2022 11th International Conference on Modern Circuits and Systems Technologies (MOCAST) | 978-1-6654-6717-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/MOCAST54814.2022.9837579

An algorithm for node clustering targeting swarm of cyberphysical systems

Christos Sad and Kostas Siozios
Department of Physics, Aristotle University of Thessaloniki
Thessaloniki, Greece

***Abstract*—The increased number of nodes found in next-generation systems, such as the swarm of CyberPhysical nodes, impose new challenges related to their organization. Throughout this paper a novel algorithm aiming to address this problem, is introduced. The proposed solution relies on a public-available genetic algorithm. Experimental results highlight the superiority of introduced solution, as it achieves superior performance as compared to well-established relevant algorithms.**

***Index Terms*—Partitioning, Clustering, Graph, Meta-heuristic, Genetic algorithm**

I. Introduction

Recently, the convergence of emerging embedded computing, information technology, and distributed control became a key enabler for future technologies. Among others, a new generation of systems with integrated computational and physical capabilities that can interact with humans through many new modalities have been introduced. Such systems that bridge the cyber world of computing and communications with the physical world are referred to as Cyber-Physical Systems (CPS) [1]. Specifically, a CPS is a collection of task-oriented or dedicated systems that pool their resources and capabilities together to create a new, more complex system which offers more functionality and performance than simply the sum of the constituent sub-systems. Among others, these new design paradigms have the ability to interact with, and expand the capabilities of, the physical world through monitoring, computation, communication, coordination, and decision making mechanisms.

The integration of physical processes and computing is not new. Embedded systems have been in place for a long time and these systems often combine physical processes (e.g. through digital/analog sensors) with computing. However, the core differentiator between a CPS and either a typical control system, or an embedded system, is the communication feature among system's components, which adds (re-)configurability and scalability, allowing instrumenting the physical world with pervasive networks of sensor-rich embedded computation.

The large amount of these devices impose among others challenges related to their proper organization in order to guarantee optimal operation. Critical challenge towards this direction is given to novel algorithms that perform node clustering [2]. Throughout this paper we emphasize on CPS that consists of swarm of nodes that have to be appropriately clustered to multiple groups in order to perform different activities. This problem can be formulated as follows:

Problem formulation: Given a graph $G(V, E)$, where V and E denote the set of swarm nodes and their connectivity, respectively, the k-way clustering refers to the partition k disjoint subsets of the V $(V_1, V_2, \ldots, V_k)$, such as $\cup_i V_i = V$ by taking into consideration a balancing constrain [2].

Typically, the problem of cluster of swarm nodes has a lot of similarities with the balanced -– minimum cut size partitioning discussed throughout this paper. In this case, the emphasis is on computing k disjoint subsets of V, such us the sum of the node's weights $(W(V))$ between the subsets to be (almost) the same, and the sum of the edge's weights, which connect nodes from different parts (*edge-cut*) to be minimized.

Graph partitioning is an *NP*-hard problem and a lot of algorithms have been proposed. For instance, Kernighan–Lin (KL) and Fiduccia–Mattheyses (FM) algorithms for circuit partitioning [3] [4]. KL was first introduced for graphs with even number of nodes and only for bisection (2-way partitioning). After a lot of modifications, the algorithm was able to work for odd number of nodes and different partitioning ways(3-way and more),but the big asymptotic run time which is $O(n^3)$ (and after some improvements $O(n^2 logn)$), is its big limitation. FM algorithm is similar with KL but much faster. For FM the run time is $O(m)$, with m being the number of interconnections. The limitation of the FM is that if the graph(circuit) is very big, with a lot of interconnections the run time would also be very big. Simulated annealing (SA) algorithms were also applied to graph partitioning [5]. However, since these algorithms include multiple parameters, their efficient fine-tuning is a challenging task. Finally, multi-level partitioning algorithms are also applied. A well-known multilevel partitioning algorithm is hMetis [6] [7]. As it is depicted at experimental results section, although hMetis exhibits superior performance (in terms of min-cut), it cannot tackle efficiently the node balance metric among partitions.

