Concurrent Real-Time optimization of detecting unexpected tasks in IoT design process using GA

Adam Górski
Dept. of Information Technologies
Jagiellonian University
Cracow, Poland
a.gorski@uj.edu.pl

Maciej Ogorzałek
Dept. of Information Technologies
Jagiellonian University
Cracow, Poland
maciej.ogorzalek@uj.edu.pl

***Abstract*— In this paper we present a genetic algorithm for concurrent real-time optimization for detecting unexpected tasks in IoT design process. The process can be split into two phases which impact each other in real-time. Modification of one phase modifies the second one. The algorithm detects unexpected tasks as a connection of parts of predicted tasks. Therefore, there can exist more than one way to resolve the unexpected situation. The algorithm searches for optimal solution making concurrent optimization of two phases of the design process. Use of genetic algorithm helps also to eliminate connections of subtasks which do not give a good result.**

Keywords*— *optimization, genetic algorithm, artificial intelligence, Internet of Things, unexpected tasks

I. Introduction

Development of new technologies requires development of new tools for optimization. In this paper we investigate an optimization problem which appeared in the Internet of Things (IoT) [1] design process. This problem is related to assignment of unexpected tasks for embedded systems [2][3][4]. Internet of Things (IoT) is a network consisting of connected "things" like: physical systems, sensors, communication protocols, embedded systems, etc. The elements of the network should be uniquely identified. One of popular methods of identification are RMID (Radio Frequency IDentification) tags [6]. Some elements of IoT network are able to perform/execute tasks. Such elements can be divided into two groups: Processing Resources (PRs) and Communication Resources (CRs). PRs are responsible for tasks execution. CRs provide communication between connected PRs. PRs can be further divided into two groups: Dedicated Resources (DRs) which are able to execute only one specific type of task, and Universal Resources (URs) which can execute more than one type of task. IoT concepts are widely used in "smart" solutions like smart cities [7][8], smart transport [9], smart houses [5][10] and many others [11]. The designers of IoT systems concentrate on communication [6] [12] and security problems [13][14][15].

The design methods, like for example [16], do not investigate unexpected situations and concentrate only on extending IoT network with new elements. When IoT network is designed, modifications to a new situation is not always possible as the network cannot be freely extended by new resources. The problem of unexpected tasks was described by Górski and Ogorzałek [2][4] but only for the case of embedded systems. The strongest constraint of their algorithm is that all of the information about unexpected tasks must be given. In [3] authors propose an algorithm to detect unexpected tasks for embedded systems. The algorithm finds unexpected tasks as connections of some numbers of subtasks of predicted tasks. The authors indicated that unexpected situation can be resolved in different ways, but did not search for optimal set of the subtasks. Therefore, the algorithm was finding the first solution, not necessary the best one. In this paper we also concentrate on situation when unexpected tasks can be presented as a set of some numbers of subtasks of other tasks that need to be executed by the network. However it is not known at the beginning which set of the subtasks is better. What is more, it is not known which sets of subtasks can be even equal to execute unexpected task. Therefore, the problem can be divided into two phases. The first phase is responsible for finding number of subtasks necessary to solve unexpected situation. The second phase finds the optimal assignment of subtasks to the resources. However every subtask can be executed on many elements of IoT network which can consist of many resources. Every resource can have different values of optimizing parameters for each subtasks. Thus we would not have been able to find the optimal set of subtasks if those two phases would have been optimized separately. Moreover, separate optimization of those phases does not give the information about optimization criteria. Therefore, these two phases must be optimized concurrently. It has to be noted that optimization in each phase impacts the other in real-time. Further description of the problem is given in chapter 4.

In this paper we propose a genetic algorithm to solve the optimization problem of two concurrent phases which impact each-other in real-time during the IoT design process. Such algorithms [17] start with an initial population of individuals, and using genetic operators: selection, crossover and mutation generate new populations of individuals. Presented algorithm enables to eliminate connections of subtasks which do not solve unexpected situations. The algorithm chooses the best connection of subtasks and concurrently assigns them to resources. Genetic algorithms are known for being able to escape local minima of optimized function in the parameter space. Further, using genetic solution to the problem investigated in this paper can help to generate different set of subtasks to solve unexpected situation.

