Scheduling Algorithm in Distributed Real-time System

Zhao Hongyu - A0224411X
Zhou Ying - A0229575R

Group: 24
February 25, 2021

Department of Electrical Engineering

Scheduling Algorithm in Distributed Real-time System

Zhao Hongyu
Electrical and Computer Engineering
National University of Singapore
Singapore, Singapore
e0572530@u.nus.edu

Zhou Ying
Electrical and Computer Engineering
National University of Singapore
Singapore, Singapore
e0680008@u.nus.edu

***Abstract*—In a Real Time System, achieving deadlines of multiple tasks efficiently is the main target of every scheduling algorithm. Therefore, four scheduling algorithms are introduced in this report to solve the problem. The ACO scheduling achieves a balance of exploration and exploitation. The D-R-EDF and EDF-RM scheduling combine both the EDF and RM algorithm to overcome their separate pitfalls but in different ways. The EMUF combines the advantages of fixed and dynamic scheduling and provide dynamically changing priority with flexible scheduling method. All of the above are evaluated in terms of Success Ratio(SR), Average CPU Utilization(ECU). The results show ACO gets a better performance in ECU and EDF-RM achieves 4% to 8% better performance compared with EDF in slightly overloaded environment. And the ECU of E-R-EDF is at least 45%while EDF under 10%. The throughput of EMUF performs compared to MUF and MMUF.**

***Keywords*—Distributed Real Time System, scheduling algorithm, multiprocessor and homogeneous system**

I. Introduction

Real Time System always executes tasks based on some time restrictions(e.g deadline interarrival period or tardiness). In a Hard RTS, missing deadline will cause serious outcomes; In a Soft RTS, missing part of deadlines will not affect the whole system. As for a firm RTS, completing tasks on time will obtain more rewards. Recently, as the development of multiprocessor and Distributed system, load balancing among all the professors increase the whole performance with the two crucial methodologies-Task migration and duplication. Therefore, achieving a good task migration and duplication method result in the huge increasement of the performance and efficiency. We will discuss some recent scheduling algorithms for RTDS, ACO [1], EDF-RM [2], D-R-EDF [3], EMUF [4], in this report. Besides, due to the queuing theory, single queue scheduling generates a healthier average response time in Global Scheduler compared to Partitioned Scheduler.

In order to directly see the comparison and performance differences among the scheduling algorithms, we share the same evaluation criterion in this report.

A. Success Ratio(SR)

$$SR = \frac{Successfully\ scheduled\ tasks}{Total\ number\ of\ tasks\ arrival} \tag{1}$$

B. Failure Ratio(SR)

$$FR = \frac{Tasks\ miss\ the\ deadline}{Total\ number\ of\ tasks\ arrival} \tag{2}$$

C. Average CPU Utilization (ECU)

$$EUC = \frac{Successfully\ scheduled\ tasks'\ computation\ time}{Total\ time\ of\ scheduling} \tag{3}$$

The rest parameters are task related. In ACO, p_i(t) is the probability of ith node at time t; i $\in$ R1 and R1 is set of nodes (schedulable tasks) at time t; τ_i(t) is pheromone on ith node at time t; η_i(t) is heuristic value of ith node at time t. s is sequence number of the node visited by the ant during the best journey. C is constant (preferably 0.1). In EDF-RM, related parameters are shown below:

Symbols	Definition
T	Task-set
τ_{arrival}	Task arrival time
τ_{period}	Interarrival period of task
τ_{dline}	Task deadline
$\tau_{\text{utilization}}$	Per task utilization
τ_{priority}	Task priority
$\tau_{p,a}$	Task a of p processor
$\rho_{\text{utilization}}$	Per processor utilization
ρ_{UB}	Processor upper bound
$E\rho_{\text{utilization}}$	Efficient processor utilization
SR	Success Ratio
FR	Failure Ratio
MR	Migration ratio
A_preemption	Average number of preemptions
B_Q	Boundary limit of global task queue
TST	Total time of scheduling
tard_{τ_i}	Tardiness of given task

Fig. 1. Symbols and definitions in EDF-RM.

In D-R-EDF, related parameters are SR and ECU.

In EMUF, the performance is estimated by average turnaround time(av.TAT), average waiting time(av.WT), throughput.

