Multi Period Assignment Problem for Social Engagement and Opportunistic IoT

Multi Period Assignment problem for Social Engagement and Opportunistic IoT

Edoardo Fadda
Politecnico di Torino
Torino, Italy
Email: edoardo.fadda@polito.it

Dario Mana
TIM Joint Open Lab
Torino, Italy
Email: dario.mana@telecomitalia.it

Guido Perboli
ICT for City Logistics and Enterprises Center
Politecnico di Torino, Torino, Italy
Email: guido.perboli@polito.it

Roberto Tadei
Politecnico di Torino
Torino, Italy
Email: roberto.tadei@polito.it

***Abstract*—Due to the diffusion of Internet of Things (IoT), many devices such as water meters, smart dumpsters, and many other objects have the capacity to record data. Gathering these data from the devices is a problem that could be solved in three ways: by building a huge network infrastructure, by using regular workforce or by using opportunistic IoT networks, i.e. by using as mobile hotspots the devices of selected users. The latter is cheaper than the others, requiring only the payment of a reward to the users. In this paper, we introduce a Multi Period Assignment problem, i.e. a problem for planning the operations of Opportunistic IoT networks. The problem minimizes the sum of user rewards, while gathering data from all devices. An effective heuristic method able to deal with realistic-sized instances is presented. The heuristic is able to find, by using a reasonable amount of time, the optimum for 124 out of 128 instances and reach gaps smaller than 0.1% for the remaining 4 instances.**

***Keywords*—Internet of Things; Multi Period Assignment problem; Heuristics**

1. Introduction

The diffusion of sensors networks for gathering data all over the city and their integration in business intelligence and operation management processes are nowadays increasing. Many of these operations simply require easy tasks such as collect data from water meters or urban sensors. Moreover, the interoperability of a large part of the sensors used in Internet of Things (IoT) applications with smartphones enables the standard mobile users to perform these data gathering tasks. On the contrary, these operations are normally performed by using ad-hoc networks (with a large infrastructural cost) or by trained staff, with high variable economic and environmental variable costs (i.e. pollution). The emerging business model able to solve this problem is called social engagement, i.e., a company uses social engagement if it uses mobile applications in order to ask people to perform tasks in order to reach a business goal, while giving a reward to people who are assigned to tasks. This strategy is already used in the e-grocery domain by Walmart (US grocery retailers): it asks in-store shoppers to carry packages to on-line shoppers for a discount [1].

In this paper, we consider a new application of the social engagement for building low-cost temporary IoT networks, the opportunistic IoT (o-IoT) networks. This business model tries to solve the problem of gather data from a distributed network of sensors in an urban area by using as mobile hotspots the devices of selected users. This application is critical because without the proposed opportunistic connections to gather data from these devices requires a huge network infrastructure able to cover the whole city. O-IoT inspires the Coiote project by TIM (the largest Italian telecommunication company) [2], this project is, as far as the authors know, the first attemp of the implementation of this framework in a real application. The goal of this project is to develop a mobile phone application enabling TIM to ask users to do some tasks in relation to the mobile phone cell where they are located. The tasks that the users are asked to do are to share their mobile connection with smart dumpsters. In this way, the smart dumpsters can transmit to the central unit the data related to the amount of waste that they have collected and the company in charge of the waste collection can plan the operations in an optimal way. This architecture is shown in Figure 1. TIM rewards the tasks. The main objective of this paper is to define a mathematical model suitable to help companies that use social engagement. The objective of the model is to minimize the total cost of the rewards that the company has to pay while doing all the tasks in every cell before the end of the considered time interval. The model that describes the problem is a customized version of the Multi Period Assignment Problem (MPAP). To our knowledge, this is the first time that such a problem is presented. The computational experiments show that exact methods perform poorly on big instances because

they require too much time with respect to the real world applications requirements. For this reason, we introduce a heuristic able to solve the test instances with a smaller computational effort and with a precision comparable to the exact method. The article is organized as follows. In Section 2 we review the literature about the IoT and the MPAP. In Section 3 we present the mathematical model. In Section 4 we describe the heuristic. Due to the lack of literature about this problem, we define some freely available benchmark instances. In Section 5 we describe and use them in order to study the performance of the heuristic. Finally, in Section 6 we outline the main results achieved in this paper.

