. The evaluation comprises the study and analysis of these models, once they have been optimized, comparing them to other state-of-the-art approaches for the formation of collective in each studied domain. Note that training only needs to be performed once for each domain, and then, evaluation can be executed several times with different instances. The obtained results are presented at the end of this section.

Training. The attention-based model is trained with ∼200000 instances obtained by sampling from the generators of problem instances provided by the authors of the articles of the two considered case studies [3, 5]. Training has been performed for 100 epochs, using a batch size of 256 instances and a learning rate of 0.0001. Training times may vary depending on the size of the training instances, but in general, it takes between 12 and 24 hours on the employed machine (2.20 GHz CPU, 128 GB RAM, and an NVIDIA RTX 2080 Ti GPU).

Evaluation. We test the algorithms mentioned for each application domain and compare them with our approach by employing a separate dataset only for evaluation purposes, obtained by employing the generators of problem instances mentioned in the previous paragraph. Specifically, we evaluate the different approaches with 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., $0, \ldots, 49$), and we compute the optimality ratio, i.e., the ratio between the average of the obtained solution values and the value of the optimal solution, which is obtained by solving Equation 4 to optimality. We then report the average over all instances of such optimality ratios. We do not report the standard deviations since, in all experiments, it is less than 0.02.

For ridesharing, we consider a time budget of 300 seconds for all approaches. For team formation, we consider the runtime of SynTeam as a time budget (see caption of Table 2), since for this approach it is not possible to set one. Given the total time budget, the portion $k \in [0, 1]$ devoted to the generation of candidates was determined by using IRACE [53], a widely used software for tuning algorithmic parameters.

Hyperparameters. We determined the values of the hyperparameters by starting from the values reported in the relevant literature [6] and adapting them, by means of some iterative tests, to the values that we found to work best in our case, reported hereafter. Specifically, we initialize the model and baseline parameters with $Uniform(-1/\sqrt{d},\, 1/\sqrt{d})$, where d is the input size. For the attention mechanism, we use $N = 8$ heads and $d_h = 256$, whereas, for the feed-forward layers, we use $d_h = 512$. The encoder is composed of $N = 3$ attention blocks. The total number of model parameters is 2463488. 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 each. For the 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. Our attention-based model is implemented in PyTorch.[11] We employ CPLEX 22.1 as an ILP solver.

4.4. Solution Quality

We now proceed to discuss the results of our comparison between our approach and the baselines discussed in Section 4.2 on the two considered collective formation domains. Table 1 reports the results of our experiments on the ridesharing domain. Our best-performing approach is comparable with PG^2 for $n = 50$, but PG^2 still outperforms our model for higher sizes. This result is not surprising, since PG^2 has been specifically designed for this domain.

Results also show that the optimality ratio obtained by our approach is superior compared to the MCTS approaches (including R-MCTS, the only MCTS approach in our comparison that does not incorporate any domain-specific subroutine), which cannot compute a solution of acceptable quality. Moreover, we observed that for $n = 200$ the G-MCTS approach by Wu and Ramchurn [7] could not compute a solution better than the initial one (i.e., all singletons with a total utility of 0) in the considered time budget. A possible explanation could be that the utility function we consider involves more computation than the ones studied in [7], where utilities are directly obtained by sampling from a uniform distribution. These results could suggest that complex utility functions might hinder the performance of MCTS in large-scale scenarios.

As for the update rule on the baseline, we observed that the approach implementing the soft update is superior to the one implementing the t-test update.

Table 2 reports the results of our experiments on the team formation domain.

[11] Our implementation is available at `https://github.com/filippobistaffa/trans-ilp`.

n	AM (t-test)	AM (soft)	G-MCTS	A-MCTS	R-MCTS	PG^2
50	0.93	0.96	0.92	0.14	0.24	0.98
100	0.85	0.90	0.89	0.04	0.09	0.98
200*	0.74	0.87	–	0.01	0.05	1.00

Table 1: Optimality ratio for the attention model (AM) and baseline approaches in the ridesharing domain. For all experiments, we use a time budget of 300 seconds. Missing values ("–") indicate that the approach did not compute any solution better than the initial one within the time budget. *For $n = 200$ we report the ratio with respect to the solution computed by PG^2 since computing the optimal in a manageable amount of time is not possible.

