A Hyper-heuristic for scheduling independent jobs in Computational Grids

Author: Juan Antonio Gonzalez Sanchez

Coauthors: Maria Serna and Fatos Xhafa

Overview

- Introduction and motivation
- Hyper-heuristic design
- Hyper-heuristic tests
- Conclusions
- Future work

Introduction and motivation

- Computational Grid
  - Parallel and distributed system that enables the sharing, selection and aggregation of geographically distributed resources
- Efficient scheduling of tasks in resources in a global, heterogeneous and dynamic environment
- Tasks
  - From different users
  - Executed in unique resource
  - Different types (intensive numeric computation vs data process) / (immediate vs batch)
- Resources
  - Dynamically added/dropped from the Grid
  - Can process one task at a time
  - Specialiced resources (intensive numeric computation vs data process)

Introduction and motivation

- *Ad-hoc* heuristics:
  - Simples
  - Deterministic
  - Short execution time
- E.g.: Opportunistic Load balancing, Minimum Completion Time, Minimum Execution Time, etc...
- No one method performs best!
  - Need to select them in an accordance with grid instance to yield best performance

Problem definition (instances)

- Tasks to be scheduled
- Resources to be used in scheduling
- Workload of each task (in millions of instructions)
- Computing capacity of each resource in mips
- Ready time: ready[m] - when the resource m will finish executing its scheduled tasks.
- ETC[t][m] – Expected Time to Compute task *t* in resource *m (from Simulation Model of Braun et al. 2001)*

Problem definition (Objectives)

- Makespan: finishing time of latest task

  max{F_t : t *in Tasks*}

  or

  max{Cr: r in Resources}

- Flowtime: sum of finishing times of tasks

  sum {F_t: *t in Tasks*}

Note: Completion time:

C_r = ready[r] + sum {ETC[t][r] for tasks scheduled in r}

Ad-hoc heuristics

- Our Ad-hoc heuristics:
  - Immediate mode: tasks are scheduled as soon as they arrive in the system
    - Opportunistic Load Balancing, Minimum Completion Time, Minimum Execution Time, Switching Algorithm and K-Percent Best
  - Batch mode: The schedule is done for a set of tasks (a *batch*).
    - Min-Min, Max-Min, Sufferage and Relative-Cost

Evaluating instance characteristics

- Instance notation x_yyzz (Braun et al. 2001)
  - X : computing consistency (c-consistent, i-inconsistent and s-semiconsistent)
  - YY: Tasks heterogeneity (hi-high and lo-low)
  - ZZ: Resources heterogeneity (hi-high and lo-low)

- Preprocess input information
  - Workload variance   task heterogeneity
  - Mips variance   resource heterogeneity
  - ETC matrix analysis   matrix consistency

Note: ETC=Expected Time to Compute

Performance of Hyper-heuristic for Braun et al.'s instances - Makespan

	OLB	MCT	MET	SA	KPB	Min-Min	Max-Min	Suff	RC
C_HIHI		X				X			
C_HILO		X							X
C_LOHI		X				X			
C_LOLO		X							X
I_HIHI		X						X	
I_HILO					X			X	
I_LOHI			X					X	
I_LOLO					X			X	
S_HIHI		X							X
S_HILO					X				X
S_LOHI					X				X
S_LOLO		X							X

X-the method was chosen most of the times out of 100 independent runs

Performance of Hyper-heuristic for Braun et al.'s instances - Flowtime

	OLB	MCT	MET	SA	KPB	Min-Min	Max-Min	Suff	RC
C_HIHI				X		X			
C_HILO				X		X			
C_LOHI				X		X			
C_LOLO				X		X			
I_HIHI			X			X			
I_HILO			X			X			
I_LOHI			X			X			
I_LOLO			X			X			
S_HIHI				X		X			
S_HILO				X		X			
S_LOHI				X		X			
S_LOLO				X		X			

X-the method was chosen most of the times out of 100 independent runs

Proposal: Hyper-heuristic design

- Parameters of the hyper-heuristic:
  - Parameter to fix task heterogeneity threshold
  - Parameter to fix resource heterogeneity threshold
  - Parameter to work with immediate or batch methods
  - Parameter to fix the measure to optimize (makespan or flowtime)

Proposal: Hyper-heuristic design

- High-level algorithm:

Input: parameters, tasks, resources, ready-times, ETC matrix

1. Evaluate task heterogeneity
2. Evaluate resource heterogeneity
3. Examine ETC matrix to deduce its consistency
4. Choose (based on parameters) the ad-hoc method to execute
5. Execute ad-hoc method

Output: schedule

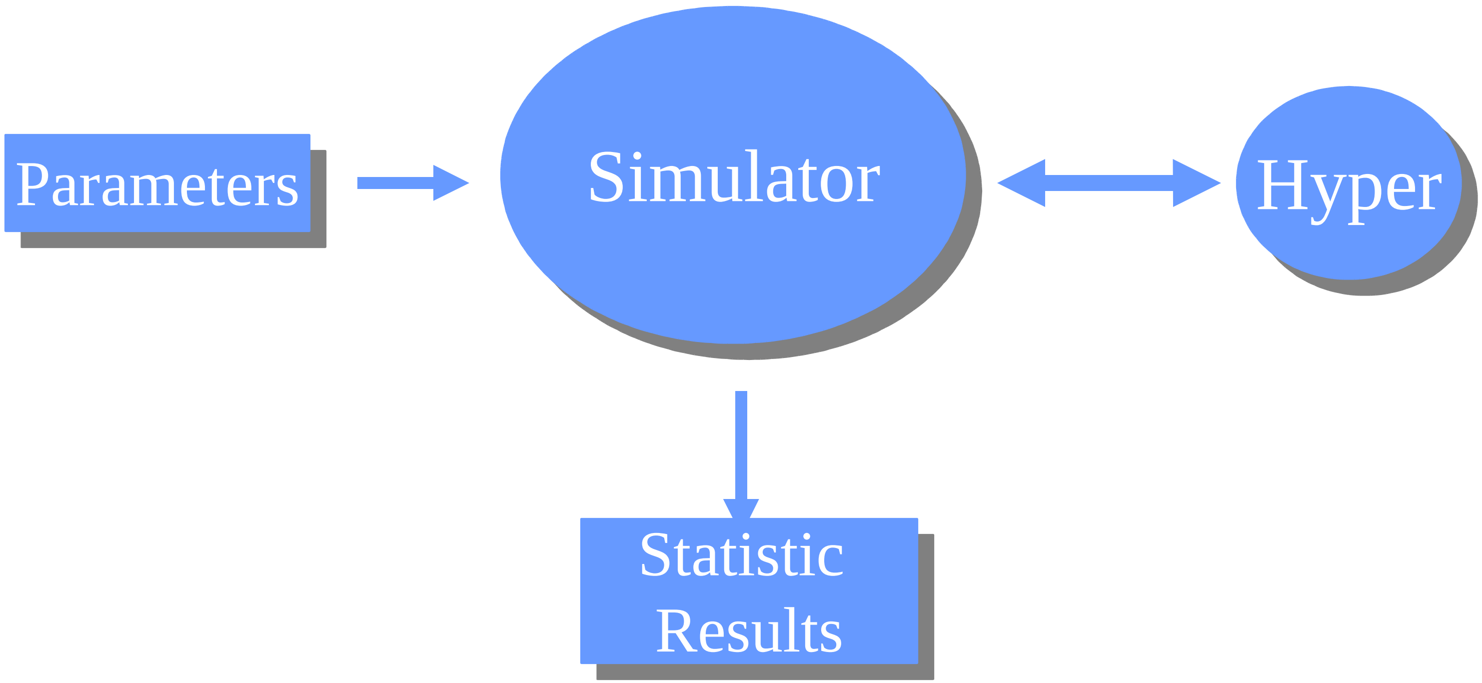
Performance Evaluation of Hyper-heuristic for a grid simulation environment

- We use a Grid Simulator implemented with a discrete event simulation library (*HyperSim*).
- Highly parametrizable:
  - Distributions of arriving and leaving of resources in the Grid and its mips
  - Distributions of task arrival to the Grid and its workloads
  - Initial resources/tasks in the system and maximum tasks to generate
  - Task and resource types
  - Percentage of immediate/batch tasks.
- For a schedule event the simulator calls the hyper-heuristic and passes it the ETC matrix, ready times, resources and tasks that will be scheduled as input