II. ALGORITHMS

In this section, we will first discuss the main scheduling algorithm structure, advantages and disadvantages separately and advice improvement. Then we will create a common table to compare all of their features.

A. ACO

The Ant Colony Optimization is an excellent method to achieve the balance between exploration and exploitation by providing inherent parallelism. Therefore the multiprocessor system based on ACO is relatively has the features of robustness, effectiveness compared with single agent algorithms. The ACO in RTDS consisted of four steps: Tour Construction,Analyze the Journey; Pheromone Update; Selection of Task to execute on each processor.

The Tour Construction is to build up possible paths for ants, in this article the author use full permutation of the tasks to represent all the possibility in a slightly overload environment. Pheromone update is crucial to increase the speed of convergence of the algorithm as well as to determine when the system can produce a decision.

1) Tour Construction: Each nods represent each task, and it contains the probability to choose the task as well as saving shared pheromone. During each time tour construction, ants must start from different nodes based on their probability, so there will be different tour sequences.

Probability$(p+i)$ of a node at particular time can be format as following equations:

$$p_i(t) = \frac{(\tau_i(t))^\alpha * (\eta_i(t))^\beta}{\sum_{i \in R_1} (\tau_i(t))^\alpha * (\eta_i(t))^\beta} \tag{4}$$

η_i can be determined by

$$\eta_i = \frac{K}{D_i - t} \tag{5}$$

where t us current time, K is a constant, D_i is absolute deadline of ith node.

Suppose at time t, there are four task A,B,C,D and four ants to search the tour journey. After calculating each of the nodes probability based on above equations, we produce a full permutation for every ants:

$Ant1 : A \rightarrow B \rightarrow C \rightarrow D$
$Ant2 : B \rightarrow A \rightarrow C \rightarrow D$
$Ant3 : C \rightarrow A \rightarrow B \rightarrow D$
$Ant4 : D \rightarrow A \rightarrow B \rightarrow C$

2) Analyze the Journey: After each ant search a different tour, they can calculate their own journey's performance with available processors and determine the ratio of the succeed tasks and failed tasks based on each tasks' inherent features(e.g arrival time, computation time, deadline).

3) Pheromone Update: Pheromone update is the most crucial step because it determine how fast the convergence speed and how soon the RTDS to make decisions. There are two method jointly to achive a better pheromone update: Pheromone Evaporation and Pheromone Laying.

In the Pheromone Evaporation, every epoch all nodes' pheromone needs to be updated in order to forget bad journey and pave the way to new paths in the future. Because among all of the ants, there will be only one optimal route in one specific epoch and all others are "bad journey" compared with the "single good journey".

$$\tau_i = (1 - \rho)\tau_i \tag{6}$$

where ρ is a contant.

Pheromone Laying is to choose the best journey and compute the amount of pheromone laid at variable nodes in that particular journey. And then based on the nodes sequences to update each of the nodes.

$$\tau_i = \tau_i + \Delta\tau_i \tag{7}$$

where i $\in$ N1, N1 is the set of nodes at time t.

$$\Delta\tau_i = \frac{ph}{s} \tag{8}$$

$$ph = C * \frac{Number\ of\ Succeed\ Tasks}{Number\ of\ Missed\ Tasks} + 1 \tag{9}$$

4) Selection of Task to execute on each processor: After updating pheromone, p of each nodes is available and select tasks based on the sequence from the highest p to lowest p with available multiprocessors.

Case study

We simulate the new Algorithm by producing a new task set,the set consists with 3 tasks. task{A,B,C}, which is different from the article's example. As shown in the simulation figure, ACO achieves better performance than EDF. And in the article, the ACO achieves SR=100% & ECU=95.32%, more than EDF with SR=96.38% & ECU=74.43%.

TABLE I
TASK SET

Task	Arrival Time	Absolute Deadline	Required Exe.Time
A	0	4	2
B	0	5	3
C	0	5	4

TABLE II
STEPS OF THE PROPOSED ALGORITHM AT TIME T=0

Tour Ant	Path	Succ Tasks	Miss Tasks	Imp.After Pheromone Update	Select Task at T=0
1	A,B,C	2	1	C(max) A B(min)	P1:C P2:A
2	B,A,C	2	1	C(max) A B(min)	P1:C P2:A
3	C,A,B	3	0	C(max) A B(min)	P1:C P2:A

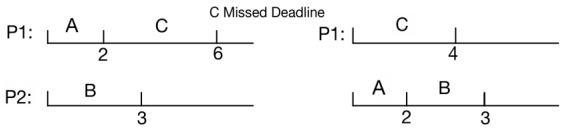

Fig. 2. Scheduling for Case study (left) EDF (right) ACO.

