



**PROJECT: COMRADE – Combinatorial Reverse Auctions: Design of
Efficient Mechanisms for Crowdsourcing Markets**

Funded under the 1st Call for HFRI Research Projects to Support Faculty Members and
Researchers and Procure High-Value Research Equipment

WP6: Design and analysis of mechanisms for combinatorial reverse
auctions

Deliverable D.6.2: Technical report on the evaluation and final selection of mechanisms

Prepared by the team members:

**Michail Fasoulakis, Evangelos Markakis, Thanassis Papaioannou, Georgios Papasotiropoulos,
Panagiotis Tsamopoulos**



1 Introduction

This is the second deliverable of WP6. It was completed by Month 43, in accordance to the amended timetable of COMRADE. The goal of the deliverable is to report on the experimental evaluation that we performed on our findings and on comparisons with selected mechanisms from the existing literature. In Deliverable D.6.1, we provided an overview of the experimental setup and of the generation of the data sets that used. This deliverable is focusing on the algorithms that we implemented and on their performance as resulting from the experiments.

We briefly outline the structure of this deliverable. The two major algorithmic problems for which we performed the experimental evaluation are described in Section 2, and have also been described in previous reports, related to WP4, such as Deliverable D.4.1. The former of these problems is a social welfare maximization problem under a given submodular valuation function for the auctioneer, whereas the latter one is a cost minimization problem arising in the context of spatial crowdsourcing. In Section 3, we describe the algorithms and the experimental results for the problem of social welfare maximization. In analogy to this, in Section 4, we do the same for the auctions that concern spatial crowdsourcing markets.

2 Basic Mathematical Models and Algorithmic Objectives

For the sake of completeness, we recall the general setup and the algorithmic questions we have been interested in. These have also been described in the deliverable D.4.1, which was the first deliverable of WP4.

We consider a procurement auction setting, and let $A = [n] = \{1, 2, \dots, n\}$ denote a set of n agents who are willing to participate in the reverse auction. The auctioneer is interested in having some predefined set of tasks completed, possibly subject to certain constraints, and is therefore aiming at hiring some of the participating agents. There are various possibilities for modeling the capabilities of each agent, the requirements of the auctioneer, and the bidding interface so that the agents can make offers on the tasks they can complete. E.g., the auctioneer may simply want to assign each task to a single agent, or assign tasks multiple times (for quality assurance). Each agent may be capable of performing a single task out of the tasks posted by the auctioneer, or she may be able to bid for performing a subset of the tasks. The interface may also allow agents to make multiple bids on different combinations of tasks, or to make bids only for single tasks. The choices made for these aspects can lead to a variety of different models for task assignment.

Based on the current state of the art, and on our Deliverable D.2.1 from WP2, which performed a literature survey, we decided to work on the following types of optimization problems that capture different aspects of task assignment in reverse auctions. We describe them separately below.

2.1 Social welfare maximization

We consider a generalization of a class of models that were originally introduced by Singer [9]. In these problems, the auctioneer asks each agent i to submit one parameter, denoted by c_i to the mechanism, corresponding to the cost for executing the tasks she is capable of. This assumption can capture several different scenarios. E.g., for an agent i , her cost c_i could simply be the cost incurred to her for executing a single task, among the

tasks advertised by the auctioneer. In other cases, it can denote the cost for executing a collection of tasks that she is capable of doing (and in which case the agent should declare along with c_i the list of these tasks). Regardless of the interpretation of the cost c_i , all these settings can be treated in a unified manner.

Furthermore, to evaluate the selected outcome, the auctioneer is also associated with a valuation function $v : P(A) \rightarrow \mathbb{R}$, where $P(A)$ is the powerset of the set of agents, A . Hence, $v(S)$ denotes the value (which can be thought of as the level of profit or happiness) derived by the auctioneer if she selects the set S of agents as the auction winners. It is also assumed that when a set S of agents is selected as the winners, these agents are committed to executing the tasks corresponding to their offer (this could be enforced by some penalty mechanism and it is beyond the scope of our work).

To summarize, the auctioneer collects the offers of the agents, and then needs to select a set $S \subseteq A$, as the winning agents, along with a payment to each winner. For this deliverable, we are interested only in the non-strategic version of these problems, where the agents are honest about their cost c_i , and therefore, we only care about the question of determining the allocation of the tasks, i.e., the set of winning agents. Enforcing the agents to be honest about their costs is a topic that has been addressed in WP5 and will therefore not be considered here.

Given the progress that has already been made in the last years for the model that was originally defined by Singer in [9], we define here a somewhat complementary direction to the previous subsection, in which there is no hard budget constraint for the auctioneer (as opposed to [9]). This is very common in several applications where the auctioneer may not have a priori a hard limit on his expenditure. Instead, it is more appropriate to focus on other objectives, which incorporate either the true costs of the agents or the sum of payments by the auctioneer into the objective function. This naturally leads to concentrate on the maximization of the produced utilitarian social welfare, which equals the auctioneer's value minus the total costs of the agents for carrying out the tasks.

To describe more formally the model, we want to optimize the *social welfare* achieved by S , denoted by $SW(S)$, which is the total value derived minus the total cost incurred, expressed as follows:

$$SW(S) = v(S) - \sum_{i \in S} c_i$$

2.2 Cost minimization in spatial crowdsourcing models

The last problem is motivated by spatial models in reverse auctions. There are three main differences compared to the previous. The first one is that we assume that the tasks lie in some geometric space, i.e., they may correspond to transportation tasks within a Euclidean space. The simplest example is to suppose that the tasks are ordered on a line. The second difference is that we now want to have all the tasks completed. This is unlike Sections ?? and 2.1, where not all tasks are necessarily assigned, either due to the budget restrictions or due to the maximization of social welfare. The third difference is that each task has a demand, on the quality of its completion, and this implies that it may have to be assigned to multiple agents, to achieve its demand constraint. In applications where it is important to complete all the tasks listed by the auctioneer, it is more natural to focus more on producing a feasible solution of minimum cost. This gives rise to a cost minimization problem that we describe below.

Cost Minimization Interval Cover (CMIC): Consider a set of tasks, say $\{1, \dots, m\}$, that are ordered on a line, and a set of available bidders $A = \{1, \dots, n\}$. An instance of

CMIC is determined by a tuple $(\mathbf{b}, \mathbf{q}, \mathbf{d})$, where:

- The vector $\mathbf{b} = (b_1, \dots, b_n)$ is the bidding profile. For each bidder $j \in A$, $b_j = (c_j, [s_j, f_j])$, where $[s_j, f_j] = I_j$ is the interval of contiguous tasks $\{s_j, s_j+1, \dots, f_j\} \subseteq [m]$, that j is able to contribute to, and $c_j \in \mathbb{R}_{\geq 0}$ is the cost incurred to her, if she is selected to contribute. We often denote $\mathbf{b}$ as $(\mathbf{c}, \mathbf{I})$ where $\mathbf{c}$ is the cost vector and $\mathbf{I}$ is the vector of the intervals.
- The vector $\mathbf{q} = (q_1, \dots, q_n)$ is the contribution vector, so that $q_j \in \mathbb{R}_{>0}$ denotes the contribution that bidder j can make to the tasks that belong to $[s_j, f_j]$.
- The vector $\mathbf{d} = (d_1, \dots, d_m)$ specifies the demand $d_j \in \mathbb{R}_{>0}$ of a task j .