The paper is organized as follows: chapter 2 gives basic notations used in the paper, chapter 3 describes representation of IoT and the IoT design process. Chapter 4 contains general presentation of new optimization problem that needs to be solved. In chapter 5 the algorithm is described. Chapter 6 gives the description of experiments. In chapter 7 conclusions are discussed and future work is also considered.

II. Basic Notation

PR – processing resource
DR – dedicated resource
UR – universal resource
CR – communication resource
T_i – i number of task
G=(T,E) – task graph
E – edge in the graph
$e_{ij} \in E$ – egde between tasks Ti and Tj

t – time of execution
ts – starting moment of execution
c – cost of execution
$tc_{i,j}$ – communication time between tasks Ti and Tj
b – bandwidth of Communication Resource
d – amount of data that need to be send between two connected tasks
n – number of predicted tasks
u – number of unexpected tasks
v – number of processing resources
m – number of universal resources
o – number of dedicated resources
Π – number of individuals in each generation

III. PRELIMINARIES

In this paper we use the concept of task graph G=(T,E) representation to describe behavior of an IoT network. Some elements of the network (especially embedded systems) execute tasks (T). Tasks are represented in the graph as nodes. Each edge $e_{ij} \in E$ represents amount of data that needs to be transferred between two connected tasks T_i and $T_j \in T$. Fig. 1 presents an example of a part of task graph.

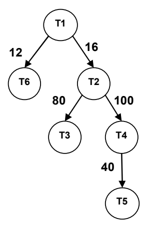

Fig. 1. Example of a task graph with inserted unexpedted tasks

Table I presents an example of a resource database for the part of the network described by the task graph form Fig.1.

TABLE I. RESOURCE DATABASE

	UR1 C=500		UR2 C=400		DR1		DR2	
	t	**c**	**t**	**c**	**t**	**c**	**t**	**c**
T1	150	7	125	5	30	150	20	350
T2	80	2	50	3	20	100	10	180
T3	300	22	350	19	100	300	80	450
T4	120	24	150	22	12	128	8	350
T5	230	32	200	28	55	45	63	40
T6	30	25	60	18	32	120	40	100
CR1, b=2	c=4		c=3		c=12			
CR2, b=10	C=5		c=4		c=18			

The example consists of 6 tasks: T1, T2, T3, T4, T5 and T6. Each task can be executed by a single resource. Resource can be a single element of the IoT or a part of an element. Tasks T6 T2, and T3, T4 are parallel and can be executed at the same time by different resources. Each CR has a bandwidth b which describes amount of data that can be transferred in a time unit. The tasks are characterized by two parameters: time and cost of its execution on each resource. DRs are fast but expensive. URs are cheaper but slower. The IoT designer must find the compromise between time and cost of the network.

The target IoT network must meet time constrains – all the tasks in the network must finish their execution before the maximum allowed time:

$$\forall_{e_{i,j}} ts_j \geq ts_i + t_{i,r_i} + tc_{i,j} \tag{1}$$

where:

$$tc_{i,j} = \left\lceil \frac{d_{i,j}}{b_{CR_{i,j}}} \right\rceil \tag{2}$$

where: $CR_{i,j}$ is communication resource which connects PRs executing tasks T_i and T_j. If T_i and T_j are executed on the same resource transmission time is equal to zero.

Every task presented in the graph from Fig. 1 can be divided into some number of subtasks.

Unexpected tasks appear when IoT network is ready and no new element cannot be added to the network. They can appear at any moment of the work of the network and must be inserted into task graph. If the target IoT network consists of m universal resources, p communication resources, the task graph contains n tasks and there are u unexpected tasks, the total cost of the network (C'_o) can be described by the following formula:

$$C'_o = \sum_{i=1}^{m} C_{UR_i} + \sum_{j=1}^{n} C_j + \sum_{k=1}^{p} \sum_{l=1}^{P_k} C_{CR_k,PC_l} + \sum_{s=1}^{u} C_l \tag{3}$$

where: CR_k is a type of k-th communication resource, which is connected to Pk processing resource, and PC_l is a type of l-th PR connected to CR_k.

