© 2018 IJSRSET | Volume 4 | Issue 10 | Print ISSN: 2395-1990 | Online ISSN : 2394-4099
Themed Section : Engineering and Technology
DOI : https://doi.org/10.32628/IJSRSET196279

Task Scheduling Using an Adaptive PSO Algorithm in Cloud Computing Environment

B. Sivaramakrishna[1], Dr. T. V. Rao[2]
[1]Research Scholar, Department of Computer Science and Engineering, Acharya Nagarjuna University, Guntur, Andhra Pradesh, India
[2]Professor, Department of Computer Science and Engineering, PVPSIT, Krishna, Andhra Pradesh, India

ABSTRACT

Now-a-days energy planners are aiming to increase the use of renewable energy sources and nuclear to meet the electricity generation. But till now coal-based power plants are the major source of electricity generation. The problem of task scheduling is one of the most important steps in taking advantage of the cloud computing environment. Various experiments show that although it is almost impossible to have an optimal solution, it seems that there is a more optimal solution using heuristic algorithms. This work compares three heuristic approaches to scheduling cloud environment tasks. These approaches are the PSO algorithm, the ACO, and the adaptive PSO algorithm for efficient task scheduling. The goal of all three of these algorithms is to generate an optimal schedule to minimize task completion time.

Keywords - Cloud environment, ACO, PSO, Task Scheduling.

I. INTRODUCTION

Cloud computing has become one of the most advanced, sophisticated and suitable technical platforms for users of all possibilities. It is widely accepted as a utility service where numerous servers are connected to the Internet. Cloud users can access, process, store, and retrieve data for both home and business use without having a data center, software, hardware, or server, paying for it from any geographic area with Internet access. The cloud mainly provides three types of services. First, there is infrastructure as a service (IaaS), which provides cloud users with infrastructure for a variety of purposes, such as storage and computing resources. Second is the platform as a service (PaaS), which provides a platform for customers to run their applications on that platform. The third is software as a service (SaaS), which provides software to users; so users don't need to install software on their computer and can use the software directly from the cloud. Cloud providers take advantage of virtualization technology and provide their customers with computing resources in the form of virtual machines (VMs). On the other hand, service providers benefit from these VMs when they provide application-level services to users. Service providers use task scheduling techniques to assign user tasks to VMs, reduce response time, provide promising quality of service (QoS), and maximize resource utilization. Therefore, the task scheduling algorithm is one of the key elements of any cloud infrastructure. Task scheduling is one of the most important and critical issues in cloud computing, and many studies have sought to find the optimal

solution for planning the resources available in a cloud environment.

1.1 The planning problem

The planning problem is how to allocate tasks with limited resources to achieve some pre-set goals. The goal of a cloud computing schedule is to achieve the optimal schedule provided by the user. In order To improve the overall throughput of cloud computing systems with specific objectives is the optimal production range, quality of service (QoS), load balance, economic principles and so on.

1.1.1. Optimal path length

Makespan is a very important and common goal in planning tasks. Typically, users expect their tasks to be completed as quickly as possible. Optimal manufacturing scope is a common goal of both the cloud service provider and the customers.

1.1.2. Quality of service (QoS)

The planning system must ensure user-defined QoS. On the one hand, it must improve resource efficiency based on application features to ensure customer efficiency and accuracy. On the other hand, it should dynamically select and redirect resources according to changes in user status to respond to user economy and satisfaction. Thus, the goal is not only to protect users, but also to contribute to the long-term sustainable development of cloud computing.

1.1.3. Load balancing

Because the cloud computing platform has a lot of computers. In addition, the complex composition and different heterogeneous cloud computing platform make it difficult to balance the load in the current situation.

1.1.4. Economic principles

Due to the extremely large and cost-effective business model, the economy is a key factor in cloud computing design. Market-based cloud users and service providers can mutually benefit from an efficient scheduling system.

1.1.5. System capacity

Mainly for cloud computing systems, the capacity system is a measure of performance scheduling performance optimization and is an objective that needs to be considered when developing business models. Increasing the capacity of users and cloud providers would benefit both.

