IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-5, Issue-4 E-ISSN: 2347-2693

Optimizing Virtual Machine Placement Using Intelligent Water Drop and Simulating Algorithm

V.O. Ramakant[1*], A. Victor[2]

[1*]School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu, India
[2] School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu, India

[*]*Corresponding Author: omkar.vaidya9@gmail.com, Tel.: +91-97434-58979*

www.ijcseonline.org

Received: 02/Mar/2017, Revised: 16/Mar/2017, Accepted: 14/Apr/2017, Published: 30/Apr/2017

Abstract— Developing approaches intended to produce top notch answers for tackle troublesome computational enhancement issues by playing out a pursuit over the space of heuristics as opposed to looking the arrangement space specifically. Significant progress in developing search methodologies for a huge variety of application areas still require specialists to integrate their expertise in every problem domain. Researchers have need for developing automated systems to replace the role of a human expert. A hyper-heuristic for the most part goes for diminishing the measure of area information in the inquiry system. Coming about approach ought to be shabby and quick to execute, requiring less mastery in either the issue area or heuristic techniques and it would be vigorous. Resulting approach is cheap and fast to implement, requiring less expertise in either the problem domain as well as hyper heuristic methods and it would be robust.

Keywords— Hyper-Heuristic Algorithms, Virtual Machine Placement problem, Genetic Algorithms, Energy efficient

I. Introduction

In distributed computing, Virtual Machine (VM) situation is a basic operation which is directed as a major aspect of the VM movement and meant to locate the best Physical Machine (PM) to have the VMs. It directly affects the execution, asset usage and power utilization of the server farms and can lessen the upkeep cost of the server farms for cloud suppliers. Various VM arrangement plans are planned and proposed for VM position in the distributed computing condition expected to enhance different components influencing the server farms, the VMs and their executions.

1.1 Virtual Machine Placement in Cloud Data Centers: Distributed computing, another figuring stage in which clients can obtain and discharge the assets on request from a Web program, ends up noticeably a standout amongst the most dangerously extending advances in the processing business today. Subsequently, the number and the size of Cloud specialist co-ops have extraordinarily expanded. More server farms mean more vitality supply, more system use, and causes expanded warmth dissemination, lessened computational thickness, and higher working expenses.

The utilization of workload combination to clear physical server hubs to enhance framework proficiency has been as of now exhibited in various works. A key issue in workload solidification is to delineate VMs to physical machines (PMs). Numerous past works have figured the VM mapping issue as a multi-dimensional container pressing issue. Each measurement speaks to an especially asset kind of a VM ask for, the objective is to use as less canister as conceivable to satisfy all the VM asks. The issue is NP-hard and can be illuminated by some heuristic techniques, for example, first-fit or best fit, be that as it may, those strategies disregard the dynamic conduct of workload.

Xinying Zheng and Yu Cai propose another element VM position plot that can progressively and viably outline VM solicitations to PMs while sparing vitality. They build a VM/PM mapping likelihood framework, in which each VM ask for is doled out with a likelihood running on a PM. The VM/PM mapping likelihood lattice consider of asset necessities, virtualization overhead, control productivity and in addition server dependability. Our plan then chooses where to execute another occupation, and whether to move existing employments to enhance worldwide framework proficiency [1].

Meng Wang, Xiaoqiao Mengy, and Li Zhangy utilize arbitrary factors to describe the future transfer speed use. They utilize arbitrary factors which take after specific dispersions that are evaluated from either verifiable movement rates or guaging calculations. Such a probabilistic portrayal can better speak to the vulnerability without bounds transmission capacity request [2].

Joe Wenjie Jiang and TianLan concentrate on the administration of system assets by abusing joint course determination and VM arrangement. They formalize it as a streamlining issue, in which, givena arrangement of employment landings, the system administrator needs settle on the directing and position choices to limit the system clog over the long haul [3].

1.2 Hyper Heuristic: Some genuine issues are perplexing. Due to their (regularly) NP-hard nature, specialists and experts as often as possible turn to issue custom-made heuristics to acquire a sensible arrangement in a sensible measure of time. Hyper-heuristics are rising systems intended to produce great answers for take care of troublesome computational enhancement issues by playing out an inquiry over the space of heuristics as opposed to seeking the arrangement space specifically. One of their principle points is to raise the level of all-inclusive statement of pursuit strategies, and to consequently adjust the calculation by consolidating the quality of every heuristic and compensating for the shortcomings of others.