2. Literature Review

The main topic of this paper is the application of optimization techniques to social engagement and, in particular, to o-IoT. Since to our knowledge there are no other studies in the field, we split the literature review in two axis. The first one considers the applications of optimization techniques to the IoT framework; the second one considers the MPAP problem.

IoT is the enrichment of devices with sensors and with the capacity of exchange data. Whether these devises have also actuators, then the range of applications increase and encompasses also smart grids, intelligent transportation and smart cities (for a survey of these applications the user is referred to [3], [4] and [5]). In the IoT framework, optimization plays an important role. For example, in [6] the authors propose an intelligent transportation system that uses information from a network of sensors in order to better plan the vehicles routing and in [7] the authors describe an heuristic that optimizes the waste collection operations using the data about the waste production collected from these vehicles.

The second axe considers optimization problems. In particular, we consider a customized version of the assignment problem (see [8] and [9] for a review). All the optimization problems related to the assignment problems have in common two features: tasks (or operations) to be done and resources to be allocated to each task. In this setting, the tasks are the collections of data from urban sensors, while the resources are the application users. The tasks can be performed by all the users and the users can perform all the tasks. Since the operations are not critical, we have a time interval to perform all the tasks. For this reason, we consider a MPAP. Unluckily, the literature about this problem is not so developed. Some similar applications that we have found are [10] and [11]. In the first paper, the authors study a binary multi-period assignment problem arising as a part of a weekly planning problem in mail processing operations. In the second paper, the authors consider the classical MPAP where the main decision variables are the binary variables describing if it is better to switch a person from the task that he/she is performing to another one or not. Both these papers consider binary decision variables while in this paper we consider integer decision variables.

3. Mathematical Model

In this section, we introduce the mathematical model that describes the problem of minimizing the total amount of rewards while satisfying all the tasks that the company must do in each cell. In the following, we call users the people available to perform tasks. The term user is because these people are users of the application through which the company asks them to do tasks. Furthermore, we call tasks or activities the operations that must be performed by the company. The mathematical model that describes the problem uses the following sets:

- $\mathcal{T}$ is the set of all time indexes. The cardinality of this set is T,
- $\mathcal{I}$ is the set of all cells. The cardinality of this set is I,
- $\mathcal{M}$ is the set of all user types. The cardinality of this set is M.

In the model, we use the following parameters:

- c_{ij}^{tm} is the cost of the reward for a customer of type m in cell i at time t that goes to cell j.
- N_j is the number of tasks that must be done in the operational cell j during the time interval $[1, 2, \ldots, T]$.
- n_m is the number of tasks that a user of type m can do.
- θ_i^{tm} is the number of users of type m in cell i during time step t.

In the model we use the variables x_{ij}^{tm}. They describe the number of customers of type m that are asked to do n_m tasks in cell j, starting from cell i, during time step t.

The optimization problem is then:

$$\min_{x_{ij}^{tm}, \forall i,j,t,m} \sum_{i=1}^{I} \sum_{j=1}^{J} \sum_{t=1}^{T} \sum_{m=1}^{M} c_{ij}^{tm} x_{ij}^{tm} \tag{1}$$

subject to

$$\sum_{t=1}^{T} \sum_{m=1}^{M} \sum_{i=1}^{I} n_m x_{ij}^{tm} \geq N_j \quad \forall \ j \in \mathcal{I} \tag{2}$$

$$\sum_{j=1}^{J} x_{ij}^{tm} \leq \theta_i^{tm} \quad \forall \ i \in \mathcal{I} \ \ t \in \mathcal{T} \ \ m \in \mathcal{M} \tag{3}$$

$$x_{ij}^{tm} \in \mathbb{Z}^+ \quad \forall \ i \in \mathcal{I} \ \ j \in \mathcal{J} \ \ t \in \mathcal{T}. \tag{4}$$

The objective function (1) is the total amount of rewards that the company has to pay. Constraints (2) impose that during the time interval $[1, 2, \ldots, T]$ all tasks must be performed. Constraints (3) bound the number of users of each type in each cell during each time step. All decision variables are non-negative integer.