Analyse

*a) **Advantages**:*

- Achieve a balance between exploration and exploitation, the ACO makes the RTDS more robust. Compared with EDF, it is more efficient in underloaded conditions.
- Ant do not need synchronization, achieving a high level of inherent parallelism.

*b) **Disadvantages**:*

- the ACO is based on too much conditions, e.g computation time, absolute deadline, and in most of time they are unavailable.
- the ACO is executed on homogeneous system, which is not common in real time. In most of time, there are heterogeneous system with different multiprocessors' situation.
- If the parameters α and β are set improperly, the solution speed is very slow and the quality of the obtained solution is particularly poor.
- The basic ACO has a large amount of calculation and takes a long time to solve.As a result. it may be difficult to make timely decisions.

Advice improvement

ACO algorithm perform badly in slightly complicated condition, especially in overload environment, large calculations might prevent it from making timely decisions. Instead, modify the ACO by using Elitist strategy for ant system or MAX-MIN ant system could help increase convergence speed as well as decrease the numbers of iterations.

B. EDF-RM

EDF and RM are dynamic and static scheduling algorithm respectively to achieve deadline of tasks. There is a common problem Domino's effect in EDF that especially in overload conditions. As for RM, it performs badly in underloaded conditions. Therefore, this EDF-RM just joint two algorithms overcome both of their inherent pitfalls to achieve a jointly increase of performance.

RM is used as a global scheduler in a global Task Queue, it can detemine the density of the Queue by setting a threshold equation. EDF is used on each of processor to execute every waiting task in its local task buffer. If the utilization of processor exceed the threshold, task migration mechanism will be launched.

1) Earliest Deadline First Scheduling: EDF is a dynamic scheduler following the nearest the highest emergency of task.

$$\tau_{priority} \propto \frac{1}{\tau_{dline}} \tag{10}$$

where the task set is schedulable only if

$$\rho_{utilization} \leq 1 \tag{11}$$

2) Rate Monotonic Scheduling: RM scheduling algorithm is based on static scheduler and follows the shortest interarrival period the highest emergency of task.

$$\tau_{priority} \propto \frac{1}{\tau_{period}} \tag{12}$$

where the task set is schedulable only if

$$\rho_{UB} \leq n(2^{\frac{1}{n}} - 1) \tag{13}$$

3) Joint EDF-RM: The joint EDF-RM is mainly doing three things: maintenance of global task queue; execution of assignen tasks on allotted processors; migration of tasks in between processors if needed.

a) global task queue: There is a global waiting task queue to obtain and save randomly arrival task(FIFO), and the Global task Queue has a bound following the RM scheduling , that is n x $(2^{\frac{1}{n}} - 1)$. For each of task utilization following the below equation:

$$\tau_{i\ utilization} = \frac{\tau_{wcet}}{\tau_{period}} \tag{14}$$

During the underloaded condition when there are not too many tasks, the sum of task utilization is much less than the threshold of RM, the task set is schedulable. However, the RM limit shows a monotonous decreasing trend as n becomes bigger. There must be a point of time the sum of tasks utilization exceeds the RM limit, then the task set is unschedulable, requiring some specific policy.

$$B_Q = \begin{cases} Schedulable, & \sum_{i=1}^{n} \tau_{i\ utilization} \leq n * (2^{\frac{1}{n}} - 1) \\ Non - schedulable, & otherwise \end{cases} \tag{15}$$

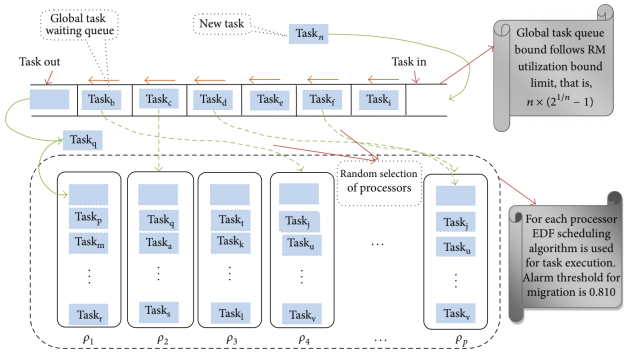

Fig. 3. Global Queue and assignment.