1.3 Hyper-heuristics Classification: Two sorts of hyper-heuristic techniques can be recognized in the writing: (i) heuristic determination systems (ii) heuristic era strategies from given parts. For both hyper-heuristic philosophies, there are two perceived sorts of heuristics: (i) valuable heuristics which handle an incomplete arrangement and manufacture an entire arrangement (ii) perturbative heuristics which work on total arrangement. The documentation of useful and perturbative shows how the pursuit through the arrangement space is overseen by the low-level heuristics [5].

An orthogonal classification of hyper-heuristics is provided in [6] (see Figure 1) depending on: (i) the nature of the heuristic search space and (ii) the source of feedback during the search process. Hyper-heuristics can be used to select or generate constructive or perturbative heuristics which determine the nature of the heuristic search space.

A hyper-heuristic can employ no learning, online learning (getting feedback from the search process while solving an instance), or offline learning (getting feedback via training over a selected set of instances to be utilized for solving unseen instances). A hyper-heuristic which combines simple random heuristic selection with a method of accepting improving and equal quality moves is an example which uses a no learning approach [10].

II. RELATED WORK

2.1 Multi-objective Hyper-Heuristics Approaches: Hyper-heuristics have recently seen an increase in attention from researchers. Although many hyper-heuristics papers have been published, they are still mainly limited to single-objective optimization. The hyper heuristics for multi-target advancement issues is another territory of research in Evolutionary Computation and Operational Research ([10], [5]). To date, few reviews, have been recognized that arrangement with Hyper-Heuristics for multi-target issues (see Table 2).

The primary approach is a multi-objective Hyper Heuristic in view of Tabu pursuit [11]. The key component of this paper lies in picking a reasonable heuristic at every cycle to handle the current issue by utilizing Tabu hunt as an abnormal state look methodology. The proposed approach was connected to space portion and timetabling issues and created comes about with worthy arrangement quality.

A versatile multi-method (multi-point) seek called Amalgam is proposed in [20]. It utilizes different hunt calculations; NSGAII, PSO, AMS, and DE at the same time utilizing the ideas of multi-strategy seek and versatile posterity creation. AMALGAM is connected to a few ceaseless multi-target test issues and it was better than different techniques. It was additionally connected to take care of a few water asset issues and it yielded great arrangements ([20], [21], [12]) display a multi-objective hyper-heuristic approach including two stages: the main stage expects to deliver an effective Pareto front (this might be of low quality in view of the thickness), while the second stage means to manage a given issue adaptably to drive a subset of the populace to the coveted Pareto front.

This approach was assessed on the multi-target voyaging sales representative issues with eleven low level heuristics. It is contrasted with other multi-objective methodologies from the writing which uncovers that the proposed approach produces great quality outcomes yet future work is yet expected to enhance the strategy [12].

In [19], they propose a hypervolume-based hyper-heuristic for an element mapped multi-target island-based model. The proposed technique demonstrates its prevalence when analyzed over the commitment based hyper-heuristic and other standard parallel models over the WFG test.

Another hyper-heuristic in view of the multi-objective transformative calculation NSGAII is proposed in [14]. The principle thought of this technique is in creating the last Pareto ideal set, through a learning procedure that advances mixes of condition-activity rules in view of NSGAII. The proposed technique was tried on many examples of sporadic 2D cutting stock benchmark issues and created promising outcomes.

A multi-procedure group multi-objective transformative calculation called MS-MOEA for element enhancement is proposed in [23]. It consolidates distinctive procedures including a memory system and hereditary and differential administrators to adaptively make posterity and accomplish quick merging velocity. Exploratory outcomes demonstrate that MS-MOEA can acquire promising outcomes.

In [13] an online determination hyper-heuristic, Markov chain based, (MCHH) is examined. The Markov chain controls the determination of heuristics and applies online support figuring out how to adjust move weights between heuristics. In MCHH, half and half meta-heuristics and Evolution Strategies were consolidated and connected to the DTLZ test issues. The MCHH has additionally been connected to genuine water dispersion systems outline issues and delivered aggressive outcomes.

In [15], a hyper-heuristic-based codification is proposed for comprehending strip pressing and cutting stock issues with two destinations that expand the aggregate benefit and limit the aggregate number of cuts. Trial comes about demonstrate that the proposed Hyper-Heuristic out performs single heuristics.

In [16], a multi-objective hyper-heuristic for the plan and advancement of a stacked neural system is proposed. The proposed approach depends on NSGAII consolidated with a nearby pursuit calculation (Quasi-Newton calculation).

