Hybrid Meta-heuristic Algorithm for Task Assignment Problem

Mohammad Jafar Tarokh[a], Mehdi Yazdani[b,*], Mani Sharifi[c], Mohammad Navid Mokhtarian[d]

[a] *Associate Professor, Department of Industrial Engineering, K.N. Toosi University of Technology, Tehran, Iran*

[b] *Instructor, Department of industrial engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran*

[c] *Assistant Professor, Department of industrial engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran*

[d] *PHD Student, Department of industrial engineering, Science and research Branch, Islamic Azad University, Tehran, Iran*

Received 23 October, 2010; Revised 28 November, 2010; Accepted 15 December, 2010

Abstract

Task assignment problem (TAP) involves assigning a number of tasks to a number of processors in distributed computing systems and its objective is to minimize the sum of the total execution and communication costs, subject to all of the resource constraints. TAP is a combinatorial optimization problem and NP-complete. This paper proposes a hybrid meta-heuristic algorithm for solving TAP in a heterogeneous distributed computing system. To compare our algorithm with previous ones, an extensive computational study on some benchmark problems was conducted. The results obtained from the computational study indicate that the proposed algorithm is a viable and effective approach for the TAP.

Keywords: Task assignment problem; Heterogeneous distributed systems; Hybrid meta-heuristic; Simulated annealing algorithm; Variable neighbourhood search algorithm.

1- Introduction

A distributed computing system is defined as a collection of computers interconnected by a telecommunication network that attempts to disperse the data processing function and fits the needs of modern decentralized organization structures [32]. There are a lot of problems regarding distributed computing systems that are essentially very hard. These problems cannot be formulated as linear programs and there are no simple rules or algorithms that may yield optimal solutions in a limited amount of computer time. One of the most difficult problems in this area is the task assignment problem (TAP).
This problem involves assigning tasks of a program among different processors of a distributed computer system in order to minimize the sum of the communication and execution (task processing) costs. In other words, TAP aims to find an assignment of tasks which minimizes the incurred costs subject to the resource constraint [31]. Specially, the number of tasks that a given processor is able to handle is restricted by its memory capability and processing ability.

TAP was firstly introduced by Stone [24]. Stone's original work lays down the task interaction graph (TIG) model to represent sequentially executing tasks. In the TIG model, the vertices of the graph correspond to the tasks and the edges correspond to the inter-task communications.

A large number of researches have been carried out on this problem and different models of TAP have been proposed in the literature. Also, over the past decades wide varieties of approaches have been proposed for solving the task assignment problem in distributed computing system. The existing approaches for tackling the TAP can be divided into three categories. First, exact mathematical programming approaches using column generation [6] and branch-and-bound [3,5] have been proposed. Second, efficient algorithms have been developed for solving TAP on special computer architectures, such as linear processor array, meshed processor graph, and partial k-tree communication graph [7,13,16]. Finally, meta-heuristic algorithms like Genetic Algorithm (GA) [2,26,27], Partial swarm optimization (PSO) [12,23,28], differential evolution (DE) [33], simulated annealing (SA) [10,17,25], Variable neighborhood search (VNS) [15,18] and hybrid meta-heuristic algorithms [9,22,31] have been used to derive good enough approximate solutions within reasonable CPU time.

It is well known that TAP is NP-complete [8]. Presentation of good exact algorithm for TAP in polynomial time is unlikely to exist. The high level of the complexity of TAP demonstrates a cogent reason for using meta-heuristic algorithms to optimize TAP.

*Corresponding Author E-mail: M_Yazdani@qiau.ac.ir

Therefore, in the last few years, there have been increasing interests for meta-heuristics approaches, as Therefore, in the last few years, there have been increasing interests for meta-heuristics approaches, as robust approaches for tackle the TAP. VNS and SA algorithms are well-known meta-heuristics which have been applied successfully to various optimization problems. In this paper, we proposed a hybrid VNS-SA algorithm for TAP in a heterogeneous distributed computing system to minimize the sum of the communication and execution costs. In the proposed method, VNS algorithm is used as main part of hybrid algorithm and SA algorithm supports it for attaining better solutions. Also, Effective neighborhood structures are applied in various parts of the presented hybrid method. In this paper an attempt is made to prove the efficiency and effectiveness of our algorithm by comparing the performance of the proposed meta-heuristic method with some previous methods. We present the computational results of our hybrid algorithm on some of test problems and compare them with the results reported by previous researchers.

The rest of the paper is organized as follows. Section 2 explains problem definition and one of the existing mathematical models for the understudied problem. In Section 3, hybrid meta-heuristic algorithm for solving the studied problem is described. Section 4 reports Computational study. The conclusion and some suggestions for further research are presented in Section 5.

2- Problem Definition

In a computation system with a number of distributed processors, it is desired to assign application tasks to these processors such that the resource demand of each task is satisfied and the system throughput is increased. However, the assignment of tasks will also incur some costs such as the execution cost and the communication cost [31]. Therefore, TAP often tries to find the minimal costs for a distributed system which subjects to several resource constraints. In this paper, we consider the TAP with the following scenarios. The processors in the system are heterogeneous and they are capacitated with various units of memory and processing resources. All the tasks may be executed in all the processors. A task will incur different execution costs if it is executed on different processors. Two tasks executing incur in a communications cost if there is a communication need between them and they are executed on different processors. This work does not consider communication cost between tasks executing in the same processor. A task will consume some units of the resources from its execution processor. Memory and processing resource capacity of each processor is limited. Our problem objective is to minimize the total execution and communication costs incurred by the task assignment subject to all of the resource constraints. There are different versions of formulations for TAP, and the formulation we consider in this paper is taken from Yin et al. [31]. Following parameters and decision variables are used in the proposed model.

n: Number of processors
r: Number of tasks
e_{ik}: Execution cost of task i if it is assigned to processor k
c_{ij}: Incurred communication cost between tasks i and j if they are assigned to different processors
m_i: Memory requirement of task i from its execution processor
M_k: Memory capacity of processor k
P_i: Processing requirement of task i from its execution processor
P_k: Processing capacity of processor k
x_{ik}: Decision variable: $x_{ik}=1$ if task i is assigned to processor k, and $x_{ik}=0$ otherwise.