b) execution of assignen tasks on allotted processors: Each task is randomly assigned to single professor's task buffer, and on each of professor, it sorts the task queue and executes task by EDF algorithm as follows:

$$\rho_{i\ utilization} = \sum_{j=1}^{p} \tau_{j\ utilization} \tag{16}$$

where p is the total number of tasks in professor i

c) task migration: When the professor utilization exceed the threshold(in the article, the threshold is fixed value 0.810), it will start the task migration procedule, sort the all professors utilization and then choose the one with least utilization.

$$\rho_{i\ utilization} \begin{cases} Schedulable, & if\ \rho \leq 0.810 \\ taskmigration, & if\ 0.810 \leq \rho < 1 \\ overloadingoccur, & if\ \rho > 1 \end{cases} \quad (17)$$

Input: Random arrival of tasks with $\tau_{arrival}$, τ_{wcet}, τ_{period}, τ_{dline}
Output: Number of tasks meet/miss the deadline along with another parameters

BEGIN
GlobalScheduler() `// Global task Queue`
(1) `The aperiodic // Periodic arrival of tasks with arrival time, wcet, assigned deadline and period`
(2) `tasku = wcet/Period;`
(3) `UB=n*(Math.pow(2, 1.0/n)-1);`
(4) **IF** `tasku<=UB`
(5) `Generated task is schedulable`
(6) `pselection(task)`
(7) **Else**
(8) `The task is non-schedulable`

pselection(task) `// Random Selection of Processors`
(1) `Random Selection of Processor`
(2) `PQueue(task);`

PQueue(task) `// Processors local queue`
(1) `Assign priorities to tasks on the basis of deadline`
(2) $\text{task}_{\text{priority}} \propto \frac{1}{\text{task}_{\text{priority}}}$
(3) `TaskExecution(task)`

TaskExecution(task) `// Task Execution by using EDF Scheduler`
(1) **If** `tasku<=1`
(2) `U= U+ tasku //Cumulative accumulation of task utilization`
(3) **If** `(U<=.810) //Processor utilization`
(4) `The task is ready for execution`
(5) **Else**
(6) `Task Migration (task, tasku)`

Taskmigration(task, tasku) `// Task Migration on the basis of a processor utilization factor`
(1) `Sort all processor utilization`
(2) `Assign task to the processor having least a utilization factor`
(3) `PQueue(task);`
END

Fig. 4. EDF-RM scheduling algorithm.

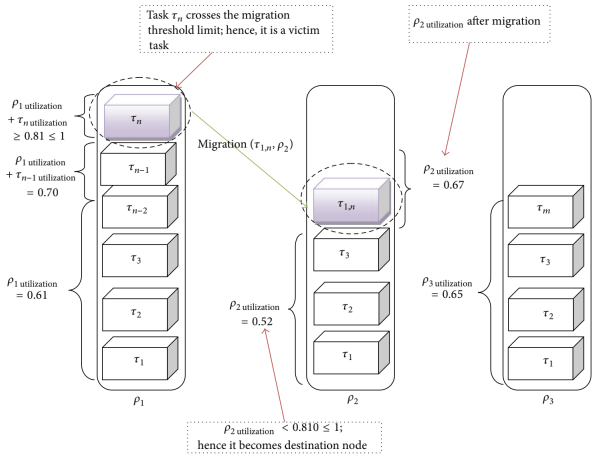

Fig. 5. Task migration.

Analyse

*d) **Advantages**:*

- Overcome both the pitfalls of EDF and RM, achieve a better performance in both underloaded and overload conditions.
- Obtain the advantages from both the algorithms, achieve a better scheduling performance in terms of two criterion: task deadline and task period.

*e) **Disadvantages**:*

- EDF-RM is based on too much conditions, e.g computation time, absolute deadline, and in most of time they are unavailable.
- EDF-RM is executed on homogeneous system, which is not common in real time. In most of time, there are heterogeneous system with different multiprocessors' situation.
- Due to the waiting task queue, all new arrival tasks need to be put inside the queue regardless of their priority to some extent, which may cause problems and latency for emergency tasks.