The goal is to select a set of bidders $S \subseteq A$ of minimum cost, that satisfies

$$\sum_{i \in \mathcal{N}_j(\mathbf{I}) \cap S} q_i \geq d_j, \quad \forall j \in [m], \quad (1)$$

where for $j \in [m]$, $\mathcal{N}_j(\mathbf{I}) := \{i \in A \mid j \in [s_i, f_i]\}$, i.e., it is the set of bidders who can contribute to task j .

3 Algorithms and Results for Social Welfare Maximization

For the problem of social welfare maximization, we outline first the algorithms that we selected for the experimental evaluation.

3.1 Greedy Algorithms

An algorithm is called greedy if it builds up a solution in small steps, making a decision in every step based on a specific criterion. Thus, each step chooses the best case based on that criterion. The most basic greedy algorithm that exists is the algorithm that starts with an empty set and keeps adding the agent with the maximum marginal gain if this gain is positive. More refined criteria are presented in the algorithms below.

3.1.1 Double Greedy

The Double Greedy algorithm was proposed by Buchbinder et al. [2]. The algorithm starts with an arbitrary order $w_1, w_2, \dots, w_m$ of the agents and iterates m times. The algorithm, throughout the execution, keeps two possible solutions X, Y . In the beginning, the first one is empty, i.e., $X = \emptyset$, and the other one is equal to the ground set of all workers, $Y = A$.

In the i -th iteration the algorithm either adds w_i to the solution X_{i-1} or removes it from Y_{i-1} . The decision is based on the marginal gain of these two options. Thus, if the marginal gain of adding the element to X is greater than removing the element from Y , then the element is added to X . Eventually, after m iterations the two solutions coincide and this is what we return as an output. It is important to note that this algorithm has a $1/3$ -approximation guarantee but only for non-negative submodular functions.

Algorithm 1 Double Greedy

Input: valuation function v , cost vector c , set of agents A .

Output: set $X_m \subseteq A$

Let $g(S) = SW(S)$ be the social welfare objective

$X_0 \leftarrow \emptyset, Y_0 \leftarrow A$

for $i \leftarrow 1$ **to** m **do**

$a_i \leftarrow g(e_i | X_{i-1})$

$b_i \leftarrow g(e_i \notin Y_{i-1})$

if $a_i \geq b_i$ **then**

$X_i \leftarrow X_{i-1} \cup e_i$

$Y_i \leftarrow Y_{i-1}$

else

$Y_i \leftarrow Y_{i-1} \setminus e_i$

$X_i \leftarrow X_{i-1}$

return X_m

3.1.2 Distorted Greedy

The Distorted Greedy algorithm was proposed by Harshaw et al. [4]. The algorithm is fairly simple, starting with an empty set $S = \emptyset$ of workers. At the i -th step, we attempt to add into the solution the worker e_i from the remaining workers) that maximizes the marginal gain based on a specific criterion. If this marginal gain is greater than zero, the worker is added to the solution. The criterion of choice is based on a function $\Phi_i(S_i, e) \triangleq (1 - \frac{1}{k})^{k-(i+1)}v(e|S_i) - c_e$ that is essentially a modified function of the most typical marginal profit.

Algorithm 2 Distorted Greedy

Input: valuation function v , cost vector c , cardinality k , set of agents A .

Output: set $S \subseteq A$

$S_0 \leftarrow \emptyset$

for $i \leftarrow 0$ **to** $k - 1$ **do**

$e_i \leftarrow \arg \max_{e \in A \setminus S_{i-1}} \Phi_i(S_i, e)$

if $\Phi_i(S_i, e) > 0$ **then**

$S_{i+1} \leftarrow S_i \cup e_i$

return S_k

$\triangleright k = m$ for the SWM problem
 $\triangleright \Phi_i(S_i, e) \triangleq (1 - \frac{1}{k})^{k-(i+1)}v(e|S_i) - c_e$

3.1.3 ROI Greedy

The ROI Greedy algorithm was proposed by Jin et al. [5]. This algorithm is very similar to the distorted greedy algorithm. The core difference is that the ROI greedy algorithm in the i -th step chooses the worker e_i that maximizes the modified marginal gain function $\frac{f(e|S_i)}{c_e}$, based on the ratio of the marginal gain divided by the cost.

Algorithm 3 ROI Greedy

Input: valuation function v , cost vector c , cardinality k , set of agents A . $\triangleright |A| = m$

Output: set $S \subseteq A$

$S_0 \leftarrow \emptyset$

for $i \leftarrow 0$ **to** $m - 1$ **do**

$e_i \leftarrow \arg \max_{e \in A \setminus S_{i-1}} \frac{v(e|S_i)}{c_e}$

if $v(e|S_i) - c_e > 0$ **then**

$S_{i+1} \leftarrow S_i \cup e_i$

else

return S_i

return S_m

3.1.4 Cost Scaled Greedy (CSG)

The CSG algorithm was proposed by Nikolakaki et al. [7]. This algorithm is another example of a greedy algorithm, but this time instead of modifying the marginal gain, the objective function is scaled instead. The algorithm works like the previous algorithms, with the main difference being that the scaled objective function $\tilde{g}(S) = v(S) - 2c(S)$, is used to calculate the marginal gain when choosing a worker to be added to the solution. Finally, the worker is added if and only if the gain of the scaled function is strictly positive. Notice that this algorithm tends to reject very costly workers (concerning the valuation).

Algorithm 4 CSG

Input: valuation function v , cost vector c , cardinality k , set of agents A .

Output: set $S \subseteq A$

Let $\tilde{g}(S) = v(S) - 2c(S)$ be the scaled objective

$S_0 \leftarrow \emptyset$

for $i \leftarrow 0$ **to** $k - 1$ **do**

$e_i \leftarrow \arg \max_{e \in A \setminus S_{i-1}} \tilde{g}(e|S_i)$

if $\tilde{g}(e|S_i) \leq 0$ **then**

return S_i

$S_{i+1} \leftarrow S_i \cup e_i$

return S_k

$\triangleright k = m$ for the SWM problem

3.2 An LP-based algorithm

For a restricted (but still challenging) class of submodular valuations, a different algorithm has been proposed based on Linear Programming. In particular, this concerns the class of submodular coverage valuations, studied in [8]. Regarding the workers' capabilities, it is assumed that each worker i can perform a set of tasks Γ_i at a total cost of c_i . The maximization of social welfare in this case is equivalent to solving the following integer program.

$$\begin{aligned} \max \quad & \sum_{j \in M} v_j x_j - \sum_{i \in A} c_i y_i \\ \text{s. t.} \quad & x_j \leq \sum_{i: \tau_j \in \Gamma_i} y_i & \forall j \in M \\ & x_j \in \{0, 1\} & \forall j \in M \\ & y_i \in \{0, 1\} & \forall i \in A \end{aligned}$$

In the above formulation, v_j is the value derived by the auctioneer when the task j is completed, for every task $j \in M$. Given that the integer program cannot be solved in polynomial time, [8] solves instead the LP relaxation of the program and then uses the fractional solution to propose a greedy algorithm. Namely, the workers are sorted according to the fractional value y_i from the optimal LP solution, since the larger their value, the more likely it is to increase the profit of the platform, and hence the more contribution that user can provide. Then each worker in turn is selected as long as she has a positive marginal effect on the current social welfare. The pseudocode follows.