As for ridesharing, the optimality ratio obtained in that case is significantly better than the one obtained by other MCTS approaches, even the one we specifically adapted for this domain (i.e., A-MCTS). Moreover, by comparing our approach to SynTeam, we can see that the gap between our approach and the domain-specific approach for team formation is smaller than for ridesharing, especially for the larger problem instances.

In contrast to the ridesharing domain, we have not observed a clear difference between the t-test update and the soft update for the baseline for team formation. This might be because the optimality ratio is already quite good compared to the one in the ridesharing domain, hence the impact of soft-update in team formation is less pronounced.

4.5. Impact of Entropy

One of the most important components of our model is the entropy term in Equation 14 controlled by the parameter τ, which allows the attention model to generate a higher variety of candidates, hence providing more options to the ILP solver and resulting in a better overall performance of the approach. In this section, we aim to evaluate the impact of entropy on the variety of candidates generated for the ridesharing and team formation domains. To this end, we conducted a second set of experiments in which we compare the distribution over

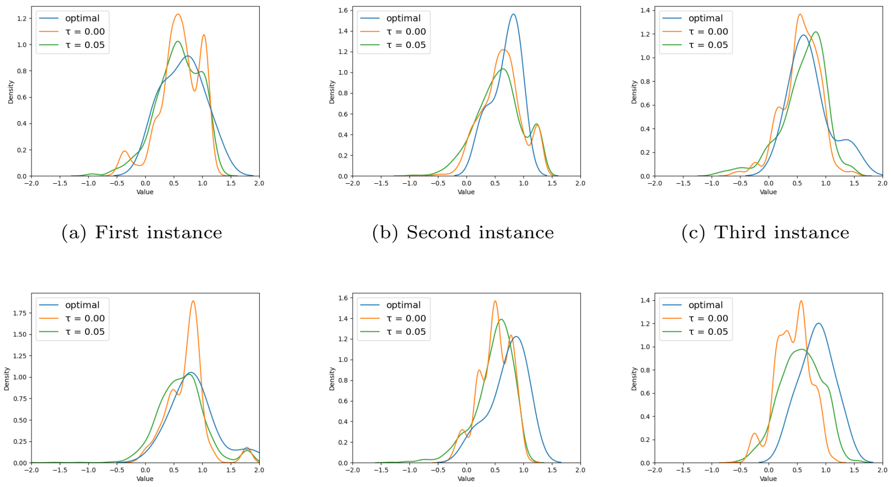
n	AM (t-test)	AM (soft)	G-MCTS	A-MCTS	R-MCTS	ST
50	0.97	0.97	–	0.84	–	0.99
60	0.95	0.93	–	0.78	–	0.99
100*	0.92	0.91	–	–	–	1.00

Table 2: Optimality ratio for the attention model (AM) and baseline approaches in the team formation domain. ST indicates the results for the SynTeam approach. For all experiments, we use the runtime of SynTeam as a time budget (i.e., 60 seconds for $n = 50$ and $n = 60$, and 180 seconds for $n = 100$). Missing values ("–") indicate that the approach did not compute any solution better than the initial one within the time budget. *For $n = 100$ we report the ratio with respect to the solution computed by SynTeam, since computing the optimal in a manageable amount of time is not possible.

the values of the collectives generated by the model with and without entropy ($\tau = 0.05$ and $\tau = 0.00$, respectively) to the distribution over the values of the collectives in the optimal solution. Increasing the τ parameter over 0.05 affected negatively the convergence of the model during training. All these distributions are obtained by performing a Gaussian Kernel Density Estimation [64] over thousands of collectives generated by our model, given a problem instance.