In [24], creator introduced a multi-objective hyper-heuristic enhancement conspire for building framework outline issues. A hereditary calculation, recreated tempering and molecule swarm enhancement are utilized as low-level heuristics.

In [17], creator proposed a multi-pointer hyper-heuristic for multi-target improvement. This was approach in light of various rank markers that taken from NSGAII, IBEA and SPEA2.
In [19], creator proposed a hypervolume-based hyper heuristic for an element mapped multi-target island-based model.

In [25], creator proposed a different neighborhood hyper-heuristic for two-dimensional rack space portion issue. The proposed hyper-heuristic depended on a reenacted toughening calculation.

In [18], creator introduce a multi-objective hyper-heuristic hereditary calculation (MHypGA) for the arrangement of Multi-Objective Software Module Clustering Problem. In MHypGA, distinctive techniques for choice, hybrid and change operations of hereditary calculations are consolidated as a low-level heuristic.
The canister pressing issue is a combinatorial NP-difficult issue. In it, objects of various volumes must be stuffed into a limited number of canisters of limit in a way that limits the quantity of receptacles utilized. Numerous varieties of this issue are available, for example, 2D pressing, direct pressing, pressing by weight, pressing by cost, et cetera. The uses of these issues incorporate, topping off compartments, stacking trucks with weight limit, and making record reinforcement in removable media.

The VM situation issue can be composed as a canister pressing issue. The Physical machines can be considered as containers and the VM's to be set can be considered as articles to be led in the receptacle. Powerful situation of virtual machines in a group of physical machines is fundamental for advancing the utilization of computational assets and decreasing the likelihood of virtual machine reallocation. Large portions of past works regard virtual machine situation as an occurrence of the canister pressing issue, as they go for sparing vitality.

III. ARCHITECTURE MODEL

3.1 System Architecture:

The Hyper heuristics system architecture is segregated in three levels. Base level contains problem description, representation, evolution function, initial solution of the problem. Base level also contains Heuristics repository. Heuristics repository holds all the heuristics algorithm to be available to solve the problem. Domain Barrier is the Bridge between Hyper level and Base Level. Domain Barrier will maintain the synchronization between the Base and Hyper Level. Hyper Level first collect all the data required for the specific problem. Hyper Level contains the mechanism to select the algorithm and apply the selected heuristic and take back the feedback. The Hyper level will store the result of the heuristic mechanism for the problem to compare the mechanism with other heuristic mechanism.

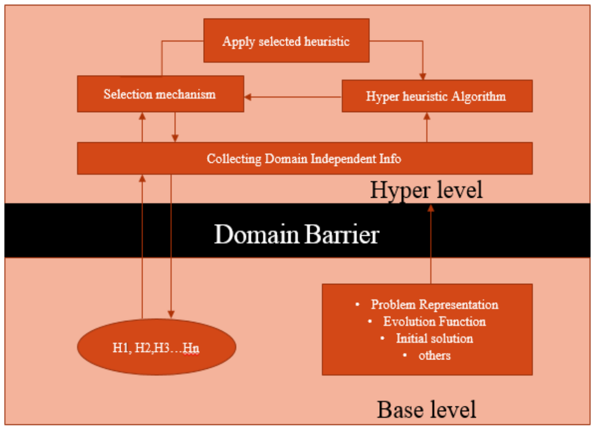

Figure 3.1: Hyper-Heuristics System Architecture

In dissertation work it is proposed that, attempt will be made to optimize virtual machine placement by using hyper heuristic framework.

3.2 Objectives of Project:

1.To develop Hyper-Heuristic Framework for single objective virtual machine placement

a. Identifying suitable low level heuristics (local search / global search techniques) to solve Virtual Machine Placement problem.
b. Designing problem specific low level heuristic.
c. Distinguishing reasonable abnormal state heuristic to pick rectify low level heuristic at every choice point.

2. To develop Hyper-Heuristic Framework for multi objective virtual machine placement
   a. Identifying suitable low level heuristics (local search / global search techniques) to solve Virtual Machine Placement problem.
   b. Designing problem specific low level heuristic.
   c. Distinguishing reasonable abnormal state heuristic to pick remedy low level heuristic at every choice point.
3. Possible Output: Hyper heuristic framework for single and multi-objective virtual machine placement problem.

