Edu2Com: an anytime algorithm to form student teams in companies.*

Athina Georgara† , **Carles Sierra** , **Juan A. Rodríguez-Aguilar**
Artificial Intelligence Research Institute
Spanish National Research Council (IIIA-CSIC)
{ageorg, sierra, jar}@iiia.csic.es

Abstract

In this paper we consider the problem of forming student teams adequate for company intership tasks. First, we provide a formalisation of the *Feasible Team-For-Task Allocation Problem*, and show the computational hardness of solving it optimally. Thereafter, we propose *Edu2Com*, an anytime heuristic algorithm that generates an initial team allocation that is then improved in an iterative process. Finally, we conduct a systematic evaluation and show that Edu2Com manages to *(a)* outperform CPLEX in computation time, and *(b)* reach optimality, in the experiments considered.

1 Introduction

In the context of education, it is increasingly common that students spend some time doing practical work in a company as part of their curriculum. This work is sometimes remunerated: companies benefit from this program as they get motivated students that will work for reduced wages, and students benefit from a first contact with the labour market. It has been found that the employability of students at the end of their studies increases thanks to these internships. Nowadays, education authorities match students with companies mostly by hand. This paper formalises this matching process as a combinatorial optimization problem, proposes an anytime heuristic algorithm that solves the combinatorial problem, and studies its computational complexity.

In what follows, in Sec 2 we formally describe the *Feasible Team-For-Task Allocation Problem* (FTAP), provide formal definitions for the problem's components, and study its complexity. In Sec 3 we provide the encoding for a linear program solver. In Sec 4 we propose our heuristic algorithm; while in Sec 5 we conduct a systematic evaluation to show its effectiveness.

*Research supported by projects AI4EU (H2020-825619), Wenet (H2020-FETPROACT-2018-01), LOGISTAR (H2020-769142), and 2019DI17.

†Contact Author

2 FTAP formalisation

Here we present the main components of the problem, and related concepts.

2.1 Basic elements of the allocation problem

An *internship program* is a specified task that bridges the educational and industrial worlds. That is, an internship program, or simply a task,[1] is characterised by a set of requirements on competencies acquired through education, and aims at applying them in a practical context. Along with the required competencies, a task may require some times more than one student. In general, we can have a large variety of other constraints, such as temporal or spatial constraints; however, within the scope of this work, we only focus on team size constraints. Formally, a task t is a tuple $\langle C, w, m\rangle$, where C is the set of required competencies, $w : C \to (0, 1]$ is a function that weighs the importance of competencies, and $m \in \mathbb{N}_+$ is the team size. The set of all tasks is denoted as T, $|T| = M$.

We also have *students*,[2] i.e., the individuals who will form teams in order carry out a task. An agent is characterised by the competencies acquired via education. Formally, an agent a is represented by a set of already acquired competencies C. The set of all agents is denoted as A, $|A| = N$. Given $t \in T$, we denote the set of all size-compliant teams for t as $\mathcal{K}_t = \{K \subseteq A : |K| = m_t\}$.[3]

In order to quantify the suitability of a team $K \in \mathcal{K}_t$ and develop a comparison measure for the different teams, we turn to the relations between a task's required competencies and a team's acquired competencies. That is, we assume the existence of a fixed and finite set of competencies, encoded in a tree graph representation where a child-node is a refined version of its parent-node. We define the semantic similarity between two competencies as: $\mathrm{sim}(c_1, c_2) = \begin{cases} 1, & \text{if } l = 0 \\ e^{-\lambda l}\frac{e^{\kappa h}-e^{-\kappa h}}{e^{\kappa h}+e^{-\kappa h}}, & \text{otherwise} \end{cases}$, where l is the shortest path in the competencies' tree between c_1 and c_2, h is the depth

[1] In the rest of the paper we will use the term task.

[2] For generality we will denote students by *agents* as the results apply in a broader context.