Advice improvement

Because task migration is also time-consuming, the EDF-RM could be modified slightly to lower the probability of task migration, e.g assign tasks to each processor based on their utilization, computation to communication ratio(CCR) in between processors e.t, instead of randomly assignment.

C. D-R-EDF

The D-R-EDF algorithm is a combination of both static and dynamic approach, which works same as EDF algorithm during underloaded condition and RM algorithm during overloaded condition. Earliest deadline first(EDF) scheduling algorithm has been proved to be the optimal dynamic scheduling, and is a necessary and sufficient condition, which CPU utilization is up to 100%.

However, in the case of instantaneous overloaded, the system behavior is unpredictable and domino phenomenon may occur that one task is lost, a series of tasks will be lost in succession.

RM scheduling algorithm gives highest priority to the tasks with shortest period,in this way all task will meet their deadline. It performs well in overloaded situation. However, it doesn't behave as efficiently as dynamic algorithm in underloaded condition.

As the paper assumed, the system model is client and server distributed system. Server is transferring tasks to client and each client makes scheduling decision independently. And the task is released with deadline and required computation time.

The D-R-EDF algorithm steps as follows:

- During the underload condition: it works as EDF algorithm. A set of N periodic task is scheduled if and only if CPU utilization factor U = $\sum \frac{C_i}{A_i}$
- During the overload condition: it works as RM algorithm. A set of N periodic task is scheduled when it meets the equation: U $\leq N * (2^{\frac{1}{N}}\text{-}1)$
- When 2 continuous jobs miss the deadline, the algorithm will switch for RM algorithm considering it is overloaded condition.
- After 5 continuous job achieved the deadline, the algorithm will shift back to EDF algorithm considering that overloaded condition is disappeared.

Analyse

*a) **Advantages**:*

- It overcomes the limitations of both EDF and RM algorithms and exploits their advantages.
- Especially, it overcomes the Domino's effect problem of EDF during overload condition.

*b) **Disadvantages**:*

- Only periodic tasks are considered.
- The efficiency of the proposed algorithm is compared with classic EDF algorithms.
- It can not handle tasks which do not satisfy the CPU equation.
  Consider there are four tasks{t1,t2,t3,t4}in a queue having arrive time, execution time and deadline as t1(0,1,4) t2(1,2,6) t3(2,2,8) t4(2,4,10).
  Task1, 2, 3 can be scheduled by D-R-EDF algorithm, but when it comes to task 4, it can not be scheduled by the proposed algorithm because

$$\sum U = \frac{C_1}{A_1} + \frac{C_2}{A_2} + \frac{C_3}{A_3} + \frac{C_4}{A_4} = 1.23 > 1 \qquad (18)$$

And as N=4,

$$\sum U \leq N * (2^{(1/N)} - 1) = 0.76 \qquad (19)$$

So, clearly in this case task4 can not be scheduled by D-R-EDF algorithm.

Advice improvement

Be inspired by the EDF-RM algorithm, the D-R-EDF algorithm can be migrated to another processor when it is not satisfy the equations.

D. EMUF

Enhanced maximum urgency first algorithm (EMUF) is urgency based scheduling algorithm. Similar to D-R-EDF algorithm, it resolves the problem during transient overload that is when CPU load factor exceeds 100%. EMUF algorithm employs both fixed and dynamic priority, that each task is given an urgency which is defined as a combination of two fixed priorities (criticality and user priority) and a dynamic priority that is inversely proportional to the laxity.
The priority of tasks are ordered by following rules:

- Critical set includes those tasks which are really critical for the system and they need to be executed for better system performance.
- Using intelligent laxity as dynamic priority, process with the minimum intelligent laxity scheduled first.
- The user priority depends on the user ordered from most important to less important.

Priority level: critical priority $>$ dynamic priority $>$ user priority
The algorithm as explained below:

Phase 1: define fixed priorities, which remain constant throughout the scheduling.

1) Order the task from most important to least important.
2) Add the critical tasks as defined before to the critical set where CPU utilization factor is less than 100%.

Phase 2: calculate the dynamic priority.

1) If there is only 1 critical task it will be executed without any pre-emption.
2) If there is more than 1 critical task, select the task with least intelligent laxity.
   a) If there is a tie in intelligent laxity then select the task with the highest user priority.
   b) After the completion of each task, again intelligent laxity is calculated for all the remaining processes and process with least intelligent laxity is selected for execution.
3) If there is no critical task in the ready queue select the task from non-critical set which has the least intelligent laxity.