IV. METHODOLOGY

4.1 Intelligent Water Drop Algorithm(IWD)

Water drops that stream in waterways, lakes, and oceans are the wellsprings of motivation for building up the IWD. This insight is more evident in waterways which discover their approaches to lakes, oceans, or seas regardless of numerous sorts of impediments on their ways. In the water drops of a stream, the gravitational constrain of the earth gives the inclination to streaming toward the goal. On the off chance that there were no hindrances or obstructions, the water drops would take after a straight way toward the goal, which is the most limited way from the source to the goal. In any case, because of various types of snags in their way to the goal, which compel the way development, the genuine way should be unique in relation to the perfect way and loads of wanders aimlessly in the stream way is watched. The fascinating point is this developed way is by all accounts ideal as far as separation from the goal and the requirements of the earth.

Envision a water drop will move from a state of waterway to the following point in the front. It is accepted that each water drop streaming in a waterway can convey a measure of soil which is appeared by the extent of the water drop in the figure. The measure of soil of the water drop increments as it ranges to the correct point while the dirt of the stream bed diminishes. Truth be told, some measure of soil of the stream bed is evacuated by the water drop and is added to the dirt of the water drop. This property is installed in the IWDs with the end goal that each IWD holds soil and expels soil from its way amid development in the earth. A water drop has likewise a speed and this speed assumes a vital part in the expelling soil from the beds of waterways. Let two water drops having a similar measure of soil move from a state of a waterway to the following point. The water drop with greater bolt has higher speed than the other one. At the point when both water drops touch base at the following point on the privilege, the quicker water drop is expected to accumulate more soil that the other one. The specified property of soil expelling which is subject to the speed of the water drop is inserted in each IWD of the IWD calculation.

- Step 1: Set the quantity of water drops NIWD to a positive whole number esteem. Here, it is proposed that NIWD is set equivalent to the quantity of things Nc. For speed refreshing, the parameters are set as av = 1, bv = 0.01, and cv = 1. For soil refreshing, as = 1, bs = 0.01, and cs = 1. The neighborhood soil refreshing parameter rn, which ought to be a little positive number short of what one, is picked as rn = 0.9. The worldwide soil refreshing parameter rIWD, which ought to be looked over [-1, 0], is set as rIWD = - 0.9. In addition, the underlying soil on every way is indicated by the steady InitSoil with the end goal that the dirt of the way between each two things i and j is set by soili; j = InitSoil: The underlying speed of IWDs is meant by the consistent InitVel. Both parameters InitSoil and InitVel are likewise client chose.
- Step 2. For each IWD, a went by hub list Vc[IWD] is viewed as and is set to the unfilled rundown. The speed of each IWD is set to InitVel and the underlying soil of each IWD is set to zero.
- Step 3. For each IWD, arbitrarily select a hub and partner the IWD to this hub.
- Step 4. Refresh the went by hub rundown of each IWD to incorporate the hubs just went to.
- Step 5. For each IWD that has not finished its answer, rehash Steps 5.1-5.4.
- Step 5.1. Pick the following hub j to be gone by the IWD among those that are not in its went to hub list and don't disregard the m imperatives characterized in condition (3). At the point when there is no unvisited hub that does not abuse the limitations, the arrangement of this IWD has been finished. Something else, pick next hub j when the IWD is in hub i with the likelihood pIWD(i,j) characterized in beneath condition and refresh its went to hub list.

$$p_i^{IWD}(j) = \frac{f(soil(i,j))}{\sum_{k \notin vc(IWD)} f(soil(i,k))} \tag{4.1}$$

$$f(soil(i,j)) = \frac{1}{\varepsilon_s + g(soil(i,j))} \tag{4.2}$$

$$\begin{aligned} g(soil(i,j)) &= (soil(i,j)) \text{ if } \min((soil(i,j))) \geq 0 \\ &= (soil(i,j)) - \min((soil(i,l))) \end{aligned} \tag{4.3}$$

$$g(soil(i,j)) = \begin{cases} (soil(i,j)) \\ \text{if } \min((soil(i,l))) \geq 0 \\ (soil(i,j)) - \min((soil(i,l))) \\ \text{else} \end{cases} \tag{4.4}$$

