International Journal of Computer Applications (0975 – 8887)
Volume 13– No.6, January 2011

Many-Objective Comparison of Twelve Grid Scheduling Heuristics

Rajendra Sahu
ABV-IIITM Gwalior, India

Anand K Chaturvedi
ABV-IIITM Gwalior, India

ABSTRACT

Different class of stakeholders of Computational Grid has their own perspective and preferences, which result in different, often contradictory, criteria for scheduling (main step of grid resource management). To increase the level of satisfaction of different class of stakeholders grid management system must use the scheduling heuristic, which provides compromise solution (i.e. a compromise schedule) using the many conflicting objectives. Present work analysed, conflicting as well as harmonious, interactions of Many-Objectives and performed many objective comparison to find the most suited heuristics out of the twelve popular heuristics by1- Visualization objectives of using 3D Bar Chart and Radar Chart in manner suggested, 2- Non-dominated Ranking of Heuristics and 3-Qualitative Comparison. Emphasis is given to computation time taken by heuristics.

Keywords

Computational Grid, Scheduling Heuristic, Many-objective comparison

1. INTRODUCTION

Connecting geographically distributed computational resources such as PCs, workstations, clusters, servers, and super computers, *Computational Grids* have emerged as a next generation computing platform for large-scale problems solving in academics, research and industry. Grid resource management involves dealing with three classes of stakeholders - end users making use of grid resources, owners of resources, and grid administrators. Each class of stakeholders has their own perspective and preferences, which result in different, often contradictory, criteria for scheduling (main step of Grid Resource Management). To increase the level of satisfaction of these stakeholders grid management system must use the scheduling heuristic, which provides compromise solution (i.e. a compromise schedule) using the many conflicting objectives.

Scheduling of task on heterogeneous grid resources is known to be a *NP-complete problem;* therefore, to get a near optimal solution within finite duration, heuristics/meta-heuristics are used instead of exact optimization methods. In some real-world situations, the meta-heuristic methods are too difficult or inappropriate, for example in fully automated systems (where we cannot tune parameters manually) or where the execution time should be very short, or for extremely large problems, etc. Therefore using pure heuristics in such situations is an appropriate solution (Izakian et al, 2009 b).

Much of the research into multi-objective algorithms concentrates on optimisation with two conflicting objectives. However, the real-world challenges to which these algorithms are applied often feature many more objectives (Coello et al, 2002). As the number of objectives increases solutions provided by methods become non-dominated and selecting one of the method from the available scheduling methods become difficult. Hence, there is a clear need to extend multi-objective optimization research into the realm of many-objectives (Farina and Amato, 2002).

Multi-objective algorithms are widely established and well developed for problems with two or three objectives. However, it is known that for many-objective optimization, where there are typically more than three objectives, applying Pareto optimality as a ranking metric may loose their effectiveness (Purhouse, 2003).

Present work analysed, conflicting as well as harmonious, interactions of Many-Objectives and performed many objective comparison to find the most suited heuristics out of the several available heuristics by following proposed methods

- Visualization of objectives using 3D Bar Chart and Radar Chart in manner suggested.
- Non-dominated Ranking of Heuristics
- Qualitative Comparison

Emphasis is given to computation time taken by heuristics.

Section 2 discussed proposed methods to compare Many-Objectives of a problem, in section 3 Many-Objectives of different stakeholders of computational grid are identified, twelve heuristics for grid scheduling are explained in section 4, set of popular twelve challenging instance of grid scheduling problem are mentioned in section 5, computational results and conclusions are presented in section 6 and 7 respectively.

2. MANY OBJECTIVE COMPARISON

In grid scheduling problem there are many objectives and these objectives are not independent. There exist conflicting as well as harmonious interaction between objectives. When objectives are in harmony improvement of objective leads to simultaneous improvement of other but when objectives are in conflict improvement of one leads to simultaneous deterioration of another.

2.1 Visualisation of Objectives

For more than three objectives it is not possible to compare the result in objective space using conventional methods like x,y scatter plot, 3d plot. Here it is suggested to use 3D Bar Chart and Radar Chart, in the manner described below, to graphically represent the many objective result.

To bring the different objective function values on common scale normalized value of objective function on 0 to 1 scale is calculated using maximum and minimum value of the objective function for all methods

$$f_{normalized} = \frac{f - f_{min}}{\mathrm{f}_{\max} - \mathrm{f}_{\min}} \tag{1}$$

Here f_{max} and f_{min} are the maximum and minimum value of objective function calculated using different heuristics

2.2 Non-dominated Ranking of Heuristics

Many-objective optimization, where there are typically more than three objectives, applying Pareto optimality as a ranking metric may loose their effectiveness (Purhouse, 2003). To overcome this Many-objective performance of different scheduling heuristics is compared by finding frequency of their attaining different ranks in non dominated comparisons.