Figures 5 and 6 report the distribution over the values for six different example instances of the ridesharing and team formation domains. By looking at the distribution of the values, it can be observed that the model with higher entropy covers a wider range of values compared to the one with lower entropy. In this respect, we observed that the model with higher entropy produces distributions that are closer to the optimal ones when compared with the distribution produced by the model with lower entropy. Indeed, the model performs better if it generates a distribution of values similar to the ones in the optimal solution.

To measure such a similarity, we compute the Kullback-Leibler divergence between the distribution produced by a model and the distribution of values in the optimal solution. We observe that the distribution of values obtained with higher entropy is closer to the optimal one in 157 instance over a total of 200 in the ridesharing domain. Moreover, the average divergence over these samples is



(a) First instance

(b) Second instance

(c) Third instance

(d) Fourth instance

(e) Fifth instance

(f) Sixth instance

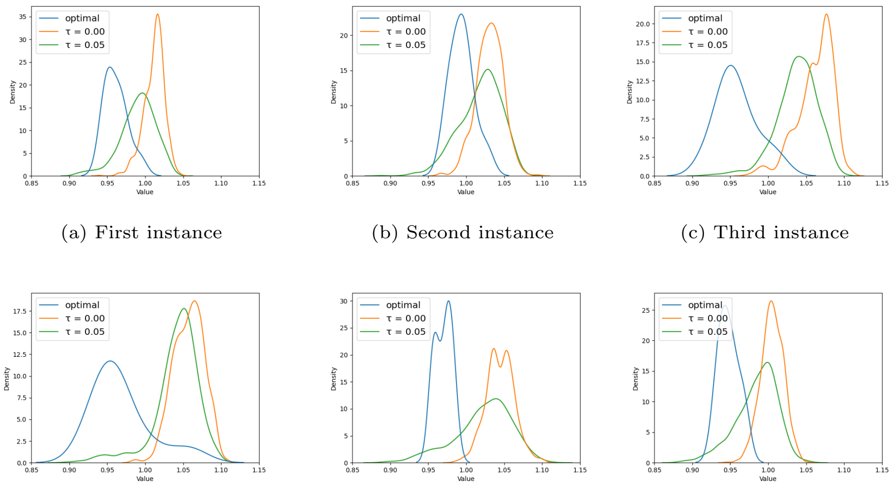
Figure 5: Probability density of the values of collectives generated by our attention model employing $\tau = 0.00$ and $\tau = 0.05$ for different ridesharing instances.

lower for the model with higher entropy, and the p-value is $3 \cdot 10^{-6}$. The analysis in the team formation domain further confirms these results. In this case, the model with higher entropy produces a distribution closer to the optimal one for all test instances.

Overall, by observing these results we conclude that entropy is beneficial since the model associated with a higher entropy can generate collectives with lower values that, when combined with higher-valued ones, lead to better solutions computed by the ILP.

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 showing superior performance.



(a) First instance

(b) Second instance

(c) Third instance

(d) Fourth instance

(e) Fifth instance

(f) Sixth instance

Figure 6: Probability density of the values of collectives generated by our attention model employing $\tau = 0.00$ and $\tau = 0.05$ for different team formation instances.

Further reducing the gap with respect to domain-specific approaches is an important direction for future research. We believe that investigating alternatives to introducing diversity in the generation of collectives is a promising line of research since entropy plays an important role in the success of our attention model. Furthermore, it might be interesting to investigate whether making the attention model “aware” of the candidates previously proposed (rather than proposing each candidate independently, as we currently do in this paper) can provide some improvements.

Overall, we believe that this work is the 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 is an indication that our attention model can indeed learn the inherent structure of the domain, allowing our approach to compute solutions of satisfactory quality.

Acknowledgment

The authors gratefully acknowledge the computer resources at Artemisa, funded by the European Union ERDF and Comunitat Valenciana as well as the technical support provided by the Instituto de Fisica Corpuscular, IFIC (CSIC-UV). This work was supported by the "ACISUD" project (PID2022-136787NB-I00) funded by MCIN/AEI/10.13039/501100011033 and by the "YOMA Operational Research" project (OPE02570) funded by the Botnar Foundation.