Simulator trace example

- time= 000000000.00 event= EVN_NEW_HOST info: Host# 00000,
  Mips = 000000200.00
- time= 000000000.00 event= EVN_NEW_HOST info: Host# 00001,
  Mips = 000000200.00
- time= 000000000.00 event= EVN_ENTER info: Task# 00000,
  Work = 000006000.00
- time= 000000000.00 event= EVN_ENTER info: Task# 00001,
  Work = 000006000.00
- time= 000000000.00 event= EVN_ENTER info: Task# 00002,
  Work = 000006000.00
- time= 000000000.00 event= EVN_ENTER info: Task# 00003,
  Work = 000006000.00
- time= 000000000.00 event= EVN_SCHEDULE info:
  Scheduled 00004 Tasks, 00002 Hosts
- time= 000000000.00 event= EVN_START info: Task# 00001 on Host# 00001
  finishTime = 000000030.00
  exeTime = 000000030.00
- time= 000000000.00 event= EVN_START info: Task# 00000 on Host# 00000
  finishTime = 000000030.00
  exeTime = 000000030.00

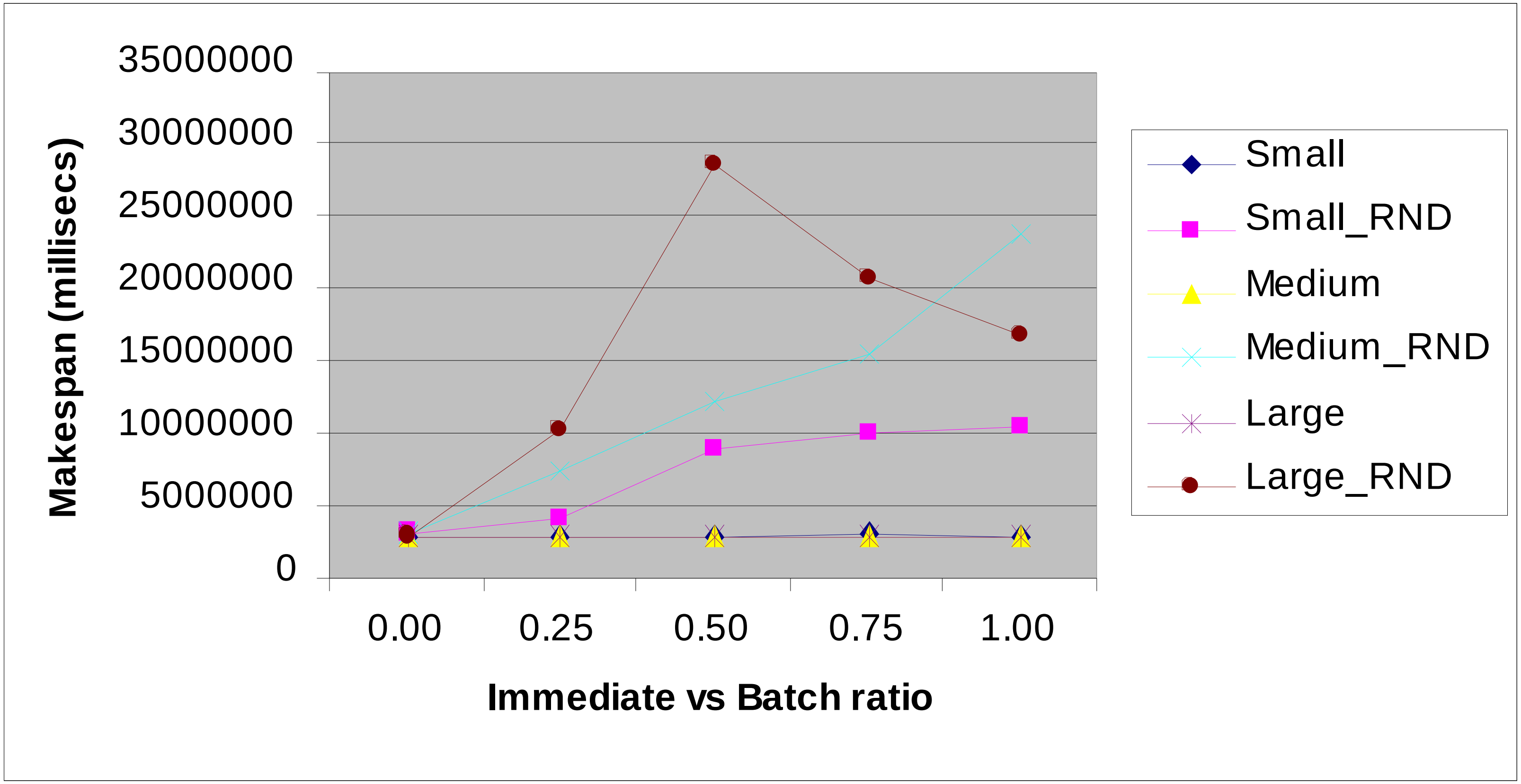
Simulator + Hyperheuristic



Performance Evaluation of Hyper-heuristic using the simulator: static *vs* dynamic

