Parallel and Distributed Software Assessment in Multi-Attribute Decision Making Paradigm

Marin ANDREICA[1], Cornel RESTEANU[2], Romica TRANDAFIR[3]

[1] Economic Studies Academy, Bucharest, Romania,
mandreica@yahoo.com

[2] National Institute for R&D in Informatics, Bucharest, Romania,
resteanu@ici.ro

[3] Technical University of Civil Engineering, Bucharest, Romania,
romica@utcb.ro

Abstract: Multi-Attribute Decision Making (MADM) theory is a way to obtain good quality assessment for parallel and distributed software. It provides adequate tools to compute a synthetic characterization, named High Performance Computing (HPC) merit, which may be used in operations like software comparisons / rankings / optimizations. The paper presents the general assessment model with its associated assessment problems and a terse and telling case study. The assessment model is described and solved by the Internet mathematical service named OPTCHOICE (MADM modeling and optimal choice problem solving). It provides a multitude of normalization and solving methods generating diverse assessments, but always a global assessment is delivered.

Keywords: Software Assessment, Parallel and Distributed Computing, Multiple Attribute Decision Making, Comparisons / Rankings / Optimizations through the agency of the High Performance Computing Merits.

1. Introduction

At the beginning, the unique method to assess the characteristics of any kind of parallel and distributed software [1, 2, 3] was the direct testing. Through the agency of this method, only few characteristics used to be assessed. Therefore, the benchmark must intervene. Primarily, even the benchmarking process has been mainly a manual process. In order to allow this time-consuming and costly analysis process to be automated, a lot of techniques working on the performance indicators were developed [4]. Moreover, general purpose software were developed, the more prominent are PARSEC and SPLASH 2. The Princeton Application Repository for Shared-Memory Computers (PARSEC) is a benchmark suite, representative for next-generation shared-memory programs for chip-multiprocessors, meant to analyze emerging workloads in their complexity (see http://parsec.cs.princeton.edu). The SPLASH 2 benchmark belongs to the Computer Architecture and Parallel System Laboratory (CAPSL) from Delaware University, (see http://www.capsl.udel.edu/splash) and implements modern parallel computation models to study the future generations of high-performance computing systems. The diversity of benchmarking techniques is enriching every day [5, 6] but still indicators which remain to be computed by real testing or by expert assessment.

The paper proposes a method to globally assess the parallel and distributed software by computing a so called *HPC merit.* This is computed starting from the elementary characteristics of software evaluated by direct testing, benchmarking and experts. The elementary characteristics refer both to source and executable formats. The HPC merit's computing procedure may be considered as an integration of elementary characteristics to give a synthetic characterization. This shows if the respective program is well realized as parallel and distributed software and well distributed on hardware configuration. It is a number in the [0, 1] interval. As close to 1 is the HPC merit, the better realized is the software - hardware implementation. In principle, every single program of the parallel and distributed software class may be globally assessed. But the main goal is assessing a set of programs with the same functionality. In this case, the merits can stay at the base of comparison / ranking / optimization problems.

In order to make a good assessment for the parallel and distributed software, it is necessary:

- To consider a parallel and distributed programs set and its characteristics. The programs must belong to the same class, meaning that they realize the same user function but by different software solutions;
- To consider, for the programs set, all hardware configurations capable to receive them in running. It is possible to operate on a collection of computing elements (scalar machines, multiprocessors, or special-purpose computers) interconnected in one or more homogeneous / heterogeneous networks;
- To imply, in the assessment process, several experts which are specialized in computer science, mathematics and programming languages.

Thus, the above specifications lead to the conviction that the MADM [7] paradigm is suitable to use in the construction of assessment models and solving the pending problems. Indeed, in this case, it is possible to define the following entities: objects (software set subject to the assessment process), attributes (software's general and parallelism / distributive elementary characteristics), states of nature (running platforms taken in consideration) [8, 9], experts / decision makers [10] (specialists in algorithms, programming and networking), objects - attributes characteristics matrix with the dimensions determined by the above entities dimensions, and finally, decision makers / states of nature / attributes weights (meaning that the elements in this entities have different importance in the assessment process). Obviously, in this manner, it was considered neither more nor less than a generalized MADM model.

The parallel and distributed software's assessment is made using a tool named OPTCHOICE [11]. It may be characterized as a pervasive Internet optimization service. An Internet service is pervasive if it is available, in conditions of performance and without delay, to anyone, from any place, at any time and free of charge. Being capable to treat generalized MADM models and being pervasive, it was the best solution for defining and solving parallel and distributed software assessment problems.