Algorithm 1: Intelligent Water Drop Algorithm
Input: Weight of object i., No. of bin j.
Output: Minimum no. of bins required to carry all objects.

```
1. Initializing of static arguments
2. Initializing of dynamic arguments
3. For each IWD, arbitrarily choose a node and associate the IWD to this node.
4. Refresh the visited node list of every IWD to include the nodes just visited.
5. For each IWD that has not completed its solution, repeat Steps
    1) Choose the next node j to be visited from non-visited node list
    2) For every IWD shifting from node i to node j, update its velocity
    3) Compute the amount of the soil that the current water with the updated velocity.
    4) Refresh the soil of the path crossed by that IWD.
6. Find the repetition-best solution
7. Update the soils of the paths that exist in the current iteration-best solution
8. Update the complete best solution
9. Go to Step 2 until the maximum number of iterations is reached.
10. The algorithm stops with the final solution.
```

- Step 5.2. For each IWD moving from hub i to hub j, refresh its speed velIWD(t) with the end goal that

$$\Delta soil(i,j) = \frac{a_s}{b_s + c_s * time^2(i,j,vel^{IWD})}$$

(4.5)

- Step 5.3. Register the measure of the dirt, soil(i; j), that the ebb and flow water drop IWD with the refreshed speed velIWD = velIWD(t + 1) loads from its ebb and flow way between two hubs i and j

$$\Delta soil(i,j) = \frac{a_s}{b_s + c_s * time^2(i,j,vel^{IWD})}$$

(4.6)

Such that,

$$time(i,j,vel^{IWD}) = \frac{W_j}{pft_j * vel^{IWD}}$$

(4.7)

- Step 5.4. Refresh the dirt of the way navigated by that IWD, soil(i, j), and the dirt that the IWD conveys, soilIWD,

$$soil(i,j) = (1-p_n) * soil(i,j) - p_n * \Delta soil(i,j)$$

(4.8)

$$soil^{IWD} = soil^{IWD} + \Delta soil(i,j)$$

(4.9)

- Step 6. Find the iteration-best solution TIB from all the solutions found by the IWDs.
- Step 7. Refresh the dirts of the ways that exist in the present emphasis best arrangement TIB

$$soil(i,j) = (1-p_{IWD}) * soil(i,j) + p_{IWD} * \frac{1}{N_c - 1} * soil_{IB}^{IWD} \, \forall (i,j) \in T^{IB}$$

(4.10)

- Step 8. Refresh the aggregate best arrangement TTB by the present cycle best arrangement TIB
- Step 9. Go to Step 2 until the maximum number of iterations is reached.
- Step 10. The calculation stops here with the last arrangement TTB.

4.2 Simulated Annealing

Toughening is the way toward cooling material in a warmth shower. The material is warmed to high vitality where there are visit state changes. It is then continuously cooled to a low vitality state where state changes are uncommon. Simulated annealing (SA) emulates this physical process whereby the material is slowly cooled until a steady state is reached. Simulated annealing can be understood as an extension of the simple random gradient descent algorithm. [33] recommended that mimicked toughening could be utilized to scan for arrangements in an advancement issue whose goal is to merge to an ideal state.

SA repeatedly considers neighbours w' of the current solution w and probabilistically decides between changing to w' or staying with w. Normally, enhancing moves are constantly acknowledged while exacerbating moves can be acknowledged probabilistically in light of a capacity P of the temperature t and assessment distinction between f(w') and f(w). The temperature is proportionate with the probability of acceptance; i.e. high temperature means high acceptance probability and vice versa. The temperature is gradually reduced as the algorithm proceeds. Acknowledgment likelihood is figured as e^(( )/t) where is the size of f(w') - f(w) (the distinction between the present and new arrangement assessment capacity) and t is the present estimation of the temperature parameter. The pseudo-code in Algorithm 5 presents a simulated annealing that iterates through M iterations. The temperature is calculated as a function of the remaining number of iterations. [32] demonstrated the significance of setting the cooling plan (begin temperature, end temperature, temperature diminishment) and the area structure to the aftereffect of SA. The temperature is set at a high value at the beginning to allow more worse moves and then it is slowly reduced to reach equilibrium.