The considered TAP can be formulated as the following 0–1 quadratic integer programming problem:

$$\min\ f(x)=\sum_{i=1}^{r}\sum_{k=1}^{n}e_{ik}x_{ik}+\sum_{i=1}^{r-1}\sum_{j=i+1}^{r}c_{ij}\left(1-\sum_{k=1}^{n}x_{ik}x_{jk}\right)$$

$$s.t.\ \sum_{k=1}^{n}x_{ik}=1,\quad \forall i=1,2,...,r$$

$$\sum_{i=1}^{r}m_i x_{ik}\le M_k\quad \forall k=1,2,...,n \tag{1}$$

$$\sum_{i=1}^{r}p_i x_{ik}\le P_k\quad \forall k=1,2,...,n$$

$$x_{ik}\in\{0,1\},\quad \forall i,k$$

The first and second terms in the objective function represent the total execution cost and communication cost, respectively, incurred by the tasks assignment. This model is limited by three kinds of constraints. Constraint (1) states that each task should be assigned to exactly one processor. Constraints (2) and (3) ensure that the memory and processing resource capacity of each processor is no less than the total amount of resource demands of all of its assigned tasks. The last constraint (5) guarantees that x_{ik} are binary decision variables.

Since this formulation is an integer program with a quadratic objective function and is computationally prohibitive due to enormous computation efforts, its transformations to linear programs have been proposed

[3,6]. However, unfortunately, solving the linear program model of this problem is still time-consuming for deriving optimal solutions to large-scaled problems. Therefore, in this paper, we applied a hybrid meta-heuristic algorithm to solve TAP with a reasonable time.

3- Proposed Hybrid Meta-Heuristic Algorithm

To search efficiently in the solution space and attain solutions with high quality in TAP, we propose hybrid VNS-SA algorithm in this section. Section 3 is subdivided into the following 5 subdivisions. Basic structures of VNS and SA algorithms are discussed in sections 3.1 and 3.2, respectively; Section 3.3 provides solution representation method; neighborhood structures employed in the presented hybrid method are explained in detail in section 3.4. In section 3.5, a comprehensive description of the proposed hybrid meta-heuristic algorithm is given**.**

3.1. Basic Structures of VNS Algorithm

VNS algorithm [11,19] is one of the renowned meta-heuristics which have been successfully applied to solve combinatorial optimization problems. This method is capable of escaping from the local optima by systematic changes of the neighborhood structures during the search process [1]. VNS was first proposed by Mladenovic [20]. The basic steps of VNS meta-heuristic are shown in Fig. 1. During the initialization step, a set of neighborhood structures and the sequence of their implementations are determined. In this step, stopping condition is delineated. In addition, an initial solution is generated and is set as the current solution y. In the search loop, the shake procedure is implemented for randomly generating a neighboring solution y' from the current solution y based on the first neighborhood structure. Then local search is carried out for obtaining local optimum y'', with y' as input solution of local search. Afterward, local optimum y'' is compared with current solution y in terms of the solution quality. If y'' is better than y, i.e., improved solution is obtained, y is replaced with y'' and search begins again at the first neighborhood with the updated y. Otherwise, the search loop is iterated by the next neighborhood. In this case, neighborhood structure is systematically changed and the shake procedure works to switch to another region of the search space so as to carry out a new local search there. After all neighborhoods are considered and no further improvement can be obtained for the current solution y, next iteration of algorithm is started and the search begins again at the first neighborhood of current solution y. The VNS algorithm continues until a stopping condition (e.g., the maximum computational time since the last improvement, or the maximal allowed CPU time, or maximum number of iterations) is satisfied [29].

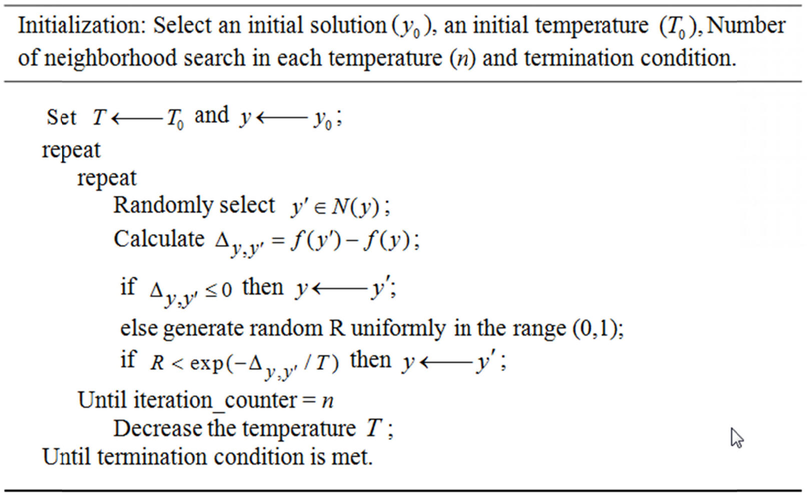
```
Initialization:
Select the set of neighborhood structures $N_k$, for $k = 1,...,k_{\max}$; find an initial solution $y$;
choose a stopping condition;
Repeat the following sequence until stopping condition is reached:
    Set $k \longleftarrow 1$;
    Repeat the following steps until $k > k_{\max}$:
        Shaking:
        Randomly generate a solution $y'$ from the $k^{th}$ neighborhood of $y$;
        Local search:
        Apply some local search methods with $y'$ as initial solution to obtain a local optimum given by $y''$;
        Move or not:
        if the local optimum $y''$ is better than $y$, move there $(y \longleftarrow y'')$, and continue the search
        with $N_1$ $(k \longleftarrow 1)$; Otherwise, set $k \longleftarrow k+1$;
```