In the following, the paper theoretically shows how can define Assessment Models (AMs) in MADM paradigm, how can generate associated Assessment Problems (APs), and practically shows how is possible to handle AMs and APs in the context of a case study. The paper ends with some conclusions.

2. AMs Defining and APs Solving

In the following one presents, in mathematical notations, the general AM defining and, in pseudo-code the AP solving. One supposes that there exists one beneficiary which possesses a number of programs with the same functionality, written in the requirements of the parallel and distributed computing, and susceptible to be exploited on various hardware platform. He / she wants to know the merit of every program and consequently which of them is the optimum and therefore will be chosen to be utilized. A demand to accomplish this task is addressed to a number of specialists in such problems. They must have excellence in algorithms, programming languages and networking.

2.1 AMs defining

By definition, a general AM, in *OPTCHOICE* vision which implements the *MADM*, involves the following elements:

- $D=\left\{d(l)\,|\,l=\overline{1,\mathbf{l}}\right\}$, (**l**=card($D$)), a set of experts whose elements are the persons with assignments in the process of building and validating the AMs, as well as in generating and solving the APs. Typically, the experts discuss and agree on their absolute weights (giving their relative importance in the assessment process) $WD=\{wd(l)\,|\,l=\overline{1,\mathbf{l}}\}$, $\sum_{l=1}^{\mathbf{l}} wd(l)=1$;
- $S=\left\{s(k)\,|\,k=\overline{1,\mathbf{k}}\right\}$, (**k**=card(S)) a set of states of nature, each one of them representing a hardware platform susceptible to run the programs subject to the assessment process. Like in the above case, there are the absolute weights $WS=\left\{ws(k)\,|\,k=\overline{1,\mathbf{k}}\right\}$, $\sum_{k=1}^{\mathbf{k}} ws(k)=1$. In order to assure the impartiality, in the software assessment it is recommended that equal absolute weights for the hardware platforms be granted;

– $O=\left\{o(i)\,|\,i=\overline{1,\mathbf{i}}\right\}$, (**i**=card($O$)) the objects, a discrete and finite set with at least one element, representing the parallel and distributed programs subject to the assessment process;

– $A=\left\{a(j)\,|\,j=\overline{1,\mathbf{j}}\right\}$, (**j**=card($A$)) the attributes, a discrete and finite set of mutual independent elements with at least one element, with its absolute weights $WA=\left\{wa(j)\,|\,j=\overline{1,\mathbf{j}}\right\}$, $\sum_{j=1}^{\mathbf{j}} wa(j)=1$. They represent those parallelism and distribution characteristics which can be established for all objects in the same time. One may defined a lot of software's parallelism and distributive characteristics, over one hundred of them, which can be considered attributes in AMs. In the following, the most important of them, taken into account in this paper, are grouped in three sections:

– *Fundamentals*. The first section contains the general characteristics of the software to be assessed:
  - a (1) = quality of the algorithm chosen for solving the given problem (by comparison with the best possible solving algorithms) and of the chosen programming language (of general use and / or special use),
  - a(2) = quality of the general parallelization solution,
  - a(3) = quality of the general distribution solution,
  - a(4) = developing cost;

– *Parallelism and distribution.* The second section contains the characteristics regarding the parallelism and the distribution of the software [12, 13]:
  - a(5) = complexity of the parallelization and distributing process,
  - a(6) = parallelization model,
  - a(7) = quality of functional decomposition,
  - a(8) = on functions dependency magnitude,
  - a(9) = number of parallelism inhibitors unsolved in the program,
  - a(10) = number of intensive computing places persisting in the program,
  - a(11) = number of bottle neck persisting in the program,
  - a(12) = data decomposition model,
  - a(13) = quality of data decomposition,
  - a(14) = on data dependency magnitude,
  - a(15) = synchronous communications magnitude,
  - a(16) = asynchronous communications magnitude,
  - a(17) = communications latency,
  - a(18) = communications efficiency,
  - a(19) = surplus time,
  - a(20) = communications time,
  - a(21) = I/O time,
  - a(22) = dead time,
  - a(23) = balanced loading,
  - a(24) = granularity,
  - a(25) = scalability,
  - a(26) = quality of memory charring,
  - a(27) = quality of buffer in/out mechanism,
  - a(28) = type of memory access,
  - a(29) = memory-cpu bus bandwidth,
  - a(30) = communications network bandwidth,
  - a(31) = massive parallelism;

– *Efficiency*. The third section contains the characteristics that the user follows in the current running:
  - a(32) = running time,
  - a(33) = speed increasing,
  - a(34) = used memory,
  - a(35) = exploitation cost,
  - a(36) = reliability,
  - a(37) = portability [14].

