8 CONCLUSIONS AND ROAD MAP FOR THE FUTURE WORK

8.1 Conclusion

This thesis has analyzed many static and dynamic scheduling algorithms with the soft real-time system in a single processor environment. It has also proposed a hybrid scheduling algorithm S_LST and also compared its performance with LST and SJF algorithm. Studies of different swarm intelligence techniques have been done and focused on Particle Swarm Optimization as a candidate solution for the problem statement. Using PSO, a scheduling algorithm for a soft real-time system has been designed and developed. The proposed algorithm has been compared with EDF and ACO based scheduling algorithm.

All algorithms have been assessed with three major categories of CPU load (U_p) values which are considered as underload, overload, and highly overload scenarios. If the load of CPU is less than 1 ($U_p \leq 1$), it is considered as underload scenario, if it is $1.0 < U_p \leq 1.5$, it considered an overload scenario, and if it is $1.5 < U_p \leq 5.0$ is considered a highly overload scenario. The task set contains a different number of tasks for different CPU load. All algorithms have been tested on 500-time units to prove their effectiveness [70]. Performance of all algorithms has been measured and evaluated concerning SR and ECU parameters. Some important results of all the proposed algorithms are compared and shown in Table 8.1 to 8.3. Comparison has been done between existing EDF, LST, RM, SJF, ACO based scheduling algorithm and proposed S_LST and PSO based scheduling algorithm.

Table 8.1 – All scheduling algorithm performance in Underload Scenario

CPU Load	ECU%							SR%						
	Dynamic		Static		Hybrid	Swarm Intelligence		Dynamic		Static		Hybrid	Swarm Intelligence	
	EDF	LST	RM	SJF	S_LST	ACO	PSO	EDF	LST	RM	SJF	S_LST	ACO	PSO
0.50	49.49	49.49	49.49	49.49	49.49	49.98	49.49	100.00	100.00	100.00	100.00	100.00	100.00	100.00
0.55	54.66	54.40	54.40	54.31	54.41	55.04	54.40	100.00	100.00	100.00	100.00	100.00	100.00	100.00
0.60	59.39	59.39	59.39	59.39	59.39	59.88	59.39	100.00	100.00	100.00	100.00	100.00	100.00	100.00
0.65	64.35	64.35	64.35	64.35	64.35	65.00	64.35	100.00	100.00	100.00	100.00	100.00	100.00	100.00
0.70	69.35	69.35	69.35	69.35	69.35	69.93	69.35	100.00	100.00	100.00	100.00	100.00	100.00	100.00
0.75	74.31	74.31	74.31	74.31	74.31	74.88	74.31	100.00	100.00	100.00	100.00	100.00	100.00	100.00
0.80	79.22	79.22	79.22	79.22	79.22	79.83	79.22	100.00	100.00	100.00	100.00	100.00	100.00	100.00
0.85	84.16	84.16	84.16	84.15	84.16	84.72	84.16	100.00	100.00	100.00	99.99	100.00	100.00	100.00
0.90	89.16	89.16	89.15	89.00	89.16	89.62	89.15	100.00	100.00	99.99	99.84	100.00	100.00	99.99
0.95	94.17	94.17	94.08	93.89	94.17	94.54	94.08	100.00	99.99	99.93	99.78	99.99	100.00	99.94
1.00	99.10	99.10	97.78	96.74	99.10	99.37	97.99	100.00	100.00	98.92	98.74	100.00	100.00	99.26

Dynamic, Static, Hybrid and Swarm Intelligence based scheduling algorithms have been compared in underload scenario with respect to SR and ECU parameters. Table 8.1 represents all observations during in underload scenario. Dynamic scheduling algorithms (e.g., EDF and LST) give the best performance in underload and especially when the load of CPU is near to 1. In the case of Static, Hybrid and PSO based scheduling algorithms, they missed a few of their deadlines in very specific case. ACO based scheduling algorithm also performs equally in this situation compare to a dynamic scheduling algorithm.

These all-scheduling algorithms are also evaluated in overload scenario where load of CPU is very between 1.0 to 1.5, and all observations are represented by Table 8.2. Performance of dynamic scheduling algorithm degrades rapidly in slightly overload situation. Whereas static, hybrid and swarm intelligence-based algorithms are still able to meet many deadlines of their task set.

Table 8.2 – All scheduling algorithm performance in Overload Scenario

CPU Load	ECU%							SR%						
	Dynamic		Static		Hybrid	Swarm Intelligence		Dynamic		Static		Hybrid	Swarm Intelligence	
	EDF	LST	RM	SJF	S_LST	ACO	PSO	EDF	LST	RM	SJF	S_LST	ACO	PSO
1.05	17.45	16.09	70.85	56.63	56.29	63.69	65.91	18.27	15.84	78.49	73.49	71.22	67.01	78.24
1.10	9.21	8.33	75.82	63.60	56.45	54.22	70.60	9.31	7.90	80.49	75.98	67.82	55.01	80.53
1.15	6.29	5.58	73.20	62.66	55.32	51.86	67.90	6.19	5.06	75.88	73.66	65.79	50.87	75.91
1.20	4.62	4.21	83.50	70.08	48.09	46.61	80.39	4.22	3.67	79.47	73.06	52.84	45.33	80.03
1.25	4.06	3.56	79.05	73.20	49.65	45.15	77.35	3.67	3.06	77.58	77.47	56.21	36.23	78.97
1.30	3.63	3.09	75.66	72.24	51.96	38.78	74.73	3.19	2.53	73.81	75.34	57.82	35.90	76.02
1.35	3.12	2.63	74.65	70.99	53.55	39.03	74.06	2.65	2.09	70.77	71.55	55.93	37.14	72.65
1.40	2.66	2.20	83.55	76.57	45.37	38.05	81.39	2.24	1.71	75.47	73.80	46.76	33.91	76.49
1.45	2.50	2.01	79.75	74.45	48.39	34.11	78.15	2.00	1.52	69.03	68.76	47.74	30.65	70.66
1.50	2.21	1.83	85.27	80.07	43.77	33.08	86.15	1.71	1.33	70.33	69.96	42.10	27.91	73.35