IV. NEW PROBLEM STATEMRNT

The problem which we investigate in this paper is the concurrent optimization of two phases which impact each-other in real-time. The first phase is responsible for providing the parameters to optimize. The second phase optimizes the parameters given in the first phase and checks if the selection made in the first phase was correct. Only the verification in the second phase can give the answer if the selection made in the first phase can even give a result. If not, the optimized parameters must be changed. However, if optimizing parameters are changed the optimization in the second phase must also be changed or even repeated from the beginning. Therefore, it is not possible to optimize the phases separately. However, in case of different choice of optimizing parameters how can we know which choice is better? To solve this problem we must assume that there exists at least one common parameter for each of the solutions which allows to compare the results. Therefore, as it can be observed, the optimization problem is multicriterial:

$$f(x) = \min(f(x_1), f(x_2), ..., f(x_n)) \tag{4}$$

$$x_1 ... x_n \in S \tag{5}$$

where: S is a set of possible solutions.

For better presentation of the problem let us consider a simple example. Let us assume that we would like to pick up an apple from a tree. There are several possibilities to do that. We can climb the tree, shake the tree, use a ladder or a tool. Every choice gives us different parameters to optimize. If we choose to climb the tree we can optimize the path. Choosing to shake the tree makes us optimize a point of shaking and a force. However, the main question is how to decide which way is better without picking up the apple? Therefore, there must be at least one parameter which is common for every choice. In that case such a parameter can be, for example, a time of getting the apple. The problem can be split into two phases. First phase chooses the way of picking up the apple. The chosen way specifies the parameters to optimize. The second phase optimizes chosen parameters. However, during the second phase it can be found that it is impossible to pick the apple according to the way chosen in the first phase. What is very important, such information was not given at the beginning of the process. In the analysed case, such a situation can appear if the decision was to climb a tree and after some distance it was observed that it was impossible to continue climbing. The second phase modifies the first one. Modification of the first phase (changing a way of picking up an apple) gives different parameters to optimize and thus immediately changes the second phase. Another question is: should all the processes be repeated in such a case or maybe it is better to connect performed steps with another way of obtaining the apple?

We believe that such an easy example gives a good insight into the problem that is investigated in this paper. In the next chapter we define the optimization problem for the IoT network design process.

A. *Concurrent Real-Time Optimization in IoT Design Process*

We assume that during work IoT network encounters unexpected situations. Therefore, the network must execute some unexpected tasks. There is no upfront information about those tasks. The only information which is given is associated with the target (what kind of data is needed to be obtained). Therefore, to solve this problem all expected tasks must be split into possible number of subtasks. In this paper we concentrate on situation where unexpected tasks can be solved as connection of some subtasks of other tasks. The optimization is divided into two phases. The first phase is responsible for choosing the number and the types of subtasks. The second phase gives the parameters to optimize and searches for their optimal values by making tasks assignment. Typical goals of optimization can be the cost of the network, time of execution of all the tasks, power consumption, etc. The optimization process of both phases must be done concurrently. Otherwise, if the phases would be optimized sequentially, the first phase could only concentrate on the number of subtasks. That information is not necessarily important for IoT designer. Without a selection of the subtasks, task assignment cannot be made. The second phase also verifies the choice made in the first phase. If anything changes in one of the phases it will affect the other one in real-time. To solve such a problem we propose to use a genetic algorithm. The algorithm is described in the next section.

V. The Algorithm

During the work of IoT, unexpected situations are met. The only information which is given is the type of target data. First, in accordance with genetic rules, the population of random genotypes must be created. The number of generated individuals (Π) is controlled by α parameter and is equal to the formula:

$$\Pi = \alpha * s * v \qquad (6)$$

Where s is a number of possible subtasks and v is a number of possible types of PR in an IoT network.

The genotype is a string which contains numbers of the resources for each subtask chosen in the first optimization phase. For each genotype subtasks are chosen randomly. Therefore, selected subtasks may be different for every genotype. The resources are also chosen randomly.

We characterize the IoT network by two parameters: execution time and cost of the system. We decided to search for the cheapest solution which satisfies the time constrains. After creating a genotype the values of time and cost are calculated. As it was mentioned in previous chapters some generated solutions are invalid. It means that for some of them, desired values were not obtained. Therefore, connections of subtasks in those solutions are not valid too.

In this paper we decided to use standard genetic operators: crossover, mutation, cloning and ranked selection. Solutions are ranked by cost. At the top of the rank list valid solutions are placed. However, invalid solutions are not necessarily passed over. Because some of the genotypes have different number of genes after crossover operation it is possible to obtain solution that is not valid. The algorithm stops after ε generations without better solution.