Fig. 1. The basic steps of VNS algorithm

3.2. Basic Structures of SA Algorithm

Simulated annealing (SA) is a random-search technique which simulates the way in which a metal cools and freezes into a minimum energy crystalline structure (the annealing process) in optimizing combinatorial optimization problems [21]. SA searches the set of all possible solutions, reducing the chance of getting stuck in local optima by allowing moves to inferior solutions under the control of a randomized scheme [30]. This algorithm was first proposed by Kirkpatrick et al. [14].

The SA algorithm starts with an initial solution and iteratively moves towards other existing solutions. The algorithm generates a neighboring solution y' in the neighborhood of the candidate solution y. Then, the change of objective function value, $\Delta_{y,y'} = f(y') - f(y)$, is calculated. In case $\Delta_{y,y'}$ value is smaller than zero or is equal to zero, to move to the neighboring solution is acceptable; otherwise, if $\Delta_{y,y'}$ value is greater than zero, to reduce the probability to get trapped in local optima, the SA may accept to move to an inferior neighboring solution depending on a randomized scheme. More precisely, the move is still accepted if $R < \exp(-\Delta_{y,y'} / T)$, where T is a control parameter, called temperature, and R is a uniform random number between interval (0,1). SA algorithm generally starts from a high temperature and then the temperature is gradually lowered [4]. At each temperature, a search is carried out for a certain number of iterations. When the termination condition is satisfied, the algorithm will stop. The general SA algorithm is shown in Fig. 2.



```
Initialization: Select an initial solution ($y_0$), an initial temperature ($T_0$), Number
of neighborhood search in each temperature ($n$) and termination condition.

Set $T \longleftarrow T_0$ and $y \longleftarrow y_0$;
repeat
    repeat
        Randomly select $y' \in N(y)$;
        Calculate $\Delta_{y,y'} = f(y') - f(y)$;
        if $\Delta_{y,y'} \le 0$ then $y \longleftarrow y'$;
        else generate random R uniformly in the range (0,1);
        if $R < \exp(-\Delta_{y,y'} / T)$ then $y \longleftarrow y'$;
    Until iteration_counter = $n$
        Decrease the temperature $T$;
Until termination condition is met.
```

Fig. 2. The basic steps of SA algorithm

3.3. Solution Representation Method

In this paper, to represent our hybrid algorithm solutions, we use a string of r elements. Number of elements in the solution string (r) is equal to the total number of tasks. Considering left-to-right ordering, element i (i=1,…,r) represents task i. In each element is placed an integer value between 1 and n (number of processors) that this integer value signifies assigned processor to Task i. Fig. 3 shows representation solution for one candidate solution, which corresponds to a task assignment that assigns six tasks to four processors. For example, number 2 in element 3 means that the third task is assigned to the second processor.

1	2	3	4	5	6
3	4	2	4	2	1

Fig. 3. Representation of a candidate solution

3.4. Neighborhood Structure

The technique of moving from one solution to its neighboring solution is delineated by a key factor known as neighborhood structure [29]. Various neighborhood structures have been applied to TAP. These neighborhood structures must work so that they prevent any infeasible solution regarding the resource constraints. Hence, it is guaranteed that the algorithm gives always a feasible solution. In this section, six types of neighbourhood structures employed in the proposed hybrid method are described.

Neighborhood structure -1 (N_1)

This neighborhood structure reallocates a task i from processor l to processor k in candidate solution string [18].

Intelligent neighborhood structure - 2 (N_2)

This neighborhood structure reallocates a task *i* from processor *l* to processor with minimum processing cost for executing task *i*.

Intelligent neighborhood structure - 3 (N_3)

This neighborhood structure reallocates a task *i* from processor *l* to processor that its assigned tasks have maximum total communication cost toward task *i*.

Neighborhood structure - 4 (N_4)

This neighborhood structure exchanges assignment of two tasks (task *i* from processor *l* to other processor and task *j* from processor *k* to other processor).

Neighborhood structure - 5 (N_5)

This neighborhood structure exchanges assignment of three tasks (task *i* from processor *k* to processor *l*, task *j* from processor *l* to processor *k* and task *t* ($t \neq i$ and *j*) from processor *v* ($v \neq l$ and *k*) to other processor.

Neighborhood structure - 6 (N_6)

This neighborhood structure reallocates a cluster of tasks from one or different processors to processor *l* [18]. The idea of this neighborhood structure is based on decrease of communication cost.