References

[1] European Commission, Collective awareness platforms for sustainability and social innovation, `https://ec.europa.eu/digital-single-market/en/collective-awareness`, 2021.

[2] United Nations, Sustainable development goals, `https://www.un.org/sustainabledevelopment/sustainable-development-goals`, 2015.

[3] F. Bistaffa, C. Blum, J. Cerquides, A. Farinelli, J. A. Rodríguez-Aguilar, A computational approach to quantify the benefits of ridesharing for policy makers and travellers, IEEE Transactions on Intelligent Transportation Systems 22 (2021) 119–130.

[4] J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli, D. Rus, On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment, Proceedings of the National Academy of Sciences 114 (2017) 462–467.

[5] E. Andrejczuk, F. Bistaffa, C. Blum, J. A. Rodríguez-Aguilar, C. Sierra, Synergistic team composition: A computational approach to foster diversity in teams, Knowledge-Based Systems 182 (2019) 104799.

[6] W. Kool, H. van Hoof, M. Welling, Attention, learn to solve routing problems!, in: Proceedings of the International Conference on Learning Representations, 2019, pp. 1–25.

[7] F. Wu, S. D. Ramchurn, Monte-carlo tree search for scalable coalition formation, in: Proceedings of the International Joint Conference on Artificial Intelligence, 2020, pp. 407–413.

[8] G. Chalkiadakis, E. Elkind, M. Wooldridge, Computational Aspects of Cooperative Game Theory, Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan and Claypool Publishers, 2011.

[9] F. Präntare, F. Heintz, An anytime algorithm for optimal simultaneous coalition structure generation and assignment, Autonomous Agents and Multi-Agent Systems 34 (2020) 1–31.

[10] J. Cerquides, A. Farinelli, P. Meseguer, S. D. Ramchurn, A tutorial on optimization for multi-agent systems, The Computer Journal 57 (2014) 799–824.

[11] O. Shehory, S. Kraus, Methods for task allocation via agent coalition formation, Artificial intelligence 101 (1998) 165–200.

[12] R. B. Myerson, Graphs and cooperation in games, Mathematics of Operations Research 2 (1977) 225–229.

[13] T. Rahwan, T. P. Michalak, M. Wooldridge, N. R. Jennings, Coalition structure generation: a survey, Artificial Intelligence 229 (2015) 139–174.

[14] C.-H. Lin, Corporate tax structures and a special class of set partitioning problems, Ph.D. thesis, Case Western Reserve University Cleveland, OH, USA, 1975.

[15] N. Changder, S. Aknine, S. Ramchurn, A. Dutta, Odss: efficient hybridization for optimal coalition structure generation, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2020, pp. 7079–7086.

[16] T. Michalak, T. Rahwan, E. Elkind, M. Wooldridge, N. R. Jennings, A hybrid exact algorithm for complete set partitioning, Artificial Intelligence 230 (2016) 14–50.

[17] T. Voice, M. Polukarov, N. R. Jennings, Coalition structure generation over graphs, Journal of Artificial Intelligence Research 45 (2012) 165–196.

[18] F. Bistaffa, A. Farinelli, G. Chalkiadakis, S. D. Ramchurn, A cooperative game-theoretic approach to the social ridesharing problem, Artificial Intelligence 246 (2017) 86–117.

[19] T. Voice, S. D. Ramchurn, N. R. Jennings, On coalition formation with sparse synergies, in: Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems, 2012, pp. 223–230.

[20] F. Bistaffa, A. Farinelli, J. Cerquides, J. Rodríguez-Aguilar, S. D. Ramchurn, Algorithms for graph-constrained coalition formation in the real world, ACM Transactions on Intelligent Systems and Technology 8 (2017) 1–24.