Algorithm 5 LP-BASED [8]

Input: valuation function v , set of agents A , cost vector and task set for all agents, $\{\Gamma_i, c_i\}_{i \in A}$.
Initialize $W = \emptyset$
Solve LP relaxation of the problem to obtain optimal x^*, y^* solutions
Sort workers in non increasing order of their value in y^* and form their index list
for $w \in A$ *according to the index list* **do**
 Select the corresponding worker w
 if $v(W \cup \{w\}) - v(W) - c_w > 0$ **then**
 Add w to W
return W

3.3 Results of the experimental evaluation

In order to evaluate the presented algorithms we used the following metrics:

- Average ratio $mean\left(\frac{SW(S)}{SW(OPT)}\right)$: The mean of the ratios $\frac{SW(S)}{SW(OPT)}$ of all the instances.
- Worst ratio $min\left(\frac{SW(S)}{SW(OPT)}\right)$: The minimum ratio $\frac{SW(S)}{SW(OPT)}$ of all the instances.

In the subsequent analysis, we delve into the results obtained from various metrics, examining the unconstrained version of the algorithms. The initial focus is on the average ratio, where, for each pair of tasks and workers (m, n) , we calculate the average social welfare ratio $\frac{SW(S)}{SW(OPT)}$. For the synthetic datasets, this information is then plotted in a three-dimensional space, with the number of tasks represented on the x -axis, the number of workers on the y -axis, and the ratio on the vertical z -axis. We use bars for the ratio, and also to help visualize, we are using the dotted lines that follow the average ratio. For the datasets that are based in real data, the results are plotted in a two dimensional figure, where the x -axis represents the size of the instance and the y -axis the approximation ratio.

Starting with the average ratio, the figures 1, 2 present graphs for all synthetic datasets per algorithm. The use of bars visually represents the ratio, accompanied by dotted lines delineating the average ratio for enhanced visualization.

Observations from the figures reveal that all algorithms consistently exhibit a high average ratio, surpassing 85%. Notably, even the Double Greedy algorithm, designed for non-negative submodular functions, performs commendably, showcasing its adaptability to a broader context. However, it's crucial to recognize that on smaller datasets ($m=5, 10$ and $m=5, 10, 20$), the CSG algorithm's performance is suboptimal due to a substantial penalty imposed by the cost function. The criterion of CSG, based on the marginal gain

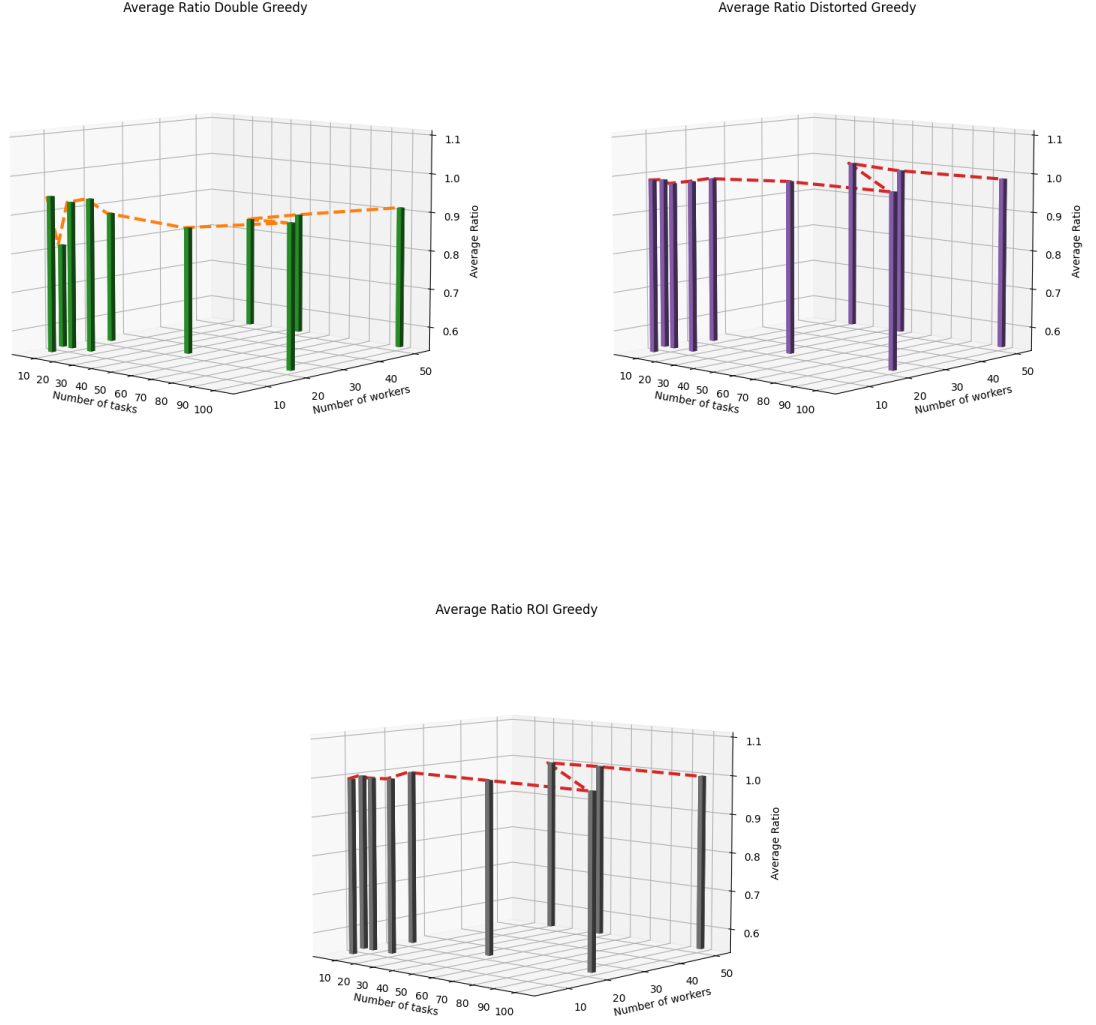


Figure 1: Average Ratio per Algorithm, Synthetic Data 1/2

of the modified objective function with doubled costs, results in insufficient candidates for the solution in smaller sets, leading to algorithmic failure.

Conversely, other greedy algorithms perform exceptionally well, aligning with expectations given their specialized design for this problem. Notably, the LP-Based algorithm emerges as the top performer, consistent with its construction specifically tailored to address this problem.

In summary, the analysis of the average ratio unveils the robust performance of the algorithms, with the LP-Based algorithm standing out as the most effective. The results provide valuable insights into algorithmic behavior across varying dataset sizes and configurations, contributing to a comprehensive understanding of their efficacy in solving the optimization problem at hand.