[3] Note: we use the subscript a in C to refer to the set of competencies of an agent $a \in A$, and the subscript t to refer to the same elements of task $t \in T$.

of the deepest competence subsuming both c_1 and c_2, and $\kappa, \lambda \in [1, 2]$ are parameters regulating the influence of l and h on the similarity metric. This is a variation of the metric introduced in [Li *et al.*, 2003], to guarantee the reflexive property of similarity.

Next, we adopt the notion of *coverage* to determine the fitness degree of agent $a \in A$ for competence c. We define a's coverage for competence c as: $\mathrm{cvg}(c, a) = \max_{c' \in C_a}\{\mathrm{sim}(c, c')\}$. Now, given a task $t \in T$, we define agent a's coverage for task t as: $\mathrm{cvg}(t, a) = \prod_{c \in C_t} \mathrm{cvg}(c, a)$. To determine a team's fitness for task t, we need to solve a *competence assignment problem*. That is, we let each agent commit themselves to a subset of the required competencies instead of the whole set, in order to guarantee the successful completion of the task by sharing duties and responsibilities. Following [Andrejczuk *et al.*, 2019] we have:

Definition 1 (*Competence Assignment Function (CAF)*)**.** *Given a task $t \in T$, and a team of agents $K \in \mathcal{K}_t$, a competence assignment η_t^K is a function $\eta_t^K : K \to 2^{C_t}$, satisfying $C_t = \bigcup_{a \in K} \eta_t^K(a)$.*

The reverse function $\theta_t^K : C_t \to 2^K$ provides us with the set of agents in K that are assigned to competence $c \in C_t$.

However, not all CAFs are equally good. Intuitively, and specially within the domain of education a notion of *fairness* in assigning responsibilities to students is needed. As such, in this work we are seeking CAFs that allow each agent to actively participate in the task (inclusive assignments *wrt* [Andrejczuk, 2018]), and at the same time no agent is overloaded with excessive responsibilities. Given a task t, and a team of agents $K \in \mathcal{K}_t$, a *Fair Competence Assignment Function (FCAF)* is a CAF that also satisfy two further conditions: *(i)* each agent is responsible for at least one competence, while an upper bound is placed to prevent a 'super-competent' agent take on too many responsibilities, $1 \leq |\eta_t^K(a)| \leq \left\lceil \frac{|C_t|}{|K|} \right\rceil \ \forall a \in K$; and *(ii)* each competence will be at least one agent's responsibility, $1 \leq |\theta_t^K(c)|$. We denote the set of all FCAFs for task t and team K as H_t^K.

Given a task t, and a team $K \in \mathcal{K}_t$, an agent $a \in K$, and FCAF η_t^K, we define the agent's *competence proximity* to the task as $\mathrm{cp}\big(t, a, \eta_t^K\big) = \prod_{c \in \eta_t^K(s)} \max\{1 - w_t(c), \mathrm{cvg}(c, a)\}$, where $w_t(c)$ is the importance weight of competence $c \in C_t$ according to t. Intuitively, the better the coverage an agent has over a competence, the lower the impact of the competence's importance; while the lower the importance of a competence, the lower the impact of the agent's coverage over the competence. That is, if a competence is not very important (e.g., $w(c) \to 0$) then it is not 'catastrophic' if the agent cannot cover this competence very well; while if the agent covers the competence very well, then, regardless the importance of the competence, this is a desirable match. Now, we define the team's competence proximity for a task given a particular FCAF as the Nash product of the competence proximity of each agent.

Definition 2 (*Team's Competence Proximity*)**.** Given a task t, a team $K \in \mathcal{K}_t$, and a competence assignment η_t^K, the competence proximity of team K for t is $\mathrm{cp}(t, K, \eta_t^K) = \prod_{a \in K} \mathrm{cp}(\mathrm{a}, \mathrm{t}, \eta_{\mathrm{t}}^{\mathrm{K}})$.