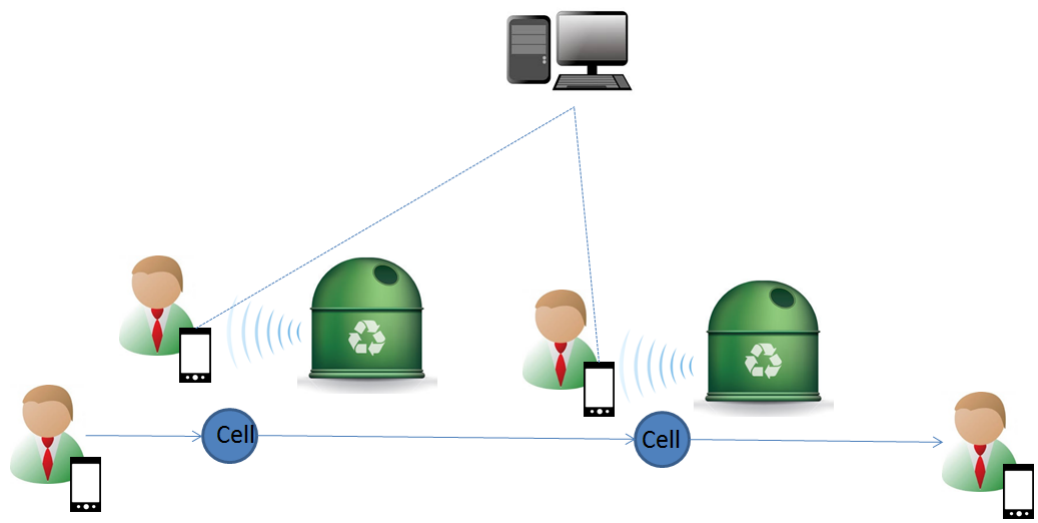

Figure 1. The figure shows how the Coiote project will work

4. Heuristic

In this section we describe the meta heuristic used in order to find a good solution of problem (1)-(4). The meta heuristic is composed by two steps: an *Outer heuristic* and a *Greedy step*. The *Outer heuristic* performs the steps shown in Algorithm 1. Basically, *Outer heuristic* produces a random sequence corresponding to the visiting order of the cells, it applies the *Greedy step* on the sequence and, if necessary, it updates the value of the best solution found. It iterates these steps until there is enough time.

Algorithm 1: Outer heuristic algorithm finds an heuristic solution ($\hat{x}_{ij}^{tm}$) given an instance

```
Outer Heuristic (Instance);
Input  : Instance File
Output: x̂_ij^tm
best_opt=+Inf;
while there is still time do
    cells_sequence = random_shuffle(cells_sequence);
    [opt, x̂_ij^tm] = Greedy step(cells_sequence);
    if opt < best_opt then
        x̂_ij^tm = x_ij^tm ∀ i, j, t, m. ;
        best_opt = opt ;
    end
end
```

The *Greedy step* is the logical core of the proposed method. It takes as input the random sequence generated by the *Outer heuristic*. The main steps are shown in Algorithm 2. For each cell in the sequence it fulfils the demand of each sink cell by considering each source cell that has available resources in an order defined by the function *Minimum_Cost*.

In the first run, the *Minimum_Cost* function orders the users by decreasing values of the ratio $\frac{c_{ij}^{tm}}{n_m}$. From the second run on, it changes this order randomly in order to increase the exploitation of the solution space.

Algorithm 2: Greedy Step algorithm finds an heuristic solution ($\hat{x}_{ij}^{tm}$) and its value *opt* given a cells sequence

```
Greedy Step (cells_sequence);
Input  : cells_sequence
Output: [opt, x̂_ij^tm]
θ̂_i^tm = θ_i^tm;
x̂_ij^tm for all i, t and m;
N̂_j = N_j ;
for each cell j in cells_sequence do
    list_m_i_t = Minimum_Cost(c_ij^tm, θ̂_i^tm);
    count = 0;
    while N̂_j ≥ 0 do
        [m, i, t] = list_m_i_t[count];
        M_i^tm = min[θ̂_i^tm, N_i/n_m] ;
        x̂_ij^tm = x̂_ij^tm + M_i^tm ;
        θ̂_i^tm = θ̂_i^tm − M_i^tm ;
        N̂_j = N̂_j − n_m θ̂_i^tm ;
        count = count+1 ;
    end
end
Try_Improve(x̂_ij^tm ∀ i, j, t, m.);
```

Finally, the function *Try_Improve* tries to find a set of changes (i.e. modifications about which groups of users perform the requested tasks) which as a whole leads to a smaller value of the objective function. In particular, starting from an already feasible solution, the function removes one or more users doing activities in a given destination cell and then it tries to find other users that are able to perform better. In case the selected customers are available, the recursion terminates with a positive result and the solution is updated.