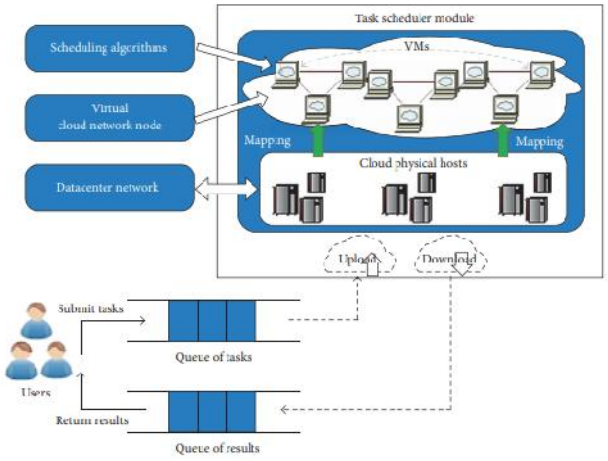


Figure 1. Cloud timing model

1.2. Scheduling the problem

The first assumption is that each task submitted by the user is independent of the other tasks. Suppose n the tasks together with the time of their completion in each processing machine are known in advance and should be processed m computational resources and aims to minimize execution time and optimize resource utilization.

Consider the set of tasks as follows: $T_i^i = \{1,2,3,\ldots,n\}$ what is n independent tasks and $R_i^i = \{1,2,3,\ldots,m\}$ is a set of computational resources. Suppose the task is completed in time i on the computational node j is known and is equal to jobRunTime (ij). The goal is to find a matrix in which as a task i filled with the resource j then

x_{ij} the element in this matrix is 1, otherwise it is 0 and it reduces the total cost of completing the task with computational resources. In this regard, two main conditions are considered

$$\sum_i x_{ij} = 1 \quad j \in T$$

$$x_{ij} \in \{0,1\} \quad i \in R, \quad j \in T$$

Makespan is equal to the time when the last task in the resource is completed. The first constraint ensures that each task is assigned to only one processing resource.

II. Task scheduling using a custom task schedule PSO algorithm

2.1. Basic description of the TSO

PSO is a metaheuristics of swarm intelligence inspired by the group behavior of animals, such as flocks or schools of fish. Similar to genetic algorithms (GA), this is a population-based method, meaning that it represents the state of a population algorithm that is repeated iteratively until the termination criterion is met. PSO algorithms for population $P = \{p_1, p_2, \ldots, p_n\}$ a variant of feasible solutions is often called a swarm. Possible solutions $p_1, p_2, \ldots, p_n$ called particles. The PSO method looks at the set R_d solutions as a 'space' where particles 'move'. To solve practical problems, the number of particles is usually chosen between 10 and 50. The goal of the particle spar optimization (PSO) algorithm is to solve the problem of unlimited minimization: find x^* such that $f(x^*) \leq f(x)$ for all real vectors of d-dimensions x . Purpose $f : R_d \to R$ called a fitness function.

2.1.1. Sweat topology

Every particle i is your neighborhood N_i (subset P). The structure of the neighborhoods is called a lap topology, which can be represented by a graph. Common topologies are: a fully integrated topology and a circle topology.

2.1.2. Suspension rule

The algorithm is terminated after a number of iterations or when the fit values of the particles (or the particles themselves) are in some way close enough.

2.1.3. Planning system

Figure 1 illustrates an overview of the scheduling system. The system consists of three modules, the first module is an application that represents a set of cloud sets (tasks). The second module is Mapping Algorithms (MA), which predicts the estimated time allocated to each cloud in each virtual machine and assumes that these values are available to the scheduler. The third module is the virtual machine (VM) that was used to launch the cloud blocks. The estimated time of the clouds is stored in a $m \times n$ matrix where m - number of virtual machines, and - n is the number of clouds. Apparently n/m is typically greater than 1 if there are more clouds than virtual machines, so some machines need to be assigned multiple clouds. Expected working time (ERT) is defined as the time spent on a task j resource r [21]. Each column of the expected time (ERT) matrix contains the estimated time (ERT) for each cloud j machine.