Non-dominated Ranking comparison segregates the heuristic methods in different ranks i.e. rank 1,2,3…so on. Rank 1 is the highest rank. The methods, which get rank 1 are non-dominated by other rank 1 methods but dominates all the methods which get lower than 1 rank. Similarly rank 2 methods dominate methods of rank 3 and lower ones.

Non-dominated ranking can be performed for different combinations of objective functions i.e. by considering all objectives as well as by considering less number of objectives. If there are N objective then there can be ${}^{N}C_{N}$, ${}^{N}C_{N-1}$, ${}^{N}C_{N-2}$ … ${}^{N}C_{4}$, ${}^{N}C_{3}$, and ${}^{N}C_{2}$ combination.

Overall performance of these heuristics is evaluated by finding frequency of their attaining different ranks in non dominated ranking for all possible combinations of objective functions and for all instances of scheduling problem. Method can be rated based on their frequency count of different ranks.

2.3 Qualitative Comparison

Value of different objective function varies in different range hence Many-objective comparison is difficult.. For comparison of quality of solution provided by a particular heuristic normalized value of each objective function is calculated using equation (1)

Quality of objective function value can be expressed subjectively using word Best, Good, Fair and Poor based on range of normalized objective function value as shown below.

Quality of Objective	Range of $f_{normalized}$
Best	$f_{normalized} \le 0.05$
Good	$0.05 < f_{normalized} \le 0.15$
Fair	$0.15 < f_{normalized} \le 0.25$
Poor	$0.25 < f_{normalized}$

Heuristic giving more number of best objectives can be considered a better method.

2.4 Correlation between Objectives

In grid scheduling problem there are many objectives and these objectives are not independent. There exist conflicting as well as harmonious interactions between objectives. To evaluate this correlation can be find out between these objectives. When correlation coefficient is positive then there is harmony between objectives and when this is negative then there is conflict between the objectives.

3. OBJECTIVES OF GRID SCHEDULING

The grid scheduling problem is multi-objective in nature. Several performance measures and optimization criteria can be considered to evaluate the quality of a given schedule and overall grid system performance.

3.1 Makespan

Most popular optimization criterion is minimization of Makespan i.e. the finishing time of the latest job. Makespan measures the throughput of grid system. It can be defined as:

$$C_{max} = \max \{Cj , j=1,\ldots,N\} \quad (2)$$

3.2 Flow time

Flow-time is the sum of the finishing times of jobs. Flow time measures the Quality of Service of the grid system. It can be defined as:

$$F = \sum C_j , j=1,\ldots,N \quad (3)$$

Flow time is minimum when jobs are processed in ascending order of processing time on a particular grid resource. It is followed while calculating Flow-time.

3.3 Resource utilization

Due to economic aspect resource providers and grid managers are interested in the maximum utilization of resource. Resource utilization defined as the degree of utilisation of resources with respect to the schedule. The resource utilisation is defined using the completion time of a machine, which indicates the time at which machine m will finalise the processing of the previous assigned jobs as well as those already planned for the machine. Formally, it is defined as follows:

$$Resource\ Utilisation = \frac{\sum_{i \in Machines} completion[i]}{makespan.nb\_machines} \quad (4)$$

Here completion[i] is the completion time of last job on machine i, nb_machines is number of machines. Objective is to maximize the resource utilisation for all possible schedules.

3.4 Matching Proximity

In grid computing effort is made to process the task on the best possible machine i.e. the machine which takes minimum execution time. Matching Proximity indicates the degree of proximity of a given schedule to the schedule produced by the Minimum Execution Time (MET) method, which assigns a job to the machine having the smallest execution time for that job. Matching proximity is an additional performance parameter of batch mode methods. A large value for matching proximity means that a large number of jobs is assigned to the machine that executes them faster. It can be defined as:

$$Matching\ Proximity = \frac{\sum_{i \in Tasks} ETC[i][S[i]]}{\sum_{i \in Tasks} ETC[i][MET[i]]} \quad (5)$$

3.5 Computation Time

Due to dynamic nature of grid, computation time needed to generate schedule is also an important criterion for selecting a suitable scheduling method. In grid scheduling problem there is no need to get the optimal solutions. In the highly dynamic environment it is essential to get high quality feasible solution in short time. Computation time of the order of 1 micro second is measured for different heuristics.

4. HEURISTIC METHODS FOR GRID SCHEDULING

There are many heuristics for grid scheduling. Twelve popular heuristics, considered in this comparative study, are described in this section. For details please refer (Braun et al, 2001), (Maheswaran et al, 1999), (Xhafa et al, 2007a), (Xhafa et al, 2007b), and (Izakian et al, 2009 a).

4.1 Opportunistic Load Balancing (OLB)

In this method earliest idle machine is selected without considering the job's execution time on the selected machine. If two or more machines are idle then machine is selected arbitrarily. In this method time required for Scheduling is less and it keeps almost all the machines busy at all possible time. Resulting schedule is not optimal.