For a given task, and a given team, different FCAFs result with different competence proximity. The optimum FCAF for t and K is the one that maximises the team's competence proximity: $\eta_t^{K\,*} = \arg\max_{\eta_t^K \in \Theta_t^K}\{\mathrm{cp}\big(t, K, \eta_t^K\big)\}$. Finding the *best* FCAF is an optimisation problem itself. Even though the above is not a linear optimisation problem, it can be linearised by considering the logarithm of $\mathrm{cp}\big(\,\cdot\,\big)$ as follows:

$$\eta_t^{K\,*} = \arg\max_{\eta_t^K \in H^K}\{\mathrm{cp}\big(t, K, \eta_t^K\big)\} \equiv$$

$$\arg\max_{\eta_t^K \in H_t^K} \log\Big\{\prod_{a \in K} \mathrm{cp}(\mathrm{t}, \mathrm{a}, \eta_{\mathrm{t}}^{\mathrm{K}})\Big\} = \arg\max_{\eta_t^K \in H_t^K} \sum_{a \in K} \log\big\{\mathrm{cp}(\mathrm{t}, \mathrm{a}, \eta_{\mathrm{t}}^{\mathrm{K}})\big\}.$$

2.2 The team allocation problem as an optimisation problem

Given a task we aim to find the team that maximises competence proximity, with the understanding that for each candidate team we need to find its optimal FCAF. Formally, for a task t the best team is given by $K^* = \arg\max_{K \in \mathcal{K}_t} \mathrm{cp}\big(t, K, \eta_t^{K\,*}\big)$.

Now, given a collection of tasks T, with $|T| > 1$, we need to assign a team to each task so that the *overall* competence proximity is maximised. Within the setting of internship programs and students we need to take into account the following restriction: each student can be assigned to at most one internship program, and each internship program can be assigned to at most one team. That is, suppose there is a function $g : T \to 2^A$, which maps each $t \in T$ with a team of agents $K \in \mathcal{K}_t$, and G is the family of all such functions.

Definition 3 (*Feasible Team Assignment Function (FTAF)*)**.** Given a set of tasks T and a set of agents A, a *feasible team assignment function* $g \in G$ is such that for each pair of tasks $t_1, t_2 \in T$ with $t_1 \neq t_2$, it holds that $g(t_1) \cap g(t_2) = \emptyset$; and for all $t \in T$ it holds that $|g(t)| = m_t$.

The family of all feasible team assignments is denoted by G_{feasible}. Next, we formalise our team allocation problem:

Definition 4 (*Feasible Team-For-Task Allocation Problem (FTAP)*)**.** Given a set of tasks T, and a set of agents A, the FTAP problem is to select the team assignment function $g^* \in G_{\mathrm{feasible}}$ that maximises the overall competence proximity $g^* = \arg\max_{g \in G_{\mathrm{feasible}}} \prod_{t \in T} \mathrm{cp}\big(g(p), p, \eta_p^{g(p)\,*}\big)$.

The following establishes that FTAP is $\mathcal{NP}$-complete by reduction to a well-known problem.

Theorem 1. *FTAP for more than one task is $\mathcal{NP}$-complete.*

Proof. The problem is in $\mathcal{NP}$ since we can decide whether a given solution is feasible in polynomial time ($\mathcal{O}(\sum_{p \in P} m_p)$). We show that the problem is $\mathcal{NP}$-complete by using a reduction from *Single Unit Auctions with XOR Constraints and Free Disposals* (referred to as BCAWDP with XOR Constraints) which is shown to be $\mathcal{NP}$-complete [Sandholm *et al.*, 2002]. In BCAWDP with XOR Constraints, the auctioneer has N items to sell, the bidders place their bids $B_i = \langle \mathbf{b}_i, b_i \rangle$ with $\mathbf{b}_i$ a subset of items and b_i the price. Between two bids an XOR constraint can exist–not necessarily for every pair of bids. The auctioneer allows free disposals, i.e., items can remain unsold. Given an instance of