Throughout this paper we propose an algorithm for node clustering targeting swarm of CPS. The proposed solution relies on a genetic algorithm in order to perform efficient design-space exploration. The proposed solution is open-source and publicly available to the GitHub for further research and experimentation. For evaluation purposes, the proposed algorithm was applied to cluster swarm of nodes ranging between 100 and 1000.

The rest of the paper is organized as follows. Section II describes the proposed framework for node clustering based on genetic algorithm. Experimental results that highlight the

efficiency of proposed solution are provided at Section III. Finally, Section IV concludes the paper.

II. Proposed Genetic Algorithm

Genetic algorithms(GAs) were first introduced by John Holland and his associates in the 1975. They are classified in the category of meta-heuristic algorithms and are mainly used for optimization problems. There are three main parts of a genetic algorithm, which are namely, "selection"," crossover" and "mutation". Based on the adaption operation, they start from random solutions and mixing them in a random manner, new generations are created. Every new generation is more likely to be better in the sense of a fitness function, which controls the whole operation [8]. One of the main advantages, of a genetic algorithm, is the ability to run in parallel. Also, another important point is the use of random choice operators for the search space exploration and so also the less possibility to trap in a local minimum.

A. Algorithm Description

The algorithm developed in this paper, is a GA, following the basic characteristics of these algorithms and using the main parts of them (selection, crossover, mutation). Also, a parallelism method is adopted, offering speedup and so the ability to solve more complex problems.

The procedure which is used is the following. First of all, a random population of solutions is created to construct the first generation. Then, follows the loop-core of the GA, where if none of the finish criterium is verified, new generation is created (the new generation creation procedure will be described next). Then, based on the fitness function, the best chromosome of the current generation is computed, also the percentage of improvement for the fitness function of the last generation's best chromosome and the current generation's best chromosome is computed and finally the overall best chromosome of all the generations (until now) is computed. After exiting the loop-core, the overall best chromosome is returned as the final solution of the problem. The pseudocode of the GA can be seen in Algorithm 1.

Algorithm 1 Genetic Algorithm

```
create first chromosome population
create Gene-0 with the first population
while finishcrit ≠ 1 do
    create new gene
    find best chromo of current gene
    find percentage of improvment
    find current best chromo
end while
return current best chromo
```

The next generation creation procedure which is used is the following. First of all, the chromosomes which will be used as parents, are selected. For this process, a couple of chromosomes is randomly selected and then from these, the one with the worst fitness function is stored in the parents' array. If, for example, the number of mixtures that will take place is set to k, a total number of $2\times k$ parents is needed, so this process is repeated $2\times k$. Selecting the worst, from the two candidate parents, gives the opportunity to the better one, to survive, as it is to the next generation. After having finished with the selection process, the parents' array gets a random shuffle. The couples from the parents' array, then are selected successively, based to their random position in the array. After this, a crossover procedure is getting done between each couple from the parents' array. In this process, the one-point crossover operation takes places, for each couple, with a probability th (for the experimentes 65%) or the couple survives as it, with probability $1-th(35\%)$. The whole process can be seen in Algorithm 2.

Algorithm 2 New generation's creation with k mixtures

```
# k:const number of mixtures
for i = 1, ..., 2k do
    candidate 1=random(chromos)
    candidate 2=random(chromos)
    if costfunction(candidate 1) > costfunction(candidate 2)
    then
        parentsarray(i) = candidate1
    else
        parentsarray(i) = candidate2
    end if
end for
random shuffle (parentsarray)
for i = 1, ..., 2 × k , step=2 do
    a = random(0, 1)
    if a < threshold then
        point = (int)random(0, nodes) #crossover point
        one-point-crossover-operation
        {parentsarray(i),parentsarray(i+1),point}
    end if
end for
return parentsarray
```

The one-point crossover is the procedure of mixing the two parents-chromosomes. First of all, a random number from 1 to maximum length of chromosome is created, which declares the point of the chromosome, based on which, the process will take place. After this, every point of the parent1 until the crossover-point migrates to parent2 and the opposite. Then, the two offsprings are created.