If the users are not available, on the other hand, the function checks whether it is possible to replace some other activities done by the chosen users in other destination cells through a recursive call of the function.

This heuristic is very effective if

$$\sum_{i=1}^{I}\sum_{t=1}^{T}\sum_{m=1}^{M} n_i\theta_i^{tm} \geq 2\sum_{i=1}^{I} N_i, \tag{5}$$

i.e. there is a surplus of resources. If (5) does not hold, a modified version of the greedy function can be used. This new version is similar to the previous one, but it is characterized by a two steps procedure. In the first iteration, the method avoids to choose users that would lead to a waste of activities (i.e. more activities done than the requested number). In the second one, this additional constraint is relaxed with the hope to be able to perform the remaining applications.

5. Numerical Experiments

In this section, we describe how we generate the instances of the problem and how we solve them by mean of the proposed heuristic. In order to explain how we generate the instances we consider the various parameters and dimensions that describe the problem. First, we set T to be 1 or 20, the first value simulates the on-line optimization, while the second one simulates a planning for a reasonable time interval (in the latter case we interpret each time step to be one hour). The coefficient M describes the number of customers types that we consider. We set $M = 3$ for all the experiments. The reason of this choice is that in this way we can model standard users m_0, business users m_1 and regular workforce m_2, which is a realistic setting. In particular, standard users are people that do a small amount of tasks for a cheap cost, business users cost more but do more tasks and finally, the regular workers perform several operations but they are the most expensive resource. We suppose that the optimal solution uses regular workforce only if it is unable to perform tasks with the other two types. Finally, the coefficient I represents the number of cells. We vary it from 30 to 300. This choice represents a very small city and a grid of a reasonable size. In real instances of big cities I can reach values near to 1000. Finally, in order to define the network we have to set how many sources and sinks there are. We define ρ to be the value that describes the ratio between sources and sinks. In these instances of the problem, the cells provide or ask resources but not both. The reason of this choice is because for a low cost, users in a cell can perform the tasks in the same cell. In this way, we obtain for each cell either a surplus of users or a surplus of tasks. Note that in the model presented in Section 3, each cell i can have both requests N_i and resources θ_i^{tm}, $\forall t, m$.

The coefficients that we need in order to describe the instance are c_{ij}^{tm}, N_i, n_m and θ_i^{tm}. The parameters c_{ij}^{tm} describe the costs of rewards. We define them by

$$c_{ij}^{tm} = \begin{cases} |\frac{i-j}{4}+1|C\log(2), & \text{if } m = m_0 \\ |\frac{i-j}{4}+1|C\log(4), & \text{if } m = m_1 \\ |\frac{i-j}{4}+1|C\log(6), & \text{if } m = m_2 \end{cases}, \tag{6}$$

where C is a realization of a random variable uniformly distributed between $C_{\min}$ and $C_{\max}$ ($C \sim \mathcal{U}[C_{\min}, C_{\max}]$). In the following we assume $C_{\min} = 2$ and $C_{\max} = 5$.

The parameters N_i are the numbers of tasks to do in cell i. We define them by sampling a uniform distribution between 0 and $N_{\max}$ ($N_i \sim \mathcal{U}[0, N_{\max}]$). In the following simulations $N_{\max} = 100$.

The parameters n_m are the numbers of tasks that each user type can perform. We impose that the standard users perform 1 task ($n_{m_0} = 1$), that the business resources perform 2 tasks ($n_{m_1} = 2$) and that the regular workers perform 10 tasks ($n_{m_2} = 3$).