Algorithm 2: Simulated Annealing Algorithm
Input: Weight of object i.,No. of bin j.
Output: Minimum no. of bins required to carry all objects.

```
1. Choose an introductory solution = (x1,..., xn) $\in \Omega$; an introductory temperature t = t0 ; control argument
   value α; finishing temperature e; a iterative schedule, M that describes total number of repetitions carried out at every temperature;
2. Incumbent solution ← f(w);
3. Repeat
4.     Make iteration counter m = 0;
5.     Repeat;
6. Choose an integer i from the set f1,2,....,ng arbitrarily;
7. If $x_i$ = 0, select item i, i.e. make $x_i$ = 1, retrieve new solution w' then
8.         While solution w' is impractical, do
9.          Create other item w;' from arbitrarily; highlight the new solution as w' ;
10.         Let $\Delta$ = f(w') - f(w);
11.         While ( $\Delta \geq 0$ ) or ( Random (0, 1) < $e^{\frac{\Delta}{t}}$ ), do w ← w';
12. Else
13.         Drop item i, and select other item arbitrarily, find new solution w';
14.       Let $\Delta$ = f(w') - f(w);
15.         While ( $\Delta \geq 0$ ) or ( Random (0, 1) < $e^{\frac{\Delta}{t}}$ ), do w ← w';
16. End if
17. If incumbent solution < f(w), Incumbent solution   f(w);
18. m++;
19.        Till m = M
20.        Make t = a * t;
21. Till t < e.
```

V. Results and Discussion

In this phase work, implementation of the IWD and SA approach solve single-objective virtual machine placement problem is completed. There is objective such as minimize server resources. In that implementation, I refer dataset of 100 physical machines and 250 virtual machine datasets which is generated by given specification. Table 5.1 shows implementation IWD and SA for various iterations. SA improves result iteration wise. IWD take longer execution time than SA because its heavy structure.

Table 5.1: IWD and SA approach solve VMP problem based on Optimization Server Resources

Iteration	Intelligent Water Drop	Simulated Annealing	Execution time(sec)for IWD	Execution time(sec)for SA
50	53	58	195.325	128.356
100	52	55	256.429	246.658
500	52	53	1245.518	1103.35
700	49	53	1853.2548	1523.931
1000	51	50	2637.6314	2267.4962

Above table, represent the result of IWD and SA implementation for optimizing server resources where main goal is minimizing number of virtual machine requirement. Intelligent water drop algorithm is global search algorithm whereas SA is local search algorithm. SA improve result after iteration increasing. IWD works better than SA for optimizing server resources.

Table 5.2: IWD and SA approach solve VMP problem based on Different size of VMs

VMs size	Intelligent Water Drop	Simulated Annealing	Execution time(sec)for IWD	Execution time(sec)for SA
50	17	18	153.256	102.631
100	23	24	193.382	99.568
150	33	35	211.35	154.315
200	41	42	1532.846	1025.3482
250	49	50	2356.148	1567.243

The above table 5.2 shows the IWD approach solve virtual machine placement problem based on different size of VMs groups. They have distinctive CPU estimate, memory limit and system transfer speed. We IWD and SA algorithm with base paper implementation of random, BFD and BF-HC algorithms.

Figure 5.1 demonstrates the quantity of dynamic PMs for different sorts of calculations, BFD and BF-HC require less PMs than arbitrary calculations. The consequence of BFD is superior to BF-HC, because BFD is composed per the extent of VMs while BF-HC is intended for activity accumulation between the VMs, however BF-HC requires practically an indistinguishable number of PMs from BFD [31].

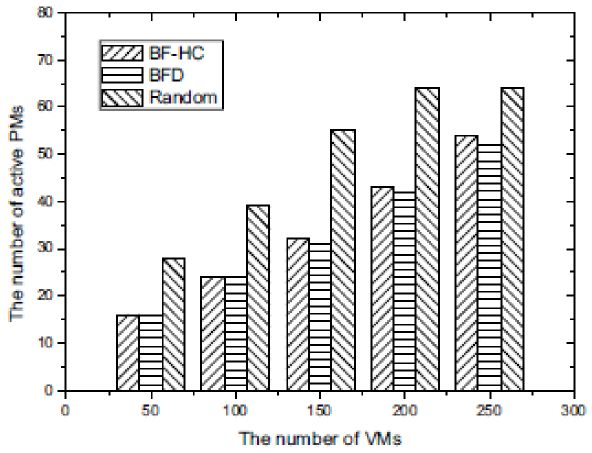

Figure 5.1: Traditional Algorithm Comparison

We also analyses performance of IWD and SA algorithm for various size virtual machines. We perform it on number of VMs like 50, 100, 150, 200 and 250. We observe that these to heuristic algorithm work efficiently for single objective virtual machine placement problem.