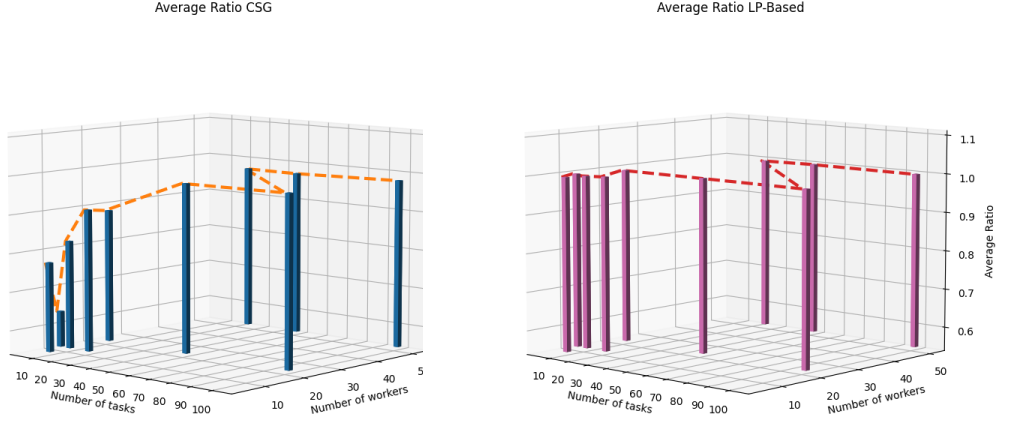


Figure 2: Average Ratio per Algorithm, Synthetic Data 2/2

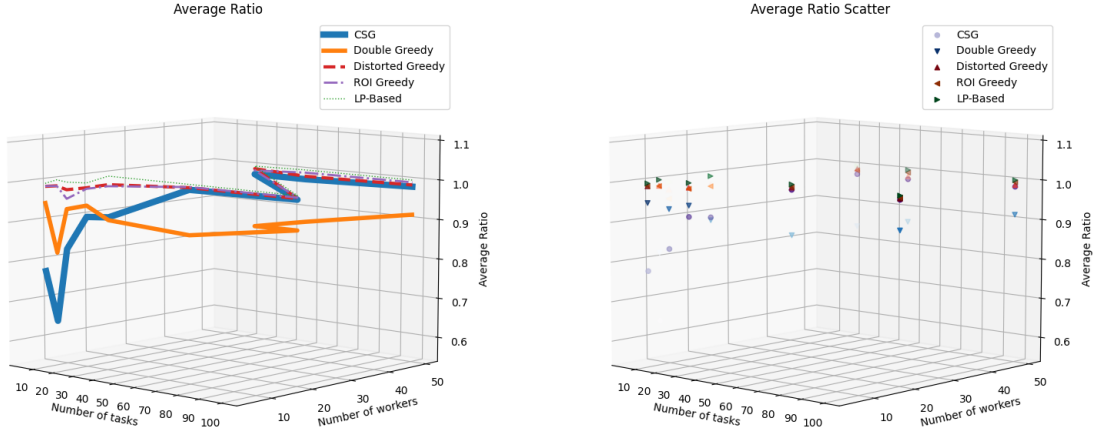


Figure 3: Average Ratio All Algorithms, Synthetic Data

In figure 3 the average ratio for the five algorithms is visualized using the synthetic dataset in one comprehensive plot. Plot lines are employed to represent results, with each line corresponding to a specific algorithm. Additionally, a scatter plot of the same data is presented within the same figure. Notably, some smaller sets exhibit a slight dip in the Double Greedy and CSG algorithms. This phenomenon is attributed to the narrow gap between the valuation function f and the cost function c , resulting in highly negative function values. In such cases, the optimal solution may involve selecting no workers at all. While less pronounced in the greedy algorithms, a general trend emerges, showcasing improved performance on larger sets. Despite low worst-case guarantees, these algorithms consistently perform well on average and in the context of larger datasets.

Furthermore, in figures 4,5, we extend our analysis to encompass the performance of all the algorithms for the datasets that are based on real data from Guru and Freelancer. Notably, the observations and patterns discerned from the synthetic data experiments persist in the experiments conducted with real data. The consistent performance across both synthetic and real datasets underscores the robustness and applicability of the algorithms in real-world scenarios.

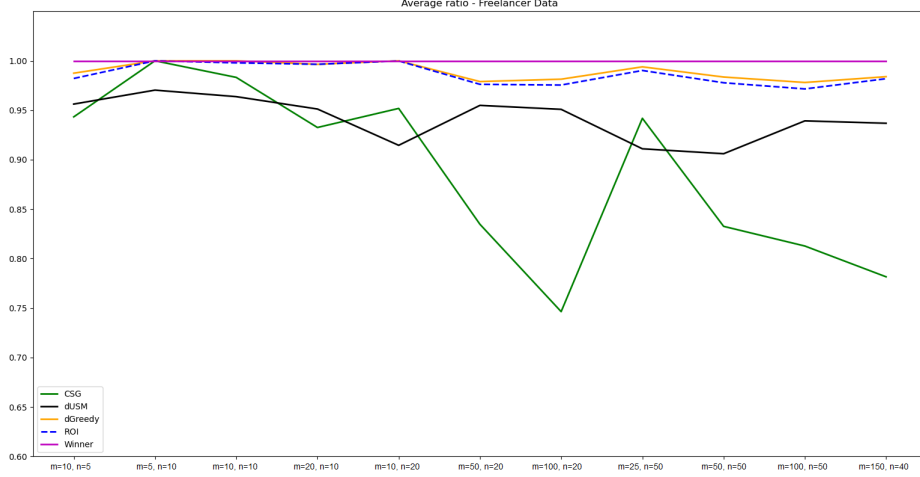


Figure 4: Average Ratio all Algorithms, Data from Freelancer

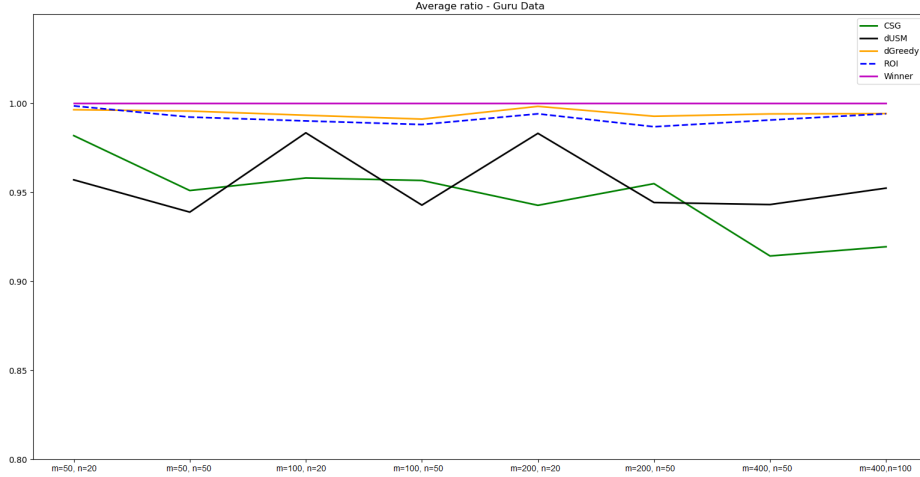


Figure 5: Average Ratio all Algorithms, Data from Guru

In this experimental setup, both synthetic and real datasets demonstrate effective approximation. Although the experiments primarily employed standard submodular functions with normal distributions for valuations and costs, the robust performance suggests that, in practical scenarios, complex functions and rules may not be necessary.

Now, turning attention to the minimum ratio metric, the worst instance for each

dataset is identified, and the social welfare ratio $\frac{SW(S)}{SW(OPT)}$, is displayed. This metric allows us to observe the worst-case performance of each algorithm. For synthetic data, bar charts and dashed lines aid in visualizing the results, with the x -axis denoting the number of workers, the y -axis representing the number of workers, and the vertical z -axis indicating the minimum ratio. For datasets generated from real data, again the results are plotted in a two dimensional figure, where x -axis represents the size of the instance and the y -axis the approximation ratio.