Finally, we have to define θ_i^{tm} $\forall$ i, t, m. In order to define these parameters we have to describe the distribution of each θ_i^{tm}. This can be done, as in [12], through a normal distribution $\mathcal{N}(\frac{N_{\max}}{2}, (\frac{N_{\max}}{2})^2)$. In particular, it is possible to verify that a feasible solution exists by checking that

$$\sum_{i=1}^{I}\sum_{t=1}^{T}\sum_{m=1}^{M} n_i\theta_i^{tm} \geq \sum_{i=1}^{I} N_i. \tag{7}$$

If (7) does not hold, then we add $\sum_{i=1}^{I} N_i - \sum_{i=1}^{I}\sum_{t=1}^{T}\sum_{m=1}^{M} n_i\theta_i^{tm}$ people to a random set of cells (i.e. we increase θ_i^{tm} for some cells) in order to satisfy (7).

The instances that we generate are called $Co\_I\_T\_n$, where I indicates the number of cells considered, T the number of time periods considered and n is the numeric identification of the instance. The instances can be downloaded from [1].

In order to compute the optimal value for all instances we use the commercial solver gurobi[2]. All the following experiments are performed on an Intel R CoreTMi7-5500U CPU @2.40 Ghz with 8 GB RAM and Microsoft R WindowsTM10 Home installed.

We compare the performances of the commercial solver and the performance of the heuristic in Tables 1, 2, 3 and 4. In Table 1 and in Table 2 we show the performances of the heuristic on the instances that considers 30 cells, 1 time period and 30 cells, 20 time periods. As the reader can notice, for these instances the proposed heuristic is slower than the solver. This is due to the time spent in order to build the knowledge base. Furthermore, the heuristic fails to find the optimum in those instances Co_30_1_NT_9 and Co_30_1_T_1. Nevertheless, the time spent in the construction of the knowledge base produces very good results in larger instances (as the reader can see in Table 3 and in Table 4). In all the instances considering 100 and 300 cells, the heuristic finds the optimal solution and, the average computational time is reduced by the 75% for the

1. https://bitbucket.org/orogroup/mpap
2. http://www.gurobi.com

TABLE 1. THE TABLE SHOWS THE OPTIMAL SOLUTION (OPT.SOL.), THE TIME USED BY THE COMMERCIAL SOLVER TO FIND IT (TIME), THE TIME USED BY THE HEURISTIC (HEU.TIME) AND THE VALUE OF THE HEURISTIC SOLUTION (HEU.SOL.) FOR DIFFERENT INSTANCES. ALL INSTANCES IN THE TABLE CONSIDER 30 CELLS AND 1 TIME PERIOD.

Instance	Opt. Sol.	Time	Heu. Time	Heu. Sol.
Co_30_1_NT_0	1041	0.014	1.25039	1041
Co_30_1_NT_1	1756	0.013	1.25028	1756
Co_30_1_NT_2	2341	0.017	1.25023	2341
Co_30_1_NT_3	2105	0.028	1.25018	2105
Co_30_1_NT_4	1477	0.018	1.25024	1477
Co_30_1_NT_5	2996	0.021	1.25025	2996
Co_30_1_NT_6	1623	0.009	1.2502	1623
Co_30_1_NT_7	1032	0.01	1.25022	1032
Co_30_1_NT_8	2288	0.018	1.25027	2288
Co_30_1_NT_9	1562	0.018	1.25024	1563
Co_30_1_T_0	1105	0.006	1.25026	1105
Co_30_1_T_1	1796	0.013	1.25022	1797
Co_30_1_T_2	2437	0.017	1.25021	2437
Co_30_1_T_3	2073	0.008	1.25025	2073
Co_30_1_T_4	1545	0.02	1.25018	1545
Co_30_1_T_5	2888	0.009	1.2502	2888
Co_30_1_T_6	1592	0.011	1.2502	1592
Co_30_1_T_7	1394	0.008	1.25018	1394
Co_30_1_T_8	2012	0.01	1.25033	2012
Co_30_1_T_9	1820	0.011	1.25026	1820

TABLE 2. THE TABLE SHOWS THE OPTIMAL SOLUTION (OPT.SOL.), THE TIME USED BY THE COMMERCIAL SOLVER TO FIND IT (TIME), THE TIME USED BY THE HEURISTIC (HEU.TIME) AND THE VALUE OF THE HEURISTIC SOLUTION (HEU.SOL.) FOR DIFFERENT INSTANCES. ALL INSTANCES IN THE TABLE CONSIDER 30 CELLS AND 20 TIME PERIODS.

