Constanta Maritime University, Faculty of Navigation and Naval Transport

Load balancing in parallel computing: an evolutionary approach

Constanta, August, 2022 **Simona DINU** **Gabriel RAICU** **ATOM –N 2022 Conference**

Introduction

► **k-way graph balanced partitioning problem:** balance computation between the k processors of a parallel architecture, while reducing interprocessor communications volume.

► **formalize these objectives:** the problem is shaped into a graph representation, where the nodes correspond to computational sequences (tasks) and the edges correspond to data dependencies between them (that is, the input of a computation is from the output of another one). The nodes of the graph need to be partitioned into k subsets with approximately the same number of edges in each of them. In addition, the set of cut edges between partitions (i.e. edges connecting nodes in two different partitions) must be minimized.

► **applicability of the problem:** scientific computing, sparse matrix decompositions, very large-scale integration (VLSI) circuits design, image processing and pattern recognition, cluster analysis in data mining; is also involved in performing scalable data analysis of large systems with complex topologies. For example, interdependencies and interactions of modern critical infrastructure systems (power, communication, road and transportation networks) can be modelled as networks.

Goals & Approaches

► **Major concerns:** advanced increase in computing power demand requires increased computational performances/acceleration of intensive computations, which can be achieved by implementing parallel processing approaches. An evenly distribution of the computational workload calculations among the available processors of parallel computers proved to be the key element in achieving these goals. Consequently, it is very important to investigate how to optimize parallel processing of tasks in these systems.

► **Proposed model:** Given a multi-core system, where *k* is the number of processor cores, and *n* is the number of tasks, the computational workload balancing problem is formulated equivalently, as a *k*-way balanced partitioning problem: decomposing the corresponding graph model of computations and communications into *k* densely-connected components, while minimizing the number of edge cuts between the components.

Problem formulation

G = (V,E), where *V* is the set of nodes, $|V| = n$ and $E \subset ExE$ is the set of edges, the goal is to partition V into disjoint blocks of vertices $V_1, V_2, \ldots, V_k$, $|V_j| = n_j$ such that:

$V_1 \cap V_2 \cap \ldots \cap V_k = \emptyset$ and $V_1 \cup V_2 \cup \ldots \cup V_k = V$, i.e.

$$\sum_{j=1}^{k} n_j = n$$

.

Also, the following restrictions must be fulfilled:

- the connectivity constraint:

$$G_i = (V_i, E_i) \text{ is a connected sub-graph}, i = 1,\ldots,k$$

- the balancing constraint:

$$|V_i| \le \frac{(1+\varepsilon)\cdot n}{k},\ i = 1,\ldots,k$$

where ε is the imbalance parameter, a very small positive constant.

- the minimum cut constraint:

$$Min \sum_{i=1}^{k} \sum_{j=i+1}^{k} |\{e \in E \,|(e \cap V_i \ne \phi) and (e \cap V_j \ne \phi)\}|$$

The equivalent combinatorial optimization problem

The n x n matrix A(a_{ij}) is called the adjacency matrix for G, where:

$$a_{ij} = \begin{cases} 1, if\,(v_i, v_j) \in E \\ 0, otherwise \end{cases}$$

$x_j = [x_{1j}, x_{2j}, \ldots, x_{nj}]^T$, where: $x_{ij} = \begin{cases} 1, \text{if node } i \text{ is assigned to partition } V_j \\ 0, otherwise \end{cases}$