BCAWDP with XOR Constraints, we construct an instance of FTAP as follows: "For each item i we create an agent a_i. For each task t_j of size m_{t_j} we create $\binom{|A|}{m_{t_j}}$ different bids $B_{jk} = \langle \mathbf{b}_{jk}, b_{jk} \rangle$, where $|A|$ is the number of items, $|\mathbf{b}_{jk}| = m_{t_i}$, and $b_{jk} = \mathrm{cp}\big(t_j, \mathbf{b}_{jk}, \eta_{t_j}^{\mathbf{b}_{jk}\,*}\big)$. All bids created for task t_j are XOR-constrained bids. Moreover, each pair of bids $B_{j,k}, B_{q,l}$ such that $\mathbf{b}_{jk} \cap \mathbf{b}_{ql} \neq \emptyset$ are also XOR-constrained." Now FTAP has a feasible solution if and only if BCAWDP with XOR constraints has a solution. □

Typically, the winner determination problem for combinatorial auctions can be cast and solved as a linear program.

3 Solving FTAP as a linear program

Here, we encode FTAP (Def. 4) as an LP. First, for each team $K \subseteq A$ and task $t \in T$, we will use a binary decision variable x_K^t. The value of x_K^t indicates whether team K is assigned to task t as part of the optimal solution. Then, solving FTAP amounts to solve the following non-linear program:

$$\max \prod_{t \in T} \prod_{k \in \mathcal{K}_t} \Big(\mathrm{cp}\big(K, t, \eta_t^{K\,*}\big)\Big)^{x_K^t} \tag{2}$$

subject to:

$$\sum_{K \subseteq S} x_K^t \cdot \mathbb{1}_{K \in \mathcal{K}_t} \leq 1 \qquad \forall t \in T \tag{2a}$$

$$\sum_{t \in T} \sum_{K \subseteq A} x_K^t \cdot \mathbb{1}_{a \in K} \cdot \mathbb{1}_{K \in \mathcal{K}_t} \leq 1 \qquad \forall a \in A \tag{2b}$$

$$x_K^t \in \{0, 1\} \qquad \forall K \subseteq A, t \in T \tag{2c}$$

Constraint (2b) ensure us that each agent will be assigned to at most one task; while constraint (2a) guarantees that for each task will be formed at most one team. Notice that the objective function (see Eq 2) is non-linear. Nevertheless, we linearise it by maximising the logarithm of $\prod_{p \in P} \prod_{k \in \mathcal{K}_p} \Big(\mathrm{cp}\big(K, p, \eta_p^{K\,*}\big)\Big)^{x_K^p}$. Thus, solving the non-linear program above is equivalent to solving the following binary linear program:

$$\max \sum_{p \in P} \sum_{K \in \mathcal{K}_p} x_K^p \cdot \log\Big(1 + \mathrm{cp}\big(K, p, \eta_p^{K\,*}\big)\Big) \tag{3}$$

subject to: equations 2a, 2b, and 2c. Therefore, we can solve this LP with the aid of an off-the-shelf LP solver such as, for example, CPLEX [IBM, 2019], Gurobi [GUROBI, 2018], or GLPK [GLPK, 2018]. Given sufficient time, an LP solver will return an optimal solution to FTAP.

At this point, it is worth mentioning that computing the objective function in equation 3 to build the LP requires the precomputation of the values of $\mathrm{cp}\big(K, p, \eta_p^{K\,*}\big)$, which amounts to solve an optimisation problem for each pair of team and task. This is bound to lead to large linear programs, as the number of agents and tasks grow, and to inefficiency as an LP solver is a general-purpose solver that does not exploit the structure of the problem. To improve this, in the next section we introduce the *edu2com* algorithm, an anytime algorithm based on local search that yields approximate solutions to the FTAP. Unlike an LP solver, edu2com is a specialised algorithm that does exploit the structure of FTAP instances.

4 A Heuristic Algorithm for FTATP

Our approach, *edu2com*, consists of two stages: *(a)* find an initial feasible allocation of agents to tasks, and *(b)* iteratively improve the current allocation by means of swaps of members between teams.