[21] S. Ieong, Y. Shoham, Marginal contribution nets: a compact representation scheme for coalitional games, in: Proceedings of the ACM Conference on Electronic Commerce, 2005, pp. 193–202.

[22] L. Tran-Thanh, T.-D. Nguyen, T. Rahwan, A. Rogers, N. R. Jennings, An efficient vector-based representation for coalitional games, in: Proceedings of the International Joint Conference on Artificial Intelligence, 2013, pp. 383–389.

[23] F. Bistaffa, G. Chalkiadakis, A. Farinelli, Efficient coalition structure generation via approximately equivalent induced subgraph games, IEEE Transactions on Cybernetics 52 (2021) 5548–5558.

[24] X. Deng, C. Papadimitriou, On the complexity of cooperative solution concepts, Mathematics of Operations Research 19 (1994) 257–266.

[25] M. E. Gaston, M. desJardins, Agent-organized networks for dynamic team formation, in: Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems, 2005, pp. 230–237.

[26] T. Lappas, K. Liu, E. Terzi, Finding a team of experts in social networks, in: Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2009, pp. 467–476.

[27] L. S. Marcolino, A. X. Jiang, M. Tambe, Multi-agent team formation: Diversity beats strength?, in: Proceedings of the International Joint Conference on Artificial Intelligence, 2013, pp. 279–285.

[28] S. Liemhetcharat, M. Veloso, Weighted synergy graphs for effective team formation with heterogeneous ad-hoc agents, Artificial Intelligence 208 (2014) 41–65.

[29] A. Georgara, J. A. Rodríguez-Aguilar, C. Sierra, Towards a competence-based approach to allocate teams to tasks, in: Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems, 2021, pp. 1504–1506.

[30] A. Georgara, J. A. Rodríguez-Aguilar, C. Sierra, O. Mich, R. Kazhamiakin, A. Palmero Approsio, J.-C. Pazzaglia, An anytime heuristic algorithm for allocating many teams to many tasks, in: Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems, 2022, pp. 1598–1600.

[31] A. Farinelli, M. Bicego, F. Bistaffa, S. D. Ramchurn, A hierarchical clustering approach to large-scale near-optimal coalition formation with quality guarantees, Engineering Applications of Artificial Intelligence 59 (2017) 170–185.

[32] Y. Bengio, A. Lodi, A. Prouvost, Machine learning for combinatorial optimization: a methodological tour d'horizon, European Journal of Operational Research (2020).

[33] O. Vinyals, M. Fortunato, N. Jaitly, Pointer networks, in: Advances in Neural Information Processing Systems, 2015, pp. 2692–2700.

[34] C. K. Joshi, T. Laurent, X. Bresson, An efficient graph convolutional network technique for the travelling salesman problem, arXiv preprint arXiv:1906.01227 (2019).

[35] M. Nazari, A. Oroojlooy, L. Snyder, M. Takác, Reinforcement learning for solving the vehicle routing problem, Advances in Neural Information Processing Systems 31 (2018).

[36] J. James, W. Yu, J. Gu, Online vehicle routing with neural combinatorial optimization and deep reinforcement learning, IEEE Transactions on Intelligent Transportation Systems 20 (2019) 3806–3817.

[37] J.-Y. Ding, C. Zhang, L. Shen, S. Li, B. Wang, Y. Xu, L. Song, Accelerating primal solution findings for mixed integer programs based on solution prediction, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2020, pp. 1452–1459.

[38] Z. Li, Q. Chen, V. Koltun, Combinatorial optimization with graph convolutional networks and guided tree search, Advances in Neural Information Processing Systems 31 (2018).

[39] A. S. Xavier, F. Qiu, S. Ahmed, Learning to solve large-scale security-constrained unit commitment problems, INFORMS Journal on Computing 33 (2021) 739–756.

[40] Y. Sun, X. Li, A. Ernst, Using statistical measures and machine learning for graph reduction to solve maximum weight clique problems, IEEE Transactions on Pattern Analysis and Machine Intelligence 43 (2019) 1746–1760.