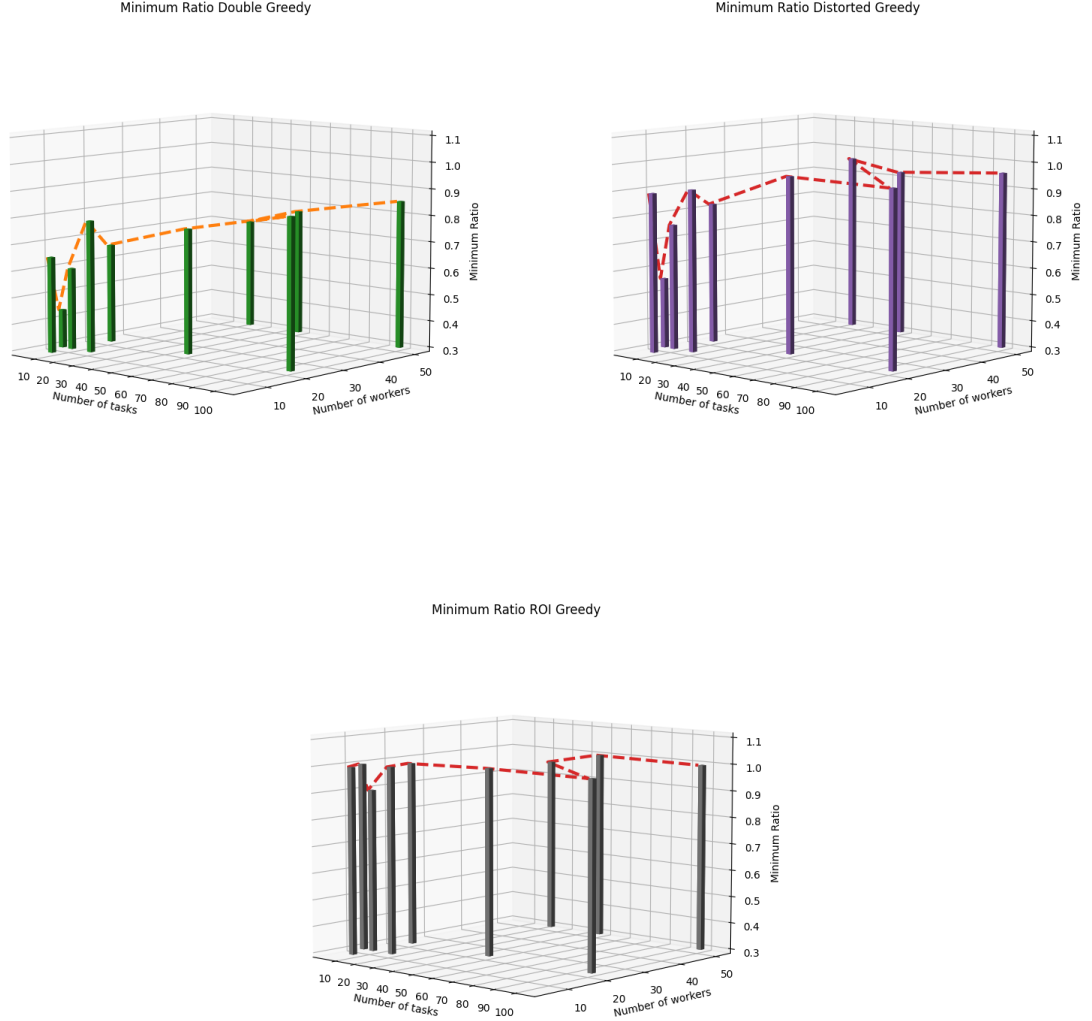


Figure 6: Minimum Ratio per Algorithm, Synthetic Data 1/2

Figures 6 and 7, depict the results for the synthetic datasets. The minimum ratio exhibits a similar trend to the average ratio, with most algorithms consistently surpassing 50%. Notably, only in extremely negative datasets do the Double Greedy and CSG algorithms exhibit suboptimal performance. It is crucial to highlight an observation regarding the CSG algorithm, where on small datasets, instances were generated with a solution of picking 0 workers, despite the optimal number of workers being positive. This

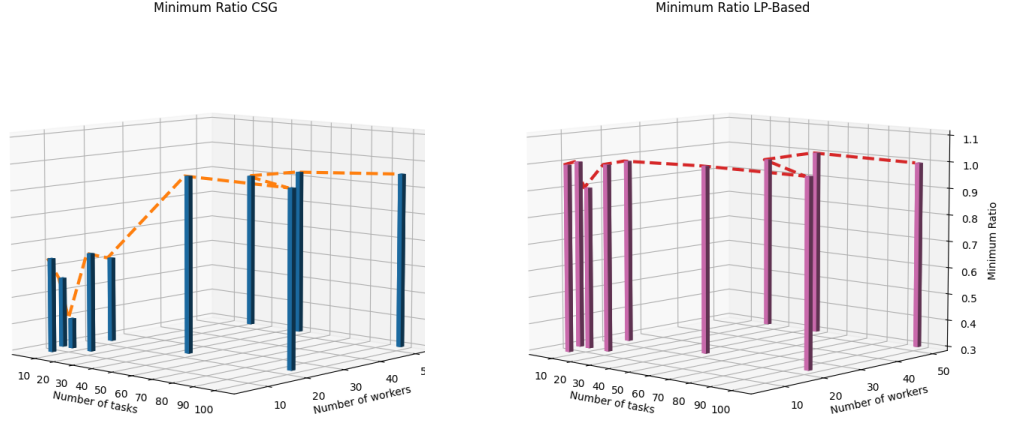


Figure 7: Minimum Ratio per Algorithm, Synthetic Data 2/2

behavior aligns with the worst-case guarantee for CSG, where the algorithm may choose no workers if the cost is exceptionally high, maintaining the worst-case guarantee validity. Instances with a negative value for $\frac{f(OPT)}{2} - c(OPT)$ led to the algorithm's decision to choose no workers, and therefore the SW was $0 > \frac{f(OPT)}{2} - c(OPT)$.

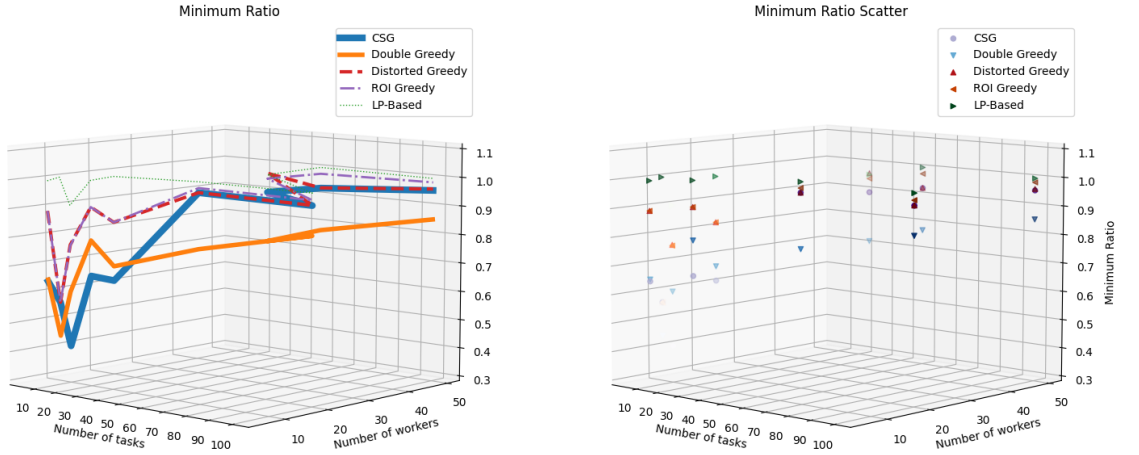


Figure 8: Minimum Ratio All Algorithms, Synthetic Data

Furthermore, Figure 8 consolidates graphs for all algorithms, featuring line plots and scatter plots. The observed behavior aligns with the average ratio, further emphasizing that these algorithms tend to perform better on larger sets, and also featuring the LP-based algorithm as a top performer.