3.5. Proposed VNS-SA Algorithm

The proposed hybrid meta-heuristic algorithm included two parts. The main part of this method is VNS algorithm and the other part is SA algorithm. For VNS algorithm, two sets of neighborhood structures are used in the presented algorithm: N_k^s and N_l^{ls}. Neighborhood structures related to N_k^s are employed in the shake procedure and neighborhood structures related to N_l^{ls} are utilized in the local search procedure. All the six mentioned neighborhood structures are implemented in the shake procedure of VNS $(N_k^s, k = 1,2,...,6)$ and the neighborhood structures N_1, N_2, N_3 and N_4 are utilized in the local search procedure of VNS $(N_l^{ls}, k = 1,2,3,4)$. These four neighborhood structures are used in the search procedure of SA part of hybrid algorithm. The performance of the hybridized meta-heuristic significantly depends on the efficiency of the applied neighborhood structures.

Fig. 4 demonstrates the general process of the proposed hybrid VNS-SA algorithm. Hybrid algorithm in each iteration begins from VNS part. If one run (a completed execution of shake and local search procedures in VNS part of hybrid algorithm) among all runs (*k*=1,.., 6) of VNS method attains better solution than current solution, the current solution is replaced with new solution and next iteration of the hybrid algorithm is started. Otherwise, in each iteration of hybrid algorithm, if all runs of VNS method doesn't attain better solution, algorithm move to SA part. In this part of algorithm, we use a linear function to reduce the temperature; the function is illustrated in equation (2) [4,30]:

$$T_i = T_0 - i.\frac{T_0 - T_f}{n_{\text{SA}}} \qquad (2)$$

In the equation, T_0 denotes initial temperature, T_f denotes final temperature, *i* represents a stage in the algorithm, T_i represents the temperature of stage *i* and n_{SA} is the maximum number of iterations of SA algorithm. Also, for obtaining initial temperature of SA part, proposed strategy by Yazdani et al. [30] is used. An appropriate initial temperature should be high enough to create equal opportunity for all states of the search space to be visited. Meanwhile, it should not be rather too high to perform quite a lot of unnecessary searches in high temperature. The initial temperature is acquired via equation (2):

$$T_0 = \left[\overline{\Delta_f^+} \right] \qquad (3)$$

where Δ_f^+ is any positive differences in fitness value when a neighboring solution is achieved from the candidate solution and $\overline{\Delta_f^+}$ is the average value of these differences. Besides, we do not consider the negative or neutral differences in fitness value to compute the average value. In equation (2), the symbol [] denotes that the initial temperature, T_0, is set to the integer part of the average value of differences.

The hybrid algorithm continues until pre-specified maximum number of iterations of algorithm is achieved. When the process of the algorithm stops, the final current solution is used as the best solution.

Initialization

a. Select a set of neighborhood structures that will be used in VNS and SA parts of hybrid algorithm, number of neighborhood search in each iteration of VNS local search $(n_{l-\text{vns}})$, initial temperature of SA (T_0), number of SA algorithm iterations (n_{SA}) based on linear function of reduce temperature and number of neighborhood search in each temperature of SA $(n_{l-\text{SA}})$.

b. Choose stoping condition (maximum number of iterations $(n_{\max})$.

c. Generate randomly a population of solutions and the best solution among them (y) is selected for initial solution.

d. D. set initial solution as current solution $\tilde{y} \longleftarrow y$.

```
for $i = 1 : n_{\max}$ (begin of hybrid algorithm)
  **VNS part of algorithm:**
  for $k = 1 : k_{\max}$ $(k_{\max} = 6)$
      Shaking:
        Generate random point $y_{\text{vns}} \in N_k^s(\tilde{y})$ with $N_k^s$;
      Local search:
          Get solution $y_{\text{vns}}$;
            for $j = 1 : n_{l-\text{vns}}$
              Select random one number among numbers 1 to 4 and set $l \longleftarrow$ selected number;
              Generate random point $y'_{\text{vns}} \in N_l^{ls}(y_{\text{vns}})$ with $N_l^{ls}$;
              If $f(y'_{\text{vns}}) \le f(y_{\text{vns}})$ then $y_{\text{vns}} \longleftarrow y'_{\text{vns}}$;
              endif
            endfor
      Updating:
          If $f(y_{\text{vns}}) \le f(\tilde{y})$ then $\tilde{y} \longleftarrow y_{\text{vns}}$ and move to next iteration of hybrid algorithm (break and
          $i \longleftarrow i+1$);
          else $k \longleftarrow k+1$;
          end if
  endfor
  **SA part of algorithm:**
  Set $T \longleftarrow T_0$ and $y_{\text{SA}} \longleftarrow \tilde{y}$
    for $n = 1 : n_{SA}$
      set $l \longleftarrow 1$
       for $m = 1 : n_{l-SA}$
          Generate random point $y'_{\text{SA}} \in N_l^{ls}(y_{\text{SA}})$ with $N_l^{ls}$;
            if $\Delta_{y_{\text{SA}}, y'_{\text{SA}}} < 0$ then $y_{\text{SA}} \longleftarrow y'_{\text{SA}}$;
            elseif $\Delta_{y_{\text{SA}}, y'_{\text{SA}}} = 0$ then generate random number $R$ uniformity in the range (0,1);
              if $R \le 0.5$ then $y_{\text{SA}} \longleftarrow y'_{\text{SA}}$ and $l \longleftarrow l$;
              endif
            elseif $\Delta_{y_{\text{SA}}, y'_{\text{SA}}} > 0$ then generate rondom number $R$ uniformity in the range (0,1);
              if $R \le \exp(-\Delta_{y_{\text{SA}}, y'_{\text{SA}}} / T)$ then $y_{\text{SA}} \longleftarrow y'_{\text{SA}}$ and $l \longleftarrow l+1$;
              endif
            else $l \longleftarrow l+1$;
            endif
          if $l > 4$ then $l \longleftarrow 1$;
          endif
      endfor
      Update $T$ : Decrease of temperature according to linear low;
    endfor
    $\tilde{y} \longleftarrow y_{\text{SA}}$
endfor (end of hybrid algorithm)
```

Fig. 4. Steps of proposed VNS-SA algorithm.