4.2 Minimum Execution Time (MET)

In this method minimum execution time is used to assign the job without considering the machine availability. Job is assigned to the machine on which it can be executed in minimum time. Allocating job without considering machine availability results in load imbalance on grid machines.

4.3 Minimum Completion Time (MCT)

In this method job is assigned to the machine that gives minimum completion time (ready time of machine + job execution time on the selected machine) for the job. Allocating job in this manner may result in execution of job on less faster grid machines.

4.4 Switching Algorithm (SA)

This method of scheduling combines the best features of MCT and MET methods of scheduling. The method tries to use better load balancing of MCT and execution on fastest machine of MET. Here the idea is to first use the MCT till a threshold of balance is reached followed by MET which creates the load unbalance by assigning jobs on faster machines. Here MCT and MET are used in cyclic manner.

4.5 k-Percent Best (kPB)

This method also attempts to combine the best features of MCT and MET simultaneously instead of cyclic manner. In this method only k percentage of best resources, on the basis of execution time, are considered while assigning the jobs. For a particular job a resource which gives minimum completion time is selected out of the k percent best resources instead of all possible resources. For k=100 this method act similar to MCT while for k=100/total number of machines this method act similar to MET. Here kPB has serious drawback that in a situation when k percentage resources are busy but other resource are free, even in such situation kPB allocates job only to one of busy k percent best resource. As a result there is large idle time of resources in the generated schedule.

4.6 Min-min

In this method completion time of all unassigned tasks ($1 \le j \le n$) on all the available machines ($1 \le i \le m$) is used to calculate the minimum completion time (MCTi) of task Ti on machine Mi*. Then task which gives minimum of MCTi is identified T*={min(MCTi) for ($1 \le i \le m$) } and assigned on the machine M*. Subsequently the task T* is removed from the list of unassigned task and workload of machine M* is updated. Above procedure is repeated till unassigned task list get exhausted.

4.7 Max-min

In this method, similar to min-min method, completion time of all unassigned tasks ($1 \le j \le n$) on all the available machines ($1 \le i \le m$) is used to calculate the minimum completion time (MCTi) of task Tj on machine Mi*. Then task which gives maximum of MCTi is identified T*={max(MCTi) for ($1 \le i \le m$) } and assigned on the machine M*. Subsequently the task T* is removed from the list of unassigned task and workload of machine M* is updated. Above procedure is repeated till unassigned task list get exhausted.

4.8 LJFR-SJFR

Largest Job on Fastest Resource – Shortest Job on Fastest Resource (LJFR-SJFR) method allocates largest job on fastest resource to reduce the makespan and allocates smallest job to fastest resource to reduce the flow time of the schedule.

In first stage the algorithm allocates m number of jobs similar to Max-min i.e. completion time of all unassigned tasks ($1 \le j \le n$) on all the available machines ($1 \le i \le m$) is used to calculate the minimum completion time (MCTi) of task Ti on machine Mi*. Then task which gives maximum of MCTi is identified T*={max(MCTi) for ($1 \le i \le m$) } and assigned on the machine M*. Subsequently the task T* is removed from the list of unassigned task and workload of machine M* is updated. In this manner m number of jobs are assigned to m number of unallocated machines.

In second stage remaining unassigned jobs are assigned alternatively using min-min and max-min method i.e. smallest job on fastest resource followed by largest job on fastest resource till the list of unassigned jobs get exhausted.

4.9 Suffrage

Suffrage for a job is the difference between second minimum completion time and first minimum completion time for that job. Suffrage method tries to allocate most suffered jobs in terms of expected completion time first. In this method suffrage is calculated for all unassigned jobs and the job which has maximum suffrage value is assigned to the machine which gives first minimum completion time. Then job is removed from unassigned job list, machine workload updated and above cycle of job allocation repeated till list of unassigned jobs get exhausted.

4.10 Work Queue (WQ)

It is a very simple method of job allocation. Jobs are randomly selected from the list of unassigned jobs and assigned to the machine with minimum workload. Job assignment repeated in similar manner till list of unassigned jobs get exhausted.

4.11 Relative Cost (RC)

While assigning jobs relative cost method considers both the load balancing of machines and the execution time of jobs on machines. Method calculates two parameters static relative cost $\gamma_{ij}^{\text{static}}$ and dynamic relative cost $\gamma_{ij}^{\text{dynamic}}$. Static relative cost is computed only once at the start of the method, on the other, the dynamic relative cost is computed at the start of each iteration k. For job i and machine j

$$\gamma_{ij}^{\text{static}} = \frac{m \times ETC_{ij}}{\sum_{j=1}^{m} ETC_{ij}} \quad (6)$$

$$\gamma_{ij}^{\text{dynamic}} = \frac{m \times completion_{ij}^{(k)}}{\sum_{j=1}^{m} completion_{ij}^{(k)}} \quad (7)$$

At each iteration k, the best job ibest is the one that minimises the expression:

$$\left(\gamma_{i,m_i^*}^{static}\right)^{\alpha} . \gamma_{i,m_i^*}^{dynamic}, \forall_i \in Jobs \qquad (8)$$

where:

mi*=argmin[completions i,m (k) |m ∈ Machines]

The value of α is fixed to 0.5 for the computational results.