These characteristics can have diverse expression modes: cardinal, ordinal, Boolean, fuzzy and random variables. In this case is preferred to utilize a variant of cardinal mode i.e. the grades from 1 to 10. The attributes have also a double vector giving their variation intervals and an optimization senses vector:

$LUA=\{(la(j),\ \ wa(j))|\ \ j=\overline{1,\mathbf{j}}\}$, whose elements are the variation intervals of attributes. The intervals are generically denoted by (L, U). In this case $1\le la(j)\le ua(j)\le 10,\ \ j=\overline{1,\mathbf{j}}$;

$SA = \{sa(j) | j = \overline{1,\mathbf{j}}\}$, whose elements are 0 (meaning *minimum*, denoted by *m*) when the attributes are considered good for the smaller possible values and 1 (meaning *maximum*, denoted by *M*) when the attributes are considered good for the larger possible values. In this case, the characteristics' evaluation being made by grades, the sense of optimization will be *maximum*, then M=1 for all attributes;

– The characteristics matrix $C : O \times A \times S \times D \rightarrow \Re$, where the element $c(o(i), a(j), s(k), d(l)) = oasd_{ijkl}$ represents the value of the attribute $a(j)$ for the object $o(i)$, in the opinion of the expert $d(k)$, in the state of nature $s(l)$, with $i = \overline{1,\mathbf{i}}$, $j = \overline{1,\mathbf{j}}$, $k = \overline{1,\mathbf{k}}$, $l = \overline{1,\mathbf{l}}$.

Final remark: The weights, in user expression, are given in percentages. All weights are normalized by the *OPTCHOICE* software and thus their mathematical expression becomes transparent for users.

2.2 APs solving

An AP may be solved by a set of *MADM* methods, $SM = \{sm(m) | \mathrm{m} = \overline{1,\mathbf{m}}\}$, **m**=card(*SM*), namely the *analyse of objects' dominance* (6 different analysis), and the c*omputing objects' merits* (maximax, maximin, linear utility function, scores, diameters, Onicescu, Pareto, TOPSIS, TODIM methods belonging to the American school) in conjunction with several normalization methods, $NM = \{nm(n) | n = \overline{1,\mathbf{n}}\}$, **n**=card(*NM*) (for example: *von Newman – Morgenstern like methods*). *MADM* domain contains also methods belonging to the French school. The latter can be seen in [15]. The APs, which are generated over an AM, by varying the input parameters, can be solved sequentially by more than one couple *sm(m) – nm (n)*. Since each method reflects a different point of view about assessment and optimality, it is obvious that applying different methods to the same set of data will often lead to different solutions. Therefore, a decisional inconsistency may appear. A procedure implemented in *OPTCHOICE* addresses this problem; it proposes a global solution by processing the results stored in a so called evaluation vector.

In the following, one presents, in an adequate pseudo-code, the AP solving:

PROGRAM implementing the *OPTCHOICE* algorithm

```
SELECT from OPTCHOICE-DB an AP
UPLOAD from database to memory the AP data
MEMORY DATA
Nature and dimensions of parameters, variables, vectors
 and matrices.
TEXT OF 256 CHARACTERS o(i), a(j), d(k), s(l), sm(m), nm(n)
PROCEDURE NAME OF 8 CHARACTERS
CURRENT_METHOD, SOLVING
BOOLEAN sa(j), bsm(m)
INTEGER PARAMETERS i, j, k, l, m, n, ii, jkl=max(j, k, l)
INTEGER VARIABLES i, j, k, l, m, n, ii, jkl
REAL  la(j), ua(j), wa(j), ws(k), wd(l), ww(jkl), oasd(i, j, k, l),
osd(i, k, l), od(i, l), c(i, jkl), method_merit(m, i), global_merit (i)
BEGIN PROGRAM
Transform the uploaded AP in so called "Work standard form of
the MADM problem" by executing:
 - Verifying that the vectors and matrices are complete defined,
 - The correctitude of data is assured at the filling-in process,
 - The passing, if necessary, from minimum to maximum in the
 EP;
- If the model in correct and complete, the normalization process
for attributes is started, see the following procedure One uses, for
example, nm(1) which  corresponds to the first Von Neumann –
Morgenstern method.
DO l = 1, l
  DO k = 1, k
     DO j = 1, j
        DO i = 1, i
           IF sa(j) = 0 THEN
              oasd(i, j, k, l)=(oasd(i, j, k, l) - la(j)) / (ua(j) - la(j))
            ELSE
              oasd(i, j, k, l)=(ua(j) - oasd(i, j, k, l)) / (ua(j) - la(j))
            ENDIF
        ENDDO
     ENDDO
  ENDDO
ENDDO
Select from SM a number of solving methods; let be, for example
sm(5), sm(7) and sm(9), which correspond respectively to
SCORES, ONICESCU and TOPSIS methods.
Mark the choice in BSM = {bsm(m) | m = 1,m} , m = 16.
Do the solving process.
Display the solution.
DO m = 1, m
    DISPLAY sm(m)
    IF sm(m) IS SELECTED
       bsm(m) = 1
    ELSE
       bsm(m) = 0
    ENDIF
ENDDO
DO m = 1, m
    IF bsm(m) = 1
       CURRENT_METHOD = sm(m)
       PERFORM PROCEDURE SOLVING USING
                                        CURRENT_METHOD
    ENDIF
ENDDO
DO i = 1, i
    global_ merit(i) = 0
    DO m = 1, m
        global_ merit(i) = global_ merit(i) + method_merit(m, i)
    ENDDO
    global_ merit(i) = global_ merit(i) / m
ENDDO
RANK DESCENDING o(i) IN ACCORDANCE WITH
                                        global_merit(i)
DISPLAY "Problem's solution"
DISPLAY FOR  i = 1, i , m = 1, m o(i), methodt(mi),
method_merit(m, i)
DISPLAY FOR  i = 1, i global_ merit(i)
STOP
END PROGRAM
```