In the previous two algorithms, the chromo (chromosome) is the class that presents a solution of the problem. One variable of this class is an array that contains the part in which belongs every node of the hypergraph. Also, Gene is another class that contains a population of chromos (solutions).

Finally, it has to be mentioned that every chromosome can be selected to participate in the selection tournament more than once, so it can be noticed in more than one couples. In this way the diversity of the species is ensured and so a trap to a local optimum solution is avoided. The above, ability is used, instead of "mutation" procedure, where a little percentage of

the population, is selected to slightly change, to avoid a local optimum solution.

B. Termination Conditions

The finish criteria, are based on the fitness function or the number of generations and describe situations that terminate the GA's execution. More specifically the following criteria are used at the proposed GA:

- maximum number of generations that will be created. It is a predefined constant and in case that none of the other criteria is verified, it is activated to avoid getting stuck in an infinite loop.
- threshold for the fitness function, if the fitness function of the overall best chromosome is better or equal to the threshold the algorithm terminates.
- number of generations that passed without remarkable improvement between best chromos of successive genes. If the number of genes is more than a predefined maximum value and the improvement, every time, very poor the GA terminates.
- number of generations that passed from the generation that the overall best chromosome (until now) was created. If no actual optimization is achieved for a number of generations the algorithm finishes its execution returning as output the over all best solution until then.

All the above criteria were used in the experiments for the GA termination.

This criterion create some predefined constant parameters. Selecting the values of this parameters, regulates the approximation between time of execution, quality of the solution and resources. For example more strict criterion about the quality of the solution may lead to bigger run-times or more necessity of resources(e.g. memory, cpu cores).

C. Parallelism

The parallelism strategy which is adopted in this paper is the Single population master-slave GA. In this model, a unique population of chromosomes exists in the master-main thread. The processing of the chromosomes is implemented at the slaves-workers threads in a parallel way. Specifically, the processes that are paralleled are the following.Firstly, for any chromosome the computation of its fitness function is being paralleled. The fitness function must be a simple metric, ease to compute and its time of computation depends on the length of the chromosome. So a parallelism on its computation process will speed up the GA. Then, the selection procedure inside the generation is also paralleled. The chromosome couples with the candidate parents are treated in parallel and not serial. This is able because the selection tournament contains only computations of fitness functions. Finally, the one-point crossover procedure is paralleled, so that the crossover of every couple is processed in parallel way.

III. Experimental results

In this section the experiments in which the genetic algorithm (GA) was tested and evaluated are described. We tested the GA on partitioning some graphs that correspond to different organizations of swarm nodes. The reference solution for this analysis were computed based on the hMetis partitioning algorithm. For demonstration purposes, graphs of swarm nodes were randomly created(random nodes' and edges' weights). The number of nodes that are included to these graphs span from 100 up to 1,000 nodes. Four different scenarios were considered for these experiments, which refer to the connectivity demand among nodes. For this purpose, we explore cases with connectivity among nodes that range between 25% and 100% (fully connected graph). In addition to that, we also explore the efficiency of proposed GA algorithm to calculate the 2-way, 3-way, ..., 10-way partitioning. For sake of completeness, the results reported at this section were retrieved as an average value from 10 runs of the proposed and hMetis algorithms. For demonstration purposes, all the results are plotted in normalized manner over the reference solution (hMetis).