4.12 Min-max

Min-max heuristic has two steps for assigning jobs to machines. In first step, similar to min-min method, completion time of all unassigned tasks ($1 \leq j \leq n$) on all the available machines ($1 \leq i \leq m$) is used to calculate the minimum completion time (MCTi) of task Ti on machine Mi*. In the second step for all tasks ratio of minimum execution time (time to execute on fastest machine) to its execution time on selected machine M_i* is computed and the task which has maximum value pf it is selected for assignment. Then job is removed from unassigned job list, machine workload updated and above cycle of job allocation repeated till list of unassigned jobs get exhausted.

5. TEST PROBLEM

For fair comparison of different scheduling methods we used ETC model of benchmark simulation experiments by (Braun et al, 2001) in our study. This model is based on Expected Time to Complete (ETC) matrix for 512 tasks and 16 machines. There are

Table 1 Heterogeneity and consistency combinations in the ETC model.

Heterogeneity		Consistency		
Task	Machine	Consistent	In-consistent	Semi-consistent
high	high	u_c_hihi	u_i_hihi	u_s_hihi
High	low	u_c_hilo	u_i_hilo	u_s_hilo
low	high	u_c_lohi	u_i_lohi	u_s_lohi
low	low	u_c_lolo	u_i_lolo	u_s_lolo

Twelve different benchmark instances of ETC matrices (512x16) each based on task heterogeneity, machine heterogeneity, and consistency. Their twelve combinations are as shown in Table 1. Machine heterogeneity represents the variation of execution times for a given task across the resources. An environment having similar resources will be represented by low machine heterogeneity, while high machine heterogeneity represents computing resources of different type and power. Task heterogeneity represents the degree of variation among the execution times of tasks for a given machine. In High task heterogeneity different types of applications are submitted to execute in the system, from simple programs to large and complex tasks which require large CPU times to be performed. An ETC matrix is considered consistent when, if a machine mi executes job t faster than machine mj, then mi executes all the jobs faster than mj. Inconsistency means that a machine is faster for some jobs and slower for others. An ETC matrix is considered semi-consistent if it contains a consistent sub-matrix. Instances are labeled as u_x_yyzz. u means uniform distribution (used in generating the matrix).

- x means the type of consistency

  (c – consistent, i – inconsistent and s– semi-consistent).

- yy indicates the heterogeneity of the jobs

  (hi means high, and lo means low).

- zz indicates the heterogeneity of the resources

  (hi means high, and lo means low).

These benchmark instances are considered one of the most demanding for the scheduling problems in heterogeneous computing environment by the large number of researchers. The main references in the literature used these instances in their scheduling methods.

6. COMPUTATIONAL RESULTS

In this section we present the results obtained for scheduling of twelve instances of test problem using the twelve heuristics of grid scheduling.

6.1 Computation Program

A computer program in C++ language is developed for all methods mentioned above which produces respective schedule and value of the various objectives. Program is executed on Intel (R) Core 2 Duo CPU T5550 @ 1.83 GHz, 1.83 GHz with 2 GB RAM and Window Vista operating system. Result obtained are discussed as follows.

6.2 Single Objective Comparison

Five objective functions are considered for this study. While considering one objective function at a time result obtained are as follows:

6.2.1 Makespan

Out of twelve instances Min-max method gives minimum Makespan in ten instances. For two instances of low task heterogeneity with consistency Min-Min method gives minimum value of Makespan.

6.2.2 Flow time

For all twelve instances, Min-Min method gives minimum value of flow time. Flow time is calculated by arranging task in ascending order of processing time on the assigned machine.

6.2.3 Resource utilization

Max-Min method gives best resource utilization for all twelve instances.

6.2.4 Matching proximity

For all instances best value of matching proximity is given by MET method.

6.2.5 Computation time

Out of twelve instances WQ method takes minimum computation time in eight instances. For rest of three, one and one instances minimum computation time is respectively taken by MET, MCT and kPB. Minimum computation time taken by WQ method is 3921 micro seconds.

6.3 Many Objective Comparison

As evident from single objective comparison of heuristic methods there is no single method which perform best on all objectives. Hence result must be analyzed considering more than one objective functions together.

6.3.1 Visualisation of objectives

It is difficult to draw conclusion while simultaneously considering result of all different heuristic methods for all twelve instances hence geometric mean of objective function values for all twelve instances is calculated for result of all heuristic methods.

This normalized value of geometric mean of all objective function values for all heuristic method is shown in Figure 1.