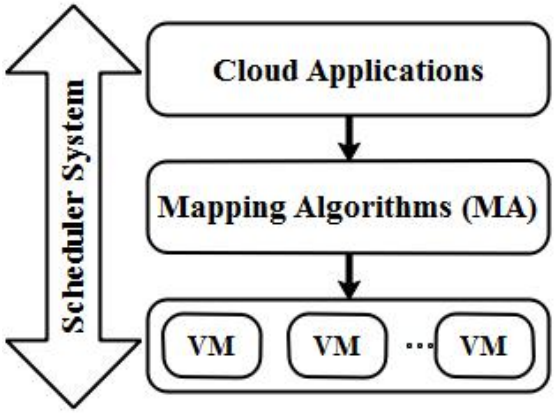


Figure 2. Timing system

2.2. Planning system

The main goal of allocating tasks to virtual machines is to reduce production time. The scope of a task is defined as the total time of completion of the task. Let's mark the time of the task T_i on top VM_j if CT_{ij}. Thus, the production range is determined by the following equation

$$CT_{max}[i,j] \mid i \in T, i = 1,2,\ldots,n \text{ and } j \in VM, j = 1,2,\ldots,m \quad (1)$$

Where $CT_{max}[i,j]$ is the maximum execution time of the task i on top VM_j and n, m is the number of tasks and virtual machines, respectively. Let $VM = VM_1, VM_2, \ldots, VM_m$ be m virtual machines to be processed n tasks represented by the group $T = T_1, T_2, \ldots, T_n$. Virtual machines are parallel and independent, and the schedule allocates independent tasks to these VMs. Also, processing a task in a virtual machine cannot interrupt (i.e.) non-preferences. Let's mark the end time of the task T_i next to CT_j. The proposed algorithms aim to reduce the range that can be denoted as CT_{max}. For planning, the run time of each operation on each virtual machine must be calculated as the processing speed of the virtual machine VM_j is PS_j, then the task processing time P_i can be calculated using the equation.

$$T_{ij} = C_i / PS_j \quad (2)$$

Where P_{ij} is the processing time of the task P_i using a virtual machine VM_j and ci is the computational complexity of the problem P_i. Processing time P_{ij} for each task P_i on top VM_j stored in the operating matrix. The processing time of each task in virtual machines can be calculated using an equation

$$P_{ij} = \sum_{i=1} P_{ij} \quad (3)$$

According to (1), (2) and (3), the task schedule algorithm should correspond to the following equation:

$$\sum_{i=1} P_{ij} < CT_{max} \quad (4)$$

When load balancing is considered, tasks are transferred from one VM to another for reduction CT_{max}, as well as response time. The task processing time varies depending on the speed of the virtual machines in different VMs. In the case of a transfer, the execution time of the task may differ optimally due to load balancing. The main purpose of scheduling adaptive tasks in the PSO algorithm is that tasks should be distributed in a virtual machine to minimize the amount of production and maximize the use of resources.

Adaptation Task Schedule PSO Algorithm

Initialization: Start with the position vector and velocity vector of each particle.

Conversion to discrete vector: Convert the continuous position vector to a discrete vector.

Fitness: Use the training function to calculate the fit value of each particle.

In the calculation P_{best} : Every particle P_{best} is determined by the best position value so far. If the current fit value of the particle is better than the particle value P_{best}, then replace P_{best} the value of the current position.

In the calculation G_{best} : From all the particles, select the particle with the best degree of suitability as G_{best}.

Updation: Update the position vector and velocity vector of each particle using the following equations:

$$V_i + 1 = \Psi V_i + c_1 rand_1 * (P_{best} - x_i) + c_2 rand_2 * (G_{best} - x_i)$$

$$X_i + 1 = X_i + V_i + 1$$

$$\Psi = inertia$$

Where ,

c_1, c_2 =acceleration coefficients

, rand1, rand2 = evenly distributed random numbers and $\varepsilon[0,1]$,

P_{best} =best position of each particle

,

G_{best} = best position of entire particles in a population

, i =iteration .

Repeat steps 2 to 6 until the stop conditions are met. The stop condition may be the maximum number of iterations or a change in the particle suitability value for successive iterations.

Output: As a final solution, print the best particle.

III. Results and analysis

This strategy is implemented in cloudsim 3.0.3 with the Eclipse Jee Oxygen IDE on the Windows 10 platform with a kernel i5 processor, 8 GB of RAM and a 2 GB Radeon graphics card. Cloudim uses JAVA, which provides cloud environment simulation. When implementing the algorithm, parameters such as the average cost of the round trip, the average execution time, the average preparation time are taken into account. The PSO used in the strategy is mutated and gives better results compared to other modified versions of the genetic algorithm and PSO. APSO is implemented and compared to the longest VM longest cloud algorithm, genetic algorithm and standard PSO, then it gives optimized results. This strategy is used for an increasing number of tasks, ie 100,200 to 1,000.