Next, we describe in detail the metrics employed for the evaluation procedure. First of all, the solutions were evaluated based on the balance metric. For this propose we used the *imbalance* metric formulated as follows:

$$\begin{aligned} imbalance = \sum |W(V_i) - \frac{W(V)}{k}|, \forall i = 1, ..., k \\ W(V_i) : sum - of - nodes' - weights - from - part - i \\ W(V) : sum - of - all - nodes' - weights \\ k : number - of - parts \end{aligned} \tag{1}$$

Next, we consider the *edge-cut* metric for the computed solutions (partitions), based on the following equation:

$$\begin{aligned} edgecut = \sum W(e_{i,j}) \times (1 - part(i) == part(j)), \\ \forall i, j = 1, \ldots, nodes, i \neq j \end{aligned} \tag{2}$$

In the previous equation $part(i) == part(j)$ equals to 1, if $part(i) = part(j)$ and 0, if $part(i) \neq part(j)$. Edge-cut describes the sum of edges' weights, connecting nodes of different parts.

Regarding the previously mentioned metrics, both of them were considered at the fitness function as a weighted sum. Namely, next equation depicts the employed fitness function:

$$fitnessfunction = a \times imbalance + (1 - a) \times edgecut \tag{3}$$

The value of parameter a affects the result of the GA, as it controls the importance of balanced and minimum edgecut, respectively. Values, near 1 focus on the imbalance metric and values near 0, give importance to the minimum edgecut target. For the experiments, the parameter was set at 0.6.

Figure 1 depicts the imbalance metric among partitions for the studied hypergraphs. Different curves at this figure correspond to different way of partitioning ranging from 2-way up to 10-way). Vertical axis at this figure gives the ratio $\frac{imbalance_{GA}}{imbalance_{hMetis}}$. According to this analysis, the ratio

gradually decreases. This imposes that as the number of the nodes increases, the GA retrieves more balanced solutions than the corresponding hMetis partitions. Also, regarding solutions with less than 6 partitions, the GA always creates more balanced solutions (since the imbalance ratio criterion is less than 1). Moreover, in case we have to partition more than 250–300 nodes, then the proposed GA algorithm results to more balanced solutions. Note that in case we compute 2-way or 3-way partitions, the curves at this figure is almost flat (and near to 0), which means that the GA creates much more balanced solutions than the hMetis. In the worst case GA was 3.9× more imbalance than the hmetis and in the best case the ratio (GA/Metis) was 0.0082.

Figure 2 gives the ratio $\frac{edgecut_{GA}}{edgecut_{hMetis}}$ for the studied partitions. According to the experimentation, the GA's solutions are similar to the hMetis solutions in the sense of edge-cut. In worst case the edge-cut ratio (GA/hMetis) is 1.12 and in the best case is 0.83.

Taking into account both imbalance and edge-cut, GA's solutions are in most cases, more balanced and have a little bigger edge-cut. The approximation between minimum edge-cut and balanced partitioning is better in the case of GA. This, can be noticed if we multiply the imbalance and edge-cut ratio metrics. The edge-cut ratio is near 1 and so the multiplications are almost near imbalance ratio's values and the graphic follows the same pattern as imbalance's. This means that the multiplications are most time less than 1, with the worst case being 3.3 and the best one 0.0088.

We have to mention here, that about the run time of the algorithms, hMetis always finished the execution very quickly, in contrast with the GA, whose time increased as the number of nodes increased. For example in worst case the hMetis was 35.76× faster than the GA and in the best case hMetis was 1.1× faster than the GA. Mentioning that, the worst cases refer to small graphs, where hmetis' run time was less than $1sec$ it is clear that run time of GA was comparable with the very fast hMetis and never exploded.