PROCEDURE *SOLVING* **USING**
CURRENT_METHOD

```
Optimization at "Attributes" level.
DO l = 1, l
    DO k = 1, k
        DO j = 1, j
            ww(j) = wa(j) / 100
        ENDDO
        DO i = 1, i
            DO j = 1, j
                c(i, j) = oasd(i, j, k, l)
            ENDDO
        ENDDO
        PERFORM PROCEDURE CURRENT_METHOD
                    IN i, j, c(i, j), ww(j) OUT merit(i)
        DO i = 1, i
            osd(i, k, l) = merit(i)
        ENDDO
    ENDDO
ENDDO

Optimization at "States of nature" level
DO l = 1, l
    DO k = 1, k
        ww(k) = ws(k) / 100
    ENDDO
    DO i = 1, i
        DO k = 1, k
            c(i, k) = osd(i, k, l)
        ENDDO
    ENDDO
    PERFORM PROCEDURE CURRENT_METHOD
                IN i, k, c(i, k), ww(k) OUT merit(i)
    DO i = 1, i
        od(i, l) = merit(i)
    ENDDO
ENDDO

Optimization at "Experts" level
DO l = 1, l
    ww(l) = wd(l) / 100
ENDDO
DO i = 1, i
    DO l = 1, l
        c(i, l) = od(i, l)
    ENDDO
ENDDO
PERFORM PROCEDURE CURRENT_METHOD
                            IN i. l, c(i, l), ww(l) OUT merit(i)
ENDPROCEDURE
```

SCORES method

```
PROCEDURE sm(5)
IN i, jkl, c(i, jkl), ww(jkl) OUT merit(i)
LOCAL i, jkl, smerit
DO i = 1, i
    merit(i) = 0
ENDDO
DO jkl = 1, jkl
    FIX jkl
    RANK DESCENDING c(i, jkl), i=1, i
    DO i = 1, i
        IF c(i, jkl) >= 0.9 THEN
            merit(i) = merit(i) + 10
        ELSE
            IF c(i, jkl) >= 0.8 THEN
                merit(i) = merit(i) + 9
            ELSE
                IF c(i, jkl) >= 0.7 THEN
                    merit(i) = merit(i) + 8
                ELSE
                    IF c(i, jkl) >= 0.6 THEN
                        merit(i) = merit(i) + 7
                    ELSE
                        IF c(i, jkl) >= 0.5 THEN
                            merit(i) = merit(i) + 6
                        ELSE
                            IF c(i, jkl) >= 0.4 THEN
                                merit(i) = merit(i) + 5
                            ELSE
                                IF c(i, jkl) >= 0.3 THEN
                                    merit(i) = merit(i) + 4
                                ELSE
                                    IF c(i, jkl) >= 0.2 THEN
                                        merit(i) = merit(i) + 3
                                    ELSE
                                        IF c(i, jkl) >= 0.1 THEN
                                            merit(i) = merit(i) + 2
                                        ELSE
                                            merit(i) = merit(i) + 1
                                        ENDIF
                                    ENDIF
                                ENDIF
                            ENDIF
                        ENDIF
                    ENDIF
                ENDIF
            ENDIF
        ENDIF
    ENDDO
ENDDO
DO i = 1, i
    smerit = smerit + merit(i)
ENDDO
DO i = 1, i
    merit(i) = merit(i) / smerit
ENDDO
ENDPROCEDURE
```