3.1. Improved VM comparison

In this section, the simulation is performed to evaluate the effectiveness of the proposed schedule approach of the optimized solution. The results of the proposed APSO algorithm are compared to other heuristic algorithms, such as PSO and ACO, through performance parameters such as speed and throughput. The simulation results show that our proposed algorithm surpasses other heuristic algorithms.

Table 1. Comparison of fixed VM

S.E i	Number of tasks	VM	APSO	PSO	ACO
1	100	50	17.04	25.32	28.55
2	200	50	52.01	109.00	117.15
3	300	50	104.96	176.19	198.90
4	400	50	175.95	271.50	279.56
5	500	50	264.96	561.50	454.29
6	600	50	371.98	579.70	667.71
7	700	50	497.03	945.23	955.52
8	800	50	647.99	834.54	1103.58
9	900	50	818.99	1032.28	1167.71
10	1000	50	1010.00	1554,21	1625,86

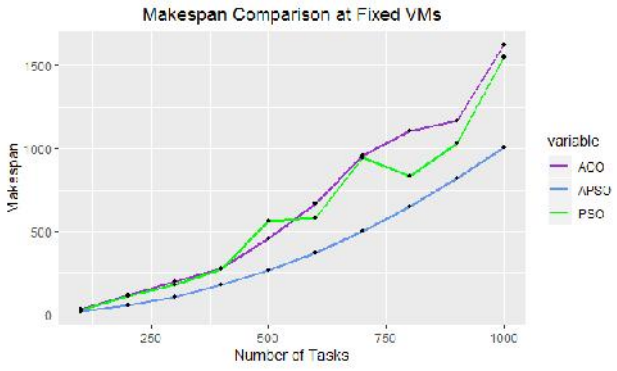


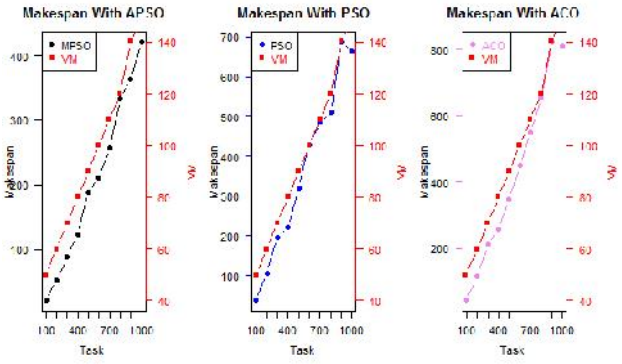
Figure 3. Fixed VM comparison

3.2. VM size comparison of variables

Comparison of the proposed APSO algorithm with the PSO standard and the ACO. This test has been applied more than 50 times using a timeshare policy, regardless of the nature of the task, and the result is presented. The prepared version has been compared by varying the number of tasks from 100 to 1000, keeping a fixed number of VMs 50. In addition, the results have been performed on different numbers of tasks between 100 and 1000 and different numbers of VMs between 40 and 140. The test results are shown in Tables 2 and 3 and shown in Figures 3 and 4. Makespan produced by the APSO algorithm is improved compared to the PSO standard and the ACO produced by ACO.

Table 2. Comparison of VMs of variables

S.Ei	Number of tasks	VM	APSO	PSO	ACO
1	100	40	20.71	37.32	43.72
2	200	50	51.99	105.50	116.51
3	300	60	88.15	197.32	213.11
4	400	70	122.60	222.62	258.22
5	500	80	189.30	319.05	347.72
6	600	90	210.30	428.01	449.02
7	700	100	257.38	484.34	549.54
8	800	110	333.30	510.88	655.93
9	900	120	362.80	687.09	822.36
10	1000	140	421.09	663.96	809.22