- **Decision variables:**

$$X_{nxk} = \{(x_{ij})\} \quad i \in \{1,\ldots n\}, j \in \{1,\ldots k\}$$

- **Objective function:**

Maximize the number of edges with both ends in the same partition:

$$Max \sum_{j=1}^{k} \sum_{r=1}^{n} \sum_{s=1}^{n} a_{rs} x_{rj} x_{sj}$$

- **Constraints:**

1) There are n_j nodes in partition V_j:

$$\sum_{i=1}^{n} x_{ij} = n_j \quad \forall j = 1,\ldots,k$$

2) Since $V_i \cap V_j = \emptyset$:

$$\sum_{j=1}^{k} x_{ij} = 1 \quad \forall i = 1,\ldots,n$$

3) By definition, x_{ij} is either 0 or 1

$$x_{ij} \in \{0,1\} \quad \forall i = 1,\ldots,n, \quad j = 1,\ldots,k$$

Problem solution

Proposed fuzzy adaptive Genetic Algorithm (GA) optimization: a solution based on canonical GA with fuzzy adaptation of parameters.

► Genetic Algorithms do not make any assumption about the objective function, they are very adaptable in dealing with any type of objective function and have generally proved to be good heuristics for solving complex combinatorial optimization problems.

► In this study, the effective settings for GA control parameters (population size, maximum number of generations, mutation rate) were determined empirically, analyzing each instance separately and for the crossover rate, the initial value was set at 0.80.

Crossover rate—fuzzy controller

► This parameter is responsible for ensuring a good balance in evolutionary search, between global exploration of the search space (global diversification) and local intense exploitation (around already determined good solutions). In this sense, an optimal approach is a self-adaptive control of this parameter, based on the mechanism of fuzzy logic.

► The entries in FLC are two variables that express the current state of the genetic search in terms of population diversity, both at the genotype level (GD – Genotype Diversity) and at the phenotype level (PD - Phenotype Diversity):

two input variables: **one output variable:**

$$GD = \frac{\bar{d} - d_{\min}}{d_{\max} - d_{\min}} \qquad PD = \frac{fitness_{\max} - \overline{fitness}}{fitness_{\max} - fitness_{\min}}$$