ONICESCU method

```
PROCEDURE sm(7)
IN i, jkl, c(i, jkl), ww(jkl) OUT merit(i)
LOCAL i, jkl, rank(i), smerit
DO i = 1, i
    merit(i) = 0
ENDDO
DO jkl = 1, jkl
    FIX jkl
    RANK DESCENDING c(i, jkl), i=1, i
    rank(1) = 1
    DO i = 2, i
        IF c(i, jkl) = c(i-1, jkl) THEN
            rank(i) = rank(i-1)
        ELSE
            rank(i) = rank(i-1)+1
        ENDIF
    ENDDO
    DO i = 1, i
        merit(i) = merit(i) + ww(jkl) / 2 ** rank(i)
    ENDDO
ENDDO
DO i = 1, i
    smerit = smerit + merit(i)
ENDDO
DO i = 1, i
    merit(i) = merit(i) / smerit
ENDDO
ENDPROCEDURE
```

TOPSIS method

```
PROCEDURE sm(9)
IN i, jkl, c(i, jkl), ww(jkl) OUT merit(i)
LOCAL i, jkl, merit⁻, merit⁺
DO i = 1, i
    merit(i) = 0
ENDDO
DO i = 1, i
    DO jkl = 1, jkl
        merit⁻ = merit⁻ + ww(jkl) * (0 - c(i, jkl))²
        merit⁺ = merit⁺ + ww(jkl) * (1 - c(i, jkl))²
    ENDDO
    merit⁻ = sqrt (merit⁻)
    merit⁺ = sqrt (merit⁺)
    merit(i) = merit⁻ / (merit⁺ - merit⁻)
ENDDO
ENDPROCEDURE
```

3. Case Study

For this small scale case study, one chooses an easy but telling assessment problem. For the beginning the AM will be defined and then the associated APs will be solved.

3.1 Experts

The experts set $D = \{d(1), d(2), d(3)\}$ contains three Information Technology specialists with a good experience in parallel and distributed computing, the authors of this paper: $d(1)$ = Marin Andreica, $d(2)$ = Cornel Resteanu and $d(3)$ = Romica Trandafir. Their weights, established by mutual consensus at the level of experts, are $WD = \{40, 30, 30\}$. In most cases, the experts must be more than one because that manner the accuracy of assessing increases.

3.2 States of nature

One considers as states of nature $S = \{s(1), s(2)\}$ with $WS = \{50, 50\}$, where:

- $s(1)$ = an *ad hoc net* consisting of one server (MB Intel Brandon2 - SE7520BD2VD2, dual processor Intel XEON 800FSB) and ten multiprocessors (MB D975XBX2KR - CPU CORE2 EXTREME, QUADCORE), both computers types configured at medium level and linked into a star net configuration, see [16];
- $s(2)$ = the well-known *EGEE GRID*, the net dedicated to European specialists in High Performance Computing, see [17, 18].

3.3 Objects

The objects set is $O\{o(1), o(2)\}$ containing two software to assess and compare. Obviously they accomplish the same function, the computing of π number (the ratio of the circumference to the diameter of a circle), but with different computing algorithms based on these different formulas:

Wallis:

$$\frac{2}{\pi} = \frac{\left(1\cdot 3\cdot 3\cdot 5\cdot 5\cdot 7\cdot 7\cdots(2n-1)\cdot(2n-1)\right)}{\left(2\cdot 2\cdot 4\cdot 4\cdot 6\cdot 6\cdot 8\cdots 2(n-1)\cdot 2(n-1)\cdot 2n\right)} \quad (1)$$

Leibnitz:

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \ldots\ldots \quad (2)$$

There is, in each case, a master program that distributes, to the slave programs (in fact one program with many launching in execution): their id, the computing formula (in an adequate format), and the number of terms / factors to be computed. That, every iteration, the slave programs known what they have to compute. The master program waits for the slave programs' results, collects them and computes the partial solution. If the stopping condition, the default running time or the default number of correct decimals, is not accomplished, then a new iteration is launched. The programs are written in FORTRAN and MPI. One works with very large numbers, larger than 2^{64}, which need computing on special representations. The tool used is GNU Multiple Precision Arithmetic Library (GMP). It is a free general-purpose library for arithmetic on large numbers. The last version of GMP appeared on 2013, May 20.

3.4 Attributes

As said in the previous section, to express the attributes, the experts will utilize grades from 1 to 10. The vector of the **j** limits will be: $LUA = \{(1, 10), (1, 10), \ldots ,(1,10)\}$.

The experts' evaluation being by grades, the sense vector will be: $SA = \{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1\}$.