Instance	Opt. Sol.	Time	Heu. Time	Heu. Sol.
Co_30_20_NT_0	872	0.106	1.25083	872
Co_30_20_NT_1	457	0.117	1.25093	457
Co_30_20_NT_2	706	0.142	1.25064	706
Co_30_20_NT_3	827	0.076	1.25242	827
Co_30_20_NT_4	437	0.133	1.25078	437
Co_30_20_NT_5	984	0.124	1.25076	984
Co_30_20_NT_6	937	0.104	1.25078	937
Co_30_20_NT_7	1132	0.09	1.25072	1132
Co_30_20_NT_8	719	0.115	1.25081	719
Co_30_20_NT_9	895	0.119	1.25079	895
Co_30_20_T_0	872	0.095	1.25076	872
Co_30_20_T_1	457	0.097	1.25078	457
Co_30_20_T_2	721	0.142	1.25077	721
Co_30_20_T_3	827	0.086	1.25082	827
Co_30_20_T_4	437	0.138	1.2508	437
Co_30_20_T_5	991	0.125	1.25072	991
Co_30_20_T_6	933	0.118	1.25071	933
Co_30_20_T_7	1143	0.085	1.25083	1143
Co_30_20_T_8	4453	0.127	1.25072	4453
Co_30_20_T_9	4530	0.108	1.25077	4530

instances with 100 cells and by the 210% for the instances with 300 cells.

6. Conclusions

In this paper, we define a new problem which goal is to minimize the costs of using social engagement. In particular, we consider o-IoT applications. Further, we develop a simulation framework and benchmark instances and, by doing so, we fill a lack both in the optimization and the IoT

TABLE 3. THE TABLE SHOWS THE OPTIMAL SOLUTION (OPT.SOL.), THE TIME USED BY THE COMMERCIAL SOLVER TO FIND IT (TIME), THE TIME USED BY THE HEURISTIC (HEU.TIME) AND THE VALUE OF THE HEURISTIC SOLUTION (HEU.SOL.) FOR DIFFERENT INSTANCES. ALL INSTANCES IN THE TABLE CONSIDER 100 CELLS AND 1 TIME PERIOD.

Instance	Opt. Sol.	Time [s]	Heu. Time [t]	Heu. Sol.
Co_100_1_NT_0	5270	5.34234	1.25027	5270
Co_100_1_NT_1	3811	5.23475	1.25029	3811
Co_100_1_NT_2	4455	5.23625	1.25157	4455
Co_100_1_NT_3	4832	5.52352	1.25033	4832
Co_100_1_NT_4	4790	5.65343	1.25043	4790
Co_100_1_NT_5	6493	5.34634	1.25027	6493
Co_100_1_NT_6	4276	5.34534	1.25039	4276
Co_100_1_NT_7	4815	5.34564	1.25031	4815
Co_100_1_NT_8	4636	5.25154	1.25045	4636
Co_100_1_NT_9	4691	5.34523	1.25035	4691
Co_100_1_T_0	5350	5.43523	1.25034	5350
Co_100_1_T_1	3919	5.26234	1.25027	3919
Co_100_1_T_2	4764	5.34523	1.25031	4764
Co_100_1_T_3	5215	5.34265	1.25031	5215
Co_100_1_T_4	5012	5.29955	1.25036	5012
Co_100_1_T_5	6730	5.32334	1.25027	6730
Co_100_1_T_6	3625	5.25543	1.25027	3625
Co_100_1_T_7	3203	5.33435	1.25035	3203
Co_100_1_T_8	2196	5.34352	1.25041	2196
Co_100_1_T_9	2182	5.32345	1.25035	2182

TABLE 4. THE TABLE SHOWS THE OPTIMAL SOLUTION (OPT.SOL.), THE TIME USED BY THE COMMERCIAL SOLVER TO FIND IT (TIME), THE TIME USED BY THE HEURISTIC (HEU.TIME) AND THE VALUE OF THE HEURISTIC SOLUTION (HEU.SOL.) FOR DIFFERENT INSTANCES. ALL INSTANCES IN THE TABLE CONSIDER 300 CELLS AND 20 TIME PERIODS.