6.3.2 Non-dominated ranking of methods

Non-dominated ranking can be performed for different combinations of objective functions i.e. by considering all objectives as well as by considering less number of objectives. We considered total five objective here hence there can be ${}^5C_5$, ${}^5C_4$, ${}^5C_3$, and ${}^5C_2$ combination i.e. total 26 combinations.

Overall performance of these heuristics is evaluated by finding frequency of their attaining different ranks in non dominated ranking for total 26 combinations of objective functions and for all 12 instances of scheduling problem. In this manner there are total 312 (=26X12) non dominated comparisons and method getting highest frequency count is the best method.

Frequency count of ranks of different scheduling heuristics is shown in Figure 2. Min-min heuristic method get maximum (312 out of 312) rank 1 frequency count hence it is the best method. At second and third place are MCT and WQ methods with 272 and 255 rank 1 frequency count respectively.

6.3.3 Qualitative Comparison

Qualitative comparison of heuristics using all twelve instances of test problem is shown in Table 3.

- Min-Min, Min-Max and RC give Best quality Makespan for all 12 instances while suffrage gives in 10 instances.
- Min-Min, and Min-Max give Best quality Flow-time for all 12 instances while RC gives in 10 instances.
- Max-Min, and LJFR-SJFR give Best quality matching proximity in 12 and 11instances respectively, while suffrage and Min-max in 8 and RC gives in 7 instances.
- MET gives Best quality matching proximity for all 12 instances.
- Best quality computation time for all 12 instances is given by OLB, MET, SA, kPB and WQ.

6.3.4 Correlation between Objectives

Values of correlation coefficient between these objectives are give in Table 2. Highest positive correlation (0.946) is between Makespan and Flow-time and most negative correlation coefficient (-0.3396) is between resource utilization and computation time.

Table 2 Correlation between objectives

	Flow-time	Resource utilization	Matching Proximity	Time
Makespan	0.9460	0.2348	0.3494	-0.1507
Flow-time	1	0.0951	0.4963	-0.1553
Resource utilization		1	-0.2682	-0.3396
Matching Proximity			1	-0.2379
Time				1

7. CONCLUSION

There is no single heuristic that is excellent in satisfying fully all the conflicting expectations of different class of stakeholders of computational grid. Computation time taken by heuristics has most negative correlation with other objectives of gird scheduling which suggests to get better schedule more computation time is needed by the heuristics. Heuristics can be segregated into two groups. One group is suitable for immediate mode consisting OLB, MCT, MET, SA, kPB, and WQ and another is suitable for batch mode scheduling consisting Max-Min, LJFR-SJFR, Suffrage, RC, Min-Min, and Min-Max.

Contrary to belief, there is harmony in objectives of minimum makespan and minimum flow time. Heuristics giving good makespan also provide comparably good flow-time.

Min-min emerges as leader providing best quality of service as well as makespan but inferior resource utilisation. Min-max gives best makespan, resource utilisation but lacks in flow time. Qualitative comparison of heuristics suggests Min-max as the best heuristics.

8. REFERENCES

[1] Braun T.D., Siegel H.J., Beck N., Boloni L.L., Maheswaran M., Reuther A.I., Robertson J.P., Theys M.D., Yao B., A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems, *Journal of Parallel and Distributed Computing* 61 (6) (2001) 810_837.

[2] Coello, C.A.C. *et al*, 2002. *Evolutionary algorithms for solving multi-objective problems*. New York: Kluwer Academic Publishers.

[3] Izakian Hesam, Abraham Ajith and Snasel Vaclav, Comparison of Heuristics for Scheduling Independent Tasks on Heterogeneous Distributed Environments, *The 2009 IEEE International Workshop on HPC and Grid Applications (IWHGA2009), China, IEEE Press, USA*, ISBN 978-0-7695-3605-7, pp. 8-12, 2009a.

[4] Izakian Hesam, Abraham Ajith and Snasel Vaclav, Performance Comparison of Six Efficient Pure Heuristics for Scheduling Meta-Tasks on Heterogeneous Distributed Environments, *Neural Network World*, Volume 19, Issue 6, pp. 695-710, 2009.b

[5] Maheswaran M., Ali S., Siegel H.J., Hensgen D., Freund R.F., Dynamic mapping of a class of independent tasks onto heterogeneous computing systems, *Journal of Parallel and Distributed Computing* 59 (2) (1999) 107_131.

[6] Purshouse Robin C.. Evolutionary many-objective optimisation: An exploratory analysis. In *The 2003 Congress on Evolutionary Computation (CEC 2003)*, volume 3, pages 2066–2073, Canberra, Australia, 8–12 December 2003. IEEE.

[7] Xhafa F., Barolli L. and Durresi A. Batch Mode Schedulers for Grid Systems. *International Journal of Web and Grid Services*, Vol. 3, No. 1, 19-37, 2007a.