Stage A: Initial Assignment We first build an initial good team allocation. We *measure* the tasks *wrt* their *hardness*, taking into account the required competencies, along with the capabilities of *all* available agents. Intuitively, the more agents have high coverage over c, the more certain we are that we can cover it, and therefore the less hard is the competence. In the opposite case, we can be certain that the agents cannot cover well the competence, and therefore the competence is hard. To express this, we let $\mathcal{H} : [0, 1] \to [0, 4 \cdot 0.5 \log(0.5)]$, and use point $x = 0.5$ as the median point of the function. In particular, we measure how hard is to cover a competence c given a subset of agents $A' \subseteq A$ as: $h(c, A') = -1/|A| \cdot \sum_{a \in A} \mathcal{H}(\mathrm{cvg}(c, C_a))$, where: [4]

$$\mathcal{H}(x) = \begin{cases} x \cdot \log(x) + (1 - x) \cdot \log(1 - x) & \text{if } x \geq 0.5 \\ 4 \cdot \mathcal{H}(0.5) - (x \cdot \log(x) + (1 - x) \cdot \log(1 - x)) & \text{otherwise} \end{cases}$$

Given a task t, we measure its hardness, or difficuly, as: $d(t, A) = \omega \cdot \sum_{c \in C_t} \frac{w_t(c)}{h(c, A) + \epsilon}$, where ω is a normalising factor over the importance weights w_t, and ϵ is a small positive number. Function d allows us to rank the tasks by hardness.

Once we have evaluated the hardness of all tasks, we proceed by greedily assigning teams starting from the hardest task. This is done by the following process: for task t we sort the required competences according to their importance; then we sort in a list the available agents so that the first agent covers best the most important competence, the second agent covers best the second most important competence, and so forth. The team assigned to t consists of the m_t first agents out of the sorted list with the available agents.

Stage B: Improve the Assignment In the second stage we try to improve the team allocation in an iterative process. At each iteration we take one or more of the following actions:

1. For a random pair of tasks, find their best possible team allocation using just the agents in their currently assigned teams.
2. If there are unassigned agents, attempt to swap anyone of them with an assigned agent, and keep the change if the competence proximity improves.
3. Force a systematic 'local search': for *every* pair of tasks, attempt to swap *any* pair of agents.

Within an iteration, we always perform action 1. In case there are unassigned agents we perform action 2. Action 3 is performed after a fixed number of iterations.

5 Empirical Analysis

The purpose of this section is to empirically evaluate our algorithm along four directions: *(a)* the quality of the solutions that it produces in terms of optimality; *(b)* the time required to produce optimal solutions *wrt* CPLEX, an off-the-shelf,

[4] Note that $\mathcal{H}(x)$ for $x \geq 0.5$ coincides with the expression for fuzzy entropy [Luca and Termini, 1974].

widely used, linear programming solver; and *(c)* the time required to yield optimal solutions as the number of agents and tasks grow. Overall, our results indicate that our algorithm significantly outperforms CPLEX, and hence it is the algorithm of choice to solve the Feasible Team Allocation for Tasks Problem introduced in this paper.

5.1 Empirical settings

For our experimental evaluation we used synthetic data generated in the following way. For each task we select the required team size $m_t \sim \mathcal{U}\{1,3\}$; the number of required competences $|C_t| \sim \mathcal{U}\{2,5\}$; and the weigh function is $w_t(c) = \mathcal{N}\big(\mu = \mathcal{U}(0,1), \sigma = \mathcal{U}(0.01,0.1)\big)$ bounded in $(0,1]$ for all $c \in C_t$. Then we generate m_t agents for each task t such that the acquired competencies of each agent contain competencies that are *(i)* identical or *(ii)* a child-node[5] of some required competence in t. With these generators we built 60 problem instances, distributed in 3 families of datasets: 20 instances with 10 tasks and ~ 20.5 agents; 20 instances with 15 tasks and ~ 30.6 agents; and 20 instances with 20 tasks and ~ 41.35 agents. The experiments were performed on a PC with Intel Core i7 (8th Gen) CPU, 8 cores, and 8Gib RAM. For all implementations we used Python3.7.