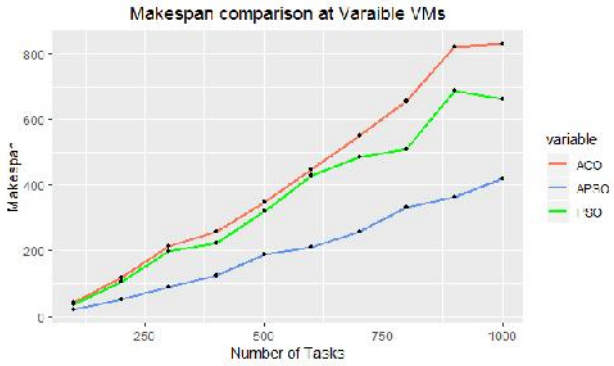


Figure 4. Comparison with modifiable VMs

3.3. Bandwidth comparison

The proposed comparison of APSO, ACO and PSO capacity is shown in Table 3 and shown in Figure 5. The performance parameter is calculated to analyze the maximum capacity. Table 3 and Figure 5 show that the performance of the APSO algorithm has improved even when the number of tasks is increased from 100 to 1000, while the fixed number of VMs is 50. Table 4 and Figure 6 also show the proposed task scheduling algorithm. capacity. improved by varying the number of tasks from 100 to 1000 and VM from 40 to 140. The throughput of the proposed algorithm is much improved compared to the PSO standard and the ACO algorithm.

Table 3. Comparison of VMs of variables

S.Ei	Number of tasks	VM	APSO	PSO	ACO
1	100	40	4.82	2.67	2.87
2	200	50	3.90	1.89	1.71
3	300	60	3.44	1.52	1.40
4	400	70	3.26	1.79	1.54
5	500	80	2.64	1.56	1.43
6	600	90	2.85	1.40	1.33
7	700	100	2.75	1.44	1.27
8	800	110	2.40	1.56	1.21
9	900	120	2.48	1.30	1.09
10	1000	140	2.37	1.50	1.23

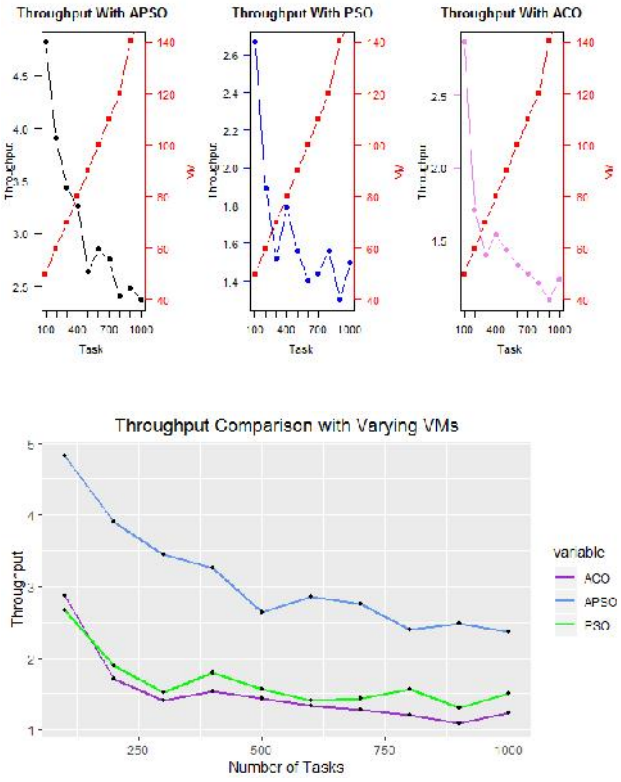


Figure 5. Comparison of variable VMs

Table.4. Comparison of success ratio with tasks

S.Ei	Task resource scheduling algorithm	Number of tasks	Success relationship
1	ACO	200	0.87
	PSO		0.9
	APSO		0.94
2	ACO	400	0.85
	PSO		0.88
	APSO		0.91
3	ACO	600	0.86
	PSO		0.86
	APSO		0.9
4	ACO	800	0.83
	PSO		0.85
	APSO		0.86
5	ACO	1000	0.81
	PSO		0.84
	APSO		0.85