[8] Xhafa F. Carretero J. Barolli L. and Durresi A Immediate Mode Scheduling in Grid Systems. *International Journal of Web and Grid Services*, Vol.3 No.2, 219-236, 2007b.

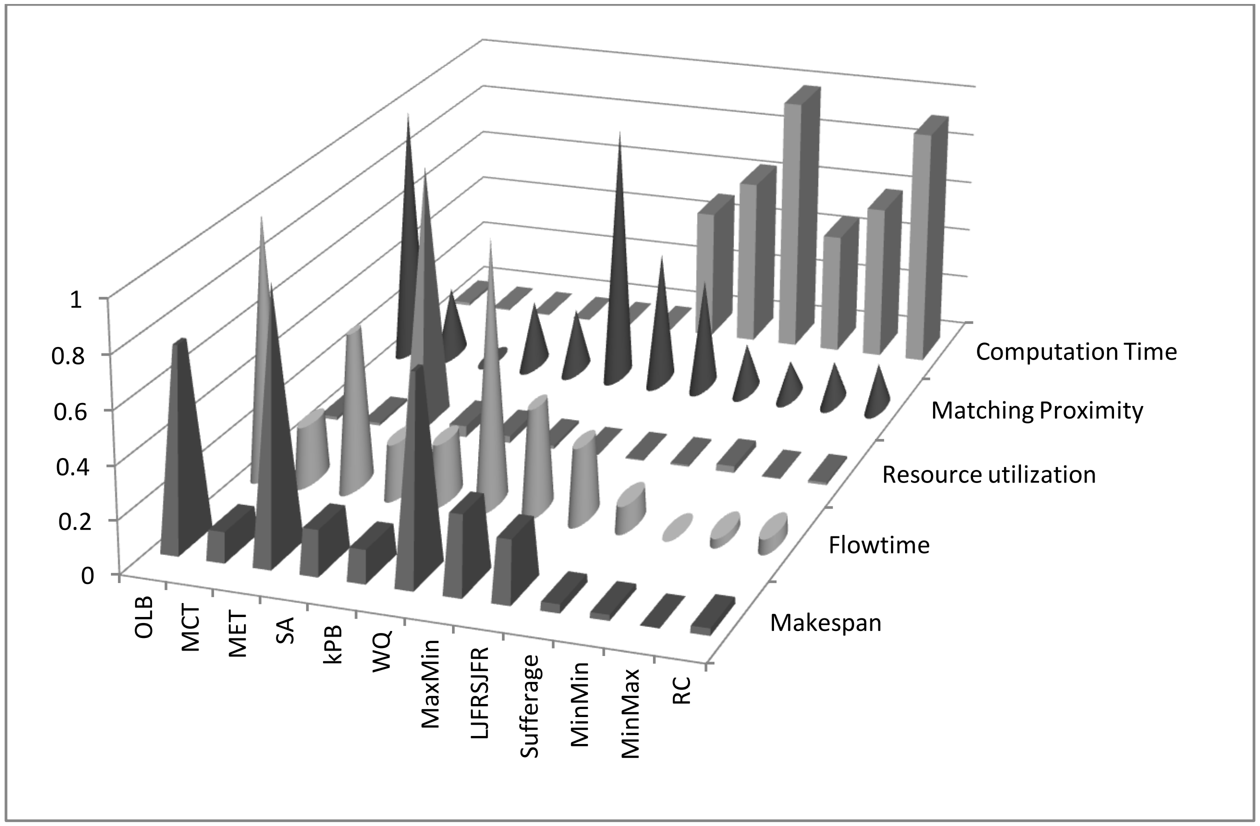
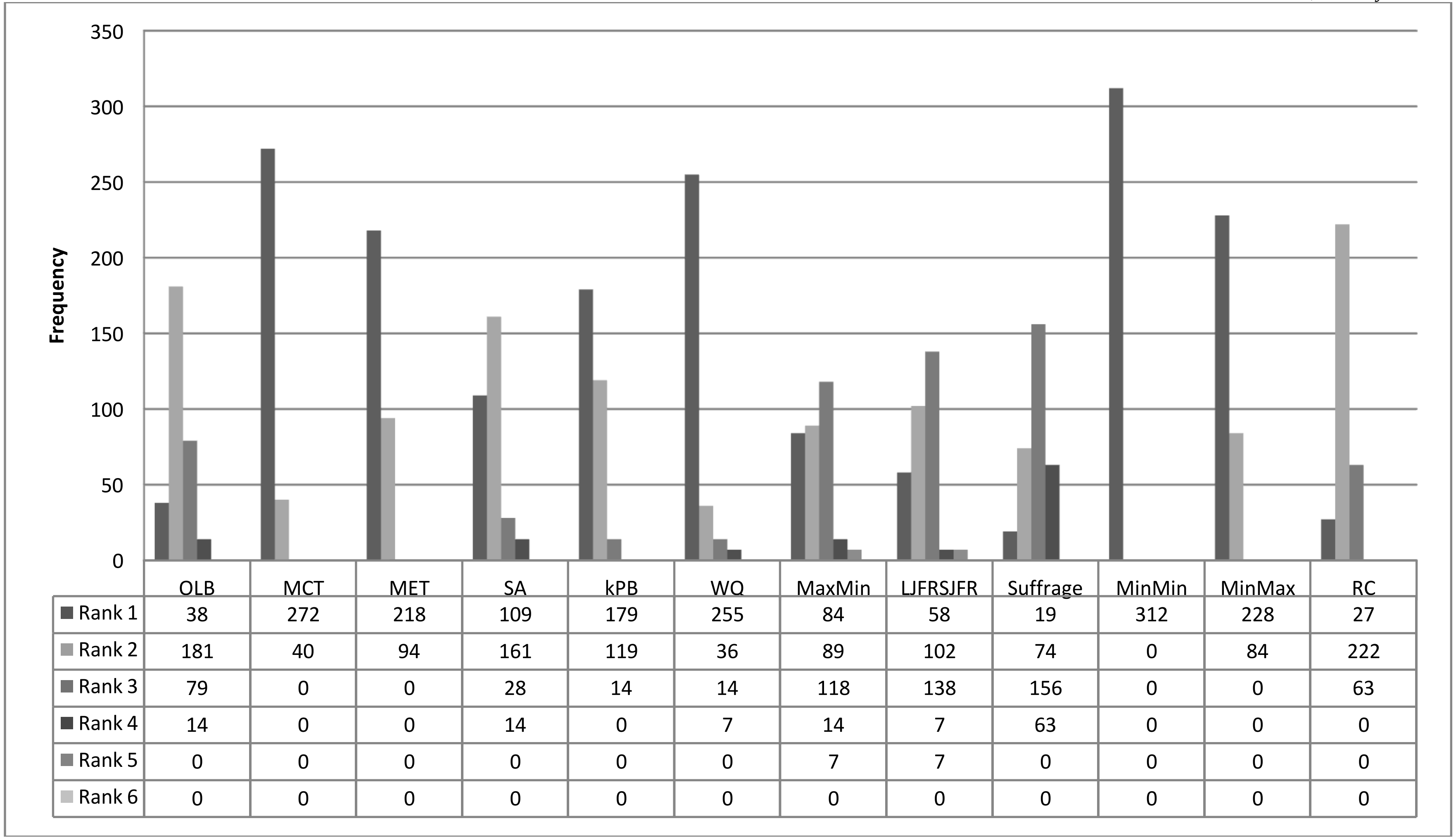