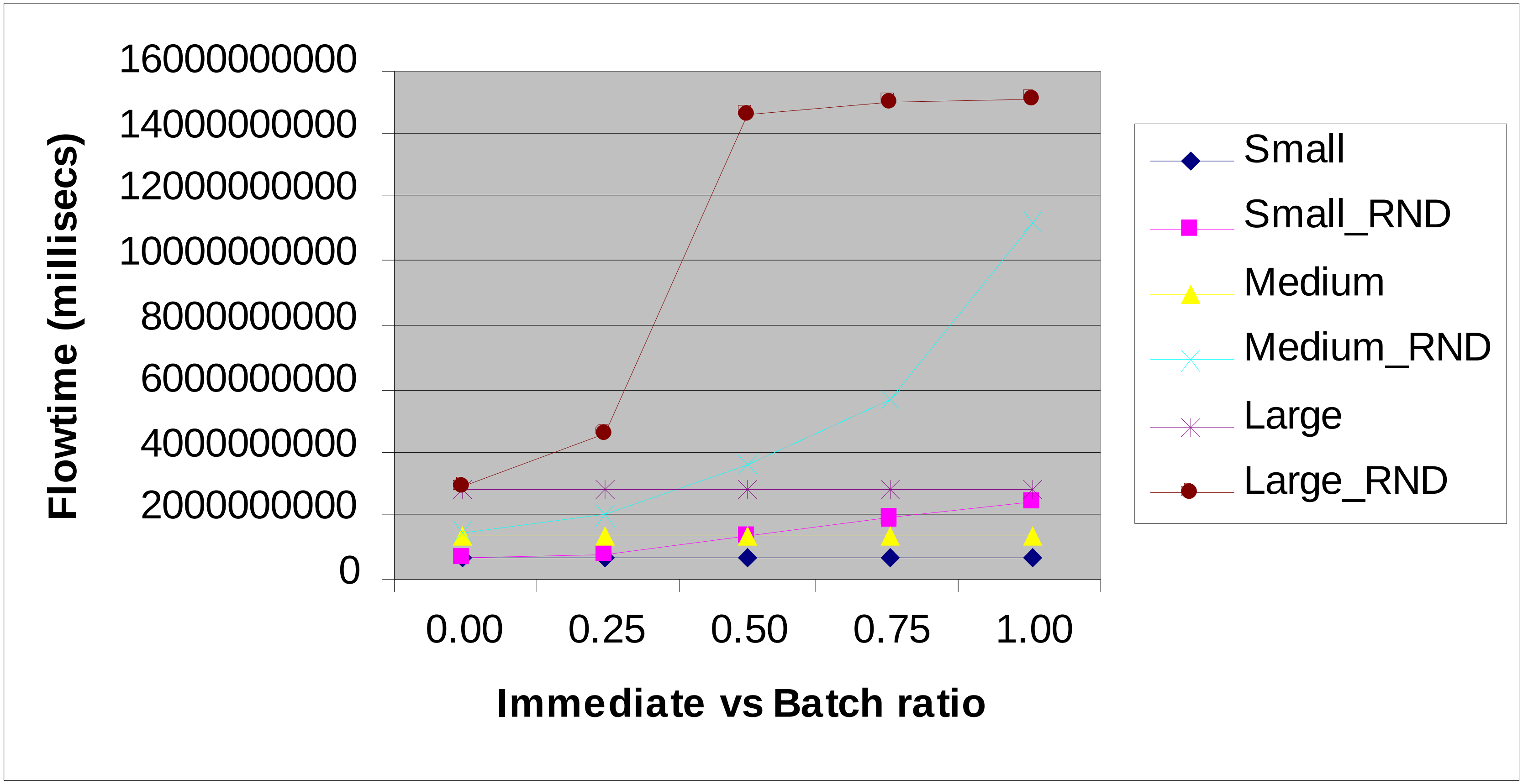
- Two environment types for tests:
  - **Static**: generate concrete instances (a priori fixed configuration)
  - **Dynamic**: the usual use of the simulator intended for a real environment
- Using the Simulator for generating 3 Grid types: Small, Medium and Large size
- Tests for the 2 measures: Makespan and Flowtime
- We compare the hyper-heuristic *versus* an hyper-heuristic with fully random decisions
- Percentage ratio of tasks immediate/batch is modified too: 0%, 25%, 50%, 75% and 100%.