Repeat the same part of phase2.(2)

Calculate average turnaround time, average waiting time and throughput of the processes.

Fig. 6. EMUF Algorithm.

Analyse

*a) **Advantages**:*

- Compared to MUF, it decreases the response time of remaining process when there is a task missing its deadline.
- It improve the throughput of system and prevents the processor from doing unwanted work.
- It reduce the turnaround and average waiting time of the task and improve the whole system efficiency.

*b) **Disadvantages**:*

- The task with high importance will be missed.
- It need to calculate intelligent laxity in every process, which may increase the cost and hard to realize.
- Performed only in a single processor environment, and the number of tasks are quite limited. Besides,it need to know attributes like numbers, burst time, priority of processes before.

GLOBAL ANALYSE OF FOUR PAPERS

Common Topic and features

All of the four algorithm target to achieve the deadline of tasks as much as possible in distributed multiprocessors environment.

TABLE III
FEATURES COMPARISON

Algorithm	Features
ACO	inherent parallelism & pheromone update
D-R-EDF	overcome Domino's effect & overload condition judgment
EDF-RM	task migration & global task queue
EMUF	intelligent laxity & critical task set

Comparison of Advantages and Disadvantages

TABLE IV
ADVANTAGES & DISADVANTAGES COMPARISON

Algorithm	Advantage	Disadvantage
ACO	exploration and exploitation & robust	lower convergence under bad parameters setting & large calculation
D-R-EDF	overcome the limitations of EDF and RM and exploits their advantages	no tasks migration
EDF-RM	work well under both underloaded and overload	latency for emergency tasks due to global task queue
EMUF	reduce turnaround time average waiting time	long switching time

Comparison of Performance

We get the result based on the largest difference between baseline and relevant algorithm in different environment with variable task load condition.

TABLE V
PERFORMANCE IMPROVEMENT

Algorithm	Baseline	SR	FR	ECU
ACO	EDF	80%	80%	75%
D-R-EDF	EDF	35%	35%	55%
EDF-RM	D-R-EDF	7%	7%	1.5%

CONCLUSION

Distributed Real Time System has developed fast recently. The requirement for high speed, deadline and fault-tolerant RTDS has became more important. Especially achievement of deadline and at the same time maintain high efficiency.Therefore, four algorithm are intriduced in this report.

ACO is an excellent method to search global optimal path with Pheromone Evaporation and Pheromone laying.

EDF-RM intelligently joint both EDF and RM to maintain a global task queue to store arrival task as well as allow task migration when processor get close to overload state.

D-R-EDF combines both the EDF and RM algorithm and works well in both underload and overload situations.

The EMUF improves the performance of systems with flexible strategy of task priority allocation during scheduling.

To get further, more merits have been included into EDF-RM. The algorithm works well both in underload and overload environment. The task migration technique can achieve better ECU with less traversing cost.

However, more complicated constraints for other conditions like heterogeneous multiprocessors, task dependence(DAG) or computation to communication ratio could be search deeper by the future works.

REFERENCES

[1] Shah, Apurva and Ketan Kotecha. "ACO Based Dynamic Scheduling Algorithm for Real-Time Multiprocessor Systems." IJGHPC 3.3 (2011): 20-30. Web. 27 Feb. 2021.

[2] Rashmi Sharma, Nitin, "Performance Evaluation of New Joint EDF-RM Scheduling Algorithm for Real Time Distributed System", Journal of Engineering, vol. 2014, Article ID 485361, 13 pages, 2014.

[3] A. Shah and K. Kotecha, "Efficient scheduling algorithms for real-time distributed systems," 2010 First International Conference On Parallel, Distributed and Grid Computing (PDGC 2010), Solan, India, 2010.

[4] Behera, H.S., Raffat, N., Mallik, M. (2012). Enhanced Maximum Urgency First Algorithm with Intelligent Laxity for Real Time Systems. International Journal of Computer Applications, 44, 20-26.

APPENDIX

Group Works

Name	Algorithms
Zhao Hongyu	ACO, EDF-RM
Zhou Ying	D-R-EDF, EMUF

Note: ALL pseudocodes and figures are made by latex