Regarding the test on different values for algorithm parameters and considering the computational results, the following settings are adjusted for the presented hybrid algorithm:

Maximum number of iterations $(n_{\max})$: 200, 400, 600, 800, 1000 (based on size of problem);
Number of neighborhood searches in each iteration of VNS local search (n_{l-vns}) : 500;
The final temperature T_f : 0.1;
Number of SA algorithm iterations (n_{SA}) : 20;
Number of neighborhood searches in each temperature of SA (n_{l-SA}) : 250.
The employment of hybrid formation in the proposed method has important effect on increasing power of intensification and diversification strategies in search of solution space where VNS method works as main algorithm and SA method supports it for attaining better solution.

4- Computational Results

This section describes the computational tests which were used to evaluate the effectiveness and efficiency of the proposed VNS-SA algorithm in finding good solution for TAP. We implemented the algorithm in MATLAB software and run on a PC with i7 CPU, 1.73 GHz, and 4GB of RAM memory. The non-deterministic nature of the presented algorithm made it necessary to carry out multiple runs on the same test problem in order to obtain meaningful results. Therefore, the best solution was selected for each problem after ten runs of the presented algorithm. We compared the results of our algorithm to the results of the hybrid PSO (HPSO) algorithm of Yin et al. [31], SA and GA algorithms of Yin et al. [32], General Variable neighborhood search (GVNS) of Lusa et al. [18] and Improved DE (IDE) algorithm of Zou et al. [33]. To measure the quality of solutions, the Cost function (total execution and communication costs) is considered as objective function value. For comparing the proposed algorithm versus other algorithm, we first generated some test problems. These test problems were generated according to different problem characteristics which are described in this section.

First key characteristic of created problem is task interaction density. The inter-task communication is rendered by a task interaction graph (TIG), G(V,E), where V is a set of r nodes indicating the r tasks to be executed and each edge $(i,j)\in E$ is associated with a communication cost c_{ij}, which incurs only when tasks i and j are assigned to different processors. Here, we define the task interaction density d of G(V,E) as one of the problem characteristics:

$$d=\frac{|E|}{r(r-1)/2} \tag{4}$$

where $|E|$ calculates the number of existing communication demands in the TIG, and $r(r-1)/2$ indicates the maximal number of communication demands among r tasks. The task interaction density (d) quantifies the ratio of the inter-task communication demands for a TIG and it is one of the key factors that affect the problem complexity [31]. In this paper, we use densities 0.3, 0.5, and 0.8 for preparing test problems. The other key factors are the number of tasks (r) and the number of processors (n).

To Generate test problem, we set the value of (r, n) to (9, 6), (15, 10), (20, 10), (30, 15), (40, 20), (50, 25), (60, 30), and (90, 60), respectively, in order to testify the algorithm with different problem scales. For each pair of (r, n), we generate three different TIGs at random with density d equivalent to 0.3, 0.5, and 0.8. The values of the other parameters are generated randomly: the execution cost is between 1 and 200, the communication cost is between 1 and 50, the memory and processing capacity of each processor varies from 50 to 250, and the memory and processing requirement of each task ranges from 1 to 50.

In Table 1, we compared the results of our hybrid VNS-SA algorithm to the results of the HPSO [31], SA and GA [32], GVNS [18] and IDE algorithm [33] on generated 24 test problems. The first, second, third and fourth columns symbolize problem number, number of tasks, number of processors and task interaction density for test problems, respectively. In the fifth column, the best cost (BC) obtained by presented algorithm is shown for each problem. The averages of the proposed algorithm results of ten runs are shown in the sixth column. The seventh column represents average of CPU times for ten runs of the presented algorithm in terms of seconds. Columns 9, 11, 13, 15 and 17 represent the best results obtained by HPSO, SA, GA, GVNS and IDE respectively. In this section, to provide a clearer view on the comparative performances among the competing methods, Cost offset (CO) measure proposed by Yin et al. [32] was used. This measure is calculated as:

$$\mathrm{CO}=\frac{Al_{sol}-min_{sol}}{Al_{sol}}.100\% \tag{5}$$

where Al_{sol} is the best cost obtained for a given algorithm and test problem and min_{sol} is the best cost obtained from all compared algorithms for the same test problem. The average CO over all the test problems are 0.7288%, 12.08%, and 10.41%, 8.78%, 4.80% and 3.49% for VNS-SA, HPSO, SA, GA, GVNS and IDE, respectively. The best performance is obtained by the VNS-SA with the average CO of 0.7288%.

In Table 2, the presented algorithm is compared with five aforesaid algorithms from another standpoint. This table illustrates to what extent VNS-SA results are equal to, worse or better than those obtained by the aforementioned algorithms on 24 generated test problems.