Instance	Opt. Sol.	Time	Heu. Time	Heu. Sol.
Co_300_20_NT_0	7019	25.628	1.29306	7019
Co_300_20_NT_1	7183	28.814	1.29125	7183
Co_300_20_NT_2	8101	21.883	1.29518	8101
Co_300_20_NT_3	7638	25.531	1.29798	7638
Co_300_20_NT_4	8193	24.294	1.29795	8193
Co_300_20_NT_5	7580	21.724	1.28707	7580
Co_300_20_NT_6	7681	23.469	1.2945	7681
Co_300_20_NT_7	8546	17.117	1.29569	8546
Co_300_20_NT_8	7129	17.382	1.29748	7129
Co_300_20_NT_9	7191	19.063	1.28753	7191
Co_300_20_NT_10	8074	25.28	1.29103	8074
Co_300_20_T_0	7024	27.258	1.30792	7024
Co_300_20_T_1	7183	30.834	1.28962	7183
Co_300_20_T_2	8102	22.448	1.29249	8102
Co_300_20_T_3	7659	25.1	1.28967	7659
Co_300_20_T_4	8223	25.942	1.29351	8223
Co_300_20_T_5	7600	21.737	1.29414	7600
Co_300_20_T_6	7647	23.244	1.29524	7647
Co_300_20_T_7	8590	19.756	1.30957	8590
Co_300_20_T_8	7140	23.853	1.28804	7140
Co_300_20_T_9	7227	19.247	1.29744	7227
Co_300_20_T_10	8047	23.834	1.29261	8047

literature. Finally, we show by means of numerical examples that the proposed heuristic is able to perform well on the set of generated instances .

Thanks to the proposed approach, o-IoT operations can be optimize in real time. Furthermore, the low computation time of the heuristic enables the company to run it several time with different parameters and to choose the solution that more suits its needs. Moreover, by lowering the price for collecting data from several distributed sensors, the proposed approach enables municipalities and companies to implement smart city policies otherwise impossible.

The deterministic problem defined does not consider the uncertainty related to the number of people in each cell during a certain time period and the uncertainty related to the people that accept to do a task but that do not perform it. Nevertheless, the proposed heuristic is useful in order to solve single scenario problems that can arise during the solution of the stochastic model, if it is solved by using techniques such as progressive hedging. Future improvement of the proposed methodology will consider the similarity of that problem with the network transportation problem.

References

[1] M. Bender, "Piloting delivery with uber, lyft and deliv," http://blog.walmart.com/business/20160603/piloting-delivery-with-uber-lyft-and-deliv, 2016, accessed: 2017/03/30.

[2] TIM Jol Swarm, "Coiote project," http://jol.telecomitalia.com/jolswarm/coiote-opportunistically-connecting-the-internet-of-things/, 2016, accessed: 2017/03/30.

[3] M. Kaur and S. Kalra, "A review on IOT based smart grid," *International Journal of Energy, Information and Communications*, vol. 7, no. 3, p. 1122, 2016.

[4] K. N. Qureshi and A. H. Abdullah, "A survey on intelligent transportation systems," *Middle-East Journal of Scientific Research*, vol. 15, no. 5, pp. 629–642, 2013.

[5] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, p. 2232, 2014.

[6] A. C. Cagliano, L. Gobbato, R. Tadei, and G. Perboli, "Its for e-grocery business: The simulation and optimization of urban logistics project," *Transportation Research Procedia*, vol. 3, p. 489498, 2014.

[7] G. Perboli, E. Fadda, L. Gobbato, M. Rosano, and R. Tadei, "Optimization for networked data in environmental urban waste collection: the ONDE-UWC project," *28th European Conference on Operational Research, Poznań, Poland, 4-7 July 2016*.

[8] U. Derigs, "The shortest augmenting path method for solving assignment problems motivation and computational experience," *Annals of Operation Research*, vol. 4, p. 57102, 1985.

[9] S. Martello and P. Toth, "Linear assignment problems," *Annals of Discrete Mathematics*, vol. 31, p. 259282, 1987.

[10] X. Zhang and J. F. Bard, "A multi-period machine assignment problem," *European Journal of Operational Research*, vol. 170, no. 2, pp. 398 – 415, 2006.

[11] J. E. Aronson, "The multiperiod assignment problem: A multicommodity network flow model and specialized branch and bound algorithm," *European Journal of Operational Research*, vol. 23, no. 3, pp. 367–381, 1986.

[12] E. Fadda, G. Perboli and R. Tadei, "Customized multi-period stochastic assignment problem for social engagement and opportunistic IoT," Submitted.