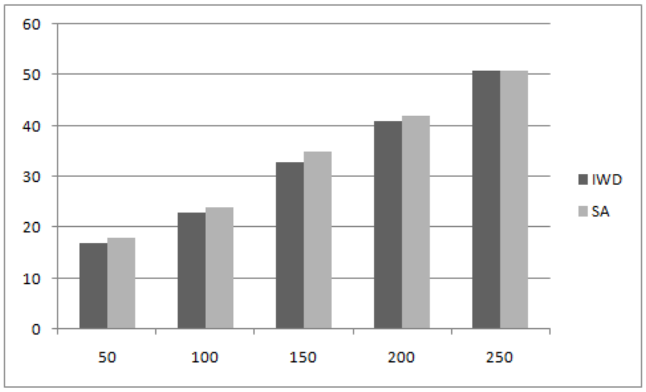

Figure 5.2: Active PMs number in IWD and SA algorithm

VI. CONCLUSION AND FUTURE SCOPE

We focused on study of multi-objective bin packing problem and Single-objective virtual machine placement problem using IWD and SA. Bin packing problem is basic and very relative domain for virtual machine placement problem. We first analyses algorithm performance for bin packing. Both IWD and SA works efficiently for VMP. We choose these two heuristics to analyses hyper-heuristic framework performance for different types heuristic. IWD is work in global search manner where SA is local search algorithm.

The results of SA improve its outcomes iteration after iteration. It forward only best available outcome except in high temperature regime. Whereas IWD is global search algorithm. Per results of bin packing problem, algorithm works efficient for minimizing bins but it performance is somewhat poor for minimizing heterogeneity. Per results of virtual machine placement problem, IWD provide best result with 49 PMs and SA provide best result with 50 Pms. IWD performance is quite better than SA algorithm.

REFERENCES

[1] N. Bobroff, A. Kochut, K. Beaty, "Dynamic Placement of Virtual Machines for Managing SLA Violations", *10th IFIP/IEEE International Symposium on Integrated Network Management*, Munich, pp.119-128, 2007.

[2] M. Wang, X. Meng, L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers", *Proceedings IEEE INFOCOM*, Shanghai, pp. 71-75, 2011.

[3] J. W. Jiang, T. Lan, S. Ha, M. Chen, M. Chiang, "*Joint VM placement and routing for data center traffic engineering*", *Proceedings IEEE INFOCOM*, Orlando, pp. 2876-2880, 2012.

[4] Cowling, Peter, Graham Kendall, Eric Soubeiga, "*A hyperheuristic approach to scheduling a sales summit*", International Conference on the Practice and Theory of Automated Timetabling, Berlin Heidelberg, pp. 176-190, 2000.

[5] K. Edmund, "*Hyper-heuristics: A survey of the state of the art*", Journal of the Operational Research Society, Vol.64, Issue.12, pp.1695-1724, 2013.

[6] K. Edmund, "*A classification of hyper-heuristic approaches", Handbook of metaheuristics*, Springer US, pp.449-468, 2010.

[7] E. Ozcan, Y. Bykov, M. Birben, E. K. Burke, "*Examination timetabling using late acceptance hyper-heuristics*", *2009 IEEE Congress on Evolutionary Computation*, Trondheim, pp. 997-1004, 2009.

[8] M. Ayob, G. Kendall, "*A Monte Carlo hyper-heuristic to optimize component placement sequencing for multi head placement machine*", In *Proceedings of the International Conference on Intelligent Technologies (ISTRD)*, Thailand, pp 132–141, 2003.

[9] M Misir, W Vancroonenburg, K Verbeeck, GV. Berghe, "*A selection hyper-heuristic for scheduling deliveries of ready-mixed concrete*", In Proceedings of the Metaheuristics International Conference, Itly, pp.289-298, 2011.

[10] Özcan, Ender, Burak Bilgin, EE Korkmaz, "*A comprehensive analysis of hyper-heuristics*", Intelligent Data Analysis, Vol.12, Issue.1, pp.3-23, 2008.

[11] EK Burke, G Kendall, E Soubeiga, "*A tabu-search hyperheuristic for timetabling and rostering*", Journal of Heuristics, Vol. 9, Issue.6, pp.451-470, 2003

[12] Nadarajen Veerapen, Dario Landa-Silva, Xavier Gandibleux, "*Hyperheuristic as Component of a Multi-Objective Metaheuristic*", Do toral Symposium on Engineering Sto hasti Lo al Sear h Algorithms, Belgium, pp.51-55, 2009.

[13] K McClymont, EC Keedwell, "*Markov chain hyper-heuristic (MCHH): an online selective hyper-heuristic for multi-objective continuous problems* ", Proceedings of the 13th annual conference on Genetic and evolutionary computation, Ireland, pp. 2003-2010, 2011.