crossover rate

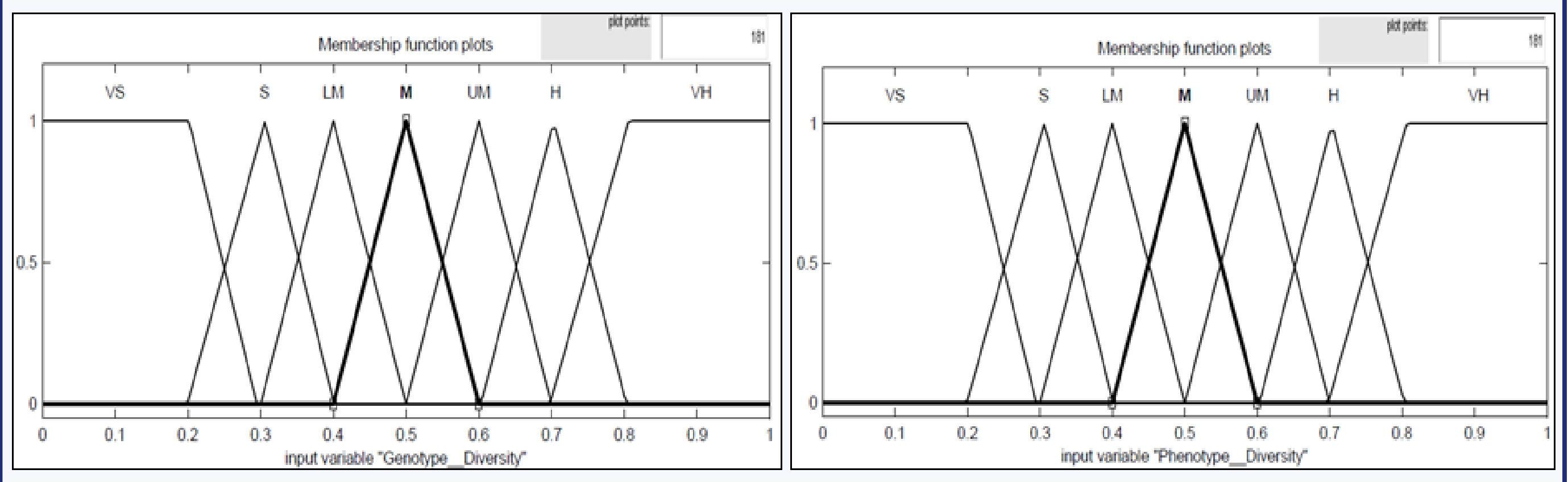

Figure 1: Membership functions for the input variables GD and PD

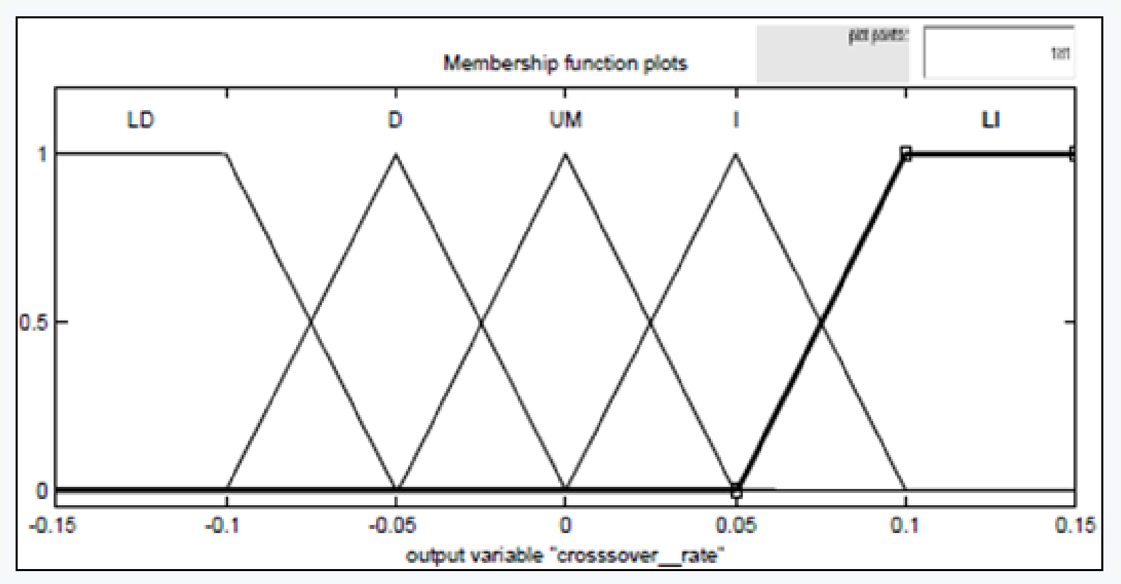

Figure 2: Membership functions for the output value: crossover rate change

Table 1: The inference table for crossover rate change

GD FD	VS	S	LM	M	UM	H	VH
VS	LI	LI	LI	I	I	UM	UM
S	LI	I	I	UM	UM	UM	D
LM	LI	I	I	UM	UM	UM	D
M	I	UM	UM	UM	D	D	D
UM	I	UM	UM	D	D	D	LD
H	I	UM	UM	D	D	D	LD
VH	UM	UM	D	D	LD	LD	LD

Simulation results

For clarity of visualization, we have chosen to present the simulation results obtained for a graph structure with |V| = 33 nodes and |E| = 75 edges, with the structure shown in the Figure 3 a). The graph partitioned into 3 clusters, with a minimum of 3 cuts is presented in Figure 3 b).