Table 8.3 – All scheduling algorithm performance in Highly Overload Scenario

CPU Load	ECU%							SR%						
	Dynamic		Static		Hybrid	Swarm Intelligence		Dynamic		Static		Hybrid	Swarm Intelligence	
	EDF	LST	RM	SJF	S_LST	ACO	PSO	EDF	LST	RM	SJF	S_LST	ACO	PSO
1.60	2.17	1.77	85.61	77.26	50.71	45.98	85.02	1.61	1.29	69.52	67.20	47.22	37.25	71.25
1.70	2.03	1.58	86.26	79.16	47.45	40.45	86.52	1.42	1.07	65.99	64.60	42.72	30.24	68.70
1.80	1.93	1.45	86.12	77.28	47.46	35.52	86.34	1.30	0.95	65.98	63.04	42.14	26.39	68.27
1.90	1.90	1.31	85.83	77.53	48.39	33.56	85.17	1.29	0.85	63.51	62.21	41.83	25.35	65.52
2.00	1.84	1.19	85.78	78.10	42.82	29.56	86.90	1.20	0.76	62.88	61.00	35.30	21.45	65.23
2.25	1.76	1.13	84.27	76.95	45.85	32.51	85.89	1.04	0.65	56.16	55.91	35.86	21.24	59.81
2.50	1.55	0.98	87.06	74.97	44.51	25.54	87.86	0.89	0.54	53.82	49.92	31.05	15.39	56.23
2.75	1.46	0.91	89.21	74.42	38.06	18.31	88.82	0.78	0.47	52.07	46.83	25.28	10.16	53.75
3.00	1.32	0.86	94.46	77.23	31.51	14.66	94.25	0.63	0.40	48.36	41.67	18.37	07.11	49.21
3.50	1.27	0.75	93.48	73.37	37.33	15.80	94.46	0.57	0.33	44.50	36.76	19.05	07.69	45.52
4.00	1.11	0.73	95.04	79.57	28.03	09.67	96.20	0.43	0.27	39.52	34.09	12.59	03.79	40.47
4.50	1.08	0.71	96.77	71.58	27.99	09.86	97.69	0.38	0.24	36.45	27.74	11.47	03.37	36.99
5.00	0.97	0.66	98.13	78.22	20.16	08.74	97.88	0.31	0.20	31.72	25.71	07.09	02.41	32.15

When the load of the CPU is more than 1.5 it considers as highly overload scenario, and all these algorithms are also evaluated in this situation. It has been observed that the performance of dynamic, scheduling algorithms are very poor compare to a static scheduling algorithm. The performance of swarm intelligence based scheduling algorithms are still batter compare to static, dynamic and hybrid. Even in swarm intelligence algorithms, PSO based scheduling algorithm performs more batter compare to ACO-based scheduling algorithm.

8.2 Road Map for the Future Work

This research has been carried out with Soft Real-Time System. Systems in which deadlines may occasionally be missed with the only degradation in performance of the entire system but not a complete failure are called Soft Real-Time systems. Sometimes, in a soft real-time system, a task that missed its deadline should nevertheless be completed, i.e., its service through late is still valid and helpful. In a special case of soft real-time systems, called a firm real-time system, there is no value for a task that has missed its deadline, but there is no catastrophe either [72][25]. The PSO based algorithm which is proposed in this thesis applies to a Soft Real-Time System, and the value of the task has been taken the same as its computation time required. This algorithm can be applied on Hard and Firm Real-Time System and can be checked its effectiveness with it.

Periodic tasks are common in many practical real-time systems, and in this research, PSO based scheduling algorithm has been applied to the periodic task set. However, some tasks in many systems are non-periodic such as tasks that handle emergency situations, operator commands, etc. If the nonperiodic tasks are of low importance, they can be treated as background tasks.

They will be scheduled whenever the processor is not being utilized by the periodic tasks [73] or will be scheduled using more sophisticated sporadic servers. However, in a realistic system, some of the sporadic (or aperiodic) tasks will definitely have high importance and hard deadlines; in this case, one tries to translate aperiodic tasks into equivalent periodic tasks based on the worst-case frequency and computation demands of the aperiodic tasks [54]. In future, this algorithm can be evaluated with respect to the sporadic and aperiodic task set.

The PSO based scheduling algorithm has been tested with a Uniprocessor environment. A multiprocessor system can be divided into tightly coupled or loosely coupled systems. For tightly coupled systems, global status and workload information on all processors can be kept current at a low cost. The system has shared memory and generally uses a centralized scheduling algorithm. If the system uses a separate scheduling algorithm for each processor, the decisions and actions of the scheduling algorithms of all the processors are coherent [5]. For a loosely coupled system (also called distributed system), there are one or more processors and a certain amount of memory in each node. The nodes don't share memory; message exchange across the network is the only mechanism for communication between them. For the uniprocessor system, a variety of optimal scheduling algorithms and heuristics for scheduling an overloaded system were presented. The problem becomes much more difficult in a multiprocessor system. As such, there is no optimal scheduling algorithm exist for multiprocessor environments [74]. The proposed PSO based scheduling algorithm or other Swarm Intelligence techniques can be explored for scheduling problems in a Multiprocessor environment.