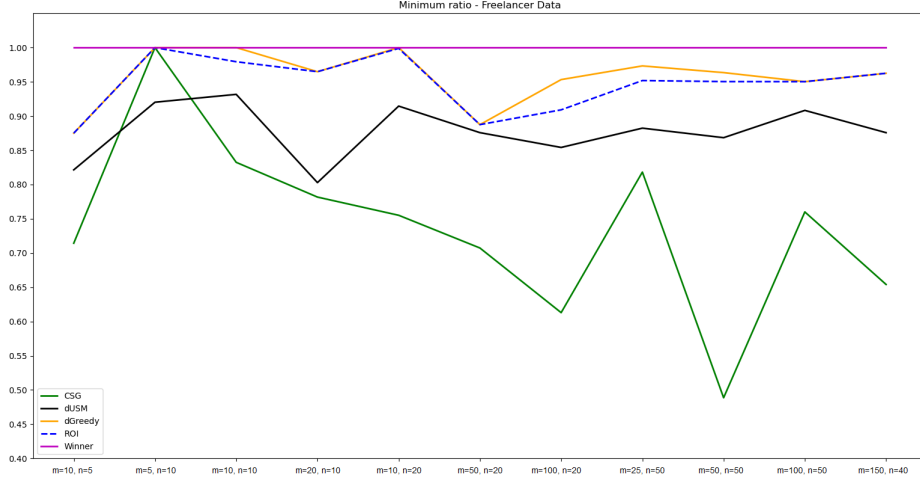


Figure 9: Minimum Ratio all Algorithms, Data from Freelancer

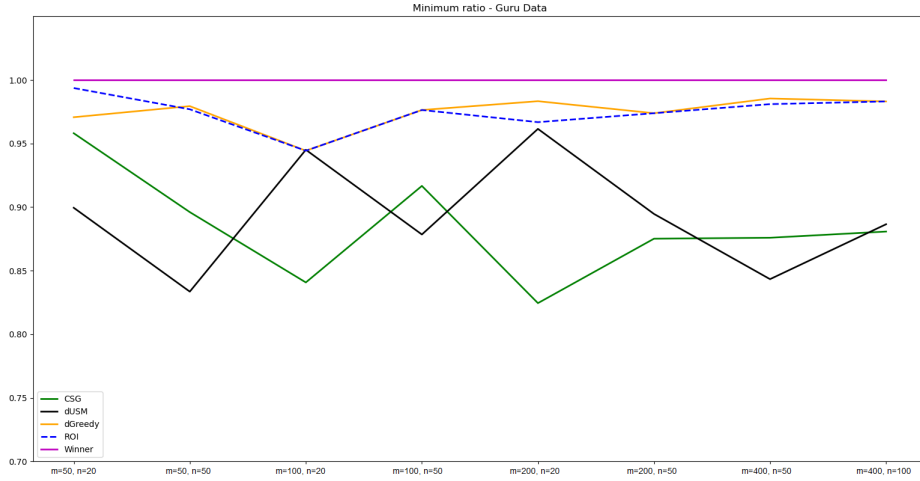


Figure 10: Minimum Ratio all Algorithms, Data from Guru

Lastly, in figures 9 and 10, we can verify that the observations on the performance of the algorithms for the synthetic datasets, carry on also for the datasets that are generated using real data.

4 Algorithms and Results for Spatial Crowdsourcing

In this section, we move to the spatial crowdsourcing model, as already described in Deliverable D.6.1 and we discuss certain algorithms for CMIC as well as the results of our implementations.

4.1 Algorithms for CMIC

We evaluated 2 algorithms in the context of spatial crowdsourcing towards solving CMIC. The first one is the algorithm that currently has the best provable worst-case approximation ratio of 4 for CMIC, following from [1]. However, as we have mentioned in the deliverables of WP5, this algorithm does not take into account strategic aspects and gives incentives to agents to misreport their preferences. This led us to propose a second algorithm, which is a modification of the algorithm by [1]. It is a greedy algorithm that employs the local-ratio technique and achieves a worse approximation guarantee but satisfies the strategyproofness criterion, i.e., it does not give incentives to workers to misreport their preferences.

At what follows, we present both algorithms in a unified way. Initially, we introduce an algorithmic scheme for CMIC, laying the foundation for the subsequent introduction of LR-CMIC. This will then undergo slight modifications to give rise to the definition of LR-DEL-CMIC. The main component of the approach is an approximation-preserving reduction to a particular job scheduling problem for a single machine, defined below. It also reveals an interesting connection between our problem and a family of job scheduling problems.

Loss Minimization Interval Scheduling (LMIS): We are given a limited resource whose amount may vary over a time period, which WLOG, is defined by the integral time instants $\{1, \dots, m\}$. We are also given a set of activities $J = \{1, \dots, n\}$, each of which requires the utilization of the resource, for an interval of time instants. An instance of LMIS is determined by a tuple $(\mathbf{p}, \mathbf{T}, \mathbf{r}, \mathbf{D})$, where:

- The vector $\mathbf{p} = (p_1, \dots, p_n)$ specifies a penalty $p_j \in \mathbb{R}_{>0}$, for each activity $j \in J$, reflecting the cost that is incurred by not scheduling the activity.
- For each activity $j \in J$, we are given an interval¹ $T_j = [s_j, f_j]$, such that $s_j, f_j \in \{1, \dots, m\}$ are the start and finish times of j respectively. Let $\mathbf{T} = (T_1, \dots, T_n)$, be the vector of all activity intervals.
- The vector $\mathbf{r}$ contains, for each activity $j \in J$, the width $r_j \in \mathbb{R}_{>0}$, reflecting how much resource the activity requires, i.e., this means that activity j requires r_j units of resource at every integral time instant of its interval T_j .
- The vector $\mathbf{D} = (D_1, \dots, D_m)$ specifies the amount of available resource $D_i \in \mathbb{R}_{>0}$, at each integral time instant $i \in [m]$.

Let $J_i(\mathbf{T}) := \{j \in J \mid i \in T_j\}$. The goal in LMIS is to select a set of activities $S \subseteq J$ to schedule, that meet the resource constraint

$$\sum_{j \in S \cap J_i(\mathbf{T})} r_j \leq D_i, \quad i = 1, \dots, m, \quad (2)$$

and such that $\sum_{j \in J \setminus S} p_j$ is minimized, i.e., we want to minimize the sum of the penalties for the non-scheduled activities.

Our work highlights an interesting connection between LMIS and CMIC. This can be seen via the reduction provided by algorithm $\hat{A}$ below, where for an instance $((\mathbf{c}, \mathbf{I}), \mathbf{q}, \mathbf{d})$, we let $\mathbf{Q}(\mathbf{I}) = (Q_1(\mathbf{I}), \dots, Q_m(\mathbf{I}))$ and $Q_j(\mathbf{I}) := \sum_{i \in N_j(\mathbf{I})} q_i, \forall j \in [m]$.

¹Originally, the problem was defined using a semi-closed interval for each activity, but it is easy to see that defining it using a closed one instead, is equivalent and more convenient for our purposes.

Algorithm 6 $\hat{A}(\mathbf{b})$

▷ **Input:** A bidding profile $\mathbf{b} = (\mathbf{c}, \mathbf{I})$ of a CMIC instance $((\mathbf{c}, \mathbf{I}), \mathbf{q}, \mathbf{d})$

Construct the LMIS instance $(\mathbf{p}, \mathbf{T}, \mathbf{r}, \mathbf{D}) = (\mathbf{c}, \mathbf{I}, \mathbf{q}, \mathbf{Q}(\mathbf{I}) - \mathbf{d})$, with $J = N$.