5.2 Results

Quality analysis. Using the optimal solutions yielded by CPLEX as a baseline, we can evaluate the quality of the solutions computed by the Edu2Com algorithm. For all problem instances, Edu2Com reaches the optimal solution. More precisely, for every problem instance, Edu2Com achieved a solution whose value, in terms of competence proximity, is equivalent to the optimal solution computed by CPLEX. Fig 1 shows the average *quality ratio* of Edu2Com *wrt* CPLEX along time. We calculate the quality ratio by dividing the competence proximity computed by Edu2Com by the optimal value computed by CPLEX, and it is depicted as a percentage.
Runtime analysis. The greatest advantage of Edu2Com is that it is way much faster than CPLEX. As shown in Fig 2 Edu2Com reaches optimality in less than half of the time required by CPLEX; while in the large setting of 20 projects our algorithm can be up to ~ 4 times faster than CPLEX. In Table 1 we show the percentage of time required by Edu2Com compared to CPLEX. Here we should note that the time consuming task for CPLEX is building of the LP encoding of the problem, while solving the actual problem is done much faster. This indicates that the problem instances at hand are rather large than hard: as the number of tasks increases, so does the number of agents, resulting in larger linear programs.
Anytime analysis. Last but not least we present our results on the anytime behaviour of Edu2Com (see Fig 1). We observe that after completing the initial stage, the solution quality produced by our algorithm reaches 80%, 70%, and 65% of the optimal solution, for problem instances with 10,15 and 20 tasks respectively. Furthermore, Edu2Com reaches 80% quality in $0.001 \times t_{CPLEX}$ for 10 programs, 70% in $0.025 \times t_{CPLEX}$ for 15 programs, and 65% in $0.0002 \times t_{CPLEX}$ for 20 programs, where t_{CPLEX} is the time CPLEX needs to build the input and compute the optimal solution. Finally we reach a quality of 80% of the optimum at 0.1%,20% and 13.5% of the time required by the base-line algorithm (CPLEX) to reach optimality.

[5]The competencies are structured in a tree graph.

dataset family	$M = 10$	$M = 15$	$M = 20$
time percentage (%)	40%	45%	29%

Table 1: Percentage of required time of edu2com compared to CPLEX

6 Conclusions and future work

In this paper, we formally defined the *Feasible Team Allocation for Tasks Problem* (FTAP), and studied its complexity. Then, we provided an encoding to optimally solve FTAP by means of linear programming. Thereafter, we proposed a heuristic anytime algorithm, *Edu2Com*, and conducted a systematic comparison of our approach versus the CPLEX LP solver. Our experimental evaluation showed that edu2com outperforms CPLEX in time, mainly because of the extremely large input that the latter requires. As future work, we intend to device more intelligent strategies during the second stage of edu2com, instead of our current randomised strategy. Furthermore, we will study the performance of edu2com on actual-world data.

References

[Andrejczuk *et al.*, 2019] E. Andrejczuk, F. Bistaffa, C. Blum, J. A. Rodríguez-Aguilar, and C. Sierra. Synergistic team composition: A computational approach to foster diversity in teams. *Knowledge-Based Systems*, 182(104799), 10/2019 2019.

[Andrejczuk, 2018] E. Andrejczuk. Artificial intelligence methods to support people management in organisations. Doctoral, 05/2018 2018.

[GLPK, 2018] GLPK. Glpk: Gnu linear programming kit). https://www.gnu.org/software/glpk/, 2018.

[GUROBI, 2018] GUROBI. Gurobi optimizer 8.0. https://www.gurobi.com/, 2018.

[IBM, 2019] IBM. Ibm ilog cplex optimization studio 12.10. https://www.ibm.com/us-en/marketplace/ibm-ilog-cplex, 2019.

[Li *et al.*, 2003] Y. Li, Z. A. Bandar, and D. Mclean. An approach for measuring semantic similarity between words using multiple information sources. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):871–882, 2003.