[14] J Gomez, H Terashima-Marín, "*Approximating multi-objective hyper-heuristics for solving 2d irregular cutting stock problems*", Mexican International Conference on Artificial Intelligence, Berlin, pp.349-360, 2010.

[15] G. Miranda, J. de Armas, C. Segura, C. León, "*Hyperheuristic codification for the multi-objective 2D Guillotine Strip Packing Problem*", *IEEE Congress on Evolutionary Computation*, Barcelona, pp.1-8, 2010.

[16] Furtuna Renata, Silvia Curteanu, Florin Leon, "*Multi-objective optimization of a stacked neural network using an evolutionary hyper-heuristic*", Applied Soft Computing, Vol.12, Issue.1, pp.133-144, 2012.

[17] Vázquez Rodríguez, José Antonio, Sanja Petrovic, "*Calibrating continuous multi-objective heuristics using mixture experiments*", Journal of Heuristics, Vol.18, Issue.5, pp.699-726, 2012.

[18] A.C. Kumari, K. Srinivas M.P. Gupta, "*Software module clustering using a hyper-heuristic based multi-objective genetic algorithm*", 3rd IEEE International Advance Computing Conference (IACC), Ghaziabad, pp. 813-818, 2013.

[19] C León, G Miranda, C Segura, "*Hyperheuristics for a dynamic-mapped multi-objective island-based model*", International Work-Conference on Artificial Neural Networks, Berlin, pp.41-59, 2009.

[20] JA Vrugt, BA Robinson, "*Improved evolutionary optimization from genetically adaptive multimethod search*", National Academy of Sciences, Vol.104, Issue.3, pp.708-711, 2007.

[21] Raad, Darian, Alexander Sinske, JV Vuuren, "*Multiobjective optimization for water distribution system design using a hyperheuristic*", Journal of Water Resources Planning and Management, Vol.136, Issue.5, pp.592-596, 2010.

[22] X Zhang, R Srinivasan, MV Liew, *"On the use of multi-algorithm, genetically adaptive multi-objective method for multi-site*

calibration of the SWAT model", Hydrological Processes, Vol.24, Issue.8, pp.955-969, 2010.

[23] Yu Wang, Li Bin, "*Multi-strategy ensemble evolutionary algorithm for dynamic multi-objective optimization*", Memetic Computing, Vol.2, Issue.1, pp.3-24, 2010.

[24] AF Rafique, S Sivasundaram, "*Multiobjective hyper heuristic scheme for system design and optimization*", AIP Conference Proceedings-American Institute of Physics, Vol.1493, No.1, pp.764-769, 2012.

[25] Ruibin Bai, "*A new model and a hyper-heuristic approach for two-dimensional shelf space allocation*", 4OR, Vol.11, Issue.1, pp.31-55, 2013.

[26] Ruibin Bai, "*An investigation of novel approaches for optimising retail shelf space allocation*", Docotral Disssertation of University of Nottingham, Nottingham, pp.1-220, 2005.

[27] E López-Camacho, H Terashima-Marin, P Ross, "*A unified hyper-heuristic framework for solving bin packing problems*", Expert Systems with Applications, Vol.41, Issue.15, pp.6876-6889, 2014.

[28] Peter Ross, "*Hyper-heuristics: learning to combine simple heuristics in bin-packing problems",* Proceedings of the 4th Annual Conference on Genetic and Evolutionary Computation, NY, pp.942-948,2002.

[29] Matthew Hyde, "*A genetic programming hyper-heuristic approach to automated packing*", Docotral Dissertation University of Nottingham, Nottingham, pp.1-175, 2010.

[30] Kevin Sim, "*Novel Hyper-heuristics Applied to the Domain of Bin Packing*", Docotral Dissertation of Edinburgh Napier University, singapore, pp.1-149, 2014.

[31] J. Dong, X. Jin, H. Wang, Y. Li, P. Zhang, S. Cheng, "*Energy-Saving Virtual Machine Placement in Cloud Data Centers",* 2013 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, Netherlands, pp. 618-624, 2013.

Authors Profile

Mr.Omkar R Vaidya is student School of Computing Science and Engineering at V.I.T. University, Vellore, India. He received his BE (Information Technology) from Shivaji University and pursuing M.Tech. (Computer Science and Engineering) from Vellore Institute of Technology. He has participated in the National Programming Contest and National Level Debate held at Shivaji University. He won 2 prizes at National Level Programming Context.