Figure 1 : Normalized Many-Objectives of schedules generated by different heuristics for geometric mean of all instances of problem represented using 3D Bar Chart



	OLB	MCT	MET	SA	kPB	WQ	MaxMin	LJFRSJFR	Suffrage	MinMin	MinMax	RC
■ Rank 1	38	272	218	109	179	255	84	58	19	312	228	27
■ Rank 2	181	40	94	161	119	36	89	102	74	0	84	222
■ Rank 3	79	0	0	28	14	14	118	138	156	0	0	63
■ Rank 4	14	0	0	14	0	7	14	7	63	0	0	0
■ Rank 5	0	0	0	0	0	0	7	7	0	0	0	0
■ Rank 6	0	0	0	0	0	0	0	0	0	0	0	0

Figure 2 : Frequency count of ranks of scheduling heuristics

Table 3 : Qualitative Comparison

		OLB	**MCT**	**MET**	**SA**	**kPB**	**WQ**	**Max-Min**	**LJFR-SJFR**	**Suffrage**	**Min-Min**	**Min-Max**	**RC**
Best	Makespan	0	3	0	2	5	1	2	2	10	**12**	**12**	**12**
	Flow time	0	0	4	1	0	0	0	0	6	**12**	**12**	10
	Resource Utilisation	2	3	0	0	0	3	**12**	11	8	0	8	7
	Matching Proximity	0	0	**12**	1	1	0	0	0	4	4	4	4
	Computation Time	**12**	11	**12**	**12**	**12**	**12**	0	0	0	0	0	0
	Total	14	17	28	16	18	16	14	13	28	28	***36***	33
Good	Makespan	2	9	4	10	7	1	3	5	2	0	0	0
	Flow time	2	6	0	7	6	2	2	4	4	0	0	2
	Resource Utilisation	9	7	0	4	6	8	0	1	2	7	4	2
	Matching Proximity	0	4	0	3	3	0	0	0	2	4	2	2
	Computation Time	0	1	0	0	0	0	0	0	0	0	0	0
	Total	13	27	4	24	22	11	5	10	10	11	6	6
Fair	Makespan	2	0	0	0	0	2	4	3	0	0	0	0
	Flow time	0	5	0	3	4	0	4	7	2	0	0	0
	Resource Utilisation	1	2	0	3	6	0	0	0	2	1	0	1
	Matching Proximity	0	2	0	2	2	0	2	2	2	0	2	2
	Computation Time	0	0	0	0	0	0	0	0	0	0	0	0
	Total	3	9	0	8	12	2	10	12	6	1	2	3
Poor	Makespan	8	0	8	0	0	8	3	2	0	0	0	0
	Flow time	10	1	8	1	2	10	6	1	0	0	0	0
	Resource Utilisation	0	0	**12**	5	0	1	0	0	0	4	0	2
	Matching Proximity	**12**	6	0	6	6	**12**	10	10	4	4	4	4
	Computation Time	0	0	0	0	0	0	**12**	**12**	**12**	**12**	**12**	**12**
	Total	30	7	28	12	8	**31**	**31**	25	16	20	16	18