Run an approximation algorithm for $(\mathbf{p}, \mathbf{T}, \mathbf{r}, \mathbf{D})$, and let S be the scheduled activities.

return $N \setminus S$

Theorem 4.1. *Algorithm $\hat{A}$ converts any α -approximation algorithm for LMIS to an α -approximation algorithm for CMIC.*

Theorem 4.1 is based on the lemma below, which shows the connection between the feasible solutions of the two problems.

Lemma 4.2. *Consider a CMIC instance $P = ((\mathbf{c}, \mathbf{I}), \mathbf{q}, \mathbf{d})$. Let also P' be the LMIS instance defined by $(\mathbf{p}, \mathbf{T}, \mathbf{r}, \mathbf{D}) = (\mathbf{c}, \mathbf{I}, \mathbf{q}, \mathbf{Q}(\mathbf{I}) - \mathbf{d})$, with $J = N$. Then, for every feasible solution S of P , it holds that $J \setminus S$ is a feasible solution for P' with the same cost, and vice versa.*

4.1.1 Algorithm LR-CMIC

We will now consider an algorithm for LMIS, which is a tailor made modification of the algorithm by [1] for our purposes. This is presented as Algorithm 7 below. We can then plug in this algorithm as the component of $\hat{A}$ for solving LMIS. This combination of procedures yields an algorithm, henceforth denoted as LR-CMIC due to the Local-Ratio interpretation that its primary component admits. Our analysis, detailed in [6], establishes that LR-CMIC achieves an approximation ratio of Δ , which is the maximum number of workers submitting a willingness to work on the same task. Interestingly, in [6], we also provided an instance in which LR-CMIC attains an approximation factor of exactly Δ , indicating that, without adjustments, the algorithm cannot ensure performance better than Δ for general instances. However, we subsequently demonstrate a slight modification to the algorithm that reduces the approximation factor to a constant, and, more precisely, from Δ to 4. We note that this adjustment comes at the expense of forfeiting the monotonicity guarantee, a critical consideration for those concerned with the strategic aspects of the problem under consideration. The algorithm that achieves the guarantee of 4 will be denoted by LR-DEL-CMIC and discussed in the subsection that follows.

We introduce a notion that will be useful both for the statement of Algorithm 7 and for our analysis. Consider an instance $(\mathbf{p}, \mathbf{T}, \mathbf{r}, \mathbf{D})$ of LMIS. Given a set of jobs $S \subseteq J$, and a time instant $i = 1, \dots, m$, we define

$$R_i(S, \mathbf{T}, \mathbf{D}) := \sum_{\ell \in S \cap J_i(\mathbf{T})} r_\ell - D_i.$$

The quantity $R_i(S, \mathbf{T}, \mathbf{D})$ measures how much (if at all), the resource constraint of Equation (2), for the i -th time instant is violated when scheduling all the activities in S . Accordingly, define $R^*(S, \mathbf{T}, \mathbf{D}) := \max_{i=1, \dots, m} R_i(S, \mathbf{T}, \mathbf{D})$. Note that a schedule S is feasible if and only if $R^*(S, \mathbf{T}, \mathbf{D}) \leq 0$.

Algorithm 7 constructs a feasible schedule S as follows: Initially, it checks if the entire set of activities $S = J$ constitutes a feasible schedule. If not, the algorithm iteratively removes one activity per iteration from S , in a greedy fashion, until S becomes feasible. The algorithm determines the time instant t^* with the most violated feasibility constraint, by computing $R^*(S, \mathbf{T}, \mathbf{D})$, and considers all activities from S whose interval contains t^* .

Then, it removes from S one of these activities that minimizes a certain ratio, dependent on the current penalties and resource requirements, while it simultaneously decreases the penalty of all other activities that contain t^* .

Algorithm 7 LMIS-LR($\mathbf{p}, \mathbf{T}, \mathbf{r}, \mathbf{D}$)

▷ **Input:** An instance $(\mathbf{p}, \mathbf{T}, \mathbf{r}, \mathbf{D})$ of LMIS

Initialize $S = J, k = 0$, and $\mathbf{p}_k = (p_{i,k})_{i \in [n]} = \mathbf{p}$.

while $R^*(S, \mathbf{T}, \mathbf{D}) > 0$ **do**

 Let $t^* \in [m]$ be a maximizer of $R^*(S, \mathbf{T}, \mathbf{D})$.

$S_k = S \cap_{t^*}(\mathbf{T})$

$\varepsilon_k = \min_{i \in S_k} \frac{p_{i,k}}{\min\{R^*(S, \mathbf{T}, \mathbf{D}), r_i\}}$

 For $i = 1, \dots, m$ let

$$p_{i,k+1} = \begin{cases} p_{i,k} - \varepsilon_k \min\{R^*(S, \mathbf{T}, \mathbf{D}), r_i\}, & \text{if } i \in S_k, \\ p_{i,k}, & \text{o/w.} \end{cases}$$

 Let $j^* \in S_k$ be a minimizer of ε_k .

 Set $S = S \setminus \{j^*\}$, and $k = k + 1$.

return S

4.1.2 Algorithm LR-DEL-CMIC

As also discussed in the previous subsection, despite the fact that LR-CMIC guarantees a factor of Δ , a variant of it, called LR-DEL-CMIC, can achieve the significantly improved approximation factor of 4. The procedure is equivalent to those presented in [1, 3], and can be seen as a combination of two phases: a forward phase and a delete phase, defined as follows:

- The *forward phase* runs LR-CMIC and its output is a feasible solution of CMIC.
- In the *delete phase*, we begin with the output of the forward phase and we keep deleting workers from the solution (in an arbitrary order) while maintaining feasibility of the solution, i.e., satisfying the covering constraint.

The delete phase produces a minimal feasible solution contained within the solution returned by LR-CMIC and the minimality feature is critical to reduce the approximation factor. However, we highlight that the delete phase is the one responsible for the loss of the monotonicity guarantee.

4.2 Results of the experimental evaluation

We dedicate this subsection to presenting the results of our experiments and drawing conclusions from them. We will analyze these results using a set of related graphs that we have plotted.

We commence our analysis by computing the average approximation ratios for both algorithms presented in Subsection 4.1, effectively illustrating their correlation (Figure 11, Left). Additionally, we extend this analysis to determine the worst approximation ratios for these algorithms (Figure 11, Right). Notably, our experiments have shown that LR-CMIC demonstrates significantly superior practical performance compared to the theoretical bound that we established on its performance, namely of a Δ -approximation. Specifically, our observations reveal that the algorithm only yielded a worst-case approximation ratio exceeding 2, in very few instances. Furthermore, LR-DEL-CMIC also

consistently outperforms the known 4-approximation bound in practice. More precisely, our sample experiments indicate that it never exceeds the 2-approximation ratio threshold. This, coupled with the absence of a tight instance in the proof of the guarantee of 4, leads us to conjecture that a theoretical guarantee of 2 might indeed be possible. Importantly, the algorithm LR-DEL-CMIC that incorporates the delete phase enhances the solution quality of LR-CMIC, as expected. However, the solution quality of LR-CMIC always remains in close proximity to the optimal solution. This, coupled with the fact that LR-CMIC satisfies the strategyproofness axiom, positions it as an excellent candidate for real-life usage in crowdsourcing applications.