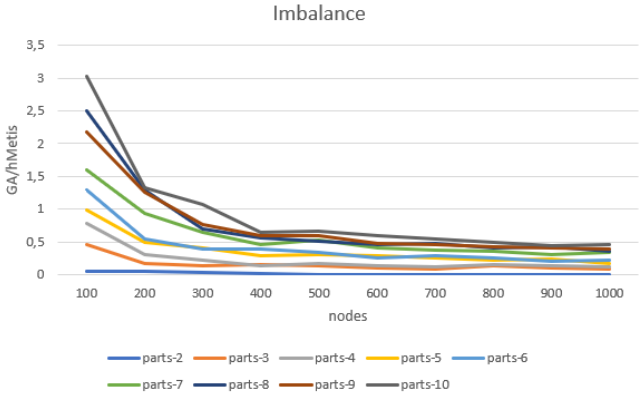

Fig. 1: Imbalance metric for the fully connected hypergraph

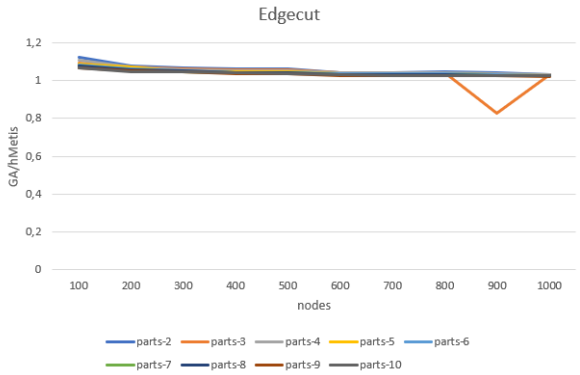

Fig. 2: Edge-cut metric for the fully connected hypergraph

IV. Conclusions

A novel framework based on genetic algorithm in order to solve the clustering of swarm nodes into balanced groups of nodes, was introduced. Experimental results highlight the superiority of introduced solution, as it achieves superior performance as compared to well-established hMetis algorithm. Finally, the proposed open-source solution is publicly available at Github for further experimentation from interested readers.

Acknowledgment

This research has been co-financed by the European Regional Development Fund of the European Union and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the call RESEARCH – CREATE – INNOVATE (project code: T2EDK-01681).

References

[1] C. Lu et al., "Real-Time Wireless Sensor-Actuator Networks for Industrial Cyber-Physical Systems," in Proceedings of the IEEE, vol. 104, no. 5, pp. 1013-1024, May 2016.

[2] V. Pal, G. Singh and R. P. Yadav, "Balanced Cluster Size Solution to Extend Lifetime of Wireless Sensor Networks," in IEEE Internet of Things Journal, vol. 2, no. 5, pp. 399-401, Oct. 2015, doi: 10.1109/JIOT.2015.2408115.

[3] B. W. Kernighan, S. Lin, "An efficient heuristic procedure for partitioning graphs," in The Bell System Technical Journal , Volume: 49, Issue: 2, pp. 291 - 307, Feb. 1970.

[4] C. M. Fiduccia, R. M. Mattheyses "A linear-time heuristic for improving network partitions,"DAC '82: Proceedings of the 19th Design Automation Conference , pp. 175–181 , 1982

[5] G. M. Slota, C. Root, K. Devine, K. Madduri and S. Rajamanickam, "Scalable, Multi-Constraint, Complex-Objective Graph Partitioning," in IEEE Transactions on Parallel and Distributed Systems, vol. 31, no. 12, pp. 2789-2801, 1 Dec. 2020, doi: 10.1109/TPDS.2020.3002150.

[6] Selvakkumaran, N. and Karypis, G, "Multi-objective hypergraph partitioning algorithms for cut and maximum subdomain degree minimization," ICCAD-2003. International Conference on Computer Aided Design (IEEE Cat. No.03CH37486),pp.726–733.

[7] Cristinei Ababel, Navaratnosothie Selvakkumaran, Kia Bazargan, and George Karypis,"Multi-objective Circuit Partitioning for Cutsize and Path-based Delay Minimization" , IEEE/ACM International Conference on Computer Aided Design (ICCAD), pp. 181 - 185, 2002

[8] K. Maragos, K. Siozios and D. Soudris, "An Evolutionary Algorithm for Netlist Partitioning Targeting 3-D FPGAs," in IEEE Embedded Systems Letters, vol. 7, no. 4, pp. 117-120, Dec. 2015, doi: 10.1109/LES.2015.2482902.