VI. Experimental Results

According to our best knowledge this is the first paper which deals with concurrent real-time optimization in IoT design process. In table II we present experimental results for randomly generated graphs with 10, 20 30, 40 and 50 nodes. It is known from literature [18][19] that crossover operator in the genetic algorithm typically gives better solutions than mutation. Therefore in the experiments 70% of obtained individuals in each population are obtained by crossover, 20% by mutation and 10% by cloning. The rest of the parameters which control the evolution process were arbitrarily set for every experiment as: α=100, v=4 and ε=5.

TABLE II. The results

N	T_m	Best results			Worst results		
		time	*cost*	*gen*	*time*	*cost*	*gen*
10	300	284	1600	5	290	1978	10
20	400	396	2649	15	303	3255	16
30	1000	975	2842	22	960	2852	19
40	1600	1175	4632	42	1156	5527	47
50	1500	1422	4538	39	1401	5121	44

To simplify computations we assumed that all of the unexpected tasks appeared after all predicted tasks were executed. Therefore, the initial cost and time of execution predicted tasks is known.

In most of cases with decreasing cost of the task execution, time of execution of all the tasks is rising. It can be observed for graphs with 20, 30 40 and 50 nodes. For the graph with 20 nodes the best result with cost equal to 2649, the time of execution of all the tasks was equal to 396. In the worst case experiment the cost was equal to 3255 and the time of execution of all the tasks was equal to 303. For graph with 30 nodes solution with lower cost (2842) was characterized by time of execution of all the tasks equal to 975, meanwhile solution 15 time units faster (975) was 10 cost units cheaper (2852). For graphs with 40 and 50 nodes such difference was even greater. The cheapest solution for a graph with 40 nodes (4632) has time equal to 1422. Solution which is only 19 time units slower has cost equal to 5527 (almost 900 cost units more in comparison with the best result). The cheapest result for graph with 50 nodes (4538) was characterized by time value equal to 1422. The worst solution for graph with 50 nodes was 21 time units faster (1401) but almost 600 cost units more expensive (cost equal to 5121). Only for graph with 10 nodes the situation was different. In the best result not only cost of the system was lower (equal to 1600 for the best individual and 1978 for the worst solution) but also time of execution of all the tasks was lower (284 for the best result and 290 for the worst result). We suppose that such a situation could appear because of several reasons. According to formula (9), for graph with 10 nodes in each population 4000 individuals were generated. For other graphs much more results were created in every population: 8000 for graph with 20 nodes, 12000 for graph with 30 nodes, 16000 for graph with 40 nodes and 20000 for graph with 50 nodes. Because subtasks are chosen randomly it is possible that any solution of detected unexpected situation contains unnecessary subtask. After generating more results the chance of not existing unnecessary tasks is lower, but not equal to zero because of probabilistic nature of the algorithm. It can be also observed that the larger the task graph the more generations of individuals were created. This is because for graph with more tasks the search space is greater.

In table II we present the results of experiment in case when more individuals are created using mutation parameter and less using crossover. The number of individuals in each population and the stop condition is the same as in previous set of experiments. Therefore the parameters which control the evolution have the following values: $\alpha=100$, $v=4$, $\beta=0{,}2$, $\gamma=0{,}6$, $\delta=0{,}2$, $\varepsilon=5$.

VII. Conclusions

In this paper we present genetic algorithm to concurrent optimization in two phases which impact each-other in real time in IoT design process. The problem appears when IoT network must execute unexpected tasks and no information about the tasks is given. The algorithm is able to detect unexpected tasks and search for their better solutions. It is also able to eliminate connections of subtasks which do not give an acceptable solution.

Future work will concentrate on providing new algorithm to solve the presented problem. We will also concentrate on other aspects such as dependence on choice of constants and parameters for the problems of concurrent real-time optimisation and unexpected tasks in IoT design process. Another important issue is to connect presented method with other algorithms to obtain cheaper or faster IoT networks.

References

[1] C. R. Srinivasan, B. Rajesh , P. Saikalyan, K. Premsagar, and E. S. Yadav, "A review on the different types of internet of things (IoT)", Journal of Advanced Research in Dynamical and Control Systems, 11(1), 2019, pp. 154-158.

[2] A. Górski and M. Ogorzałek "Assignment of unexpected tasks in embedded system design process", Microprocessors and Microsystems, Vol. 44, 2016, pp. 17-21.