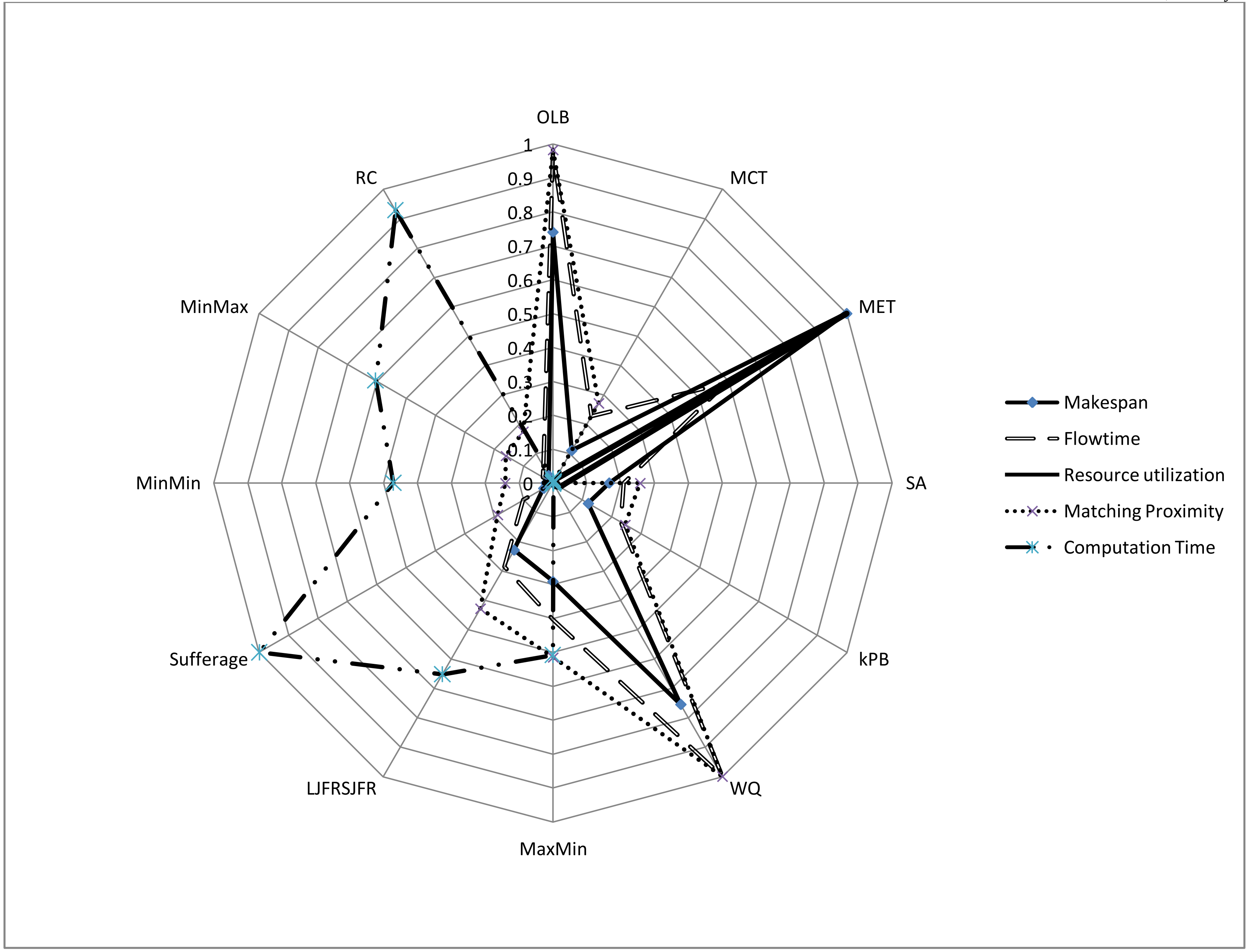


Figure 3 : Normalized Many-Objectives of schedules generated by different heuristics for geometric mean of all instances of problem represented using Radar Chart