[Luca and Termini, 1974] Aldo De Luca and Settimo Termini. Entropy of l-fuzzy sets. *Information and Control*, 24(1):55 – 73, 1974.

[Sandholm *et al.*, 2002] T. Sandholm, S. Suri, A. Gilpin, and D. Levine. Winner determination in combinatorial auction generalizations. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 1*, AAMAS '02, page 69–76, 2002.

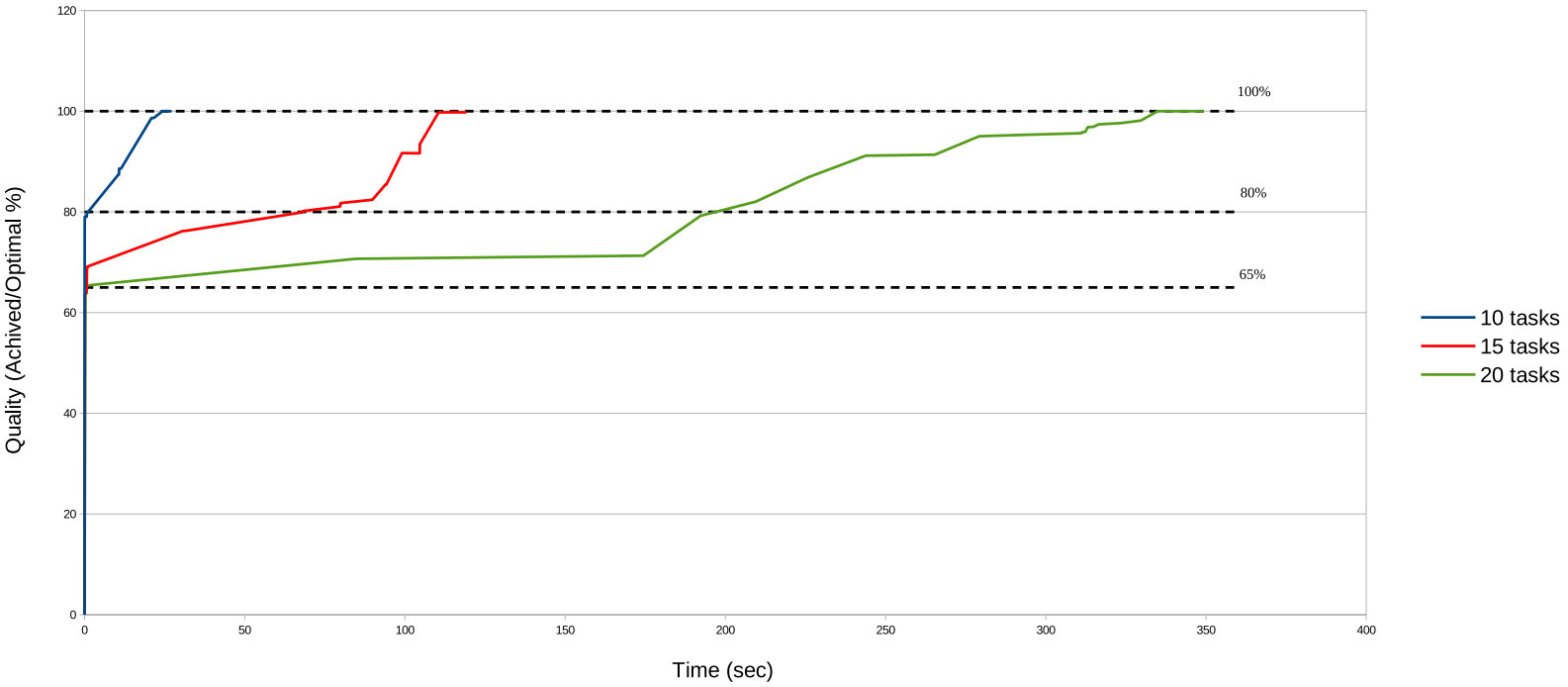

Figure 1: Solution Quality: achieved competence proximity by Edu2Com / optimal competence proximity %. The values are averages over 20 problem instances per family of datasets.