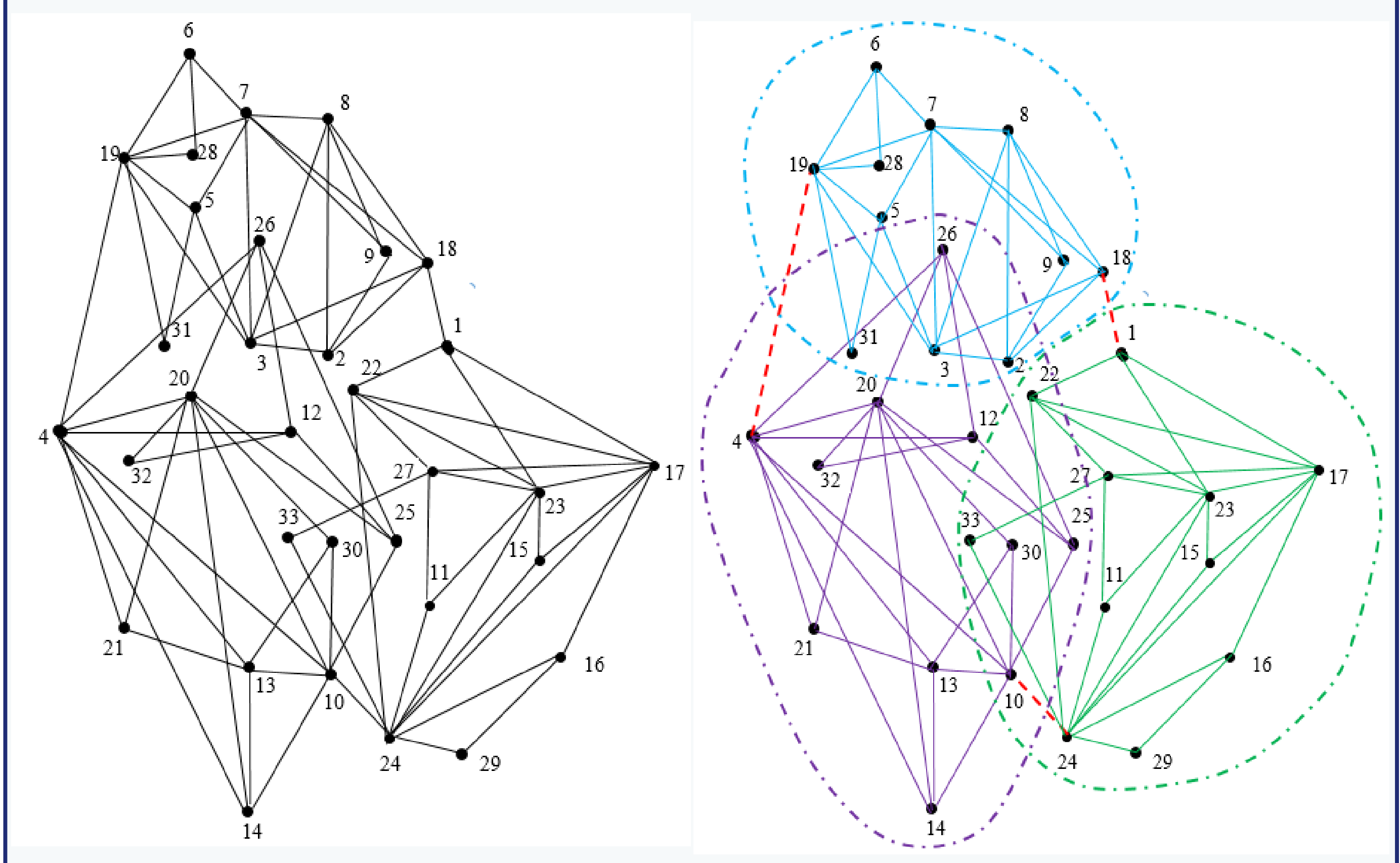

Figure 3

a) Graphical representation of the instance with 33 nodes

b) Result of partitioning the graph into 3 clusters

Conclusions and future work

►The computational experiments on all data sets showed good performance of the proposed algorithm.

►The algorithm is easy to design and implement and is tractable for practical problems with a large number of tasks and processors

►An important feature of the algorithm is that it can be used with minor modifications for uneven partitions sizes, which characterizes the problems of optimal allocation of computations to heterogeneous processors, with different computing powers.