[3] A. Górski and M. Ogorzałek "Auto-detection and assignment of unexpected tasks in embedded systems design process", in proceedings of the 23rd International Workshop of the European Group for Intelligent Computing in Engineering, 2016, pp. 179-188.

[4] A. Górski and M. Ogorzałek "Assignment of unexpected tasks for a group of embedded systems", IFAC-PapersOnLine, vol. 51 issue 6, 2018, pp. 102-106.

[5] T. Perumal, E. Ramanujam, S. Suman, A. Sharma and H. Singhal, "Internet of Things Centric-Based Multiactivity Recognition in Smart Home Environment," in IEEE Internet of Things Journal, vol. 10, no. 2, pp. 1724-1732, 15 Jan.15, 2023, doi: 10.1109/JIOT.2022.3209970.

[6] I. Yaqoob, I. A. Targio Hashem, A. Ahmed, S.M. Ahsan Kazmi and C. S Hong, "Internet of things forensics: Recent advances, taxonomy, requirements, and open challenges", Future Generation Computer Systems 92, 2019, pp. 265–275.

[7] G. Pan, G. Qi, W. Zhang, S. Li, Z. Wu and L. T. Yang, "Trace analysis and mining for smart cities: issues, methods, and applications," in IEEE Communications Magazine, vol. 51, no. 6, pp. 120-126, June 2013, doi: 10.1109/MCOM.2013.6525604.

[8] G. Chen, W. Zou, W. Jing, W. Wei and R. Scherer, "Improving the Efficiency of the EMS-Based Smart City: A Novel Distributed Framework for Spatial Data," in IEEE Transactions on Industrial Informatics, vol. 19, no. 1, pp. 594-604, Jan. 2023, doi: 10.1109/TII.2022.3194056.

[9] S. Greengard, "Smart transportation networks drive gains". Communications of the ACM 58 (1), 2015, pp. 25–27.

[10] A. Al Hammadi, A. Al Zaabi, B. Al Marzooqi, S. Al Neyadi, Z. Al Hashmi, and M. Shatnawi, "Survey of IoT-Based Smart Home Approaches", Proceedings of the International conference on Advances in Science and Engineering Technology (ASET), 2019 pp. 1-6, doi: 10.1109/ICASET.2019.8714572.

[11] J. L. Hernández-Ramos, M. P. Pawlowski, A. J. Jara, A. F. Skarmeta and L. Ladid, "Toward a Lightweight Authentication and Authorization Framework for Smart Objects," in IEEE Journal on Selected Areas in Communications, vol. 33, no. 4, pp. 690-702, April 2015, doi: 10.1109/JSAC.2015.2393436.

[12] J. Jagannath, N. Polosky, A. Jagannath, F. Restucci and T. Melodia, "Machine learning for wireless communications in the Internet of Things: A comprehensive survey", Ad Hoc Networks 93 101913, 2019.

[13] Y. Li, Y. Zuo, H. Song and Z. Lv, "Deep Learning in Security of Internet of Things," in IEEE Internet of Things Journal, vol. 9, no. 22, pp. 22133-22146, 15 Nov.15, 2022, doi: 10.1109/JIOT.2021.3106898.

[14] K. Sha, W. Wei, T. A. Yang, Z. Wan and W. Shi, "On security challenges and open issues in Internet of Things". Future Generation Computer Systems 83, 2019, pp. 326–337.

[15] Z. A. Baig, S. Sanguanpong, S. N. Firdous, V. N. Vo, T. G. Nguyen and C. So-In, ''Averaged dependence estimators for DoS attack detection in IoT networks,'' Future Generation Computer Systems 102, 2020 pp. 198–209.

[16] A. Bąk, R. Czarnecki and S. Deniziak, "Synthesis of Real-Time Applications for Internet of Things," Lecture Notes in Computer Science, Vol. 7719, 2013, pp. 35-49.

[17] J. H. Holland, "Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence University of Michigan Press, Ann Arbor, MI (reprinted, MIT Press, Cambridge, MA), 1992.

[18] D. Zaharie, "Influence of crossover on the behavior of Differential Evolution", Applied Soft Computing, volume 9, issue 3, 2009 pp. 1126-1138.

[19] A. Sharma and M. Sinha, "Influence of crossover and mutation on the behavior of Genetic algorithms in Mobile Ad-hoc Networks," 2014 International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 2014, pp. 895-899, doi: 10.1109/IndiaCom.2014.6828092.