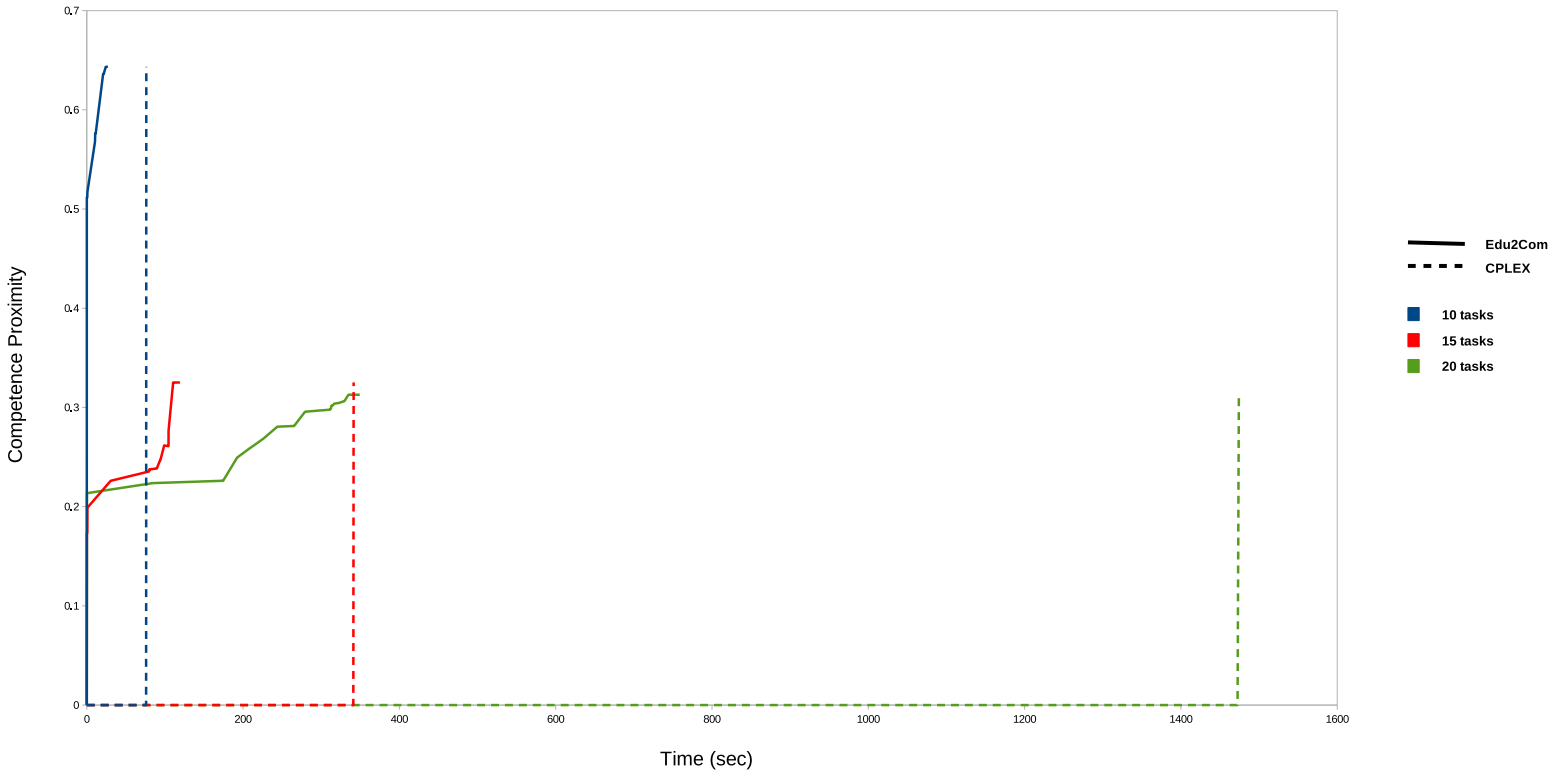

Figure 2: Average Competence Proximity vs Time