To correctly fill-in the assessment matrix, one considers of great importance to correctly interpret the significance of *SA* elements. For example, the objects' attribute a (1), representing the quality of the algorithms chosen for solving the given problem and the quality of their programming, will receive good grades if this characteristic has large values. On the other hand, the objects' attribute $a(4)$, representing the developing costs of the programs, will receive good grades if this characteristic has small values.

The weights vector, given by the user, is: WA = {5, 5, 4, 2, 3, 3, 3, 2, 2, 2, 3, 3, 2, 2, 3, 1, 2, 1, 1, 2, 2, 3, 3, 3, 2, 2, 3, 3, 3, 3, 4, 4, 3, 2, 2, 5, 2}.

3.5 Characteristics matrix

The above announced specialists must independently fill-up the characteristics matrix. For a good filling-up of this matrix, initially, the objects and the attributes which give the dimensions of the matrix must be filled-up. Once these dimensions have been established, the information associated to the attributes, i.e. the weights, optimization sense (minimum / maximum), lower and upper values must be filled-up. They help to the validation process. The very elements of the matrix, $c(o(i), a(j), s(k), d(l))$ with $i=\overline{1,2}$, $j=\overline{1,37}$, $k=\overline{1,2}$, $l=\overline{1,3}$, will be the last filled-up. They are in the decision makers' possession by direct testing, benchmarking and expert evaluation.

Every model element will be validated at its entry. Proceeding like described above, the model will be contained in Table 1. One notices that in this table there are lines with equal elements on all columns. These lines do not discriminate the objects but contribute to their merits' computing.

Table 1. The characteristics matrix

Decision makers →					Marin Andreica				Cornel Resteanu				Romica Trandafir			
States of Nature →					Ad-hoc net		EGEE grid		Ad-hoc net		EGEE grid		Ad-hoc net		EGEE grid	
Objects →					o1	o2	o1	o2	o1	o2	o1	o2	o1	o2	o1	o2
Attributes ↓	Weights	L	U	m/M												
a1	5	1	10	1	8.50	9	8.50	9	8	9	8	9	8	9	8	9
a2	5	1	10	1	8.50	8.50	8	8	9	9	8	8	9.50	9.50	9	9
a3	4	1	10	1	9.75	9.75	9.75	9.75	9.75	9.75	9.75	9.75	9.75	9.75	9.75	9.75
a4	2	1	10	1	9	9	9.50	9.50	9	9	9.25	9.25	8.75	8.75	9.50	9.50
a5	3	1	10	1	7	7	7	7	6	6	6	6	6.50	6.50	6.50	6.50
a6	3	1	10	1	8.50	8.50	8.50	8.50	9	9	9	9	9	9	9	9
a7	3	1	10	1	9.50	9.50	9.50	9.50	9.75	9.75	9.75	9.75	9.50	9.50	9.50	9.50
a8	2	1	10	1	8.50	8.50	8.50	8.50	9	9	9	9	9	9	9	9
a9	2	1	10	1	10	10	10	10	10	10	10	10	10	10	10	10
a10	2	1	10	1	10	10	10	10	10	10	10	10	10	10	10	10
a11	3	1	10	1	10	10	10	10	10	10	10	10	10	10	10	10
a12	3	1	10	1	8.50	8.50	8.50	8.50	9.25	9.25	9.25	9.25	9	9	9	9
a13	2	1	10	1	9.50	9.50	9.50	9.50	9.75	9.75	9.75	9.75	9.50	9.50	9.50	9.50
a14	2	1	10	1	9	9	9	9	9.25	9.25	9.25	9.25	9	9	9	9
a15	3	1	10	1	9.75	9.75	9.75	9.75	10	10	10	10	9.90	9.90	9.90	9.90
a16	1	1	10	1	9.75	9.75	9.75	9.75	10	10	10	10	9.90	9.90	9.90	9.90
a17	2	1	10	1	9.50	9.50	9.75	9.75	9.25	9.25	9.50	9.50	8	8	9	9
a18	1	1	10	1	9.50	9.50	9.75	9.75	9.25	9.25	9.50	9.50	8	8	9	9
a19	1	1	10	1	8	8	9	9	8.50	8.50	9	9	9	9	9.25	9.25
a20	2	1	10	1	9.50	9.25	9.75	9.50	9	8.50	9.50	9	9.25	9	9.50	9.25
a21	2	1	10	1	9	8.75	9.25	9	8.50	8	9	8.50	8.75	8.50	9	8.75
a22	3	1	10	1	9.50	9.50	9.75	9.75	9.75	9.75	9.90	9.90	9.25	9.25	9.50	9.50
a23	3	1	10	1	9.75	9.75	9.75	9.75	9.80	9.80	9.80	9.80	9.85	9.85	9.85	9.85
a24	3	1	10	1	8.50	8.50	9.50	9.50	8	8	9	9	8.50	8.50	9	9
a25	2	1	10	1	7	7	9.75	9.75	7.50	7.50	9.50	9.50	7.50	7.50	9.90	9.90
a26	2	1	10	1	7	7	9.75	9.75	7.50	7.50	9.50	9.50	7.50	7.50	9.50	9.50
a27	3	1	10	1	9	9	9.75	9.75	9	9	9.50	9.50	8.75	8.75	9.50	9.50
a28	3	1	10	1	9	9	9.50	9.50	9	9	9.75	9.75	9.25	9.25	9.75	9.75
a29	3	1	10	1	9.50	9.50	9.75	9.75	9.50	9.50	9.90	9.90	9.25	9.25	9.75	9.75
a30	3	1	10	1	9	9	9.75	9.75	9.25	9.25	9.50	9.50	9	9	9.50	9.50
a31	4	1	10	1	6	6	9.50	9.50	7	7	9	9	6	6	9.25	9.25
a32	4	1	10	1	7	8	9	9.25	7.50	8	9.25	9.50	7.75	8	9.50	9.75
a33	3	1	10	1	7.50	8.5	9.25	9.50	7.75	8.20	9.50	9.75	8	8.50	9.50	9.75
a34	2	1	10	1	9	9	8	8	9.50	9.50	8	8	9	9	7.50	7.50
a35	2	1	10	1	9	9	9.15	9.15	9	9	9.25	9.25	8.50	8.50	9	9
a36	5	1	10	1	9.50	9.50	9.25	9.25	9.50	9.50	9	9	9.75	9.75	9.50	9.50
a37	2	1	10	1	8.50	8.50	7.50	7.50	8	8	7	7	8.25	8.25	6.75	6.75