[41] P. Kilby, J. Slaney, T. Walsh, et al., The backbone of the travelling salesperson, in: Proceedings of the International Joint Conference on Artificial Intelligence, 2005, pp. 175–180.

[42] A. Hottung, K. Tierney, Neural large neighborhood search for the capacitated vehicle routing problem, in: Proceedings of the European Conference on Artificial Intelligence, 2020, pp. 443–450.

[43] E. Khalil, P. Le Bodic, L. Song, G. Nemhauser, B. Dilkina, Learning to branch in mixed integer programming, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2016, pp. 724–731.

[44] M. Gasse, D. Chételat, N. Ferroni, L. Charlin, A. Lodi, Exact combinatorial optimization with graph convolutional neural networks, in: Advances in Neural Information Processing Systems, 2019, pp. 15580–15592.

[45] V. Nair, S. Bartunov, F. Gimeno, I. von Glehn, P. Lichocki, I. Lobov, B. O'Donoghue, N. Sonnerat, C. Tjandraatmadja, P. Wang, et al., Solving mixed integer programs using neural networks, arXiv preprint arXiv:2012.13349 (2021).

[46] M. Kruber, M. E. Lübbecke, A. Parmentier, Learning when to use a decomposition, in: Proceedings of the International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research, 2017, pp. 202–210.

[47] N. Karalias, A. Loukas, Erdos goes neural: an unsupervised learning framework for combinatorial optimization on graphs, in: Advances in Neural Information Processing Systems, 2020, pp. 6659–6672.

[48] I. Bello, H. Pham, Q. V. Le, M. Norouzi, S. Bengio, Neural combinatorial optimization with reinforcement learning, arXiv preprint arXiv:1611.09940 (2016).

[49] R. J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, Machine Learning 8 (1992) 229–256.

[50] V. Konda, J. Tsitsiklis, Actor-critic algorithms, Advances in Neural Information Processing Systems 12 (1999).

[51] Y. Zhang, L. Bliek, P. da Costa, R. R. Afshar, R. Reijnen, T. Catshoek, D. Vos, S. Verwer, F. Schmitt-Ulms, A. Hottung, T. Shah, M. Sellmann, K. Tierney, C. Perreault-Lafleur, C. Leboeuf, F. Bobbio, J. Pepin,

W. A. Silva, R. Gama, H. L. Fernandes, M. Zaefferer, M. López-Ibáñez, E. Irurozki, The first AI4TSP competition: Learning to solve stochastic routing problems, Artificial Intelligence 319 (2023) 103918.

[52] R. M. Karp, Reducibility among Combinatorial Problems, Springer, 2010.

[53] M. López-Ibáñez, J. Dubois-Lacoste, L. P. Cáceres, M. Birattari, T. Stützle, The IRACE package: Iterated racing for automatic algorithm configuration, Operations Research Perspectives 3 (2016) 43–58.

[54] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.

[55] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, 2017, pp. 5998–6008.

[56] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[57] J. L. Ba, J. R. Kiros, G. E. Hinton, Layer normalization, arXiv preprint arXiv:1607.06450 (2016).

[58] A. G. Barto, R. S. Sutton, C. W. Anderson, Neuronlike adaptive elements that can solve difficult learning control problems, IEEE Transactions on Systems, Man, and Cybernetics (1983) 834–846.

[59] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, arXiv preprint arXiv:1707.06347 (2017).

[60] Y. Kwon, J. Choo, B. Kim, I. Yoon, Y. Gwon, S. Min, POMO: policy optimization with multiple optima for reinforcement learning, Advances in Neural Information Processing Systems 33 (2020).

[61] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).

[62] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, arXiv preprint arXiv:1509.02971 (2015).

[63] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: Proceedings of the International Conference on Machine Learning, 2018, pp. 1861–1870.

[64] B. W. Silverman, Density Estimation for Statistics and Data Analysis, Routledge, 2018.