For more investigations in Computational results, another measure, relative deviation index (RDI), is used in this study. RDI is obtained by given formula below:

$$\mathrm{RDI} = \frac{Al_{sol} - min_{sol}}{max_{sol} - min_{sol}}.100\% \qquad (6)$$

Using this measure, we obtained an index between 0 and 100 for each method. The more the RDI is closer to zero, the more the algorithm is preferable. Note that if the worst and the best solutions take the same value, all the methods provide the best (same) solution and, hence, the index value will be 0 (best index value) for all methods. The results of the experiments, averaged for each problem (10 data per average), are reported in Table 3. The best performance is obtained by the VNS-SA algorithm with the RDI of 9.91. In order to verify the statistical validity of the results, we performed an analysis of variance (ANOVA). The results demonstrate that there is a clear statistically significant difference between performances of our algorithm and other five algorithms, with respect to RDI criterion. The means plot and LSD intervals (at the 95% confidence level) for algorithms are shown in Fig. 5. It can be inferred from the Fig. 5 that the proposed VNS-SA statistically works better than other algorithms.
Regarding the results obtained from the computational study, it seems that the proposed VNS-SA algorithm can be an effective approach for TAP when the objective function is the minimization of the sum of the total execution and communication costs.

5- Conclusions and Future Research

In many problem domains, we are required to assign the tasks of an application to a set of distributed processors such that the incurred cost is minimized and the system throughput is maximized. In this paper, we developed a hybrid VNS-SA algorithm for solving TAP in a heterogeneous distributed computing system. Minimization of the sum of the total execution and communication costs was considered as the objective function. In the proposed hybrid meta-heuristic algorithm, VNS method worked as main algorithm and SA method supported it for attaining better solution. Employment of hybrid formation in the proposed

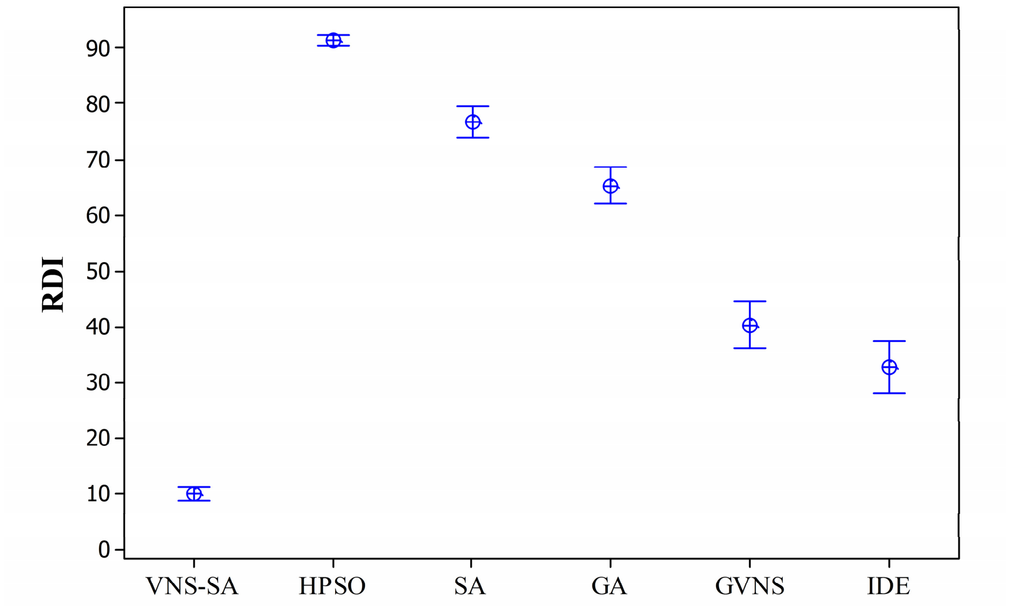


Fig. 5. Means plot and LSD intervals for algorithms, regarding RDI criterion

Table 1
Comparison of the presented VNS-SA algorithm with the five famous TAP algorithms on 24 test problems