3.6 Solving

In solving, *OPTCHOICE* is very versatile. It is possible to specify a lot of input parameters, the most important being the entities' elements taken in a current model and the normalization - solving methods couples. Thus, in accordance with the goals of the case study, one generates and solves three associated APs:

- Assessment when only the ad-hoc platform is used,
- Assessment when only the EGEE grid platform is used,
- Assessment when both above platform are used.

All three APs will be solved in conformity with three *MADM* methods: SCORES, ONICESCU and TOPSIS, using the first von Newman – Morgenstern method.

Correspondingly, the *OPTCHOICE* software will give the results presented in Tables 2, 3, 4. These tables contain the objects' merits given by the chosen solving methods. Obviously, there are, as well, the merits after the GLOBAL method, which are the main results of the assessment problems. Working upon these results, it is possible to edit another two tables, containing the objects merits when $o(1)$ is executed on Ad-hoc platform and $o(2)$ on EGEE grid platform, see Table 5, and when $o(1)$ is executed on EGEE grid platform and $o(2)$ on Ad-hoc platform, see Table 6.

Table 2. Ad-hoc platform – problem's solutions

Objects	SCORES	ONICESCU	TOPSIS	GLOBAL
$o(2)$	0.72	0.85	0.88	0.82
$o(1)$	0.67	0.82	0.86	0.78

Table 3. EGEE grid platform – problem's solutions

Objects	SCORES	ONICESCU	TOPSIS	GLOBAL
$o(2)$	0.73	0.86	0.89	0.83
$o(1)$	0.68	0.83	0.88	0.80

Table 4. Both platforms - problem's solutions

Objects	SCORES	ONICESCU	TOPSIS	GLOBAL
$o(2)$	0.78	0.88	0.91	0.86
$o(1)$	0.69	0.85	0.89	0.81

Table 5. Ad-hoc platform for $o(1)$ and EGEE platform for o(2) – problem's solutions

Objects	SCORES	ONICESCU	TOPSIS	GLOBAL
$o(2)$	0.73	0.86	0.89	0.83
$o(1)$	0.67	0.82	0.86	0.78

Table 6. EGEE grid platform for o(1) and Ad-hoc platform for $o(2)$ – problem's solutions

Objects	SCORES	ONICESCU	TOPSIS	GLOBAL
$o(2)$	0.72	0.85	0.88	0.82
$o(1)$	0.68	0.83	0.88	0.80

Analyzing the results, it is obvious that $o(2)$ has better *merits* than $o(1)$ in all five cases (it is true after every method separately, including the globalization procedure, and also in every running platform circumstances). In consequence, $o(2)$, as optimum object, will be preferred instead $o(1)$ and, independently of reference platform, it will be chosen for using.