Static – Makespan
(small, medium and large size instances)

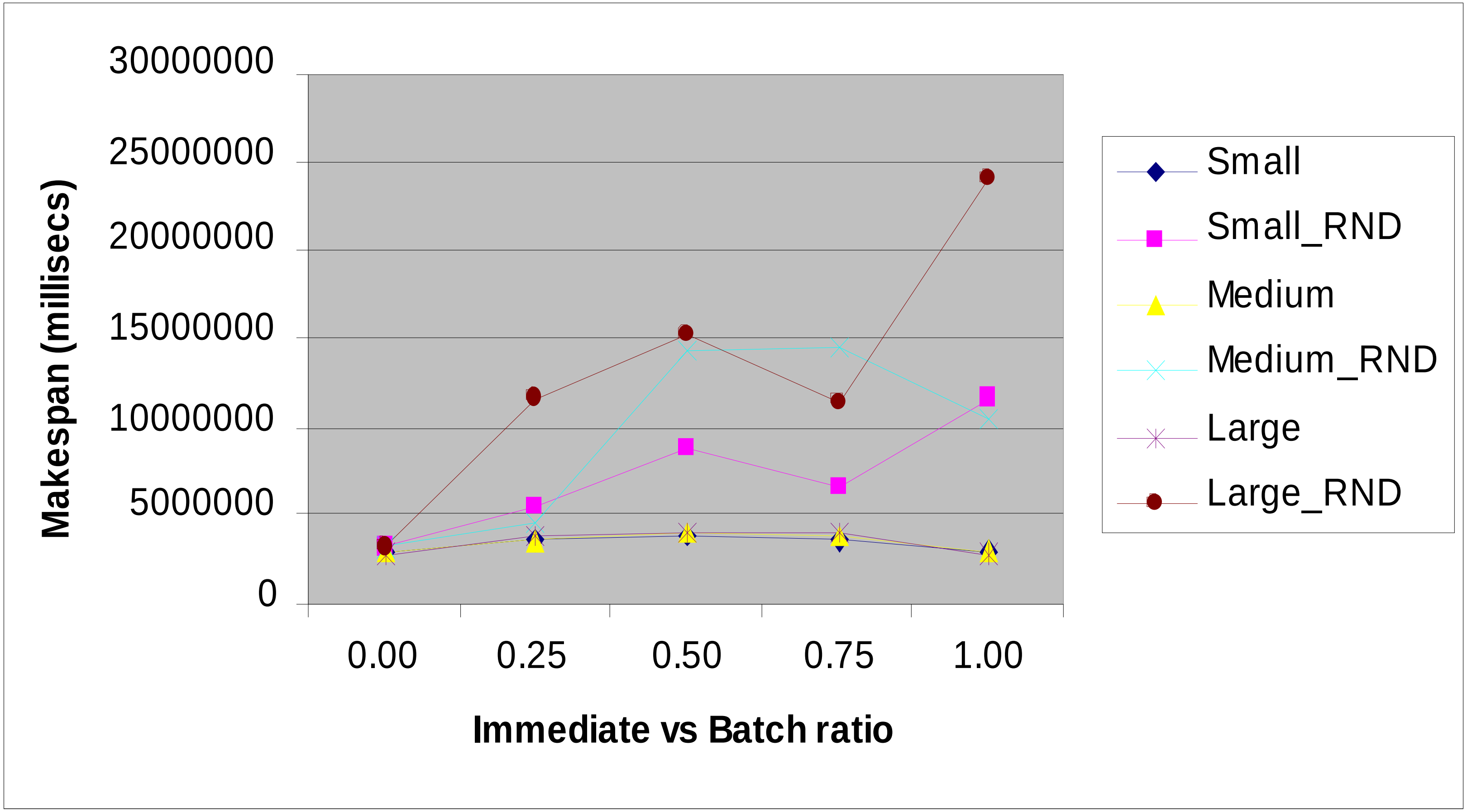


Static – Flowtime

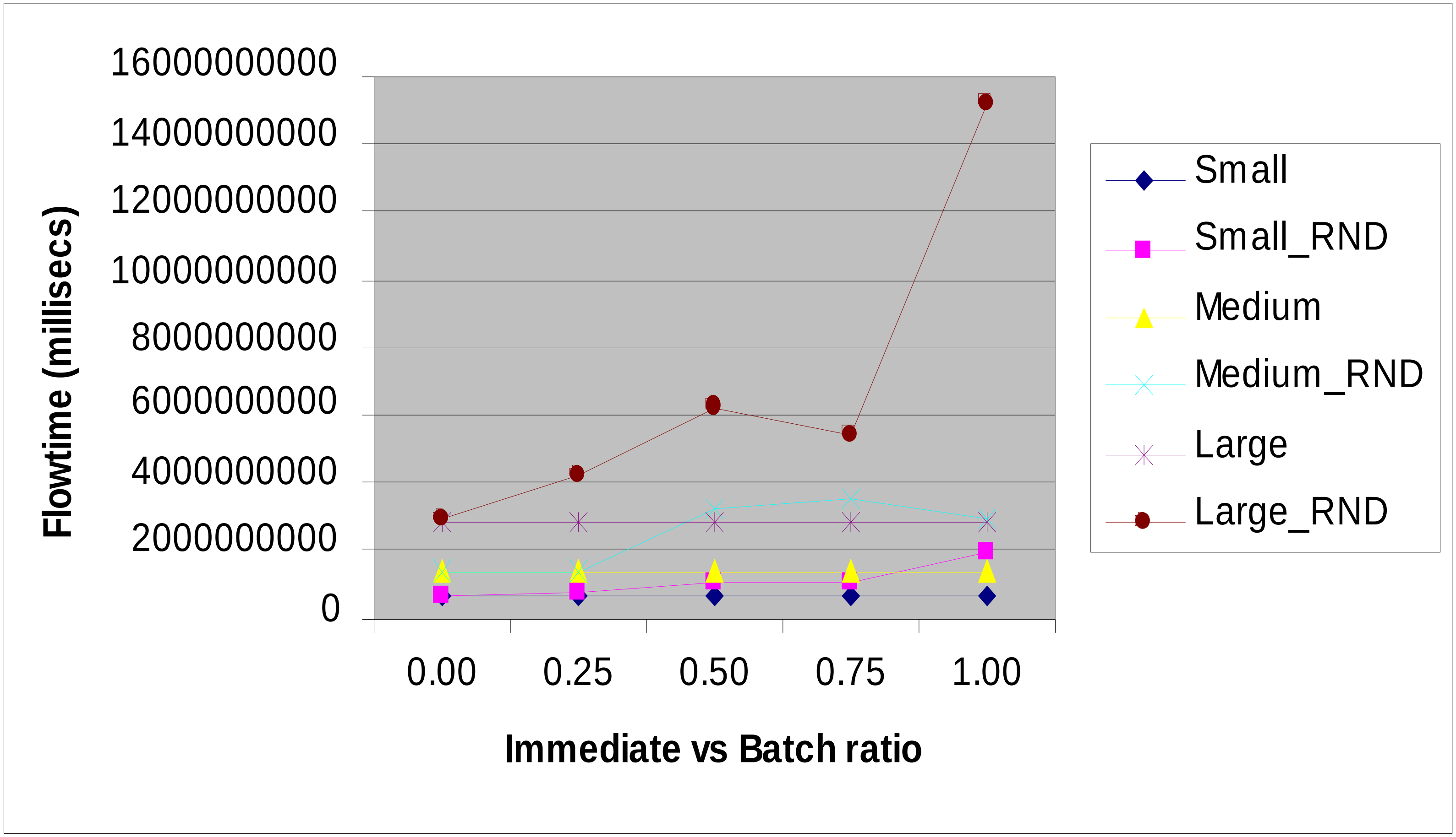
(small, medium and large size instances)



Dynamic - Makespan



Dynamic - Flowtime



Conclusions

- As expected, the schedules done by the HH using guided decisions are better than decisions without any knowledge.

- For Makespan, we have seen that the results increase when the ratio of immediate/batch is 0.5, this indicates that both types of ad-hocs *damage* each other's strategy

- For Flowtime, when the ratio of immediate/batch is favorable to batch, better results are produced.

Future Work

- Add transmission time in our simulations
- Make the hyperheursitic more “intelligent” in decision-taking
- Evaluate the HH in a real grid:
  - – Develop an interface to use it in a real grid
  - – Extract the state of the net (grid characteristics, job characteristics etc.)