Problem				Proposed hybrid algorithm				HPSO		SA		GA		GVNS		IDE	
No.	*r*	*n*	*d*	BC	AV(C)	AV(CPU)	CO %	BC	CO %	BC	CO %	BC	CO %	BC	CO %	BC	CO %
1	9	6	0.3	412	421.6	11.85	0	448	8.03	432	4.62	412	0	412	0	412	0
2			0.5	563	572.3	12.46	0	598	5.85	579	2.767	579	2.76	563	0	563	0
3			0.8	723	730.4	14.47	0	763	5.24	741	2.428	747	3.21	723	0	723	0
4	15	10	0.3	1134	1168.5	22.87	2.46	1295	14.59	1254	11.80	1212	8.74	1165	5.06	1106	0
5			0.5	1538	1598.6	21.55	0	1763	12.76	1737	11.45	1715	10.32	1651	6.84	1596	3.63
6			0.8	1732	1758.4	25.24	0	1922	9.88	1890	8.35	1874	7.57	1823	4.99	1804	3.99
7	20	10	0.3	2153	2176.5	34.71	4.50	2513	18.18	2472	16.82	2433	15.49	2056	0	2174	5.42
8			0.5	2969	3044.8	35.14	0	3378	12.10	3249	8.61	3310	10.30	3129	5.11	3104	4.34
9			0.8	4272	4414.2	34.93	0	4914	13.06	4890	12.63	4829	11.53	4756	10.17	4694	8.99
10	30	15	0.3	4520	4784.8	62.46	4.31	5632	23.20	5496	21.36	5071	14.71	4325	0	4569	5.34
11			0.5	5075	5422.6	66.29	0	6426	21.02	6278	19.16	6152	17.50	5682	10.68	5413	6.24
12			0.8	7419	7634.8	68.86	2.07	8624	15.75	8435	13.87	8104	10.35	7659	5.14	7265	0
13	40	20	0.3	9625	9711.7	161.93	0	11277	14.64	1111	13.41	1082	11.06	1022	5.87	9935	3.12
14			0.5	9549	9732.4	152.37	0.90	1101	14.08	1092	13.38	1076	12.07	1014	6.69	9463	0
15			0.8	12289	12528.5	172.59	0	1423	13.68	1399	12.20	1366	10.04	1324	7.21	1332	7.76
16	50	25	0.3	15226	15516.1	232.95	1.73	1664	10.10	1618	7.53	1591	5.97	1552	3.60	1496	0
17			0.5	19969	20108.3	265.31	1.49	2182	9.86	2141	8.13	2082	5.54	2013	2.32	1967	0
18			0.8	23747	24389.4	287.61	0	2638	9.99	2604	8.80	2586	8.18	2527	6.05	2492	4.74
19	60	30	0.3	29171	29556	304.89	0	3323	12.22	3287	11.27	3198	8.78	3123	6.59	3099	5.88
20			0.5	31052	31433.8	352.83	0	3499	11.26	3471	10.55	3431	9.49	3348	7.27	3315	6.35
21			0.8	34425	34982.4	334.59	0	3960	13.08	3927	12.35	3890	11.52	3824	9.98	3751	8.23
22	90	60	0.3	52624	52923	456.68	0	55234	4.725	5467	3.74	5414	2.80	5332	1.31	5318	1.06
23			0.5	61211	61982.8	443.92	0	6579	6.96	6505	5.91	6453	5.14	6378	4.04	6317	3.11
24			0.8	71532	72245.6	512.23	0	7928	9.77	7832	8.67	7751	7.71	7644	6.436	7574	5.56
				Average CO %			0.72		12.08		10.41		8.78		4.80		3.49

Table 2
VNS-SA versus the five famous TAP algorithms on 24 generated test problems

Approach	Better	Equal	Worse
HPSO	24	0	0
SA	24	0	0
GVNS	19	3	2
IDE	16	3	5

Table 3
Average relative percentage deviation $(\overline{RDI})$ for compared algorithms

Problem number	Algorithms					
	VNS-SA	HPSO	SA	GA	GVNS	IDE
1	10.21	88.29	54.25	19.14	13.82	5.31
2	8.15	80.7	47.36	43.85	17.54	13.15
3	6.16	83.33	46.66	57.5	16.66	8.33
4	20.47	92.76	74.2	55.2	31.67	7.23
5	12.02	94.64	84.32	75.59	50.19	45.03
6	5.32	88.3	75.4	68.95	44.35	52.98
7	18.52	88.92	81.94	75.29	11.07	31.17
8	7.43	90.09	64.8	72.74	41.27	50.087
9	8.35	87.72	84.9	77.73	69.15	61.86
10	21.98	93.88	84.75	56.21	6.11	22.49
11	10.7	91.62	82.5	74.73	45.77	29.2
12	16.52	92.87	80.94	60.06	31.98	7.129
13	2.23	92.53	84.24	69.15	38.41	29.45
14	8.91	88.91	84.44	76.31	45.15	11.08
15	4.43	86.1	77.2	64.78	49.29	55.93
16	21.1	93.36	69.58	55.7	35.52	6.63
17	13.74	90.27	74.83	52.87	27.22	9.72
18	10.87	94.63	83.05	77.09	57.17	55.55
19	4.49	97.42	89.08	68.17	50.63	45.14
20	4.71	98.69	91.84	81.81	61.51	53.33
21	5.37	96.05	89.67	82.55	69.74	55.62
22	4.59	95.51	72.76	56.48	31.44	27.26
23	7.66	91.03	80.87	70.44	59.67	46.54
24	4.09	94.46	83.54	74.17	62.25	53.89
Average	9.91	91.33	76.79	65.27	40.31	32.67

method had important effect on increasing power of intensification and diversification strategies in the quest of solution space. Also, six effective neighborhood structures were applied in various parts of the presented hybrid method for searching the solution space. The proposed algorithm was tested on 24 test problems. Experimental results show that the VNS-SA algorithm performs well on finding the optimal/near optimal task assignment, and it is a viable approach for TAP. In the following, suggestions are offered for future works:

- Developing another hybrid algorithms such as GA-VNS, VNS-PSO, PSO-SA for TAP;
- Investigation on another version of the TAP where each processor and each communication link has a failure ratio and the problem objective is to maximize the reliability for accomplishing the task execution;
- Developing efficiently intelligent neighborhood structures for better and more diverse searches in the solution space of TAP;

6- References

[1] M. Amiri, M. Zandieh, M. Yazdani, A. Bagheri, A variable neighbourhood search algorithm for the flexible job-shop scheduling problem. International journal of production research, 48(19), 5671–5689, 2010.

[2] S. M. Alaoui, O. Frieder, T. El-Ghazawi, A parallel genetic algorithm for task mapping on parallel machines. Lecture Notes in Computer Science, 949, 1999.

[3] A. Billionnet, M. C. Costa, A. Sutter, An efficient algorithm for a task allocation problem, Journal of ACM, 39, 502–518, 1992.

[4] V. Cerny, Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. Journal of Optimization Theory and Applications, 45(1), 41–51, 1985.

[5] G. H. Chen, J. S. Yur, A branch-and-bound-with-underestimates algorithm for the task assignment problem with precedence constraint, Proc. of the 10th International Conf. on Distributed Computing Systems, pp. 494–501, 1990.