4. Conclusions

It is visible that the method presented in this paper is very good for the assessment of the parallel and distributed software. The fact that the elementary evaluation process must be made by persons with high qualification is a guaranty for the final assessment. The computing is made by using a very good theory, named *MADM*, belonging to the Operation Research field. The *MADM* theory gives the possibility to develop an assessment model containing all parallel and distributed software to be assessed through all characteristics associated to the notions of parallelism and distribution. In the same time, the *MADM* theory provides tools for solving the assessment problems generated from the assessment models.

Because the *MADM* models and problems are difficult enough, a special software tool is necessary to address them. In this case the *OPTCHOICE* software was utilized.

The authors, working together or separately, have a significant experience in parallel and distributed software assessment. In many cases their comments, after the assessment, have led to operations like corrections / redesigning / reprogramming of assessed software.

REFERENCES

1. GRAMA, G., G. KARPIS, V. KUMAR, A. GUPTA, **Introduction to Parallel Computing: Design and Analysis of Parallel Algorithms**, Addison Wesley, 2003.

2. ALAGHBAND, G., H. F. JORDAN, **Fundamentals of Parallel Processing**, Prentice Hall, 2002.

3. DONGARRA, J., K. MADSEN, J. WASNIEWSKI (Eds), **Applied Parallel Computing: State of the Art in Scientific Computing**. In: Lecture Notes in Computer Science, Springer; Volume 3732, 1 edition, April 11, 2006.

4. BOGETOFT, P., **Performance Benchmarking**, Springer, Series: Management for Professionals, ISBN 978-1-4614-6042-8, 2012.

5. KAELI, D. (Ed.), **Computer Performance Evaluation and Benchmarking**, Series: Lecture Notes in Computer Science, Vol. 5419, Subseries: Programming and Software Engineering, ISBN 978-3-540-93798-2, 2009.

6. MAHANTI, R., J. R. EVANS, **Critical Success Factors for Implementing Statistical Process Control in the Software Industry**, Benchmarking: An International Journal, vol. 19, issue 3, 2012, pp. 374-394.

7. YOON, K., C.-L. HWANG, **Multiple Attribute Decision Making: An Introduction**, SAGE Publications, Thousand Oaks, London, New Delhi, 1995.

8. EL-REWINI, H., T. G. LEWIS, **Scheduling Parallel Program Tasks onto Arbitrary Target Machines**, Journal of Parallel and Distributed Computing, vol. 9, June 1990, pp. 138-153.

9. KRAUTER, K., R. BUYYA, M. MAHESWARAN, **A Taxonomy and Survey of Grid Resource Management Systems for Distributed Computing**, in Software: Practice and Experience, vol. 32, issue 2, 2002, pp. 135-164.

10. HWANG, C.-L., M. J. LIN, **Group Decision Making under Multiple Criteria**, Springer-Verlag, Berlin Heidelberg New York, 1997.

11. RESTEANU, C., M. ANDREICA, **Distributed and Parallel Computing in MADM Domain Using the OPTCHOICE Software**, International Journal of Mathematical Models and Methods in Applied Sciences, NAUN (North Atlantic University Union), ISSN: 1998-0140, Issue 3, Volume 1, 2007, pp. 159-167.

12. LEOPOLD, C., **Parallel and Distributed Computing: A Survey of Models, Paradigms and Approaches**, LAVOISIER S.A.S., 2001.

13. ROS, A. (Ed.), **Parallel and Distributed Computing,** Publisher: InTech, ISBN 978-953-307-057-5, January 01, 2010.

14. HUGHES, C., T. HUGHES, **Parallel and Distributed Programming Using C++**, Published by Addison-Wesley Professional, ISBN-10: 0-13-101376-9, ISBN-13: 978-0-13-101376-6, August 25, 2003.

15. ZAVADSKAS, E. K., Z. TURSKIS, R. VOLVAČIOVAS, S. KILDIENE, **Multi-criteria Assessment Model of Technologies**, in: Studies in Informatics and Control, ISSN 1220-1766, vol. 22(4), 2013, pp. 249-258.

16. BOKHARI, S. H. **Assignment Problems in Parallel and Distributed Computing**, Kluwer Academic Publishers, 1987.

17. BAKER, M., R. BUYYA, D. LAFORENZA, **Grids and Grid Technologies for Wide-area Distributed Computing**, In Software: Practice and Experience, Volume 32, Issue 15, 2002, pp. 1437-1466.

18. BUYYA, R., K. BUBENDORFER (Eds), **Market-Oriented Grid and Utility Computing**, Wiley, ISBN: 978-0-470-28768-2, November 2009.