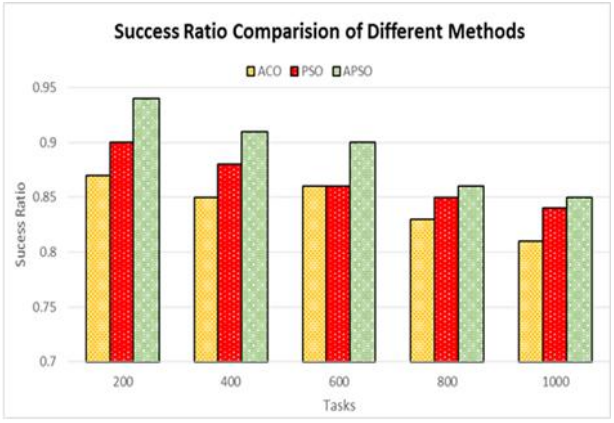


Figure 6. Comparison of the success ratio with the tasks

IV. Conclusion

This paper evaluates the problem of task scheduling in a cloud computing environment. The PSO algorithm and ACO are the most famous algorithms for scheduling distributed system tasks. To improve the performance of the standard PSO algorithm, an adaptive PSO algorithm is proposed in which an objective function is changed in a conventional PSO algorithm to generate an initial population to reduce the minimum range. Our experiments show that even if both the ACO and PSO algorithms show acceptable results, it can be said that, in general, PSO algorithms offer better results than ACO, but the modified PSO algorithm exceeds these two algorithms, reducing the input perspective. This algorithm can be used to efficiently schedule tasks for resources available in a cloud computing environment

V. REFERENCES

[1] Chinese cloud computing. Peng Liu: the definition and characteristics of cloud computing, http: //www.chinacloud.cn/.2009-2-25.

[2] R. Buyya, CS Yeo, S. Venugopal, J. Broberg, I. Brandic, “Cloud Computing and Emerging IT Platforms”, Vision, Leap and Reality in Computing as 5th Utility, Next Generation

Computer Systems 25 (6), 599–616 (2009). http://dx.doi.org/10.1016/j.future.2008.12.001

[3] P. Kumar, A. Verma, "Independent Timing of Tasks in Cloud Computing Using an Advanced Genetic Algorithm", International Journal of International Computer Science and Software Engineering Research, Vol2, 5th Edition, May 2012.

[4] Z. Yingfeng, L. Yulin, "Network Computing Resource Management Schedule Based on the Evolution Algorithm j]", Computer Engineering Conference, 2003, 29 (15): 1102175.

[5] P. Roy, M. Mejbah, N. Das. "Timing of Heuristic-Based Tasks with a Genetic Algorithm for Multiprocessor Systems by Selecting a Suitable Processor," International Journal of Distributed and Parallel Systems (IJDPS), Vol3, No. 4, July 2012.

[6] Abraham, R. Buyya and B. Nath. "Natural Heuristics in Computer Network Workplace Design", IEEE Eighth International Advanced Computing and Communication Conference (ADCOM 2000), India, 2000.

[7] H. Yin, H. Wu, J. Zhou, "Advanced Genetic Algorithm Constrained Rat Ion Grid Design," IEEE Sixth International Network and Collaborative Processing Conference, GCC 2007, Los Alamitos, CA, p. 221-227, 2007.

[8] R. Verma, S. Dhingra, "Genetic Algorithm for Scheduling Multiprocessor Tasks", IJCSMS International Journal of Computer Science and Management Studies, Volume 1, Edition 02, pp. 181-185, 2011

[9] J. Kennedy, RC Eberhart, "Optimization of Particle Swarms," Proc, IEEE Conf. Neural network, vol. IV, IEEE, Piscataway, NJ, 1995, pp. 1942-1948.

[10] L. Zhang, Y. Chen, B. Yang, "PSO Algorithm-Based Task Schedule in a Computer Network," Proceedings of the 6th International Conference on Intelligent Systems Design and Applications, Part 2, pp. 16-18. October 2006, Jinan, China.

[11] T. Chen, B. Zhang, X. Hao, Y. Dai, "Scheduling Tasks in a Network Based on Particle Swarm Optimization," Fifth International Symposium on Parallel and Distributed Computing, ISPDC '06. pp. 238-245, 2006.

Cite this article as :

B. Sivaramakrishna, Dr. T. V. Rao, "Task Scheduling Using an Adaptive PSO Algorithm in Cloud Computing Environment", International Journal of Scientific Research in Science, Engineering and Technology (IJSRSET), Online ISSN : 2394-4099, Print ISSN : 2395-1990, Volume 4 Issue 10, pp. 419-426, September-October 2018. Available at doi : https://doi.org/10.32628/IJSRSET196279
Journal URL : http://ijsrset.com/IJSRSET196279