[6] A. Ernst, H. Hiang, M. Krishnamoorthy, Mathematical programming approaches for solving task allocation problems, Proc. of the 16th National Conf. Of Australian Society of Operations Research, 2001.

[7] D. Fernandez-Baca, A. Medepalli, Parametric task allocation on partial k-trees, IEEE Trans Computers, 46, pp. 738-742, 1993.

[8] M. R. Garey, D. S. Johnson, Computers and Intractability: a Guide to the Theory of NP-Completeness. Freeman, New York, 1979.

[9] A. B. Hadj-Alouane, J. C. Bean, K. G. Murty, A hybrid genetic optimization algorithm for a task allocation problem. Journal of Scheduling, 2, 189–201, 1999.

[10] Y. Hamam, K.S. Hindi, Assignment of program tasks to processors: a simulated annealing approach, European Journal of Operational Research, 122, 509–513, 2000.

[11] P. Hansen, N. Mladenovic, Variable neighborhood search: principles and applications. European Journal of Operational Research, 130(3), 449–67, 2001.

[12] S. Y. Ho, H. S. Lin, W. H. Liauh, S. J. Ho, OPSO: orthogonal particle swarm optimization and its application to task assignment problems. IEEE Transactions on Systems, Man, and Cybernetics - Part A, 38(2), 288–298, 2008.

[13] M. Kafil, and I. Ahmad, Optimal task assignment in heterogeneous distributed computing systems, IEEE Concurrency, 6, pp. 42–50, 1998.

[14] S. Kirkpatrick, C.D. Gelatt, M. P. Vecchi, Optimization by simulated annealing. Science, 220, 671–680, 1983.

[15] J. Kratica, A. Savić, V. Filipović, M. Milanović, Solving the Task Assignment Problem with a Variable Neighborhood Search. Serdica Journal of Computing, 4(4), 435–446, 2010.

[16] C. H. Lee, K. G. Shin, Optimal task assignment in homogeneous networks, IEEE Trans Parallel and Distributed Systems, 8, pp. 119–129, 1997.

[17] F. T. Lin, C. C. Hsu, Task assignment scheduling by simulated annealing, IEEE Region 10 Conference on Computer and Communication Systems, pp. 279–283, 1990.

[18] A. Lusa, C. N. Potts, A variable neighbourhood search algorithm for the constrained task allocation problem. Journal of the Operational Research Society, 59 (6), 812–822, 2008.

[19] N. Mladenovic, P. Hansen, Variable neighborhood search. Computers and Operations Research, 24(11), 1097–1100, 1997.

[20] N. Mladenovic, A variable neighborhood algorithm—a new metaheuristic for combinatorial optimization. Abstracts of papers presented at Optimization Days. Montréal, pp. 112, 1995.

[21] R. Rabieyan, M. Seifbarghi, Maximal Benefit location problem for a congested system, Journal of Industrial Engineering, 4(5), 73-83, 2010.

[22] S. Salcedo-Sanz, Y. Xu, X. Yao, Hybrid meta-heuristics algorithms for task assignment in heterogeneous computing systems. Computers & Operations Research, 33, 820–835, 2006.

[23] A. Salman, I. Ahmad, S. Al-Madani, Particle swarm optimization for task assignment problem. Microprocessors and Microsystems, 26 (8), 363–371, 2002.

[24] H. S. Stone, Multiprocessor scheduling with the aid of network flow algorithms. IEEE Transactions on Software Engineering, 3 (1), 85–93, 1977.

[25] E. G. Tabi, , T. Muntean, Hill-climbing, Simulated Annealing and Genetic Algorithms: a Comparative Study and Application to the Mapping Problem, IEEE 26th Hawaii International Conference System Sciences, pp. 565–573, 1993.

[26] A. K. Tripathi, B. K. Sarker, N. Kumar, A GA based multiple task allocation considering load, International Journal of High Speed Computing, 203–214, 2000.

[27] S. H. Woo, S. B. Yang, S. D. Kim, T. D. Han, Task scheduling in distributed computing systems with a genetic algorithm. In: Proceedings of High Performance on Information superhighway, pp. 301–305, 1997.

[28] Q. Y. Yang, C. J. Wang, C. S. Zhang, An efficient discrete particle swarm algorithm for task assignment problems. In: Granular Computing, GRC '09, pp. 686–690, 2009.

[29] M., Yazdani, M. Amiri, M. Zandieh, Flexible job-shop scheduling with parallel variable neighborhood search algorithm. Expert Systems with Applications, 37 (1) 678–687, 2010.

[30] M. Yazdani, M. Gholami, M. Zandieh, M. Mousakhani, A simulated annealing algorithm for flexible job-shop scheduling problem. Journal of applied sciences, 9(4), 662–670, 2009.

[31] P. Y. Yin, S. S. Yu, P. P. Wang, Y.T. Wang, A hybrid particle swarm optimization algorithm for optimal task assignment in distributed systems. Computer Standards & Interfaces, 28, 441–450, 2006.

[32] P. Y. Yin, B. B. M. Shao, Y. P. Cheng, and C. C. Yeh, Metaheuristic algorithms for Task Assignment in Distributed Computing Systems: A Comparative and Integrative Approach. The Open Artificial Intelligence Journal, 3, 16–26, 2009.

[33] D. Zou, H. Liu, L. Gao, S. Li, An improved differential evolution algorithm for the task assignment problem, 24(4), 616-624, 2010.