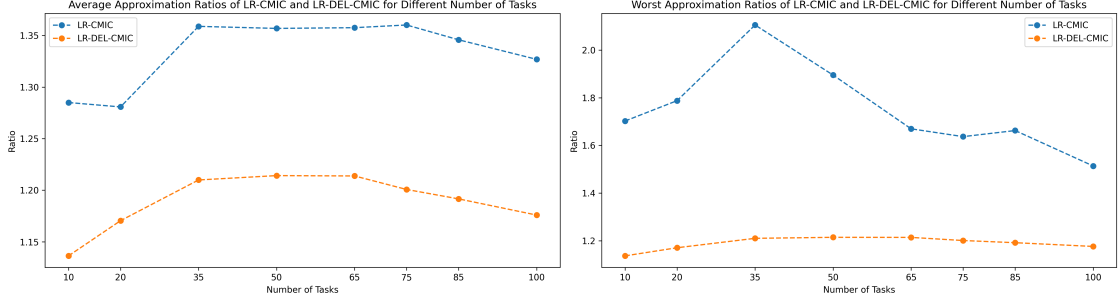


Figure 11: Average and Worst Approximation Ratios of LR-CMIC and LR-DEL-CMIC.

Moving forward, our objective is to evaluate the cost-effectiveness of the LR-DEL-CMIC algorithm compared to LR-CMIC, focusing on both average performance and worst-case scenarios (Figure 12). Once again, the findings present intriguing insights favoring the use of the monotone LR-CMIC over the theoretically superior LR-DEL-CMIC. It becomes evident that the improvement on the cost of the final solution, that is brought by the deletion phase is not substantial on the average case. Therefore, LR-CMIC not only holds up well against the optimal solution (Figure 11) but also compares favorably to the cost of the solution returned by LR-DEL-CMIC. This observation suggests that, although LR-CMIC theoretically provides a worse approximation guarantee than LR-DEL-CMIC, in practice, it can deliver results very close to LR-DEL-CMIC while maintaining strategyproofness, unlike LR-DEL-CMIC. Nonetheless, there are multiple instances where we observe a significant improvement of about 25-45%.

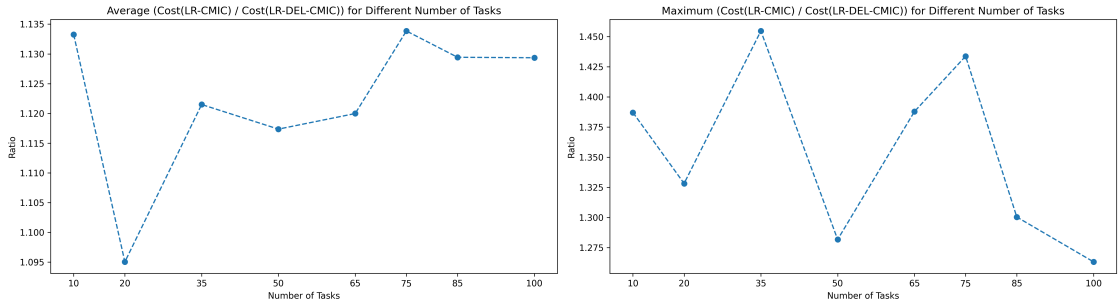


Figure 12: Average and Worst Cost Ratios of LR-CMIC compared to LR-DEL-CMIC.

To conclude our analysis, we highlight some key findings related to the Δ parameter. Recall that Δ is the number of workers assigned to a single task: a crucial factor in the theoretical examination of LR-CMIC. As depicted in Figure 13, the average value of Δ shows a slight increase with the growing number of tasks. This aligns with expectations,

given that the total number of workers also escalates with the growth in the number of tasks. While the conclusion for the maximum observed value of Δ may not be as straightforward, a similar observation can be drawn in that regard.

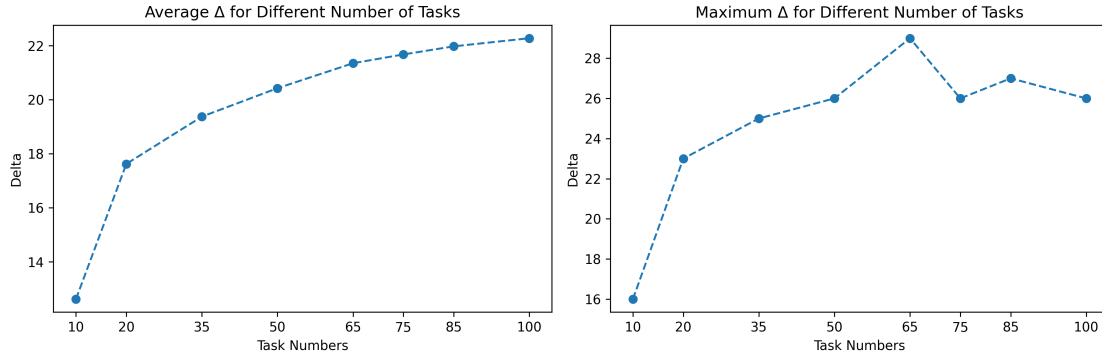


Figure 13: Average and maximum observed value of Δ , for different number of tasks.

5 Conclusions

We have completed an experimental evaluation for two major algorithmic questions regarding task assignment in procurement auctions. For the problem of social welfare maximization, the main take-home story is that algorithms based on Linear Programming methodologies seem to have the best performance. At the same time, some carefully constructed greedy algorithms also behave favorably, whereas simpler greedy choices may have inferior performance. Regarding the problem of cost minimization, the main message from the experiments is that algorithms based on the local ratio technique seem to be very good candidates for practical deployment. We believe that these insights can also help in designing algorithms for real applications in procurement auctions and crowdsourcing markets.

References

- [1] Amotz Bar-Noy, Reuven Bar-Yehuda, Ari Freund, Joseph Naor, and Baruch Schieber. A unified approach to approximating resource allocation and scheduling. *Journal of the ACM*, 48(5):1069–1090, 2001.
- [2] Niv Buchbinder, Moran Feldman, Joseph Seffi Naor, and Roy Schwartz. A tight linear time $(1/2)$ -approximation for unconstrained submodular maximization. *SIAM Journal on Computing*, 44, 2015.
- [3] Venkatesan T Chakaravarthy, Amit Kumar, Sambuddha Roy, and Yogish Sabharwal. Resource allocation for covering time varying demands. In *European Symposium on Algorithms*, pages 543–554. Springer, 2011.
- [4] Christopher Harshaw, Moran Feldman, Justin Ward, and Amin Karbasi. Submodular maximization beyond non-negativity: Guarantees, fast algorithms, and applications. *36th International Conference on Machine Learning, ICML 2019*, 2019-June, 2019.
- [5] Tianyuan Jin, Yu Yang, Renchi Yang, Jieming Shi, Keke Huang, and Xiaokui Xiao. Unconstrained submodular maximization with modular costs: Tight approximation and application to profit maximization. volume 14, 2021.

- [6] Evangelos Markakis, Georgios Papasotiropoulos, and Artem Tsikiris. On improved interval cover mechanisms for crowdsourcing markets. In *International Symposium on Algorithmic Game Theory*, pages 94–112. Springer, 2022.
- [7] Sofia Maria Nikolakaki, Alina Ene, and Evimaria Terzi. An efficient framework for balancing submodularity and cost. 2021.
- [8] Hamed Shah-Mansouri and Vincent W. S. Wong. Profit maximization in mobile crowdsourcing: A truthful auction mechanism. In *Proceedings of the IEEE International Conference on Communications, ICC 2015*, pages 3216–3221, 2015.
- [9] Yaron Singer. Budget feasible mechanisms. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010*,, pages 765–774, 2010.